

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ДЛЯ ЗАДАЧИ О ПОКРЫТИИ^{*)}

А. В. Еремеев

Изучается известная задача о нахождении совокупности подмножеств минимального суммарного веса, покрывающих данное множество. Предложен новый вариант генетического алгоритма, в котором при выборе подпокрытия из объединения двух родительских покрытий используются методы линейного программирования. Проведен вычислительный эксперимент на эталонных тестовых задачах большой размерности, построенных случайным образом или имеющих комбинаторное содержание. Результаты счета и сравнение с другими подходами подтверждают хорошую работоспособность алгоритма.

Введение

Пусть $A = (a_{ij})$ — произвольная матрица размера $m \times n$ с элементами $a_{ij} \in \{0, 1\}$ без нулевых строк и столбцов. Будем говорить, что в A строка i покрывается столбцом j , если $a_{ij} = 1$. Подмножество столбцов называется *покрытием*, если в совокупности они покрывают все строки матрицы A . Пусть каждому столбцу поставлено в соответствие положительное число c_j , называемое *весом* столбца. Требуется найти покрытие минимального суммарного веса. Вводя переменные x_j , равные 1, если столбец j входит в искомое покрытие, и равные 0 в противном случае, приходим к следующей формулировке задачи о *покрытии*: минимизировать сумму $\sum_{j=1}^n c_j x_j$ при ограничениях

$$\sum_{j=1}^n a_{ij} x_j \geq 1, \quad i = 1, \dots, m, \quad x_j \in \{0, 1\}, \quad j = 1, \dots, n.$$

Известны различные алгоритмы решения этой NP-трудной задачи [3], основанные на методе ветвей и границ, отсечениях, переборе

^{*)} Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (код проекта 97-01-00771) и INTAS (код проекта 96-0820).

L -классов и др. [2, 4, 5, 7, 8, 11]. Время работы таких алгоритмов быстро увеличивается с ростом параметров m и n . Поэтому в последние годы все большее внимание уделяется поиску приближенных решений.

Построение приближенных алгоритмов с априорной оценкой точности получаемых решений хотя и возможно [13], но существование полиномиального алгоритма, имеющего точность менее $(1 - \varepsilon) \ln m$, где $\varepsilon > 0$, представляется проблематичным [17]. В связи с этим более широко разрабатываются алгоритмы, не имеющие теоретических оценок точности, но показывающие хорошие результаты в вычислительных экспериментах. Таковы, в частности, методы лагранжевой релаксации [12], генетические алгоритмы [6, 10], алгоритм муравьиной колонии [1], нейронные сети [19] и др. Большинство из них основано на использовании эмпирических процедур и учете статистики, накапливаемой в процессе решения задачи.

В настоящей статье предлагается новый вариант генетического алгоритма. Его основное отличие заключается в применении линейного программирования при решении вспомогательных задач. Конструкции алгоритма удобнее излагать в терминах множеств. Введем обозначения:

$M = \{1, \dots, m\}$, $N = \{1, \dots, n\}$ — множества номеров строк и столбцов матрицы A ;

$M_j = \{i \in M \mid a_{ij} = 1\}$ — множество строк, покрываемых столбцом $j \in N$;

$N_i = \{j \in N \mid a_{ij} = 1\}$ — множество столбцов, покрывающих строку $i \in M$.

Подмножество столбцов $J \subseteq N$ является покрытием, если $\bigcup_{j \in J} M_j = M$.

Вес покрытия J обозначим через $f(J) = \sum_{j \in J} c_j$. В принятых обозначениях задача состоит в нахождении минимума величины $f(J)$ среди всех покрытий J . Покрытие J называется *тупиковым*, если при любом $j \in J$ множество $J \setminus \{j\}$ не является покрытием. Очевидно, что решение задачи следует искать среди тупиковых покрытий.

1. Общая схема алгоритма

Многие приближенные алгоритмы заключаются в частичном переборе покрытий, генерируемых тем или иным способом. В генетических алгоритмах этот процесс напоминает развитие биологической популяции, что объясняет название алгоритмов и терминологию.

Рассмотрим одну из возможных схем, которая используется и в предлагаемом алгоритме. Сначала выбирается семейство покрытий $P = \{J_1, \dots, J_s\}$ заданного объема s . Множество P называют *популяцией*, а его элементы — *особями*. Из популяции P выбираются две особи,

например J_u и J_v , называемые *родителями*, и из покрытия $J_u \cup J_v$ выделяется покрытие J , называемое *потомком*. Потомок подвергается *мутации* (случайному изменению), после чего он замещает в P наихудшую особь. Этот цикл — выбор родителей, создание потомка, его мутация и обновление популяции — повторяется заданное число раз t_{\max} . Результатом работы алгоритма является наилучшее из покрытий, возникших в ходе развития популяции.

