ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Январь—июнь 1999. Серия 2. Том 6, № 1, 12–32

УДК 519.854.33

ПОВЕДЕНИЕ ВЕРОЯТНОСТНЫХ ЖАДНЫХ АЛГОРИТМОВ ДЛЯ МНОГОСТАДИЙНОЙ ЗАДАЧИ РАЗМЕЩЕНИЯ*)

Е. Н. Гончаров, Ю. А. Кочетов

Для многостадийной задачи размещения дано описание вероятностных жадных алгоритмов, основанных на идеях координатного спуска. Приведены результаты экспериментального исследования этих алгоритмов на трудных в вычислительном отношении классах исходных данных. Установлен характер зависимости относительной погрешности получаемых решений от параметров алгоритмов. Показано, что вероятность получения точного решения может быть сделана сколь угодно близкой к единице. Определены статистические значения математического ожидания погрешности, ее среднего квадратического отклонения, а также вероятности нахождения точного решения задачи и приближенного решения с погрешностью не более одного процента. Показано, что вероятностные алгоритмы имеют лучшие характеристики, чем их детерминированные аналоги, хотя и являются более трудоемкими.

Введение

Многостадийная задача размещения относится к числу NP-трудных в сильном смысле задач дискретной оптимизации. Она является обобщением классической задачи о минимальном покрытии множествами [7], а для последней маловероятно существование полиномиального алгоритма, максимальная относительная погрешность которого росла бы медленнее логарифмической функции от размерности задачи [10]. Столь мрачная перспектива свидетельствует, по нашему мнению, не столько о трудности самой задачи, сколько о критерии оценки качества алгоритмов. Анализ поведения алгоритмов в худшем случае является малоинформативным. Результат такой оценки излишне пессимистичен.

В данной работе проводится экспериментальный анализ поведения алгоритмов в среднем. Предлагаются новые вероятностные жадные

*) Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (коды проектов 99–01–00601, 98–07–90259).

алгоритмы и приводится их сравнение по пяти критериям: математическому ожиданию относительной погрешности, ее среднему квадратическому отклонению, вероятности нахождения точного и приближенного решения с погрешностью не более одного процента, времени работы алгоритмов. Экспериментальные оценки этих характеристик позволяют, на наш взгляд, судить о реальном поведении алгоритмов и могут служить стимулом для разработки эффективных методов решения NP-трудных задач.

1. Постановка задачи

На содержательном уровне многостадийная задача размещения формулируется следующим образом. Даны множества потребителей, предприятий и ассоциаций (или производственных союзов), в которые могут объединяться предприятия, чтобы выпускать продукцию для потребителей. Известны стоимости открытия предприятий и ассоциаций и стоимости удовлетворения спроса потребителей каждой ассоциацией. Требуется найти такой набор предприятий и ассоциаций, чтобы с минимальными суммарными затратами удовлетворить запросы потребителей. В состав ассоциации может входить произвольное число предприятий. Продукция проходит обработку на всех предприятиях ассоциации, т. е. имеет много стадий обработки. На рис. 1 изображена схема из восьми предприятий, которые могут объединяться в семь ассоциаций. Выделены предприятия и ассоциации, которые образуют одно из допустимых решений задачи.

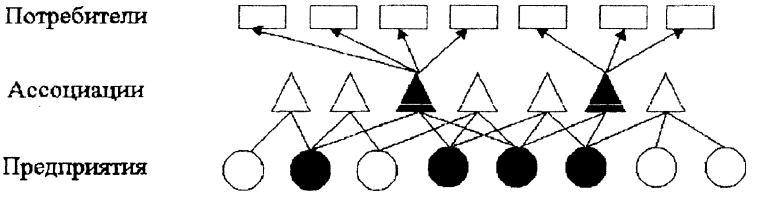


Рис. 1. Схема размещения предприятий и ассоциаций

Многие задачи дискретной оптимизации могут быть сведены к многостадийной задаче размещения. В частности, задача минимизации полиномов от булевых переменных эквивалентна данной задаче [2, 3]. Двухуровневая задача размещения предприятий, задача выбора оптимального набора строк пары матриц, задача выбора оптимальных рядов изделий и комплектующих узлов [3] также могут быть представлены в виде многостадийной задачи размещения.

Приведем математическую постановку задачи в терминах целочисленного линейного программирования. Введем следующие обозначения:

$J = \{1, \ldots, \mathbf{J}\}$ — множество потребителей;

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

$L = \{1, \ldots, \mathbf{L}\}$ — множество ассоциаций;

$I_l \subset I$ — множество предприятий, которые входят в l-ю ассоциацию;

$L_i \subset L$ — множество ассоциаций, в которые входит i-е предприятие;

$f_i \geqslant 0$ — стоимость открытия i-го предприятия;

$g_l \geqslant 0$ — стоимость открытия l-й ассоциации;

$c_{lj} \geqslant 0$ — стоимость удовлетворения спроса j-го потребителя l-й ассоциацией.

Переменные задачи:

$$z_l = \begin{cases} 1, & \text{если принимается решение об открытии } l\text{-й ассоциации,} \\ 0 & \text{в противном случае,} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{если принимается решение об открытии } i\text{-го предприятия,} \\ 0 & \text{в противном случае,} \end{cases}$$

$$x_{lj} = \begin{cases} 1, & \text{если спрос } j\text{-го потребителя удовлетворяется } l\text{-й ассоциацией,} \\ 0 & \text{в противном случае.} \end{cases}$$

С использованием введенных обозначений многостадийная задача размещения может быть записана следующим образом [5]:

$$\min_{x,y,z} \left\{ \sum_{l \in L} g_l z_l + \sum_{i \in I} f_i y_i + \sum_{j \in J} \sum_{l \in L} c_{lj} x_{lj} \right\}, \tag{1}$$

$$\sum_{l \in L} x_{lj} = 1, \quad j \in J, \tag{2}$$

$$z_l \geqslant x_{lj}, \quad l \in L, \quad j \in J, \tag{3}$$

$$y_i \geqslant \sum_{l \in L_i} x_{lj}, \quad i \in I, \quad j \in J, \tag{4}$$

$$z_l,\ y_i,\ x_{lj} \in \{0, 1\}, \quad l \in L, \quad i \in I, \quad j \in J. \tag{5}$$

Целевая функция (1) имеет смысл суммарных затрат на открытие предприятий, ассоциаций и удовлетворения спроса потребителей. Равенства (2) гарантируют удовлетворение спроса всех потребителей. Условия (3), (4) требуют внесения затрат на открытие ассоциаций и предприятий, если они участвуют в удовлетворении спроса.

Для исследования алгоритмов решения задачи (1)–(5) выделим четыре класса (R, R_0, E, E_0) тестовых задач:

R: $g_l = 3000$, $f_i = 3000$, $c_{lj} \in [0, 1000]$;
R_0: $g_l = 0$, $f_i = 3000$, $c_{lj} \in [0, 1000]$;
E: $g_l = 3000$, $f_i = 3000$, c_{lj}— евклидовы расстояния;
E_0: $g_l = 0$, $f_i = 3000$, c_{lj}— евклидовы расстояния.

