УДК 519.21

Ю. А. Кочетов, М. Г. Сивых, А. В. Хмелёв, А. В. Яковлев

МЕТОДЫ ЛОКАЛЬНОГО ПОИСКА ДЛЯ ОДНОЙ ЗАДАЧИ О ПЕРЕСТАНОВКЕ СТОЛБЦОВ БИНАРНОЙ МАТРИЦЫ*

Разработаны и исследованы итерационные методы локального поиска для решения известной задачи о перестановке столбцов бинарной матрицы. Эта задача является NP-трудной в сильном смысле, и для широкого класса полиномиальных алгоритмов относительная погрешность в худшем случае может оказаться сколь угодно большой величиной. В данной работе показано, что разработанные итерационные методы локального поиска быстро находят оптимальное решение или решение с малой относительной погрешностью, если число строк и столбцов матрицы не превосходит 100.

Ключевые слова: локальный поиск, метаэвристики, сжатие информации.

Введение

При сжатии и хранении информации возникает следующая комбинаторная задача [1]. Задана матрица, состоящая 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 указанной перестановки не существует, то задача минимизации числа таких нулей является NP-трудной в сильном смысле [3]. Аналогичная задача возникает и при составлении расписаний в кинопроизводстве [4]. В этом случае бинарная матрица задает участие актеров в снимаемых сценах. Строки матрицы соответствуют актерам, а перестановкой столбцов можно влиять на *окна* в расписании актеров. Каждая сцена снимается ровно один день. Каждый актер имеет определенный гонорар за каждый рабочий день. Окна в расписании считаются рабочими днями. Минимизация нулей, зажатых между единицами, соответствует стремлению сократить число окон в расписании или снизить суммарную зарплату актеров.

Для решения этой задачи в [4] разработан метод ветвей и границ. Он позволяет решать задачи малой размерности — до 15 строк и столбцов. Эвристические алгоритмы исследовались в [1]. Установлено, что для широкого класса полиномиальных алгоритмов их относительная погрешность в худшем случае может оказаться сколь угодно большой величиной. В работе [5] исследовались три формулировки задачи в терминах целочисленного линейного программирования. Они позволяют использовать классические методы целочисленного линейного программирования и, в частности, метод ветвей и границ для поиска оптимального решения. Все формулировки приводят к большому разрыву целочисленности. Как следствие, с их помощью удается решать задачи только

* Работа выполнена при поддержке целевой программы АВЦП Рособразования (проект № 2.1.1/3235).

ISSN 1818-7897. Вестник НГУ. Серия: Математика, механика, информатика. 2012. Т. 12, вып. 1. С. 91–101

малой размерности. Для матриц размерности 22×22 даже за 24 часа машинного времени не удается найти точное решение пакетом CPLEX 11,0 на PC с процессором 2,8 GHz. В работе [6] разработан генетический алгоритм, однако его эффективность оказалась невысокой. В связи с этим нами разработаны итерационные алгоритмы локального поиска: гибридный алгоритм имитации отжига, вероятностный алгоритм поиска с запретами и генетический алгоритм локального поиска, позволяющие в асимптотике находить точное решение задачи. Эти алгоритмы используют новую конструктивную эвристику для получения хорошего начального приближения и вероятностный локальный поиск с чередующимися окрестностями, включающий новую окрестность, получившую название «окрестность *отражения*». Новая окрестность удачно дополняет хорошо известные окрестности вставки столбцов и парных замен и существенно повышает эффективность поиска. Приводятся результаты численных экспериментов, свидетельствующие о получении оптимального решения или решения с малой относительной погрешностью за несколько минут расчетов на персональном компьютере, если число строк и столбцов матрицы не превосходит ста.

1. Постановка задачи

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

Переменные задачи:
$s_i \geq 1$ — номер первой единицы в i-й строке;
$f_i \geq 1$ — номер последней единицы в i-й 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:

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

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

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

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

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

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

Целевая функция (1) с точностью до константы задает взвешенное число зажатых между единицами нулей. Равенство (2) определяет одну позицию для каждого столбца, а

равенство (3) выделяет один столбец для каждой позиции. Ограничение (4) определяет первую и последнюю единицы в каждой строке. Переменные f_i, s_i однозначно определяются матрицей (x_{jt}). Поэтому далее решением будем называть только матрицу (x_{jt}).