Перейдем к более подробному изложению. С покрытием J свяжем вектор $g = (j_1, \dots, j_m)$ с компонентами $j_i \in N_i \cap J$, $i \in M$. Очевидно, что $J = \bigcup_{i \in M} \{j_i\}$. Такое представление покрытия называют *недвоичным* в отличие от *двоичного* представления в виде вектора $x = (x_1, \dots, x_n)$ с компонентами $x_j = 1$, если $j \in J$, и $x_j = 0$ в противном случае [10]. Элементы j_1, \dots, j_m называют *генами*, а вектор g — *генотипом* особи J . Через (J, g) будем обозначать покрытие и его генотип.

Введем несколько процедур, подробное описание которых дано в следующем пункте. Эти процедуры зададим в виде функций.

Функция $\text{Krs}(J', g', J'', g'')$ вычисляет покрытие $J \subseteq J' \cup J''$ и его генотип g , исходя из родительских покрытий J' , J'' и их генотипов g' , g'' .

Функция $\text{Mut}(g')$ осуществляет *мутацию* генотипа g' , вычисляя новый генотип g путем случайного изменения компонент вектора g' .

Функции $\text{Lmin}(J')$, $\text{Grd}(J')$ и $\text{DGrd}(J')$ разными способами выделяют из покрытия J' тупиковое покрытие J и строят его генотип g .

Будем предполагать, что столбцы исходной матрицы A занумерованы по невозрастанию их весов, т. е. $c_1 \geq \dots \geq c_n$. Пусть α — заданное целое число. Через $N_i(\alpha)$ обозначим совокупность α столбцов наименьшего веса из N_i . Полагаем $N_i(\alpha) = N_i$, если $\alpha \geq |N_i|$. Множества $N_i(\alpha)$, $i \in M$, выделяют «перспективные» столбцы, из которых выбирается покрытие. Предлагаемый алгоритм обозначим через GANP (генетический алгоритм с недвоичным представлением).

Алгоритм GANP

Формирование популяции

Шаг 1. Положить $t = 1$, $k = 1$, $P = \emptyset$.

Шаг 2. Построить вектор $g = (j_1, \dots, j_m)$, выбирая его компоненты j_i случайно и равномерно из $N_i(\alpha)$, $i \in M$. Построить покрытие $J = \bigcup_{i \in M} \{j_i\}$. Вычислить $(J_k, g_k) = \text{Lmin}(J)$ и включить покрытие J_k в популяцию: $P = P \cup \{J_k\}$.

Шаг 3. Если $k < s$, то положить $k = k + 1$ и вернуться к шагу 2.

Шаг 4. Найти начальное рекордное покрытие R из условия

$$f(R) = \min \{f(J_k) \mid k = 1, \dots, s\}.$$

Развитие популяции (итерация алгоритма)

Шаг 5. Выбор родителей. Вычислить показатели *пригодности* особей

$$a_k = \max_{r=1, \dots, s} f(J_r) - f(J_k) + c_n, \quad k = 1, \dots, s,$$

и величины $a = \sum_{k=1}^s a_k$, $b_k = a_k/a$, $k = 1, \dots, s$. В соответствии с распределением вероятностей (b_1, \dots, b_s) выбрать из P родительские покрытия J_u, J_v . Пусть g_u, g_v — их генотипы.

Шаг 6. Создание потомка. Вычислить $(H', h') = \text{Krs}(J_u, g_u, J_v, g_v)$.

Шаг 7. Мутация. Вычислить генотип $h = \text{Mut}(h')$ и соответствующее ему покрытие H .

Шаг 8. Вычислить $(J', g') = \text{Grd}(H)$ и $(J'', g'') = \text{DGrd}(H)$. Пусть J — лучшее из покрытий J' и J'' , т. е. $f(J) = \min\{f(J'), f(J'')\}$, и $g \in \{g', g''\}$ — его генотип.

Шаг 9. Сформировать покрытие W и его генотип w следующим образом. Положить $(W, w) = (J, g)$, если $J \notin P$, и $(W, w) = (H, h)$ в противном случае.

Шаг 10. Обновление популяции. Вычислить $r = \arg \max\{f(J_k) \mid k = 1, \dots, s\}$ и положить $(J_r, g_r) = (W, w)$. Если $f(W) < f(R)$, то положить $R = W$.

Шаг 11. Если $t < t_{\max}$, то положить $t = t + 1$ и вернуться к шагу 5. В противном случае алгоритм заканчивает работу с результатом R .

Конец

Условие на шаге 9 обеспечивает предотвращение преждевременной сходимости процесса к локальным экстремумам. Аналогичная проверка не проводится при формировании популяции, поскольку в этом случае, как показал эксперимент, повторение особей происходит редко.

