Metaheuristics for a data compression problem *

Kochetov Y. A.
Sobolev Institute of Mathematics, Siberian Branch of the Russian Academy of Sciences, Novosibirsk State University, Novosibirsk, Russia

Iterative local search methods are developed and studied for a data compression problem. Computational experiments for the random generated benchmarks confirm the hight efficiency of the approach.

Метаэвристики для одной задачи сжатия информации*

Кочетов Ю. А.
jkochet@math.nsc.ru
Новосибирск, Институт математики им. С. Л. Соболева СО РАН, Новосибирский госуниверситет

Разработаны и исследованы итерационные методы локального поиска для одной задачи сжатия информации. Проведённые численные исследования подтверждают высокую эффективность методов на тестовых примерах, порождаемых с помощью датчика псевдослучайных чисел.

Введение

При сжатии и хранении информации возникает следующая комбинаторная задача [1]. Задана матрица из нулей и единиц. Требуется так переставить ее столбцы, чтобы число нулей, "зажатых"между единицами, было минимальным. Если существует перестановка столбцов, при которой все единицы в каждой строке идут подряд, то задача решается с полиномиальной трудоемкостью [2]. Если же такой перестановки нет, то задача минимизации числа зажатых нулей является NP-трудной [3]. В настоящей работе для решения этой задачи разработаны три итерационных алгоритма локального поиска: алгоритм имитации отжига, вероятностный алгоритм поиска с запретами и генетический алгоритм, позволяющие в асимптотике находить точное решение задачи. Приводятся результаты численных экспериментов, свидетельствующие о высокой частоте получения оптимума за несколько минут расчетов на персональном компьютере при размерности матриц 100×100.

Постановка задачи

Введем следующие обозначения:
$I = \{1 \dots n\}$ — множество строк матрицы,
$J = \{1 \dots m\}$ — множество столбцов матрицы,
$T = \{1 \dots m\}$ — множество позиций для столбцов матрицы,
(a_{ij}) — 0–1 матрица размерности $n \times m$,
$w_i > 0$ — вес i-й строки.

Переменные задачи:
$s_i \geqslant 1$ — номер первой единицы в i-строке,
$f_i \geqslant 1$ — номер последней единицы в i-строке,
$x_{jt} = 1$, если столбец j стоит на позиции t и $x_{jt} = 0$ в противном случае.

Математическая постановка задачи в терминах целочисленного линейного программирования может быть представлена следующим образом [4]:

$$\min \sum_{i \in I} w_i (f_i - s_i + 1) \tag{1}$$

$$\sum_{t \in T} x_{jt} = 1, \quad j \in J, \tag{2}$$

$$\sum_{j \in J} x_{jt} = 1, \quad t \in T, \tag{3}$$

$$a_{ij} s_i \leqslant \sum_{t \in T} t a_{ij} x_{jt} \leqslant f_i, \quad i \in I, j \in J, \tag{4}$$

$$f_i \geqslant 1,\ s_i \geqslant 1,\ \text{целые}, \quad i \in I, \tag{5}$$

$$x_{jt} \in \{0, 1\}, \quad j \in J, t \in T. \tag{6}$$

Целевая функция (1) с точностью до константы задает взвешенное число зажатых между единицами нулей. Равенство (2) требует одной позиции для каждого столбца, а равенство (3) выделяет один столбец для каждой позиции. Ограничение (4) определяет первую и последнюю единицу в каждой строке. Заметим, что переменные f_i, s_i однозначно определяются матрицей (x_{jt}). Поэтому далее решением будем называть только матрицу (x_{jt}).

Конструктивная эвристика

Идея алгоритма заключается в получении перестановки столбцов матрицы (a_{ij}), в которой столбцы с наименьшим расстоянием по Хеммингу стояли бы как можно ближе друг к другу. Приведём общую схему алгоритма.

1. Выбрать один из столбцов матрицы и поставить его первым.

2. Цикл по $i = 2 \dots m$

Выбрать из неиспользованных столбцов наиболее близкий к $(i-1)$-му столбцу. Поставить его на i-е место.

*Работа выполнена при поддержке грантов РФФИ (проекты № 09-01-00059, № 08-07-00037), целевой программы АВЦП Рособразования (проект № 2.1.1/3235)

Для уменьшения погрешности алгоритма будем использовать все столбцы в качестве первого и выбирать из полученных решений наилучшее.

Результаты тестовых расчётов показывают, что данный алгоритм имеет наибольшую погрешность на сильно разреженных матрицах. Поэтому в дальнейшем в качестве тестовых примеров будем брать разреженные матрицы.