Задачи имеют следующую размерность: $\mathbf{J} = 100$, $\mathbf{I} = 50$, $\mathbf{L} = 100$, $|I_l| = 4$, $l \in L$. Различие между классами состоит в матрице (c_{lj}) и векторе (g_l). В классах R, R_0 величины c_{lj} выбираются случайно с равномерным распределением на интервале [0,1000]. В классах E, E_0 величины c_{lj} являются евклидовыми расстояниями на плоскости между точками l и j. Координаты точек выбираются случайно с равномерным распределением в квадрате 1000×1000.

Указанные классы задач являются трудными для метода ветвей и границ [5] и требуют для точного решения от 10 до 120 мин машинного времени (здесь и далее время счета указано для PC Pentium-166). Если величины g_l, f_i выбирать случайно из некоторого интервала, то тестовые задачи становятся проще. Выбор константы 3000 принципиального значения не имеет. Увеличение константы приводит к сокращению множества открываемых предприятий и ассоциаций, уменьшение константы — к его расширению.

В каждом классе взято 30 примеров. Для каждого из них найдено точное решение алгоритмом из [5]. Все приближенные алгоритмы тестировались на этих примерах, и результаты сравнивались с точными решениями. В итоге для каждого алгоритма получены статистические оценки следующих характеристик: $M(\varepsilon)$ — математического ожидания относительной погрешности ε, $D(\varepsilon)$ — ее среднего квадратического отклонения (квадратный корень из дисперсии), $Pr(\varepsilon = 0)$ — вероятности получения точного решения, $Pr(\varepsilon \leqslant 0,01)$ — вероятности получения приближенного решения с относительной погрешностью не более одного процента, а также среднего времени работы алгоритмов.

2. Вероятностные жадные алгоритмы

В данном разделе представлены новые вероятностные жадные алгоритмы «лидер группы» и «случайный аутсайдер», приведены их характеристики для классов задач R, R_0, E, E_0 и указаны возможные пути модификации этих алгоритмов.

Поставим в соответствие булевому вектору z_l множество $S = \{l \in L \mid z_l = 1\}$. Минимум целевой функции (1) по переменным y_i и x_{lj} при фиксированном ненулевом векторе (z_l) равен величине

$$F(S) = \sum_{l \in S} g_l + \sum_{i \in K} f_i + \sum_{j \in J} \min_{l \in S} c_{lj},$$

где $K = \{i \in I \mid L_i \cap S \neq \varnothing\}$. Таким образом, задача (1)–(5) может быть записана как задача минимизации функции $F(S)$ по подмножествам $S \subseteq L$, $S \neq \varnothing$.

Одним из наиболее простых и естественных способов приближенного решения этой задачи является алгоритм координатного спуска (КС). Он начинает свою работу с нулевого решения. На каждой итерации алгоритма выбирается некоторая координата $l \in L$ и соответствующая переменная z_l полагается равной единице. Выбор координаты осуществляется по принципу наибольшего уменьшения значения целевой функции. Если на некоторой итерации увеличение любой координаты не приводит к уменьшению целевой функции, то алгоритм заканчивает свою работу. Обозначим через $\rho_l = F(S) - F(S \cup \{l\})$ изменение целевой функции при увеличении l-й координаты и положим $M = \sum\limits_{l \in L} g_l + \sum\limits_{i \in I} f_i + \sum\limits_{j \in J} \max\limits_{l \in L} c_{lj}$. Алгоритм координатного спуска заключается в следующем.

Алгоритм КС

Шаг 1. Полагаем $S := \varnothing$, $F(\varnothing) := M$.

Шаг 2. Вычисляем ρ_l, $l \in L$.

Шаг 3. Находим координату l^* с наибольшим приращением $\rho_{l^*} = \max\{\rho_l \mid l \in L\}$. Если $\rho_{l^*} \leqslant 0$, то переходим на шаг 5.

Шаг 4. Добавляем координату l^* в множество S и возвращаемся на шаг 2.

Шаг 5. Полагаем $F_{\text{КС}} := F(S)$.

Алгоритм КС является детерминированным полиномиальным алгоритмом и, по-видимому, его относительная погрешность

$$\varepsilon = (F_{\text{КС}} - F^*)/F^*, \quad \text{где } F^* \text{ — оптимум задачи,}$$

в худшем случае растет не медленнее логарифмической функции от размерности задачи [10]. Однако в среднем относительная погрешность алгоритма невелика. Для классов R, R_0, E, E_0 характеристики алгоритма (в процентах) приведены в табл. 1. Среднее время работы алгоритма 0,02 с.

Таблица 1

Характеристики алгоритма КС

Характеристика	R	R_0	E	E_0
$M(\varepsilon)$	3,38	7,34	2,06	10,22
$D(\varepsilon)$	2,77	4,77	2,91	10,87
$Pr(\varepsilon = 0)$	16,6	10,0	30,0	0
$Pr(\varepsilon \leqslant 0,01)$	26,6	10,0	46,6	0

2.1. Алгоритм «лидер группы»

Приведем описание вероятностной версии алгоритма КС, которая имеет в среднем меньшую погрешность и большую вероятность получения точного решения. Единственное отличие этого алгоритма от предыдущего состоит в том, что на шаге 2 величины ρ_l вычисляются не для всего множества L, а только для его части L', выбранной случайно из множества $L \setminus S$. Пусть величина p задает вероятность включения номера $l \in L$ в подмножество L'. Вероятностный алгоритм, назовем его «лидер группы» (ЛГ), может быть записан следующим образом.

Алгоритм ЛГ

Шаг 1. Полагаем $S := \varnothing$, $F(\varnothing) := M$.

Шаг 2. Формируем случайным образом подмножество L' и вычисляем ρ_l, $l \in L'$.

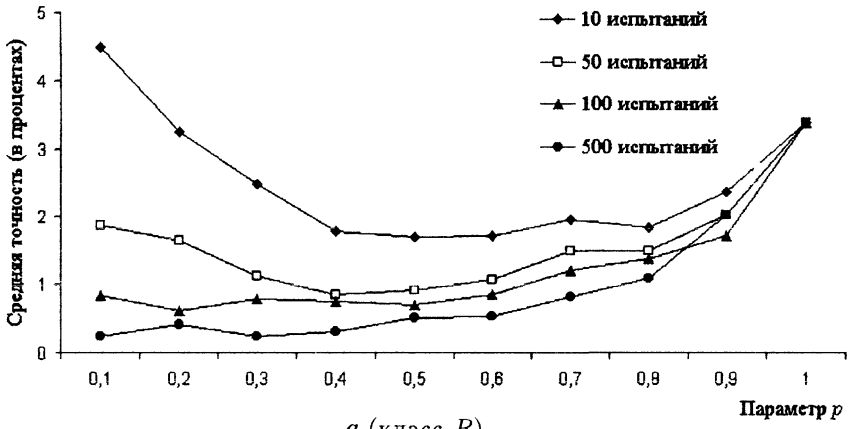
Шаг 3. Находим координату l^*, для которой $\rho_{l^*} = \max\{\rho_l \mid l \in L'\}$. Если $\rho_{l^*} \leqslant 0$ или $L' = \varnothing$, то переходим на шаг 5.