2. Процедуры алгоритма

Функция $\text{Krs}(J', g', J'', g'')$. Обозначим через J результирующее покрытие и через $g = (j_1, \dots, j_m)$ его генотип. Положим $U = J' \cup J''$ и $S = \{i \in M \mid |N_i \cap U| = 1\}$. Каждая строка $i \in S$ покрывается единственным столбцом $q_i \in U$. Пусть $Q = \{q_i \in U \mid i \in S\}$ — множество таких столбцов и $V = \bigcup_{j \in Q} M_j$ — множество покрываемых ими строк.

При каждом $i \in V$ ген j_i выбирается произвольно из множества $N_i \cap Q$.

Сформируем задачу линейного программирования: минимизировать сумму $\sum_{j \in U \setminus Q} c_j x_j$ при ограничениях

$$\sum_{j \in U \setminus Q} a_{ij} x_j \geq 1, \quad i \in M \setminus V, \quad x_j \geq 0, \quad j \in U \setminus Q.$$

Обозначим через x^* ее решение, полученное, например, двойственным симплекс-методом. Если вектор x^* целочисленный, то на его основе произвольно выбираются остальные гены j_i , $i \in M \setminus V$. В результате получаем покрытие $J = \bigcup_{i \in M} \{j_i\}$. Оно имеет минимальный вес среди всех покрытий столбцами из U . Если вектор x^* содержит дробные компоненты, то полагается $(J, g) = (J', g')$, т. е. функция Kgs возвращает копию одного из родителей.

Решение линейной задачи отменяется, если число $|M \setminus V|$ ограничений в ней превышает порог μ , или если выполнено более ν итераций симплекс-метода. (В вычислительном эксперименте использовались значения $\mu = 150$ и $\nu = 300$.) В этом случае также полагается $(J, g) = (J', g')$.

С помощью дробного вектора x^* можно находить покрытие методом из [7]. В эксперименте это не дало устойчивого повышения качества решений на задачах большой размерности. Вместо описанной функции Kgs можно использовать более простой способ, когда каждый ген потомка копируется из родительских генов с равными вероятностями. Этот подход оказался в эксперименте гораздо хуже предложенного выше.

Функция $\text{Mut}(g')$. Результирующий генотип $g = (j_1, \dots, j_m)$ получается из $g' = (j'_1, \dots, j'_m)$ следующим образом. Каждый ген j'_i с заданной вероятностью мутации p_m заменяется на элемент $j_i \in N_i(\alpha)$, выбираемый в соответствии с распределением вероятностей

$$p_{ij} = \frac{1}{c_j} \bigg/ \sum_{k \in N_i(\alpha)} \frac{1}{c_k}, \quad j \in N_i(\alpha),$$

и с вероятностью $1 - p_m$ ген j'_i не изменяется, т. е. $j_i = j'_i$.

Можно рассматривать и другие варианты мутации. В частности, был опробован подход, в котором мутируемый ген выбирается равномерно из $N_i(\alpha)$, а также вариант мутации с запретом на использование уже имеющихся столбцов. Предложенная функция мутации в ходе экспериментов оказалась лучше других.

Следующие три функции выделяют из данного покрытия J' тупиковое покрытие J и строят его генотип $g = (j_1, \dots, j_m)$.

Функция $\text{Lmin}(J')$

Шаг 1. Положить $J = J'$.

Шаг 2. Перебирая элементы $j \in J$ в порядке их возрастания, выполнить следующие действия: если множество $J \setminus \{j\}$ является покрытием, то положить $J = J \setminus \{j\}$.

Шаг 3. Ген j_i выбрать произвольно из $N_i \cap J$, $i \in M$.

Конец

Функция $\text{Grd}(J')$. Алгоритм предложен в [13]. Он строит покрытие путем последовательного добавления столбцов из J' . Обозначим через U и V текущие множества столбцов и строк.

Шаг 1. Положить $U = \emptyset$ и $V = M$.

Шаг 2. Выделить множество столбцов $K = \{j \in J' \setminus U \mid M_j \cap V \neq \emptyset\}$. (Отметим, что $K \neq \emptyset$.) Вычислить номер $k = \arg \min_{j \in K} \frac{c_j}{|M_j \cap V|}$ и положить $U = U \cup \{k\}$, $V = V \setminus M_k$.

Шаг 3. Если $V \neq \emptyset$, то вернуться к шагу 2. В противном случае вычислить $(J, g) = \text{Lmin}(U)$.

Конец

Функция $\text{DGrd}(J')$. Алгоритм выделяет тупиковое покрытие J и его генотип $g = (j_1, \dots, j_m)$ путем последовательного отбрасывания столбцов из J' . Через U и V обозначим текущие множества столбцов и строк.

Шаг 1. Положить $J = J'$, $U = \emptyset$ и $V = M$.

Шаг 2. Выделить множество строк $S = \{i \in V \mid |N_i \cap J| = 1\}$.