Окрестности

Решение (x_{jt}) определяет перестановку столбцов матрицы (a_{ij}). Рассмотрим следующие окрестности для такой перестановки:

Окрестностью *Вставка* назовем множество решений, получаемых из исходного перемещением одного столбца в новую позицию.

Окрестностью *Отражение* назовем множество решений, получаемых из исходного путем обращения порядка столбцов между двумя произвольно выбранными позициями.

Окрестностью *k-Опт* назовем множество решений, получаемых из исходного путём перехода к перестановке, отличающейся от исходной ровно в k позициях.

Окрестности *Вставка*, *Отражение*, *2-Опт* являются квадратичными по мощности. Окрестности *3-Опт* и *4-Опт* имеют мощность большего порядка, что затрудняет их использование для локального поиска.

В таблице 1 приведены результаты численных экспериментов для пяти примеров. Для каждого из них 100 раз применялась процедура локального спуска со случайно выбранных решений. Среднее значение целевой функции (F), среднее число итераций ($Iter$) и средняя относительная погрешность (ε) показывают превосходство окрестности *Отражение*. Она даёт в 6 раз меньшую погрешность, хотя и тратит в 1,5 раза больше итераций. В дальнейшем переходы по этой окрестности будут преобладать в процедурах локального поиска.

Таблица 1. Сравнение окрестностей

№	*Вставка + 2-Опт*			*Отражение*		
	F	Iter	ε	F	Iter	ε
1	53444	98	100,32	30180	145	13,12
2	52015	95	104,11	28283	138	10,98
3	55116	99	128,11	28769	151	19,07
4	57230	93	108,51	33604	140	22,43
5	53163	99	102,62	32778	146	24,93

Вероятностный поиск с запретами

Основоположником метода поиска с запретами (Tabu search) является Ф. Гловер, который предложил оригинальную схему локального поиска [5]. Она позволяет алгоритму не останавливаться в точке локального оптимума, как это предписано в стандартном алгоритме локального спуска, а путешествовать от одного оптимума к другому в надежде найти среди них глобальный оптимум. Основным механизмом, позволяющим алгоритму выбираться из локального оптимума, является список запретов. Он строится по предыстории поиска, то есть по нескольким последним пройденным решениям, и запрещает часть окрестности текущего решения. Список запретов учитывает специфику задачи и, как правило, запрещает использование тех "фрагментов" решения (ребер графа, координат вектора, цвета вершин), которые менялись на последних шагах алгоритма. Общая схема вероятностного поиска с запретами может быть представлена следующим образом.

Алгоритм поиска с запретами

1. Построить стартовое решение (x_{jt}). Установить текущий рекорд $(x^*_{jt}) := (x_{jt})$. Положить список запретов пустым.
2. Пока не выполнен критерий остановки делать следующее.
   - Сформировать рандомизированную окрестность решения (x_{jt}), исключая решения, попадающие под запреты.
   - Выбрать в ней наилучшее решение (x'_{jt}) и положить $(x_{jt}) := (x'_{jt})$.
   - Если $F(x^*_{jt}) > F(x'_{jt})$, то сменить рекорд $(x^*_{jt}) := (x_{jt})$.
   - Обновить список запретов.

Если длительное время рекорд не меняется, то используется процедура интенсификации: алгоритм возвращается в наилучшее найденное решение. Далее поиск продолжается по объединению окрестностей *2-Опт* и *Вставка* некоторое число итераций, а затем снова по окрестности *О*тражение.

Если интенсификация не приводит к получению нового рекорда, используется процедура диверсификации. Доля просматриваемой окрестности уменьшается, и алгоритм в течении некоторого числа итераций работает с окрестностью меньшей мощности. Диверсификация позволяет сменить район поиска с целью найти новые перспективные области.

Метод имитации отжига

Экзотическое название данного подхода связано с методами имитационного моделирования в статистической физике, основанными на технике Монте–Карло. Исследование кристаллической решетки и поведения атомов при медленном остывании тела привело к появлению на свет вероятностных алгоритмов, которые оказались чрезвычайно эффективными в комбинаторной оптимизации. Впервые это было замечено в 1983 году [6, 7].

Сегодня этот подход является популярным как среди практиков благодаря своей простоте, гибкости и эффективности, так и среди теоретиков, поскольку для него удается аналитически исследовать свойства и доказать асимптотическую сходимость [8].

