Таврический Вестник Информатики и Математики

№ 1 (20) ' 2012

МЕЖДУНАРОДНОЕ НАУЧНО-ТЕОРЕТИЧЕСКОЕ ИЗДАНИЕ
КРЫМСКИЙ НАУЧНЫЙ ЦЕНТР НАЦИОНАЛЬНОЙ АКАДЕМИИ НАУК
И МИНИСТЕРСТВА ОБРАЗОВАНИЯ И НАУКИ, МОЛОДЕЖИ И СПОРТА УКРАИНЫ
ТАВРИЧЕСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО

ОСНОВАН В 2002 ГОДУ

Свідоцтво про державну реєстрацію друкованого засобу масової інформації

КВ №7826 від 04.09.2003

Згідно до постанови ВАК України від 26.05.2010 р. № 1-05/4 журнал "Таврійський вісник информатики та математики" внесено до переліку фахових видань України, в яких можуть публікуватися результати дисертаційних робіт на здобуття наукових ступенів доктора і кандидата фізико-математичних наук (01.01 – математика, 01.05 – інформатика і кібернетика).

КРЫМСКИЙ НАУЧНЫЙ ЦЕНТР НАЦИОНАЛЬНОЙ АКАДЕМИИ НАУК
И МИНИСТЕРСТВА ОБРАЗОВАНИЯ И НАУКИ, МОЛОДЕЖИ И СПОРТА УКРАИНЫ
ТАВРИЧЕСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ им. В.И. ВЕРНАДСКОГО

АДРЕС РЕДАКЦИИ:

Крымский научный центр Национальной Академии наук
и Министерства образования и науки, молодежи и спорта Украины
пр-т Вернадского, 2, г. Симферополь, Крым, 95007, Украина

ДЛЯ ПЕРЕПИСКИ:

Факультет математики и информатики ТНУ
пр-т Вернадского, 4, г. Симферополь, Крым, 95007, Украина

Тел. гл. редактора:	(0652) 63-75-42
Тел. редакции:	(0652) 602-466
e-mail (гл. редактор):	donskoy@tnu.crimea.ua
e-mail (для переписки):	article@tvim.info
сайт журнала:	www.tvim.info

Журнал публикует оригинальные и обзорные статьи по вопросам теоретической и прикладной информатики и математики

Ведущие тематические разделы:

Функциональный анализ и его приложения

Интегральные, дифференциальные уравнения и динамические системы

Нелинейный анализ и его применение

Спектральные и эволюционные задачи

Математические проблемы гидродинамики

Дискретная оптимизация

Математическая логика, теория алгоритмов и теория сложности вычислений

Математические модели и методы прогнозирования

Машинное обучение и извлечение закономерностей

Дедуктивные системы и базы знаний

Знаниеориентированные и гибридные математические модели принятия решений

Синтез моделей принятия решений при неполной начальной информации

Вычислительная математика

Математическая теория, алгоритмы и системы распознавания образов

Печатается по решению научно технического Совета КНЦ НАН и Министерства образования и науки, молодежи и спорта Украины
Протокол № 3 от 5 июня 2012 г.

СОДЕРЖАНИЕ

ИСПОЛНИЛОСЬ 10 ЛЕТ СО ДНЯ ОСНОВАНИЯ ЖУРНАЛА ТВИМ

Журнал Таврический вестник информатики и математики (ТВИМ) был создан в 2002 году. Учредители ТВИМ — Крымский научный центр НАНУ и Таврический национальный университет им. В. И. Вернадского. Целью создания журнала была поддержка дальнейшего развития научно-педагогических коллективов Крымского региона, работающих в области математики и информатики. Особенное значение имела задача — получить научное издание, в котором работы, посвященные компьютерной математике, математическим моделям вычислений и математической информатике и кибернетике в целом, нашли бы почетное место, не менее значительное, чем работы по прикладной математике-механике, доминировавшие в отечественной научной математической литературе XX века.

Огромное и решающее значение для открытия ТВИМа имела поддержка академиков НАНУ И. В. Сергиенко и Ю. И. Журавлева (действительного члена и НАН Украины, и РАН). ТВИМ поддержали и другие известные ученые: чл.-корр. РАН К. В. Рудаков, чл.-корр. НАНУ Ю. С. Самойленко, чл.-корр. НАНУ А. А. Чикрий, профессор А. Г. Наконечный. Несмотря на то, что первые выпуски ТВИМ еще не были ВАКовскими, в ТВИМ направили свои статьи профессора Е. П. Белан, Н. Д. Копачевский, М. А. Муратов, А. Г. Наконечный и другие.

Большую работу по редактированию, компьютерной верстке и выпуску журнала ТВИМ выполняли и выполняют доценты А. С. Анафиев, В. Ф. Блыщик, М. Г. Козлова, В. А. Лукьяненко, сотрудник КНЦ НАНУ Г. К. Мамедов, сотрудник кафедры информатики ТНУ Л. В. Царёва.

Неоценима спонсорская поддержка журнала математиком-программистом, руководителем проекта *Artisteer*, О. В. Дудко, кандидатом физико-математических наук, директором фирмы Гарант, А. М. Петровым.

Выражаю огромную признательность всем упомянутым коллегам, сердечно желаю вам новых творческих достижений, а нашему юному ТВИМУ — счастливого плавания в океане математических наук с двойным попутным ветром — нетерпеливым дыханием информатики и надежным плодотворным вездесущим веянием математики.

Главный редактор ТВИМ, профессор В. Донской

ИСПОЛНИЛОСЬ 100 ЛЕТ СО ДНЯ РОЖДЕНИЯ ВЫДАЮЩЕГОСЯ МАТЕМАТИКА, ОДНОГО ИЗ ОСНОВОПОЛОЖНИКОВ ИНФОРМАТИКИ АЛАНА ТЬЮРИНГА

Alan Mathison Turing (23 июня 1912 – 7 июня 1954) — английский математик, логик, криптограф, оказавший большое влияние на развитие информатики. Предложенная им в 1936 г. абстрактная вычислительная «Машина Тьюринга» позволила формализовать понятие алгоритма и в настоящее время широко используется во множестве теоретических и практических исследований.

Тьюринг был не просто одарённым человеком. Он угадывал самое главное в будущем развитии компьютерной математики.

Как и все дети-вундеркинды, Алан был своеобразным ребенком. Дирекция школы, где учился Тьюринг, долго терпела его необычное поведение, но однажды все-таки направила матери Алана записку следующего содержания: «Ваш сын, видимо, хочет быть только научным специалистом. Может быть, математиком — такие ученики, как он, рождаются раз в 200 лет. Но что он вообще забыл в Public School?»

Осознав свое предназначение, Тьюринг поступил в Королевский Кембриджский колледж, где занимался изучением квантовой физики и математики.

Под воздействием идей Гёделя Тьюринг начал разрабатывать алгоритмический метод для изучения разрешимости исчислений (1936). Это привело его к созданию универсальной вычислительной модели — машины Тьюринга. Уместно заметить, что машина Тьюринга содержала все необходимые элементы для построения универсального программного вычислителя. Оставалось лишь сделать практическое усилие. И появилась машина Z1 — вычислительное устройство, созданное в 1938 году (но уже позже, через два года) немецким инженером Конрадом Цузе. Это устройство было первой программируемой вычислительной машиной с вводом данных с помощью клавиатуры, которая работала в десятичной системе счисления в формате чисел с плавающей запятой.

Ощущая необходимость дальнейшего углубления своих знаний в области математики, Тьюринг продолжил учебу в США, в Принстонском университете, где под руководством знаменитого американского математика Алонзо Черча в 1938 году получил докторскую степень. Его диссертация играла большую роль в развитии теории алгоритмов и алгоритмической разрешимости.

Во время II мировой войны Тьюринг вёл секретную работу в Британском криптоаналитическом бюро. Используя научный алгоритмический подход, основанный на статистическом анализе данных, он взломал криптографическую икону фашистской Германии — трехдисковый шифратор «Энигма». Для этих целей Тьюринг сконструировал собственную узкоспециализированную вычислительную машину «Бомба» (1943).

После окончания Второй мировой войны Тьюринг предложил амбициозный проект автоматической вычислительной машины (Automatic Computation Engine, 1946), который, к сожалению, не был реализован. Затем (1948) он участвовал в создании компьютера с самой большой по тому времени памятью — Манчестерской автоматической цифровой машины «Мадам» (Manchester Automatic Digital Machine). Тьюринг написал для нее несколько программ, пользуясь буквенно-цифровым кодом.

В 1950 году Алан Тьюринг опубликовал свою знаменитую статью под названием «Может ли машина мыслить?» (Can the machine think?), в которой сформулировал знаменитый тест существования искусственного интеллекта и предсказал время (2000 год) создания интеллектуальных программ.

Как и многие гениальные люди, Тьюринг не был похож на большинство своих современников; он был своеобразным, эксцентричным человеком. Многие коллеги завидовали его таланту. Многим не нравилось что Алан — не такой как все, и Тьюринга изводили.

8 июня 1954 года Алан Мэтисон Тьюринг был найден мертвым в своем доме — отравился цианидом. Надкушенное яблоко, начиненное этой отравой, лежало рядом на ночном столике. До сих пор точно неизвестно, было ли это самоубийством или Тьюринга погубили завистники. Это надкушенное яблоко стало эмблемой одного из талантливейших конструкторов компьютеров XX века.

УДК 519.7

ПОДХОД К РЕШЕНИЮ ЗАДАЧ ОПТИМИЗАЦИИ С ПРЕЦЕДЕНТНОЙ НАЧАЛЬНОЙ ИНФОРМАЦИЕЙ

© А. С. Анафиев

Таврический национальный университет им. В.И. Вернадского
факультет математики и информатики
пр-т Вернадского, 4, г. Симферополь, 95007, Украина
e-mail: *anafiyev@gmail.com*

Abstract. The formulation of the optimization problem with precedent initial information is proposed. The main problems and tasks of the constructing reliable schemes for solving such optimization problems are highlighted. The approach for solving such problems based on the loss function is described. The example based on the metric classifiers is considered. The new class of the collective learning by precedent is introduced.

Введение

Рассмотрим задачу оптимизации в общем виде

$$\operatorname{extr} f(x) \,/\, x \in \Omega \subseteq X, \tag{1}$$

где $f : X \to Y$ — целевая функция, оптимальный (близкий к оптимальному) аргумент x^* которой требуется отыскать в ходе решения задачи, X — множество объектов, Y — множество значений (ответов) и Ω — множество (область) допустимых решений (ОДР), из которого выбирается оптимальное решение x^*.

Если при этом целевая функция и/или область допустимых решений (система ограничений) полностью не заданы, то мы имеем дело с задачей оптимизации с неполными данными (слабоопределенную задачу оптимизации). Неполная информация об элементах оптимизационной задачи может быть задана различными способами. Далее будут рассматриваться задачи, в которых информация о целевой функции и/или ограничениях представлена в виде прецедентов. Более подробно такие задачи рассмотрены в [1].

Сформулируем слабоопределенную задачу оптимизации с прецедентной начальной информацией в общем виде.

Пусть X — множество объектов, Y — множество значений целевой функции, W — множество допустимости. Рассмотрим задачу (1), информация о которой задана в виде набора прецедентов $X^\ell = \left\{(x_i, y_i, w_i)_{i=1}^{\ell}\right\}$, где $x_i \in X$, $y_i \in Y$ (если при этом значение целевой функции неизвестно, то ставится прочерк «$-$»); $w_i \in W$ — определяет степень принадлежности объекта x_i области допустимых объектов. Например, если $W = \{0, 1\}$, то w_i определяет принадлежность объекта x_i множеству допустимых объектов: 1 — принадлежит и 0 — не принадлежит; если $W = [0, 1]$, то w_i можно охарактеризовать как вероятность принадлежности объекта x_i множеству Ω.

Необходимо построить алгоритм, который, в некотором смысле, наилучшим образом[1] определяет множество оптимальных объектов (оптимальный объект) и оптимальное значение целевой функции задачи (1) или сводит задачу к известной задаче оптимизации с полностью определенными данными, допускающую эффективное решение.

Одним из подходов к решению задачи оптимизации с прецедентной начальной информацией, конечно же, является разделение исходной задачи на две подзадачи обучения по прецедентам: восстановление целевой функции (обычно, задача регрессии) и восстановление области допустимых решений (задача классификации), решение которых приводит к полностью определенной оптимизационной задаче, решив которую, можно получить окончательное решение.

Однако, при таком подходе теряется важная информация: например, при восстановлении области допустимых решений определяющую роль может играть информация о значениях целевой функции на объектах обучения, а при восстановлении целевой функции важным может оказаться знание того на сколько далеко от границы класса расположен объект, значение в котором определяется. Кроме этого, после восстановления целевой функции и области допустимых решений может получится задача оптимизации, которая не допускает эффективного решения, что является недопустимым при решении реальных практических задач.

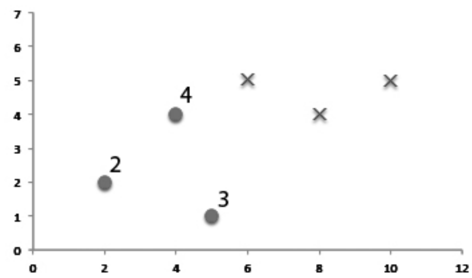
Таким образом, актуальной становится проблема построения схем решения оптимизационных задач с прецедентной начальной информацией, в которых процессы восстановления целевой функции и области допустимых решений выполняются ***совместно*** *друг с другом, «обмениваясь» между собой необходимой информацией. При построении таких схем важно учитывать, что решается именно задача оптимизации, а не две различные задачи обучения.*

Более того, мы получаем новый класс задач — класс *задач совместного обучения по прецедентам.* В нашем случае имеются две совместно решаемые задачи обучения по прецедентам: восстановление целевой функции и восстановления области допустимых решений, решение которых приводит к решению оптимизационной задачи. Следствием этого является необходимость изучения и разработки нового класса алгоритмов обучения, которые позволяют решать *совместно* целое *множество задач обучения* для достижения некоторого общего оптимального решения.

[1]Понятие «наилучшим образом» можно формализовать введением некоторого функционала качества.

Пример 1. Имеется следующая задача оптимизации (в данном случае, максимизации) по прецедентам: $X = \mathbb{R}^2$, $Y = \mathbb{R}$, $W = \{0, 1\}$. Обучающая выборка X^ℓ задана в виде обучающей таблицы:

x_1	x_2	y	w
2	2	2	1
5	1	3	1
4	4	4	1
6	5	-	0
8	4	-	0
10	5	-	0



Кружочками обозначены допустимые объекты, крестиками — недопустимые. Кроме этого объекты помечены значениями функции, если они известны.

Необходимо восстановить область допустимых решений и найти максимальное значение неизвестной целевой функции.

Как видно из рисунка, чем ближе мы приближаемся к воображаемой границе ОДР, тем выше значение функции на объекте. Очевидно, что данная информация, как уже отмечалось выше, не может не учитываться при решении данной задачи. И наоборот, при восстановлении целевой функции важным является расположение объектов из разных классов («ОДР» и «не ОДР») множества объектов X.

1. Функция потерь. Функционал качества

Аналогично задачам обучения по прецедентам можно определить функцию потерь и функционал качества и для задач оптимизации с прецедентной начальной информацией.

Функция потерь — это неотрицательная функция $\mathscr{L}(a, x)$, характеризующая величину ошибки алгоритма a на объекте x [2].

Функционал качества алгоритма a на выборке X^ℓ можно определить как суммарную потерю на всех объектах обучения:

$$Q(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathscr{L}(a, x_i). \tag{2}$$

Обозначим через $a_f(x) \in Y$ ответ алгоритма a о значении целевой функции на объекте x, а через $a_\Omega(x) \in W$ — ответ алгоритма a о принадлежности объекта области допустимых решений. Тогда в качестве функции потерь, например, можно рассмотреть функцию $\mathscr{L}(a, x) = \alpha[a_\Omega(x_i) \neq y_i] + \beta\big(a_f(x_i) - y_i\big)^2$, где α, β — некоторые параметры алгоритма. Данная функция потерь совместно учитывает и правильность определения принадлежности к ОДР и качество восстановления целевой функции.

Одним из способов решения задач оптимизации с прецедентной начальной информацией является модификация функции потерь в сторону взаимного учета потерь при восстановлении целевой функции и области допустимых решений и последующим использованием алгоритмов обучения для решения задачи оптимизации по прецедентам.

Один из подобных методов рассмотрен в работе [3], в которой исходная задача оптимизации сводится, с помощью применения модифицированной функции потерь L_ε, к задаче регрессии, для решения которой предлагается использовать метод *Support Vector Regression* — SVR [4].

Используя огромный набор методов классификации и регрессии, можно получать большое количество подходов к решению задач оптимизации с неполными данными. Конечно же, такой подход порождает множество вопросов: «*какой набор методов машинного обучения и как необходимо использовать при решении той или иной конкретной практической задачи?*»; «*как вычислять адекватность полученных моделей и, следовательно, надежность полученных решений?*»; «*как правильно синтезировать различные алгоритмы обучения для построения схем решения задач оптимизации с прецедентной начальной информацией?*» и т.д.

2. Применение метрических алгоритмов классификации для решения задачи оптимизации с прецедентной начальной информацией

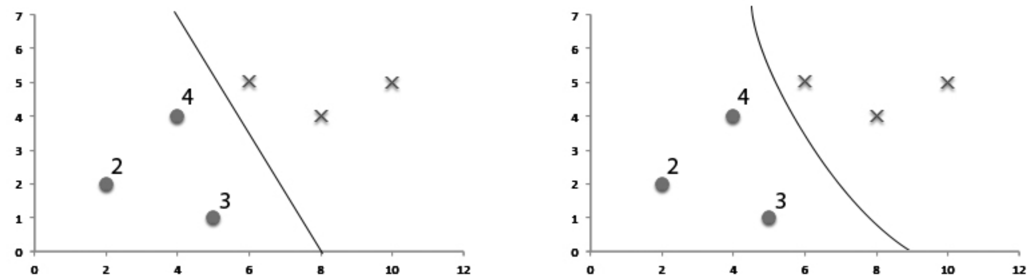
Как известно, метрические алгоритмы классификации основаны на гипотезе компактности, предполагающей, что схожие (по некоторой метрике) объекты расположены внутри одного класса, а объекты далекие по метрике в разных.

Обобщенный метрический классификатор можно записать в виде [2]:

$$a(u; X^\ell) = \arg\max_{y \in Y} \Gamma_y(u, X^\ell); \qquad \Gamma_y(u, X^\ell) = \sum_{i=1}^{\ell} [y_u^{(i)} = y]\, w(i, u) \tag{3}$$

где u — классифицируемый объект, $y_u^{(i)}$ — метка класса i-го соседа объекта u, $\Gamma_y(u, X^\ell)$ — суммарный вес ближайших к u обучающих объектов из X^ℓ, $w(i, u)$ — весовая функция, оценивающая степень важности i-го соседа для классификации объекта u.

Рис. 1. Метод потенциальных функций для решения задач оптимизации



а) $h_i = 5$, $i = 1..6$.

б) Переменная ширина окна равная величине потенциала, $h_i = y_i$.

Рассматривая различные весовые функции, можно получать различные метрические алгоритмы классификации, а следовательно и различные методы решения задач оптимизации с прецедентной начальной информацией.

- Для метода ближайших соседей весовая функция имеет вид $w(i,u) = [i \leqslant k]$. Модернизируем ее для задачи оптимизации. Для этого будем учитывать не только номер соседа, но и значение целевой функции в нем: $w(i,u) = [i \leqslant k]K\big(y_u^{(i)}\big)$, где $K(z)$ — некоторая функция, характеризующая степень важности i-го соседа в зависимости от значения целевой функции.
- Для метода парзеновского окна и метода потенциальных функций достаточно переопределить функцию расстояния между обучающими объектами, так, чтобы она учитывала не только координаты объектов, но и значения целевой функции в них. Например, для метода потенциальных функций

$$a(u, X^\ell) = \arg\max_{y \in Y} \sum_{i=1}^{\ell} \gamma_i \, K\left(\frac{\rho(x_i, u)}{h_i}\right), \quad \gamma_i \geqslant 0,\ h_i \geqslant 0,$$

  в качестве значений потенциалов γ_i можно рассматривать значения (или некоторую функцию от них) целевой функции на объекте, а если взять $W = \{-1, +1\}$, то принадлежность области допустимых решений можно рассматривать как знак заряда (см. рисунок 1).

Заключение

Сформулирована задача оптимизации с прецедентной начальной информацией в общем виде. Выделено направление применения широкого класса методов машинного обучения для решения задач оптимизации по прецедентам. Очерчены основные проблемы и задачи получения схем построения адекватных, согласованных с начальной информацией, моделей задач оптимизации с прецедентной начальной информацией.

Выделен новый класс задач обучения — класс совместного обучения по прецедентам.

Список литературы

1. Анафиев А. С. Оптимизационные модели с прецедентной начальной информацией / А. С. Анафиев, В. Ф. Блыщик // Таврический вестник информатики и математики. — Симферополь: КНЦ НАНУ, 2011. — №2. — С. 51–57.
2. Воронцов К. В. Математические методы обучения по прецедентам. Курс лекций по машинному обучению. [Электронный ресурс] / К. В. Воронцов. — 2011. — 141 с. Режим доступа к курсу: http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf
3. Таратынова Н. Ю. Построение оптимизационной модели по прецедентной начальной информации как задача нелинейной регрессии / Н. Ю. Таратынова // Искусственный интеллект. — Донецк: Институт проблем искусственного интеллекта, 2006. — №2. — С. 238–241.
4. Smola A. J. Tutorial on Support Vector Regression / Smola A. J., Scholkopf B. A. // Statistics and Computing, 2004. — Vol. 14. — P. 199–222.

Статья поступила в редакцию 02.06.2012

УДК 519.95

СИНТЕЗ СОГЛАСОВАННЫХ ЛИНЕЙНЫХ ОПТИМИЗАЦИОННЫХ МОДЕЛЕЙ ПО ПРЕЦЕДЕНТНОЙ ИНФОРМАЦИИ: ПОДХОД НА ОСНОВЕ КОЛМОГОРОВСКОЙ СЛОЖНОСТИ

© В. И. Донской

Таврический национальный университет им. В. И. Вернадского
факультет математики и информатики
пр-т Вернадского, 4, г. Симферополь, 95007, Украина
e-mail: *donskoy@tnu.crimea.ua*

Abstract. In the paper, approach is expounded to the analysis of the linear optimization models built on precedent initial learning information. In supposition, that all numerical parameters are rational and limited by the bit net of computer, estimations of Kolmogorov's complexity and nonrandomness of extraction of model as empirical regularity are got from the sample.

Введение

Линейные оптимизационные модели во многих случаях хорошо описывают процессы экономического планирования производства, распределения ресурсов, размещения объектов. Активно развивающееся в математике направление, связанное с алгоритмической сводимостью, позволяет увидеть, как часто возникающие на практике задачи сводятся к задачам линейной оптимизации. В то же время на этапе внедрения линейных моделей в производственное управление выявились две основные проблемы, связанные, главным образом, с ростом размерности. Первая — сложность сбора данных для фиксации модели, вторая — неадекватность модели реальному процессу или объекту вследствие неполноты и/или неадекватности данных.

Основная идея, на которой основан развиваемый в статье подход, состоит в том, что *линейная модель должна быть согласована с реальной, фактической информацией об объекте моделирования. Такой фактической информацией являются прецеденты — наблюдения, зафиксированные в процессе функционирования объекта.*

Многие задачи линейного программирования с вещественными неотрицательными переменными имеют вид

$$\left\{ \begin{array}{l} \max f(X) \\ AX^T \leqslant B \end{array} \right. , \qquad (1)$$

где $X = (x_1, ..., x_n)$, $A = \|a_{i,j}\|_{m\times n}$, $B = (b_1, ..., b_m)^T$, $x_i \geqslant 0$, $a_{i,j} \in \mathbb{R}$, $b_i \in \mathbb{R}$, $f : \mathbb{R}^n \to \mathbb{R}$. Будем называть $\Omega = \{X : AX^T \leqslant B\}$ множеством допустимых решений задачи (1).

Задача, которая решается в данной статье, состоит в следующем. Предполагается, что для решения оптимизационной задачи вида (1) в некоторой проблемной области представлена только начальная информация в виде конечного набора точек из $\mathbb{R}^n$ вместе со значением неизвестной целевой функции в каждой точке (если это значение измеряно), а также логическим значением 1, если точка удовлетворяет всем требуемым ограничениям, которые определяются проблемной областью, или 0 — если эта точка ограничениям не удовлетворяет. Требуется: а) обосновать возможность применения модели линейного программирования (1) и б) — если применение линейной модели возможно, — синтезировать модель по предоставленной прецедентной начальной информации, т.е. построить матрицу $\hat{A}$ и вектор $\hat{B}$, определив при этом число линейных ограничений-неравенств $\hat{m}$, а также найти целевую функцию $\hat{f}$. Затем синтезированная модель

$$\begin{cases} \max \hat{f}(X) \\ \hat{A}X^T \leqslant \hat{B} \end{cases} \tag{2}$$

должна быть некоторым образом обоснована, т.е. требуется с) дать оценку адекватности синтезированной модели (2).

Формально начальная информация имеет вид набора троек $D = \{(X_k, y_k, \omega_k),\ k = 1, ..., l\}$. Будем называть эту информацию *обучающей выборкой или данными* D, а число l заданных точек — *длиной выборки.* $X_k \in \mathbb{R}^n$, $y_k \in \mathbb{R}$ (y_k может быть не определено для части элементов выборки; для пропущенных значений будет использованться обозначение Δ); $\omega_k = 1$, если выборочная точка удовлетворяет ограничениям, и $\omega_k = 0$ — если она ограничениям не удовлетворяет. Обозначим $D_\oplus = \{X_k : \omega_k = 1\}$; $|D_\oplus| = l_\oplus$ — число допустимых решений, представленных в начальной информации; $D_\ominus = \{X_k : \omega_k = 0\}$; $|D_\ominus| = l_\ominus$ — число точек, не являющихся допустимыми решениями.

Множество допустимых решений модели (2) будет обозначаться $\hat{\Omega}_D = \{X : \hat{A}X^T \leqslant \hat{B}\}$.

Синтезированная модель (2) является аппроксимацией модели (1), которая полагается существующей и представленной неполной начальной информацией D.

Впервые постановка подобных задач была сделана В. Д. Мазуровым в работах [10] и [11]. Для их решения использовались итерационные методы, основанные на фейеровских отображениях [12].

Синтетический подход к решению линейных оптимизационных задач по частичной (прецедентной) информации был предложен и разрабатывался в работах [8, 6, 18, 15, 2] и был, в основном, основан на эвристиках. Исследования последних лет в области теории колмогоровской сложности и её применения в машинном обучении [1, 7, 19] определили интерес к дальнейшему исследованию проблем синтеза оптимизационных моделей по прецедентной информации.

1. Синтез линейной модели как машинное обучение

Применяя методы машинного обучения по прецедентной информации D, возможно использовать линейные итерационные модели, основанные на процедуре Розенблатта-Новикова, которые позволяют построить набор из $\hat{m}$ линейных гиперплоскостей, определяющих область допустимых решений $\hat{\Omega}_D$ в виде $\hat{L}_q(X) \leqslant \hat{b}_q$, $q = 1, ..., \hat{m}$, где $\hat{L}_q$ — линейные функции, а $\hat{b}_q$, $q = 1, ..., \hat{m}$, — пороговые вещественные значения. Коэффициенты этих функций и числа $\hat{b}_q$, $q = 1, ..., \hat{m}$, будут определять параметры $\hat{A}$ и $\hat{B}$ синтезированной модели (2). А число $\hat{m}$ полученных неравенств (вместе с заданным n) определит размерность матрицы $\hat{A}$ и вектора $\hat{B}$. Синтез целевой функции может быть осуществлён, например, по методу наименьших квадратов.

Такой подход был предложен в [8]. Интуитивно заложенный в алгоритм синтеза ограничений, описанного в этой работе, принцип «бритвы Оккама» (Occam's Razor), как будет показано ниже, может быть достаточно строго обоснован на базе понятия колмогоровской алгоритмической сложности.

Нужно подчеркнуть, что излагаемые в настоящей статье методы и алгоритмы рассчитаны на компьютерную реализацию. Это еще не раз будет подчёркиваться по ходу изложения материала.

2. Согласованная модель-гипотеза

Модель (1) с фиксированными значениями (параметрами) n, m, f, A, B является индивидуальным представителем класса $\mathscr{L}$ моделей линейного программирования. Синтезированная модель (2), как правило, не совпадает с истинной неизвестной моделью (1) и является её аппроксимацией.

Определение 1. Синтезированная модель (2) называется *согласованной (согласованной гипотезой)*, если при подстановке точек из обучающей выборки D в целевую функцию и линейные неравенства модели (2) выполняются условия: $\hat{f}(X_k) \equiv y_k$ и $[\hat{A}X_k \leqslant \hat{B}] \leftrightarrow \omega_k$ для всех $k = 1, .., l$, соответствующих полностью заданным (определенным) тройкам значений из D.

Согласованная линейная модель будет далее обозначаться $\mathscr{M}_D$.

Нужно заметить, что в широком семействе моделей $\mathscr{L}$ существует более узкий подкласс согласованных моделей $\mathscr{L}_D$. Выбор согласованной модели из $\mathscr{L}_D$ — сложная, некорректная по Адамару задача, требующая серьёзного обоснования.

Учитывая, что все излагаемые в статье методы и алгоритмы рассчитаны на компьютерную реализацию, *можно и нужно сузить рассматриваемые семейства согласованных моделей до конечных, ограничив представления чисел рациональными значениями, лежащими в отрезке, который определяется разрядной сеткой применяемого компьютера* .

3. Колмогоровская сложность модели линейной оптимизации

Комогоровская сложность может определяться как в терминах класса машин Тьюринга (MT), так и в терминах класса частично рекурсивных функций P_{pr}, поскольку эти два класса эквивалентны. Поэтому, если не возникает недоразумений, в дальнейшем один и тот же объект U может быть назван $MT\ U$ или функцией $U \in P_{pr}$. Тьюринговская интерпретация предпочтительнее, когда рассуждения проводятся в терминах строк и алфавитных отображений. Частично рекурсивная — когда необходимо использовать свойства алгоритмов как функций при доказательстве теорем.

Если $\mathscr{A} = \{\alpha_1, ..., \alpha_\mu\}$ — конечный алфавит, то $\mathscr{A}^*$ — множество любых строк любой конечной длины над алфавитом $\mathscr{A}$. Будем обозначать $p \in \{0,1\}^*$ — двоичные строки. Обозначаемые другими буквами строки могут состоять из символов иных, отличных от $\{0,1\}^*$ алфавитов. Длина произвольной строки s (число символов в ней) обозначается $|s|$. Символом Λ будет обозначаться пустая строка; $|\Lambda| = 0$.

(Префиксная) колмогоровская сложность произвольной строки s при заданной МТ U определяется как $K_U(s) = min\{\,|p| : U(p) = s\}$, где p — бинарная строка, которую назвают наименьшей длиной программы, вычисляющей s (или описания, кодирующего s). Поскольку универсальная МТ может моделировать любую другую МТ, выбор U в формуле $K_U(s)$ может изменить колмогоровскую сложность не более чем на константу, зависящую только от U [9]. Поэтому U в обозначении $K_U(s)$ часто опускается, и используется обозначение $K(s)$.

Нужно отметить, что существует МТ U_s такая, которая, получая вход в виде пустой строки, генерирует заданную строку s. Чтобы это понять, достаточно указать последовательность команд, определяющую U_s, в которой соответствующая последовательность символов вывода в правых частях идущих строго друг за другом команд, завершающихся командой остановки, в точности совпадает с s. Можно сказать, что в этом случае s «вмонтирована» в тело программы машины U_s. Но тогда получается, что в определении $K_U(s) = min\{|p| : U(p) = s\}$ машина U не может быть любой, поскольку для пустой входной строки Λ машина U_s выдаст s, и тогда $K_{U_s}(s) = 0$ для любой строки s. Будем называть МТ U_s *генератором* строки s, если эта машина для любой входной строки выдаёт строку s, и обозначать $\mathscr{U}_s$ — множество генераторов для всевозможных строк s.

Определение 2. *Точной колмогоровской сложностью произвольной строки s* называется

$$KC(s) = \min_{U \notin \mathscr{U}_s} (\min\{|p| : U(p) = s\}),$$

если равенство $U(p) = s$ выполняется для хотя бы одной МТ, не являющейся генератором, иначе полагается что $KC(s) = \infty$.