Шаг 3. Если $S \neq \emptyset$, то выполнить следующие действия.

а) Сформировать множество столбцов $Q = \{q_i \in J \mid i \in S\}$, где столбец q_i покрывает строку i .

б) Выбрать ген j_i произвольно из $N_i \cap Q$, $i \in \bigcup_{j \in Q} (M_j \cap V)$.

с) Положить $V = V \setminus \bigcup_{j \in Q} M_j$ и $U = U \cup Q$.

д) Выделить множество столбцов $K = \{j \in J \setminus U \mid M_j \cap V = \emptyset\}$ и положить $J = J \setminus K$.

Если $S = \emptyset$, то вычислить номер $k = \arg \max_{j \in J \setminus U} \frac{c_j}{|M_j \cap V|}$ и положить $J = J \setminus \{k\}$.

Шаг 4. Если $V \neq \emptyset$, то вернуться к шагу 2.

Конец

Как показал эксперимент, отказ от использования функций Grd или DGrd в основном алгоритме приводит к худшим решениям. С другой стороны, применение этих процедур для корректировки особей начальной популяции вызывает слишком быструю сходимость процесса к локальным экстремумам, что затрудняет поиск лучших покрытий. Поэтому в алгоритме GANP при формировании популяции используется функция Lmin , позволяющая получать большее разнообразие особей.

3. Вычислительный эксперимент

3.1. Тестовые задачи

Алгоритм GANP реализован на языке Borland Pascal. Расчеты проводились на компьютере Pentium-100. В эксперименте использовались тестовые задачи о покрытии из электронной библиотеки, описанной в [9].

Каждая задача именуется как *серия.номер*. Рассматривались серии 4–6, A–H, CLR и Stein.

Серии 4–6 и A–H состоят из задач, построенных случайным выбором весов $c_j \in [1, 100]$ и матриц A , в которых доля единиц колеблется от 2% до 20%. Серии CLR и Stein содержат задачи, имеющие комбинаторную постановку. Они описаны ниже в п. 3.4.

На каждой задаче алгоритм GANP испытывался десять раз. Результаты работы алгоритма сравнивались с весами покрытий, наименьшими из известных. Для краткости будем называть эти веса *рекордами*. Рекорды равны оптимумам задач в сериях 4–6 и A–D. Среди серий E–H оптимальность рекордов установлена только для серии F [12].

Общие характеристики серий приведены в табл. 1.

Т а б л и ц а 1
Характеристики серий

Серия	m	n	n'	$\rho, \%$	S	χ
4	200	1000	640–696	2	10	0,97
5	200	2000	628–672	2	10	0,89
6	200	1000	293–315	5	5	0,70
A	300	3000	694–724	2	5	0,86
B	300	3000	312–329	5	5	0,58
C	400	4000	720–737	2	5	0,86
D	400	4000	328–341	5	5	0,65
E	500	5000	171–180	10	5	0,34
F	500	5000	91–96	20	5	0,14
G	10^3	10^4	808–817	2	5	0,81
H	10^3	10^4	353–370	5	5	0,48

Примечание: m, n — размерности задач; $n' = \left| \bigcup_{i \in M} N_i(10) \right|$ — число столбцов при $\alpha = 10$, из которых алгоритм выбирает покрытие; ρ — процент единиц в матрицах задач; S — число задач в серии; χ — доля успешно решенных линейных подзадач при расчете функции Krs.

3.2. Параметры алгоритма

Алгоритм GANP зависит от размера популяции s , величины t_{\max} , уровня α , вероятности мутации p_m и порогов μ, ν при решении линейных подзадач в процедуре Krs. Эти параметры подбирались экспериментально. В результате были взяты значения

$$s = 100, t_{\max} = 10^4, \alpha = 10, p_m = 0,1, \mu = 150, \nu = 300$$

при решении всех задач серий 4–6 и А–Н. В сериях CLR и Stein полагалось $\alpha = n$, а величина p_m подбиралась индивидуально для каждой задачи.

Приведем обоснование выбора некоторых параметров. При $\alpha = 10$ алгоритм GANP нашел рекорды во всех задачах серий 4–6 и А–Н, кроме задач Н.1 и Н.2. Уменьшение параметра α приводит к задачам меньшей размерности и более быстрому счету, но оптимальное решение (например, в задаче 5.3 при $\alpha = 5$) может оказаться недоступным. С другой стороны, при $\alpha = 5$ удается найти рекорд в задаче Н.2, а среднее время получения оптимальных решений на серии 4 уменьшается более чем в три раза.