Алгоритм имитации отжига относится к классу пороговых алгоритмов локального поиска. На каждом шаге в окрестности текущего решения выбирается случайным образом соседнее решение. Если разность по целевой функции между новым и текущим решением неположительна (для задач на максимум — неотрицательна), то новое решение заменяет текущее. В противном случае переход к новому решению происходит с некоторой вероятностью, зависящей от этой разности и управляющего параметра, называемого температурой. Общая схема может быть представлена следующим образом.

Алгоритм имитации отжига

1. Построить стартовое решение (x_{jt}). Установить текущий рекорд $(x^*_{jt}) := (x_{jt})$. Определить стартовую температуру c и коэффициент охлаждения $\gamma \in (0, 1)$.
2. Пока не выполнен критерий остановки делать следующее.
   (а) Выполнить заданное количество итераций ν:
   — Выбрать произвольного соседа (x'_{jt}) для текущего решения и вычислить величину $\Delta = F(x'_{jt}) - F(x_{jt})$.
   — Если $\Delta \leqslant 0$, то положить $(x_{jt}) := (x'_{jt})$ иначе перейти к решению (x'_{jt}) с вероятностью $e^{-\Delta/c}$.
   — Если $F(x^*_{jt}) > F(x_{jt})$, то сменить рекорд $(x^*_{jt}) := (x_{jt})$.
   (б) Понизить температуру: $c := \gamma c$.

В качестве стартового решения берётся решение, полученное с помощью конструктивной эвристики. Это позволяет начинать поиск с низкой стартовой температуры. Величина ν во внутреннем цикле 2 берётся пропорциональной мощности окрестности *Отражение*.

В алгоритме используется идея чередующихся окрестностей. Она заключается в том, что поочерёдно используются все окрестности, причём окрестность *Отражение* используется чаще других, точнее в 35% случаев.

Если алгоритм долгое время не меняет рекорд, то в качестве диверсификации осуществляется принудительный переход по окрестности *3-Opt* в случайно выбранном направлении. Каждая смена рекорда и смена температуры сопровождается локальным улучшением по объединению окрестностей *2-Opt* и *Вставка*.

Генетический алгоритм

Идея генетических алгоритмов заимствована у живой природы. Она состоит в организации эволюционного процесса на множестве допустимых решений задачи, конечной целью которого является получение оптимального решения. Впервые эти нестандартные идеи были применены к решению оптимизационных задач в середине 70-х годов прошлого столетия [9, 10]. Примерно через десять лет появились первые теоретические обоснования этого подхода [11]. На сегодняшний день генетические алгоритмы доказали свою конкурентоспособность при решении многих NP-трудных задач [12]. Приведем общую схему генетического алгоритма.

Генетический алгоритм

1. Построить начальную популяцию из K решений. Определить рекорд (x^*_{jt}) как лучшее решение в популяции.
2. Пока не выполнен критерий остановки делать следующее:
   (а) Получить L потомков:
   — Выбрать два родительских решения.
   — Применить алгоритм скрещивания.
   — К полученному решению с вероятностью p_m применить алгоритм мутации.
   — Добавить новое решение к популяции.
   (б) Убрать из популяции L наихудших решений, пересчитать рекорд (x^*_{jt}).

В пункте 2 используется следующий алгоритм скрещивания: По двум родительским перестановкам столбцов получим перестановку столбцов потомка. Сначала случайным образом фиксируем $[m/2]$ столбцов первого из родителей, добавляем их на соответствующие места в перестановку потомка, затем на остальные места ставим отсутствующие в текущей перестановке элементы в порядке, в котором они идут в перестановке второго родителя. Мутации потомков происходят путем случайного перехода в соседнее решение по одной из окрестностей. Стартовая популяция генерируется на основе конструктивной эвристики.

Если заданное число итераций не происходит смена рекорда, то применяется процедура интенсификации. Ко всем решениям в популяции применяется алгоритм локального улучшения по рандомизированным окрестностям *Вставка*, *2-Opt*. Каждый элемент этих окрестностей включался в рандомизированную окрестность независимо от других элементов с вероятностью p_o.

Численные эксперименты

Разработанные алгоритмы тестировались на случайно порождаемых примерах размерности $m = n = 100$ с известными оптимальными решениями. Генерировались матрицы, в которых меж-

ду любыми двумя единицами в строке нет нулей. Затем столбцы матриц переставлялись случайным образом. Плотность матриц составляла 5%. Величины w_i выбирались из интервала от 0 до 100 случайно с равномерным распределением.