Шаг 4. Добавляем координату l^* в множество S и возвращаемся на шаг 2.

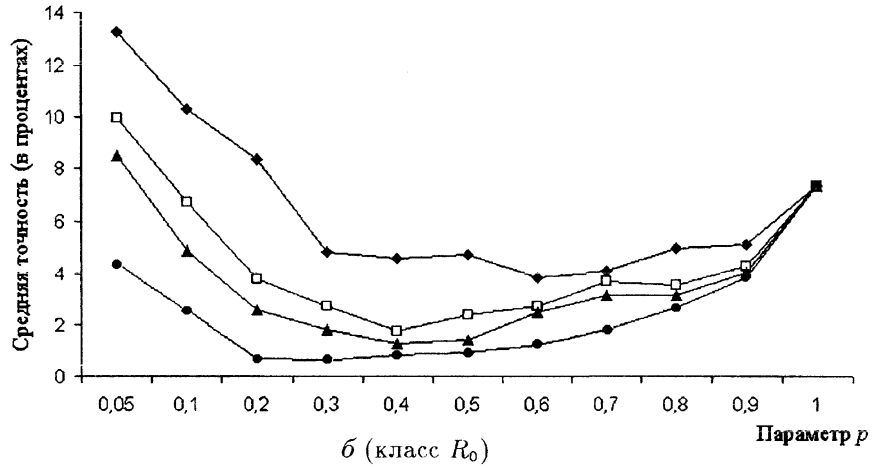
Шаг 5. Полагаем $F_{\text{ЛГ}}(p) := F(S)$.

Результатом работы алгоритма является случайная величина $F_{\text{ЛГ}}(p)$. Применяя алгоритм k раз и выбирая из найденных решений наилучшее, получаем величину $F_{\text{ЛГ}}(p, k)$. Очевидно, что $F_{\text{ЛГ}}(1, k) = F_{\text{КС}}$. Если $p < 1$ и число испытаний k достаточно велико, то погрешность такого вероятностного алгоритма оказывается меньше погрешности детерминированного алгоритма (рис. 2).

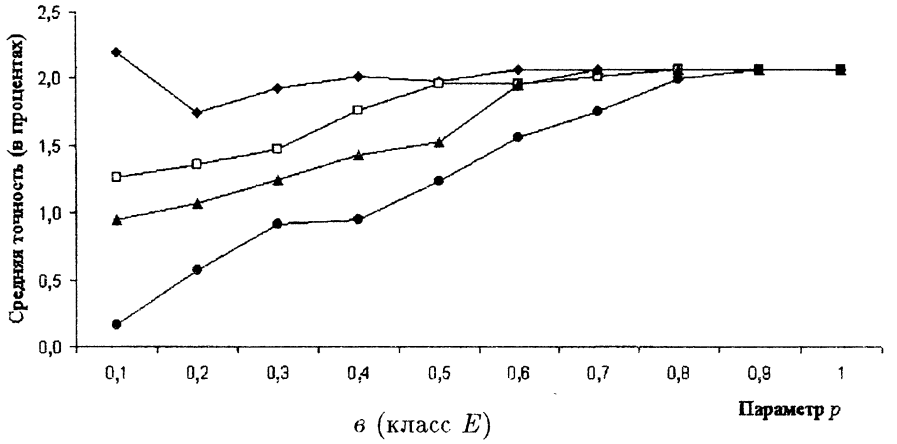
При $k = 500$ и $0,2 \leqslant p \leqslant 0,7$ величина $F_{\text{ЛГ}}(p, k)$ существенно меньше значения $F_{\text{КС}}$. Интересно отметить, что характер зависимости средней относительной погрешности алгоритма от вероятности p различен для классов R, R_0 и E, E_0. График этой зависимости напоминает выпуклую функцию для классов R, R_0. Для классов E, E_0 при $p \geqslant 0,2$ зависимость



а (класс R)



б (класс R_0)



в (класс E)

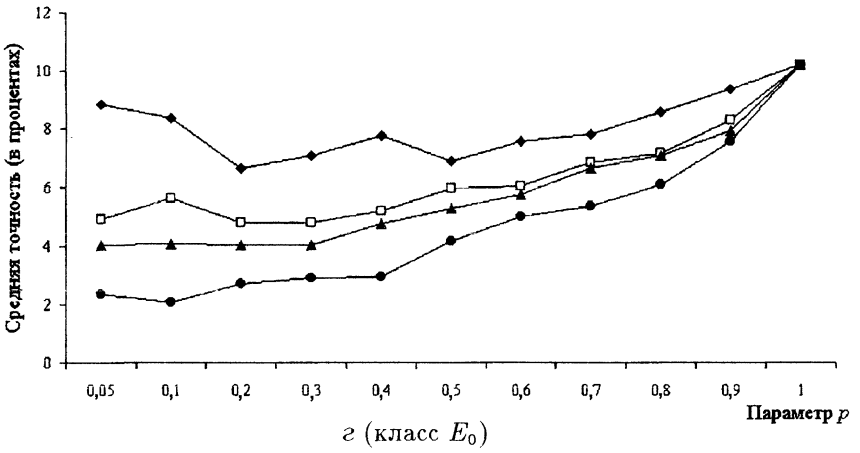


Рис. 2. Зависимость средней точности алгоритма ЛГ от параметра p

от p монотонная. В среднем погрешность алгоритма на классе E_0 всегда больше, чем на классе E, и погрешность на классе R_0 больше, чем на классе R. В этом смысле классы E_0 и R_0 более трудные для жадных алгоритмов, чем классы E и R.

Таблица 2

Характеристики алгоритма ЛГ ($k = 100$)

Характеристика	R	R_0	E	E_0
$M(\varepsilon)$	0,61	1,27	0,95	4,03
$D(\varepsilon)$	1,15	1,67	2,06	5,23
$Pr(\varepsilon = 0)$	66,6	49,0	53,3	13,6
$Pr(\varepsilon \leqslant 0,01)$	80,0	62,3	80,0	23,9

На каждой итерации алгоритма ЛГ случайным образом формируется множество L' и для каждого элемента этого множества вычисляется приращение целевой функции. В среднем этот шаг алгоритма в $1/p$ раз менее трудоемкий, чем шаг алгоритма КС. Многократные испытания алгоритма ЛГ увеличивают время его работы, но позволяют сократить погрешность и увеличить вероятность нахождения точного решения. При $k = 100$ характеристики алгоритма для классов задач R, R_0, E, E_0 значительно превосходят соответствующие показатели детерминированного алгоритма КС (табл. 2). Среднее время работы алгоритма 1,5 с.