Введение определения точной колмогоровской сложности объясняется необходимостью избавиться от константы, связанной с переходом от одной оптимальной машины U к другой. Эта константа фигурирует в классическом определении и усложняет получение оценок неслучайности выбора моделей в задачах машинного обучения, основанных на принципе минимальной длины описания (MDL).

Определение 3. *Колмогоровской сложностью согласованной модели при заданной МТ U* называется

$$KC_U(\mathscr{M}_D) = \min\left\{|p| : U(p) = s_{\hat{C}} * s_{\hat{A}} * s_{\hat{B}}\right\},$$

где $s_{\hat{C}} * s_{\hat{A}} * s_{\hat{B}}$ — строка, являющаяся конкатенацией трёх строк, определяющих такие элементы модели $\mathscr{M}_D$, что

$$(\hat{f}_{\hat{C}}(X_k) \equiv y_k) \wedge ([\hat{A}X_k \leqslant \hat{B}] \leftrightarrow \omega_k), (X_k, y_k, \omega_k) \in D, \quad k = 1, ..., l,$$

а $\hat{f}_{\hat{C}}$ — целевая функция, определяемая набором коэффициентов $\hat{C}$.

Определение 4. *Колмогоровской сложностью согласованной системы линейных ограничений при заданной МТ U* называется

$$KC_U(\hat{\Omega}_D) = \min\left\{|p| : U(p) = s_{\hat{A}} * s_{\hat{B}}\right\},$$

где $s_{\hat{A}} * s_{\hat{B}}$ — строка, являющаяся конкатенацией двух строк, определяющих такие элементы модели $\hat{\Omega}_D$, что

$$([\hat{A}X_k \leqslant \hat{B}] \leftrightarrow \omega_k), (X_k, \Delta, \omega_k) \in D, \quad k = 1, ..., l.$$

Определение 5. *Точной колмогоровской сложностью семейства $\mathfrak{M}_D$* согласованных моделей называется

$$KC(\mathfrak{M}_D) = \min_{U \notin \mathscr{U}_s} \max_{\mathscr{M}_D \in \mathfrak{M}_D} KC_U(\mathscr{M}_D).$$

Важно заметить, что в определении точной колмогоровской сложности семейства моделей берётся «оптимальная» по сжатию самой сложной модели в классе $\mathfrak{M}_D$ машина Тьюринга. Если существует более одной такой МТ (тогда для всех из них значение $KC(\mathfrak{M}_D)$ одинаково), то берется одна любая «оптимальная» по сжатию МТ.

Определение 6. *Точной колмогоровской сложностью семейства* $\tilde{\Omega}_D$ согласованных систем линейных ограничений называется

$$KC(\tilde{\Omega}_D) = \min_{U \notin \mathscr{U}_s} \max_{\Omega_D \in \tilde{\Omega}_D} KC_U(\Omega_D).$$

Точная колмогоровская сложность не является вычислимой функцией. Для использования этого понятия будут применяться оценки сложности сверху. С техникой получения подобных оценок можно ознакомиться в [5].

4. Обоснование согласованных моделей на основе колмогоровской сложности

Описанием числовых векторов и матриц являются последовательности подходящим образом закодированных чисел. Если значительная часть коэффициентов матрицы (векторов) согласованной модели — нулевые, то описание (сложность) модели может оказаться существенно короче.

Лемма 1. *Согласованные модели из семейства $\mathfrak{N}_D$, имеющего точную колмогоровскую сложность d, могут быть извлечены путём применения «оптимальной» по сжатию машиной U_D не более чем 2^d способами.*

Доказательство. Пусть для решения поставленной задачи синтеза применяется произвольное согласованное семейство $\mathfrak{N}_D \subseteq \mathfrak{M}_D$ такое, что $KC(\mathfrak{N}_D) = d$, и зафиксирована некоторая «оптимальная» по сжатию МТ U_D. Применение этой машины к двоичной строке $p : |p| = d$ обеспечивает получение линейной модели $\mathscr{M}_D(p)$, представленной в форме конечной строки-описания $S_p = U_D(p)$, где $S_p = s_{\hat{C}} * s_{\hat{A}} * s_{\hat{B}}$. Поскольку U_D является функцией ($U_D \in P_{pr}$), каждому значению p соответствует единственная строка $S_p = U_D(p)$. Тогда 2^d различных возможных значений строки p длины d исчерпывают все порождаемые строки вида $S_p = s_{\hat{C}} * s_{\hat{A}} * s_{\hat{B}}$ и, возможно, строки, не имеющие смысла как, например, код согласованной модели $S_p = U_D(11...11)$. □

Следствие 1. *Пусть $KC(\mathfrak{N}_D) < h$, где h — верхняя оценка сложности семейства $\mathfrak{N}_D$. Тогда по кодам (программам) длины h можно получить не более 2^h согласованных моделей, и $|\mathfrak{N}_D| \leqslant 2^h$.*

Напомним, что изучаемая задача синтеза будет решаться на вычислительной машине, в которой число бит, используемое для представления одного числа, равно b; именного столько бит будет использоваться для записи одного числа и в обучающих данных D, и в описании кода модели $S_p = s_{\hat{C}} * s_{\hat{A}} * s_{\hat{B}}$. Тогда в семействе $\mathfrak{M}_D$ будет конечное число моделей. И программирование верхних оценок сложности [5] будет связано с конструированием строк, длина которых будет определяться, в числе прочего, константой b.

Обозначим $conv(D_\oplus)$ выпуклую оболочку множества допустимых точек из обучающей выборки.

Лемма 2. *При условии $D_\ominus \cap conv(D_\oplus) = \varnothing$, мощность семейства $|\tilde{\Omega}_D|$ удовлетворяет неравенству*

$$2^{log(\lceil \hat{\rho}/\delta \rceil - 2)l_\ominus} \leqslant |\tilde{\Omega}_D|,$$

где $\hat{\rho}$ — наименьшее по всем точкам обучающего множества $D_\ominus$, лежащим вне множества допустимых решений, евклидово расстояние до выпуклой оболочки точек из $D_\oplus$, которые являются допустимыми решениями, а δ — модуль наименьшего рационального числа в используемом машинном формате.

Доказательство. Пусть $\hat{\rho} = \min\limits_{X_k \in D_\ominus} \rho(X_k, conv(D_\oplus))$ — кратчайшее расстояние по всем точкам множества $D_\ominus$ до выпуклой оболочки точек множества $D_\oplus$. Отрезок прямой, длина которой равна $\hat{\rho}$, соединяющий точку из $D_\ominus$ с точкой оболочки, можно отделить от $conv(D_\oplus)$ не менее чем $\lceil \hat{\rho}/\delta \rceil - 2$ перпендикулярными ему гиперплоскостями. Поэтому каждая точка из $D_\ominus$ линейно отделима не менее чем $\lceil \hat{\rho}/\delta \rceil - 2$ гиперплоскостями, и из них можно выбрать одну любую для отделения каждой точки. Тогда число линейных моделей, отделяющих все точки множества $D_\ominus$, не менее $(\lceil \hat{\rho}/\delta \rceil - 2)^{l_\ominus} = 2^{log(\lceil \hat{\rho}/\delta \rceil - 2)l_\ominus}$. □

Решение задачи поиска модели, «объясняющей» выборочные данные, происходит как машинное обучение, приводящее к извлечению гипотезы — закономерности в данных. Важно оценить неслучайность выбора гипотезы.

В теории статистических решений предполагается существование вероятностных распределений в пространстве выбора гипотез. В данной работе рассматривается нестатистическая постановка задачи: выборочные данные отражают регулярные модели. Поэтому применяется подход к выбору модели в условиях неопределенности, предполагающий возможность равновероятного извлечения любой согласованной модели. Но синтез согласованной модели, как процесс машинного обучения, должен приводить не к случайной гипотезе, а к закономерности.

Для понимания доказательства следующих утверждений нужно обратить внимание на то, что сложность линейной согласованной модели всегда не меньше сложности входящей в неё системы ограничений.

Теорема 1. *Пусть модели согласованного семейства* $\mathfrak{M}_D$ *распределены в нем равновероятно. Тогда вероятность случайного извлечения согласованной модели из класса* $\mathfrak{N}_D \subset \mathfrak{M}_D$*, имеющего оценку точной колмогоровской сложности* $KC(\mathfrak{N}_D) < h$*, не превысит*

$$2^{h-log(\lceil\hat{\rho}/\delta\rceil-2)l_{\ominus}}$$

Доказательство. Напомним, что семейство $\mathfrak{M}_D$ является конечным в силу условия сужения моделей до реализуемых на компьютере с ограничением рационального промежутка представления чисел. Оцениваемая вероятность случайного выбора есть $|\mathfrak{N}_D|/|\mathfrak{M}_D|$. Согласно условию теоремы, $KC(\mathfrak{N}_D) < h$, и $|\mathfrak{N}_D| \leqslant 2^h$ по следствию 1. Согласно лемме 2,

$$|\mathfrak{M}_D| \geqslant 2^{log(\lceil\hat{\rho}/\delta\rceil-2)l_{\ominus}},$$

поскольку модели класса $\mathfrak{M}_D$ включают в себя все модели линейных ограничений семейства $\tilde{\Omega}_D$. Поэтому

$$|\mathfrak{N}_D|/|\mathfrak{M}_D| \leqslant 2^h/|\mathfrak{M}_D| \leqslant 2^h/2^{log(\lceil\ \hat{\rho}/\delta\rceil-2)l_{\ominus}} = 2^{h-log(\lceil\hat{\rho}/\delta\rceil-2)l_{\ominus}}$$

□

Замечание. Полученная в теореме 1 оценка справедлива для расширения компьютерных моделей до бесконечного класса вещественных.

Следствие 2. *Для того, чтобы построенная модель* $\mathscr{M}_D$*, имеющая верхнюю оценку точной колмогоровской сложности* h*, с вероятностью не меньшей* $1-\varepsilon$ *была неслучайной, т. е. была закономерностью, необходимо, чтобы число примеров в выборке удовлетворяло неравенству*

$$l_{\ominus} \geqslant (h + log(1/\varepsilon))/log(\lceil\hat{\rho}/\delta\rceil - 2). \tag{3}$$

Доказательство. Основываясь на колмогоровском представлении о достоверности как неслучайности, из неравенства

$$2^{h-l_{\ominus}log(\lceil\hat{\rho}/\delta\rceil-2)} \leqslant \varepsilon$$

легко вывести эквивалентное неравенство (3). □

Например, $l_{\ominus}(\varepsilon = 0.01,\ \hat{\rho} = 1,\ \delta = 10^{-4}, h = 3000) \approx 230$.

Согласно полученным результатам, необходимая величина объема $l_{\ominus}$ подвыборки $D_{\ominus}$ определяется, главным образом, колмогоровской сложностью модели (оценкой сложности) и кратчайшим расстоянием $\hat{\rho}$, взятым по всем точкам множества $D_{\ominus}$, до выпуклой оболочки точек множества $D_{\oplus}$. Невхождение длины $l_{\oplus}$ подвыборки в явном виде в оценку (3) объясняется тем, что от нее в значительной степени зависит оценка кратчайшего расстояния $\hat{\rho}$, которая в эту оценку входит явно. Рост числа «положительных» примеров $l_{\oplus}$ в обучающей информации влечет всё более равномерное заполнение области допустимых решений и уменьшение $\hat{\rho}$.

VC-размерность Вапника-Червоненкиса [4] $h_{VC}(\mathfrak{M})$ некоторого семейства моделей $\mathfrak{M}$ связана с его колмогоровской сложностью КС($\mathfrak{M}$) соотношением

$$h_{VC}(\mathfrak{M}) \leqslant \text{КС}(\mathfrak{M}) < h_{VC}(\mathfrak{M}) logl,$$

где l — длина обучающей выборки [17]. Это показывает, что нестатистическая оценка качества моделей — закономерностей хорошо согласуется со статистической теорией В. Н. Вапника и А. Я Червоненкиса.

5. Рекомендации к построению согласованных моделей

1. Необходимо стремиться к получению модели с минимальной колмогоровской сложностью. Достигнуть снижение сложности синтезированной модели можно за счет:
   - минимизации числа неравенств в $\hat{\Omega}_D$;
   - минимизации ненулевых членов оценочных матриц и векторов, поскольку pVCD оценка [5] сжатия строки описания модели уменьшается при этом на число нулевых элементов, умноженное на константу b.

   При построении неравенств согласованной линейной оптимизационной модели каждая точка выборочных данных из $D_{\ominus}$ должна линейно отделяться ото всех допустимых точек из $D_{\oplus}$. Поэтому всегда можно синтезировать $l_{\ominus}$ гиперплоскостей. Затем следует произвести переобучение для каждой из них, включив в выборку для каждой плоскости уже все отделённые ею точки. И только затем решать задачу о кратчайшем наборе гиперплоскостей, как задачу минимизации [8].

2. Алгоритмы построения линейных отделителей необходимо усовершенствовать введением выполняемого в ходе синтеза «сброса» в ноль коэффициентов, значение которых ниже некоторого порогового значения. Такая модификация может позволить понизить сложность линейных функций, для синтеза которых представляется полезным применение алгоритмов SVM [14].

3. Необходимо тщательно разработать структуру слова p, являющееся сжатым описанием модели. Уменьшение его длины может быть достигнуто, например, за счет экономного кодирования идущих «подряд» нулей в описании модели. Это позволит уменьшить значение верхней оценки сложности модели h.

4. Если в n-мерном пространстве задано $l_{\oplus}$ точек то при $n \geqslant 6$ число граней выпуклой оболочки имеет порядок $O({l_{\oplus}}^{\lfloor n/2 \rfloor})$ [13]. Следовательно, построение выпуклой оболочки точек множества $D_{\oplus}$ трудоёмко. Указанная в работе [3] процедура $FullConvexhull$ строит описание граней всех размерностей оболочки и нормальных конусов к ним. Подобные процедуры слишком сложны в случае большой размерности. Для оценивания кратчайшего расстояния до выпуклой оболочки её построение в явном виде не требуется.

Заключение

В статье предложен колмогоровский подход к определению сложности линейных согласованных моделей оптимизации. На основе оценивания колмогоровской сложности моделей даны рекомендации к обоснованию методов синтеза таких моделей по прецедентной начальной информации.

Вопросы формирования обучающей прецедентной информации в статье не затронуты. Но растущая большими темпами компьютеризация объектов и большие объемы доступной для хранения информации памяти на сегодняшний день позволяют извлекать из хранилищ данных выборки, имеющие длину свыше десятков тысяч наблюдений.

Направление дальнейшей работы связано с усовершенствованием алгоритмов синтеза линейных согласованных моделей с учетом данных выше рекомендаций.

Список литературы

1. Анафиев А. С. М-Модели алгоритмов. Ёмкость и колмогоровская сложность класса М-полиномов / А. С. Анафиев // Таврический вестник информатики и математики. – 2010. – № 1. – С. 51–57.
2. Блыщик В. Ф. Интеллектуализированная программная система Intman поддержки принятия решений в задачах планирования и управления / В. Ф. Блыщик, В. И. Донской, Г. А. Махина // Искусственный интеллект. — 2002. — С. 406–415.
3. Буровский П. А. Алгоритм Моцкина-Бургера и вычисление выпуклых оболочек точек n -мерного пространства / П. А. Буровский // Вестник КрасГУ. Сер. физ.-мат. науки. — Красноярск, 2005. — Вып.1. — С. 48–53.
4. Вапник В. Н. Теория распознавания образов / В. Н. Вапник, А. Я. Червоненкис. — М.: Наука, 1974. — 416 с.

5. Донской В. И. Оценки емкости основных классов алгоритмов эмпирического обобщения, полученные pVCD методом / В. И. Донской // Ученые записки ТНУ им. В. И. Вернадского. Серия «Физико-математические науки», 2010. — Т.23 (62). — №2. — С. 56–65.
6. Донской В. И. Слабоопределенные задачи линейного булева программирования с частично заданным множеством допустимых решений / В. И. Донской // ЖВМ и МФ, 1988. — Т. 28. — №9. — С. 1379–1385.
7. Донской В. И. Сложность семейств алгоритмов обучения и оценивание неслучайности извлечения эмпирических закономерностей / В. И. Донской // Кибернетика и системный анализ. — 2012. — №2. — С. 86–96.
8. Донской В. И. Частично определенные задачи оптимизации: подход к решению на основе теории распознавания образов / В. И. Донской // Динамические системы. — К.: Вища школа, 1989. — Вып. 8. — С. 71–77.
9. Колмогоров А. Н. Теория информации и теория алгоритмов / А. Н. Колмогоров — М.: Наука, 1987. —- 304 с.
10. Мазуров Вл. Д. Применение методов теории распознавания образов в оптимальном планировании и управлении Вл. Д. Мазуров // Труды I Всесоюзной конференции по оптимальному планированию и управлению народным хозяйством. — М., ЦЭМИ, 1971. — С. 49.
11. Мазуров Вл. Д. Нестационарные процессы математического программирования. — М.: Наука, 1979. — 288 с.
12. Мазуров Вл. Д. Распознавание образов как средство автоматического выбора процедуры в вычислительных методах / Вл. Д. Мазуров // ЖВМ и МФ, 1970. — Т. 10. — №6. — С. 1520—1525.
13. Препарата Ф. Вычислительная геометрия. Введение / Ф. Препарата, М. Шеймос. — М.:Мир, 1989. — 478 с.
14. Таратынова Н. Ю. Задача линейной оптимизации с частично заданной информацией / Н. Ю. Таратынова // Таврический вестник математики и информатики, 2005. — №1. — С. 82–93.
15. Таратынова Н. Ю. Построение оптимизационной модели по прецедентной начальной информации как задача нелинейной регрессии / Н. Ю. Таратынова // Искусственный интеллект, 2006. — №2. — С. 82–93.
16. Chan T. M. Output-Sensitive Construction of Convex Hulls / T. M. Chan / Ph.D. thesis. — Department of Computer Science, University of British Columbia. — 1995. — 104 p.
17. Donskoy V. I. The Estimations Based on the Kolmogorov Complexity and Machine Learning from Examples / V. I. Donskoy // Proceedings of the Fifth International Conference „Neural Networks and Artificial Intelligence” (ICNNAI'2008). — Minsk: INNS. — 2008. — P. 292–297.
18. Donskoy V. I. Pseudo-Boolean Scalar Optimization Models with Incomplete Information / V. I. Donskoy. — GMOOR Newsletters, 1996. — № 1–2. — P. 20–26.
19. Li Ling. Data Complexity in Machine Learning and Novel Classification Algorithms /Ling Li // Thesis for the Degree of PhD. — California Institute of Technology, 2006. — 103 p.

Статья поступила в редакцию 27.05.2012

УДК 519.8

МАТЕМАТИЧНА МОДЕЛЬ РЕГІОНУ: ЕКОЛОГО-ЕКОНОМІЧНИЙ АСПЕКТ

© О. О. Ємець, О. О. Черненко

Полтавський університет економіки і торгівлі
вул. Коваля, 3, м. Полтава, 36014, Україна
e-mail: *yemetsli@mail.ru, oksanachernenko7@gmail.com*

Abstract. The problem of the ecological safety of production on regional level is considered in the article. The model of functioning of a region is built taking into account the technogenic loading on an environment and the rational use of natural resources.

Вступ

Постановка задачі в загальному вигляді. Проблема запобігання природно-техногенним надзвичайним ситуаціям, зменшення їх впливу на населення, природу й економіку має пріоритетне загальнодержавне значення.

Аналіз останніх досліджень і публікацій. Велика кількість публікацій, особливо в останні роки, присвячена проблемі збалансування економіки та екології (див., наприклад, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]). Зокрема, в [4] розглядається проблема забезпечення екологічної безпеки в умовах трансформації економіки України, а також удосконалення державного регулювання регіональної екологічної безпеки; в [5] розкрито еколого-економічне регулювання природокористування в системі глобальних стратегій розвитку.

Невирішені раніше задачі загальної проблеми. Дана проблема є предметом вивчення низки науковців, однак при цьому задача моделювання функціонування регіону, що мінімізує техногенне навантаження на довкілля, в літературі не розглядалася.

Отже, актуальним є питання про таку модель розвитку галузей промисловості та сільського господарства, яка б забезпечувала економічне зростання і водночас враховувала потенціал навколишнього середовища з точки зору можливостей його використання.

1. Постановка задачі

Розглянемо функціонування регіону протягом року для оцінки його екологічної безпеки та побудуємо модель.

Система критеріїв оцінки екологічної безпеки промислового виробництва регіону орієнтована на оцінку екологічної безпеки окремих промислових об'єктів. Під промисловим об'єктом розуміється окремо розташований проммайданчик підприємства, промислове підприємство або група промислових підприємств, які можуть розглядатися, як єдине джерело техногенного впливу [4].

Вважаємо, що в регіоні зосереджені промислові підприємства (заводи, фабрики), розвинутий гірничо-металургійний комплекс (добування корисних копалин), а також сільське господарство, зокрема, рослинництво та тваринництво.

В результаті діяльності промислових агломерацій, відбувається забруднення літосфери, гідросфери, атмосфери та наноситься шкода біосфері. Крім того, функціонування регіону в цілому потребує і використання природних та добутих ресурсів. В роботі забруднення визначається як використання, зокрема, якщо забруднено один куб. метр повітря, то він рахується як використаний.

Для аналізу функціонування промислового регіону введемо такі позначення: x_i — обсяг виготовленої i-ої продукції в регіоні (тонн), X — множина виготовленої продукції; y_i — кількість добутої i-ої корисної копалини (тонн), Y — множина усіх копалин; z_i — обсяг i-ої вирощеної культури (тонн), Z — множина всіх культур; v_i — обсяг вигодуваного i-ого виду худоби (тонн), V — множина всіх тварин. J_1 — множина забруднюючих речовин, що дає промисловий сектор (підприємства та гірничодобувний комплекс); J_6 — множина забруднюючих речовин, що дає с/г виробництво; I_1 — кількість промислових підприємств; I_2 — кількість майданчиків добування корисних копалин.

В роботі вважається, що одне підприємство може спеціалізуватися на випуску різної продукції, а одна і та ж продукція випускається різними підприємствами.

Аналогічне припущення має місце і для сільського господарства, зокрема, для тваринництва. Для рослинництва вважаємо, що на одному полі може бути посіяна одна культура і одна і та ж культура може бути посіяна на різних полях.

2. Математична модель задачі

Оцінка діяльності промислових підприємств по використанню (забрудненню) природних ресурсів.

1) Забруднення повітря. Концентрація C^n_{jk} забруднюючої речовини j в повітрі за напрямом вітру (швидкість вітру U, м/с) на віддалі b від k-ої промислової агломерації розраховується за формулою [8]:

$$C^n_{jk} = \frac{\sum\limits_{i\in X} m^1_{ji} x_i}{\left[\left(\frac{l^n_k}{2} + b\right) 2tg\,(5,5U^{-0,4}) + h^n_k\right] HU}, \qquad (1)$$

де m_{ji}^1 — інтенсивність викидів j-ої речовини для виробництва одиниці i-ої продукції за одиницю часу, l_k^n, h_k^n — довжина і ширина зони викидів k-ого промислового підприємства; H — висота шару приземної атмосферної циркуляції. Формула (1) дозволяє розрахувати концентрацію окремої забруднюючої речовини j, що випускає окреме підприємство. $\sum\limits_{i \in X} m_{ji}^1 x_i$ — сума викидів j-ої речовини по всім продукціям, що випускає деяке підприємство, можливо по частині продукції з X.

Зауваження 1. Тут і далі будемо вважати залежність інтенсивності викиду забруднюючих речовин від обсягу виготовлення лінійною.

Зауваження 2. Забруднення повітря промисловим сектором оцінюється через порівняння розрахованої концентрації з гранично допустимою (ГДК), яка є документально затвердженою для кожного окремого підприємства та гірничодобувного комплексу. Аналогічне припущення має місце і для оцінки забруднення води. Як наслідок, сума ГДК j-ої речовини від усіх промислових об'єктів не перевищує ГДК j-ої речовини в повітрі.

Зауваження 3. Для оцінки забруднення повітря та води розглядається той момент часу, коли викиди поширились так, що їх можна вважати сталими в часі та просторі.

Порівнюємо C_{jk}^n розрахованою за (1) з $C_{jk}^{\text{ГДК}}$ для j-ої речовини для кожного підприємства регіону. Виділяємо J_2 — множину забруднюючих речовин промислових підприємств, концентрація яких в повітрі перевищує норму. Запишемо функцію у вартісному співвідношенні, що визначає штраф:

$$\sum_{j \in J_2} k_j^1 \left(C_{jk}^n - C_{jk}^{\text{ГДК}} \right), \tag{2}$$

де k_j^1 — коефіцієнт штрафу за перевищення ГДК забруднюючих речовин у повітрі. Співвідношення (2) має місце для кожного підприємства. Враховуючи весь промисловий комплекс регіону, (2) записується так:

$$\sum_{k \in I_1} \sum_{j \in J_2} k_j^1 \left(C_{jk}^n - C_{jk}^{\text{ГДК}} \right) \to \min. \tag{3}$$

З іншого боку, просумуємо концентрації всіх забруднюючих речовин $j \in J_1$ для кожного підприємства і порівняємо їх з $C_{jk}^{\text{ГДК}}$:

$$\sum_{j \in J_1} C_{jk}^n \le \sum_{j \in J_1} C_{jk} \;\; \forall k \in I_1. \tag{4}$$

Для тих підприємств $k \in I_1$, для яких (4) не виконується, рахуємо кількість використаного (забрудненого) повітря, а саме:

$$\sum_{k \in I_1} l_k^n h_k^n H \tag{5}$$

використане (забруднене) повітря для $\forall k \in I_1$, т. що (4) не виконується.

2) Забруднення води. З одного боку, позначивши η_i — кількість використаної води для виробництва одиниці i-ої продукції, об'єм використаної води в промисловому секторі регіону дорівнює:

$$\sum_{i \in X} \eta_i x_i. \tag{6}$$

З іншого, позначивши $D_{jk}^{x_i}$ — об'єм викиду j-ої речовини у воду при виробництві одиниці i-ої продукції, отримаємо об'єм забрудненої води в результаті функціонування k-ого підприємства, $k \in I_1$:

$$\sum_{i \in X} D_{jk}^{x_i} x_i = D_{jk}^n. \tag{7}$$

Порівнюємо (7) з $D_{jk}^{x_i}$ — ГДК j-ої речовини у воді. Виділяємо J_3 — множину забруднюючих речовин промислових підприємств, концентрація яких у воді перевищує норму. Функція у вартісному співвідношенні, що визначає штраф, запишеться так:

$$\sum_{j \in J_3} k_j^2 \left(D_{jk}^n - D_{jk}^{\text{ГДК}} \right), \tag{8}$$

де k_j^2 - коефіцієнт штрафу за перевищення ГДК забруднюючих речовин у воді. Враховуючи весь промисловий комплекс регіону, (8) запишеться так:

$$\sum_{k \in I_1} \sum_{j \in J_3} k_j^2 \left(D_{jk}^n - D_{jk}^{\text{ГДК}} \right) \to \min. \tag{9}$$

3) Використана енергія обчислюється за формулою:

$$\sum_{i \in X} p_i x_i, \tag{10}$$

де p_i — кількість використаної енергії для виробництва одиниці i-ої продукції. Отже, модель функціонування промислових підприємств регіону набуде вигляду:

$$\sum_{i \in X} m_i x_i - \sum_{k \in I_1} \sum_{j \in J_2} k_j^1 \left(C_{jk}^n - C_{jk}^{\text{ГДК}} \right) - \sum_{k \in I_1} \sum_{j \in J_3} k_j^2 \left(D_{jk}^n - D_{jk}^{\text{ГДК}} \right) \to \max \tag{11}$$

за обмежень

$$a_i \leq x_i \ \ \forall i \in X, \tag{12}$$

де m_i — прибуток з виробництва одиниці i-ої продукції, a_i — потреба в i-ій продукції.

Використання природних ресурсів гірничодобувною галуззю.

1) Забруднення повітря. Міркування аналогічні до попередніх приводять до співвідношення:

$$\sum_{k\in I_2}\sum_{j\in J_4} k_j^1\left(C_{jk}^{\text{г}} - C_{jk}^{\text{ГДК}}\right) \to \min, \tag{13}$$

де $C_{jk}^{\text{г}}$ — концентрація j-ої забруднюючої речовини в повітрі від діяльності k-ого гірничодобувного комплексу, J_4 — множина забруднюючих речовин діяльності гірничодобувного комплексу, концентрація яких в повітрі перевищує норму. З іншого боку, просумуємо концентрації всіх забруднюючих речовин $j \in J_1$ для кожного промислового майданчика і порівняємо їх з $C_{jk}^{\text{ГДК}}$:

$$\sum_{j\in J_1} C_{jk}^{\text{г}} \le \sum_{j\in J_1} C_{jk}^{\text{ГДК}} \quad \forall k \in I_2. \tag{14}$$

Для тих гірничодобувних комплексів $k \in I_2$, для яких (14) не виконується, обчислюємо кількість використаного повітря

$$\sum_{k\in I_2} l_k^{\text{г}} h_k^{\text{г}} H. \tag{15}$$

2) Забруднення води обчислюється за формулою:

$$\sum_{i\in Y} \beta_i y_i, \tag{16}$$

де β_i — кількість використаної (забрудненої) води для добування одиниці обсягу i-ої корисної копалини. З іншого боку, запишемо функцію у вартісному співвідношенні, що визначає штраф

$$\sum_{k\in I_2}\sum_{j\in J_5} k_j^2\left(D_{jk}^{\text{г}} - D_{jk}^{\text{ГДК}}\right) \to \min, \tag{17}$$

де $D_{jk}^{\text{г}} = \sum\limits_{i\in Y} D_{jk}^{y_i} y_i$ – об'єм забрудненої води в результаті функціонування k-ого гірничого комплексу, J_5 — множина забруднюючих речовин діяльності гірничо-добувного комплексу, концентрація яких у воді перевищує норму.

3) Використання енергії обчислимо за формулою:

$$\sum_{i\in Y} \gamma_i y_i, \tag{18}$$

де γ_i — кількість використаної енергії для добування 1 тонни i-ої корисної копалини. Отже, функціонування гірничого комплексу регіону може бути представлено моделлю:

$$\sum_{i\in Y} n_i y_i - \left(\sum_{k\in I_2}\sum_{j\in J_4} k_j^1\left(C_{jk}^{\text{г}} - C_{jk}^{\text{ГДК}}\right) + \sum_{k\in I_2}\sum_{j\in J_5} k_j^2\left(D_{jk}^{\text{г}} - D_{jk}^{\text{ГДК}}\right)\right) \to \max \tag{19}$$

за обмежень:

$$b_i \leq y_i \leq B_i, \quad i \in Y, \tag{20}$$

де b_i — кількісна потреба в i-ій корисній копалині; B_i — максимально можлива кількість добування i-ої корисної копалини; n_i — прибуток з добування одиниці i-ої корисної копалини.

Аналіз та оцінка використання природних ресурсів рослинною галуззю сільського господарства.

1) Забруднення повітря обчислюється за формулою:

$$\sum_{i \in Z} \delta_i z_i, \tag{21}$$

де δ_i — сумарна кількість використаного повітря для вирощування одиниці i-ої культури (розпилювання міндобрив, випаровування, робота техніки).

З іншого боку, $C_{ji}^{\text{р}} z_i \leq C_j^{\text{ГДК}}, \quad j \in J_6$, де $C_j^{\text{р}}$ — об'єм викиду j-ої забруднюючої речовини в повітря при вирощуванні одиниці i-ої культури. Міркуючи аналогічно побудові оцінці діяльності промислового сектора регіону, запишемо обмеження по всім забруднюючим речовинам, які надходять у повітря, в результаті посіву та вирощування культур:

$$\sum_{i \in Z} \sum_{j \in J_6} C_{ji}^{\text{р}} z_i \leq \sum_{j \in J_6} C_j^{\text{ГДК}}. \tag{22}$$

Права частина (22) — ГДК забруднення повітря рослинною галуззю сільського господарства (може бути розрахована від величини $C_j^{\text{ГДК}}$ в повітрі як відсоток техногенного навантаження рослинною галуззю сільського господарства на навколишнє середовище).