Заметим, что требование целочисленности переменных f_i, s_i в ограничении (5) можно отбросить. Оптимальное значение целевой функции (1) от этого не изменится, а задача упростится для решения методом ветвей и границ. В этом смысле приведенная формулировка отличается в лучшую сторону от аналогичных формулировок в [5]. Она содержит только m^2 целочисленных переменных. Тем не менее применение метода ветвей и границ снова наталкивается на проблему большого разрыва целочисленности. В связи с этим возникает потребность в разработке специализированных численных методов и оценке качества их работы. Вопрос о формулировке задачи с малым разрывом целочисленности пока остается открытым.

2. Конструктивная эвристика

Рассмотрим следующий полиномиальный алгоритм, который будет использоваться в дальнейшем для построения начального решения в методах локального поиска. Идея алгоритма заключается в получении перестановки столбцов матрицы (a_{ij}), в которой столбцы с наименьшим расстоянием Хэмминга стояли бы как можно ближе друг к другу. Приведем общую схему такого алгоритма.

Эвристический алгоритм

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

2. Цикл по $i = 2, \ldots, m$

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

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

На рис. 1 показаны результаты численных экспериментов для данной эвристики. Ось абсцисс соответствует процентному содержанию единиц в случайно порождаемых матрицах. Ось ординат — средней относительной погрешности в процентах по 30 тестовым примерам.

Матрицы порождались следующим образом. В каждой строке выбирались две позиции, между которыми выставлялись единицы. Остальные элементы полагались равными нулю. Очевидно, что это оптимальное решение. Затем столбцы матрицы случайным образом перемешивались.

Результаты тестовых расчетов показывают, что данный алгоритм получает решения с небольшой погрешностью на матрицах с высокой плотностью. На таких примерах часто находится оптимальное решение. На примерах с небольшим содержанием единиц алгоритм сильно ошибается, иногда почти в два раза. Поэтому в дальнейшем в качестве тестовых примеров будут использоваться матрицы с небольшим процентным содержанием единиц.

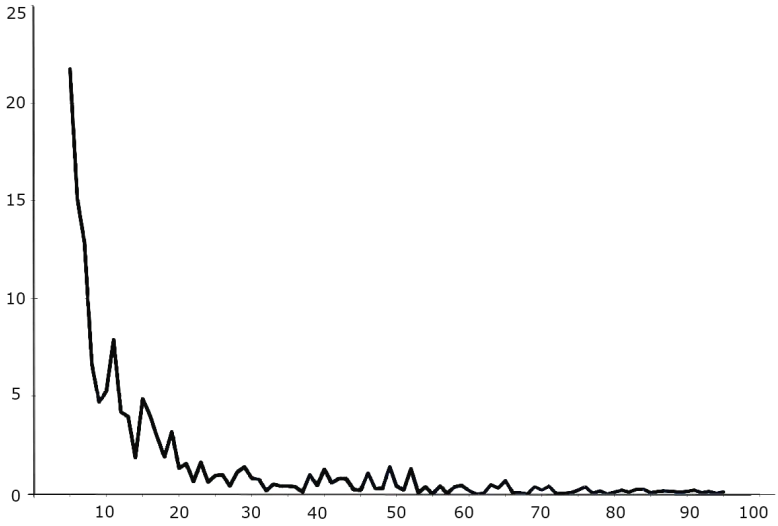

Рис. 1. Средняя погрешность (в процентах)

3. Окрестности

Пусть (x_{jt}) некоторое допустимое решение задачи (1)–(6). Определим по этому решению перестановку столбцов матрицы (a_{ij}). Рассмотрим следующие окрестности для решения (x_{jt}).

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 путем перехода к перестановке, отличающейся от исходной ровно в k позициях.