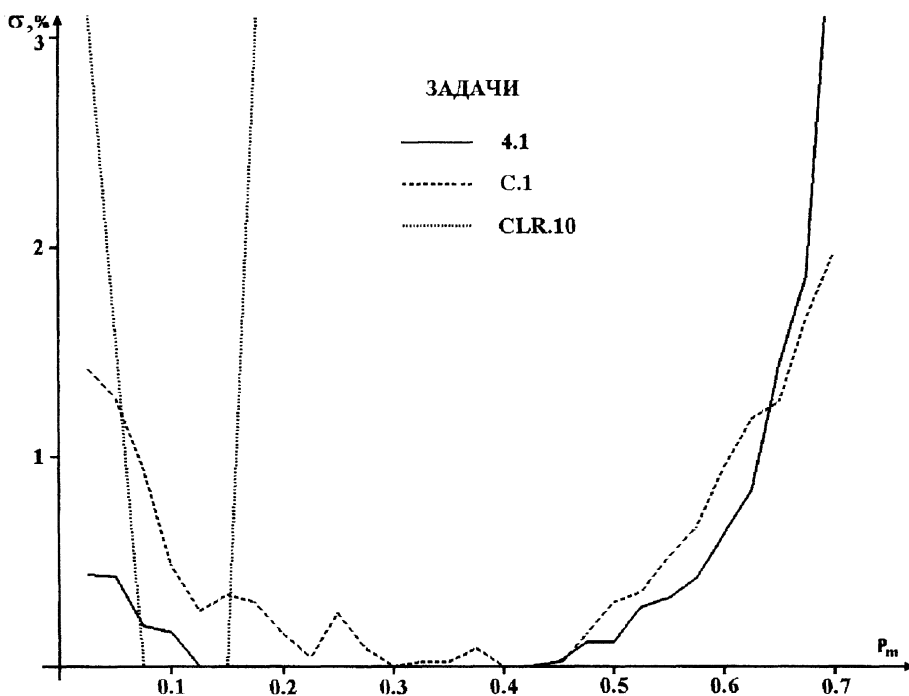


Рис. 1. Зависимость средней погрешности от вероятности мутации

Выбор параметра p_m проиллюстрируем с помощью рис. 1, на котором для задач 4.1, C.1 и CLR.10 изображены графики среднего отклонения

$$\sigma(p_m) = \frac{f - f^*}{f^*} \cdot 100\%.$$

Здесь f^* — рекорд задачи, f — средний вес наилучших покрытий, полученных алгоритмом за 3000 итераций. Видно, что характер зависимости

$\sigma(p_m)$ аналогичен для задач 4.1 и С.1. Эксперименты с другими задачами серий 4–6 и А–Н показали похожие результаты. Отметим, что с ростом вероятности мутации p_m увеличивается средняя длительность шагов алгоритма. Поэтому было выбрано значение $p_m = 0,1$, хотя при $p_m = 0,3$ величина σ заметно меньше.

При решении комбинаторных задач алгоритм GANP оказался более чувствителен к изменениям параметра p_m , что видно на примере задачи CLR.10. Поэтому величина p_m экспериментально подбиралась в каждой задаче серий CLR и Stein.

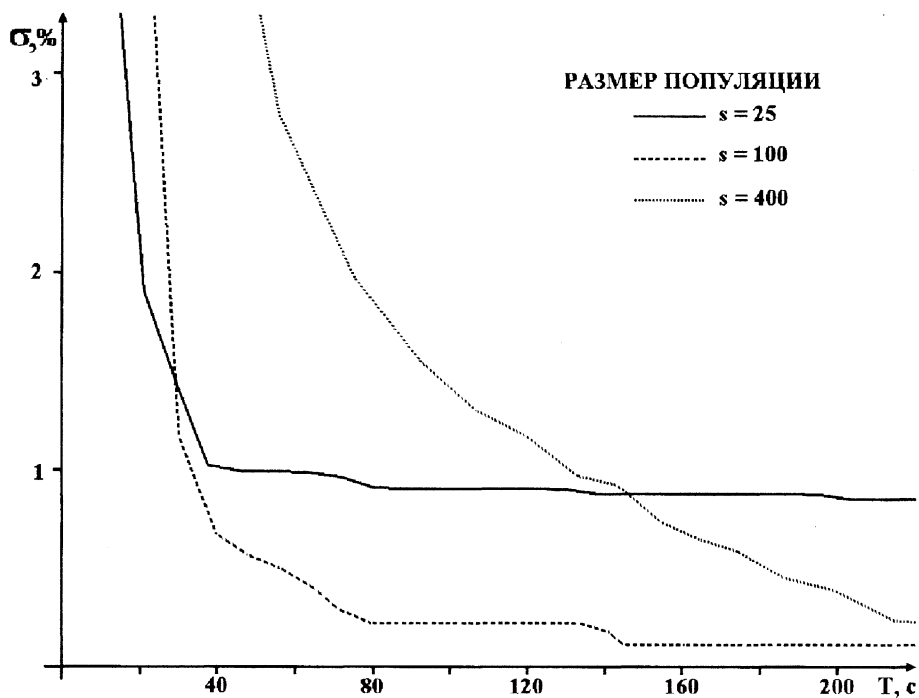


Рис. 2. Зависимость средней погрешности от времени

На рис. 2 изображены графики отклонения σ , усредненного по задачам G.1–G.5, в зависимости от размера популяции s и времени T работы алгоритма.