2) Забруднення води розраховуємо за формулою:

$$\sum_{i \in Z} \phi_i z_i, \tag{23}$$

де ϕ_i — кількість використаної води для вирощування одиниці i-ої культури (зрошення і т. д.). З іншого боку, $D_j^{\text{р}} z_i \leq D_j$, де $D_j^{\text{р}}$ — обсяг викиду (надходження) j-ої забруднюючої речовини у воду при вирощуванні одиниці i-ої культури. Для регіону оцінка забруднення водних ресурсів в результаті вирощування зернових культур запишеться співвідношенням:

$$\sum_{i \in Z} \sum_{j \in J_6} D_j^{\text{р}} z_i \leq \sum_{j \in J_6} D_j^{\text{ГДК}}. \tag{24}$$

3) Використання енергії розрахуємо так:

$$\sum_{i \in Z} \varphi_i z_i, \tag{25}$$

де φ_i - кількість використаної енергії для вирощування одиниці i-ої культури. Отже, вирощування культур може бути описане моделлю:

$$\sum_{i \in Z} k_i z_i \to \max, \tag{26}$$

де k_i — прибуток з вирощеної 1 тонни i-ої культури, за обмежень

$$z_i \geq c_i, \quad \forall i \in Z, \tag{27}$$

$$\sum_{i \in Z} s_i^p z_i \leq S^p_{\max} \tag{28}$$

та за обмежень (22), (24), де c_i — кількісна потреба в i-ій культурі; $s_i{}^p$ — площа необхідна для вирощування одиниці i-ої культури, $S^p_{\max}$ — максимальна площа, що відводиться під посів.

Використання природних ресурсів тваринницькою галуззю сільського господарства.

1) Забруднення повітря обчислюється за формулою:

$$\sum_{i \in V} \alpha_i v_i, \tag{29}$$

де α_i — сумарна кількість використаного повітря для однотонного приросту маси тварини i-го виду (робота техніки, випаровування стоків).

2) Забруднення води розраховуємо за формулою:

$$\sum_{i \in V} \mu_i v_i, \tag{30}$$

де μ_i — кількість використаної води для однотонного приросту маси тварини i-го виду (стоки тваринницьких комплексів).

3) Використання енергії обчислимо так:

$$\sum_{i \in V} \sigma_i v_i, \tag{31}$$

де σ_i — кількість використаної енергії для однотонного приросту маси тварини i-го виду, $i \in V$.

4) використання рослинництва як корму

$$\sum_{i \in Z_1} z_i \geq \sum_{j \in V} r_j v_j, \tag{32}$$

де r_j — об'єм культур необхідних для вирощування тонни тварини i-го виду, Z_1 — множина кормових культур. Отже, функціонування тваринницького сектора може бути описане моделлю:

$$\sum_{i \in V} p_i v_i \to \max, \tag{33}$$

де p_i — прибуток від вирощування одиниці маси i-ого виду тварин, за обмежень

$$d_i \leq v_i, \quad \forall i \in V, \tag{34}$$

$$\sum_{i \in V} s_i^m v_i \leq S_{\max}^m \tag{35}$$

та за обмежень (32), де d_i — кількісна потреба в i-ому виді тварин; ${s_i}^m$ — площа необхідна для одиниці маси i-ого виду тварин, $S_{\max}^m$ — максимальна площа, що відводиться під тваринницький сектор.

Інтегральна оцінка екологічного стану території регіону.

В цілому по регіону запишемо обмеження на використання ресурсів (повітря, вода, енергія) у вигляді

$$H_\phi^r \leq K^r, \tag{36}$$

де H_ϕ^r — показник фактичного стану r-ої компоненти природного середовища (його використання), K^r — показник критичного стану (норма). Зокрема, маємо такі обмеження: а) на використання повітря

$$\sum_{k \in I_1} l_k^n h_k^n H + \sum_{k \in I_2} l_k^{\text{г}} h_k^{\text{г}} H + \sum_{i \in Z} \delta_i z_i + \sum_{i \in V} \alpha_i v_i + A^{\text{н.в.}} \leq A, \tag{37}$$

де $A^{\text{н.в.}}$ — непромислове використання повітря (населення, транспорт і т. і.); A — константа, що визначає можливість відновлення повітря (визначається з к-ті насаджень; встановлених очисних споруд і т. і.); б) на використання води

$$\sum_{i \in X} \eta_i x_i + \sum_{i \in Y} \beta_i y_i + \sum_{i \in Z} \phi_i z_i + \sum_{i \in V} \mu_i v_i + W^{\text{н.в.}} \leq W, \tag{38}$$

де $W^{\text{н.в.}}$ — непромислове використання води, W — кількісна константа можливої до використання води в регіоні; в) на використання енергії

$$\sum_{i \in X} p_i x_i + \sum_{i \in Y} \gamma_i y_i + \sum_{i \in Z} \varphi_i z_i + \sum_{i \in V} \sigma_i v_i + E^{\text{н.в.}} \leq E, \tag{39}$$

$E^{\text{н.в.}}$ — непромислове використання енергії, E — кількісна константа можливої до використання енергії в регіоні.

Використовуючи роботу [9], визначимо інтегральний показник деградації екологічного стану території регіону:

$$P_{ec} = \sum_{r=1}^{3} \left(a_r \ln \left(1 - \frac{H_\phi^r - K^r}{K^r} \right) + c_r \right) g_r, \tag{40}$$

де a_r, c_r — коефіцієнти нормувальної функції, g_r — відповідні вагові коефіцієнти, що обумовлені відносною перевагою і встановленою важливістю для забезпечення стабільності функціонування компонентів природного середовища в цілому, $\sum_{r=1}^{t} g_r = 1$, де t — кількість компонент середовища [9]. Коефіцієнти a_r, c_r, g_r пропонується визначати експертам.

Підставляючи (37)-(39) в (40), отримаємо таку функцію:

$$g_1 \left(a_1 \ln \left(1 - \frac{H_\phi^n - K^n}{K^n} \right) + c_1 \right) + g_2 \left(a_2 \ln \left(1 - \frac{H_\phi^{\text{в}} - K^{\text{в}}}{K^{\text{в}}} \right) + c_2 \right) +$$
$$+ g_3 \left(a_3 \ln \left(1 - \frac{H_\phi^{\text{e}} - K^{\text{e}}}{K^{\text{e}}} \right) + c_3 \right) \to \min . \qquad (41)$$

Окремо оцінимо екологічну шкоду регіону від забруднення ґрунту. Для цього скористаємося формулою [10]:

$$T = M \; T' \; k_z, \qquad (42)$$

де T — екологічна шкода від забруднення ґрунту, M — маса викидів у ґрунт (т./рік), T' — питома шкода від забруднення 1 тонни ґрунту (грн./т.), k_z — коефіцієнт цінності земельних ресурсів [10]. Розіб'ємо регіон на територіальні одиниці по цінності земельних ресурсів, $z \in N$, враховуючи відходи. Співвідношення маси готової продукції та відходів при виробництві визначається експертами. Як приклад, будемо вважати, що відходи промислового комплексу (важка та легка промисловість) складають $0,5(x_i + y_i)$ [11], с/г виробництва – $0,7(z_i + v_i)$ [12]. Таким чином, маса відходів складає $0,5(x_i + y_i) + 0,7(z_i + v_i)$. Екологічна шкода регіону від забруднення ґрунту запишеться так:

$$\sum_{z \in N} (0,5(x_i + y_i) + 0,7(z_i + v_i))\, T' k_z \to \min . \qquad (43)$$

Питома шкода T' може бути уточнена шляхом врахування ступеня шкідливості викидів на людину та довкілля через введення вагових коефіцієнтів λ_q (визначаються експертами), $\sum_{q=1}^{4} \lambda_q = 1$, де q визначає клас небезпеки відходу: надзвичайно небезпечні, високо небезпечні, помірно і мало небезпечні [12]. Оцінимо еколого-економічну ефективність регіонального природокористування за формулою [10]:

$$E_p = \frac{B_e - B_n}{R_n + R_0 K}, \qquad (44)$$

де E_p — ефективність регіонального природокористування; B_e — вартість екологічно чистої продукції підприємств регіону (безвідходні та маловідходні) технології; B_n — вартість продукції, що вироблена з порушенням екологічних норм; R_n — поточні витрати на охорону, відновлення та експлуатацію природних ресурсів; K — нормативний коефіцієнт ефективності природоохоронних затрат; R_0 — короткотермінові витрати на охорону та відновлення природного середовища (ресурсів) [10].

Для запропонованої моделі функціонування регіону і враховуючи, що с/г виробництво є маловідходним (великі тваринницькі комплекси відносимо до виробництва), отримаємо таку функцію

$$\frac{\sum\limits_{i\in V} p_i x_i + \sum\limits_{i\in Z} k_i z_i - \sum\limits_{i\in X} m_i x_i - \sum\limits_{i\in Y} n_i y_i}{R_n + R_0 K} \to \max\,. \tag{45}$$

Таким чином, функціонування регіону протягом року з врахуванням економічного зростання та екологічної безпеки описується моделлю: оптимізувати критерії (11), (19), (26), (33), (41), (43), (45) за обмежень (4), (12), (14), (20), (22), (24), (27), (28), (32), (34), (35). У найпростішому випадку маємо багатокритеріальну задачу лінійного програмування. Враховуючи, що цільові функції мають різну вимірність, необхідно ввести деяке перетворення, що приведе критерії до безрозмірного вигляду та застосувати метод обмежень [13].

Висновки

Висновки з даного дослідження. У роботі розглянуто проблему збалансування економіки та екології і побудовано модель функціонування регіону у вигляді багатокритеріальної задачі лінійного програмування.

Перспективи. В подальшому доцільно дослідити побудовану модель та створити програмне забезпечення для розв'язування практичних задач, що зводяться до неї.

Список літератури

1. Інтегроване управління та поводження з твердими побутовими відходами у Вінницькій області: Монографія / під ред. В. Г. Петрука. — Вінниця: УНІВЕРСУМ. — 2007. — 159 с.
2. Герасимчук З. В. Екологічна безпека регіону: діагностика та механізм забезпечення. Монографія / З. В. Герасимчук, А. О. Олексюк. — Луцьк: Надстир'я. — 2007. — 280 с.
3. Скалецький Ю. М. Реконструкція і верифікація доз опромінювання військових ліквідаторів. Монографія / Ю. М. Скалецький. — К.: Логос, 2007. — 223 с.
4. Хлобистов Є. В. Екологічна безпека трансформаційної економіки / Є. В. Хлобистов. — К.: Чорнобильінтерінформ, 2004. — 336 с.
5. Хвесик М. А. Размещение производительных сил и региональная экономика: Уч. пособ. / М. А. Хвесик, Л. М. Горбач, П. П. Пастушенко. — Изд-во: Кондор, 2009. — 344 с.
6. Гахович Н. Состояние и проблемы экологизации промышленного производства / Н. Гахович // Экономика Украины. — 2008. — № 4. — С. 73-81.
7. Хвесик М. А. Стратегічні пріоритети сталого розвитку легкої промисловості регіонів країни / М. А. Хвесик, О. В. Царенко // Продуктивні сили і регіональна економіка. — 2008. — Ч. 2. — С. 46–53.

8. Шварцман В. М. Підвищення екологічної безпеки шляхом зменшення кислотоутворення в атмосфері на техногенно навантажених територіях гірничо-металургійного комплексу: автореф. дис... канд. техн. наук: 21.06.01 / В. М. Шварцман; Нац. гірн. ун-т. — Д., 2004. — 20 с.
9. Копач П. І. Кількісна оцінка екологічної безпеки гірничодобувних районів // Екологія і природокористування, 2009. Випуск 12. — С. 48–53.
10. Регіональна економіка. Навчально-методичний посібник. — Полтава: РВВ ПУСКУ — 2008. — 185 с.
11. Васюкова Г. Т. Екологія. Підручник / Г. Т. Васюкова, Ярошева О. І. — К.: Кондор, 2009. — 524 с.
12. Черевко Г.В. Економіка природокористування / Г. В. Черевко, М. І. Яцків. — Львів: Світ, 1995. — 208 с.
13. Таха Х. А. Введение в исследование операций. 7-е издание.: Пер. с англ / Таха Х. А. — Москва: Издательский дом «Вильямс», 2005. — 912 с.

Статья поступила в редакцию 20.12.2011

УДК 519.6

МИНИМАЛЬНЫЕ ПО ВКЛЮЧЕНИЮ ДЕРЕВЬЯ ШТЕЙНЕРА: АЛГОРИТМ ПОСТРОЕНИЯ

© А. В. Ильченко, В. Ф. Блыщик

Таврический национальный университет им. В. И. Вернадского
факультет математики и информатики
пр-т Вернадского, 4, г. Симферополь, 95007, Украина
e-mail: *veb@land.ru*

Abstract. The concept of Steiner tree minimal with respect to inclusion is under consideration. The algorithm of constructing all Steiner trees minimal with respect to inclusion and its substantiation are given. The Steiner tree minimal with respect to inclusion which has minimal weight is considered as the solution of the Steiner tree problem.

Введение

Задача Штейнера используется в качестве математической модели широкого круга задач, связанных с исследованием и разработкой различных сетевых структур. Детальное рассмотрение и обсуждение различных аспектов этой задачи отражено в обзоре [1].

Целью данной работы является рассмотрение понятия минимального по включению дерева Штейнера; описание и обоснование алгоритма построения всех минимальных по включению деревьев Штейнера, как одного из алгоритмов точного решения задачи Штейнера на графе.

1. Задача Штейнера и минимальное по включению дерево Штейнера

Определение 1. Задан связный неориентированной граф $G = (V, E, w)$, где

V — конечное множество вершин;

E — конечное множество рёбер графа;

$w: E \to R_+^0$ — весовая функция, сопоставляющая каждому ребру графа некоторое неотрицательное число — «вес» этого ребра;

$T \subseteq V$ — непустое подмножество вершин графа. Вершины подмножества T называются терминальными.

Вес любого подграфа — это сумма весов рёбер этого подграфа.

Требуется найти связный подграф наименьшего веса, вершины которого содержат все терминальные вершины исходного графа.

Определение 2. Связный ациклический подграф, вершины которого содержат все терминальные вершины, называется *деревом Штейнера.*

Определение 3. Дерево Штейнера называется *минимальным по включению*, если подграф, получаемый из этого дерева в результате удаления любой его вершины или любого его ребра, не является деревом Штейнера.

Утверждение 1. *Для каждого дерева Штейнера существует единственное содержащееся в нём минимальное по включению дерево Штейнера.*

Извлечь минимальное по включению дерево Штейнера, содержащееся в ациклическом графе, можно, например, с помощью следующего алгоритма [2]: удалить из исходного дерева все висячие нетерминальные вершины. Этот алгоритм удобно называть «стрижка Штейнера».

Примечание 1. Если в задаче Штейнера исходный граф — ациклический, то минимальное по включению дерево Штейнера является решением задачи Штейнера для этого графа.

Примечание 2. В случае, когда исходный граф содержит циклы, решение задачи Штейнера надо искать среди минимальных по включению деревьев Штейнера. Поэтому, один из возможных подходов к решению задачи заключается в следующем: построить все минимальные по включению деревья Штейнера и из них выбрать дерево наименьшего веса. Если деревьев наименьшего веса окажется несколько, то в качестве решения задачи можно взять любое из этих деревьев.

Именно такой подход рассматривается далее.

Определение 4. Компонента связности, вершины которой одержат все терминальные вершины, называется *компонентой Штейнера* (КШ).

Примечание 3. Дерево Штейнера — частный случай подграфа, содержащего компоненту Штейнера.

Утверждение 2. *Минимальный по включению подграф, содержащий компоненту Штейнера, состоит из одной компоненты связности, то есть, является связным подграфом.*

Доведення. Пусть $A \in P(E)$ — минимальный по включению подграф, содержащий компоненту Штейнера. Допустим, что подграф A — не связный и, значит, состоит более, чем из одной компоненты. Удалим из подграфа A те компоненты, которые не являются компонентой Штейнера. Подграф, получившийся в результате такого удаления, содержит компоненту Штейнера. Следовательно, подграф A не был минимальным по включению подграфом, содержащим компоненту Штейнера. Противоречие. $\square$

2. Решётка реберно-порожденных подграфов

Определение 5. Пусть $G' = (V', E', w)$ — подграф графа G. Подграф G' называется *реберно-порожденным*, если каждая из его вершин инцидентна хотя бы одному из рёбер подмножества E' [3].

Поскольку рёберно-порождённый подграф однозначно определяется множеством своих рёбер, то для задания такого подграфа достаточно указать множество его рёбер. Множество всех рёберно-порождённых подграфов будем отождествлять с множеством всех подмножеств, порождаемых множеством рёбер E исходного графа $G = (V, E, w)$ и обозначать $P(E)$.

Отношение включения «$\subseteq$» на элементах семейства $P(E)$ определяет отношение предшествования «$\leq$» на рёберных подграфах.

Определение 6. Пусть $A, B \in P(E)$. Подграф A предшествует подграфу B, если множество рёбер подграфа A является подмножеством множества рёбер подграфа B :

$$A \leq B \Leftrightarrow A \subseteq B.$$

В силу свойств отношения включения «$\subseteq$», отношение предшествования «$\leq$» является отношением порядка. Семейство реберно-порожденных подграфов $P(E)$, упорядоченное таким образом, является полной решёткой [3].

Обычным образом отношение предшествования «$\leq$» порождает отношение строгого «$<$» и непосредственного «$\prec$» предшествования:

$$A < B \Leftrightarrow A \leq B \,\&\, A \neq B,$$

$$A \prec B \Leftrightarrow A < B \,\&\, \overline{\exists} C \in P(E) : \ A < C < B.$$

Утверждение 3. *Подграф A непосредственно предшествует подграфу B тогда и только тогда, когда*

$$A \subset B \,\&\, |A| = |B| - 1.$$

Множество всех подграфов, непосредственно предшествующих графу A, обозначается $N\_(A)$:

$$N\_(A) = \{B \in P(E) : \ B \prec A\}.$$

3. Свойства элементов решётки рёберно-порождаемых подграфов

Утверждение 4. *Всякий подграф ациклического графа, не содержащего компоненту Штейнера, является ациклическим подграфом, не содержащим компоненту Штейнера.*

Следствие 1. *Из утверждения 4 следует, что семейство рёберно-порождаемых ациклических подграфов, не содержащих компонент Штейнера, является порядковым идеалом* [4] *решётки* $(P(E), \leq)$.

Утверждение 5. *Всякий надграф графа, содержащего цикл или компоненту Штейнера, является графом, содержащим цикл или компоненту Штейнера.*

Следствие 2. *Из утверждения 5 следует, что семейство рёберно-порождаемых подграфов, содержащих циклы или компоненты Штейнера, является порядковым фильтром* [4] *решётки* $(P(E), \leq)$.

Определение 7. Рёберно-порождаемый подграф называется кандидатом (кандидатом в порядковый идеал, состоящий из ациклических подграфов, не содержащих компонент Штейнера), если все строго предшествующие ему рёберно-порождаемые подграфы являются ациклическими подграфами, не содержащими компонент Штейнера.

Утверждение 6. *Рёберно-порождаемый подграф является кандидатом тогда и только тогда, когда каждый непосредственно предшествующий ему рёберно-порождаемый подграф является ациклическим подграфом, не содержащим компоненту Штейнера.*

Утверждение 7. *Кандидат, содержащий компоненту Штейнера, является минимальным по включению подграфом, содержащим компоненту Штейнера.*

Действительно, если подграф — кандидат, то, в соответствии с определением 7, ни один из строго предшествующих ему подграфов не содержит компонент Штейнера. Следовательно, кандидат — минимальный по включению подграф, содержащий компоненту Штейнера.

Утверждение 8. *Кандидат, содержащий компоненту Штейнера, является ациклическим подграфом.*

Доведення. Пусть $A \in P(E)$ — кандидат, содержащий компоненту Штейнера.

Согласно утверждению 7, подграф A — минимальный по включению подграф, содержащий компоненту Штейнера. Следовательно, подграф A состоит из одной компоненты связности (утверждение 2).

Допустим, что A содержит цикл. Удалим одно из рёбер этого цикла. Подграф, получившийся в результате удаления ребра, обозначим B. В силу утверждения 3, подграф B — один из подграфов, непосредственно предшествующих подграфу A.

Удаление ребра не влечёт удаления вершин [3]. Следовательно, множества вершин графа A и его подграфа B совпадают.

Удаление ребра цикла не нарушает связности [3]. Поэтому, подграф B также состоит из одной компоненты. И эта компонента содержит все терминальные вершины. Следовательно, подграф B содержит компоненту Штейнера.

Это противоречит тому, что подграф A — кандидат. □

Утверждение 9. *Кандидат, содержащий компоненту Штейнера, является минимальным по включению деревом Штейнера.*

Доведення. Пусть $A \in P(E)$ — кандидат, содержащий компоненту Штейнера. Тогда A — минимальный по включению подграф, содержащий компоненту Штейнера (утверждение 7). В соответствии с утверждением 2, A — связный подграф.

Кроме этого, A — ациклический подграф (утверждение 8).

Таким образом, A — минимальный по включению связный ациклический подграф, содержащий все терминальные вершины исходного графа. Следовательно, подграф A — минимальное по включению дерево Штейнера. □

Из утверждения 9 следует, что одним из возможных подходов к построению всех минимальных по включению деревьев Штейнера является построение всех кандидатов и отбор тех из них, которые содержат компоненту Штейнера.

Определение 8. Рёберно-порождаемый подграф называется потенциальным кандидатом (ПК), если хотя бы один, из непосредственно предшествующих ему рёберно-порождаемых подграфов, является ациклическим подграфом, не содержащим компоненту Штейнера.

Утверждение 10. *Каждый кандидат является потенциальным кандидатом.*

Утверждение 11. *Потенциальный кандидат не является кандидатом тогда и только тогда, когда среди непосредственно предшествующих ему подграфов есть хотя бы один, содержащий цикл или компоненту Штейнера.*

4. Диаграмма Хассе и линейный порядок на элементах одного уровня

Удобно считать, что для ациклических, не содержащих компонент Штейнера подграфов семейства $P(E)$ построена диаграмма Хассе, нумерация уровней в которой проведена следующим образом:

Уровень 0 — подграф, множество рёбер которого пусто;

Уровень 1 — все одно рёберные подграфы;

Уровень 2 — все двух рёберные подграфы;

Уровень k — все k-рёберные подграфы.

В диаграмме Хассе, внутри каждого уровня, подграфы упорядочиваются в каком-либо линейном порядке. Например, лексикографическом. Тогда, для каждого подграфа A, состоящего из k элементов, непосредственно предшествующие ему подграфы (элементы семейства $N\_(A)$), также линейно упорядочены.

Согласно утверждению 3, подграф A может быть построен из любого подграфа семейства $N\_(A)$ добавлением соответствующего ребра. Чтобы избежать дублирования, для порождения подграфа A всегда будет использоваться тот подграф семейства $N\_(A)$, который является линейно наибольшим подграфом этого семейства.

$lexmax N\_(A)$ – обозначение линейно наибольшего подграфа семейства $N\_(A)$.

Если подграф $lexmax N\_(A)$ является циклическим или содержит компоненту Штейнера, то он отсутствует в уровне, предшествующем уровню расположения подграфа A. В этом случае подграф A не порождается, так как в соответствии с утверждением 5, все следующие за ним (в решёточном порядке) подграфы исключаются из рассмотрения и не вычисляются.

5. Алгоритм вычисления всех минимальных по включению деревьев Штейнера

Таблица 1. Обозначения алгоритма

Обозначение	**Пояснение**
n	Количество вершин исходного графа G
k	Номер уровня, количество рёбер подграфа, номер итерации
$\widetilde{C_k}$	Семейство всех k -рёберных потенциальных кандидатов
C_k	Семейство всех k-рёберных кандидатов
L_k	Семейство всех k-рёберных ациклических подграфов, не содержащих компонент Штейнера
S	Семейство всех деревьев Штейнера наименьшего веса

Псевдокод алгоритма

Вход: $G = (V, E, w)$ – связный неориентированный взвешенный граф,
$w : E \to R_+$ — весовая функция,
$T \subseteq V$ — подмножество терминальных вершин,
S_0 — одно из минимальных по включению деревьев Штейнера.

Выход: S – семейство всех деревьев Штейнера наименьшего веса.

$W_0 :=$ Вес(S_0);
$S := \{S_0\}$;
$n := |V|$;
$L_0 := \{\varnothing\}$;
$k := 0$;
Цикл пока $k < n - 1$ и $L_k \neq \varnothing$
    $k := k + 1$;
    $\widetilde{C_k} := \{A \in P(E) | lexmaxN\_(A) \in L_{k-1}\}$;
    $C_k := \{C \in \widetilde{C_k} | N\_(C) \subseteq L_{k-1}\}$;
    $L_k := \varnothing$
    Цикл по всем $A \in C_k$
        Если A —- ациклический, не содержащий КШ, **то**
            $L_k := L_k \cup \{A\}$;
        end Если
        Если A —- ациклический, содержащий КШ, **то**
            $W^* :=$ Вес(A);
            Если $W^* = W_0$ и $A \notin S$, **то**
                $S := S \cup \{A\}$;
            end Если
            Если $W^* < W_0$, **то**
                $S := \{A\}$;
                $W_0 := W^*$;
            end Если
            Конец если $W^* < W_0$
        end Если
        Конец если A —- ациклический, содержащий КШ
    end Цикл пока
    Конец цикла по всем $A \in C_k$
end Цикл пока
Конец цикла пока $k < n - 1$ и $L_k \neq \varnothing$
Возврат S;

Пояснения к псевдокоду алгоритма.

Шаг 1 – определяется вес W_0 минимального по включению дерева Штейнера S_0.

Подграф S_0 — это одно из минимальных по включению деревьев Штейнера. Метод, с помощью которого получен подграф S_0, не фиксируется. Например, один из возможных способов получить S_0 следующий:

1. Используя какой-либо известный алгоритм, (например, алгоритм Краскала [3]), вычислить остов минимального веса.

2. К полученному остову применить алгоритм «стрижка Штейнера».

Шаг 2 — текущее семейство деревьев Штейнера наименьшего веса инициализируется подграфом S_0.

Шаг 3 — определяется количество вершин исходного графа.

Шаг 4 — строятся все 0-рёберные подграфы. Такой подграф один: подграф, множество рёбер которого пусто.

Шаг 5 — инициализируется счётчик итераций.

Шаг 6 — условия $k < n - 1$ и $L_k \neq \varnothing$ определяют условие выполнения очередной итерации, реализующей построение необходимых конструкций следующего уровня из ациклических подграфов, предшествующего уровня, не содержащих компонент Штейнера.

Шаг 7 — определяет номер следующей итерации (номер следующего уровня).

Шаг 8 — формирует $\widetilde{C_k}$ - семейство k-рёберных потенциальных кандидатов, непосредственно следующих за подграфами из семейства L_{k-1} (определение 8).

Шаг 9 – из потенциальных кандидатов семейства $\widetilde{C_k}$ отбираются кандидаты, формирующие семейство C_k (утверждение 6).

Шаг 10 – изначально семейство ациклических подграфов текущего уровня, не содержащих компонент Штейнера – пусто.

Шаг 11 – цикл по всем подграфам-кандидатам k-го уровня.

Шаг 13 – ациклический подграф, не содержащий компоненты Штейнера (шаг 12), добавляется в семейство L_k.

Шаг 14 – если текущий подграф A — ациклический и содержит компоненту Штейнера, то

Шаг 15 – вычислить вес подграфа A.

Шаг 16-17 – если вес текущего подграфа равен порогу W_0 и подграф A не включён в семейство решений задачи, то добавить его в семейство S.

Шаг 18 – если вес текущего подграфа меньше порога W_0, то

Шаг 19 – текущее подмножество дерева Штейнера наименьшего веса инициализируется подграфом A.

Шаг 20 – в качестве весового порога использовать вес подграфа A.

Шаг 25 – возвращается результат работы алгоритма: множество всех деревьев Штейнера наименьшего веса

Примечание 4. Если желательно ограничиться получением какого-либо одного решения задачи Штейнера, то для этого в семейство S необходимо включать только одно из деревьев Штейнера наименьшего веса. Чтобы это обеспечить, достаточно исключить из рассмотренного алгоритма шаги (шаги 16, 17), добавляющие в текущее семейство S новые подграфы наименьшего веса

Заключение

В работе получены следующие результаты. Определены понятия минимального по включению дерева Штейнера и минимальной по включению компоненты Штейнера, определено понятие кандидата в порядковый идеал, состоящий из ациклических подграфов, не содержащих компонент Штейнера. Показано, что кандидат, содержащий компоненту Штейнера, является минимальным по включению деревом Штейнера. Предложен алгоритм поуровневого вычисления всех деревьев Штейнера, минимальных по включению.

Список літератури

1. Гордеев Э. Н. Задача Штейнера. Обзор / Э. Н. Гордеев, О. Г. Тарасцов// Дискретная математика. — 1993. — Т. 5. Вып. 2. — С. 3–28.
2. Beasley J. An algorithm for the Steiner problem in graphs / J. Beasley // Networks 14 (1984). — 1984. — pp. 147—159.
3. Лекции по теории графов / [Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И.] / Изд. 2-е, испр. — М.: Книжный дом «ЛИБРОКОМ», 2009. — 392 с.
4. Биркгоф Г. Теория решеток: Пер. с англ. / Г. Биркгоф — М.: Наука, 1984. — 568 с.

Статья поступила в редакцию 01.06.2012

УДК 519.658

АЛГОРИТМ ВЫДЕЛЕНИЯ БЛОЧНО-ДРЕВОВИДНОЙ СТРУКТУРЫ В РАЗРЕЖЕННЫХ ЗАДАЧАХ ДИСКРЕТНОЙ ОПТИМИЗАЦИИ

© Д. В. Лемтюжникова, А. В. Свириденко, О. А. Щербина

Таврический национальный университет им. В. И. Вернадского
факультет математики и информатики
пр-т Вернадского, 4, г. Симферополь, 95007, Украина
e-mail: *darabbt@gmail.com, oleks.sviridenko@gmail.com, oshcherbina@gmail.com*

Abstract. Authors propose an algorithm of computing block-tree structure for sparse matrices. Finkelshtein's algorithm for constructing quasiblock structures is modified and implemented. Preliminary benchmarking with test problems was done. Quality of computed quasiblock structures is investigated.

Введение

Многие задачи дискретной оптимизации (ДО), возникающие на практике, содержат огромное число переменных и/или ограничений, что создает сложности при попытке решения этих задач с помощью современных решателей. Эти задачи зачастую имеют разреженные матрицы ограничений [5], причем блочная структура в них не всегда явно выделена. Использование разреженности матриц ограничений в задачах ДО может весьма эффективно организовать процесс поиска решения задачи ДО. *Ф. Гловер* заметил в [1], что «... разработка специальных вычислительных методов, ориентированных на операции с такими разреженными матрицами ... может в ближайшем будущем в определенной степени изменить существующие представления о непреодолимых сложностях решения отдельных классов задач целочисленного программирования, имеющих большую размерность».

Перспективными декомпозиционными методами, использующими разреженность матрицы ограничений задач ДО, являются локальные элиминационные алгоритмы [10], включающие локальные алгоритмы декомпозиции [7], алгоритмы несериального динамического программирования (НСДП) [14], [24], алгоритмы сегментной элиминации [16]. Локальные элиминационные алгоритмы относятся к классу локальных алгоритмов, общая теория которых для решения ряда важных задач дискретной математики была развита Ю. И. Журавлевым [2].

Теоретико-графовые методы и интерпретации [3] играют важную роль при выделении блочных структур в разреженных задачах ДО. Для выделения специальных структур задач ДО представляются перспективными такие графовые декомпозиционные подходы, как методы древовидной декомпозиции (ДД) [9].

Задача поиска наилучшей ДД состоит в нахождении *триангуляции* графа (триангуляция графа — его вложение в хордальный граф, т.е. в граф, каждый цикл которого длиной ≥ 4 обладает хордой[1]) с минимальным размером наибольшей клики. Актуальность решения этой задачи обусловлена тем, что многие NP-полные задачи разрешимы за полиномиальное время, если они рассматриваются на графах с ограниченной древовидной шириной [15]. К сожалению, как задача поиска триангуляции с минимальным пополнением, так и задача поиска наилучшей ДД являются NP-полными [12, 27]. Однако для обеих задач имеются полиномиально вычислимые альтернативы нахождения так называемой *минимальной триангуляции* [20].

Определение 1. Триангуляция H данного графа G минимальна, если ни одна триангуляция G не является подграфом H.

Для известных алгоритмов LexBFS[2] [23] и MCS[3] [26] разработаны модификации, позволяющие находить минимальные триангуляции: Lex-M [23] и MCS-M [13]. Результаты экспериментального исследования алгоритмов LexBFS, MCS и Lex-M описаны в работе [21].