Критерием остановки метаэвристик было число обращений к процедуре вычисления целевой функции, равное 1 500 000.

Для имитации отжига стартовая температура бралась равной 18, $\gamma = 0.99$, $\nu = 20000$. Для генетического алгоритма мутации проводились по окрестности *Вставка*. Параметры алгоритма выбирались следующим образом: $K = 10$, $L = 10$, $p_m = 0.2$, $p_o = 0.05$. Для поиска с запретами длина списка запретов выбиралась равной 30, а мощность рандомизированной окрестности равной 10% от полной окрестности.

В таблице 2 представлены средние значения относительных погрешностей (в процентах), полученные за 10 запусков каждого из алгоритмов по каждому примеру. В скобках указано сколько раз было найдено оптимальное решение.

Все метаэвристики существенно улучшают решение конструктивной эвристики, часто находят оптимальное решение и в среднем имеют малую погрешность.

Выбор наилучшего элемента в каждой квадратичной по мощности окрестности требует $O(nm^3)$ операций. Применение специальных структур данных позволяет понизить эту трудоёмкость до $O(nm^2)$, что существенно сказывается на времени работы метаэвристик.

Дальнейшее повышение эффективности возможно за счёт внедрения элементов искусственного интеллекта. Таким способом можно автоматически настраивать управляющие параметры.

Автор выражает искреннюю благодарность студентам ФИТ НГУ Хмелеву А.В., Сивых М.Г. и Яковлеву А.В. за помощь в проведении численных экспериментов.

Литература

[1] *Veldhorst M.* Approximation of the consecutive ones matrix augmentation problem. — SIAM Journal on Computing, 1985. — v. 14, p. 709–729.

[2] *Booth K.S., Lueker G. S.* Testing for the consecutive ones property, intervals graphs, and graph planarity using PQ-tree algorithm. — J.Comput.System.Sci., 1976. — v. 13, p. 335–379.

[3] *Cheng T.C.E., Diamond J.E., Lin B.M.T.* Optimal scheduling in film production to minimize talent hold cost. — Journal of Optimization Theory and Applications, 1993. — v. 79 N. 3, p. 479–492.

[4] *Кононова П. А., Кочетов Ю. А.* Об одной задаче выбора последовательности столбцов 0-1 матрицы. — Труды 14-й Байкальской международной школы-семинара "Методы оптимизации и их приложения". Иркутск, 2008. — Том 1. с. 444–451.

[5] *Glover F., Laguna M.* Tabu Search. — Boston: Kluwer Acad. Publ. 1997.

[6] *Černý V.* Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. — J. Opt. Theory Appl. 1985. — V. 45. p. 41–51.

[7] *Kirkpatrick S., Gelatt C. D., Vecchi M. P.* Optimization by simulated annealing. — Science. 1983. — V. 220, p. 671–680.

[8] *Aarts E. H. L., Korst J. H. M., Laarhoven van P. J. M.* Simulated annealing. — Local Search in Combinatorial Optimization. Chichester: Wiley, 1997. — p. 91–120.

[9] *Растригин Л. А.* Случайный поиск — специфика, этапы истории и предрассудки. — Вопросы кибернетики. 1978. — Вып. 33, с. 3–16.

[10] *Bremermann H. J., Roghson J., Salaff S.* Global properties of evolution processes. — Natural automata and useful simulations. London: Macmillan. 1966. — p. 3–42.

[11] *Holland J. H.* Adaptation in Natural and Artificial Systems. — Ann Arbor: University of Michigan Press. 1975.

[12] *Goldberg D. E.* Genetic Algorithms in Search, Optimization, and Machine Learning. — Reading, MA: Addison-Wesley. 1989.

Таблица 2. Результаты численных экспериментов

№	конструктивная эвристика	имитация отжига	генетический алгоритм	поиск с запретами
1	1	0 (10)	0.001 (9)	0 (10)
2	22.1	0 (10)	0.006 (7)	1.82 (6)
3	54.35	0 (10)	0 (10)	4.73 (6)
4	24.73	0 (10)	0 (10)	1.10 (4)
5	53.8	0 (10)	0 (10)	2.92 (7)
6	63.46	0.23 (8)	0.006 (9)	3.87 (4)
7	4.81	0 (10)	0 (10)	0.01 (9)
8	16.2	0.15 (9)	0.006 (8)	1.20 (5)
9	5.74	0 (10)	0 (10)	0 (10)
10	88	0 (10)	0 (10)	0 (10)