Окрестности *Вставка*, *Отражение*, 2-*Опт* являются квадратичными по мощности. Окрестности k-*Опт*, $k > 2$, имеют мощность бóльшего порядка, что затрудняет их использование для локального поиска.

Заметим, что каждая из окрестностей обладает следующим важным свойством. Для любых двух решений (x'_{jt}) и (x''_{jt}) существует такое число q и решения (x^l_{jt}), $l = 0, \ldots, q$, что $(x^0_{jt}) = (x'_{jt})$, $(x^q_{jt}) = (x''_{jt})$ и каждое решение (x^l_{jt}) является соседним для решения (x^{l-1}_{jt}). Другими словами, используя каждую из этих окрестностей, можно из любого решения перейти в любое другое, включая и оптимальное, за конечное число шагов. Это свойство является необходимым для асимптотической сходимости вероятностных методов локального поиска (см., например, [7]).

Несмотря на указанное свойство достижимости, приведенные окрестности сильно отличаются по погрешности получаемых локальных минимумов. В табл. 1 приведены результаты численных экспериментов для пяти примеров размерности 100×100. Для каждого из них 100 раз применялась процедура локального спуска со случайно выбранных стартовых решений. Средние значения целевой функции (F), числа итераций ($Iter$) и относительной погрешности (ε) показывают превосходство окрестности *Отражение*. Она дает в 6 раз меньшую погрешность, хотя и требует в 1,5 раза больше итераций.

Таблица 1. Сравнение окрестностей

	Вставка ∪ *2-Opt*			*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

В дальнейшем переходы по этой окрестности будут преобладать в процедурах локального поиска. Применение именно этой окрестности и позволило получить высокоэффективные методы локального поиска и превзойти ранее предложенные генетические алгоритмы из [6].

4. Поиск с запретами

Основоположником метода поиска с запретами (Tabu search) является Ф. Гловер, который предложил оригинальную схему локального поиска [8]. 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, т. е. 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, пары столбцов матрицы), которые менялись на последних шагах алгоритма. Необходимые асимптотические свойства достигаются за счет рандомизации окрестности [9]. Каждый элемент исходной окрестности включается в рандомизированную окрестность независимо от других элементов с заданной вероятностью p_o. Общая схема такого стохастического локального поиска с запретами может быть представлена следующим образом.

Вероятностный поиск с запретами

1. Построить стартовое решение (x_{jt}). Положить $(x^*_{jt}) := (x_{jt})$.
   Определить 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.
3. Предъявить лучшее найденное решение (x^*_{jt}).

Стартовое решение на шаге 1 строится конструктивной эвристикой, представленной в разд. 3. В качестве списка запретов используются номера передвигаемых столбцов для окрестностей *Вставка* и 2-*Опт* и номера столбцов, между которыми меняется порядок на противоположный, для окрестности *Отражение*. Длина списка запретов остается постоянной, и добавление новой записи происходит на место наиболее старой записи (последний пункт шага 2).

Для повышения эффективности поиска приведенная базовая схема дополняется двумя процедурами: интенсификации и диверсификации. Если определенное число итераций 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 по окрестности *Отражение*.

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

В качестве критерия остановки используется либо общее число итераций, либо число обращений к процедуре вычисления целевой функции. Последний критерий наиболее удобен для сравнения различных методов локального поиска, так как итерации различных методов могут иметь различную трудоемкость.

5. 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 вероятностных алгоритмов, которые оказались чрезвычайно эффективными в комбинаторной оптимизации. Впервые это было замечено в 1983 г. [10]. Сегодня этот подход является популярным как среди практиков, благодаря своей простоте, гибкости и эффективности, так и среди теоретиков, поскольку для него удается аналитически доказать асимптотическую сходимость [7].

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. При понижении температуры вероятность перехода к худшему соседнему решению падает. При низкой температуре алгоритм вырождается в вероятностный локальный спуск. Ниже представлена общая схема гибридного алгоритма, в котором наряду с данной идеей применяется идея чередующихся окрестностей [11]. Она

состоит в систематической смене окрестности для локального поиска. Такой подход приводит к большему разнообразию получаемых локальных оптимумов и в конечном счете к лучшему финальному решению.