Выделение древовидной декомпозиции может быть осуществлено и с помощью элиминационной игры [22].

Частным случаем ДД является *блочно-древовидная структура*, введенная в работе [8] и подробно рассматриваемая ниже.

Финкельштейн [6] предложил алгоритм выделения *квазиблочной* структуры в разреженной матрице. Отметим, что алгоритм Финкельштейна тесно связан с *корневыми уровневыми структурами* [11, 19].

При применении локального алгоритма для решения [8] разреженных задач ДО необходимо выделение блочно-древовидной структуры, алгоритм построения которой для разреженных матриц до настоящего времени был неизвестен. Алгоритм Финкельштейна выделения квазиблочных структур в разреженных матрицах также не использовался, авторам неизвестны попытки его программной реализации, исследование качества полученных с его помощью квазиблочных структур не производилось.

Настоящая работа посвящена разработке алгоритма построения блочно-древовидных структур для разреженных матриц, программной реализации алгоритма Финкельштейна и исследованию качества полученных с его помощью квазиблочных структур.

[1]хорда — ребро, соединяющее 2 несмежные вершины цикла

[2]LEX-BFS — лексикографический поиск в ширину

[3]MCS (Maximum Cardinality Search) — поиск по максимальной степени

1. Структура задачи дискретной оптимизации

Рассмотрим задачу ДО с ограничениями

$$F(x_1, x_1, \ldots, x_n) = \sum_{k \in K} f_k(Y^k) \to \max \tag{1}$$

при ограничениях

$$g_i(X^i) \sim 0, \ i \in M = \{1, 2, \ldots, m\}, \tag{2}$$

$$x_j \in D_j, \ j \in N = \{1, 2, \ldots, n\}, \tag{3}$$

где $Y^k \subseteq \{x_1, x_2, \ldots, x_n\}$, $K = \{1, \ldots, t\}$, $X^i \subseteq \{x_1, x_2, \ldots, x_n\}$, $i \in M$; — одно из отношений $\leq$, $\geq$, $=$, D_j — конечное множество возможных значений переменной $x_j \in D_j$, $j \in N = \{1, 2, \ldots, n\}$. Функции $f_k(k \in K)$ называются *компонентами целевой функции.*

Введем необходимые для дальнейшего изложения понятия.

Определение 2. Две переменные x и y *взаимосвязаны* в задаче ДО с ограничениями, если они появляются вместе в одной компоненте ЦФ или в одном и том же ограничении (другими словами, если переменные x, y входят одновременно во множество X^i или во множество Y^k.

Определение 3. *Графом взаимосвязей* задачи ДО называется неориентированный граф $G = (X, E)$, для которого:

- множество вершин X соответствует множеству переменных X задачи ДО;
- две вершины графа соединены ребром тогда и только тогда, когда соответствующие им переменные взаимосвязаны.

Граф взаимосвязей переменных в литературе называется также *графом ограничений* [24].

В дальнейшем будем отождествлять переменные задачи ДО с вершинами графа взаимосвязей. Основным понятием в теории локальных алгоритмов является понятие *близости элементов*, определяемое понятием *окрестности.*

Определение 4. Две вершины x и y в графе $G = (X, E)$ называются *соседними*, если $x, y \in E$.

Определение 5. Множество переменных, соседних с переменной $x \in X$ в графе $G = (X, E)$, обозначается $Nb_G(x)$(или $Nb(x)$) и называется *окрестностью* переменной x.

Пусть $\Omega_1 = (S_1, U_1)$, $\Omega_2 = (S_2, U_2), \ldots,$ $\Omega_k = (S_k, U_k)$ — система окрестностей некоторых переменных $x_{j_1}, \ldots, x_{j_k}$ задачи ДО (1)–(3), где S_r, U_r — соответственно множества индексов переменных и ограничений для r-й окрестности, $r = 1, \ldots, k$, причём

$$\bigcup_{r=1}^{k} U_r = M = \{1, \ldots, m\}, \qquad (4)$$

$$\bigcup_{r=1}^{k} S_r = N = \{1, \ldots, n\}, \qquad (5)$$

$$U_{r_1} \bigcap U_{r_2} = \emptyset, r_1 \neq r_2, \qquad (6)$$

$$S_{r_1} \bigcap S_{r_2} \bigcap S_{r_3} = \varnothing, \qquad (7)$$

для любой тройки индексов r_1, r_2, r_3.

Введем понятие графа инцидентности (или графа пересечений) G_Ω окрестностей. Граф пересечений окрестностей G_Ω имеет вершины $\nu_1, \nu_2, \ldots, \nu_k$, соответствующие выделенным окрестностям Ω_1, $\Omega_2, \ldots,$ Ω_k, причем две вершины графа ν_{r_1} и ν_{r_2}, отвечающие окрестностям Ω_{r_1} и Ω_{r_2}, соединены ребром $(r_1,\ r_2)$, если $S_{r_1} \bigcap S_{r_2} \neq \varnothing$.

Определение 6. *Блочно-древовидной* (БД) назовем задачу ДО[4], для которой можно выделить систему окрестностей Ω_1, $\Omega_2, \ldots,$ Ω_k, удовлетворяющую свойствам (4)–(7), причем граф пересечений G_Ω окрестностей является деревом.

Замечание. Частным случаем БД структуры является квазиблочная структура [7], для которой граф пересечений окрестностей (или блоков) является цепью. Отметим, что блочно-древовидная структура является частным случаем древовидной декомпозиции.

1.1. **Гиперграф и двойственный граф.** Введем гиперграфовое представление задачи ДО, которое является наиболее адекватным представлением структуры задачи ДО. В гиперграфовом представлении множество вершин гиперграфа H соответствует множеству переменных X задачи ДО, а гиперребра гиперграфа образованы подмножествами взаимосвязанных переменных.

Часто вместо гиперграфового представления используется более наглядное представление структуры задачи ДО с помощью *двойственного* и *первичного* графов. Первичный граф — это описанный выше *граф взаимосвязей* задачи ДО.

[4]Будем говорить, что описанная задача ДО имеет блочно-древовидную структуру.

Определение 7. *Двойственным графом* гиперграфа $H = (V, S)$ называется граф, вершины которого соответствуют гиперребрам гиперграфа, причем пара таких вершин соединяется ребром в двойственном графе, если они имеют общие вершины из V.

2. Алгоритм Финкельштейна выделения квазиблочной структуры

Финкельштейн [6] предложил следующий алгоритм выделения квазиблочной структуры в разреженной матрице. Пусть $H_{m\times n}$ — разреженная матрица, p — индекс некоторой переменной $x_p, p \in \{1, 2, \dots, n\}$. Найдем окрестности переменной x_p последовательно возрастающих порядков, т.е.

$$S_0' = \{x_p\},\, U_{r+1}' = U(S_r'(x_p)), \quad r \geq 0; \tag{8}$$

$$S_{r+1}' = S(U_r'(p)), \quad r \geq 1 \tag{9}$$

причем процесс выделения окрестностей продолжаем до тех пор, пока при некотором $r = k$ не выполнится условие $S_{k+1}'(x_p) = S_k'(x_p)$. Далее находится разбиение множества индексов ограничений, входящих в блоки:

$$U_r = \begin{cases} U_1'(x_p), & r = 1, \\ U_r'(x_p) \backslash U_{r-1}'(x_p), & 1 < r < k, \\ U_k'(x_p) \bigcup U_{k+1}'(x_p) \backslash U_{k-1}'(x_p), & r = k, \end{cases}$$

и находятся следующие множества переменных:

$$S_{r,r+1} = S(U_r(x_p)) \bigcap S(U_{r+1}(x_p)),$$

$$S_r = \begin{cases} S\left(U_1\left(x_p\right)\right) \backslash S_{12}, & r = 1, \\ S\left(U_1\left(x_p\right)\right) \backslash \left(S_{r-1,r} \cup S_{r,r+1}\right), & 1 < r < k, \\ S\left(U_k\left(x_p\right)\right) \backslash S_{k-1,k}, & r = k. \end{cases}$$

Описанный алгоритм находит n разбиений матрицы H на квазиблоки (не обязательно различных), хотя возможных разбиений существует гораздо больше.

Стоит отметить, что в рамках данного исследования алгоритм Финкельштейна был усовершенствован. Так, если находились вырожденные блоки[5], то они сливались вместе, т.е. вырожденный блок становился частью впереди идущего блока.

[5]Блок B_r считается вырожденным, если $S_r = \varnothing$.

Также был установлен максимальный размер сепаратора. При превышении допустимого размера сепаратора в блоке, выполнялась вышеупомянутая процедура слияния блоков. Предусмотрена также возможная несвязность графа ограничений.

3. Метод построения блочно-древовидной структуры

Как показано в [8], переход от блочно-древовидной структуры к квазиблочной путем объединения «слоев» дерева нецелесообразен. В связи с этим представляет интерес выделение блочно-древовидной структуры в разреженном графе с помощью модификации описанного выше алгоритма Финкельштейна.

Структура задачи ДО, как отмечено выше, может быть описана с помощью *двойственного* (или *реберного* [3]) графа. Полученная с помощью алгоритма Финкельштейна (см. раздел 2) квазиблочная структура $\{U_1, \ldots, U_k\}$ задачи ДО на двойственном графе D имеет вид цепи. Далее рассмотрим граф $D_s = D\left(\bigcup_{r=s}^{k} U_r\right)$, индуцируемый множеством $\bigcup_{r=s}^{k} U_r$ в двойственном графе D $(s = 2, \ldots, k)$ и выделим в нем компоненты связности $C_1, \ldots, C_p$. Далее каждый блок U_s заменяется на подблоки $U_s \bigcap C_1, \ldots, U_s \bigcap C_p$. В результате получается блочно-древовидная структура.

Рассмотрим алгоритм выделения компонент связности [17] в неориентированном графе согласно [4]. В основу рассматриваемого алгоритма выделения компонент связности положен предложенный R. Tarjan [25] алгоритм поиска в глубину на графе $G(X, E)$. Алгоритм находит вектор $Mark$[x] меток вершин $x \in X$ графа. Элементу $Mark$[x] присваивается общий номер той компоненты, которой принадлежит вершина $x \in X$. Сложность алгоритма, как и алгоритма поиска в глубину, составляет $O(|X| + |E|)$.

Пример 1. Рассмотрим построение блочно-древовидной структуры для задачи ДО, состоящей из следующих ограничений:

$$
\begin{aligned}
B_1 &: x_4 + x_5 + 4x_6 + 2x_7 \le 4,\\
B_2 &: 3x_1 + 2x_2 + 2x_3 + 3x_4 \le 6,\\
B_3 &: 3x_7 + 8x_8 + 3x_9 + x_{10} \le 8,\\
B_4 &: x_9 + 2x_{11} + 3x_{12} \le 4,\\
B_5 &: x_{10} + x_{13} + 2x_{14} \le 2,\\
&\quad x_j = 0, 1, \quad j = 1, 2, \ldots, 14.
\end{aligned}
$$

Построим двойственный граф этой задачи ДО и выделим квазиблочную структуру $\{U_1, U_2, U_3\}$ с помощью алгоритма Финкельштейна (рис. 2).

```
Цикл пока v ∈ X
    Mark[v] = 0;    // Начальная установка
end Цикл пока
count = 0;    // Счетчик числа компонент
Цикл пока v ∈ X
    Если Mark[v] = 0 то
        count = count + 1;
        Component(v, count);
    end Если
end Цикл пока
procedure COMPONENT(x, count)
    Mark[x] = count;
    Цикл пока v ∈ Nb(x)
        Если Mark[v] = 0 то
            Component(v, count);
        end Если
    end Цикл пока
end procedure
```

Рис. 1. Алгоритм выделения связных компонент неориентированного графа.

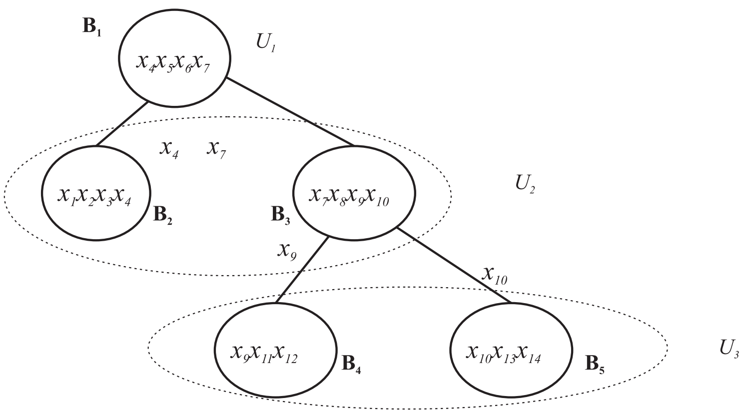


Рис. 2. Выделение квазиблочной структуры $\{U_1, U_2, U_3\}$ с помощью алгоритма Финкельштейна

Найдем компоненты связности с помощью описанного выше алгоритма в графе $D_2 = D(U_2 \bigcap U_3)$, индуцируемом множеством $U_2 \bigcap U_3$, в двойственном графе D (рис. 3). Далее выделим компоненты связности C_1' и C_2' в графе $D_3 = D(U_3)$, индуцируемом множеством U_3 в двойственном графе D (рис. 4).

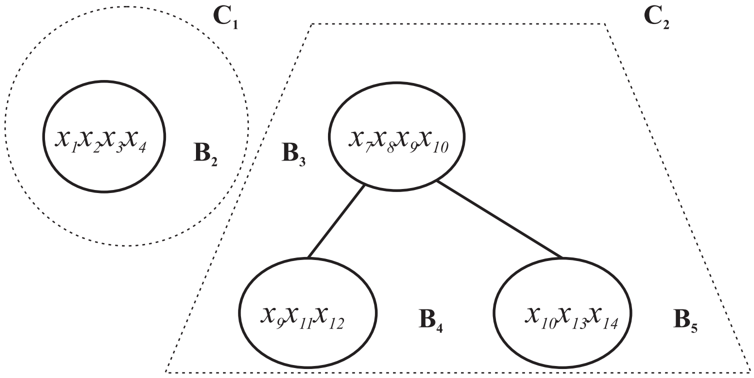


Рис. 3. Выделение компонент связности C_1 и C_2 в графе $D_2 = D(U_2 \bigcup U_3)$, индуцируемом множеством $U_2 \bigcup U_3$ в двойственном графе D.

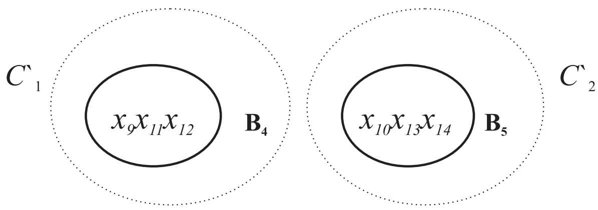


Рис. 4. Выделение компонент связности C_1' и C_2' в графе $D_3 = D(U_3)$, индуцируемом множеством U_3 в двойственном графе D.

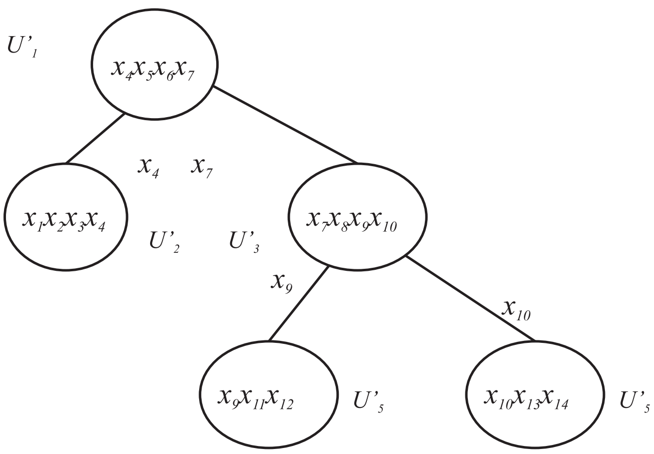


Рис. 5. Блочно-древовидная структура $\{U_1', U_2', U_3', U_4', U_5'\}$.

Заменяя каждый блок U_s на подблоки $U_s \bigcap C_1, \ldots, U_s \bigcap C_p$ для каждого разбиения на связные компоненты, получим в результате блочно-древовидную структуру (рис. 5).

4. Описание вычислительного эксперимента

Алгоритм Финкельштейна реализован с помощью класса FinkelsteinQBDecomposition на языке программирования C++ и является частью библиотеки LES[6].

Все вычисления проводились на базе процессора Intel Core 2 Duo @2.66 GHz, 2 Gb ОЗУ и операционной системы Linux, версия ядра 2.6.35-24-generic.

Для проведения вычислительного эксперимента были построены пять групп тестовых задач 'dubois', 'bridge', 'adder', 'pret' и 'grid' для 17 гиперграфов из библиотеки CSP[7] [18], причем каждая тестовая задача решалась с помощью исходного и модифицированного алгоритмов Финкельштейна. Результаты вычислительного эксперимента приведены в таблице 1.

Таблица 1. Результаты декомпозиции задач ДО с помощью исходного и модифицированного алгоритма Финкельштейна.

Задача	Исходный алгоритм		Модифицированный алгоритм	
	Кол-во блоков	Максимальный сепаратор	Кол-во блоков	Максимальный сепаратор
adder_15	31	4	16	4
adder_50	61	6	51	6
adder_99	121	6	101	6
dubois23	23	2	23	2
dubois30	30	2	30	2
dubois50	50	2	50	2
dubois100	98	6	2	4
pret60_25	8	10	7	10
pret60_60	8	10	7	10
pret150_25	15	12	11	10
pret150_75	15	12	11	10
grid2d_10	8	10	2	3
grid10	16	10	3	9
grid3d_4	5	12	2	6
bridge_15	21	10	4	10
bridge_50	56	10	5	8
bridge_75	78	9	4	9

[6]Local Elimination Solver https://github.com/d2rk/les/

[7]CSP (Constraint Satisfaction Problem) — задача удовлетворения ограничений.

Следует отметить, что в качестве первой переменной, используемой алгоритмом Финкельштейна, для которой находилась окрестность, выбиралась x_1, т.е. $p = \{1\}$.

Анализируя результаты, полученные в таблице 1, нетрудно увидеть преимущества модифицированного алгоритма Финкельштейна по сравнению с исходным алгоритмом Финкельштейна. Так, во многих задачах были существенно уменьшены количество построенных блоков и размеры сепараторов.

Рассмотрим некоторые результаты для тестовых задач более подробно. Можно отметить задачу 'dubois100', в которой существенно сократилось количество построенных блоков и максимальный сепаратор уменьшился на 2. Для задач 'pret150_25' и 'pret150_75' существенно уменьшилось количество построенных блоков.

Заключение

В статье предложен алгоритм выделения блочно-древовидной структуры для разреженных матриц. Реализован в виде программы на C++ и протестирован алгоритм Финкельштейна для выделения квазиблочных структур в разрежённых матрицах. В качестве тестовых задач взяты пять групп задач: 'dubois','bridge','adder','pret' и 'grid'. Произведен сравнительный эксперимент для модифицированной и исходной версий алгоритма, показавший существенное уменьшение количества построенных блоков и размеров сепараторов для модифицированного алгоритма Финкельштейна. В дальнейшем планируется проведение сравнительного эксперимента для алгоритмов нахождения блочно-древовидной структуры и древовидной декомпозиции.

Список літератури

1. Гловер Ф. Целочисленное программирование и комбинаторика / Ф. Гловер // Исследование операций. Т.1: Методологические основы и математические методы. — М.: Мир, 1981. — С. 122–182.
2. Журавлев Ю. И. Избранные научные труды / Ю. И. Журавлёв. — М.: Магистр, 1998. — 420 с.
3. Лекции по теории графов / [Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И.] / Изд. 2-е, испр. — М.: Книжный дом «ЛИБРОКОМ», 2009. — 392 с.
4. Иванов Б. Н. Дискретная математика. Алгоритмы и программы: Учеб. пособие / Б. Н. Иванов. — М.: Лаборатория Базовых Знаний, 2003. — 288 с.
5. Писсанецки С. Технология разреженных матриц / С. Писсанецки. — М: Мир, 1988. — 356 с.
6. Финкельштейн Ю. Ю. Методы решения некоторых дискретных задач математического программирования: Дисс. … канд. физ.-матем. наук / Ю. Ю. Финкельштейн. — М., 1966. — 110 с.

7. Щербина О. А. О локальных алгоритмах решения квазиблочных задач дискретного программирования / О. А. Щербина // Проблемы кибернетики. — М.: Наука, 1983. — Вып. 40. — С. 171–200.
8. Щербина О. А. Локальные алгоритмы для блочно-древовидных задач дискретного программирования / О. А. Щербина // Журн. вычисл. математики и матем. физики. — 1985. — Т. 25, N 8. — С. 1143–1154.
9. Щербина О. А. Древовидная декомпозиция и задачи дискретной оптимизации (обзор) / О. А. Щербина // Кибернетика и системный анализ. — 2007. — № 4. — С. 102–118.
10. Щербина О. А. Локальные элиминационные алгоритмы решения разреженных дискретных задач / О. А. Щербина // Журнал вычислительной математики и математической физики. — 2008. — Т. 48, N 1. — С. 161—177.
11. Arany I. An improved method for reducing the bandwidth of sparse symmetric matrices / I. Arany, W. F. Smyth, L. Szoda // Proc. of IFIP Congress "Information Processing 71", North-Holland, Amsterdam, 1972. — P. 1246–1250.
12. Arnborg S. Complexity of finding embeddings in a k-tree / S. Arnborg, D. G. Corneil, A. Proskurowski // SIAM J. Alg. Disc. Meth. — 1987. — V. 8. — P. 277–284.
13. Maximum cardinality search for computing minimal triangulations of graphs / [A. Berry, J. Blair, P. Heggernes, B. Peyton] // Algorithmica. — 2004. — V. 39. — P. 287–298.
14. Bertele U. Nonserial Dynamic Programming / U. Bertele, F. Brioschi. — New York: Academic Press, 1972. — 235 p.
15. Bodlaender H. L. A tourist guide through treewidth / H. L. Bodlaender // Acta Cybernetica. — 1993. — V. 11. — P. 1-21.
16. Dechter R. Constraint Processing / R. Dechter. — Morgan Kaufmann, 2003. — 481 p.
17. Even S. Graph Algorithms / S. Even. — Rockville: Computer Science Press, 1983. — 249 p.
18. A CSP hypergraph library: Technical Report / [T. Ganzow, G. Gottlob, N. Musliu, M. Samer]. — Vienna: Technische Universität Wien, 2005. — 10 p. — DBAI-TR-2005-50.
19. George J. A. Computer solution of large sparse positive definite systems / J. A. George, J. W. H. Liu. — Englewood Cliffs: Prentice-Hall Inc., 1981. — 324 p.
20. Heggernes P. Minimal triangulations of graphs: A survey / P. Heggernes // Discrete Mathematics. — 2006. — V. 306, № 3. — P. 297-317.
21. Koster A. M. C. A. Treewidth: Computational experiments / A. M. C. A. Koster, H. L. Bodlaender, S. P. M. van Hoesel // Electronic Notes in Discrete Mathematics. — 2001. — V. 8. — P. 54–57.
22. Parter S. The use of linear graphs in Gauss elimination / S. Parter // SIAM Review. — 1961. — V.3. — P. 119–130.
23. Rose D. Algorithmic aspects of vertex elimination on graphs / D. Rose, R. E. Tarjan, G. Lueker // SIAM J. Comput. — 1976. — V. 5. — P. 266–283.
24. Shcherbina O. Nonserial dynamic programming and tree decomposition in discrete optimization / O. Shcherbina // Proceedings of Int. Conference on Operations Research "Operations Research 2006" (Karlsruhe, 6-8 September, 2006). — Berlin: Springer Verlag, 2007. — P. 155–160.

25. Tarjan R. E. Depth first search and linear graph algorithms / R. E. Tarjan // SIAM J. Comput. — 1972. — V. 1. — P. 146–160.
26. Tarjan R.E. Simple linear-time algorithms to test chordality of graphs, test acyclity of hypergraphs, and selectively reduce acyclic hypergraphs / R.E. Tarjan, M. Yannakakis // SIAM J. Comput. — 1984. — V. 13. — P. 566–579.
27. Yannakakis M. Computing the minimum fill-in is NP-complete / M. Yannakakis // SIAM J. Alg. Disc. Meth. — 1981. — V. 2. — P. 77—79.

Статья поступила в редакцию 02.06.2012

УДК 519.658

О РАСПАРАЛЛЕЛИВАНИИ ЛОКАЛЬНОГО ЭЛИМИНАЦИОННОГО АЛГОРИТМА

© Д. В. Лемтюжникова, О. А. Щербина

Таврический национальный университет им. В.И. Вернадского
факультет математики и информатики
пр-т Вернадского, 4, г. Симферополь, 95007, Украина
e-mail: *darabbt@gmail.com, oshcherbina@gmail.com*

Abstract. We introduce strategies for parallelizing a sequential local elimination algorithm for sparse discrete optimization problems. The local elimination procedure exploits the structure of the underlying problem graph using extended elimination tree. We propose to use hybrid Master-Worker scheme where worker processors (GPUs) solve concurrently subproblems corresponding to super-nodes of extended elimination tree that are generated by a single master process (CPU). Subproblem generation is embedded into an local elimination algorithm and takes previous subproblem solutions into account. The current state of parallel architectures and parallelization technics is discussed.

Введение

Задачи дискретной оптимизации (ДО) с блочной структурой возникают естественным образом во многих приложениях. При этом, задачи ДО являются NP-трудными, поэтому при больших размерах их решение затруднительно в связи с перебором экспоненциального числа вариантов.

Перспективными методами структурной декомпозиции, использующими разреженность матрицы ограничений задач ДО, представляются локальные элиминационные алгоритмы [5], включающие локальные алгоритмы декомпозиции [1], [2, 4, 3], алгоритмы несериального динамического программирования (НСДП) [8, 17, 25], алгоритмы сегментной элиминации [10]. Высокая вычислительная сложность локального алгоритма при больших сепараторах между блоками [5] может быть снижена с помощью параллельных вычислений.

1. Параллельные алгоритмы для задач дискретной оптимизации

Для параллельной реализации алгоритмов ДО могут использоваться системы с распределенной памятью: их параллельное программное обеспечение использует декомпозицию решаемой задачи на подзадачи и назначение их процессорам, а также быструю коммуникационную сеть, обеспечивающую синхронный обмен данными между процессорами. Программирование такой системы может осуществляться с помощью стандартного языка высокого уровня типа C++ или Fortran с явной передачей сообщений. Также может быть использована параллельная система ПО для решения задач смешанного целочисленного линейного программирования (ЦЛП) — SYMPHONY [24], подобная COIN/BCP [23]. COIN/BCP и SYMPHONY объединены в новом решателе ALPS [22].

Ранее при разработке параллельных алгоритмов использовались обычно суперкомпьютеры и кластеры рабочих станций.

Остановимся на новых возможностях применения современных параллельных вычислительных архитектур для ускорения работы алгоритмов ДО, таких как графические процессоры (GPU) и GRID.

В последнее десятилетие наблюдается повышенный интерес к GRID-вычислениям [11]. GRID-вычисления используют совокупность слабосвязанных разнородных вычислительных ресурсов как единый вычислительный ресурс, которым может пользоваться большое число пользователей.

В течение последних нескольких лет все более популярным в научном сообществе становится использование графических процессоров GPU[1]. Графические процессоры GPU, обычно предназначенные для компьютерной графики, могут использоваться для выполнения расчётов в приложениях, которые обычно проводятся центральным процессором CPU [20]. Среди программных продуктов, которые позволяют программировать для GPU независимо от платформы, можно упомянуть следующие: CUDA[2], DirectCompute[3] от Microsoft, OpenGL Shading Language (GLSL)[4].

Khronos Group разработала специализированный язык OpenCL[5], который является основой для написания программ, выполняемых на разных платформах, состоящих из CPU и GPU. OpenCL является открытым стандартом, который может быть использован для программирования CPU, GPU и других устройств различных производителей, в то время как CUDA специализирована для NVIDIA GPU. OpenCL обеспечивает переносимость различных аппаратных GPU, ОС, программного обеспечения, а также поддерживается многоядерными процессорами. Поэтому представляется перспективным использование именно OpenCL для реализации алгоритмов структурной декомпозиции разреженных задач ДО.

Существенное ускорение параллельных вычислений можно получить с помощью программируемых графических процессоров (GPU), которые могут обрабатывать

[1]GPU — Graphics Processing Unit.

[2]CUDA = Compute Unified Device Architecture.

[3]DirectCompute специализирован исключительно под Microsoft Windows.

[4]GLSL не перспективен с научной точки зрения.

[5]OpenCL (Open Computing Language) — открытый язык вычислений, см. OpenCL: открытый стандарт для параллельного программирования гетерогенных систем. http://www.khronos.org/opencl.

тысячи потоков данных. Среди публикаций, посвященных параллельным алгоритмам ДО на базе GPU, отметим следующие работы. Обзор [21], посвященный исследованиям параллельных алгоритмов муравьиных колоний (АМК)[6] содержит анализ 106 научных публикаций. Можно отметить, что для параллельной реализации АМК часто используется парадигма «Мастер-рабочий»[7] в основном из-за концептуальной простоты и легкости в применении. В [15, 16] рассмотрены особенности параллельной реализации алгоритмов локального поиска и эволюционных алгоритмов на базе GPU. Boyer и соавторы [9] предложили параллельную реализацию метода динамического программирования для задачи о ранце с помощью CUDA для системы из нескольких GPU. Fujimoto и Tsutsui [12] разработали параллельный решатель задачи о коммивояжере для GPU. Gulati и Khatri [13] внедрили новый подход упорядочивания переменных в процедуре решения задачи выполнимости на GPU. Beckers и соавторы [7] предлагают параллельный решатель задачи выполнимости SAT, реализованный на языке OpenCL на GPU.

В [14] описывается распараллеливание метода ветвей и границ (ВГ). Последовательный алгоритм ВГ заключается в неявном переборе подмножеств допустимых решений, с помощью дерева поиска ВГ, вершины которого являются множествами решений рассматриваемой задачи. Размер дерева, то есть количество создаваемых вершин, непосредственно связан с методом, используемым для его построения. Для алгоритма ВГ известны два вида стратегий распараллеливания:

1. Вершинные стратегии, цель которых — ускорить вычисления на уровне вершины дерева поиска, например, параллельное вычисление нижней или верхней границы, параллельная оценка вершин-потомков и т.д.
2. Древовидные стратегии, цель которых — параллельное построение и исследование дерева поиска ветвей и границ.

При распараллеливании алгоритма ВГ можно отметить следующие сложности: структура дерева поиска и граф зависимостей между задачами заранее неизвестны, задачи для процессоров создаются динамически в процессе работы алгоритма. Для того, чтобы избежать дополнительных временных затрат, необходимо принимать во внимание такие алгоритмические аспекты, как балансировка нагрузки и передача сообщений между процессорами.

Хотелось бы отметить работы [6] и [18], [19], посвященные вопросам разработки параллельных версий алгоритмов структурной декомпозиции для дискретных задач.

[6] АМК является метаэвристикой для решения задач оптимизации.

[7] Master-Worker.

Использование ЛЭА при решении задач ДО с большими перемычками (сепараторами) затруднительно в связи с большим объёмом вычислений. Поэтому разработка параллельной версии локального блочно-элиминационного алгоритма для задач ДО представляет значительный интерес.

2. Блочно-элиминационный локальный алгоритм

2.1. Формулировка локального блочно-элиминационного алгоритма. Рассмотрим задачу ЦЛП с бинарными переменными:

$$CX = \sum_{j=1}^{n} c_j x_j \to \max \tag{1}$$

при ограничениях

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1,\ 2,\ \ldots,\ m, \tag{2}$$

$$x_j = 0,\ 1, \quad j = 1,\ 2,\ \ldots,\ n. \tag{3}$$

Определение 1. Множество переменных, соседних с переменной $x \in X$ в графе $G = (X, E)$, обозначается $Nb_G(x)$ (или $Nb(x)$) и называется *окрестностью* переменной x. *Замкнутой окрестностью* переменной $x \in X$ в графе $G = (X, E)$ называется множество: $Nb[x] = Nb(x) \cup \{x\}$.