Критерии завершения работы алгоритмов ЛГ и КС не являются самыми «жадными» из возможных. Если на некотором шаге алгоритма все $\rho_l \leqslant 0$, $l \in L$, то отсюда не следует, что множество S нельзя расширить с улучшением значения целевой функции. Рассмотрим пример. Пусть $|J| = |L| = 3$, $|I| = 4$, $g_l = 0$, $f_i = 0,6$, $L_i = \{l \in L \mid q_{li} = 1\}$ и

$$(q_{li}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad (c_{lj}) = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

Алгоритм КС сначала выберет $l^* = 1$ и получит $S = \{1\}$, $F(S) = 0 + 0,6 + 0 + 1 + 1 = 2,6$. На следующем шаге добавление любого из элементов $l = 2$, $l = 3$ приводит к увеличению целевой функции, так как $\rho_2 = \rho_3 = -0,2$. Включение же сразу двух элементов дает оптимальное решение: $F^* = 2,4$; $S = \{1,2,3\}$.

Критерий окончания работы алгоритмов можно изменить. Например, в качестве нового критерия можно взять условие $L' = \varnothing$, а результатом работы считать наилучшее из найденных решений. Покажем, что такая модификация делает алгоритм ЛГ асимптотически точным, т. е. вероятность получения оптимального решения стремится к единице с ростом числа испытаний.

Пусть (z_l^*) — оптимальное решение задачи. Без ограничения общности можно считать, что $z_l^* = 1$ при $l \leqslant s$ и $z_l^* = 0$ при $l > s$. Алгоритм начинает свою работу с нулевого решения, и на каждой итерации одна из переменных z_l становится равной 1. На первой итерации вероятность того, что z_1 станет равной 1, больше нуля, так как с вероятностью $p(1-p)^{|L|-1} > 0$ множество L' будет содержать только один элемент $\{1\}$. Аналогично на второй итерации вероятность того, что z_2 станет равной 1, также больше нуля и так далее. Следовательно, существует вероятность $q > 0$ того, что алгоритм найдет оптимальное решение. При увеличении параметра k вероятность неполучения оптимального решения, равная $(1-q)^k$, стремится к нулю, откуда и следует требуемое. Из приведенных рассуждений вытекает, что с ростом k вероятность неполучения оптимального решения убывает со скоростью геометрической прогрессии.

2.2. Условия дополняющей нежесткости

Заменим условия (5) булевости переменных z_l, y_i, x_{ij} на условия неотрицательности

$$z_l \geqslant 0, \quad y_i \geqslant 0, \quad x_{lj} \geqslant 0, \quad l \in L, \quad i \in I, \; j \in J. \tag{6}$$

Ограничениям (3) и (4) поставим в соответствие переменные u_{lj} и v_{ij}. Рассмотрим задачу, двойственную к (1)–(4), (6):

$$\max_{u,v} \sum_{j \in J} \min_{l \in L} \left(c_{lj} + u_{lj} + \sum_{i \in I_l} v_{ij} \right), \tag{7}$$

$$\sum_{j \in J} v_{ij} \leqslant f_i, \quad i \in I, \tag{8}$$

$$\sum_{j \in J} u_{lj} \leqslant g_l, \quad l \in L, \tag{9}$$

$$u_{lj} \geqslant 0, \quad v_{ij} \geqslant 0, \quad l \in L, \quad i \in I, \quad j \in J. \tag{10}$$

Условия дополняющей нежесткости для этих задач имеют вид

$$y_i \left(\sum_{j \in J} v_{ij} - f_i \right) = 0, \quad i \in I, \quad z_l \left(\sum_{j \in J} u_{lj} - g_l \right) = 0, \quad l \in L.$$

Из этих условий следует, что если $(v_{ij},\ u_{lj})$ — оптимальное решение задачи (7)–(10) и для некоторых i, l неравенства (8), (9) строгие, то в оптимальном решении задачи (1)–(4), (6) должно быть $y_i = z_l = 0$. Эта информация о решении непрерывной задачи может оказаться полезной при решении дискретной задачи [4].

В работах [3, 5] исследовалось поведение приближенных алгоритмов решения задачи (7)–(10). Эти алгоритмы имеют полиномиальную трудоемкость и строят так называемые «тупиковые» решения двойственной задачи (7)–(10), имеющие небольшую погрешность (около 10 – 15% на классах задач R, R_0, E, E_0). Тупиковое решение $(v_{ij},\ u_{lj})$ обладает тем свойством, что его нельзя улучшить за счет только увеличения значений переменных v_{ij}, u_{lj}. Точное определение тупикового решения дано в работах [3, 5].

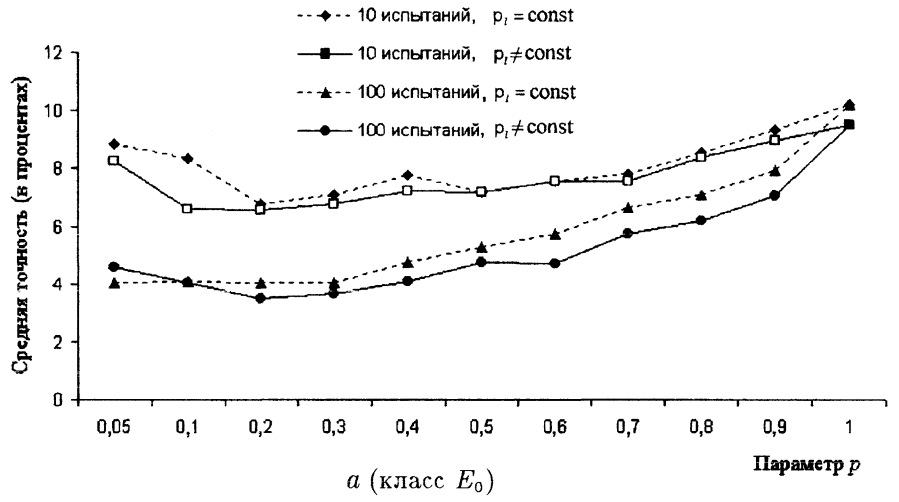
Пусть $(v_{ij},\ u_{lj})$ — тупиковое решение задачи (7)–(10). Обозначим через Δf_i и Δg_l невязки в ограничениях (8), (9):

$$\Delta f_i = f_i - \sum_{j \in J} v_{ij}, \quad i \in I, \quad \Delta g_l = g_l - \sum_{j \in J} u_{lj}, \quad l \in L,$$

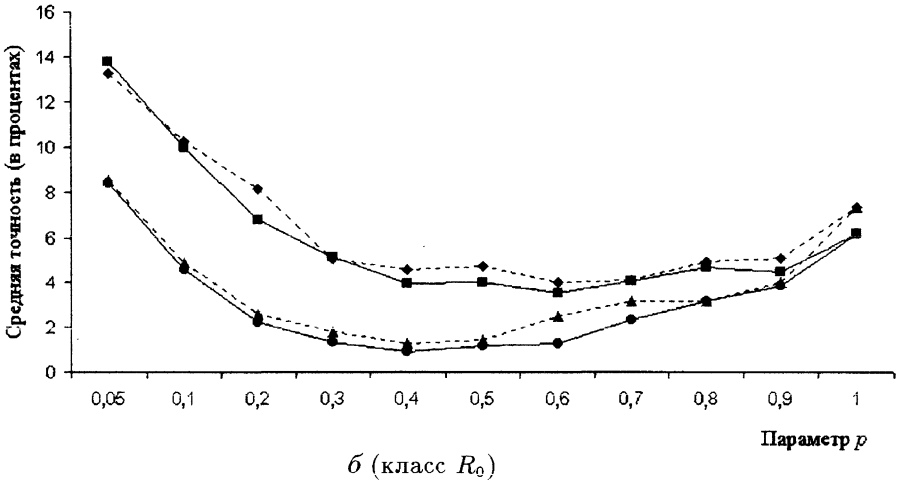
и положим

$$p_l = p \left(1 - \left(\Delta g_l + \sum_{i \in I_l} \Delta f_i \right) \Big/ \left(g_l + \sum_{i \in I_l} f_i \right) \right)^3, \quad l \in L.$$