Гибридный алгоритм имитации отжига

1. Построить стартовое решение (x_{jt}). Положить $(x^*_{jt}) := (x_{jt})$. Определить стартовую температуру c и коэффициент охлаждения $\gamma \in (0, 1)$.
2. Пока не выполнен критерий остановки делать следующее:
   а) выполнить заданное число итераций ν:
      - выбрать случайно соседнее решение (x'_{jt}) 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 $(x^*_{jt}) := (x_{jt})$;
      - сменить окрестность;

   б) понизить температуру: $c := \gamma c$.;
3. Предъявить лучшее найденное решение (x^*_{jt}).

Стартовое решение строится конструктивной эвристикой. Это позволяет начинать поиск с низкой стартовой температуры. Величина ν берется порядка m^2. Если рекордное решение (x^*_{jt}) не меняется в течение заданного числа итераций, то для диверсификации поиска осуществляется принудительный переход по окрестности 3-*Onm* в случайно выбранном направлении. При каждой смене рекорда и смене температуры с вероятностью 0,02 применяется процедура локального улучшения по объединению окрестностей *Вставка* и 2-*Onm*.

В качестве критерия остановки используется либо общее число обращений к процедуре подсчета целевой функции, либо получение решения, из которого алгоритму не удается сдвинуться в течение заданного числа итераций. При низкой стартовой температуре второй критерий срабатывает достаточно быстро, и лучшее найденное решение, как правило, оказывается локальным оптимумом по окрестностям 2-*Onm*, *Вставка* и *Отражение*.

6. Генетический локальный поиск

Идея генетических алгоритмов заимствована у живой природы. Она состоит в организации эволюционного процесса на множестве допустимых решений задачи. На каждой итерации такого процесса рассматривается некоторый набор допустимых решений задачи, который принято называть *популяцией*. Этот набор используется для целенаправленного поиска новых решений с меньшей погрешностью. Конечной целью такого итерационного процесса является получение оптимального решения. Впервые эти нестандартные идеи были применены к решению оптимизационных задач в середине 70-х гг. прошлого столетия [12; 13]. Теоретическое обоснование этого подхода можно

найти в [14]. На сегодняшний день генетические алгоритмы доказали свою конкурентоспособность при решении многих NP-трудных задач [15], особенно в приложениях, где математические модели имеют сложную структуру и применение классических методов затруднено или связано с неоправданно большими затратами вычислительных ресурсов.

Ниже приводится общая схема генетического алгоритма, в которой начиная с некоторой итерации элементами популяции являются локальные оптимумы задачи. Начальная популяция строится конструктивной эвристикой. На каждой итерации выбираются два *родительских* решения для построения нового допустимого решения задачи. Новое решение должно сохранить структурные особенности родительских решений. Для этого разрабатываются специализированные алгоритмы скрещивания, или так называемые *кроссоверы*. Успех того или иного генетического алгоритма во многом зависит от того, насколько удачно такой кроссовер учитывает специфику решаемой задачи. К полученному с его помощью новому решению применяется вероятностная процедура *мутации*, состоящая в малой перестройке решения, например переходе к соседнему решению по некоторой окрестности. Полученное решение добавляется в популяцию, а наихудшее решение удаляется из популяции.

Генетический локальный поиск

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;
      - с вероятностью p_m применить алгоритм мутации;
      - добавить новое решение к популяции.
      - обновить рекорд (x^*_{jt}).

   б) убрать из популяции L наихудших решений.
3. Предъявить лучшее найденное решение (x^*_{jt}).

В алгоритме скрещивания по двум родительским перестановкам столбцов строится новая перестановка. Сначала случайным образом фиксируется $[m/2]$ столбцов первого родителя. Они добавляются на соответствующие места в перестановку нового решения. На оставшиеся места ставятся остальные столбцы в том порядке, в котором они идут в перестановке второго родителя.