Как видно, малый размер популяции влечет раннюю сходимость к решениям с относительно высокими значениями целевой функции. При больших s поиск лучших решений значительно замедляется вследствие чрезмерной инертности популяции. На основании проведенных экспериментов был выбран размер популяции $s = 100$ для всех рассматриваемых задач.

3.3. Сравнение с другими алгоритмами

Для сравнения с предлагаемым алгоритмом GANP были выбраны три алгоритма:

BC — генетический алгоритм из [10];

CFT — алгоритм лагранжевой релаксации [12];

BaCa — алгоритм ветвей и границ [7].

В качестве испытательного полигона рассматривались задачи серий 4–6 и А–Н. Во всех задачах, кроме 5.3, G.2, Н.1 и Н.2, алгоритм BC находил рекорд по крайней мере один раз за десять испытаний [10]. Алгоритм CFT нашел рекорды во всех случаях [12].

Результаты счета сведены в табл. 2 и 3, где

T_{ls} — среднее время последнего обновления решения (все времена указаны в секундах работы компьютера Pentium-100);

T_{ex} — среднее время работы алгоритма на одной задаче;

f^* — рекорд задачи;

LP — нижняя оценка оптимума задачи, полученная решением линейной релаксации;

f — наилучшее значение целевой функции, найденное алгоритмом.

Для генетических алгоритмов BC и GANP приведены значения

γ — частоты получения наилучшего из найденных алгоритмом решений при десяти испытаниях;

$\sigma = \sum_{k=1}^{10} \frac{f_k - f^*}{10f^*} \cdot 100\%$ — среднего отклонения получаемых решений от рекорда (f_k — вес наилучшего покрытия, найденного при k -м испытании).

Т а б л и ц а 2

Эксперименты с сериями 4–6 и А–D

Серия	CFT		BC				GANP				BaCa
	T_{ls}	T_{ex}	γ	σ	T_{ls}	T_{ex}	γ	σ	T_{ls}	T_{ex}	T_{ex}
4	2,9	2212,4	0,88	0,07	45,0	315,1	0,86	0,09	33,5	369,2	0,4
5	1,4	2212,4	0,76	0,17	27,2	550,8	0,7	0,27	37,7	318,0	1,0
6	4,1	2212,4	0,94	0,07	25,2	319,1	0,86	0,24	41,0	486,0	7,8
A	47,2	2212,4	0,86	0,20	66,0	787,6	0,44	0,35	71,8	1693,0	27,9
B	3,3	2212,4	1,00	0,00	68,6	938,3	1,00	0,00	20,8	343,9	343,2
C	29,2	2212,4	0,68	0,41	87,9	1141,1	0,74	0,26	52,4	721,3	142,0
D	7,6	2212,4	0,96	0,06	101,7	1372,6	0,94	0,08	23,3	582,7	2051,5

Т а б л и ц а 3

Эксперименты с сериями Е–Н

	f^*	LP	CFT			BC					GANP				
			f	T_{ls}	T_{ex}	f	γ	σ	T_{ls}	T_{ex}	f	γ	σ	T_{ls}	T_{ex}
E.1	29	21,3	29	11,5	2212	29	1,0	0,0	16,9	3014	29	1,0	0,0	1,0	562
E.2	30	22,4	30	180,5	2212	30	0,4	2,0	266,9	3102	30	1,0	0,0	94,8	1768
E.3	27	20,5	27	41,7	2212	27	0,3	2,6	85,1	3110	27	1,0	0,0	23,1	1345
E.4	28	21,4	28	11,6	2212	28	1,0	0,0	238,5	3094	28	1,0	0,0	11,0	1234
E.5	28	21,3	28	16,2	2212	28	1,0	0,0	15,5	3122	28	1,0	0,0	2,1	1067
F.1	14	8,8	14	14,7	2212	14	1,0	0,0	33,8	5026	14	1,0	0,0	7,9	1699
F.2	15	10,0	15	13,8	2212	15	1,0	0,0	34,5	5536	15	1,0	0,0	1,3	1549
F.3	14	9,5	14	110,0	2212	14	1,0	0,0	117,9	5318	14	1,0	0,0	55,4	1765
F.4	14	8,5	14	13,7	2212	14	1,0	0,0	92,6	5710	14	1,0	0,0	20,4	5686
F.5	13	7,8	13	89,0	2212	13	0,3	5,4	67,1	5760	13	0,3	5,4	151,2	8290
G.1	176	159,9	176	65,0	2212	176	0,2	1,0	451,3	3222	176	0,7	0,3	96,0	795
G.2	154	142,1	154	346,6	2212	155	0,5	1,5	159,3	3628	154	0,5	0,7	226,6	837
G.3	166	148,3	166	432,7	2212	166	0,1	1,1	312,1	3152	166	0,1	0,8	319,1	973
G.4	168	148,9	168	167,5	2212	168	0,2	1,4	665,4	3504	168	0,4	0,7	172,5	885
G.5	168	148,2	168	105,0	2212	168	0,2	0,8	242,6	3696	168	0,7	0,2	170,4	952
H.1	63	48,1	63	642,1	2212	64	1,0	1,6	743,0	5383	64	1,0	1,6	90,5	1574
H.2	63	48,7	63	392,5	2212	64	1,0	1,6	234,3	5750	64	1,0	1,6	34,7	9650
H.3	59	45,2	59	690,4	2212	59	0,9	0,2	796,6	5704	59	1,0	0,0	493,2	5620
H.4	58	44,0	58	105,1	2212	58	0,4	1,6	962,9	5535	58	1,0	0,0	218,2	3458
H.5	55	42,4	55	68,8	2212	55	0,9	0,2	198,6	5610	55	1,0	0,0	25,2	1049

