ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Январь—июнь 2007. Серия 2. Том 14, № 1. 3–31

УДК 519.85

ГЕНЕТИЧЕСКИЙ ЛОКАЛЬНЫЙ ПОИСК ДЛЯ ЗАДАЧИ О p–МЕДИАНЕ С ПРЕДПОЧТЕНИЯМИ КЛИЕНТОВ*)

Е. В. Алексеева, Ю. А. Кочетов

Рассматривается обобщение задачи о p–медиане, когда клиенты выбирают поставщиков, исходя из собственных предпочтений. Для решения этой задачи разработан генетический алгоритм, использующий в качестве популяции локальные оптимумы по окрестности Лина–Кернигана. Для оценки качества получаемых решений используются сведéния исходной задачи к задачам целочисленного линейного программирования. Предложено новое сведéние, доминирующее уже известные по значению целевой функции линейной релаксации. Приведены численные эксперименты на примерах с большим разрывом двойственности.

Введение

В классической задаче о p-медиане [24] заданы множество клиентов, множество предприятий и матрица производственно-транспортных затрат на обслуживание клиентов этих предприятий. Требуется выбрать ровно p предприятий так, чтобы суммарные затраты на обслуживание клиентов были бы минимальными. Неявно предполагается, что клиенты согласны с любым выбором поставщиков и их мнением можно пренебречь. Содержательно более интересная ситуация возникает в случае, когда это не так, и мнения клиентов следует явным образом учитывать в математической модели.

В задаче о p-медиане с предпочтениями клиентов (МПК) имеются два уровня принятия решений. Они неравноправны. Сначала на верхнем уровне принимается решение об открытии p предприятий. Затем на нижнем уровне, зная места расположения этих предприятий, клиенты самостоятельно выбирают поставщиков, руководствуясь собственными предпочтениями. Задача состоит в том, чтобы на верхнем уровне выбрать открываемые предприятия и обслужить всех клиентов с минимальными суммарными затратами, принимая во внимание тот факт, что

*)Исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований (проект 06–01–00075).

поставщиков выбирают не на верхнем, а на нижнем уровне. Если предпочтения на нижнем уровне совпадают с предпочтениями на верхнем уровне, то получаем классическую задачу о p-медиане. Следовательно, задача МПК является NP-трудной в сильном смысле и не принадлежит классу APX [6].

Впервые задачи с предпочтениями клиентов рассматривались в [17]. В [2, 3] установлена тесная связь задачи с псевдобулевыми функциями, найдены полиномиально разрешимые случаи задачи и показано сведéние МПК к задачам целочисленного линейного программирования (ЦЛП). В данной статье также исследуются сведéния МПК к задачам ЦЛП и предложено новое сведéние, которое доминирует предшествующие по значению целевой функции линейной релаксации. Такие релаксации позволяют получать нижнюю оценку оптимума. Для получения верхних оценок разработан генетический алгоритм, который концентрирует поиск только на локальных оптимумах относительно заданных окрестностей.

Статья построена следующим образом. В § 1 вводятся основные обозначения и приводится математическая постановка задачи. В § 2 исследуется вопрос о вычислении нижних оценок оптимума. В § 3 излагается общая схема генетического локального поиска и варианты её адаптации к задаче МПК. В § 4 приводятся результаты численных экспериментов. В последнем параграфе обсуждаются направления дальнейших исследований.

1. Постановка задачи

Введём следующие обозначения:

$I = \{1, \ldots, m\}$ — множество предприятий;

$J = \{1, \ldots, n\}$ — множество клиентов;

$c_{ij} \geqslant 0,\ i \in I, j \in J$, — матрица производственно-транспортных затрат на обслуживание клиентов;

$g_{ij} \geqslant 0,\ i \in I, j \in J$, — матрица предпочтений клиентов: если $g_{i_1 j} < g_{i_2 j}$, то j-й клиент из предприятий i_1 или i_2 выберет предприятие i_1;

p, $0 < p < m$, — число открываемых предприятий.

Переменные задачи:

$$y_i = \begin{cases} 1, & \text{если открывается } i\text{-е предприятие;} \\ 0 & \text{в противном случае;} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{если } j\text{-й клиент обслуживается } i\text{-м предприятием;} \\ 0 & \text{в противном случае.} \end{cases}$$

С использованием введённых обозначений задача МПК может быть записана в виде следующей задачи двухуровневого программирования [3]:

найти

$$\min_{y} \sum_{i \in I} \sum_{j \in J} c_{ij} x^*_{ij}(y) \tag{1}$$

при ограничениях:

$$\sum_{i \in I} y_i = p, \tag{2}$$

$$y_i \in \{0, 1\}, i \in I, \tag{3}$$

где $x^*_{ij}(y)$ – оптимальное решение задачи клиентов: найти

$$\min_{x} \sum_{i \in I} \sum_{j \in J} g_{ij} x_{ij} \tag{4}$$

при ограничениях:

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J, \tag{5}$$

$$x_{ij} \leqslant y_i, \tag{6}$$

$$x_{ij} \in \{0, 1\}, \tag{7}$$

где $i \in I,\ j \in J$. Целевая функция $F(y, x^*(y)) = \sum\limits_{i \in I} \sum\limits_{j \in J} c_{ij} x^*_{ij}(y)$ задаёт суммарные затраты ЛПР_1 на обслуживание всех клиентов. При заданном векторе y матрица $x^*(y)$ указывает оптимальный выбор поставщиков согласно предпочтениям клиентов. В общем случае оптимальный выбор $x^*(y)$ может быть не единственным. Тогда задача (1)–(7) требует дополнительных уточнений, что именно является её оптимальным решением. В частности, можно рассматривать кооперативные и некооперативные стратегии для ЛПР_1 и ЛПР_2, когда ЛПР_2 на множестве своих оптимальных решений минимизирует или максимизирует суммарные затраты ЛПР_1. Ниже будет рассматриваться более простой случай, когда для любого решения ЛПР_1 существует единственное оптимальное решение ЛПР_2. Гарантировать это свойство задачи можно, например, в случае, когда при каждом j все элементы матрицы (g_{ij}) различны. Другими словами, каждый клиент для любой пары предприятий может сказать, какое из них для него предпочтительнее. В этом случае целевая функция $F(y, x^*(y))$ однозначно определяется вектором y и вместо $F(y, x^*(y))$ можно пользоваться обозначением $F(y)$. Под оптимальным решением задачи понимается вектор y^*, удовлетворяющий условиям (2), (3) и доставляющий минимум функции $F(y)$.

2. Нижние оценки оптимума

Рассмотрим различные способы сведéния задачи (1)–(7) к задачам ЦЛП и покажем их различия при вычислении нижних оценок методами линейного программирования.

2.1. Сведéния к задачам ЦЛП

Для каждого столбца матрицы (g_{ij}) определим перестановку $(i_1^j, ..., i_m^j)$ такую, что

$$g_{i_1 j} < g_{i_2 j} < \cdots < g_{i_m j}, \tag{8}$$

и множества $S_{ij} = \{k \in I \mid g_{kj} < g_{ij}\}$, $T_{ij} = \{k \in I \mid g_{kj} > g_{ij}\}$, $i \in I$. Заметим, что для оптимального решения задачи клиентов имеет место следующая импликация:

$$(x_{ij} = 1) \Longrightarrow (y_k = 0),\ k \in S_{ij}. \tag{9}$$

Тогда задача (1)–(7) может быть представлена в виде: найти

$$\min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{10}$$

при ограничениях:

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J, \tag{11}$$

$$\sum_{i \in I} y_i = p, \tag{12}$$

$$y_k \leqslant 1 - x_{ij}, \quad k \in S_{ij}, \tag{13}$$

$$x_{ij} \leqslant y_i, \tag{14}$$

$$x_{ij}, y_i \in \{0, 1\}, \tag{15}$$

где $i \in I, j \in J$.

Действительно, для оптимального решения задачи (10)–(15) все ограничения исходной задачи будут выполнены, а группа ограничений (13) гарантирует, что x_{ij} будет оптимальным решением задачи клиентов. Отбрасывая условия булевости переменных, получаем линейную релаксацию задачи. Обозначим через LB_1 её оптимальное значение. Эта величина, очевидно, даёт нижнюю оценку оптимума исходной задачи. При таком сведéнии число переменных осталось прежним и равным $m + mn$,

а число ограничений стало на $O(m^2 n)$ больше. Чтобы избежать большого числа дополнительных ограничений, условие (9) можно переписать в виде:

$$\sum_{k \in S_{ij}} y_k \leqslant |S_{ij}|(1 - x_{ij}), \qquad i \in I; j \in J. \tag{16}$$

Эти ограничения получаются суммированием ограничений (13). Новое сведéние приводит аналогичным образом к новой нижней оценке. Обозначим её через LB_2. Число дополнительных ограничений сократилось до mn, оптимальное решение задачи от этого не изменилось, но линейная релаксация стала более слабой, т. е. $LB_1 \geqslant LB_2$. Оптимальное решение не изменится, если условие (9) заменить на следующее условие:

$$y_i \leqslant x_{ij} + \sum_{k \in S_{ij}} y_k, \qquad i \in I; j \in J, \tag{17}$$

что даёт ещё одно сведéние и нижнюю оценку LB_3. Приведённые нижние оценки были предложены и обоснованы в [2]. Рассмотрим новое сведéние к задаче ЦЛП. Учитывая ограничения (11), (14), легко показать, что условие (13) можно заменить на условие:

$$y_i \leqslant x_{ij} + \sum_{k \in S_{ij}} x_{kj}, \qquad i \in I; j \in J. \tag{18}$$

Покажем, что получаемая с помощью (18) новая нижняя оценка, обозначим её через LB_4, доминирует три предшествующие.