Определение 2. *Монотонной* окрестностью вершины x_i называется множество вершин, соседних с x_i, с индексами, большими, чем i (согласно упорядочению α):

$$\overline{Nb}^{\alpha}_{G}(x_i) = \{x_j \in Nb_G(x_i) | j > i\}.$$

Процедура элиминации может быть применена для элиминации не только отдельных переменных, но и множеств переменных в виде «блочной элиминации переменных» [2, 8], которая позволяет элиминировать несколько переменных «блоком». При построении блочных переменных целесообразно использовать понятие *«супервершины»*, объединяющей в себе так называемые «неразличимые вершины»[8]. Если задача ДО разбита на блоки, соответствующие подмножествам переменных (называемых супер-переменными), то полученная блочная структура может быть описана с помощью структурного конденсированного графа (фактор-графа), супер-вершины которого соответствуют подмножествам переменных (или блокам) исходного графа и супер-ребра соответствуют соседним блокам. Рассмотрим процедуру локальной

[8]В графе G две смежные вершины u и v называются *неразличимыми*, если $Nb[u] = Nb[v]$.

блочной элиминации [8], в которой элиминация блока (т.е. подмножества переменных) может рассматриваться как элиминация супер-переменной. Рассмотрим упорядоченное разбиение $\mathbf{X}$ множества X переменных на блоки:

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_p\}, \ p \leq n,$$

где $\mathbf{x}_l = X_{K_l}$ (K_l — множество индексов, соответствующих $\mathbf{x}_l$, $l = 1, \cdots, p$). Для этого упорядоченного разбиения $\mathbf{X}$, задача ДО P (1.1)–(1.3) может быть решена с помощью ЛЭА, используя *фактор-граф взаимосвязей* $\mathbf{G}$.

При этом $\alpha : \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_p\}$ является порядком элиминации супер-вершин (или блоков) и представляет собой аналог порядка элиминации переменных.

Определение 3. Фактор-граф $\mathbf{G} = (\mathbf{X}, \mathbf{E})$ — это структурный граф супер-задачи ДО, получаемый с помощью представления множества «сжатых» вершин в виде одной супер-вершины и соединения каждой супер-вершины со всеми супер-вершинами, вершины которых были смежны некоторым из сжатых вершин, составляющих рассматриваемую супер-вершину.

Определение 4. Множество супер-переменных, соседних с супер-переменной $\mathbf{x} \in \mathbf{X}$ в фактор-графе $\mathbf{G} = (\mathbf{X}, \mathbf{E})$, обозначается $Nb_{\mathbf{G}}(\mathbf{x})$ (или $Nb(\mathbf{x})$) и называется *окрестностью* супер-переменной $\mathbf{x}$. *Замкнутой окрестностью* супер-переменной $\mathbf{x} \in \mathbf{X}$ в фактор-графе $\mathbf{G} = (\mathbf{X}, \mathbf{E})$ называется множество: $Nb[\mathbf{x}] = Nb(\mathbf{x}) \cup \{\mathbf{x}\}$.

Перейдем к формулировке блочного ЛЭА, состоящего из двух частей.

A. Прямая часть

Рассмотрим вначале блок $\mathbf{x}_1$. Тогда

$$\max_X \{C_N \cdot X_N \mid A_{iS_j} X_{S_j} \leq b_i, i \in M, x_j = \{0,1\}, j \in N\} =$$
$$= \max_{X_{K_2}, \cdots, X_{K_p}} \{C_{N-K_1} \cdot X_{N-K_1} + h_1(Nb(X_{K_1})) \mid A_{iS_j} X_{S_j} \leq b_i,$$
$$i \in M - U_1, x_j = \{0,1\}, j \in N - K_1\}$$

где $U_1 = \{i : S_i \cap K_1 \neq \varnothing\}$ и

$$h_1(Nb(X_{K_1})) = \max_{X_{K_2}, \cdots, X_{K_p}} \{C_{K_1} \cdot X_{K_1} \mid A_{iS_j} X_{S_j} \leq b_i,$$
$$i \in U_1, x_j = \{0,1\}, j \in Nb[x_1]\}$$

Первый шаг локальной блочной элиминационной процедуры состоит в решении, используя полный перебор значений X_{K_1}, следующей задачи оптимизации

$$h_1(Nb(X_{K_1})) = \max_{X_{K_2}, \cdots, X_{K_p}} \{C_{K_1} \cdot X_{K_1} \mid A_{iS_j} X_{S_j} \leq b_i,$$
$$i \in U_1, x_j = \{0,1\}, j \in Nb[x_1]\}$$

и запоминании оптимальных локальных решений X_{K_1} в виде функций окрестностей X_{K_1}, т.е. $X^*_{K_1}Nb(X_{K_1})$. Максимизация $f(X)$ по всем возможным присвоениям $Nb(X_{K_1})$ называется *элиминацией блока* (или элиминацией супер-переменной) X_{K_1}. После элиминации необходимо решить задачу оптимизации:

$$\max_{X-X_{K_1}}\{C_{N-K_1}\cdot X_{N-K_1}+h_1(Nb(X_{K_1}))\mid A_{iS_j}X_{S_j}\le b_i,$$
$$i\in M-U_1, x_j=\{0,1\}, j\in N-K_1\}$$

Заметим, что эта задача имеет тот же вид, что и исходная задача, а табличная функция $h_1(Nb(X_{K_1}))$ может быть рассмотрена как новый компонент измененной целевой функции. Следовательно, та же процедура может быть использована для поочередной элиминации блоков – супер-переменных $\mathbf{x}_2 = X_{K_2},\cdots,\mathbf{x}_p = X_{K_p}$. На каждом шаге j запоминаются новый компонент $h_{\mathbf{x}_j}$ и оптимальные локальные решения $X^*_{K_j}$ в виде функций от $Nb(X_{K_j}\mid X_{K_1},\cdots,X_{K_{j-1}})$ т.е. от множества переменных, взаимосвязанных по крайней мере с одной переменной из X_{K_j} в текущей задаче, полученной из исходной задачи с помощью элиминации $X_{K_1},\cdots,X_{K_{j-1}}$. Так как множество $Nb(X_{K_p}\mid X_{K_1},\cdots,X_{K_{p-1}})$ пустое, элиминация X_{K_p} позволит получить оптимальное значение целевой функции $f(X)$.

B. Обратная часть

Эта часть процедуры состоит в последовательном выборе $X^*_{K_p}, X^*_{K_{p-1}},\cdots,X^*_{K_1}$ — оптимальных локальных решений из сохраненных в прямой части таблиц $X^*_{K_1}(Nb(X_{K_1}))$, $X^*_{K_2}(Nb(X_{K_2}\mid X_{K_1}))$, $\cdots$, $X^*_{K_p}(Nb(X_{K_p}\mid X_{K_1},\cdots,X_{K_{p-1}}))$.

Графовой интерпретацией блочного ЛЭА может служить *обобщенная элиминационная игра*, входом которой является фактор-граф взаимосвязей $\mathbf{G}$ и обобщенный порядок элиминации (упорядоченное разбиение) $\alpha:\{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_p\}$ супер-вершин графа $\mathbf{G}$. На каждом шаге $\nu(1\le\nu\le p)$ обобщенной элиминационной игры окрестность $Nb(\mathbf{x}_\nu)$ супер-вершины $\mathbf{x}_\nu$ преобразуется в клику, а $\mathbf{x}_\nu$ удаляется из графа $\mathbf{G}$ вместе со смежными ребрами.

Определение 5. Граф $\mathbf{G}^+_{\mathbf{X}}=(\mathbf{X},\mathbf{E}^+)$, образованный путем добавления в $\mathbf{G}$ всех ребер, добавленных алгоритмом, называется *обобщённым пополненным графом.*

Понятие монотонной окрестности, введенное выше, может быть обобщено на случай супер-переменной в фактор-графе.

Определение 6. *Монотонной* окрестностью супер-вершины $\mathbf{x}_i$ в фактор-графе $\mathbf{G}=(\mathbf{X},\mathbf{E})$ называется множество супер-вершин, соседних с $\mathbf{x}_i$ в фактор-графе $\mathbf{G}$, с индексами, большими, чем i (согласно упорядочению $\alpha:\{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_p\}$):

$$\overline{Nb}^\alpha_{\mathbf{G}}(\mathbf{x}_i)=\{\mathbf{x}_j\in Nb_{\mathbf{G}}(\mathbf{x}_i)|j>i\}.$$

Бесконтурный орграф вычислительной процедуры локального блочного элиминационного алгоритма содержит вершины, соответствующие вычислению функций $h_{\mathbf{x}_j}(Nb_{G_{\mathbf{X}}^{(j-1)}}(\mathbf{x}_j))$ и является *обобщенным элиминационным деревом.*

Определение 7. Обобщённым элиминационным деревом (ОЭД) графа $\mathbf{G}$ для упорядоченного разбиения множества вершин на подмножества (супер-вершины) называется ориентированное дерево $\vec{\mathbf{T}}_{\mathbf{X}}$, имеющее то же множество супер-вершин $\mathbf{X}$, что и фактор-граф $\mathbf{G}$, а множество дуг ОЭД определяется с помощью отношения «отец–сын» следующим образом: отцом супер-вершины $\mathbf{x}$ является первая супер-вершина из монотонной окрестности $\overline{Nb}_{\mathbf{G}_{\mathbf{X}}^{+}}(\mathbf{x})$ супер-вершины в пополненном графе $\mathbf{G}_{\mathbf{X}}^{+}$.

3. Источники распараллеливания

Для ЛЭА структура элиминационного дерева известна заранее и граф зависимостей между задачами предсказуем, что позволяет избежать ряда описанных в разделе 1 проблем, свойственных параллельным методам ВГ.

Для ЛЭА при решении задачи (1)–(3) существуют следующие источники распараллеливания:

1) *Параллельная обработка независимых ветвей обобщенного элиминационного дерева.*

Используемый граф потоков данных для блочного ЛЭА задачи является обобщенным элиминационным деревом, которое должно обрабатываться от листьев к корню. При параллельной реализации блочного ЛЭА супер-вершины каждого слоя ОЭД могут быть рассмотрены и обработаны параллельно. При расчетах множество промежуточных данных передается в родительскую вершину из вершин-сыновей. Как только промежуточная информация от всех вершин-сыновей собрана в родительской вершине, она используется при решении соответствующей родительской вершине задачи ДО. Таким образом, независимые ветви дерева могут обрабатываться параллельно. Распараллеливание ЛЭА более эффективно в нижней части дерева, нежели вблизи ее корневой вершины.

2) *Параллельное решение задач в блоках.*

Для каждого блока (супер-вершины) можно рассмотреть множество независимых задач[9], которые порождаются в прямой части ЛЭА. Там для каждого значения сепаратора $\mathbf{X}_{Nb_{G_{\mathbf{X}}^{(j-1)}}(\mathbf{x}_j)}$ нужно решить параллельно задачи $h_{\mathbf{x}_j}(Nb_{G_{\mathbf{X}}^{(j-1)}}(\mathbf{x}_j))$.

[9]Независимость задач состоит в том, что информация, полученная в результате решения одной задачи, не используется при решении другой.

Так как структура графа потоков данных известна (ОЭД)[10], такие задачи могут быть решены при помощи парадигмы «Master-Worker», в которой имеются два различных типа процессоров: мастер-процессор и рабочие процессоры. При этом управление алгоритмом осуществляется мастер-процессором, который разбивает задачу на отдельные подзадачи и назначает их для решения рабочим процессорам, учитывая взаимосвязи между ними. Мастер-процессор выбирается в начале решения пакета задач, соответствующих блоку ОЭД. Далее, рабочие процессоры выбираются мастером динамически из списка процессоров-кандидатов, используя критерий хорошей балансировки нагрузки.

Для параллельной реализации ЛЭА мы предлагаем использовать гибридную схему «мастер-рабочий», которая допускает одновременное использование CPU и GPU. GPU, множество основных параллельных машин с общей памятью, выступают в качестве рабочих процессоров и выполняют решение подзадач ДО для блоков. Синтез решений выполняется процессором CPU, который выступает в роли мастера.

Заключение

В работе выделены стратегии распараллеливания локального элиминационного алгоритма для разреженных задач дискретной оптимизации на основе использования современных вычислительных архитектур. Независимые подзадачи, соответствующие разным блокам, и подзадачи, соответствующие независимым ветвям обобщенного элиминационного дерева, могут решаться параллельно при помощи таких вычислительных архитектур, как многоядерные процессоры, графические процессоры (GPU) и GRID. Для параллельной реализации ЛЭА предлагается использовать гибридную схему «мастер-рабочий», которая допускает одновременное использование CPU и GPU, причем GPU, множество основных параллельных машин с общей памятью, выступают в качестве рабочих процессоров и выполняют решение подзадач ДО для блоков, а CPU используется в качестве мастера. В дальнейшем планируется реализовать эту схему при помощи виртуального программирования, что позволит оценить эффективность распараллеливания локального элиминационного алгоритма и определить дальнейший ход исследования.

Список літератури

1. Журавлев Ю. И. Избранные научные труды / Ю. И. Журавлев. — М.: Магистр, 1998. — 420 с.
2. Щербина О. А. О локальных алгоритмах решения квазиблочных задач дискретного программирования / О. А. Щербина // Проблемы кибернетики. — М.: Наука, 1983. — Вып. 40. — С. 171–200.

[10] В качестве графа потоков данных может использоваться и древовидная декомпозиция.

3. Щербина О. А. О несериальной модификации локального алгоритма декомпозиции задач дискретной оптимизации / О. А. Щербина // Динамические системы. — 2005. — Вып. 19. — С. 179–190.
4. Щербина О. А. О решении квазиблочных задач целочисленного линейного программирования с помощью локального алгоритма / О. А. Щербина // Кибернетика. — 1984. — № 2. — С. 118–121.
5. Щербина О. А. Локальные элиминационные алгоритмы решения разреженных дискретных задач / О. А. Щербина // Журнал вычислительной математики и математической физики. — 2008. — Т. 48, № 1. — С.161–177.
6. Allouche D. Towards parallel non serial dynamic programming for solving hard weighted CSP / David Allouche, Simon De Givry, and Thomas Schiex // Proceedings of the 16th international conference on Principles and practice of constraint programming (CP'10), David Cohen (Ed.). — Berlin, Heidelberg: Springer-Verlag, 2010. — P. 53-60.
7. Parallel SAT-solving with OpenCL. [Электронный ресурс] / [Beckers S., De Samblanx G., De Smedt F. et al.]. — 2011. Режим доступа: http://hgpu.org/?p=5769.
8. Bertele U. Nonserial Dynamic Programming / U. Bertele, F. Brioschi. — New York: Academic Press, 1972. — 235 p.
9. Boyer V. Solving knapsack problems on GPU / V. Boyer, D. Baz El, M. Elkihel // Comput. Oper. Res. — 2012. — 39 (1). — P. 42–47.
10. Dechter R. Bucket elimination: A unifying framework for reasoning / R. Dechter // Artificial Intelligence. — 1999. — V. 113. — P. 41–85.
11. Foster I. The Grid: Blueprint for a New Computing Infrastructure / I. Foster, C. Kesselman. — San Francisco: Morgan Kaufmann Publishers Inc, 2002. — 677 p.
12. Fujimoto N. A highly-parallel TSP solver for a GPU computing platform / N. Fujimoto, S. Tsutsui // Proc. of the 7th int. conf. on Numerical methods and applications (NMA'10) // I. Dimov, et al. (Eds.). — Springer-Verlag, Berlin, Heidelberg, 2010. — P. 264–271.
13. Gulati K. Boolean Satisfiability on a Graphics Processor / K. Gulati, S. P. Khatri // ACM Great Lakes Symposium on VLSI 2010. — P. 123–126.
14. Le Cun B. Parallel branch and bound algorithms / B. Le Cun, C. Roucairol, T.G. Crainic // Parallel Combinatorial Optimization, chapter 1. — USA: John Wiley and Sons, 2006. — P. 1–28.
15. Luong T-V. Parallel Hybrid Evolutionary Algorithms on GPU / T-V. Luong, N. Melab, E-G. Talbi // IEEE Congress on Evolutionary Computation (CEC). — Barcelone. — 2010.
16. Luong T-V. GPU-based Multi-start Local Search Algorithms / T-V. Luong, N. Melab, E-G. Talbi // Proceedings of Learning and Intelligent Optimization (LION'2011). — Rome, Italy, 2011. — P. 321–335.
17. Neumaier A. Nonserial dynamic programming and local decomposition algorithms in discrete programming [Электронный ресурс] / A. Neumaier, O. Shcherbina. Режим доступа: http://www.optimization-online.org/DB_HTML/2006/03/1351.html.
18. Otten L. Towards parallel search for optimization in graphical models / L. Otten, R. Dechter // Proc. of ISAIM 2010. — Fort Lauderdale, USA, 2010.
19. Otten L. Advances in distributed branch and bound / L. Otten and R. Dechter // Proceedings of ECAI'12, Montpellier, France, August 2012.

20. A survey of general-purpose computation on graphics hardware / [Owens J.D., Luebke D., Govindaraju N. et al.] // State of the Art Reports, August 2005. — Eurographics, 2005. — P. 21–51.
21. Pedemonte M. A survey on parallel ant colony optimization / M. Pedemonte, S. Nesmachnow, H. Cancela. // Applied Soft Computing. — 2011. — V. 11. — P. 5181–5197.
22. Ralphs T. K. A library hierarchy for implementing scalable parallel search algorithms / T. K. Ralphs, L. Ladanyi, M. J. Salzman // Journal of Supercomputing. — 2004. — V. 28(2). — P. 215–234.
23. Ralphs T. K. COIN/BCP User's Guide [Электронный ресурс] / T. K. Ralphs, L. Ladanyi. — 2001. Режим доступа: http://www.coin-or.org.
24. Ralphs T. K. Parallel branch, cut and price for large-scale discrete optimization / T. K. Ralphs, L. Ladanyi, M. J. Salzman // Mathematical Programming. —2003. — B. 98. — P. 253–280.
25. Shcherbina O. Nonserial dynamic programming and tree decomposition in discrete optimization / O. Shcherbina // Proceedings of Int. Conference on Operations Research "Operations Research 2006" (Karlsruhe, 6-8 September, 2006). — Berlin: Springer Verlag, 2007. — P. 155–160.

Статья поступила в редакцию 02.06.2012

УДК 621.391:517.518:510.52

НАБЛИЖЕНЕ ОБЧИСЛЕННЯ ПОДВІЙНИХ ІНТЕГРАЛІВ З ВИКОРИСТАННЯМ ЛАГРАНЖЕВОЇ ПОЛІНОМІАЛЬНОЇ ІНТЕРЛІНАЦІЇ

© О. М. Литвин, О. П. Нечуйвітер

Українська інженерно-педагогічна академія
кафедра вищої та прикладної математики
вул. Університетська 16, м. Харків, 61003, Україна
e-mail: *olesya@email.com*

Abstract. Formula of the evaluating of multiple integrals with using Lagrange polynomial interlineation was submitted. Cubature formula is investigated in the case when information about function is set of values on lines in $G=[-1,1]^2$ on the class of functions with condition $\left|f^{(p_1,p_2)}(x,y)\right| \leqslant M$. The estimation of error of approaching of the cubature formula is presented.

Вступ

Задача наближеного обчислення кратних інтегралів є однією з найбільш важливих задач обчислювальної математики. Однак на цей час виникає необхідність наближеного обчислення інтегралів від функцій багатьох змінних за допомогою інформаційних операторів різних типів. В якості даних можуть бути значення функції у вузлових точках, сліди функцій на системі взаємноперпендикулярних ліній або площин. Таку задачу ефективно дозволяє розв'язувати апарат інтерлінації функцій [1]. В роботах [1], [2] у випадку, коли інформація про функцію задана у вузлових точках, проведено порівняння наближеного обчислення інтегралу від функції двох змінних за кубатурою формулою центральних прямокутників та мішаною кубатурною формулою центральних прямокутників за точністю та кількістю використаної інформації. В цих же роботах наведений алгоритм розв'язання задачі наближеного обчислення інтегралу від функції двох змінних з використанням класичних базисних сплайнів, а також з використанням сплайн-інтерлінації функцій. Доведено, що для досягнення однієї і тієї ж точності кубатурні формули, що використовують сплайн-інтерлінацію використовують на порядок меншу кількість значень підінтегральної функції порівняно з класичними кубатурними формулами. Не дослідженим залишилось питання побудови кубатурних формул з використанням лагранжевої поліноміальної інтерлінації. Це питання є важливим і актуальним, оскільки піднімає питання оптимального вибору вузлів або ліній, при побудові кубатурних формул обчислення інтегралу від функції двох змінних.

Постановка задачі: побудувати кубатурну формулу наближеного обчислення інтегралу від функцій двох змінних з використанням лагранжевої поліноміальної інтерлінації функцій на класі дійсних функцій, визначених на $G = [-1,1]^2$ і таких, що $\left|f^{(p_1,p_2)}(x,y)\right| \leqslant M$, у випадку, коли інформація про функцію задана на лініях. Отримати оцінку похибки кубатурної формули.

1. Найкраща в $L_q[-1,1]^2, q=\infty,1,2$ лагранжева поліноміальна інтерлінація

Наведемо деякі теореми.

Теорема 1. [3] *Нехай $g(t) \in C^r(I),\ I=[0,1],\ 1 \leqslant r \leqslant n,\ L_{n-1}g(t)$ — інтерполяційний поліном Лагранжа степеня $n-1$ функції $g(t)$, із властивостями*

$$L_{n-1}g(t) = \sum_{k=1}^{n} g(t_k)\ell_{n-1,k}(t), \quad \ell_{n-1,k}(t) = \prod_{i=1,i\neq k}^{n} \frac{t-t_i}{t_k-t_i}, \ \ k=\overline{1,n}.$$

Тоді для залишку $R_n g(t) := g(t) - L_{n-1}g(t)$ справедливе інтегральне зображення

$$R_n g(t) = \sum_{k=1}^{n} \ell_{n-1,k}(t) \int\limits_{t_k}^{t} g^{(r)}(\tau)\frac{(t_k-\tau)^{r-1}}{(r-1)!}d\tau.$$

При знаходженні найкращої в $L_q[-1,1]^2$, $q=\infty,1,2$ лагранжевої поліноміальної інтерлінації на системі взаємно перпендикулярних прямих слід враховувати, що:

1) залишок інтерлінації дорівнює операторному добуткові залишків по кожній змінній окремо;
2) поліноми степеня n з найменшим відхиленням від нуля в метриці $L_q[-1,1]$ — це поліноми з коефіцієнтом одиниця при старшому степені, що є розв'язком екстремальної задачі

$$\max_{t\in[-1,1]}\left|t^n - \sum_{k=0}^{n-1} c_k t^k\right| \to \min_{c_0,\dots,c_{n-1}},\ q=\infty,$$

$$\int\limits_{-1}^{1}\left|t^n - \sum_{k=0}^{n-1} c_k t^k\right|^q dt \to \min_{c_0,\dots,c_{n-1}},\ 1\leqslant q<\infty.$$

Для них справедлива наступна теорема.

Теорема 2. [2] *При $q = \infty, 1, 2$ поліномами найкращого наближення є відповідно поліноми Чебишова 1-го роду*

$$T_{n,}(t) = \frac{\cos(n \times \arccos t)}{2^{n-1}},$$

поліноми Чебишова 2-го роду

$$T_{n,1}(t) = \frac{\sin\left((n+1)\arccos t\right)}{2^n\sqrt{1-t^2}},$$

поліноми Лежандра

$$T_{n,2}(t) = \frac{n!}{(2n)!}\frac{d^n}{dt^n}(t^2-1)^n.$$

Це означає, що при побудові поліноміальних інтерлінантів треба вибирати прямі інтерлінації $x = x_i$, $i = \overline{1,m}$; $y = y_j$, $j = \overline{1,n}$, так, щоб числа x_i, $i = \overline{1,m}$; y_j, $j = \overline{1,n}$, були коренями відповідних поліномів з найменшим відхиленням. Зокрема, при $q = 1$ справедлива наступна теорема.

Теорема 3. [1] *Нехай $p = (p_1, p_2)$, $f(x_1, x_2) \in C^p(J^2)$, $J = [-1, 1]$, $D^p = \dfrac{\partial^{p_1+p_2}}{\partial x_1^{p_1}\partial x_2^{p_2}}$, $B^p = \left\{g(x) | g \in C^p(J^2),\ D^p g = 0\right\}$, $E(f)$ — величина найкращого наближення функції множиною B^p за нормою $\|\cdot\| = \|\cdot\|_{L_1(J^2)}$; $g^* \in B^p$ — елемент найкращого наближення; $x_{1i_1} = \cos(i_1\pi/(p_1+1))$, $i_1 = \overline{1,p_1}$, $x_{2i_2} = \cos(i_2\pi/(p_2+1))$, $i_2 = \overline{1,p_2}$ — нулі поліномів Чебишова 2-го роду відповідно степеня p_1 та p_2:*

$$U_m(t) = \frac{\sin(m+1)\theta}{\sin\theta}, \quad \cos\theta = t, \quad m = 0, 1, \ldots,$$

$l_{kp_ki_k}$ — базисні поліноми Лагранжа, $l_{kp_ki_k}(x_{ki'_k}) = \delta_{i_ki'_k}$,

$$g^*(x) = \sum_{i_1=1}^{p_1} f(x_{1i_1}, x_2) l_{1p_1i_1}(x_1) + \sum_{i_2=1}^{p_2} f(x_1, x_{2i_2}) l_{2p_2i_2}(x_2) - \tag{1}$$

$$-\sum_{i_1=1}^{p_1}\sum_{i_2=1}^{p_2} f(x_{1i_1}, x_{2i_2}) l_{1p_1i_1}(x_1) l_{2p_2i_2}(x_2).$$

Для $f(x)$ існує єдиний найближчий до $f(x)$ за нормою $\|\cdot\|$ елемент $g^* \in B^p$ і цей елемент має вигляд (1) тобто є інтерлінантом, який інтерлінує $f(x)$ на лініях $x_k = x_{ki_k}$, $i_k = \overline{1,p_k}$; $k = 1, 2$. Залишок наближення функції $f(x)$ найкращим елементом, має наступний вид.

$$f(x) - g^*(x) = \frac{U_{p_1}(x_1)U_{p_2}(x_2)}{2^{p_1+p_2}p_1!p_2!} f^{(p_1,p_2)}(\xi_1, \xi_2), (\xi_1, \xi_2) \in J^2, \tag{2}$$

де (ξ_1, ξ_2) — деяка точка, що залежить від $(x_1, x_2) \in J^2$. Тому найменше значення величини $\| f(x) - g^*(x) \|$ досягається на тих g^*, для яких величина $\left\| \prod\limits_{k=1}^{2} \prod\limits_{i_k=1}^{p_k} (x_k - x_{ki_k}) \right\|$ є найменшою. Цю умову задовольняють поліноми, вузли яких є нулями поліномів Чебишова 2-го роду.

2. Кубатурна формула наближеного обчислення кратного інтегралу з використанням лагранжевої поліноміальної інтерлінації функцій

Для наближеного обчислення інтегралу

$$I(f) = \int\limits_{-1}^{1} \int\limits_{-1}^{1} f(x_1, x_2) dx_1 dx_2$$

пропонується кубатурна формула

$$\tilde{I}(f) = \int\limits_{-1}^{1} \int\limits_{-1}^{1} Lf(x_1, x_2) dx_1 dx_2,$$

де

$$Lf(x_1, x_2) = \sum_{i_1=1}^{p_1} f(x_{1i_1}, x_2) l_{1p_1i_1}(x_1) + \sum_{i_2=1}^{p_2} f(x_1, x_{2i_2}) l_{2p_2i_2}(x_2) -$$

$$- \sum_{i_1=1}^{p_1} \sum_{i_2=1}^{p_2} f(x_{1i_1}, x_{2i_2}) l_{1p_1i_1}(x_1) l_{2p_2i_2}(x_2).$$

В розгорнутому вигляді кубатурна формула має вид:

$$\tilde{I}(f) = \sum_{i_1=1}^{p_1} \int\limits_{x_{1i_1}}^{x_{1i_1+1}} l_{1p_1i_1}(x_1) dx_1 \int\limits_{x_{2i_2}}^{x_{2i_2+1}} f(x_{1i_1}, x_2) dx_2 +$$

$$+ \sum_{i_2=1}^{p_2} \int\limits_{x_{2i_2}}^{x_{2i_2+1}} l_{2p_2i_2}(x_2) dx_2 \int\limits_{x_{1i_1}}^{x_{1i_1+1}} f(x_1, x_{2i_2}) dx_1 -$$

$$- \sum_{i_1=1}^{p_1} \sum_{i_2=1}^{p_2} f(x_{1i_1}, x_{2i_2}) \int\limits_{x_{1i_1}}^{x_{1i_1+1}} l_{1p_1i_1}(x_1) dx_1 \int\limits_{x_{2i_2}}^{x_{2i_2+1}} l_{2p_2i_2}(x_2) dx_2.$$

Теорема 4. *Справедлива наступна оцінка похибки наближення інтегралу $I(f)$ кубатурою формулою $\tilde{I}\left(f\right)$*

$$\left|I\left(f\right)-\tilde{I}\left(f\right)\right|\leqslant\frac{Mp_1p_2\pi^2}{2^{p_1+p_2}(p_1+1)!(p_2+1)!}.$$

Доведення. Знайдемо оцінку похибки наближення. Маємо

$$\left|I\left(f\right)-\tilde{I}\left(f\right)\right|\leqslant$$

$$\leqslant\int\limits_{-1}^{1}\int\limits_{-1}^{1}\left|f\left(x_1,x_2\right)-Lf\left(x_1,x_2\right)\right|dx_1dx_2=$$

$$=\sum_{i_1=1}^{p_1}\sum_{i_2=1}^{p_2}\int\limits_{x_{1i_1}}^{x_{1i_1+1}}\int\limits_{x_{2i_2}}^{x_{2i_2+1}}\left|\frac{U_{p_1}(x_1)U_{p_2}(x_2)}{2^{p_1+p_2}p_1!p_2!}f^{(p_1,p_2)}(\xi_1,\xi_2)\right|dx_1dx_2\leqslant$$

$$\leqslant M\sum_{i_1=1}^{p_1}\sum_{i_2=1}^{p_2}\int\limits_{x_{1i_1}}^{x_{1i_1+1}}\int\limits_{x_{2i_2}}^{x_{2i_2+1}}\frac{\left|U_{p_1}(x_1)\right|\left|U_{p_2}(x_2)\right|}{2^{p_1+p_2}p_1!p_2!}dx_1dx_2=$$

$$=\frac{M}{2^{p_1+p_2}p_1!p_2!}\sum_{i_1=1}^{p_1}\sum_{i_2=1}^{p_2}\int\limits_{x_{1i_1}}^{x_{1i_1+1}}\left|U_{p_1}(x_1)\right|dx_1\int\limits_{x_{2i_2}}^{x_{2i_2+1}}\left|U_{p_2}(x_2)\right|dx_2=\frac{M}{2^{p_1+p_2}p_1!p_2!}\times$$

$$\times\sum_{i_1=1}^{p_1}\sum_{i_2=1}^{p_2}\int\limits_{x_{1i_1}}^{x_{1i_1+1}}\left|\frac{\sin\left((p_1+1)\arccos x_1\right)}{\sqrt{1-x_1^2}}\right|dx_1\int\limits_{x_{2i_2}}^{x_{2i_2+1}}\left|\frac{\sin\left((p_2+1)\arccos x_2\right)}{\sqrt{1-x_2^2}}\right|dx_2\leqslant$$

$$\leqslant\frac{M}{2^{p_1+p_2}p_1!p_2!}\sum_{i_1=1}^{p_1}\sum_{i_2=1}^{p_2}\int\limits_{x_{1i_1}}^{x_{1i_1+1}}\frac{dx_1}{\sqrt{1-x_1^2}}\int\limits_{x_{2i_2}}^{x_{2i_2+1}}\frac{dx_2}{\sqrt{1-x_2^2}}=$$

$$=\frac{M}{2^{p_1+p_2}p_1!p_2!}\sum_{i_1=1}^{p_1}\sum_{i_2=1}^{p_2}\left(\frac{\pi}{2}-\arccos x_1\right)\Bigg|_{x_{1i_1}}^{x_{1i_1+1}}\left(\frac{\pi}{2}-\arccos x_2\right)\Bigg|_{x_{2i_2}}^{x_{2i_2+1}}=$$

$$=\frac{M}{2^{p_1+p_2}p_1!p_2!}\sum_{i_1=1}^{p_1}\sum_{i_2=1}^{p_2}\left(\arccos x_{1i_1}-\arccos x_{1i_1+1}\right)\left(\arccos x_{2i_2}-\arccos x_{2i_2+1}\right)=$$

$$=\frac{M}{2^{p_1+p_2}p_1!p_2!}\sum_{i_1=1}^{p_1}\sum_{i_2=1}^{p_2}\left(\frac{\pi i_1}{p_1+1}-\frac{\pi\left(i_1+1\right)}{p_1+1}\right)\left(\frac{\pi i_2}{p_2+1}-\frac{\pi\left(i_2+1\right)}{p_2+1}\right)=$$

$$=\frac{M}{2^{p_1+p_2}p_1!p_2!}\sum_{i_1=1}^{p_1}\sum_{i_2=1}^{p_2}\frac{\pi}{p_1+1}\frac{\pi}{p_2+1}=\frac{Mp_1p_2\pi^2}{2^{p_1+p_2}(p_1+1)!(p_2+1)!}.$$

Теорема доведена. □

3. Чисельний експеримент

Для функції

$$f\left(x_1, x_2\right) = \cos\left(x_1 + x_2\right),$$

у якої

$$\left|f^{(p_1,p_2)}\left(x_1, x_2\right)\right| \leqslant 1,$$

наведемо в таблиці наближені значення інтегралів за кубатурною формулою $\tilde{I}\left(f\right)$.