При выборе родительской пары одно решение выбирается случайным образом из популяции. В качестве второго решения всегда берется наилучшее найденное решение (x^*_{jt}). Такой способ выбора родительской пары взят из метаэвристик пчелиного роя [16]. В этих методах новые решения всегда порождаются *маткой* — наилучшим найденным решением. Кроме того, такой способ идейно близок к методу локального поиска с чередующимися окрестностями [11], в котором в качестве текущего решения всегда берется наилучшее найденное решение. Мутации происходят путем случайного перехода

в соседнее решение по одной из рассматриваемых окрестностей. Стартовая популяция генерируется на основе конструктивной эвристики. В ходе ее работы порождаются m решений, из которых выбираются K наилучших.

Если заданное число итераций не приводит к смене рекорда, то применяется процедура интенсификации. Ко всем решениям в популяции применяется алгоритм локального улучшения по рандомизированным окрестностям *Вставка* и 2-*Опт*. В качестве критерия остановки используется общее число обращений к процедуре вычисления целевой функции.

7. Численные эксперименты

Разработанные алгоритмы запрограммированы на языке C++ и тестировались на случайно порождаемых примерах размерности $m = n = 100$ с известными оптимальными решениями. Генерировались матрицы, в которых между любыми двумя единицами в строке нет нулей. Затем столбцы матриц переставлялись случайным образом. Плотность матриц составляла 5 %. Величины w_i выбирались из интервала от 1 до 100 случайно с равномерным распределением.

Критерием остановки алгоритмов является общее число обращений к процедуре вычисления целевой функции, равное 1 500 000. Для имитации отжига стартовая температура выбиралась равной 18, $\gamma = 0{,}99$, $\nu = 20000$. Для генетического алгоритма мутации проводились по окрестности *Вставка*, $K = L = 10$, $p_m = 0{,}2$, $p_o = 0{,}05$. Для поиска с запретами длина списка запретов полагалась равной 30, $p_o = 0{,}1$. Настройка параметров алгоритмов проводилась экспериментально в ходе предварительных испытаний. При увеличении плотности матриц и росте размерности эти значения параметров, возможно, следует скорректировать для повышения эффективности методов, но и представленные значения, по нашему опыту, должны приводить к хорошим результатам.

В табл. 2 представлены средние значения относительных погрешностей в процентах, полученные для 10 испытаний каждого алгоритма по каждому примеру. В скобках показано число испытаний, в ходе которых найдено оптимальное решение. Все алгоритмы существенно улучшают решение конструктивной эвристики, часто находят оптимальное решение и в среднем имеют малую погрешность.

Выбор наилучшего элемента в каждой квадратичной окрестности требует $O(nm^3)$ операций. Применение специальных структур данных позволяет понизить эту трудоемкость до $O(nm^2)$. Для этого достаточно хранить величины $f_i, s_i, i \in I$, и пересчитывать их на каждой итерации. При таком подходе время решения одной задачи с указанным критерием остановки не превышает трех минут для каждого метода на персональном компьютере Pentium 2,8 GHz, RAM 1 Gb, операционная система Windows XP professional. Дальнейшее повышение эффективности возможно за счет внедрения элементов искусственного интеллекта. Таким способом можно автоматически настраивать управляющие параметры алгоритмов.

Как уже упоминалось, приведенный класс тестовых примеров является полиномиально разрешимым. Поэтому интересно проверить эффективность разработанных методов и на других классах тестовых примеров. Для этого были рассмотрены как примеры

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)

из Интернета[1] и предшествующих работ [6; 17], так и новые примеры, получающиеся из уже описанных включением малого числа нулей (один или два в строке) между подряд идущими единицами до случайного изменения порядка столбцов матрицы. Отметим, что для всех уже опубликованных работ размерность рассматриваемых матриц не превышает 70 строк и 100 столбцов. Для таких примеров новые методы локального поиска легко находят известные оптимальные решения. Для последнего случая, когда оптимальные решения неизвестны, сам метод генерации дает хорошую верхнюю оценку оптимума. Во всех случаях новые методы быстро находят приближенное решение с лучшим значением целевой функции.