Теорема 1. $LB_4 \geqslant \max(LB_1, LB_2, LB_3)$.

Доказательство. Ограничение (11) перепишем в виде

$$1 - \sum_{k \in T_{ij}} x_{kj} = x_{ij} + \sum_{k \in S_{ij}} x_{kj}.$$

Тогда ограничение (18) примет вид $y_i \leqslant 1 - \sum\limits_{k \in T_{ij}} x_{kj} \leqslant 1 - x_{kj}$, $k \in T_{ij}$, откуда следует (13). Так как $x_{ij} \leqslant y_i$, $(i \in I, j \in J)$, то

$$x_{ij} + \sum_{k \in S_{ij}} x_{kj} \leqslant x_{ij} + \sum_{k \in S_{ij}} y_k.$$

Следовательно, (17) выполняется. Таким образом, значение LB_4 не может быть меньше LB_1, LB_2, LB_3. Теорема 1 доказана.

2.2. Сведéние к задаче о паре матриц

Рассмотрим матрицы $A = (a_{ij}), i \in I, j \in J_1$, и $B = (b_{ij}), i \in I, j \in J_2$ с одинаковым числом строк, а число столбцов может быть различным. Задача о паре матриц заключается в том, чтобы найти непустое подмножество строк $S \subseteq I$, на котором достигается минимум следующей целевой функции [1]:

$$R(S) = \sum_{j \in J_1} \max_{i \in S} a_{ij} + \sum_{j \in J_2} \min_{i \in S} b_{ij}.$$

Заметим, что если $J_1 = I$, $a_{ij} = a_i$ при $i = j$ и $a_{ij} = 0$ при $i \neq j$, то получаем простейшую задачу размещения [24]. Таким образом, задача о паре матриц является NP-трудной в сильном смысле.

Известно [3], что для задачи (1)–(7) можно построить сведéние к задаче о паре матриц с дополнительным ограничением $|S| = p$. Представим матрицу (c_{ij}) в виде суммы двух матриц $c_{ij} = a_{ij} + b_{ij}, \quad i \in I, j \in J$, где

$$a_{i_k j} = \sum_{l=1}^{k-1} \min\{0; c_{i_{l+1} j} - c_{i_l j}\}, k = 1, \ldots, m, j \in J, \tag{19}$$

$$b_{i_k j} = c_{i_1 j} + \sum_{l=1}^{k-1} \max\{0; c_{i_{l+1} j} - c_{i_l j}\}, k = 1, \ldots, m, j \in J, \tag{20}$$

и перестановка $(i_1^j, \ldots, i_m^j), j \in J$, соответствует (8). Заметим, что

$$a_{i_1 j} \geqslant a_{i_2 j} \geqslant \cdots \geqslant a_{i_m j}, \quad b_{i_1 j} \leqslant b_{i_2 j} \leqslant \cdots \leqslant b_{i_m j}$$

при всех $j \in J$. Теперь задача (1)–(7) может быть записана в следующем виде: найти

$$\min_{S \subset I, |S| = p} \sum_{j \in J} \max_{i \in S} a_{ij} + \sum_{j \in J} \min_{i \in S} b_{ij}.$$

Для получения нижней оценки эту задачу можно переписать в терминах ЦЛП и вычислить оптимум в линейной релаксации. Однако представление в терминах ЦЛП может оказаться неединственным. Как мы уже видели, разные представления могут приводить к разным нижним оценкам. Ниже будут рассматриваться два представления для задачи о паре матриц, первое из которых приводит к слабой нижней оценке, а второе — к нижней оценке, которая не хуже, чем L_4. Введём вспомогательные переменные $z_{ij} \in \{0, 1\}, i \in I, j \in J$, и представим задачу о паре матриц

в следующем виде: найти

$$\min\left\{\sum_{i\in I}\sum_{j\in J}a_{ij}z_{ij}+\sum_{i\in I}\sum_{j\in J}b_{ij}x_{ij}\right\} \tag{21}$$

при ограничениях:

$$y_i \leqslant \sum_{k\in S_{ij}} z_{kj} + z_{ij}, \tag{22}$$

$$\sum_{i\in I} x_{ij} = 1, \quad j \in J, \tag{23}$$

$$\sum_{i\in I} z_{ij} = 1, \quad j \in J, \tag{24}$$

$$\sum_{i\in I} y_i = p, \tag{25}$$

$$0 \leqslant x_{ij} \leqslant y_i, \tag{26}$$

$$0 \leqslant z_{ij} \leqslant y_i, \tag{27}$$

$$z_{ij}, x_{ij}, y_i \in \{0,1\}, \tag{28}$$

где $i \in I, j \in J$. Так как $b_{i_1 j} \leqslant b_{i_2 j} \leqslant \cdots \leqslant b_{i_m j}, j \in J$, то j-й столбец матрицы (b_{ij}) задаёт тот же порядок, что и j-й столбец матрицы (g_{ij}). Ограничение (22) гарантирует, что j-й клиент делает выбор согласно собственным предпочтениям. Обозначим через LB_5 оптимальное значение линейной релаксации этой задачи, а через LB_5' оптимальное значение линейной релаксации этой же задачи с дополнительными ограничениями $x_{ij} = z_{ij}, i \in I$, $j \in J$. Заметим, что $LB_5' \geqslant LB_5$ и $LB_5' = LB_4$.

Теорема 2. *Для любого $N > 0$ существуют исходные данные задачи* (1)–(7) *такие, что $LB_5 \leqslant -N$.*

Доказательство. При $k \geqslant N + 2$ полагаем $I = \{1, \ldots, k + p\}$, $J = \{1, 2\}, p \in Z^+$, $d_{ij} = k + p - i$ и

$$c_{ij} = \begin{cases} 0, & \text{если} \quad 1 \leqslant i < p, & j = 1; \\ k, & \text{если} \quad i = p, & j = 1; \\ 1, & \text{если} \quad p+1 \leqslant i \leqslant k+p, & j = 1; \\ 1, & \text{если} \quad 1 \leqslant i < p, & j = 2; \\ 0, & \text{если} \quad p \leqslant i \leqslant k+p, & j = 2. \end{cases}$$

Согласно (19) и (20) получаем

$$a_{ij} = \begin{cases} -k, & \text{если} \quad i < p, \quad j = 1; \\ 0, & \text{если} \quad i \geqslant p, \quad j = 1; \\ 0, & \text{если} \quad j = 2; \end{cases} \quad b_{ij} = \begin{cases} k, & \text{если} \quad i \leqslant p, \quad j = 1; \\ 1, & \text{если} \quad i > p, \quad j = 1; \\ c_{ij}, & \text{если} \quad j = 2; \end{cases}$$

Решение

$$x_{ij} = \begin{cases} 0, & \text{если} \quad i \leqslant p, \quad j \in J; \\ 1/k, & \text{если} \quad i > p, \quad j \in J; \end{cases} \quad y_i = \begin{cases} 1, & \text{если} \quad i < p; \\ 0, & \text{если} \quad i = p; \\ 1/k, & \text{если} \quad i > p; \end{cases}$$

$$z_{ij} = \begin{cases} 1 - 1/k, & \text{если} \quad i = p - 1, \quad j = 1; \\ 1/k, & \text{если} \quad i = p + k, \quad j = 1; \\ 0, & \text{если} \quad i \neq p - 1, \quad i \neq p + k, \quad j = 1; \\ 0, & \text{если} \quad i \leqslant p, \quad j = 2; \\ 1/k, & \text{если} \quad i > p, \quad j = 2; \end{cases}$$

является допустимым решением релаксированной задачи и

$$LB_5 \leqslant (-k)(1 - 1/k) + \sum_{i=1}^{k} 1/k = 2 - k.$$

Теорема 2 доказана.