Рассмотрим модификацию алгоритма ЛГ, в которой вместо константы p используется вектор (p_l), $l \in L$. Смысл такой замены состоит в том, чтобы уменьшить вероятность рассмотрения переменных z_l, y_i с ненулевой невязкой в ограничениях (8), (9). Влияние этой замены на



а (класс E_0)



б (класс R_0)

Рис. 3. Влияние невязок на точность алгоритма ЛГ

погрешность алгоритма изображено на рис. 3. Средняя погрешность действительно уменьшается, но говорить о кардинальных изменениях нельзя. Столь незначительное уменьшение погрешности (в среднем около 0,5 %) связано в первую очередь с большим разрывом двойственности $\delta = 100\%(F^* - \overline{F})/F^*$, где $\overline{F}$ — оптимум задачи (1)–(4), (6). На классах R, R_0, E, E_0 эта величина составляла

$$\delta_R \approx 15,2\%, \quad \delta_{R_0} \approx 33,55\%, \quad \delta_E \approx 7,06\%, \quad \delta_{E_0} \approx 24,14\%,$$

что, по-видимому, ослабляло влияние условий дополняющей нежесткости на работу алгоритма.

2.3. Алгоритм «случайный аутсайдер»

С каждой ассоциацией $l \in L$ свяжем дополнительное предприятие, стоимость открытия которого равна g_l. Этот прием позволяет свести задачу (1)–(5) к случаю, когда стоимость открытия всех ассоциаций равна нулю [5]. Далее считаем такое сведение выполненным.

Вектору (y_i) поставим в соответствие множество $K = \{i \in I \mid y_i = 1\}$ и положим $S = \{l \in L \mid I_l \subseteq K\}$. Минимум целевой функции (1) по переменным z_l, x_{lj} при фиксированном векторе (y_i) равен величине

$$F(K) = \sum_{i \in K} f_i + \sum_{j \in J} \min_{l \in S} c_{lj},$$

если $S \neq \varnothing$. Полагаем $F(K) = M$, если $S = \varnothing$. Таким образом, задачу (1)–(5) можно представить как задачу минимизации функции $F(K)$ по всем подмножествам $K \subseteq I$.

Предлагаемый ниже алгоритм, используя тупиковое решение задачи (7)–(10) и соответствующие невязки Δf_i, строит начальное решение $K_1 = \{i \in I \mid \Delta f_i = 0\}$ и затем пытается его улучшить с помощью вероятностной жадной процедуры.

Обозначим через $\rho_i = F(K) - F(K \setminus \{i\})$ приращения функции $F(K)$ на текущем решении K. Введем в рассмотрение параметры φ, $\psi \in [0, 1]$ и определим множества $I'(\varphi) = \{i \in I \mid 0 < \Delta f_i < \varphi f_i\}$ и $I''(\psi) = \left\{i \in K \mid \rho_i \geqslant \psi \max_{k \in K} \rho_k\right\}$. Алгоритм «случайный аутсайдер» (СА) последовательно сокращает множество K, удаляя из него случайным образом элементы множества $I''(\psi)$ (список аутсайдеров), и через заданное число шагов $\xi \geqslant 2$ расширяет множество K за счет элементов множества $I'(\varphi)$ (список претендентов).

Алгоритм СА

Шаг 1. Полагаем $K := K_1$, $K_2 := I'(\varphi)$, $t := 0$.

Шаг 2. Вычисляем ρ_i, $i \in K$. Если $\max\limits_{i \in K} \rho_i \leqslant 0$, то переходим на шаг 6.

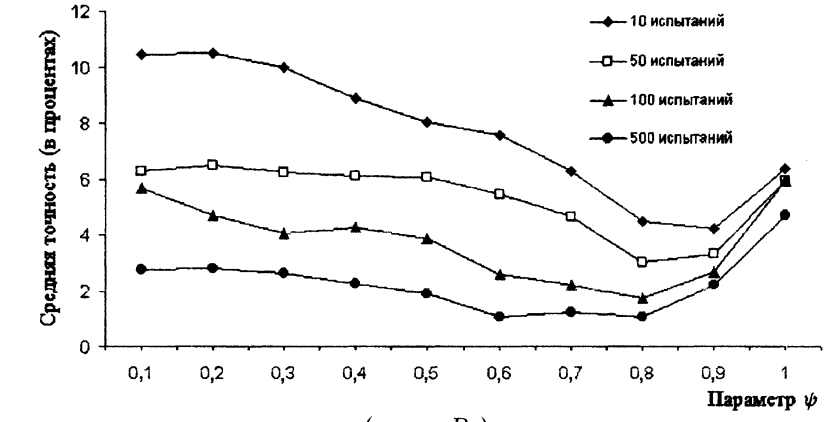
Шаг 3. Формируем множество $I''(\psi)$, выбираем случайным образом элемент $i'' \in I''(\psi)$ и удаляем его из множества K.

Шаг 4. Если $t < \xi$, то полагаем $t := t + 1$ и возвращаемся на шаг 2.

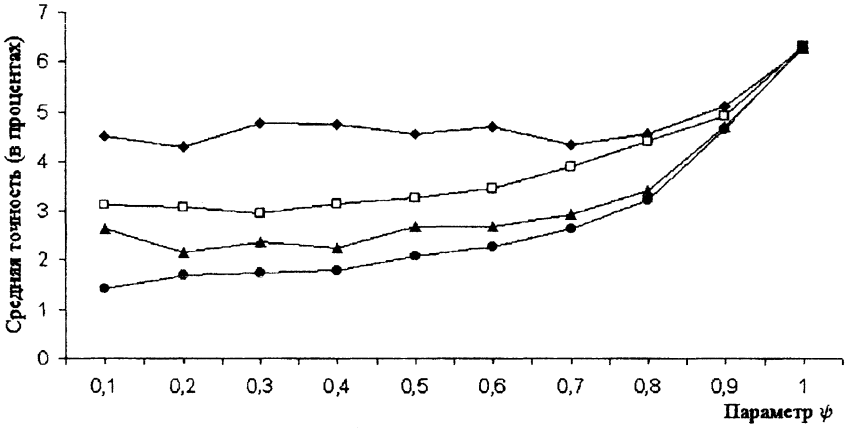
Шаг 5. Выбираем случайным образом элемент $i' \in K_2$ и переносим его из множества K_2 в множество K. Устанавливаем $t = 0$ и возвращаемся на шаг 2.