Список литературы

1. *Veldhorst M.* Approximation of the Consecutive Ones Matrix Augmentation Problem // SIAM J. Comp. 1985. Vol. 14. P. 709–729.

2. *Booth K. S., Lueker G. S.* Testing for the Consecutive Ones Property, Interval Graphs and Graph Planarity Using PQ-Tree Algorithm // J. Comput. System. Sci. 1976. Vol. 13. P. 335–379.

3. *Booth K. S.* PQ-Tree Algorithms: Ph.D. Thesis. Univ. of California, Berkeley: 1975.

4. *Cheng T. C. E., Diamond J. E., Lin B. M. T.* Optimal Scheduling in Film Production to Minimize Talent Hold Cost // J. of Optim. Theory and Appl. 1993. Vol. 79. No. 3. P. 479–492.

5. *Кононова П. А., Кочетов Ю. А.* Об одной задаче выбора последовательности столбцов 0–1 матрицы // Тр. 14-й Байкальской международной школы-семинара «Методы оптимизации и их приложения». Иркутск, 2008. Т. 1. С. 444–451.

6. *Nordström A. L., Tufekci S.* A Genetic Algorithm for the Talent Scheduling Problem // Computers and Oper. Res. 1994. Vol. 21. P. 927–940.

[1] `http://ww2.cs.mu.oz.au/pjs/talent/`

7. *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.

8. *Glover F., Laguna M.* Tabu Search. Boston: Kluwer Acad. Publ., 1997.

9. *Гончаров Е. Н., Кочетов Ю. А.* Вероятностный поиск с запретами для дискретных задач безусловной оптимизации // Дискрет. анализ и исслед. операций. Сер. 2. 2002. Т. 9, № 2. С. 13–30.

10. *Kirkpatrick S., Gelatt C. D., Vecchi M. P.* Optimization by Simulated Annealing // Science. 1983. Vol. 220. P. 671–680.

11. *Hansen P., Mladenović N.* Developments of Variable Neighborhood Search // Essays and Surveys of Metaheuristics. Boston: Kluwer Acad. Publ., 2002. P. 415–440.

12. *Растригин Л. А.* Случайный поиск — специфика, этапы истории и предрассудки // Вопросы кибернетики. 1978. Вып. 33. С. 3–16.

13. *Bremermann H. J., Roghson J., Salaff S.* Global Properties of Evolution Processes // Natural Automata and Useful Simulations. L.: Macmillan, 1966. P. 3–42.

14. *Holland J. H.* Adaptation in Natural and Artificial Systems. Ann Arbor: Univ. of Michigan Press, 1975.

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

16. *Lučić P., Teodorović D.* Computing with Bees: Attacking Complex Transportation Engineering Problems // Intern. J. Artif. Intel. Tools 2003. Vol. 12. P. 375–394.

17. *De la Banda M. G., Stuckey P. J., Chu G.* Solving Talent Scheduling with Dynamic Programming // INFORMS J. Computing. 2011. Vol. 23. No. 1. P. 120–137.

Материал поступил в редколлегию 18.05.2010

Адреса авторов

КОЧЕТОВ Юрий Андреевич
Институт математики им. С. Л. Соболева СО РАН
пр. Акад. Коптюга, 4, Новосибирск, 630090, Россия
e-mail: jkochet@math.nsc.ru

СИВЫХ Михаил Геннадьевич
Новосибирский государственный университет
ул. Пирогова, 2, Новосибирск, 630090, Россия
e-mail: sivykh-mikl@yandex.ru

ХМЕЛЁВ Алексей Владимирович
Новосибирский государственный университет
ул. Пирогова, 2, Новосибирск, 630090, Россия
e-mail: avhmel@gmail.com

ЯКОВЛЕВ Андрей Вадимович
Новосибирский государственный университет
ул. Пирогова, 2, Новосибирск, 630090, Россия
e-mail: ya-a-v@ya.ru