2.3. Новое сведéние к задаче о паре матриц

Представим новую формулировку задачи о паре матриц в виде задачи ЦЛП и покажем, что релаксация полученной задачи даёт нижнюю оценку не хуже, чем LB_4. Для j-го столбца матрицы (c_{ij}) и соответствующей (8) перестановки $(i_1^j, ..., i_m^j)$ вычислим величины

$$\Delta_l^j = \min\{0; c_{i_{l+1}j} - c_{i_l j}\}, l = 1, \ldots, m - 1.$$

Пусть $L_j = \{l \in \{1, ..., m-1\} \mid \Delta_l^j < 0\}$. Заметим, что при заданном $j \in J$ по номеру $l \in L_j$ однозначно восстанавливается номер $i_l \in I$. Заменим j-й столбец матрицы (a_{ij}) на $|L_j|$ столбцов вида

$$\overline{a}_{il} = \begin{cases} 0, & \text{если} \quad i \in T_{i_l j} \\ -\Delta_l^j, & \text{если} \quad i \notin T_{i_l j}. \end{cases} \quad i \in I, \ l \in L_j.$$

Тогда

$$a_{ij} = \sum_{l \in L_j} (\overline{a}_{il} + \Delta_l^j), i \in I, j \in J$$

и задача о паре матриц может быть переписана в виде: найти

$$\min_{y_i\in\{0,1\},\sum\limits_{i\in I} y_i=p}\left\{\sum_{j\in J}\sum_{l\in L_j}\max_{i|y_i=1}\overline{a}_{il}+\sum_{j\in J}\min_{i|y_i=1}b_{ij}\right\}+\sum_{j\in J}\sum_{l\in L_j}\Delta_l^j. \tag{29}$$

Введём дополнительные переменные $v_l^j\in\{0,1\},\ l\in L_j$ и $j\in J$, и представим эту задачу следующим образом: найти

$$\min\left\{\sum_{j\in J}\sum_{l\in L_j}-\Delta_l^j v_l^j+\sum_{i\in I}\sum_{j\in J}b_{ij}x_{ij}\right\}+\sum_{j\in J}\sum_{l\in L_j}\Delta_l^j \tag{30}$$

при ограничениях:

$$y_i\leqslant x_{ij}+\sum_{k\in S_{ij}}x_{kj}, \tag{31}$$

$$\sum_{i\in I}x_{ij}=1,\ j\in J, \tag{32}$$

$$\sum_{i\in I}y_i=p, \tag{33}$$

$$0\leqslant x_{ij}\leqslant y_i, \tag{34}$$

$$v_l^j\geqslant\sum_{k\notin T_{i_lj}}x_{kj'},\ l\in L_j,\ j'\in J, \tag{35}$$

$$v_l^j,y_i,x_{ij}\in\{0,1\},\ l\in L_j, \tag{36}$$

где $j\in J$ и $i\in I$.

Обозначим через LB_6 оптимальное значение линейной релаксации этой задачи.

Теорема 3. $LB_6\geqslant LB_4$.

Доказательство. Покажем, что задача (10), (18), (11)–(15) эквивалентна задаче (30)–(34), (36) с дополнительным ограничением

$$v_l^j=\sum_{k\notin T_{i_lj}}x_{kj},\ l\in L_j,\ j\in J. \tag{37}$$

Так как $c_{ij}=a_{ij}+b_{ij}$, где $i\in I$ и $j\in J$, то

$$\sum_{i\in I}c_{ij}x_{ij}=\sum_{l\in L_j}-\Delta_l^j v_l^j+\sum_{l\in L_j}\Delta_l^j+\sum_{i\in I}b_{ij}x_{ij},$$

для всех $j \in J$ и допустимого решения v_l^j, x_{ij} задачи (30)–(34), (36), (37). Действительно,

$$\sum_{l\in L_j} -\Delta_l^j v_l^j + \sum_{l\in L_j} \Delta_l^j = \sum_{l\in L_j} \Delta_l^j (1 - v_l^j) = \sum_{l\in L_j} \Delta_l^j \sum_{k\in T_{i_l j}} x_{kj}$$
$$= \sum_{k=2}^{m} x_{i_k j} \sum_{l=1}^{k-1} \Delta_l^j = \sum_{k=2}^{m} a_{i_k j} x_{i_k j} = \sum_{i\in I} a_{ij} x_{ij}.$$

Заметим, что это равенство верно как для целочисленных значений переменных, так и для дробных. Следовательно, задача (10), (18), (11)–(15) эквивалентна задаче (30)–(34), (36), (37). Тогда значение LB_4 равно оптимальному значению линейной релаксации задачи (30)–(34), (37), причём если ограничения (37) заменить на ограничения

$$v_l^j \geqslant \sum_{k\notin T_{i_l j}} x_{kj},\ l \in L_j,\ j \in J, \tag{38}$$

то оптимальное значение не изменится, так как рассматриваемая задача на минимум, коэффициенты в целевой функции (30) при v_l^j положительны ($-\Delta_l^j \geqslant 0$). Следовательно, значения v_l^j однозначно определяются по переменным x_{ij}. Но ограничения (38) составляют часть ограничений (35), следовательно $LB_4 \leqslant LB_6$. Теорема 3 доказана.

Теорема 4. *Для любого $N > 0$ существуют исходные данные задачи* (1)–(7) *такие, что $LB_6/LB_4 \geqslant N$.*

Доказательство. Пусть $k = N$ и исходные данные те же, что и в доказательстве теоремы 2. Тогда решение

$$y_i = \begin{cases} 1, & \text{если} \quad i < p; \\ 0, & \text{если} \quad i = p; \\ 1/k, & \text{если} \quad i > p; \end{cases}$$

$$x_{ij} = \begin{cases} 1 - 1/k, & \text{если} \quad i = p-1, & j = 1; \\ 1/k, & \text{если} \quad i = p+k, & j = 1; \\ 0, & \text{если} \quad i \neq p-1; i \neq p+k, & j = 1; \\ 0, & \text{если} \quad i \leqslant p, & j = 2; \\ 1/k, & \text{если} \quad i > p, & j = 2; \end{cases}$$

является допустимым для задачи (10), (11)–(14), (18). Следовательно, $LB_4 \leqslant 1/k$. Двойственная задача к задаче (30)– (31) имеет вид: найти

$$\max\left\{\sum_{j\in J} u_j + pr\right\} + \sum_{j\in J}\sum_{l\in L_j} \Delta_l^j$$

при ограничениях:

$$\sum_{j' \in J} t_{lj'} \leqslant -\Delta_l^j, \quad l \in L_j, j \in J,$$
$$\sum_{j \in J} (w_{ij} - \alpha_{ij}) + r \leqslant 0, \quad i \in I,$$
$$\alpha_{ij} + \sum_{k \in T_{ij}} \alpha_{kj} + u_j - w_{ij} - \sum_{j' \in J} \sum_{l \in L_{j'}} (t_{lj'} | l \notin T_{ij'}) \leqslant b_{ij}, \quad i \in I, j \in J,$$
$$w_{ij} \geqslant 0, \alpha_{ij} \geqslant 0, t_{lj'} \geqslant 0, \quad l \in L_j, j \in J, j' \in J, i \in I.$$

Для приведённых исходных данных имеем: $L_j = \{1\}, \Delta_l^j = -k$ при $j = 1, L_j = \varnothing$ при $j = 2$ и

$$\bar{a}_{ij} = \begin{cases} 0, & \text{если} \quad i < p, \quad j = 1; \\ k, & \text{если} \quad i \geqslant p \quad j = 1; \end{cases}$$

$$b_{ij} = \begin{cases} k, & \text{если} \quad i \leqslant p, \quad j = 1; \\ 1, & \text{если} \quad i > p, \quad j = 1; \\ 1, & \text{если} \quad i < p, \quad j = 2; \\ 0, & \text{если} \quad i \geqslant p, \quad j = 2. \end{cases}$$

Решение

$$w_{ij} = \alpha_{ij} = r = 0, u = (k, 1), t_{lj'} = \begin{cases} k-1, & \text{если} \quad l = 1 \text{ и } j' = 1; \\ 1, & \text{если} \quad l = 1 \text{ и } j' = 2; \end{cases}$$

является допустимым в двойственной задаче. Следовательно, $LB_6 \geqslant k + 1 - k = 1$. Теорема 4 доказана.

4. Генетические алгоритмы

Идеи генетических алгоритмов основаны на аналогиях с эволюцией живых организмов в ходе естественного отбора. Они представляют собой итерационный процесс, конечной целью которого является получение оптимального решения в сложной комбинаторной задаче. Разработчик генетических алгоритмов выступает в данном случае как «создатель», который должен разумно установить законы эволюции, чтобы быстро достичь желаемой цели. Впервые эти нестандартные идеи были применены к решению оптимизационных задач в середине 70-х годов прошлого столетия [8, 14]. Примерно через десять лет появились первые теоретические обоснования этого подхода [19, 26]. На сегодняшний день генетические алгоритмы доказали свою конкурентоспособность при решении многих

NP-трудных задач [4, 16] и особенно в практических приложениях, где математические модели имеют сложную структуру и применение стандартных методов типа ветвей и границ, динамического или линейного программирования крайне затруднено.