Шаг 6. Полагаем $F_{\text{CA}} := F(K)$.

Результатом работы алгоритма «случайный аутсайдер» является случайная величина F_{CA}. За счет ее дисперсии можно уменьшать



а (класс R_0)



б (класс E_0)

Рис. 4. Зависимость средней точности алгоритма СА от параметра ψ

погрешность приближенного решения задачи, увеличивая число испытаний и выбирая из найденных решений наилучшее.

Алгоритм СА устроен сложнее алгоритма ЛГ. Во-первых, сокращая множество K, алгоритм через каждые ξ шагов расширяет это множество. Так как $\xi \geqslant 2$, то это мало влияет на трудоемкость алгоритма, но позволяет повысить его точность. Во-вторых, этот алгоритм использует известное правило выбора направления спуска, предложенное в [10]. Согласно этому правилу на шаге 3 алгоритм формирует множество $I''(\psi)$, которое может быть пустым только в том случае, когда все приращения ρ_i не положительны. В этом смысле он действует аккуратнее алгоритма ЛГ.

Алгоритм СА не является асимптотически точным, но его можно сделать таковым, изменив правило выбора множества K на первом шаге. Пусть величины p_1 и p_2 задают вероятности включения в множество K элементов множеств K_1 и $I \setminus K_1$. Тогда с ненулевой вероятностью уже на первом шаге алгоритма может быть получено оптимальное решение задачи и он становится асимптотически точным.

Средняя погрешность алгоритма при различных значениях параметра ψ изображена на рис. 4. Значения $\varphi = 0,3$ и $\xi = 4$ выбраны близкими к наилучшим. Характеристики алгоритма (в процентах) для $k = 100$, $p_1 = 1$, $p_2 = 0$ и лучших значений параметра ψ приведены в табл. 3. Среднее время работы алгоритма равно 6,7 с.

Таблица 3

Характеристики алгоритма СА

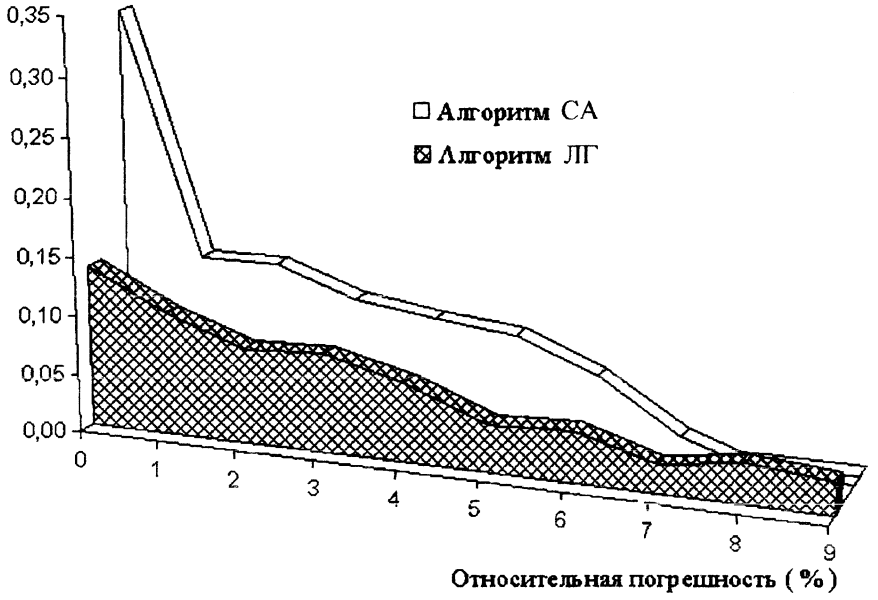
($\varphi = 0,3$, $\xi = 4$, $k = 100$)

Характеристика	R	R_0	E	E_0
$M(\varepsilon)$	2,02	1,77	2,21	2,16
$D(\varepsilon)$	1,67	1,75	2,42	2,34
$Pr(\varepsilon = 0)$	23,3	45,6	13,3	34,0
$Pr(\varepsilon \leqslant 0,01)$	36,6	57,2	50,0	47,3

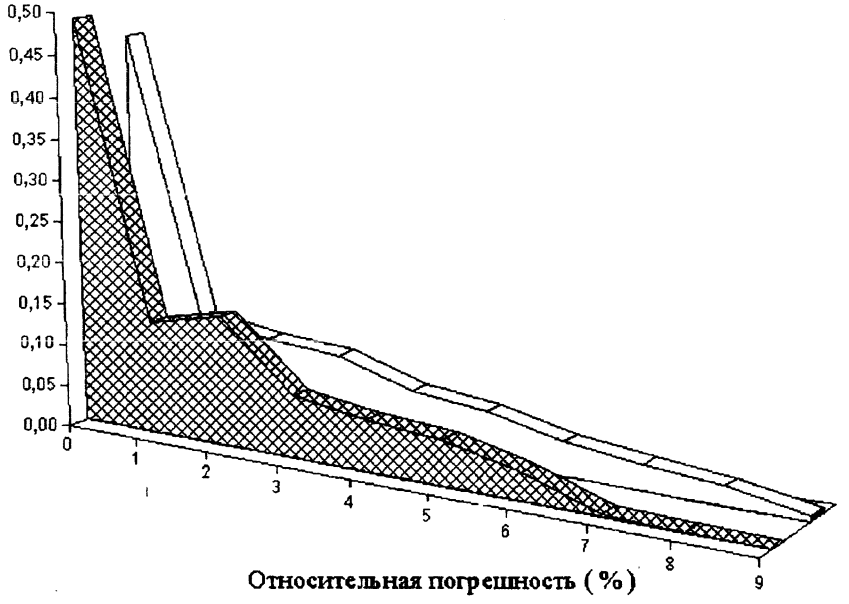
Алгоритм СА в среднем ведет себя не хуже алгоритма ЛГ. На классе E_0 он имеет явное преимущество по всем показателям, но на классах R, E уступает алгоритму ЛГ.

На рис. 5 изображены графики распределения вероятностей для относительной погрешности алгоритмов на классах задач E_0 и R_0. Значения в нуле соответствуют доле числа задач, решенных точно. Значения в точке i соответствуют доле числа задач, для которых относительная погрешность оказалась в интервале от $i-1$ до i процентов. Для повышения достоверности результатов каждая из тридцати тестовых задач решалась вероятностными алгоритмами 10 раз, что позволило увеличить выборку до 300. Из рис. 5 видно, что распределения вероятностей не подчиняются нормальному закону. Это обстоятельство затрудняет получение доверительных интервалов для относительной погрешности алгоритмов. Тем не менее, каков бы ни был закон распределения, всегда имеется возможность вычислить доверительные интервалы для вероятностей $Pr(\varepsilon = 0)$ и $Pr(\varepsilon_1 < \varepsilon < \varepsilon_2)$.