Точне значення

$$I\left(f\right) = \int\limits_{-1}^{1}\int\limits_{-1}^{1} f\left(x_1, x_2\right) dx_1 dx_2 = 2.83229367309428.$$

Оцінку похибки наближення інтегралу $I(f)$ кубатурою формулою $\tilde{I}\left(f\right)$, знайдену теоретично позначимо, через

$$\varepsilon = \frac{p_1 p_2 \pi^2}{2^{p_1+p_2}(p_1+1)!(p_2+1)!}.$$

Таблица 1. Наближене обчислення інтегралу $I(f)$ за кубатурою формулою $\tilde{I}\left(f\right)$.

$p_1 = p_2 = \ell$	$\tilde{I}\left(f\right)$	$\left\|I\left(f\right) - \tilde{I}\left(f\right)\right\|$	ε
2	2.82707748909675	0.005216183997534	0.068538919452009
3	2.832286830474443	0.000006842619856	0.002409571386985
4	2.83229271424136	0.000000958852927	0.000042836824658
5	2.83229367288868	0.000000000205606	0.000000464809295
6	2.83229367305887	0.000000000035412	0.000000003414925
7	2.83229367309428	$1.3 \cdot 10^{-15}$	0.000000000018157

Заключення

На цей час актуальною є задача наближеного обчислення інтегралів від функцій багатьох змінних у випадку, коли в якості даних можуть бути сліди функцій на системі взаємноперпендикулярних ліній. Таку задачу ефективно дозволяє розв'язувати апарат інтерлінації функцій. В роботі розглянута кубатурна формула наближеного обчислення інтегралу від функцій двох змінних з використанням лагранжевої поліноміальної інтерлінації функцій на класі дійсних функцій, визначених на $G = \left[-1, 1\right]^2$ і таких, що $\left|f^{(p_1,p_2)}(x,y)\right| \leqslant M$. Отримана оцінка похибки кубатурної формули. Чисельний експеримент підтверджує теоретичний результат.

Побудована кубатурна формула є першим кроком в дослідженні питання оптимального вибору вузлів, ліній, площин при побудові кубатурних формул наближеного обчислення, як кратних інтегралів, так і інтегралів від швидкоосцилюючих функцій багатьох змінних.

Список літератури

1. Литвин О.М. Інтерлінація функцій та деякі її застосування / О.М. Литвин. — Харків.: Основа, 2002. — 544 с.
2. Литвин О.М. Методи обчислень. Додаткові розділи. Навчальний посібник / О.М. Литвин. — К.: Наукова думка, 2005. — 331 с.
3. Литвин О.М. Интерполирование функций. Учеб. пособ. / О.М. Литвин. — Киев: Учеб.-метод. каб. высш. образования (УМК ВО), 1988. — 31 с.

Статья поступила в редакцию 27.12.2011

УДК 519.711+519.6+681.3.01+556:06

МОДЕЛЮВАННЯ СТОКУ Р. ТИСА ІЗ УРАХУВАННЯМ МАКСИМАЛЬНИХ ВИТРАТ ПРИТОК

© Т. П. Мельник

Львівський національний університет імені Івана Франка
вул. Коперніка, 12, кв. 12, м. Львів, 79000, Украина
e-mail: *Tatiana_Krizhovec@mail.ru*

Abstract. Considered the mathematical model of the flow of the river Tisza in view the maximum cost flows. Determined of the value of maximum flows by the method of bandwidth analysis. Provided recommendations on further use of this method.

Вступ

Основними причинами формування стихійних явищ Карпатського регіону є природно-кліматичні та геологічні особливості, внаслідок чого різкий підйом рівнів води в річках призводить до затоплення територій населених пунктів, виробничих об'єктів і завдає значних збитків економіці області [1]. Така ж проблема існує і в інших державах, які розташовані в басейні р. Тиси, що потребує міжнародної координації системи спостережень та оповіщення населення і необхідність забезпечення [2]:

- постійних замірів в характерних створах витрат та рівнів води, які дадуть достовірну основу для розрахунків гідрологічних процесів;
- оперативної передачі для обробки первинної інформації та подаль-шого імітаційного моделювання гідрологічних процесів.

Дослідження *грунтується* на застосуванні матеріалів спостережень служби гідрологічного моніторингу, зокрема, на водомірних постах і гідроморфометричнтих створах з використанням даних геодезичних вимірювань на ділянках рік Закарпаття, а також топографічні і тематичні карти, довідково-методична і спеціальна література.

Метою є визначення величини максимальних потоків, які можна пропустити через русло річки, виходячи із даних водомірних постів.

1. Постановка задачі

Перш за все варто відмітити, що довготермінові точні прогнози неможливі із-за випадковості кліматичних проявів. Природа ймовірності є настільки різнобічною, що неможливо знайти явище, яке б не несло випадковості. Тому наявна вона і в математичних рівняннях, що відображають об'єктивно існуючу реальність у вигляді моделей.

Нехай, маємо мережу з n витоків приток і одним стоком р. Тиси і нехай дуги приток матимуть обмежену здатність. Необхідно знайти такі потоки по руслах приток, що сформують максимальний потік, що відповідає потоку русла р. Тиси під час ситуації паводку.

Використаємо метод міток, зафіксувавши деяке допустиме значення залежності $Q(h)$. Вузли розглянемо як проміжні водомірні пости річки, а дуги — як розподільчі канали.

Нехай мітку водомірного поста будемо використовувати для встановлення величини Q і джерела притоки або самої річки, що впливає на зміну стоку по річці. q_j — величина потоку із вузла i у вузол j це викликає збільшення потоку по цій дузі, то будемо приймати, що вузол j мічений із вузла i символом $(+q_j)$. Вузлу j припишемо мітку $[+q_j, i]$. Якщо q_j одиниць потоку викликають зменшення потоку по дузі, то вузол j мічений із вузла i символом $(-q_j)$, тоді вузлу j приписують мітку $[-q_j, i]$.

Потік із вузла i у вузол j збільшується, коли q_j одиниць додаткового потоку направляється у вузол j по дузі (i, j) в напрямку, який співпадає з орієнтацією. Тоді дуга (i, j) буде наближатися до прямої. Якщо вузол j мічений із вузла i дуга (i, j) пряма, то потік по даній дузі збільшується і величина, яка відповідає решті невикористаної пропускної здатності дуги, повинна бути потрібним чином скорегована. Це і буде *надлишковою пропускною здатністю.*

Якщо деякому вузлу припишемо мітку i при цьому використаємо пряму гілку, то вона може мати тільки надлишкову пропускну здатність. Шлях потоку, який збільшує видатки визначимо як зв'язну послідовність прямих і обернених дуг, по яких об'єднуються декілька потоків в один. Потік по кожній прямій дузі збільшується, при цьому за наявності перевищення утворюються зони затоплення. Оптимальний шлях потоку використаємо для вибору такого способу зміни потоку, при якому потік у вузлі збільшується і для кожного внутрішнього вузла мережі при цьому не буде відхилено умову збереження потоку.

Візьмемо за основу алгоритм Гоморі, ідея якого полягає в побудові максимального дерева, гілки якого відповідають перерізам, а параметр гілок — величині перетинів.

Нехай (i, j) має потік $x_{i,j} \geqslant 0$, а $p_{i,j}$ — пропускна здатність дуги, тоді:

якщо $x_{i,j} \leqslant p_{i,j}$ — ситуація відсутності затоплюваної території,

якщо $x_{i,j} \geqslant p_{i,j}$ — то наявна паводкова ситуація.

Різниця $x_{i,j} - p_{i,j}$ буде наявною надлишковою витратою, що формує зону затоплення, потік по прямій дузі (i,j) може збільшуватися на $q_j = \min|q_i, x_{i,j} - u_{i,j}|$. Зменшення потоку по дузі (i,j) можливе тільки тоді, коли $x_{i,j} > 0$. Відповідно, потік по оберненій дузі (i,j) може бути зменшеним на $q_j = \min|\, q_i, x_{i,j}\,|$. Потоку р. Тиси присвоїмо мітку $[+q_j, n]$, де n — номер вузла, тоді максимальний потік може бути збільшений, або зменшений на q_i вузла.

Нехай $F = (N, M)$, де N — множина вузлів, M — множина дуг, і нехай $c_{i,j}$ — пропускна здатність дуги (i,j) із множини М, і $c_{i,j} = c_{j,i}$. Максимальний потік між вузлами i і j дорівнює $q\max_{i,j}$, $(P,\bar{P})_{i,j}$ — мінімальний перетин, який відокремлює вузол i від $j : i \in P; J \in \bar{P}; c(P,\bar{P})_{i,j}$ — пропускна спроможність мінімального перетину, який відокремлює вузол i від j. Згідно теореми про максимальний потік і мінімальний перетин $q\max_{i,j} = c(P,\bar{P})_{i,j}$.

Якщо деякий вузол $s \in \bar{P}$, то $q\max_{i,s} \leqslant c(P,\bar{P})_{i,j}$.

Якщо деякий вузол $s \in P$, то $q\max_{s,j} \leqslant c(P,\bar{P})_{i,j}$.

Звідси випливає, що $q\max_{i,j} \geqslant q\max_{i,s}$ і $q\max_{i,j} \geqslant q\max_{s,j}$, тому $q\max_{i,j} \geqslant |q\max_{i,s}; q\max_{s,j}\,|$. У загальному можна записати,

що $q\max_{i,j} \geqslant \min[q\max_{i,p}, q\max_{p,s}, q\max_{s,q}, q\max_{q,j}]$,

але $q\max_{i,j} \leqslant \min[q\max_{i,p}, q\max_{p,s}, q\max_{s,q}, q\max_{q,j}]$,

де (i,j) — довільна дуга, яка не належить дереву. Отже, замість цієї дуги можна взяти іншу з більшою вагою і для будь-якої дуги, яка не належить дереву: $q\max_{i,j} = \min[q\max_{i,p}, q\max_{p,s}, q\max_{s,q}, q\max_{q,j}]$.

При необхідності визначити максимальний потік між двома вузлами в дереві знаходять шлях, який з'єднує ці вузли і вибрати при цьому дугу з мінімальною вагою. Її вага і буде максимальним потоком між цими вузлами. Нехай d — це джерело, а вузол x — стік, то максимальний потік між ними $q\max_{d,x} = c(P,\bar{P})_{d,x}$. Нехай $\bar{W}_{i,j}$ — множина вузлів, яка утворюється у результаті конденсації всіх вузлів, які знаходяться по одну сторону перерізу, де не містяться вузли i і j; $\bar{M}_{i,j}$ — множина дуг, які об'єднують вузли із множини $\bar{W}_{i,j}$. Якщо відома пропускна здатність дуг, які належать множині $\bar{M}_{i,j}$, то для знаходження максимального потоку між вузлами i і j використаємо метод міток.

Нехай $j_1, j_2, \ldots, j_\ell$ — вузли із множини $\bar{P}$, де $i \in P$, пропускні здатності дуг із множини $\bar{M}_{i,j}$. При конденсації множини $\bar{P}$ дуги $(\mathrm{i}, j_1), (\mathrm{i}, j_2), \ldots, (\mathrm{i}, j_\ell)$ приймають за одну дугу, яка з'єднує вузол i і кондиційований вузол $\bar{P}$. Пропускна здатність цієї дуги $c_{i,\bar{p}} = c_{i,j_1} + c_{i,j_2} + \ldots + c_{i,j} = \sum_{z=1}^{\ell} c_{i,j_z}$. Для визначення величини $q\max_{i,j}$ знову потрібно знайти мінімальний переріз, який відокремлює вузли i і j, тобто $(P,\bar{P})_{i,j}$

з мінімальною пропускною здатністю. Далі вибираємо другу пару вузлів, яка належить P, або $\bar{P}$, і побудуємо конденційовану мережу. Далі методом міток визначимо другий перетин і побудуємо нову конденційовану мережу і так далі.

Висновки

Для аналізу формування величини стоку р. Боржави здійснено обстеження від автодорожнього мосту через р. Тису в південній частині м. Рахів до злиття рік Біла Тиса та Чорна Тиса. Для можливості врахування максимальних витрат приток здійснено дослідження водозборів річок Тересви, Шопурки і Косівської. На основі чого можна зробити висновок, що рівень води у р. Боржаві формує об'єм стоку русла р. Тиси, що значною мірою поповнюється водами приток. Звідси випливає необхідність акумуляції паводкового стоку основного русла р. Тиси та її приток, що дозволить уникнути критичних рівнів на р. Боржаві, забезпечити збалансованість фактора зволоження грунтів сільськогосподарських земель, запобігти збитків.

Список літератури

1. Чіпак В. П. Система протипаводкових заходів у басейні р. Боржава / Чіпак В. П. — Рівне: Волинські обереги, 2008. — 202 с.
2. Якушев А. І. Гідроморфологічний моніторинг стоку річок басейну р. Тиси і її приток / Якушев А. І., Зубач В. М., Мельник Т. П. — Рівне: Волинські обереги, 2009. — 64 с.
3. Лук'янець О. І. Визначення характерних рівнів води при нестабільних умовах переміщення водних мас по русловій мережі / Лук'янець О. І., Сусідко М. М. // Наук. праці УкрНДГМІ, 2007. — Вип. 256. — С. 207–213.
4. Лазарчук Н. А. Математичне моделювання дощового стоку на осушуваних системах гірських районів України / Лазарчук Н. А., Петрук В. А. // Екологічні проблеми при зрошенні і осушенні: Тези докладів міжнар. конф. — Київ, 1993. — С. 80–89.
5. Ромащенко М. І. Катастрофічна повінь і затоплення на Закарпатті у березні 2001 р. / Ромащенко М. І., Савчук Д. П. — К.: Водне господарство України, 2002. — №1-2. — С. 4–10.
6. Ромащенко М. Водні стихії. Карпатські повені / Ромащенко М., Савчук Д. — К.: Аграрна наука, 2002. — 304 с.
7. Приймаченко Н. В. Структурні особливості дощових паводків на гірських водозборах / Приймаченко Н. В. // Наук. праці УкрНДГМІ, 2003. — Вип. 251. — С. 49–53.
8. Сапсай Г. І. Елементи автоматизації системи державного моніторингу гідрологічного режиму річок / Сапсай Г.І., Чіпак В.П., Мельник Т.П. // Тези доповідей V Міжн. наук.-практ. конф. «Передові наукові розробки — 2006». — Дніпропетровськ, 01-15 вересня 2006 р.
9. Мельник Т. П. Негативний вплив дощових паводків на водний режим грунтів території басейну річки Тиса / Мельник Т. П. // Materialy IV mezinarodni vedecko - prakticka konference „Veda a technologie: krok do budoucnosti — 2008". — Dil 14. Zemepis a geologie.Ekologie: Praha. Publishing House „Education and Scince" s.r.o. — S. 21–24.

Статья поступила в редакцию 20.11.2011

УДК 517.977.58; 517.977.54

ДОСТАТНІ УМОВИ ОПТИМАЛЬНОСТІ В ЗАДАЧІ СТРУКТУРНО-ПАРАМЕТРІЧНОЇ ОПТИМІЗАЦІЇ

© В. В. Пічкур, Є. М. Страхов

Київський національний університет імені Тараса Шевченка
факультет кібернетики
вул. Володимирська, 64, 01601, Київ, Україна
e-mail: *vpichkur@gmail.com*

Одеський національний університет імені І. І. Мечникова
інститут математики, економіки і механіки
вул. Дворянська, 2, 65026, Одеса, Україна
e-mail: *swebus86@gmail.com*

Abstract. In this paper dynamical system with fixed switching points is considered. Existence conditions for a solution of a structural and parametric optimization problem in a class of structural controls and sufficient conditions for optimality are proved. A numerical method for structural optimization of linear system with quadratic functional is created.

Вступ

При оптимізації складних технічних систем функції керування, як правило, задаються у структурно-параметричній формі. Задачі структурно-параметричної оптимізації вивчалися із застосуванням варіаційного методу, зокрема, в праці [3] були побудовані алгоритми типу градієнтного спуску. Важливим випадком структурного представлення керування є випадок релейних керувань, або, більш широко, керувань, що вибираються з дискретної множини. В [5, 6] описано алгоритми, які використовують структуризацію фазового простору, при цьому керування вибирається в класі кусково-постійних. У роботі [1] обгрунтований принцип оптимальності Беллмана для задачі вибору оптимальної структури, одержано рівняння Беллмана у інтегральній та диференціальній формах. Випадок структур керування, що містять фазову змінну, був розглянутий у статті [7]. Для задачі з фіксованими точками перемикання був доведений принцип оптимальності Беллмана та одержано рівняння Беллмана в інтегральній та інтегро-диференціальній формах.

У статті отримані достатні умови оптимальності в задачі структурно-параметричної оптимізації з фіксованими точками перемикання. Для випадку лінійної системи керування доведені умови існування та єдиності розв'язку. Запропонований чисельний алгоритм оптимізації за параметрами лінійної системи з квадратичним критерієм якості.

1. Достатні умови оптимальності в задачі структурно-параметричної оптимізації з фіксованими точками перемикання

Розглянемо задачу оптимального керування

$$J(x,\,u) = \int\limits_{t_0}^{T} f_0\left(x\left(t\right),\,u\left(t\right),\,t\right)\,dt + \Phi\left(x\left(T\right)\right)\ \to\ \inf \tag{1}$$

за умов

$$\frac{dx}{dt} = f\left(x,\,u,\,t\right),\ x\left(t_0\right) = x_0\,. \tag{2}$$

Тут x — вектор фазових координат розмірності n, $x(t) \in X(t) \subseteq \mathbb{R}^n$, $t \in [t_0,\,T]$ — фазові обмеження, u — вектор керування розмірності m, $f_0(x,\,u,\,t)$, $\Phi(x)$ — неперервні функції, $f(x,\,u,\,t)$ — n-вимірна вектор-функція така, що справджуються умови: для будь-якого фіксованого $t \in [t_0,\,T]$ функція $f(x,\,u,\,t)$ є неперервною за x; для будь-яких фіксованих $x \in X$ та u функція $f(x,\,u,\,t)$ є вимірною за t; існує така інтегрована функція $p(t)$, що $\|f(x,\,u,\,t)\| \leq p(t)$, $t \in [t_0,\,T]$; на проміжку $[t_0,\,T]$ виконується умова Ліпшиця за керуванням і за фазовою змінною.

Нехай керування в задачі (1)–(2) задано в структурній формі

$$u\left(t\right) = \Psi_i\left(t,\,b_i,\,x\left(t\right)\right),\quad t \in \left[t_i,\,t_{i+1}\right)\,. \tag{3}$$

де $t_0 < t_1 < \ldots < t_N = T$ — точки перемикання, $b_i \in R_i$ — числові параметри, $R_i \subset \mathbb{R}^{k_i}$, $i = 0,\,\ldots,\,N-1$, k_i — натуральні числа. Функції $\Psi_i(t,\,b_i,\,x)$, $t \in [t_i,\,t_{i+1})$ є такими, що: на кожному з відрізків $[t_i,\,t_{i+1})$ функції $\Psi_i(t,\,b_i,\,x)$ є неперервними за t і b_i; на проміжку $[t_0,\,T]$ виконується умова Ліпшиця за фазовою змінною. Точки перемикання $t_0 < t_1 < \ldots < t_N = T$ є фіксованими. Умови на праву частину системи (2) і функцію керування (3) дозволяють задовольнити умови існування та єдиності розв'язку задачі Коші у формі Каратеодорі [9]. Крім того, мають виконуватись умови продовжуваності розв'язку на інтервал $t \in [t_0,\,T]$ [1].

Задача (1)–(3) є задачею структурно-параметричної оптимізації з керуванням, заданим в структурному класі (3). Припустимо, що розв'язок задачі (1)–(3) існує, $u^*(t) = \Psi_i(t,\,b_i^*,\,x^*(t))$, $t \in [t_i,\,t_{i+1})$ — оптимальне керування в класі (3), $x^*(t)$ — оптимальна траєкторія, $\{b_i^*\}$, $i = 0,...,N-1$ — оптимальний набір параметрів. Розглянемо допоміжну задачу. Зафіксуємо $t_s \in \{t_1,\,\ldots,\,t_{N-1}\}$. Задача полягає в тому,

[1]Продовжуваність розв'язку на довільний інтервал забезпечує, наприклад, умова квазілінійності

щоб мінімізувати функціонал

$$J_s\left(x,\,u\right)=\int\limits_{t_s}^{T}f_0\left(x\left(t\right),\,u\left(t\right),\,t\right)\,dt+\Phi\left(x\left(T\right)\right)$$

за умов

$$\frac{dx}{dt}=f\left(x,\,u,\,t\right),\,x\left(t_s\right)=x^*\left(t_s\right),$$
$$u\left(t\right)=\Psi_i\left(t,\,b_i,\,x\left(t\right)\right),\,t\in\left[t_i,\,t_{i+1}\right)\subset\left[t_s,\,T\right],\,b_i\in R_i.$$

Для задачі (1)–(3) справедливий принцип оптимальності Беллмана [7].

Теорема 1. Якщо набір параметрів $\left\{\tilde{b}_i\right\}_{i=0}^{N-1}$ та траєкторія $\tilde{x}\left(t\right)$ є оптимальними у допоміжній задачі, то на відрізку $t\in\left[t_s,\,T\right]$ вони співпадають з оптимальним набором параметрів та оптимальною траєкторією задачі (1)–(3).

Означення 1. [7] Функція

$$B\left(z,\,t_s\right)=\inf_{b_j\in R_j}\left\{\sum_{j=s}^{N-1}\int\limits_{t_j}^{t_{j+1}}f_0\left(x\left(t\right),\,\Psi_j\left(t,\,b_j,\,x\left(t\right)\right),\,t\right)\,dt+\Phi\left(x\left(T\right)\right)\right\},\qquad(4)$$

що визначена на розв'язках системи (2) при початковій умові $x\left(t_s\right)=z$, називається функцією Беллмана задачі (1)–(3). Тут $z\in X\left(t_s\right)$, $x\left(\cdot\right)$ — розв'язок системи (2) при допустимому керуванні $u\left(t\right)=\Psi\left(t,\,b_i,\,x\left(t\right)\right),\,t\in\left[t_s,\,T\right],\;t_s\in\left\{t_0,\,\dots,\,t_N\right\}$, інфінум в правій частині співвідношення (4) береться за допустимими керуваннями при $t\in\left[t_s,\,T\right]$.

За означенням функції Беллмана $B\left(x_0,\,t_0\right)$ дорівнює оптимальному значенню функціоналу (1) для задачі (1)–(3) з фіксованим лівим кінцем $x\left(t_0\right)=x_0$, тобто

$$B\left(x_0,\,t_0\right)=J\left(x^*,\,u^*\right).$$

Рівняння Беллмана для задачі (1)–(3) в інтегральній формі має вигляд

$$B\left(z,\,t_s\right)=\inf_{b_s\in R_s}\left\{\int\limits_{t_s}^{t_{s+1}}f_0\left(x\left(\tau\right),\,\Psi_s\left(\tau,b_s,\,x\left(\tau\right)\right),\,\tau\right)\,d\tau\;+\;B\left(x\left(t_{s+1}\right),\,t_{s+1}\right)\right\}.$$

Інтегро-диференціальне рівняння Беллмана задачі (1)–(3) має вигляд

$$\inf_{b_s\in R_s}\left\{\int\limits_{t_s}^{t_{s+1}}\left[f_0\left(x\left(\tau\right),\,\Psi_s\left(\tau,b_s,\,x\left(\tau\right)\right),\,\tau\right)+\frac{\partial B\left(x\left(\tau\right),\,\tau\right)}{\partial\tau}+\right.\right.$$
$$\left.\left.+\left\langle grad_x B\left(x\left(\tau\right),\,\tau\right),\,f\left(x\left(\tau\right),\,\Psi_s\left(\tau,\,b_s,\,x\left(\tau\right)\right),\,\tau\right)\right\rangle\right]\,d\tau\right\}=0.\qquad(5)$$

Має місце така теорема.

Теорема 2 (достатні умови оптимальності)**.** Нехай функція $B(z,t)$ є кусково неперервно диференційованим розв'язком рівняння (5) з граничною умовою $B(z,T) = \Phi(z)$. Параметри $b_s^*, s = 0, ..., N-1$, знайдені з умов

$$\int\limits_{t_s}^{t_{s+1}} \Big[f_0\left(x\left(\tau\right),\, \Psi_s\left(\tau, b_s^*,\, x\left(\tau\right)\right),\, \tau\right) + \frac{\partial B\left(x\left(\tau\right),\, \tau\right)}{\partial \tau} +$$

$$+ \left\langle grad_x B\left(x\left(\tau\right),\, \tau\right),\, f\left(x\left(\tau\right),\, \Psi_s\left(\tau,\, b_s^*,\, x\left(\tau\right)\right),\, \tau\right)\right\rangle \Big]\, d\tau =$$

$$= \inf_{b_s \in R_s} \Bigg\{ \int\limits_{t_s}^{t_{s+1}} \Big[f_0\left(x\left(\tau\right),\, \Psi_s\left(\tau, b_s,\, x\left(\tau\right)\right),\, \tau\right) + \frac{\partial B\left(x\left(\tau\right),\, \tau\right)}{\partial \tau} +$$

$$+ \left\langle grad_x B\left(x\left(\tau\right),\, \tau\right),\, f\left(x\left(\tau\right),\, \Psi_s\left(\tau,\, b_s,\, x\left(\tau\right)\right),\, \tau\right)\right\rangle \Big]\, d\tau \Big\}, s = 0, ..., N-1, \qquad (6)$$

при підстановці у функції керування

$$u_s^*\left(z,t\right) = \Psi_s\left(t,\, b_s^*,\, z\right), s = 0,\, ...\,,\, N-1$$

породжують при $z = x$ єдиний розв'язок $x^*(\cdot)$ системи (2). Тоді набір параметрів $b_s^*, s = 0, ..., N-1$ є оптимальним у задачі (1)–(3).

Доведення. Оскільки вектори $b_s^*, s = 0, ..., N-1$ є розв'язком задачі (6) і для $B(z,t)$ має місце (5), то

$$\int\limits_{t_s}^{t_{s+1}} \Big[f_0\left(x^*\left(\tau\right),\, \Psi_s\left(\tau, b_s^*,\, x^*\left(\tau\right)\right),\, \tau\right) + \frac{\partial B\left(x^*\left(\tau\right),\, \tau\right)}{\partial \tau} +$$

$$+ \left\langle grad_x B\left(x^*\left(\tau\right),\, \tau\right),\, f\left(x^*\left(\tau\right),\, \Psi_s\left(\tau,\, b_s^*,\, x^*\left(\tau\right)\right),\, \tau\right)\right\rangle \Big]\, d\tau = 0.$$

Звідси

$$\int\limits_{t_s}^{t_{s+1}} f_0\left(x^*\left(\tau\right),\, \Psi_s\left(\tau, b_s^*,\, x^*\left(\tau\right)\right),\, \tau\right) d\tau = -\int\limits_{t_s}^{t_{s+1}} \Big[\frac{\partial B\left(x^*\left(\tau\right),\, \tau\right)}{\partial \tau} +$$

$$+ \left\langle grad_x B\left(x^*\left(\tau\right),\, \tau\right),\, f\left(x^*\left(\tau\right),\, \Psi_s\left(\tau,\, b_s^*,\, x^*\left(\tau\right)\right),\, \tau\right)\right\rangle \Big]\, d\tau =$$

$$= -\int\limits_{t_s}^{t_{s+1}} \frac{dB\left(x^*(\tau), \tau\right)}{d\tau} d\tau = B\left(x^*\left(t_s\right), t_s\right) - B\left(x^*\left(t_{s+1}\right), t_{s+1}\right).$$

Тоді, з урахуванням останньої рівності, оптимальне значення функціоналу дорівнює

$$J\left(u^*, x^*\right) = \int\limits_{t_0}^{T} f_0\left(x^*(\tau), u^*(\tau), \tau\right) d\tau + \Phi\left(x(T)\right) =$$

$$= \sum_{s=0}^{N-1} \int_{t_s}^{t_{s+1}} f_0 \left(x^*(\tau), \Psi_s \left(\tau, b_s^*, \, x^* \left(\tau \right) \right), \tau \right) d\tau + \Phi \left(x(T) \right) =$$

$$= \sum_{s=0}^{N-1} \left[B \left(x^* \left(t_s \right), t_s \right) - B \left(x^* \left(t_{s+1} \right), t_{s+1} \right) \right] + \Phi(x(T)).$$

Враховуючи, що $B\left(x^*\left(T\right), T\right) = \Phi\left(x(T)\right)$, отримуємо

$$J\left(u^*, x^*\right) = B\left(x^*\left(t_0\right), t_0\right) = B(x_0, t_0). \tag{7}$$

При довільних значеннях $b_s, s = 0, ..., N-1$ з (10) та (9) маємо

$$\int_{t_s}^{t_{s+1}} f_0 \left(x\left(\tau\right), \, \Psi_s \left(\tau, b_s, \, x\left(\tau\right) \right), \, \tau \right) d\tau \geq - \int_{t_s}^{t_{s+1}} \left[\frac{\partial B\left(x\left(\tau\right), \, \tau\right)}{\partial \tau} + \right.$$

$$\left. + \left\langle grad_x B\left(x\left(\tau\right), \, \tau\right), \, f\left(x\left(\tau\right), \, \Psi_s\left(\tau, \, b_s, \, x\left(\tau\right)\right), \, \tau\right)\right\rangle \right] \, d\tau =$$

$$= - \int_{t_s}^{t_{s+1}} \frac{dB\left(x(\tau), \tau\right)}{d\tau} d\tau = B\left(x\left(t_s\right), t_s\right) - B\left(x\left(t_{s+1}\right), t_{s+1}\right).$$

Тоді, аналогічно до попередніх міркувань, отримаємо нерівність

$$J\left(u, x\right) \geq B\left(x\left(t_0\right), t_0\right). \tag{8}$$

З (7) та (8) випливає, що $J\left(u, x\right) \geq J\left(u^*, x^*\right)$. □

2. Умови існування та єдиності розв'язку для задачі структурно-параметричної оптимізації лінійної системи керування

Розглянемо лінійну систему

$$\frac{dx\left(t\right)}{dt} = A\left(t\right)\, x\left(t\right) + C\left(t\right) u\left(t\right), \; x\left(t_0\right) = x_0, \tag{9}$$

де x — вектор фазових координат розмірності n, $x\left(t\right) \in X\left(t\right) \subseteq \mathbb{R}^n$, $t \in [t_0, T]$ — фазові обмеження, u — вектор керування розмірності m, $A\left(t\right)$ — матриця розмірності $n \times n$ з неперервними компонентами, $C\left(t\right)$ — матриця розмірності $n \times m$ з неперервними компонентами. Нехай керування задано у вигляді

$$u\left(t\right) = M_i(t)x(t) + N_i(t)b_i + q_i(t), \; t \in [t_i, \, t_{i+1}), \; i = 0, \, \ldots, N-1, \tag{10}$$

де $b_i \in R_i \subset \mathbb{R}^{k_i}$ — невідомі числові параметри, $M_i(t), N_i(t)$ — матриці з неперервними компонентами розмірностей $m \times n$ та $m \times k_i$ відповідно, $q_i(t)$ — вектори розмірності m з неперервними компонентами. Критерій якості має вигляд

$$J\left(x,\, u\right) = \Phi\left(x\left(T\right)\right). \tag{11}$$

Має місце наступна теорема.