Общую схему генетических алгоритмов проще всего понять, рассматривая задачи безусловной оптимизации: найти

$$\min\{F(x) \mid x \in B^n\}, \quad B^n = \{0,1\}^n.$$

Стандартный генетический алгоритм начинает работу с формирования начальной *популяции* $\mathcal{X}_0$ — конечного набора допустимых решений задачи. Эти решения могут быть выбраны случайным образом, получены с помощью вероятностных жадных алгоритмов или другими методами. Как мы увидим ниже, выбор начальной популяции не имеет значения для сходимости процесса в асимптотике. Однако формирование «хорошей» начальной популяции (например, из множества локальных оптимумов) может заметно сократить время достижения глобального оптимума.

Для оценки допустимых решений кроме целевой функции часто используется функция пригодности $\Phi(x)$. Она выбирается с учётом специфики конкретной оптимизационной задачи и должна быть чувствительна к небольшим изменениям аргумента. Целевая функция не всегда удовлетворяет этому свойству, как, например, в задачах упаковки в контейнеры, задачах теории расписаний на параллельных машинах или при календарном планировании в условиях ограниченных ресурсов. Удачный выбор функции $\Phi(x)$ может существенно повысить эффективность генетических алгоритмов, хотя в простейшем случае можно предполагать, что $\Phi(x) = F(x)$.

На каждом шаге генетического алгоритма с помощью вероятностного оператора *селекции* в качестве *родителей* из популяции выбираются два решения x^1, x^2. Оператор *скрещивания* по этим решениям строит новое решение x'. Оно подвергается небольшим случайным модификациям, которые принято называть *мутациями*. Затем решение добавляется в популяцию, а решение с наибольшим значением функции пригодности удаляется из популяции. Общая схема такого алгоритма может быть записана следующим образом.

Генетический алгоритм

1. Выбрать начальную популяцию $\mathcal{X}_0$ и положить
$\Phi^* = \min\{\Phi(x) \mid x \in \mathcal{X}_0\}, k := 0.$
2. Пока не выполнен критерий остановки, делать следующее:

2.1. Выбрать родителей x^1, x^2 из популяции $\mathcal{X}_k$.

2.2. Построить x' по решениям x^1, x^2.

2.3. Модифицировать x'.

2.4. Если $\Phi^* > \Phi(x')$, то положить $\Phi^* := \Phi(x')$.

2.5. Обновить популяцию и положить $k := k + 1$.

Остановимся подробнее на основных операторах этого алгоритма: селекции, скрещивании и мутации. Среди операторов селекции наиболее распространёнными являются два вероятностных оператора *пропорциональной* и *турнирной* селекции. При пропорциональной селекции вероятность выбора на k-м шаге решения x в качестве одного из родителей обратно пропорциональна значению $\Phi(x)$. При турнирной селекции формируется случайное подмножество из элементов популяции и среди них выбирается один элемент с наименьшим значением функции пригодности. Турнирная селекция имеет определённые преимущества перед пропорциональной, так как не теряет своей избирательности, когда в ходе эволюции все элементы популяции становятся примерно одинаковыми по пригодности. Операторы селекции строятся таким образом, чтобы с ненулевой вероятностью любой элемент популяции мог бы быть выбран в качестве одного из родителей. Более того, допускается ситуация, когда оба родителя представлены одним и тем же элементом популяции.

Как только два решения выбраны, к ним применяется вероятностный оператор скрещивания (*crossover*). Существует много различных версий этого оператора, некоторые из них не ограничиваются двумя родителями. Однородный оператор скрещивания является, наверное, одним из наиболее простых и понятных. По решениям x^1, x^2 он строит решение x', присваивая с вероятностью 0,5 каждой координате этого вектора соответствующее значение одного из родителей. Если векторы x^1, x^2 совпадали, скажем, по первой координате, то вектор x' *унаследует* это значение. Геометрически, оператор скрещивания случайным образом выбирает в гиперкубе вершину x', которая принадлежит минимальной грани, содержащей вершины x^1, x^2. Можно сказать, что оператор скрещивания старается выбрать новое решение x' между x^1, x^2, полагаясь на удачу. Более аккуратная процедура могла бы выглядеть таким образом. Новым решением x' является оптимальное решение исходной задачи на соответствующей грани гиперкуба. Конечно, если расстояние Хемминга между x^1, x^2 равно размерности гиперкуба, то задача оптимального скрещивания совпадает с исходной. Тем не менее даже приближённое решение этой задачи вместо случайного выбора заметно улучшает работу генетических алгоритмов [13, 9, 11]. Адаптация этой идеи к двухуровневой

задаче о p-медиане будет рассмотрена ниже.

Оператор мутации, применяемый к решению x' в п. 2.3. генетического алгоритма, с заданной вероятностью $p_m \in (0,1)$ меняет значение каждой координаты на противоположное. Например, вероятность того, что вектор $(0,0,0,0,0)$ в ходе мутации перейдет в вектор $(1,1,1,0,0)$, равна $p_m^3 \times (1-p_m)^2 > 0$. Таким образом, с ненулевой вероятностью решение x' может перейти в любое другое решение, что гарантирует в асимптотике получение точного решения задачи. Отметим, что модификация решения x' может состоять не только в случайной мутации, но и в частичной перестройке решения алгоритмами локального поиска. Применение локального спуска позволяет генетическому алгоритму сосредоточиться только на локальных оптимумах. Множество локальных оптимумов может оказаться экспоненциально большим, и на первый взгляд кажется, что такой вариант алгоритма не будет иметь больших преимуществ. Однако экспериментальные исследования распределения локальных оптимумов свидетельствуют о высокой концентрации их в непосредственной близости от глобального оптимума [12]. Это наблюдение известно как тезис о существовании «большой долины» для задач на минимум и «центрального горного массива» для задач на максимум. Этот тезис отчасти объясняет работоспособность генетических алгоритмов. Если в популяции собираются локальные оптимумы, которые сконцентрированы в одном месте, и очередное решение x' выбирается где-то между двумя произвольными локальными оптимумами, то такой процесс имеет много шансов найти глобальный оптимум. Аналогичные рассуждения объясняют работоспособность и других локальных алгоритмов. Более того, этот тезис подсказывает способ построения сложных тестовых примеров. Если локальные оптимумы удаётся распределить по всей допустимой области, и нет концентрации хороших локальных оптимумов в малой части допустимой области, то методами локального поиска будет трудно найти точное решение задачи. Примеры таких классов исходных данных для задач размещения можно найти, например, в [22, 21].

3.1. Генетический локальный поиск для задачи МПК

Рассмотрим варианты адаптации общей схемы генетических алгоритмов к задаче МПК. В качестве пространства поиска естественно взять p-й слой гиперкуба B^m, т. е. все вектора $y_i \in \{0,1\}, i \in I$, удовлетворяющие условию $\sum\limits_{i \in I} y_i = p$, а в качестве функции пригодности $\Phi(y)$ — целевую функцию $F(y)$. Такой выбор функции пригодности представляется обоснованным, так как она чувствительна к небольшим изменениям аргумента.

Операторы селекции используют функцию $F(y)$ для выбора родительской пары. Наряду с турнирной и пропорциональной селекцией рассмотрим третий оператор, получивший название «лучший и случайный». Согласно этому оператору в качестве одного из родителей всегда берётся лучший элемент популяции. Второй родитель выбирается из оставшихся элементов случайным образом с равномерным распределением. Такой оператор позволяет имитировать поведение алгоритма локального поиска с чередующимися окрестностями [18], когда хранится только наилучший найденный локальный оптимум.

После выбора родительской пары y^1, y^2 выполняется оператор скрещивания. Рассмотрим четыре типа таких операторов: равномерный, жадный, одноточечный и связывающих путей (path relinking).

При равномерном операторе новое допустимое решение y' строится следующим образом. Если $y_i^1 = y_i^2$ при некотором $i \in I$, то y'_i наследует это значение: $y'_i = y_i^1 = y_i^2$. Для остальных $i \in I$ сначала полагается $y'_i = 0$, а затем выполняется следующая итерационная процедура. Выбирается такая координата, что $y'_i = 0$, и полагается $y'_i = 1$. Выбор координаты проводится случайным образом с равномерным распределением. Процедура останавливается, как только будет выполняться равенство $\sum\limits_{i \in I} y'_i = p$.

При жадном операторе итерационная процедура предусматривает более осмысленный выбор координаты, для которой y'_i полагается равной 1. Среди претендентов выбирается та координата, для которой уменьшение целевой функции будет наибольшим. Если уменьшение целевой функции невозможно таким способом, то выбирается координата с наименьшим увеличением целевой функции. В остальном жадный оператор совпадает с равномерным. По сути оператор требует решить исходную задачу с помощью жадной процедуры на множестве координат, на котором родители отличаются друг от друга.