Время работы алгоритмов ВаСа, ВС и CFT получено пересчетом данных из [7, 10, 12] по таблице быстроедействий ЭВМ [14]. В табл. 2 приведены величины, усредненные по всем задачам в каждой из серий 4–6 и А–D.

Как видно из табл. 2, алгоритм ВаСа работает быстрее приближенных алгоритмов на задачах малой размерности, но далее время счета резко увеличивается. Для алгоритмов CFT и GANP не наблюдается явной зависимости времени работы и погрешности получаемых решений от размерности задачи. Генетические алгоритмы ВС и GANP имеют сходные результаты и большее время T_{ls} по сравнению с алгоритмом CFT. Анализ табл. 3 показывает, что алгоритм GANP превосходит алгоритм ВС и уступает по точности алгоритму CFT только в двух задачах H.1 и H.2.

3.4. Серии CLR и Stein

Задачи серий CLR и Stein являются комбинаторными и трудно поддаются численным методам [18, 19]. Серия CLR связана со следующей

проблемой раскраски гиперграфов [15]. Требуется найти минимальное число $\varphi(d, r)$ таких r -элементных подмножеств из базового d -элементного множества V , чтобы при любой раскраске элементов из V в два цвета хотя бы одно выбранное r -элементное подмножество оказывалось монохроматичным. Предполагается, что $d \geq 2r - 1$. Нетрудно видеть, что если $d_1 < d_2$, то $\varphi(d_1, r) \geq \varphi(d_2, r)$.

При данных d и r проблема сводится к задаче о покрытии [19], в которой M — совокупность всех раскрасок базового множества, N — множество всех r -элементных подмножеств в V . Вес каждого подмножества равен 1. Подмножество из r элементов *покрывает* раскраску базового множества V , если при данной раскраске это подмножество оказывается монохроматичным. С учетом симметрии половину раскрасок можно опустить, поэтому задача имеет размерности $m = 2^{d-1} - 1$, $n = C_d^r$. Величина $\varphi(d, r)$ равна весу наименьшего покрытия.

Т а б л и ц а 4
Эксперименты с сериями CLR и Stein

Задача	m	n	f^*	f	γ	T_{ls}	p_m	χ
CLR.9	255	126	26	26	1,0	15,32	0,05	0,8
CLR.10	511	210	25	25	1,0	27,0	0,05	0,6
CLR.11	1023	330	23	23	1,0	274,6	0,05	0,6
CLR.12	2047	495	23	23	0,2	979,7	0,02	0,7
CLR.13	4095	715	23	23	0,3	4615,4	0,01	0,6
Stein.27	117	27	18	18	1,0	1,2	0,1	0,01
Stein.45	330	45	30	30	1,0	12,6	0,1	0,03
Stein.81	1080	81	61	61	1,0	16,6	0,02	0,1
Stein.135	3015	135	104	104	0,2	212,4	0,01	0,1
Stein.243	9801	243	202	198	0,6	536,1	0,005	0,4

В вычислительном эксперименте рассматривался случай $r = 4$. В табл. 4 приведены размерности m и n получаемых задач, известные рекорды f^* (их оптимальность не доказана [19]), а также вес f покрытия, найденного алгоритмом GANP. Номера задач в серии CLR равны значениям параметра d . Величины γ , T_{ls} и χ имеют тот же смысл, что и выше.