Теорема 3. Якщо у задачі (9)–(11) множина X є опуклою, функція $\Phi(\cdot)$ є строго опуклою на множині X, а множини R_i для всіх $i = 0, ..., N-1$ є опуклими та компактними, то існує точка екстремуму функціоналу $J\left(x,\, u\right)$ за параметрами $b_i \in R_i$, $i = 0, ..., N-1$, причому ця точка єдина.

Доведення. Підставимо вираз (10) у систему (9). В результаті на кожному з проміжків часу $t \in \left[t_i, t_{i+1}\right)$, $i = 0,\, \ldots,\, N-1$ отримаємо таку систему

$$\frac{dx\left(t\right)}{dt} = A\left(t\right)\, x\left(t\right) + C\left(t\right)\left[M_i(t)x(t) + N_i(t)b_i + q_i(t)\right].$$

Перепишемо її у вигляді

$$\frac{dx\left(t\right)}{dt} = \left[A\left(t\right) + C(t)M_i(t)\right]\, x\left(t\right) + C\left(t\right)N_i(t)b_i + C(t)q_i(t),$$
$$t \in \left[t_i,\, t_{i+1}\right),\; i = 0,\, \ldots,\, N-1. \tag{12}$$

Запишемо розв'язок системи (12) за допомогою формули Коші. При $t \in [t_0, t_1]$ він має вигляд

$$x(t) = \Theta_0(t, t_0)x_0 + \int\limits_{t_0}^{t} \Theta_0(t, s)\left[C(s)N_0(s)b_0 + C(s)q_0(s)\right]ds. \tag{13}$$

Тут $\Theta_0(t, t_0)$ — нормована за моментом t_0 фундаментальна матриця системи (12) при $i = 0$. Далі, при $t \in [t_1, t_2]$ розв'язок системи має вигляд

$$x(t) = \Theta_1(t, t_1)x(t_1) + \int\limits_{t_1}^{t} \Theta_1(t, s)\left[C(s)N_1(s)b_1 + C(s)q_1(s)\right]ds. \tag{14}$$

Тут $\Theta_1(t, t_1)$ — нормована за моментом t_1 фундаментальна матриця системи (12) при $i = 1$. З (13) отримаємо вираз для $x(t_1)$

$$x(t_1) = \Theta_0(t_1, t_0)x_0 + \int\limits_{t_0}^{t_1} \Theta_0(t_1, s)\left[C(s)N_0(s)b_0 + C(s)q_0(s)\right]ds.$$

Підставимо цей вираз у (14). Маємо при $t \in [t_1, t_2]$

$$x(t) = \Theta_1(t, t_1)\Theta_0(t_1, t_0)x_0 + b_0\Theta_1(t, t_1)\int_{t_0}^{t_1} \Theta_0(t_1, s)C(s)N_0(s)ds+$$

$$+\Theta_1(t, t_1)\int_{t_0}^{t_1} \Theta_0(t_1, s)C(s)q_0(s)ds + b_1\int_{t_1}^{t} \Theta_1(t, s)C(s)N_1(s)ds + \int_{t_1}^{t} \Theta_1(t, s)C(s)q_1(s)ds.$$

Тоді

$$x(t_2) = \Theta_1(t_2, t_1)\Theta_0(t_1, t_0)x_0 + b_0\Theta_1(t_2, t_1)\int_{t_0}^{t_1} \Theta_0(t_1, s)C(s)N_0(s)ds+$$

$$+\Theta_1(t_2, t_1)\int_{t_0}^{t_1} \Theta_0(t_1, s)C(s)q_0(s)ds + b_1\int_{t_1}^{t_2} \Theta_1(t_2, s)C(s)N_1(s)ds + \int_{t_1}^{t_2} \Theta_1(t_2, s)C(s)q_1(s)ds.$$

Останній вираз показує, що $x(t_2)$ є лінійним за b_0 та b_1. Продовжуючи цей процес далі для $i = 2, 3, ..., N-1$, можна показати, що $x(t_i)$ лінійно залежить від параметрів $b_0, b_1, ..., b_{i-1}$. Це означає, що $x(T)$ лінійно залежить від усіх параметрів задачі $b_0, b_1, ..., b_{N-1}$. Враховуючи строгу опуклість функції $\Phi(\cdot)$, отримуємо, що функція $\Phi\left(x\left(T\right)\right)$ є строго опуклою на всіх множинах $R_i, i = 0, ..., N-1$ як суперпозиція лінійної та строго опуклої функцій. А це означає, з урахуванням опуклості та компактності множин R_i, що функціонал $J\left(x,\, u\right) = \Phi\left(x\left(T\right)\right)$ має екстремум за параметрами $b_i, i = 0, ..., N-1$, причому цей екстремум єдиний [4]. □

Нехай тепер критерій якості в задачі (9)–(10) має вигляд

$$J\left(x,\, u\right) = \int_{t_0}^{T} f_0\left(x\left(t\right),\, u\left(t\right),\, t\right)\, dt + \Phi\left(x\left(T\right)\right) \ \to\ \inf. \tag{15}$$

Для задачі (9), (10), (15) справедлива теорема, аналогічна теоремі 3.

Теорема 4. Якщо у задачі (9), (10), (15) множина X є опуклою, функція $f_0(x, u, t)$ є опуклою за x та строго опуклою за u, функція $\Phi(\cdot)$ є опуклою за x, а множини R_i для всіх $i = 0, ..., N-1$ є опуклими та компактними, то існує точка екстремуму функціоналу $J\left(x,\, u\right)$ за параметрами $b_i \in R_i$, $i = 0, ..., N-1$, причому ця точка єдина.

Доведення проводиться аналогічно до попередньої теореми.

3. Побудова оптимальної структури лінійної системи з квадратичним критерієм якості

Розглянемо лінійну систему (9) з керуванням вигляду (10). Нехай критерій якості має вигляд

$$J(x,u) = \int\limits_{t_0}^{T} \left(\langle D(\tau)x(\tau), x(\tau)\rangle + \langle E(\tau)u(\tau), u(\tau)\rangle\right) d\tau + \langle P_0 x(T), x(T)\rangle , \tag{16}$$

де $D(\tau), P_0$ — невід'ємновизначені симетричні матриці розмірності $n \times n$, $E(\tau)$ — додатновизначена симетрична матриця розмірності $n \times m$. Будемо шукати функцію Беллмана у вигляді квадратичної форми

$$B\left(z,\, t\right) = \left\langle P\left(t\right)\, z,\, z\right\rangle ,$$

де $P\left(t\right)$ — невідома матриця розмірності $n \times n$ з абсолютно неперервними компонентами. При цьому

$$\frac{\partial B\left(z,\,\tau\right)}{\partial \tau} = \left\langle \frac{dP\left(\tau\right)}{d\tau}\, z,\, z\right\rangle ,\; grad_z B\left(z,\,\tau\right) = \big(P\left(\tau\right) + P^T\left(\tau\right)\big)\; z.$$

З означення функції Беллмана випливає, що $P\left(T\right) = P_0$. Запишемо рівняння (5) для цього випадку. Одержуємо

$$\inf_{b_s\in R_s} \left\{ \int\limits_{t_s}^{t_{s+1}} \Bigg[\langle D(\tau)x(\tau), x(\tau)\rangle + \langle E(\tau)u(\tau), u(\tau)\rangle + \left\langle \frac{dP\left(\tau\right)}{d\tau} x\left(\tau\right),\, x\left(\tau\right)\right\rangle + \right.$$

$$\left. + \left\langle \big(P\left(\tau\right) + P^T\left(\tau\right)\big)\; x\left(\tau\right),\, A\left(\tau\right) x\left(\tau\right) + C\left(\tau\right) u\left(\tau\right)\right\rangle \Big]\; d\tau \right\} = 0.$$

Підставимо у останній вираз замість $u(\tau)$ вираз (10), зробимо перетворення та згрупуємо подібні члени

$$\inf_{b_s\in R_s} \left\{ \int\limits_{t_s}^{t_{s+1}} \left[\left\langle \frac{dP\left(\tau\right)}{d\tau} x\left(\tau\right), x\left(\tau\right)\right\rangle + \langle D(\tau)x(\tau), x(\tau)\rangle + \right.\right.$$

$$+ \langle E(\tau)M_s(\tau)x(\tau) + E(\tau)N_s(\tau)b_s + E(\tau)q_s(\tau), M_s(\tau)x(\tau) + N_s(\tau)b_s + q_s(\tau)\rangle +$$

$$+ \left\langle \left(A(\tau) + C(\tau)M_s(\tau)\right)^T P(\tau)x(\tau), x(\tau)\right\rangle + \left\langle \left(A(\tau) + C(\tau)M_s(\tau)\right)^T P^T(\tau)x(\tau), x(\tau)\right\rangle +$$

$$\left.\left. + \left\langle \big(P(\tau) + P^T(\tau)\big)\, x(\tau), q_s(\tau)\right\rangle + \left\langle N_s^T(\tau)C^T(\tau)\left(P(\tau) + P^T(\tau)\right) x(\tau), b_s\right\rangle \right] d\tau \right\} =$$

$$= \inf_{b_s\in R_s} \left\{ \int\limits_{t_s}^{t_{s+1}} \left[\left\langle \left(\frac{dP\left(\tau\right)}{d\tau} + D(\tau) + M_s^T(\tau)E(\tau)M_s(\tau) + \right.\right.\right.\right.$$

$$\left. + \left(A(\tau) + C(\tau)M_s(\tau)\right)^T P(\tau) + P(\tau)\left(A(\tau) + C(\tau)M_s(\tau)\right) \Big) x(\tau), x(\tau) \right\rangle +$$

$$+\left\langle \left(E(\tau)M_s(\tau)+E^T(\tau)M_s(\tau)+P(\tau)+P^T(\tau)\right)x(\tau)+E^T(\tau)q_s(\tau),q_s(\tau)\right\rangle+$$
$$+\left\langle \left(N_s^T(\tau)E(\tau)M_s(\tau)+N_s^T(\tau)E^T(\tau)M_s(\tau)+N_s^T(\tau)C^T(\tau)\left(P(\tau)+P^T(\tau)\right)\right)x(\tau)+\right.$$
$$\left.+N_s^T(\tau)E^T(\tau)q_s(\tau)+N_s^T(\tau)E(\tau)q_s(\tau),b_s\right\rangle+\left\langle N_s^T(\tau)E(\tau)N_s(\tau)b_s,b_s\right\rangle\Big]\,d\tau\Big\}=0.$$

Введемо позначення

$$Q_1^s(\tau)=\frac{dP(\tau)}{d\tau}+D(\tau)+M_s^T(\tau)E(\tau)M_s(\tau)+$$
$$+\left(A(\tau)+C(\tau)M_s(\tau)\right)^T P(\tau)+P(\tau)\left(A(\tau)+C(\tau)M_s(\tau)\right),$$
$$Q_2^s(\tau)=E(\tau)M_s(\tau)+E^T(\tau)M_s(\tau)+P(\tau)+P^T(\tau),$$
$$Q_3^s(\tau)=E^T(\tau)q_s(\tau),$$
$$Q_4^s(\tau)=N_s^T(\tau)E(\tau)M_s(\tau)+N_s^T(\tau)E^T(\tau)M_s(\tau)+N_s^T(\tau)C^T(\tau)\left(P(\tau)+P^T(\tau)\right),$$
$$Q_5^s(\tau)=N_s^T(\tau)E^T(\tau)q_s(\tau)+N_s^T(\tau)E(\tau)q_s(\tau),$$
$$Q_6^s(\tau)=N_s^T(\tau)E(\tau)N_s(\tau)$$

та перепишемо останню рівність у вигляді

$$\inf_{b_s\in R_s}\left\{\int_{t_s}^{t_{s+1}}\left[\langle Q_1^s(\tau)x(\tau),x(\tau)\rangle+\langle Q_2^s(\tau)x(\tau)+Q_3^s(\tau),q_s(\tau)\rangle+\right.\right.$$
$$\left.\left.+\langle Q_4^s(\tau)x(\tau)+Q_5^s(\tau),b_s\rangle+\langle Q_6^s(\tau)b_s,b_s\rangle\right]d\tau\right\}=0.$$

Нехай матриця $P(t)$ є такою, що $Q_1^s(t)=0$, тобто

$$\frac{dP(t)}{dt}=-D(t)-M_s^T(t)E(t)M_s(t)-\left(A(t)+C(t)M_s(t)\right)^T P(t)-P(t)\left(A(t)+C(t)M_s(t)\right),$$
$$t\in[t_s,t_{s+1}),\,P(T)=P_0. \tag{17}$$

Розв'яжемо задачу $\inf\limits_{b_s\in R_s} I=0$, де

$$I=\int_{t_s}^{t_{s+1}}\left[\langle Q_2^s(\tau)x(\tau)+Q_3^s(\tau),q_s(\tau)\rangle+\langle Q_4^s(\tau)x(\tau)+Q_5^s(\tau),b_s\rangle+\langle Q_6^s(\tau)b_s,b_s\rangle\right]d\tau.$$

Запишемо похідну

$$\frac{dI}{db_s}=\int_{t_s}^{t_{s+1}}\left[\left\langle Q_2^s(\tau)\frac{\partial x(\tau)}{\partial b_s},q_s(\tau)\right\rangle+\left\langle Q_4^s(\tau)\frac{\partial x(\tau)}{\partial b_s},b_s\right\rangle+\right.$$
$$\left.+Q_4^s(\tau)x(\tau)+Q_5^s(\tau)+\left(Q_6^s(\tau)+\left(Q_6^s\right)^T(\tau)\right)b_s\right]d\tau.$$

Позначимо $U_s(t) = \frac{\partial x(t)}{\partial b_s}$. Матриці чутливості $U_s(t)$, $s = 0, ..., N-1$ задовольняють системам лінійних диференціальних рівнянь

$$\frac{dU_s(t)}{dt} = [A(t) + C(t)M_s(t)]\, U_s(t) + C(t)N_s(t)\,,\; U_s(t_s) = 0,\; t \in [t_s, t_{s+1}]\,. \tag{18}$$

З умови $\frac{dI}{db_s} = 0$ випливає система лінійних алгебраїчних рівнянь відносно вектору параметрів b_s

$$\left(\int\limits_{t_s}^{t_{s+1}} \left[(U_s(\tau))^T (Q_4^s(\tau))^T + Q_6^s(\tau) + (Q_6^s(\tau))^T \right] d\tau \right) b_s =$$

$$= \int\limits_{t_s}^{t_{s+1}} \left[\langle Q_2^s(\tau) U_s(\tau),\, q_s(\tau) \rangle + Q_4^s(\tau) x(\tau) + Q_5^s(\tau) \right] d\tau. \tag{19}$$

Запишемо псевдорозв'язок системи (19)

$$b_s = \left(\int\limits_{t_s}^{t_{s+1}} \left[(U_s(\tau))^T (Q_4^s(\tau))^T + Q_6^s(\tau) + (Q_6^s(\tau))^T \right] d\tau \right)^+ \times$$

$$\times \left(\int\limits_{t_s}^{t_{s+1}} \left[\langle Q_2^s(\tau) U_s(\tau),\, q_s(\tau) \rangle + Q_4^s(\tau) x(\tau) + Q_5^s(\tau) \right] d\tau \right), \tag{20}$$

де W^+ — псевдообернена матриця до матриці W [2].

Тепер, переписуючи систему (9) з урахуванням структури керування (10) та формули (20), отримуємо, що на кожному відрізкові часу $[t_s, t_{s+1}]$ оптимальна траєкторія задовольняє інтегро-диференціальному рівнянню. Підставляючи оптимальну треєкторію у (20), знаходимо оптимальну структуру системи.

Висновки

Отже, в роботі одержано достатні умови оптимальності для задачі структурно-параметричної оптимізації з фіксованими точками перемикання, в якій структура містить фазову змінну. Для лінійної системи керування доведено умови існування та єдиності розв'язку задачі структурно-параметричної оптимізації, отримано алгоритм знаходження оптимальної структури у випадку квадратичного критерію якості.

Список літератури

1. Башняков О. М. Практична стійкість, оцінки та оптимізація / О. М. Башняков, Ф. Г. Гаращенко, В. В. Пічкур. — К.: Київський університет, 2008. — 383 с.
2. Беклемишев Д. В. Дополнительные главы линейной алгебры / Д. В. Беклемишев. — М.: Наука, 1983. — 336 с.
3. Бублик Б. Н. Структурно-параметрическая оптимизация и устойчивость динамики пучков / Б. Н. Бублик, Ф. Г. Гаращенко, Н. Ф. Кириченко — К.: Наукова думка, 1985. — 304 с.
4. *Карманов В. Г.* Математическое программирование / В. Г. Карманов. — М.: ФИЗМАТЛИТ, 2008. — 264 с.
5. Михалевич В. С. Методы последовательной оптимизации в дискретных сетевых задачах оптимального распределения ресурсов / В. С. Михалевич, А. И. Кукса — М.: Наука, 1983. — 208 с.
6. Моисеев Н. Н. Элементы теории оптимальных систем / Н. Н. Моисеев — М.: Наука, 1975. — 528 с.
7. Пічкур В. В. Застосування методу динамічного програмування до задачі структурно-параметричної оптимізації з фіксованими точками перемикання / В. В. Пічкур, Є. М. Страхов // Вісник Одеського національного університету. Математика і механіка. — 2010. — Т. 15, Вип. 19. — С. 94–102.
8. Розенвассер Е. Н. Чувствительность систем управления / Е. Н. Розенвассер, Р. М. Юсупов. — М.: Наука, 1981. — 464 с.
9. Филиппов А. Ф. Дифференциальные уравнения с разрывной правой частью / А. Ф. Филиппов. — М.: Наука, 1985. — 255 с.

Статья поступила в редакцию 16.03.2012

УДК 519.1

ВОЗМОЖНОСТЬ И СЛОЖНОСТЬ РАСПОЗНАВАНИЯ ГРАФОВ ТРЕМЯ АГЕНТАМИ

© А. В. Стёпкин

Институт прикладной математики и механики
Национальной академии наук Украины
ул. Розы Люксембург, 74, г. Донецк, Украина, 83114
e-mail: *stepkin.andrey@rambler.ru*

Abstract. Problem of exploration finite graphs by three agents is considered. Constructed an algorithm for exploration undirected graphs with $O\left(n^2\right)$ (n is amount of nodes of graph) time and space complexities. Two agents (which move on graph) needs two different colors (in total three colors) for graph exploration. An algorithm is based on depth-first traversal method.

Введение

В настоящее время важным и актуальным направлением кибернетики является распознавание неизвестной среды [1, 2]. В данной работе рассматривается проблема распознавания среды, заданной конечным графом [3, 4, 5], несколькими агентами. Ранее подобное распознавание рассматривалось в [6], в случае, когда два агента-исследователя (АИ) поочередно передвигаются по неизвестному конечному графу, обмениваются данными с агентом-экспериментатором (АЭ), который и восстанавливает исследуемый граф по данным, полученным от АИ.

Рассмотрим решение этой проблемы в случае, когда два агента-исследователя A и B одновременно передвигаются по неизвестной среде, заданной конечным графом (АИ могут окрашивать вершины, ребра и инциденторы графа, воспринимать эти отметки и на их основании осуществлять перемещение), которые обмениваются данными с АЭ (восстанавливает граф, по данным полученным от АИ, а также передает АИ информацию, необходимую им для дальнейшего функционирования). В работе предложен алгоритм построения маршрутов АИ по графу, позволяющих АЭ точно восстановить граф среды. Для этого каждому АИ требуется две краски: у A это r и b, у B — y и b. Алгоритм основан на методе обхода графа в глубину [7, 8]. При описании алгоритма используются результаты и обозначения из [6, 9].

Основным результатом данной работы является оптимизация алгоритма распознавания перешейков (ребра которые соединяют деревья, построенные различными АИ) и обратных ребер (ребра, которые соединяют несмежные вершины дерева, построенного АИ), что позволило, в итоге, улучшить временную сложность с $O\left(n^3\right)$ [6, 9] до $O\left(n^2\right)$, где n — число вершин графа.

1. Основные определения и обозначения

В работе рассматриваются конечные, неориентированные графы без петель и кратных ребер. Понятия, которые не рассмотрены ниже, общеизвестны и с ними можно ознакомиться в [7, 8, 10].

Пусть $G = (V, E)$ — граф, где V — множество вершин, E — множество ребер (двухэлементных подмножеств (v, u), где $v, u \in V$). Тройку $((v, u), u)$ будем называть инцидентором ребра (v, u) и вершины u. Множество таких троек обозначим I. Множество $L = V \cup E \cup I$ назовем множеством элементов графа G. Функцией раскраски графа G назовем сюръективное отображение $\mu : L \to \{w, r, y, ry, b\}$, где w интерпретируется как белый цвет, r — красный, y — желтый, ry — красно-желтый b — черный. Пара (G, μ) называется раскрашенным графом. Последовательность $u_1, u_2, ...u_k$ попарно смежных вершин называется путем в графе G, а k — длинной пути. При условии $u_1 = u_k$ путь называется циклом. Окрестностью $Q(v)$ вершины v будем называть множество элементов графа, состоящее из вершины v, всех вершин u смежных с v, всех ребер (v, u) и всех инциденторов $((v, u), v), ((v, u), u)$. Мощность множеств вершин V и ребер E обозначим через n и m соответственно. Ясно что $m \leq \frac{n(n-1)}{2}$. Изоморфизмом графа G и графа H назовем такую биекцию $\varphi : V_G \to V_H$, что $(v, u) \in E_G$ точно тогда, когда $(\varphi(v), \varphi(u)) \in E_H$. Таким образом, изоморфные графы равны с точностью до обозначения вершин и раскраски их элементов.

Мобильные агенты A и B в начале работы помещаются в произвольные несовпадающие вершины графа G, которые сразу добавляются АЭ в множество вершин V_H. Агенты передвигаются по графу из вершины v в вершину u по ребру (v, u), могут изменять окраску вершин v, u, ребер (v, u), инциденторов $((v, u), v), ((v, u), u)$. Находясь в вершине v, АИ воспринимает метки всех элементов окрестности $Q(v)$ и на основании этих меток определяет, по какому ребру будет дальше перемещаться, и как будет окрашивать элементы графа. АЭ передает, принимает и идентифицирует сообщения, полученные от АИ, обладает конечной, неограниченно растущей внутренней памятью, в которой фиксируется результат функционирования АИ на каждом шаге и, кроме того, строится представление графа G, вначале неизвестного агентам, списками ребер и вершин.

2. Стратегия решения

Принцип работы рассматриваемого алгоритма основан на методе поиска в глубину и заключается в том, что АИ идут «в глубину», пока это возможно, возвращаются назад, ищут другой путь с еще не посещенными вершинами и не пройденными ребрами.

Рассмотрим подробно режимы работы АИ. При описании режимов, в скобках указываются сообщения, которые отправят АИ, попав в рассматриваемую ситуацию («СООБЩЕНИЕ_АГЕНТА_А»; «СООБЩЕНИЕ_АГЕНТА_В»).

1. *Обычный режим работы.* (ОРР) Работая в этом режиме, АИ двигается вперед по белым вершинам, окрашивая эти вершины, соединяющие их ребра и дальние инциденторы в «свой» цвет («ВПЕРЕД_А»; «ВПЕРЕД_В»). Если белый путь и другие возможные пути перемещения отсутствуют, то АИ возвращается назад по своему пути, окрашивая пройденные вершины, соединяющие их ребра и ближние инциденторы в черный цвет («НАЗАД_А»; «НАЗАД_В»). АИ завершает работу тогда, когда его исходная вершина, вследствие отсутствия возможных путей перемещения, окрашивается в черный цвет («СТОП_А», «СТОП_В»).

2. *Режим распознавания обратных ребер.* (РРОР) Если, при движении вперед, в вершине v было обнаружено обратное ребро, то АИ переключается в режим распознавания обратных ребер и красит в «свой» цвет ближние инциденторы всех обратных ребер инцидентных вершине v («МЕТКА_ОР_А»; «МЕТКА_ОР_В»). Завершив покраску инциденторов, АИ передвигается назад по своему пути (каждый шаг назад, отправляется сообщение «ОТСТУП_А»; «ОТСТУП_В»), до обнаружения вершины инцидентной помеченному обратному ребру (под помеченным обратным ребром понимается белое ребро, у которого дальний инцидентор и дальняя вершина окрашены в «свой» цвет), переходит по этому ребру, окрашивая его в черный цвет («ВПЕРЕД_ОР_А»; «ВПЕРЕД_ОР_В»). На этом этапе возможны два случая:

2.1. Распознаны не все, помеченные рассматриваемым АИ, обратные ребра. В этом случае АИ возвращается назад по пройденному на предыдущем шаге ребру, окрашивая в черный цвет ближний инцидентор, и продолжает движение назад по своему пути (каждый шаг назад, отправляется сообщение «ОТСТУП_А»; «ОТСТУП_В»), до обнаружения следующей вершины, инцидентной помеченному обратному ребру.

2.2. Распознаны все, помеченные рассматриваемым АИ, обратные ребра. В этом случае АИ окрашивает ближний инцидентор ребра, по которому он перешел на предыдущем шаге, в черный цвет («РЕБРА_РАСПОЗНАНЫ_А»; «РЕБРА_РАСПОЗНАНЫ_В») и завершает работу в режиме распознавания обратных ребер.

3. *Режим пометки перешейков.* (РПП) Если в процессе обхода графа в вершине v был обнаружен перешеек, то при условии, что все ранее помеченные данным АИ перешейки были распознаны, агент переключается в режим пометки перешейков (если второй АИ ещё не распознал ранее помеченные перешейки, то первый АИ не может

метить новые перешейки и в случае, когда у первого АИ отсутствуют другие возможные варианты перемещения, кроме как пометить новый перешеек, он останавливается до того момента, пока второй АИ не распознает все помеченные перешейки). В этом режиме АИ окрашивает ближние инциденторы всех перешейков, инцидентных вершине v, в черный цвет (на каждый окрашенный инцидентор отправляется одно сообщение «МЕТИМ_АВ»; «МЕТИМ_ВА»). Когда все перешейки помечены работа АИ в данном режиме завершается («ФИКС_А»; «ФИКС_В»). По завершению режима пометки перешейков АЭ содержит информацию о количестве помеченных перешейков. В данном режиме работы агент A имеет приоритет над агентом B, поэтому в ситуации, когда оба АИ одновременно обнаружат один и тот же перешеек, он будет помечен агентом A.

4. *Режим распознавания перешейков.* (РРП) Получив от АЭ команду о необходимости распознавания перешейков, АИ переключается в режим распознавания перешейков. Если в этот момент агент работает в РРОР, то АИ переключится в РРП лишь по завершению распознавания обратных ребер. При переключении в этот режим АИ проверяет наличие из вершины, в которой он находится, других возможных путей перемещения кроме как движение назад по своему пути. Если такие пути есть, то АИ возвращается назад по своему пути, ничего не окрашивая (каждый шаг назад, отправляется сообщение «ОТСТУП_А»; «ОТСТУП_В»), до обнаружения ближайшей вершины, инцидентной помеченному перешейку (под помеченным перешейком понимается белое ребро, у которого дальний инцидентор окрашен в черный цвет, а дальняя вершина окрашена в «чужой» цвет). Если же таких путей нет, то возвращаясь назад по своему пути, АИ окрашивает его в черный цвет (каждый шаг назад, отправляется сообщение «НАЗАД_А»; «НАЗАД_В») до тех пор, пока не попадет в вершину инцидентную помеченному перешейку либо же в вершину с другими возможными путями перемещения. Во втором случае АИ продолжает возвращаться назад по своему пути ничего не окрашивая (каждый шаг назад, отправляя сообщение «ОТСТУП_А»; «ОТСТУП_В») до обнаружения ближайшей вершины, инцидентной помеченному перешейку.

При обнаружении помеченного перешейка возможны два случая:

4.1. Помечено один перешеек. АИ окрашивает ближний инцидентор помеченного перешейка в черный цвет («РАСП_АВ»; «РАСП_ВА»). Далее движется вперед в конец пути «своего» цвета.

4.2. Помечено не менее двух перешейков. АИ окрашивает ближний инцидентор помеченного перешейка в «свой» цвет («РАСП_АВ»; «РАСП_ВА»). Далее АИ движется назад по своему пути (каждый шаг назад, отправляется сообщение «ОТСТУП_A»; «ОТСТУП_B»), пока не будет найден следующий помеченный перешеек. При обнаружении помеченного перешейка возможно два варианта:

4.2.1. Следующий помеченный перешеек не последний. АИ окрашивает ближний инцидентор в черный цвет («РАСП_AB»; «РАСП_BA»). На следующем шаге АИ снова возвращается назад по своему пути до следующего помеченного перешейка (каждый шаг назад, отправляется сообщение «ОТСТУП_A»; «ОТСТУП_B»).

4.2.2. Следующий помеченный перешеек последний. АИ окрашивает ближний инцидентор в «свой» цвет («РАСП_AB»; «РАСП_BA»). На следующем шаге АИ сообщает АЭ о распознавании всех помеченных другим АИ перешейков («ОБН_A»; «ОБН_B»). Далее переходит по последнему перешейку в чужую область, окрашивая ближний инцидентор в черный цвет. На следующем шаге АИ переходит по первому распознанному перешейку в свою область, окрашивая дальний инцидентор в черный цвет. Далее АИ движется вперед в конец пути «своего» цвета.

5. *Одновременное попадание двух АИ в одну белую вершину.* При одновременном попадании двух АИ в одну белую вершину, каждый АИ окрашивает вершину наполовину, и она становится красно-желтой. Агент B на следующем шаге отступает назад по своему пути. При этом удаляет краску с ближнего инцидентора и ребра, окрашивает дальний инцидентор в черный цвет («ВОЗВРАТ_B») и переходит в режим пометки перешейков (при этом ребро, по которому он вернулся уже посчитано как первый перешеек, а длина желтого пути уменьшена на одну вершину). Агент A видит разноцветную вершину как свою, но при распознавании окрашивает в черный цвет обе половинки.

Если АИ попадает в ситуацию, когда в вершине возможен выбор сразу нескольких режимов работы, то первым будет выбран РРП, потом РПП, за ним РРОР и наконец ОРР. Режим работы при попадании двух АИ в одну белую вершину в этом списке не рассматривается потому, что такая ситуация приведет к изменениям в работе только агента B и, в этот момент, другие режимы работы для него будут недоступны.

Выполняя обход графа, агенты A и B создают соответственно красный и желтый пути. Рассмотрим принцип, по которому АИ строит путь «своего» цвета. При движении в белую вершину красный (желтый) путь удлиняется, при движении назад по своему пути — укорачивается. При движении АИ назад, для распознавания обратных ребер или перешейков, длина пути не изменяется. Вершина, из которой возможно лишь движение назад по своему пути, либо вовсе отсутствуют варианты перемещения, окрашивается в черный цвет. Алгоритм заканчивает работу, когда красный и желтый пути становятся пустыми, а все вершины черными.

Выполняя обход графа G агенты создают неявную нумерацию посещенных вершин. Первый раз посетив вершину агент A окрашивает её в красный цвет (агент B - в желтый цвет), и АЭ ставит ей в соответствие номер, равный значению переменной $Сч\_A$ ($Сч\_B$ для агента B). Отметим что переменные $Сч\_A$ и $Сч\_B$ принимают соответственно нечетные и четные значения. Восстановление графа G происходит на основе созданной агентами нумерации путем построения графа H изоморфного G.

Рассмотрим алгоритм работы АЭ:

Вход: списки сообщений M и N от АИ.

Выход: список вершин V_H и ребер E_H графа H, изоморфного графу G.