Одноточечный оператор скрещивания хорошо известен и часто применяется в генетических алгоритмах [19]. Согласно правилам этого оператора выбирается одна координата, скажем, i_0, и полагается $y'_i = y_i^1$ при $i \leqslant i_0$ и $y'_i = y_i^2$ при $i > i_0$. Координата i_0 выбирается случайным образом, например, с равномерным распределением. Для задачи МПК такой способ может приводить к недопустимым решениям: $\sum\limits_{i \in I} y'_i \neq p$. Поэтому требуется модификация этого оператора. Пусть как и раньше $y'_i = y_i^1$ при $i \leqslant i_0$. Если $\sum\limits_{i \leqslant i_0} y'_i < p$, то среди единичных компонент векто-

ра $y^2_i, i > i_0$, выберем первые $(p - \sum\limits_{i \leqslant i_0} y'_i)$ компонент и для них положим $y'_i = 1$. Остальные компоненты положим равными 0. Если же вектор $y^2_i, i > i_0$, не содержит достаточное число единиц, то аналогичным образом используются такие компоненты вектора y^2_i, что $y^2_i \neq y^1_i, i \leqslant i_0$. Заметим, что так построенное решение имеет не менее двух недостатков. Единичные компоненты вектора y^2 с большими номерами имеют мало шансов быть унаследованными в векторе y'. Кроме того, вектор y' может отличаться от родителей в тех координатах, в которых они совпадают. Тем не менее, эти недостатки, как мы увидим ниже, сглаживаются при правильном выборе параметров алгоритма и эффективном локальном поиске.

Наконец, оператор связывающих путей строит вектор y' следующим образом. Из двух решений y^1 и y^2 выбирается решение с меньшим значением целевой функции и строится последовательность допустимых решений $y^1 = \overline{y}^1, \overline{y}^2, \ldots, \overline{y}^k = y^2$ таких, что расстояние Хемминга между соседними решениями равно 2. Другими словами, переход к соседнему решению осуществляется за счёт закрытия одного предприятия и открытия другого. Решение y' выбирается среди элементов этой последовательности. Для заданных y^1, y^2 можно построить экспоненциально много таких последовательностей. Чтобы решение y' унаследовало общие компоненты родителей, следует рассматривать только кратчайшие последовательности. Однако таких последовательностей тоже много. Среди кратчайших последовательностей принято выбирать ту, в которой переход к соседнему решению сопровождается наибольшим уменьшением (или наименьшим увеличением) целевой функции [15]. Это аналог жадной процедуры, но основанной на других принципах. Решение y' выбирается в этой последовательности таким образом, чтобы соседние с ним решения не имели меньших значений целевой функции [25]. Если такой выбор невозможен, то в качестве y' берётся решение из середины последовательности.

Оператор мутации случайным образом преобразует решение y' в другое допустимое решение y''. С заданной вероятностью каждая единичная компонента вектора y' становится равной 0, а одна из нулевых компонент, выбранная случайным образом с равномерным распределением, получает значение 1.

К решению y'' применяется процедура локального спуска сначала по окрестности 1-*замена* (Swap), а затем по окрестности Лина–Кернигана [20, 21]. Полученный локальный оптимум добавляется в популяцию, если он в ней ещё не содержится, и наихудшее решение удаляется из по-

пуляции. Критерием остановки генетического алгоритма служит число полученных локальных оптимумов.

3.2. Процедуры локального спуска

В работе [6] исследовался вопрос о вычислительной сложности получения локальных оптимумов для классической задачи о p-медиане. В частности, было установлено, что задача локального поиска для p-медианы с окрестностью 1-*замена* является плотно PLS-полной. Это означает, в частности, что в худшем случае стандартный алгоритм локального спуска с любым правилом замещения требует экспоненциального числа итераций для получения локального оптимума. Так как задача МПК является обобщением классической задачи о p-медиане, то это свойство сохраняется и для неё. Таким образом, одна итерация генетического алгоритма, т. е. построение очередного локального оптимума, в худшем случае не является полиномиальной при использовании стандартного алгоритма локального улучшения. Однако поведение алгоритмов в худшем случае может сильно отличаться от их поведения в среднем [27]. Более того, это поведение может сильно зависеть от правил замещения, т. е. от выбора направления спуска, если таковых несколько. Обозначим через $N(y)$ множество соседних решений для допустимого решения y, а через $N^*(y) = \{y' \in N(y) \mid F(y') < F(y)\}$ подмножество соседних решений с меньшим значением целевой функции. Рассмотрим следующие правила замещения:

1) *Спуск в направлении наилучшего элемента.* На каждом шаге локального спуска в множестве N^* выбирается допустимое решение с наименьшим значением целевой функции. Такой выбор кажется наиболее естественным, но требует просмотра всей окрестности.

2) *Спуск в направлении наихудшего элемента.* В некотором смысле это правило противоположно предыдущему, так как в множестве N^* выбирается элемент с наибольшим значением целевой функции. Такая стратегия даёт пологий спуск к локальному минимуму, но может потребовать значительно больше шагов, чем в предыдущем случае.

3) *Спуск в направлении случайного элемента* предполагает выбор элемента из множества N^* случайным образом, например, с равномерным распределением.

4) *Спуск в направлении первый подходящий.* Поиск соседнего решения завершается, как только обнаружен первый элемент из множества N^*. В отличие от предыдущих правил в данном случае не требуется просмотра всей окрестности. По смыслу это правило близко к правилу *случайного элемента*, если просмотр окрестности начинается со случайной

точки. Однако чаще всего используется один и тот же порядок, например, лексикографический, и поиск начинается с наименьшего элемента.

5) *Спуск по круговому правилу* [7]. Просмотр окрестности начинается с того места, где был найден элемент на предшествующем шаге и заканчивается на первом найденном элементе из множества N^*. Это правило основано на том наблюдении, что для многих задач при переходе к соседнему решению целевая функция в окрестности меняется незначительно. Скорее всего, невыгодные решения в окрестности $N(y)$ будут невыгодными и в окрестности соседнего решения. Поэтому лучше продолжить просмотр, чем повторять его сначала. Конечно, данные соображения нужно проверять для каждой задачи отдельно.

6) *Спуск по правилу максимальной свободы.* Для каждого элемента $y' \in N^*(y)$ вычисляется мощность множества $N^*(y')$, так называемая «свобода», а затем в множестве $N^*(y)$ выбирается элемент с наибольшей свободой для дальнейшего движения вниз.

Любое из этих правил может быть использовано при реализации генетического локального поиска. Для сравнения правил проведён вычислительный эксперимент на случайных матрицах. Каждый элемент матрицы (c_{ij}) выбирался из интервала $[0, \ldots, 1000]$ с равномерным распределением и независимо от других элементов. Матрица приоритетов (g_{ij}) на 70% совпадала с матрицей (c_{ij}) и в каждом столбце содержала только различные элементы. Цель эксперимента состояла в сравнении правил замещения по точности и числу итераций для получения локальных оптимумов, а также в выработке рекомендаций для генетических алгоритмов.

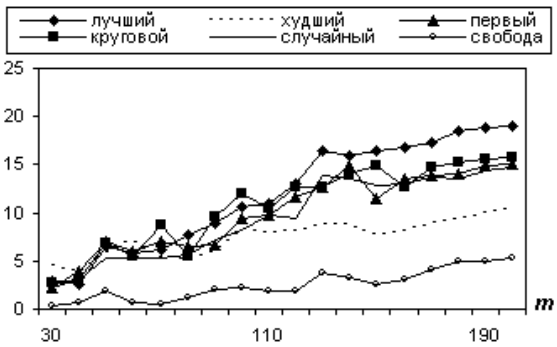

Рис. 1. Средняя относительная погрешность (%), $p = m/10$

Для каждого $m = 30, 40, \ldots, 200$ генерировалось 10 тестовых примеров с $p = m/10$ и $n = m$. Для каждого из них порождалось 1000 случайных допустимых решений. К каждому допустимому решению применя-

лась стандартная процедура локального спуска с окрестностью 1-*замена* и одним из шести правил замещения. Поэтому для каждого примера получалось 6000 локальных оптимумов. Лучшее среди них обозначим через y^*. Так как точное решение задачи даже при небольшой размерности $n = m = 50$ не удаётся найти с помощью коммерческого программного обеспечения, то узнать отклонение y^* от оптимума не представляется возможным. Однако для сравнения правил замещения этого не требуется. На рис. 1 показана средняя погрешность $\varepsilon = (F(y) - F(y^*))/F(y^*)$ получаемых локальных оптимумов относительно y^*. Каждая точка на графике показывает среднее значение по 10^4 испытаниям. Первое правило приводит к наибольшей погрешности. Правила 3–5 дают примерно одинаковые результаты. Наименьшая погрешность соответствует последнему правилу, которое является более трудоёмким, чем предыдущие.