В табл. 4 приведены также результаты работы алгоритма GANP на задачах серии Stein, имеющих следующее содержание. Совокупность M , состоящая из m трехэлементных подмножеств базового множества N , $|N| = n$, называется *системой троек Штейнера*, если любая пара элементов из N содержится ровно в одной тройке. Элемент из N

покрывает тройку, если он содержится в ней. Требуется найти наименьшее покрытие множества M элементами из N . Размерности m и n задачи должны удовлетворять условиям существования системы троек Штейнера: $m = n(n - 1)/6$, $n \equiv 1, 3 \pmod{6}$. Рекорды f^* первых трех задач серии Stein из табл. 4 равны оптимумам.

Алгоритм GANP нашел рекорды во всех задачах. В задаче Stein.243 полученное решение улучшает современный рекорд [20].

Проведенный вычислительный эксперимент показывает, что GANP может эффективно применяться как метод нахождения начального приближения для точных алгоритмов, а также для быстрого поиска приближенных решений задач большой размерности. В связи с этим представляется интересной комбинация GANP с точными методами (например, по аналогии с подходом, использованным в [16]).

Автор благодарен В. П. Ильеву, А. А. Колоколову и Ю. А. Кочетову за обсуждение полученных в работе результатов и ряд ценных советов.

ЛИТЕРАТУРА

1. Александров Д. А. Алгоритм муравьиной колонии для задачи о минимальном покрытии // Тр. 11-й Байкальской междунар. школы-семинара «Методы оптимизации и их приложения». Иркутск, 1998. Т. 3. С. 17–20.
2. Герман О. В., Ефремов О. В. Алгоритм решения обобщенной задачи о покрытии // Экон. и мат. методы. 1998. Т. 34, № 4. С. 134–140.
3. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.
4. Заозерская Л. А. Об одном алгоритме перебора L -классов для решения задачи о покрытии множества // Тр. 11-й Байкальской междунар. школы-семинара «Методы оптимизации и их приложения». Иркутск, 1998. Т. 1. С. 139–142.
5. Колоколов А. А. Регулярные разбиения и отсеечения в целочисленном программировании // Сиб. журн. исслед. операций. 1994. Т. 1, № 2. С. 18–39.
6. Al-Sultan K., Hussain M., Nizami M. A genetic algorithm for the set covering problem // J. Oper. Res. Soc. 1996. V. 47. P. 702–709.
7. Balas E., Carrera M. C. A dynamic subgradient-based branch and bound procedure for set covering // Oper. Res. 1996. V. 44, N 6. P. 875–890.
8. Balash E., Ho A. Set covering algorithms using cutting planes, heuristics and subgradient optimization: a computational study // Math. Programming Study. 1980. V. 12. P. 37–60.
9. Beasley J. E. OR-library: distributing test problems by electronic mail // J. Oper. Res. Soc. 1990. V. 41, N 11. P. 1069–1072.

10. **Beasley J. E., Chu P. C.** A genetic algorithm for the set covering problem // European J. Oper. Res. 1996. V. 94, N 2. P. 392–404.
11. **Beasley J. E., Jörnsten K.** Enhancing an algorithm for set covering problems // European J. Oper. Res. 1992. V. 58, N 3. P. 293–300.
12. **Caprara A., Fischetti M., Toth P.** Algorithms for the set covering problem / DEIS-Operations Research Group. 1998. Technical Rep. No. OR-98-3.
13. **Chvátal V.** A greedy heuristic for the set covering problem // Math. Oper. Res. 1979. V. 4, N 3. P. 233–235.
14. **Dongarra J. J.** Performance of various computers using standard linear equations software / Computer Science Department, Univ. of Tennessee. 1998. Technical Rep. No. CS-89-85.
15. **Erdős P.** On a combinatorial problem // Nordisk Mat. Tidskrift. 1963. V. 11. P. 5–10.
16. **Eremeev A. V., Kolokolov A. A.** On some genetic and L -class enumeration algorithms in integer programming // Proc. 1st Intern. Conf. on Evolutionary Computation and Its Applications. Moscow, 1996. P. 297–303.
17. **Feige U.** A threshold of $\ln n$ for approximating set cover // Proc. 28th Ann. ACM Symp. on Theory of Comp. ACM, 1996. P. 314–318.
18. **Fulkerson D. R., Nemhauser G. L., Trotter L. E.** Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triple systems // Math. Programming Study. 1974. V. 2. P. 72–81.
19. **Grossman T., Wool A.** Computational experience with approximation algorithms for the set covering problem // European J. Oper. Res. 1997. V. 101, N 1. P. 81–92.
20. **Mannino C., Sassano A.** Solving hard set covering problems // Oper. Res. Letters. 1995. V. 18, N 1. P. 1–5.

Адрес автора:

Омский филиал
Института математики
им. С. Л. Соболева СО РАН,
ул. Певцова, 13,
644099 Омск, Россия.
E-mail:
eremeev@iitam.omsk.net.ru

Статья поступила

17 марта 1999 г.,
переработанный вариант —
2 марта 2000 г.