Данные: V_H, E_H списки вершин и ребер графа H, изоморфного графу G. $Сч\_A$, $Сч\_B$ — счетчики числа посещенных вершин графа G агентами A и B соответственно. AN — переменная, в которой значение «1» дает агенту A сигнал для возврата и распознания помеченных агентом B перешейков, значение «0» позволяет агенту A работать дальше в обычном режиме. BN — переменная, в которой значение «1» дает агенту B сигнал для возврата и распознания помеченных агентом A перешейков, значение «0» позволяет агенту B работать дальше в обычном режиме. N_A, N_B — переменные, в которых хранятся номера вершин, из которых агенты A и B соответственно, последний раз помечали перешейки. F — количество перешейков из вершины N_A, помеченных для распознания. K — количество перешейков из вершины N_B, помеченных для распознания. M, N — списки сообщений от агентов A и B соответственно. E — переменная, в которой делается отметка о том, был ли на предыдущем шаге агентом A помечен перешеек (значение «1») или нет (значение «0»). L — переменная, в которой делается отметка о том, был ли на предыдущем шаге агентом B помечен перешеек (значение «1») или нет (значение «0»). i, j — счетчики используемые агентами A и B соответственно при подсчете номеров вторых вершин помеченных перешейков и номеров вторых вершин помеченных обратных ребер. $STOP\_A, STOP\_B$ — переменные, используемые агентами A и B соответственно, для сигнализации АЭ, о завершении распознавания своей области. UDP_A, UDP_B — логические переменные, используемые агентами A и B соответственно, для определения способа окраски инциденторов, рассматриваемого в определенный момент, перешейка. $UDOBR\_A, UDOBR\_B$ — логические переменные, используемые агентами A и B соответственно для определения является ли рассматриваемое обратное ребро последним из помеченных. $KOBR\_A, KOBR\_B$ —

переменные, в которые агенты A и B соответственно записывают количество помеченных обратных ребер. $r(1), r(2), ..., r(t)$ — список номеров вершин красного пути, где t – длинна этого списка. $y(1), y(2), ..., y(p)$ — список номеров вершин желтого пути, где p — длинна этого списка. Mes — текущее сообщение.

1. $Сч\_A := 1;$
2. $Сч\_B := 2;$
3. $AN := 0, BN := 0, N\_A := 0, N\_B := 0, F := 0, K := 0, M := \varnothing, N := \varnothing, E := 0,$
   $L := 0, i := 0, j := 0, E_H := \varnothing, STOP\_A := 0, STOP\_B := 0,$
   $UDP\_A := FALSE, UDP\_B := FALSE, UDOBR\_A := FALSE,$
   $UDOBR\_B := FALSE, KOBR\_A := 0, KOBR\_B := 0;$
4. $t := 1;$
5. $p := 1;$
6. $r(t) := Сч\_A;$
7. $y(p) := Сч\_B;$
8. $V_H := \{1, 2\};$
9. *while* $(STOP\_A = 0)$ *or* $(STOP\_B = 0)$ *do*
10. *if* $M \neq \varnothing$ *then do*
11. прочитать в Mes сообщение и удалить его из очереди M;
12. *ОБР_СП_A*();
13. *end do*;
14. *if* $N \neq \varnothing$ *then do*
15. прочитать в Mes сообщение и удалить его из очереди N;
16. *ОБР_СП_B*();
17. *end do*;
18. *end do*;
19. печать V_H, E_H.

ОБР_СП_A():

1. *if* $Mes =$ "ВПЕРЕД_A" *then ВПЕРЕД_A*();
2. *if* $Mes =$ "МЕТИМ_AB" *then МЕТИМ_AB*();
3. *if* $Mes =$ "НАЗАД_A" *then НАЗАД_A*();
4. *if* $Mes =$ "РАСП_AB" *then РАСП_AB*();
5. *if* $Mes =$ "ФИКС_A" *then ФИКС_A*();
6. *if* $Mes =$ "ОБН_A" *then ОБН_A*();
7. *if* $Mes =$ "РЕБРА_РАСПОЗНАНЫ_A" *then РЕБРА_РАСПОЗНАНЫ_A*();
8. *if* $Mes =$ "ОТСТУП_A" *then ОТСТУП_A*();
9. *if* $Mes =$ "МЕТКА_ОР_A" *then МЕТКА_ОР_A*();

10. *if* $Mes =$ "ВПЕРЕД_ОР_А" *then* $ВПЕРЕД\_OP\_A()$;
11. *if* $Mes =$ "СТОП_А" *then* $СТОП\_A()$.

$ВПЕРЕД\_A()$: выполняются операции: $Сч\_A{:=}Сч\_A{+}2$; $t := t + 1$; $r\,(t) := Сч\_A$; $V_H := V_H \cup \{Сч\_A\}$; $E_H := E_H \cup \{(r\,(t-1)\,, r\,(t))\}$;

$МЕТИМ\_AB()$: $F := F + 1$; $E := 1$;

$НАЗАД\_A()$: из списка $r(1), ..., r(t)$ удаляется элемент $r(t)$; $t := t - 1$;

$РАСП\_AB()$: $E_H := E_H \cup \{(N\_B, r\,(t-i))\}$; $K := K - 1$; $UDP\_B := (((K = Z)\,or\,(K = 1))\,and\,(Z \neq 1))$;

$ФИКС\_A()$: $N\_A := Сч\_A$; $BN := 1$; $E := 0$; $Q := F$; $UDP\_A := (((F = Q)\,or\,(F = 1))\,and\,(Q \neq 1))$;

$ОБН\_A()$: $AN := 0$; $i := 0$;

$РЕБРА\_РАСПОЗНАНЫ\_A()$: $i := 0$;

$ОТСТУП\_A()$: $i := i + 1$;

$МЕТКА\_OP\_A()$: $KOBR\_A := KOBR\_A + 1$;

$ВПЕРЕД\_OP\_A()$:$KOBR\_A := KOBR\_A - 1$; $UDOBR\_A := (KOBR\_A = 0)$; $E_H := E_H \cup \{(r\,(t)\,, r\,(t-i))\}$;

$СТОП\_A()$: $STOP\_A := 1$;

Процедуры работы со списком сообщений от агента B, которые не рассмотрены ниже, аналогичны процедурам работы со списком сообщений от агента A.

$ОБР\_СП\_B()$:
1. *if* $Mes =$ "ВПЕРЕД_В" *then* $ВПЕРЕД\_B()$;
2. *if* $Mes =$ "МЕТИМ_ВА" *then* $МЕТИМ\_BA()$;
3. *if* $Mes =$ "НАЗАД_В" *then* $НАЗАД\_B()$;
4. *if* $Mes =$ "РАСП_ВА" *then* $РАСП\_BA()$;
5. *if* $Mes =$ "ФИКС_В" *then* $ФИКС\_B()$;
6. *if* $Mes =$ "ОБН_В" *then* $ОБН\_B()$;
7. *if* $Mes =$ "РЕБРА_РАСПОЗНАНЫ_В" *then* $РЕБРА\_РАСПОЗНАНЫ\_B()$;
8. *if* $Mes =$ "ОТСТУП_В" *then* $ОТСТУП\_B()$;
9. *if* $Mes =$ "МЕТКА_ОР_В" *then* $МЕТКА\_OP\_B()$;
10. *if* $Mes =$ "ВПЕРЕД_ОР_В" *then* $ВПЕРЕД\_OP\_B()$;
11. *if* $Mes =$ "ВОЗВРАТ_В" *then* $ВОЗВРАТ\_B()$;
12. *if* $Mes =$ "СТОП_В" *then* $СТОП\_B()$.

$ВОЗВРАТ\_B()$: $E_H := E_H \backslash \{(y\,(p-1)\,, y\,(p))\}$; $V_H := V_H \backslash \{Сч\_B\}$; $Сч\_B{:=}\ Сч\_B{-}2$; $p := p - 1$; $y\,(p) := Сч\_B$; $L := 1$; $.K := K + 1$;

3. Свойства алгоритма распознавания

В начале работы алгоритма распознавания, при $n \geq 3$, как минимум, по одному разу выполняются процедуры: ВПЕРЕД_А() и ВПЕРЕД_В(). Эти сообщения передаются агенту-экспериментатору, когда АИ посещают белые вершины исследуемого графа G. Процедурами агента АЭ ВПЕРЕД_А() и ВПЕРЕД_В() создаются две новые вершины (по одной вершине для каждой из процедур) графа H.

При одновременном попадании двух АИ в одну белую вершину процедурами ВПЕРЕД_А() и ВПЕРЕД_В() будет создано две новые вершины графа H. Вершина созданная агентом B, на следующем шаге будет удалена процедурой ВОЗВРАТ_В(), так как она дублирует вершину, созданную агентом A. Таким образом, процесс выполнения описанного алгоритма индуцирует отображение $\varphi : V_G \to V_H$ вершин графа G в вершины графа H. Причем $\varphi(v) = t$ (когда вершина v окрашена в красный цвет и $t = Сч\_A$ и $\varphi(s) = p$ (когда вершина s окрашена в желтый цвет и $p = Сч\_B$). Указанное отображение естественным образом устанавливает неявную нумерацию вершин графа G. Более того, отображение φ является биекцией, поскольку в связном графе G все вершины достижимы из начальных вершин. Поэтому все вершины посещаются агентами, то есть окрашиваются в красный и желтый цвета.

Из описания алгоритма следует, что АИ проходят все ребра графа G, поскольку по окончании алгоритма все ребра становятся черными. При выполнении процедуры ВПЕРЕД_А() или ВПЕРЕД_В() АЭ распознает древесное ребро (v, u) и так нумерует вершину u, что ребру (v, u) однозначно соответствует ребро $(\varphi(v), \varphi(u))$ графа H. При выполнении процедур *ВПЕРЕД_ОР_А*() или *ВПЕРЕД_ОР_В*() АЭ распознает обратное ребро (v, u) графа G и ставит ему в однозначное соответствие ребро $(\varphi(v), \varphi(u))$ графа H. При выполнении процедур *РАСП_АВ*() или *РАСП_ВА*() АЭ распознает перешеек (v, u) графа G и ставит ему в однозначное соответствие ребро $(\varphi(v), \varphi(u))$ графа H. Следовательно, φ является изоморфизмом графа G на граф H.

Теорема 1. *Три агента, выполнив алгоритм распознавания на графе G, распознают этот граф с точностью до изоморфизма.*

Подсчитаем временную и емкостную сложности алгоритма в равномерной шкале [8]. Рассмотрим подробнее свойства красного и желтого путей. Из описания алгоритма следует, что на каждом шаге алгоритма красный (желтый) путь — это простой путь, соединяющий начальную вершину v (s — в случае агента B) с номером $\varphi(v) = 1(\varphi(s) = 2)$ с вершиной u (z) с номером $\varphi(u) = Сч\_A$ ($\varphi(z) = Сч\_B$). Следовательно, общая длина красного и желтого пути не превосходит n.

Каждый раз при выполнении процедур из ОРР каждый АИ проходит одно ребро. При однократном выполнении процедуры из РРОР АИ метят не более $n-2$ (изначально одна вершина уже окрашена в «чужой» цвет, поэтому обратного ребра в неё быть не может) обратных ребер, по одному разу проходит не более $n-2$ ребер красного (желтого) пути, а так же по два раза проходится не более $n-2$ обратных ребер. При выполнении процедур из РПП АИ не совершают перехода по перешейкам, а просто окрашивают их ближние инциденторы, после чего, не делая передвижений, отправляют сообщение АЭ о завершении РПП, на что так же уходит один ход. При однократном выполнении процедуры из РРП АИ проходят не более $n-2$ ребер красного (желтого) пути, после чего помечая перешейки как распознанные АИ не совершают перехода по перешейку, а просто окрашивают его ближний инцидентор. Завершив покраску всех помеченных перешейков АИ уведомляет об этом АЭ, на что так же уходит один ход. Возвращаясь после распознавания всех перешейков АИ может пройти не более чем по одному разу два перешейка (либо не пройти ни одного перешейка в случае, когда был только один помеченный перешеек), а так же не более чем $n-2$ ребер красного (желтого) пути.

При подсчете временной сложности алгоритма будем считать, что инициализация алгоритма, анализ окрестности $Q\left(v\right)$ рабочей вершины и выбор одной из возможных процедур занимают некоторое постоянное число единиц времени. Так же будем считать, что выбор ребер, проход по ним АИ и обработка сообщений АЭ полученных на данном этапе от АИ осуществляется за 1 единицу времени. Тогда временная сложность алгоритма определяется следующими соотношениями:

1. Процедуры из ОРР выполняются не более чем $2\times(n-1)$ раз, общее время их выполнения оценивается как $O\left(n\right)$.
2. Время, затрачиваемое на работу в РРОР оценивается как $n\times 4\times O(n)$, то есть как $O\left(n^2\right)$.
3. На работу в РПП уходит время, оцениваемое как $n\times O\left(n\right)+O\left(n\right)=O\left(n^2\right)$.
4. Время выполнения РРП оценивается как
   $O\left(n^2\right)+O\left(n^2\right)+O\left(n\right)+O\left(n\right)+O\left(n^2\right)=O\left(n^2\right)$
5. Время простоя агентов при ожидании в общей сложности оценивается как
   $O\left(n\right)+O\left(n^2\right)=O\left(n^2\right)$.

Следовательно, суммарная временная сложность $T(n)$ алгоритма удовлетворяет соотношению: $T\left(n\right)=O\left(n^2\right)$.

Емкостная сложность $S(n)$ алгоритма определяется сложностью списков $V_H, E_H, r(1)...r(t), y(1)...y(p)$, сложность которых соответственно определяется величинами $O\left(n\right), O\left(n^2\right), O\left(n\right), O\left(n\right)$. Следовательно: $S\left(n\right)=O\left(n^2\right)$.

Теорема 2. *Временная и емкостная сложности алгоритма равны $O(n^2)$, где n — число вершин графа, при этом алгоритм использует 3 краски.*

Заключение

В работе предложен новый алгоритм распознавания графа среды временной и емкостной сложностей $O(n^2)$. АИ имеют конечную память, независимую от n, и используют по две краски каждый (всего три краски).

Автор выражает глубокую признательность своему научному руководителю к.ф.-м.н., с.н.с ИПММ НАНУ Грунскому И. С. за оказанную помощь в работе.

Список литературы

1. Кудрявцев В. Б. О поведении автоматов в лабиринтах / В. Б. Кудрявцев, Щ. Ушчумлич, Г. Калибарда // Дискретная математика. — 1992. — т.4, №3 — С. 3–28.
2. Albers S. Exploring unknown environments / S. Albers, M. R. Henzinger // SIAM Journal on Computing. — 2000. — 29(4). — P. 1164–1188.
3. Кудрявцев В. Б. Введение в теорию автоматов / В. Б. Кудрявцев, С. В. Алешин, А. С. Подколзин. — М.: Наука, 1985. — 320 с.
4. Грунский И. С. Эксперименты с помеченными графами / И. С. Грунский, С. В. Сапунов, Е. А. Татаринов // Дискретные модели в теории управляющих систем: электронный сборник материалов VIII международной конференции (г. Москва МГУ, 2009 р.). — Москва, 2009. — С. 43–44.
5. Грунский И. С. Распознавание конечного графа блуждающим по нему агентом / И. С. Грунский, Е. А. Татаринов // Вестник Донецкого университета. Серия А. Естественные науки. — 2009. — Вып. 1. — С. 492–497.
6. Грунский И. С. Распознавание конечного графа коллективом агентов. / И. С. Грунский, А. В. Стёпкин // Труды ИПММ НАН Украины. — 2009. — Т.19. — С. 43–52.
7. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. — М.: МЦНМО, 2001. — 960 с.
8. Ахо А. Построение и анализ вычислительных алгоритмов / А. Ахо, Дж. Хопкрофт, Дж. Ульман. — М.: Мир, 1979. — 536 с.
9. Стёпкин А. В. Распознавание конечных графов тремя агентами / А. В. Стёпкин // Искусственный интеллект. — 2011. — № 2. — С. 84–93.
10. Касьянов В. Н. Графы в программировании: обработка, визуализация и применение / В. Н. Касьянов, В. А. Евстигнеев. — СПБ.: БХВ — Петербург, 2003. — 1104 с.

Статья поступила в редакцию 10.01.2012

Анафиев А. С. Подход к решению задач оптимизации с прецедентной начальной информацией / А. С. Анафиев // Таврический вестник информатики и математики. — 2012. — № 1 (20). — С. 7-12.

УДК 519.7

Запропонована постановка задачі оптимізації з прецедентною початковою інформацією. Виділені основні проблеми та завдання побудови надійних схем розв'язання подібного роду оптимізаційних задач. Описаний підхід на основі функції втрат для розв'язання таких задач. Розглянутий приклад на основі метричних алгоритмів класифікації. Виділений новий клас задач спільного навчання за прецедентами.

Предложена постановка задачи оптимизации с прецедентной начальной информацией. Выделены основные проблемы и задачи построения надежных схем решения подобного рода оптимизационных задач. Описан подход на основе функции потерь для решения таких задач. Рассмотрен пример на основе метрических алгоритмов классификации. Выделен новый класс задач совместного обучения по прецедентам.

Донской В. И. Синтез согласованных линейных оптимизационных моделей по прецедентной информации: подход на основе колмогоровской сложности / В. И. Донской // Таврический вестник информатики и математики. — 2012. — № 1 (20). — С. 13-23.

УДК 519.95

У статті викладено підхід до аналізу лінійних оптимізаційних моделей, побудованих за прецедентною початковою повчальною інформацією. У припущенні, що всі числові параметри є раціональними і обмежені розрядною сіткою комп'ютера, одержані оцінки колмогоровської складності і невипадковісті витягання моделі з даних як емпіричної закономірності.

В статье изложен подход к анализу линейных оптимизационных моделей, построенных по прецедентной начальной обучающей информации. В предположении, что все числовые параметры являются рациональными и ограничены разрядной сеткой компьютера, получены оценки колмогоровской сложности и неслучайности извлечения модели из данных как эмпирической закономерности.

Ємець О. О. Математична модель регіону: еколого-економічний аспект / О. О. Ємець, О. О. Черненко // Таврический вестник информатики и математики. — 2012. — № 1 (20). — С. 24-34.

УДК 519.8

У роботі розглянуто проблему екологічної безпеки виробництва на регіональному рівні. Побудовано модель функціонування регіону з врахуванням техногенного навантаження на навколишнє середовище та раціонального використання природних ресурсів.

В работе рассмотрена проблема экологической безопасности производства на региональном уровне. Построена модель функционирования региона с учетом техногенной нагрузки на окружающую среду и рационального использования природных ресурсов.

Ильченко А. В. Минимальные по включению деревья Штейнера: алгоритм построения / А. В. Ильченко, В. Ф. Блыщик // Таврический вестник информатики и математики. — 2012. — № 1 (20). — С. 35-43.

УДК 519.6

Розглядається поняття мінімального по включенню дерева Штейнера. Наводиться і обгрунтовується алгоритм побудови усіх мінімальних по включенню дерев Штейнера. Мінімальне по включенню дерево Штейнера найменшої ваги розглядається як рішення задачі Штейнера на графі.

Рассматривается понятие минимального по включению дерева Штейнера. Приводится и обосновывается алгоритм построения всех минимальных по включению деревьев Штейнера. Минимальное по включению дерево Штейнера наименьшего веса рассматривается как решение задачи Штейнера на графе.

Лемтюжникова Д. В. Алгоритм выделения блочно-древовидной структуры в разреженных задачах дискретной оптимизации / Д. В. Лемтюжникова, А. В. Свириденко, О. А. Щербина // Таврический вестник информатики и математики. — 2012. — № 1 (20). — С. 44-55.

УДК 519.658

У статті запропонований алгоритм виділення блочно-деревоподібної структури для розріджених матриць. Реалізований у вигляді програми на C++ і протестований алгоритм Фінкельштейна для виділення квазіблочних структур в розріджених матрицях. Зроблено порівняльний експеримент для модифікованої та вихідної версій алгоритму, який показав істотне зменшення кількості побудованих блоків і розмірів сепараторів для модифікованого алгоритму Фінкельштейна.

В статье предложен алгоритм выделения блочно-древовидной структуры для разреженных матриц. Реализован в виде программы на C++ и протестирован алгоритм Финкельштейна для выделения квазиблочных структур в разрежённых матрицах. Произведен сравнительный эксперимент для модифицированной и исходной версий алгоритма, показавший существенное уменьшение количества построенных блоков и размеров сепараторов для модифицированного алгоритма Финкельштейна.

Лемтюжникова Д. В. О распараллеливании локального элиминационного алгоритма / Д. В. Лемтюжникова, О. А. Щербина // Таврический вестник информатики и математики. — 2012. — № 1 (20). — С. 56-65.

УДК 519.658

Метою цієї роботи є визначення наступних стратегій розпаралелювання локального елімінаційного алгоритму для розріджених задач дискретної оптимізації на основі використання сучасних обчислювальних архітектур. Незалежні підзадачі, які відповідають різним блокам, і підзадачі, відповідні незалежним гілкам узагальненого елімінаційного дерева, можуть вирішуватися паралельно за допомогою таких обчислювальних архітектур, як багатоядерні процесори, графічні процесори (GPU) і GRID. Для паралельної реалізації локального елімінаційного алгоритму пропонується використовувати гібридну схему «майстер-робітник», яка допускає одночасне використання CPU і GPU, причому GPU, множина основних паралельних машин із загальною пам'яттю, виступають в якості робочих процесорів і виконують рішення підзадач ДО для блоків, а CPU використовується як майстер.

Целью настоящей работы служит определение следующих стратегий распараллеливания локального элиминационного алгоритма для разреженных задач дискретной оптимизации на основе использования современных вычислительных архитектур. Независимые подзадачи, соответствующие разным блокам, и подзадачи, соответствующие независимым ветвям обобщенного элиминационного дерева, могут решаться параллельно при помощи таких вычислительных архитектур, как многоядерные процессоры, графические процессоры (GPU) и GRID. Для параллельной реализации ЛЭА предлагается использовать гибридную схему «мастер-рабочий», которая допускает одновременное использование CPU и GPU, причем GPU, множество основных параллельных машин с общей памятью, выступают в качестве рабочих процессоров и выполняют решение подзадач ДО для блоков, а CPU используется в качестве мастера.

Литвин О. М. Наближене обчислення подвійних інтегралів з використанням лагранжевої поліноміальної інтерлінації / О. М. Литвин, О. П. Нечуйвітер // Таврический вестник информатики и математики. — 2012. — № 1 (20). — С. 66-72.

УДК 621.391:517.518:510.52

Побудована кубатурна формула наближеного обчислення інтегралу від функцій двох змінних з використанням лагранжевої поліноміальної інтерлінації функцій на класі дійсних функцій, визначених на $G = [-1, 1]^2$ і таких, що $\left| f^{(p_1, p_2)}(x, y) \right| \leqslant M$. Отримана оцінка похибки кубатурної формули.

Построена кубатурная формула приближенного вычисления интеграла функции двух переменных с использованием лагранжевой полиномиальной интерлинации функций на классе действительных функций, определенных на $G = [-1,1]^2$ и таких, что $\left|f^{(p_1,p_2)}(x,y)\right| \leqslant M$. Получена оценка погрешности кубатурной формулы.

Мельник Т. П. Моделювання стоку р. Тиса із урахуванням максимальних витрат приток / Т. П. Мельник // Таврический вестник информатики и математики. — 2012. — № 1 (20). — С. 73-76.

УДК 519.711+519.6+681.3.01+556:06

Розглянуто ймовірне математичне моделювання стоку р.Тиси із урахуванням максимальних витрат приток. Здійснено визначення величини максимальних потоків методом аналізу пропускної здатності. Надано рекомендації щодо подальшого використання методики.

Предложена математическая модель стока р. Тисы с учетом максимальных расходов притоков. Определена величина максимальных потоков методом анализа пропускной способности. Даны рекомендации по дальнейшему использованию предложенной методики.

Пічкур В. В. Достатні умови оптимальності в задачі структурно-параметричної оптимізації / В. В. Пічкур, Є. М. Страхов // Таврический вестник информатики и математики. — 2012. — № 1 (20). — С. 77-87.

УДК 517.977.58; 517.977.54

У роботі доведені умови існування розв'язку задачі структурно-параметричної оптимізації з фіксованими точками перемикання у класі структурних керувань та достатні умови оптимальності керування. Запропонований алгоритм оптимізації структури лінійної системи з квадратичним критерієм якості.

В работе доказаны условия существования решения задачи структурно-параметрической оптимизации с фиксированными точками переключения в классе структурных управлений и достаточные условия оптимальности управления. Предложен численный метод оптимизации структуры линейной системы с квадратичным критерием качества.

Стёпкин А. В. Возможность и сложность распознавания графов тремя агентами / А. В. Стёпкин // Таврический вестник информатики и математики. — 2012. — № 1 (20). — С. 88-98.

УДК 519.1

Розглядається проблема розпізнавання скінченних графів трьома агентами. Побудовано алгоритм розпізнавання неорієнтованих графів, часова та ємнісна складності якого дорівнюють $O(n^2)$. При розпізнаванні два агенти, що рухаються графом, використовують по дві різні фарби (усього три фарби). Алгоритм базується на методі обходу графа в глибину.

Рассматривается проблема распознавания конечных графов тремя агентами. Построен алгоритм распознавания неориентированных графов, временная и емкостная сложности которого равны $O(n^2)$. При распознавании два агента, передвигающиеся по графу, используют по две различные краски (всего три краски). Алгоритм основан на методе обхода графа в глубину.

СПИСОК АВТОРОВ НОМЕРА

Анафиев Айдер Сератович — к. ф.-м. н., доцент кафедры информатики факультета математики и информатики Таврического национального университета им. В. И. Вернадского, ученый секретарь редакции журнала ТВИМ
e-mail: anafiyev@gmail.com

Блыщик Владимир Федорович — к. ф.-м. н., доцент кафедры информатики факультета математики и информатики Таврического национального университета им. В. И. Вернадского
e-mail: veb@land.ru

Донской Владимир Иосифович — д. ф.-м. н., профессор, заведующий кафедры информатики факультета математики и информатики Таврического национального университета им. В. И. Вернадского, главный редактор журнала ТВИМ
e-mail: donskoy@tnu.crimea.ua

Емец Олег Алексеевич — д. ф.-м. н., профессор, зав. кафедри мат. моделирования и социальной информатики Полтавского университета экономики и торговли
e-mail: yemetsli@mail.ru

Ильченко Анатолий Васильевич — старший преподаватель кафедры информатики факультета математики и информатики Таврического национального университета им. В. И. Вернадского
e-mail: ilch@crimea.edu

Лемтюжникова Дарья Владимировна — аспирант кафедры информатики факультета математики и информатики Таврического национального университета им. В. И. Вернадского
e-mail: darabbt@gmail.com

Литвин Олег Николаевич — д. ф.-м. н., профессор, заведующий кафедры высшей и прикладной математики Украинской инженерно-педагогической академии
e-mail: academ@kharkov.ua

Мельник Татьяна Павловна	преподаватель Львовского национального университета им. Ивана Франко *e-mail: Tatiana_Krizhovec@mail.ru*
Нечуйвитер Олеся Петровна	к. ф.-м. н., докторант кафедры высшей и прикладной математики Украинской инженерно-педагогической академии *e-mail: olesya@email.com*
Пичкур Владимир Владимирович	д. ф.-м. н., доцент кафедры моделиирования сложных систем факультета кибернетики Киевского национального университета им. Тараса Шевченко *e-mail: vpichkur@gmail.com*
Свириденко Александр Васильевич	соискатель кафедры информатики факультета математики и информатики Таврического национального университета им. В. И. Вернадского *e-mail: oleks.sviridenko@gmail.com*
Стёпкин Андрей Викторович	Соискатель ученой степени кандидата наук Института прикладной математики и механики Национальной академии наук Украины *e-mail: stepkin.andrey@rambler.ru*
Страхов Евгений Михайлович	ассистент кафедры оптимального управления и экономической кибернетики Института математики, экономики и механики Одесского национального университета им. И. И. Мечникова *e-mail: swebus86@gmail.com*
Черненко Оксана Алексеевна	к. ф.-м. н., доцент кафедры мат. моделирования и социальной информатики Полтавского университета экономики и торговли *e-mail: oksanachernenko7@gmail.com*
Щербина Олег Александрович	д. ф.-м. н., профессор кафедры информатики факультета математики и информатики Таврического национального университета им. В. И. Вернадского *e-mail: oshcherbina@gmail.com*

ДО ВІДОМА АВТОРІВ

Загальні положення

Для опублікування в журналі «**Таврійський вісник інформатики і математики**» приймаються раніше не опубліковані наукові праці в галузі математики та теоретичної інформатики, згідно зі списком провідних тематичних розділів.
Автору(-ам) потрібно надавати такі документи:

1. Відомості про автора(-ів) (прізвище, ім'я, по батькові, учені ступені та звання, місце роботи та посада, адреси проживання та організації, телефон, факс, адреса електронної пошти тощо).
2. Рецензію сторонньої організації (бажано).
3. Статтю, надруковану на принтері.
4. Падати заявку на сайті журналу **www.tvim.info**.

Вимоги до рукописів

1. Основні елементи статті розміщуються у такій послідовності: індекс УДК, ініціали та прізвище автора, назва статті, анотація (до 10 рядків) українською, російською та англійською мовами (анотація повинна містити конкретну інформацію про отримані результати), текст, список літератури.

2. Стаття може бути написана українською, російською або англійською мовою. Обсяг статті повинен не перевищувати 10 сторінок разом з малюнками, таблицями, графіками (не більше трьох) та бібліографією. Стаття повинна бути структурована (поділена на розділи із заголовками).

3. ***Відповідно до постанови Президії ВАК України від 15 січня 2003 року №7-05/1*** текст статті повинен бути викладений лаконічно, зрозуміло і відповідати такій структурній схемі.

   У *вступі* необхідно чітко виділити (курсивом) такі пункти:
   – *Постановка проблеми* у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями
   – *Аналіз останніх досліджень і публікацій,* в яких започатковано розв'язання даної проблеми і на які спирається автор
   – *Невирішені* раніше частини загальної проблеми, котрим присвячується зазначена стаття
   – *Формулювання цілей статті (постановка задачі)*

   У *висновку* з данного дослідження необхідно чітко виділити (курсивом) *результати* дослідження *та перспективи подальших розвідок у цьому напрямку.*

4. У статті необхідно дотримуватись термінології, прийнятої державним стандартом; використовуючи новий термін або абревіатуру, автор повинен розшифрувати та пояснити їх.
5. Використана література наводиться загальним списком наприкінці статті за порядком посилання на неї в тексті (в квадратних дужках) мовою орігіналу, відповідно до форми Ф23 бюлетеню ВАК України, 2008, № 3.
6. Стаття має бути підготовлена за допомогою видавничої системи LATEX з використанням стильового пакету twim.sty, який можно отримати за адресою **www.tvim.info**. Файли статті у форматі TeX і PDF (плюс графічні файли, якщо потрібні) необхідно прикріпити до заявки на публікацію статті на сайті журналу.

Робота редакції з авторами

1. Матеріали необхідно надіслати за допомогою сайта **www.tvim.info**, а також у вигляді "твердої" копії за адресою редакції: **Таврійський національний університет ім. В. І. Вернадського, пр-т Вернадського, 4, м. Симферопіль, Крим, Україна, 95007**.
2. Редакція залишає за собою право внесення змін редакційного характеру без згоди з автором (-ами).
3. За необхідності автору (-ам) надсилається коректура статті.
4. Остаточне рішення про публікацію приймає редакційна колегія.
5. Рукопис, який надійшов до редакції з порушенням зазначених правил оформлення, не реєструється і не розглядається, а повертається автору (-ам) для доопрацювання.

ДО УВАГИ АВТОРІВ!

Про підвищення вимог до фахових видань, внесених до переліків ВАК України

ПОСТАНОВА
ПРЕЗИДІЇ ВИЩОЇ АТЕСТАЦІЙНОЇ КОМІСІЇ УКРАЇНИ
від 15.01.2003 р. №7-05/1

Необхідною передумовою для внесення видань до переліку наукових фахових видань України є їх відповідність вимогам пункту 7 постанови Президії ВАК України від 10.02.1999 р. №1-02/3 ”Про публікації результатів дисертацій на здобуття наукових ступенів доктора і кандидата наук та їх апробацію“.

... Редакційним колегіям організувати належне рецензування та ретельний відбір статей до друку. Зобов’язати їх приймати до друку у видання 2003 року та й у подальші роки лише наукові статті, які мають такі необхідні елементи: постановку проблеми у загальному вигляді та її зв’язок із важливими науковими чи практичними завданнями; аналіз останніх досліджень і публікацій, в яких започатковано розв’язання данної проблеми і на які спирається автор; виділення невирішених раніше частин загальної проблеми, котрим присвячується зазначена стаття; формулювання цілей статті (постановка задачі); виклад основного матеріалу дослідження с повним обгрунтуванням наукових результатів; висновки з даного дослідження і перспективи подальших розвідок у цьому напрямку.

Голова ВАК України В.В.Скопенко

Вчений секретар Л.М.Артюшин

Подписано к печати 06.06.2012. Формат 38х30/2. Бумага тип ОП. Объем 11,62 п.л. Тираж 500 экз. Заказ 335.
Издано в редакционном отделе КНЦ НАНУ
просп. Вернадского, 2, г. Симферополь, АРК, 95007, Украина