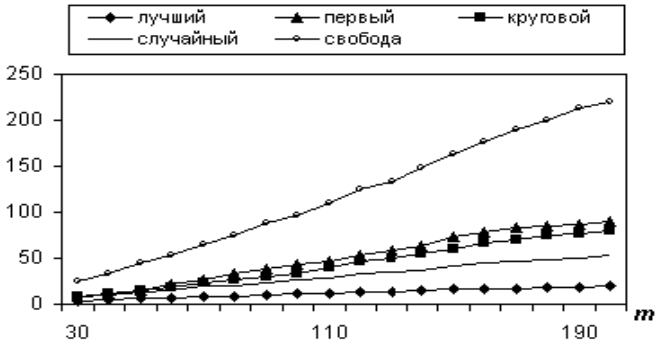

Рис. 2. Среднее число шагов, $p = m/10$

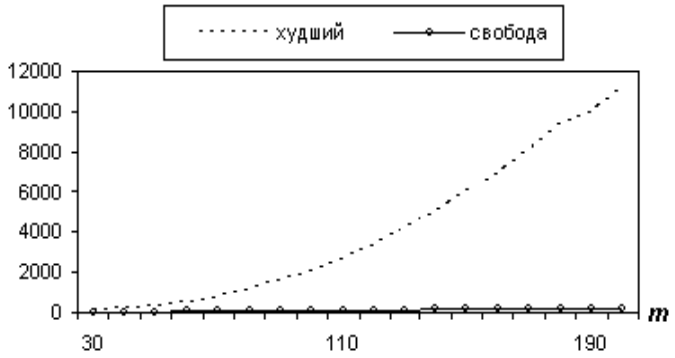

Рис. 3. Среднее число шагов, $p = m/10$

Зависимость среднего числа шагов локального спуска от размерности задачи показана на рис. 2 и 3. Для всех правил, кроме второго, число ша-

гов растёт как линейная функция. Второе правило приводит к наибольшему числу шагов локального спуска. Так, например, при $n = m = 200$ среднее число шагов для шестого правила не превышает 200, а для второго правила оно не меньше 8000. Таким образом, выбор правила замещения играет важную роль как с точки зрения трудоёмкости, так и точности получаемых локальных оптимумов. По-видимому, правила 3–4 являются вполне подходящими для применения в генетических алгоритмах.

3.3. Расположение локальных оптимумов

В [6] установлено, что задача локального поиска для p-медианы с окрестностью Лина-Кернигана также является PLS-полной. На сегодняшний день не известно ни одного полиномиального алгоритма нахождения локального оптимума для данной окрестности. Тем не менее практическое использование больших окрестностей и, в частности, окрестности Лина–Кернигана, даёт прекрасные результаты как для классической задачи о p-медиане, так и для других трудных комбинаторных задач [10, 21].

Чтобы исследовать возможности локального поиска для задачи МПК, был проведён следующий эксперимент. Его цель состояла в исследовании свойств ландшафта задачи и взаимного расположения локальных оптимумов. Напомним, что под ландшафтом понимают взвешенный ориентированный граф, вершины которого соответствуют допустимым решениям, а дуга $e = (v_1, v_2)$ присутствует в графе тогда и только тогда, когда v_2 смежна с v_1 и $F(v_1) > F(v_2)$ (для задач на максимум используется обратное неравенство). Локальные оптимумы оптимизационной задачи соответствуют стоковым вершинам в этом графе. Если генетический алгоритм «работает» только с локальными оптимумами, то возникают следующие вопросы. Как быстро можно перейти от одного локального оптимума к другому? Как много шагов с ухудшением нужно сделать, чтобы алгоритмом локального спуска можно было получить другой (лучший) локальный оптимум? Правда ли, что из «плохого» локального оптимума легко найти путь к «хорошему»?

Для ответа на эти вопросы использовалась следующая процедура. Пусть y – локальный оптимум для окрестности 1-*замена* и y'– соседнее решение, отличающееся по координатам i_1 и i_2. Применим к решению y' процедуру локального спуска, не меняя этих координат. К полученному таким образом условному локальному оптимуму снова применим процедуру локального спуска, убрав запрет на i_1 и i_2. В результате получим локальный оптимум y'', который может совпадать с y, а может

быть лучше или хуже его. Применим эту процедуру к каждому соседнему решению y' и для определённости используем четвёртое правило замещения. Диаграммы на рис. 4 ($4a$–$4e$) показывают среднее число различных (всех) локальных оптимумов с меньшим, равным и большим значением целевой функции в абсолютных числах и в процентах для одного из наиболее трудных примеров библиотеки Резенде, Вернека [25] при $n = m = 250, p = 25$. Матрица (g_{ij}) строилась так же, как и в предыдущем случае.

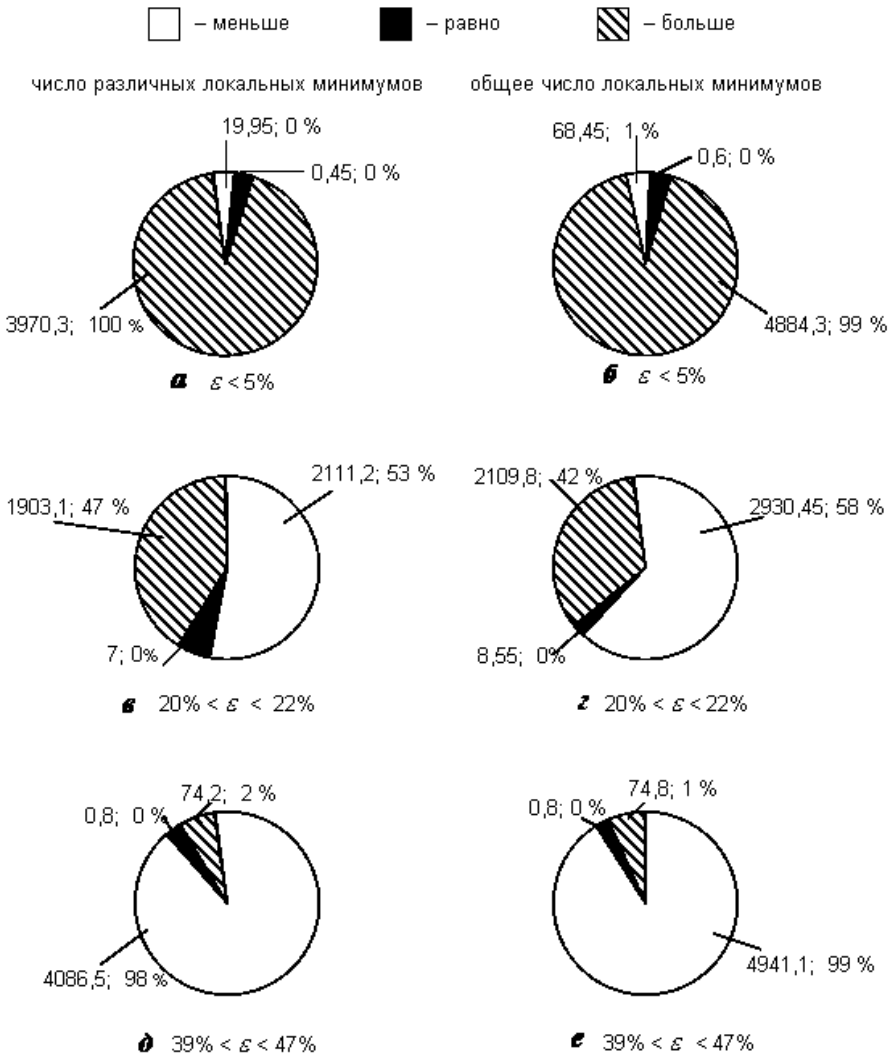

Рис. 4. Расположение локальных минимумов, $m = n = 250, p = 25$

Диаграммы 4*а*, 4*б* соответствуют малой погрешности $\varepsilon < 5\%$ стартового локального оптимума. Диаграммы 4*в*, 4*г* и 4*д*, 4*е* соответствуют средней ($20\% < \varepsilon < 22\%$) и большей ($39\% < \varepsilon < 47\%$) погрешностям. Результаты проведённого эксперимента показывают, что чем больше погрешность локального оптимума, тем легче найти с помощью данной процедуры лучший локальный оптимум. Даже при малой погрешности это удаётся сделать, хотя процент лучших локальных оптимумов оказывается малым (см. рис. 4*а*, 4*б*). При большой погрешности (см. рис. 4*д*, 4*е*) почти каждый новый локальный оптимум имеет меньшую погрешность, чем y. На рис. 5 показано среднее число получаемых локальных оптимумов на заданном расстоянии от y. Интересно отметить, что наибольшее число локальных оптимумов оказалось на расстоянии 40 при диаметре допустимой области 50 и не нашлось ни одного локального оптимума ближе 28. Другими словами, ландшафт в данном примере устроен таким образом, что стоит сделать один шаг с ухудшением даже из хорошего локального оптимума ($\varepsilon < 5\%$) и можно получить новый локальный оптимум, достаточно далекий от исходного. Остановка в произвольном локальном оптимуме может приводить к большим погрешностям, но использование больших окрестностей, типа окрестности Лина–Кернигана, должно исправлять этот недостаток.