Рассмотрим случайную величину $x \in \{0, 1\}$, принимающую значение 1, если алгоритм получил точное решение задачи, и 0 в противном случае. Обозначим через p вероятность того, что $x = 1$, и положим $q = 1 - p$. Пусть $x_1, \ldots, x_N$ — результаты независимых испытаний



а (класс E_0)



б (класс R_0)

Рис. 5. Распределения вероятностей для относительной погрешности алгоритмов СА и ЛГ

алгоритма. Интервал (γ_1, γ_2) называется доверительным интервалом с коэффициентом доверия $1 - \delta$, если

$$Pr(\gamma_1 < p < \gamma_2) = 1 - \delta.$$

Согласно [8] величина $p^* = \sum_i x_i/N$ является эффективной оценкой, которая асимптотически нормальна с параметрами $(p, \sqrt{pq/N})$. Полагая $\gamma_1 = p^* - \lambda(\delta, N)\sqrt{p^*(1-p^*)/N}$ и $\gamma_2 = p^* + \lambda(\delta, N)\sqrt{p^*(1-p^*)/N}$, получаем доверительный интервал для p. При $\delta = 0{,}05$, $N \geqslant 120$ величина $\lambda(\delta, N)$ полагается равной 1,96. Аналогично строятся доверительные интервалы для вероятности $Pr(\varepsilon_1 < \varepsilon < \varepsilon_2)$.

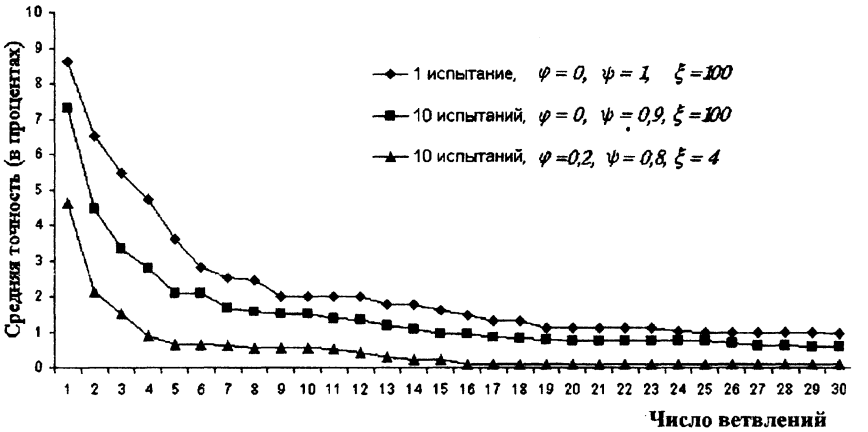
3. Принцип ветвления

В работе [5] исследовалось поведение метода ветвей и границ (МВГ) для задачи (1)–(5). Этот метод позволяет находить точное решение за приемлемое время при размерностях $I \leqslant 50$, $L \leqslant 100$, $J \leqslant 100$. Для алгоритма характерно быстрое нахождение точного решения и затем длительная проверка того, что лучшего решения не существует. Однако с ростом размерности задачи картина меняется. Алгоритм по-прежнему быстро находит хорошее приближенное решение задачи, но предсказать, когда будет найдено точное решение, уже затруднительно. Общая схема МВГ ориентирована на достижение одновременно двух целей: найти оптимальное решение и проверить его оптимальность. Вторая цель является трудно достижимой. Что же касается первой цели, то кратчайший путь к ней может лежать в сочетании идей ветвления и вероятностных эвристик.

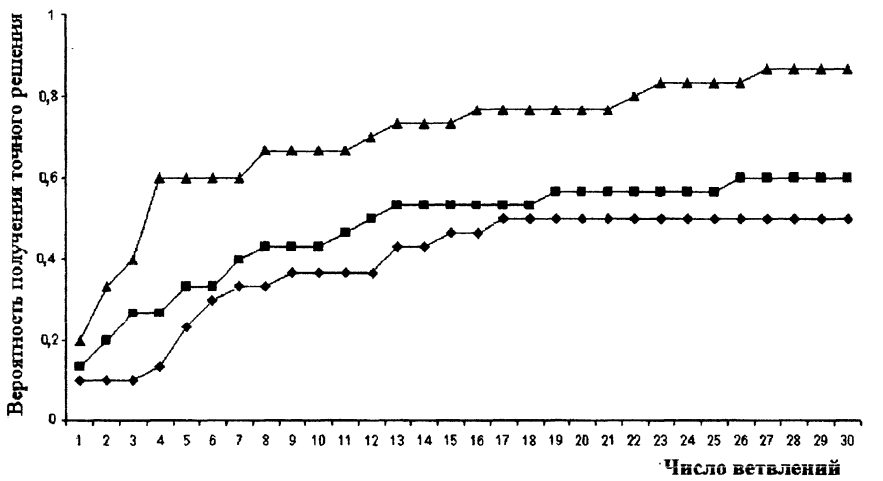
Рассмотрим приближенный полиномиальный алгоритм решения задачи (1)–(5), который выполняет несколько первых итераций МВГ и на каждой итерации использует вероятностные жадные алгоритмы. Впервые такой подход был предложен в [6]. Ключевую роль в этом методе играют схема ветвления, выбор переменной для ветвления и применяемый вероятностный жадный алгоритм.

Пусть $(y_i,\ z_l,\ x_{lj})$ — допустимое решение, полученное на текущей итерации МВГ. Как и прежде, $K = \{i \in I \mid y_i = 1\}$ и $\rho_i = F(K) - F(K \setminus \{i\})$, $i \in K$. В качестве переменной для ветвления будем брать такую переменную y_t, для которой $\rho_t = \min\limits_{i \in K} \rho_i$, и использовать это правило выбора в многосторонней схеме ветвления.

Изменение характеристик алгоритма СА с ростом числа итераций МВГ изображено на рис. 6. Верхняя кривая (рис. 6, *а*) соответствует погрешности детерминированного алгоритма координатного спуска, нижняя — погрешности алгоритма СА при наилучшем выборе параметров,



а



б

Рис. 6. Изменение характеристик алгоритма СА на классе R_0 с ростом числа ветвлений:
а — изменение средней относительной погрешности $M(\varepsilon)$;
б — изменение вероятности получения точного решения $Pr(\varepsilon = 0)$

средняя кривая — погрешности алгоритма СА в случае «чистого» координатного спуска ($I'(\varphi) = \varnothing$). На первых итерациях МВГ погрешность всех алгоритмов резко падает и уже к 10-й итерации сокращается более чем в три раза. После 20-й итерации эффективность ветвления уменьшается и погрешность меняется медленно. На классе R_0 первых 10 ветвлений оказалось достаточно, чтобы значительно сократить погрешность и резко увеличить вероятность получения точного решения (рис. 6, *б*). С ростом числа ветвлений растет и число испытаний алгоритма. Интересно отметить, что если число испытаний зафиксировать, например положить $k = 600$, то средняя погрешность на классе R_0 составит без ветвления 1,16%, а с ветвлением — 0,09%, т. е. станет на порядок меньше.