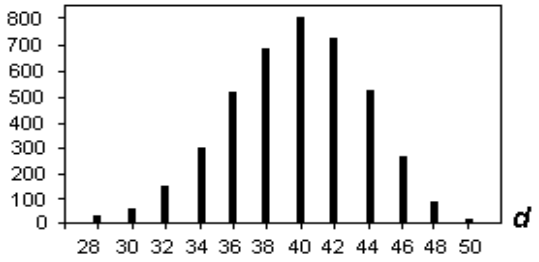

Рис. 5. Среднее число локальных минимумов на расстоянии d, $\varepsilon < 5\%$

4. Численные эксперименты

Разработанный генетический алгоритм тестировался на примерах из электронной библиотеки «Дискретные задачи размещения» (http://math.nsc.ru/AP/benchmarks/). Размерность примеров составляла $n = m = 100$. Матрица (c_{ij}) порождалась случайным образом (класс GAP-C, [21]) и имела низкую плотность: ровно 10 элементов в каждом столбце и строке имели целые значения от 0 до 4. Остальные элементы полагались равными достаточно большому числу. Матрица приоритетов (g_{ij}) формировалась в два этапа. Сначала каждый столбец матрицы

(c_{ij}) упорядочивался по неубыванию $c_{i_1 j} \leqslant c_{i_2 j} \leq \ldots \leqslant c_{i_m j}$ и полагалось $g_{i_k j} = k, k = 1, \ldots m$. Затем в каждом столбце матрицы (g_{ij}) среди элементов со значением от 0 до 4 случайным образом выбирались три пары элементов и они менялись друг с другом своими значениями. Число p полагалось равным 14. Это минимальное число, при котором всё ещё удаётся обслужить всех клиентов, не используя большие значения в матрице (c_{ij}). Нахождение оптимального решения задачи с помощью коммерческого программного обеспечения GAMS оказалось трудоемким: после 270 часов работы пакета на PC Pentium IV разрыв между верхними и нижними оценками составил около 18% и оптимальное решение всё ещё не было получено. Более того, найденное решение было хуже решения, полученного генетическим алгоритмом за 2 минуты на том же компьютере. Это свидетельствует о более высокой эффективности разработанного генетического алгоритма. В качестве оптимизационной процедуры в GAMS использовался пакет CPLEX 6.0.

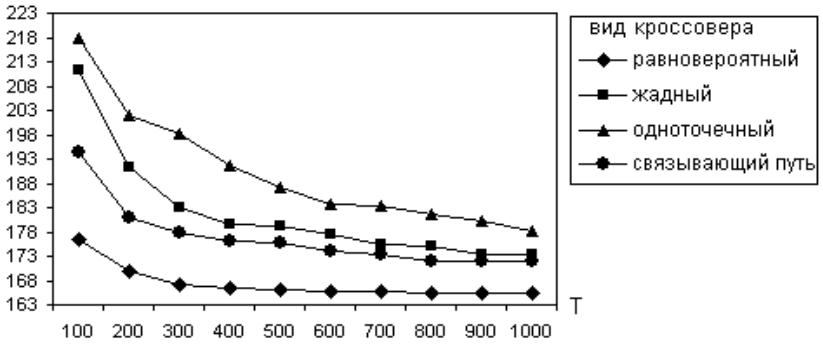

Рис. 6. Средние значения F^* при турнирной селекции, $p_m = 0{,}02$

4.1. Выбор параметров алгоритма

Поведение генетического алгоритма существенно зависит от его параметров и применяемых операторов селекции, скрещивания и мутации. Для повышения эффективности алгоритма был проведён следующий эксперимент. При заданном числе T получаемых локальных оптимумов проводилось сравнение лучших элементов в популяции при разных способах селекции и скрещивания. На рис. 6 показано изменение этих значений в ходе эволюции при турнирной селекции и вероятности мутации, равной 0,02. Каждая точка на графике показывает среднее значение по 10 примерам и 100 испытаниям генетического алгоритма для каждого примера. Популяция содержала 20 локальных оптимумов. Для выбора

каждого родителя из популяции случайным образом с равномерным распределением выбирались 3 элемента и лучший из них по целевой функции становился родителем. Наихудшие результаты получились, как и ожидалось, при одноточечном операторе скрещивания. Его отставание от других велико и объясняется, по-видимому, неудачностью самой конструкции. Неожиданно хорошие результаты показал самый простой, равномерный оператор, который доминирует во всех экспериментах на данном классе. Его успех, наверное, объясняется необычайной сложностью самого класса тестовых примеров, где нахождение глобального оптимума или хорошего приближения представляется весьма сложным делом. Жадные стратегии или стратегии связывающих путей, показавшие хорошие результаты на других задачах [15], здесь явно проигрывают.

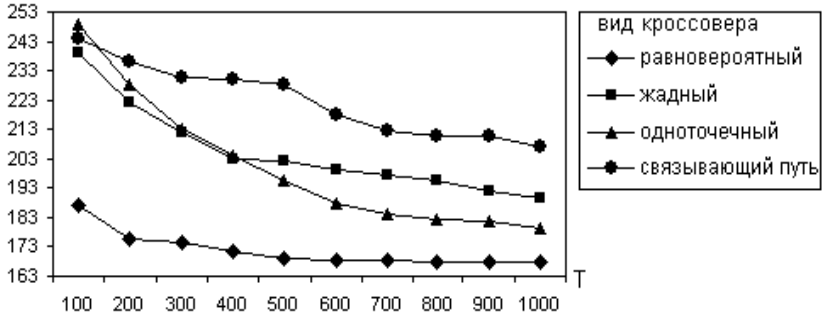

Рис. 7. Средние значения F^* при селекции «лучший и случайный», $p_m = 0,02$

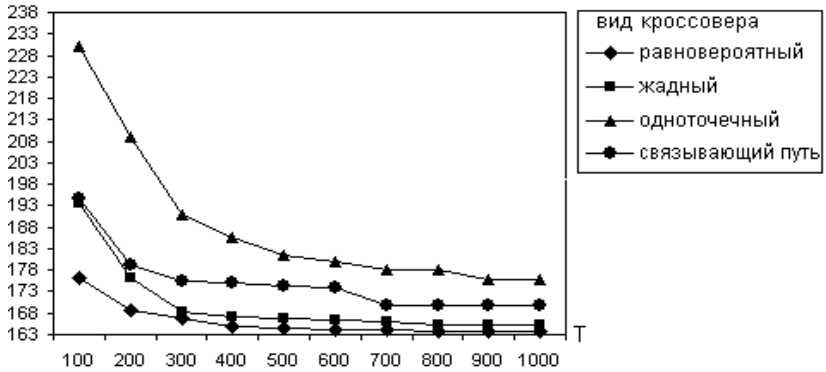

Рис. 8. Средние значения F^* при равновероятной селекции, $p_m = 0,02$

На рис. 7 и 8 представлены результаты расчётов для двух других способов селекции. Стратегия «лучший и случайный» проигрывает двум

другим, а равновероятная селекция даёт, как и равновероятное скрещивание, лучшие результаты. Интересно отметить, что повышение вероятности мутации (рис. 9 и 10) приводит к сглаживанию различий между операторами скрещивания и даже отсутствие скрещивания (рис. 10 и 11) при наличии высокой вероятности мутации приводит к хорошим результатам. Тем не менее удаление из алгоритма операторов скрещивания не является оправданным (см. рис. 11). Для примеров Резенде и Вернека большой размерности $n = m = 500$, $p = 50$ [25] генетический алгоритм с любым из рассмотренных операторов скрещивания даёт лучшие результаты, чем без этих операторов.

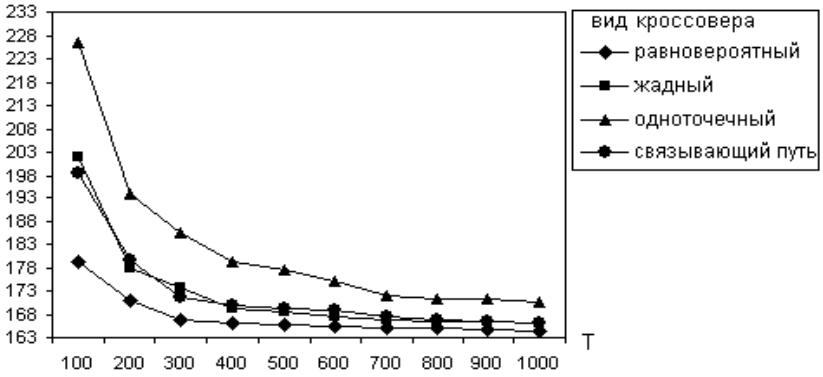

Рис. 9. Средние значения F^* при турнирной селекции, $p_m = 0,1$

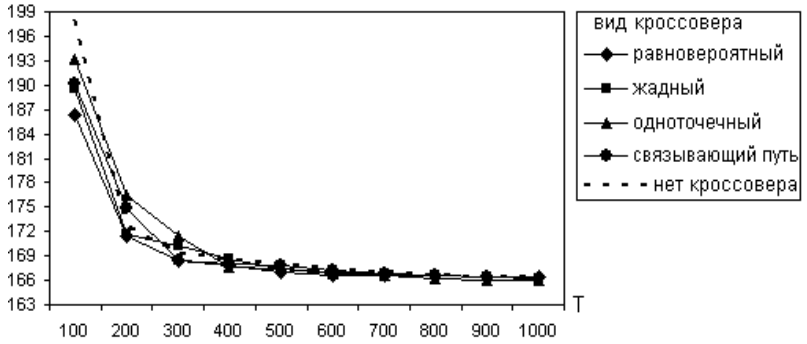

Рис. 10. Средние значения F^* при турнирной селекции, $p_m = 0,5$

4.2. Сравнение нижних оценок

Так как оптимальное решение найти не удалось, то представляет интерес вычисление нижних оценок оптимума и, в частности, оценок LB_4 и LB_6. В табл. 1 приводятся результаты расчётов для тридцати тестовых

примеров. Наряду с указанными нижними оценками линейного программирования приведены еще оценки LB_7, получающиеся точным решением задачи о p-медиане без учёта предпочтений клиентов. Очевидно, что это действительно нижняя оценка, так как учёт предпочтений клиентов может только увеличить значение оптимума. Заметим, что получение оценки LB_7 предполагает точное решение NP-трудной задачи о p-медиане, что для данного класса тестовых примеров является очень трудоёмкой процедурой. Для сравнения приводятся верхние оценки, полученные разработанным генетическим алгоритмом, GA и разрыв Gap между верхними и наилучшими нижними оценками в процентах. Заметим, что этот разрыв достаточно велик, поэтому требуются дальнейшие исследования.

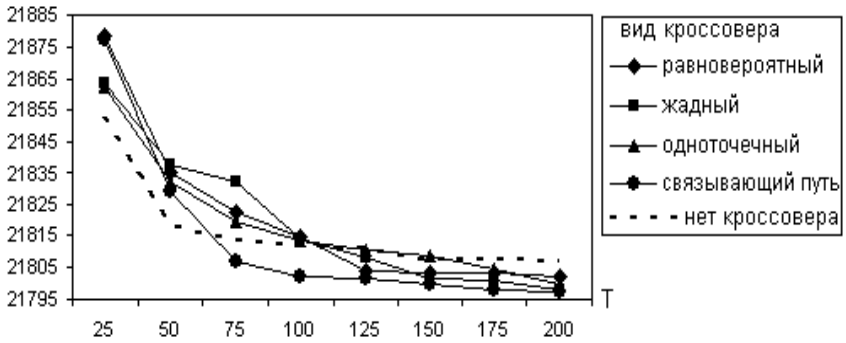

Рис. 11. Средние значения F^* при турнирной селекции, $p_m = 0,5$, класс *Euclidean*

5. Дальнейшие исследования

В рассматриваемой математической модели предполагалось, что на верхнем уровне только один игрок принимает решение о размещении своих предприятий, и в этом смысле конкуренция на рынке отсутствует. Наличие нескольких игроков, размещающих предприятия и стремящихся к максимизации доходов (или минимизации издержек), значительно обогащает модель. При этом возникает не менее двух различных направлений. При одном направлении игроки независимы друг от друга, не образуют коалиций и придерживаются эгоистических стратегий. В этом случае представляет интерес оценка равновесий по Нэшу и алгоритмы получения таких равновесий [23, 28]. Ко второму направлению можно отнести модели двухуровневого программирования. В таких моделях игрок на верхнем уровне ищет оптимальное решение в условиях, когда он первым открывает предприятия, а конкуренты, зная это решение, открывают собственные предприятия, преследуя свои интересы на

том же рынке [5]. Оба направления являются развитием исследуемой в статье модели и представляются авторам интересными для дальнейших исследований.

Т а б л и ц а 1. Сравнение верхних и нижних оценок

Код задачи	LB_6	LB_4	GA	LB_7	Gap(%)
333	118,6	113,1	172	147	17,01
433	115,9	110,5	156	145	7,59
533	123,6	119,1	188	177	6,21
633	118,5	112,2	165	144	12,73
733	112,0	105,7	159	137	16,06
833	122,5	118,5	170	144	18,06
933	110,4	104,8	160	130	23,08
1033	107,6	104,4	159	138	15,22
1133	114,6	109,1	163	147	10,88
1233	112,4	108,2	163	142	14,79
1333	114,5	108,6	168	140	20
1433	113,6	108,2	172	152	13,16
1533	105,5	101,4	152	133	14,29
1633	110,9	105,5	156	141	10,64
1733	104,8	99,6	152	134	13,43
1833	110,6	103,9	154	139	10,79
1933	114,3	108,6	158	137	15,33
2033	112,7	108,0	161	140	15,00
2133	120,3	113,2	166	138	20,29
2233	108,0	101,5	154	121	27,27
2333	113,1	105,3	155	133	16,54
2433	152,2	109,8	155	139	11,51
2533	105,4	99,0	147	131	12,21
2633	110,4	104,3	156	132	18,18
2733	113,3	109,3	159	139	14,39
2833	107,8	101,8	161	137	17,52
2933	104,3	97,2	152	124	22,58
3033	112,2	108,1	157	137	14,60
3133	103,8	99,4	155	141	9,93
3233	108,2	102,7	155	129	20,16

ЛИТЕРАТУРА

1. **Береснев В. Л., Гимади Э. Х., Дементьев В. Т.** Экстремальные задачи стандартизации. Новосибирск: Наука, 1978.

2. **Горбачевская Л. Е.** Полиномиально разрешимые и NP-трудные двухуровневые задачи стандартизации. Дисс. ... канд. физ.-мат. наук. Новосибирск, 1998.

3. Горбачевская Л. Е., Дементьев В. Т., Шамардин Ю. В. Двухуровневая задача стандартизации с условием единственности оптимального потребительского выбора // Дискрет. анализ и исслед. операций. Сер. 2. 1999. Т. 6, № 2. С. 3–11.

4. Еремеев А. В. Генетический алгоритм для задачи о покрытии // Дискрет. анализ и исслед. операций. Сер. 2. 2000. Т. 7, № 1. С. 47–60.

5. Кочетов Ю. А. Двухуровневые задачи размещения // Труды Института вычислительной математики и математической геофизики СО РАН. Сер. Информатика. Вып. 6. Новосибирск, 2007. С. 25–31.

6. Кочетов Ю. А., Пащенко М. Г., Плясунов А. В. О сложности локального поиска в задаче о p–медиане // Дискрет. анализ и исслед. операций. Сер. 2. 2005. Т. 12, № 2. С. 44–71.

7. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. М.: Мир, 1985.

8. Растригин Л. А. Случайный поиск — специфика, этапы истории и предрассудки // Вопросы кибернетики. М.: Научный совет по комплексной проблеме «Кибернетика» АН СССР. 1978. Вып. 33. С. 3–16.

9. Aggarwal C. C., Orlin J. B., Tai R. P. Optimized crossover for maximum independent set // Oper. Res. 1997. V. 45, N 2, P. 226–243.

10. Ahuja R. K., Ergun O., Orlin J. B., Punnen A. P. A survey of very large–scale neighborhood search techniques // Discrete Appl. Math. 2002. V. 123, N 1–3. P. 75–102.

11. Balas E., Niehaus W. Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems // J. Heuristics. 1998. V. 4, N 4. P. 107–122.

12. Boese K. D., Kahng A. B., Muddu S. A new adaptive multi-start technique for combinatorial global optimizations // Oper. Res. Lett. 1994. V. 16, N 2. P. 101–114.

13. Borisovsky P., Dolgui A., Eremeev A. Genetic algorithms for supply management problem with lower-bounded demands // Preprints of the 12th IFAC Symposium «Information Control Problems in Manufacturing 2006», Saint-Etienne, France: Elsevier Science, 2006. V. 3. P. 535–540.

14. Bremermann H. J., Roghson J., Salaff S. Global properties of evolution processes // Natural automata and useful simulations. London: Macmillan. 1966. P. 3–42.

15. Glover F., Laguna M. Tabu search. Boston: Kluwer Acad. Publ., 1997.

16. Goldberg D. E. Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley, 1989.

17. Hanjoul P., Peeters D. A facility location problem with clients' preference orderings // Regional Science and Urban Economics. 1987. V. 17, N 3. P. 451–473.