Таблица 4

Характеристики алгоритмов ЛГ и СА *с ветвлением*

Характеристика	Алгоритм ЛГ				Алгоритм СА			
	R	R_0	E	E_0	R	R_0	E	E_0
$M(\varepsilon)$	0,40	0,16	0,53	1,20	0,27	0,36	0,89	0,92
$D(\varepsilon)$	0,64	0,28	1,36	1,18	1,02	0,13	1,27	1,41
$Pr(\varepsilon = 0)$	77,0	88,0	67,7	55,3	78,0	75,7	43,7	58,7
$Pr(\varepsilon \leqslant 0,01)$	86,6	90,7	86,3	67,3	88,8	84,3	70,3	73,4

В табл. 4 приведены характеристики вероятностных алгоритмов при 10 ветвлениях (21 вершина в дереве ветвления). В каждой вершине вероятностные алгоритмы подвергались 10 испытаниям. Сравнивая эти данные, можно заметить, что разница в результатах незначительная.

Таблица 5

Доверительные интервалы для $Pr(\varepsilon = 0)$

Алгоритм	R	R_0	E	E_0
ЛГ	(72,2; 81,8)	(84,3; 91,7)	(62,4; 73,0)	(49,7; 60,9)
СА	(73,1; 82,7)	(70,9; 80,5)	(38,1; 49,3)	(53,1; 63,3)

Процедура ветвления позволила улучшить все показатели алгоритмов и сгладить их различия. Среднее время работы алгоритмов ЛГ и СА равно 8,5 и 25 с. В табл. 5 приведены доверительные интервалы для вероятности получения точного решения задачи при коэффициенте доверия 0,95. Таким образом, сочетание вероятностных эвристик с ветвлением приводит к лучшим результатам, чем «простое» увеличение числа испытаний.

Заключение

Для многостадийной задачи размещения дано описание вероятностных жадных алгоритмов, основанных на идеях координатного спуска. Проведено экспериментальное исследование этих алгоритмов на классах задач R, R_0, E, E_0. Установлен характер зависимости погрешности получаемых решений от параметров алгоритмов. Показано, что с ростом числа испытаний эта погрешность может быть сделана сколь угодно малой величиной и вероятность не найти оптимальное решение задачи убывает со скоростью геометрической прогрессии. Рассмотрен вариант метода ветвей и границ, в котором для получения верхних границ применяются вероятностные жадные алгоритмы. Такой метод позволяет уже на первых итерациях получать приближенное решения задачи с небольшой относительной погрешностью. Сочетание вероятностных жадных алгоритмов с процедурой ветвления приводит к лучшим результатам, чем простое увеличение числа испытаний.

Вероятностные жадные алгоритмы являются одним из наиболее быстрых и эффективных способов приближенного решения NP-трудных задач дискретной оптимизации. Эти алгоритмы просто устроены, легко адаптируются к изменениям модели. Мы подробно рассмотрели алгоритмы «лидер группы» и «случайный аутсайдер». Эти алгоритмы поразному используют в своей работе элемент рандомизации. Первый алгоритм случайным образом формирует подмножество координат и в нем находит лучший элемент «на добавление». Второй алгоритм просматривает все координаты и среди них случайно выбирает одну «на удаление». Первый алгоритм работает с множеством ассоциаций, второй — с множеством предприятий. Кроме того, второй алгоритм случайным образом формирует начальное множество и время от времени случайно «расширяет» это множество. Применяя данные приемы рандомизации, легко построить другие вероятностные алгоритмы. Например, работа первого алгоритма могла бы начинаться с частичного решения, построенного с помощью условий дополняющей нежесткости. Элемент «на добавление» мог бы выбираться случайно из всего множества координат. Работа второго алгоритма могла бы опираться не на множество предприятий, а на множество ассоциаций, и т. д. Комбинируя эти приемы, можно построить семейство вероятностных жадных алгоритмов и использовать их параллельно для повышения точности решения задачи.

Вероятностные жадные алгоритмы имеют хорошие показатели качества вследствие многократности испытаний. При этом испытания алгоритмов проходили независимо друг от друга. Одним из способов повышения точности решения задачи является организация совместной

работы алгоритмов на основе зависимых испытаний, использующих анализ предшествующей «коллективной» работы [1]. Другое перспективное направление исследований — сочетание вероятностных жадных алгоритмов с итеративными асимптотически точными метаэвристиками: имитацией отжига, поиском с запретами, генетическими алгоритмами [9, 12] и др. Возможно, что именно эти направления позволят быстро и с высокой вероятностью находить оптимальные решения NP-трудных задач.

ЛИТЕРАТУРА

1. **Александров Д. А.** Алгоритм муравьиной колонии для задачи о минимальном покрытии // Методы оптимизации и их приложения: Тр. 11-й Байкальской школы-семинара. 1998. Т. 3. С. 17–20.
2. **Береснев В. Л., Гончаров Е. Н.** Приближенный алгоритм для задачи минимизации полиномов от булевых переменных // Дискрет. анализ и исслед. операций. Сер. 2. 1998. Т. 5, № 2. С. 3–19.
3. **Береснев В. Л., Гимади Э. Х., Дементьев В. Т.** Экстремальные задачи стандартизации. Новосибирск: Наука, 1978.
4. **Гимади Э. Х., Глебов Н. И., Дементьев В. Т.** Об одном методе построения нижней нижней оценки и приближенного решения с апостериорной оценкой точности для задачи стандартизации // Управляемые системы: Сб. науч. тр. Новосибирск: Ин-т математики СО АН СССР. 1974. Вып. 13. С. 26-31.
5. **Гончаров Е. Н.** Метод ветвей и границ для простейшей двухуровневой задачи размещения предприятий // Дискрет. анализ и исслед. операций. Сер. 2. 1998. Т. 5, № 1. С. 19–39.
6. **Гончаров Е. Н., Кочетов Ю. А.** Вероятностный жадный алгоритм с ветвлением для многостадийной задачи размещения // Методы оптимизации и их приложения: Тр. 11-й Байкальской школы-семинара. 1998. Т. 1. С. 121–124.
7. **Гэри М., Джонсон Д.** Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.
8. **Крамер Г.** Математические методы статистики. М.: Мир, 1975.
9. **Aarts E. H. L., Lenstra J. K. (Eds.)** Local Search in Combinatorial Optimization. Chichester: John Wiley & Sons, 1997.
10. **Feige U.** A threshold of $\ln n$ for approximating set cover // Proceedings of the twenty-eighth annual ACM symposium on the theory of computing. N. Y.: ACM Press, 1996. P. 314–318.

11. Feo T. A. Greedy randomized adaptive search procedures // J. Global Optim. 1995. V. 6. P. 109–133.

12. Glover F., Laguna M. Tabu Search. Dordrecht: Kluwer Acad. Publ., 1997.

Адрес авторов:

Институт математики
им. С. Л. Соболева СО РАН,
пр. Академика Коптюга, 4,
630090 Новосибирск,
Россия

Статья поступила
30 октября 1998 г.,
переработанный вариант —
10 марта 1999 г.