УДК 519.8

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ДЛЯ ЗАДАЧИ О НАИМЕНЬШЕМ ПОКРЫТИИ МНОЖЕСТВА [1]

А.В.Еремеев

Предлагается новый вариант генетического алгоритма (ГА) для задачи о наименьшем покрытии множества (ЗНП). В рассматриваемом алгоритме используется недвоичное представление решений, которое обеспечивает сохранение допустимости при мутации генотипа, а также новый оператор кроссинговера, основанный на решении задачи оптимальной комбинации генотипов родителей. Операторы локальной оптимизации реализуются алгоритмами градиентного типа. Приводятся результаты вычислительных экспериментов, позволяющие сравнить предложенный алгоритм с другими известными эвристиками для ЗНП. В частности, для всех 50 тестовых задач со случайными данными и известным оптимумом при числе переменных от 1000 до 5000 и от 200 до 500 ограничений рассматриваемым ГА найдены оптимальные решения. Для трех случаев классических комбинаторных задач получены решения, улучшающие известные ранее оценки оптимума. Проведенные эксперименты показывают, что предложенный ГА может эффективно применяться как метод нахождения начального приближения для точных алгоритмов, а также для быстрого поиска приближенных решений задач большой размерности.

1 Введение

Рассматривается задача наименьшего покрытия (ЗНП) в следующей формулировке. *Пусть даны множество* $M = \{1, ..., m\}$ *и набор его подмножеств* $M_j \subseteq M$, *где* $j \in N = \{1, ..., n\}$. *Подмножество* $J \subseteq N$ *называется покрытием* M, *если* $\bigcup\limits_{j \in J} M_j = M$. *Каждому* M_j *приписана стоимость* $c_j > 0$. *Требуется найти покрытие минимальной суммарной стоимости.*

Часто оказывается предпочтительнее рассматривать ЗНП как задачу целочисленного линейного программирования: найти

$$\min\{cx : Ax \geq e, x \in \{0,1\}^n\},$$

где A - $(m \times n)$-матрица, причем $a_{ij} = 1$, если элемент i покрывается подмножеством M_j (т.е. $i \in M_j$), и $a_{ij} = 0$ в противном случае; e-единичный вектор. Пользуясь этой формулировкой, ЗНП можно рассматривать как задачу оптимального покрытия

[1] Работа выполнена при поддержке РФФИ, проект 97-01-00771 и INTAS, проект 96-0820.

строк матрицы A ее столбцами, где столбцы соответствуют множествам M_j. Далее будем предполагать, что эти множества упорядочены по невозрастанию стоимостей c_j.

Известно, что сформулированная задача относится к числу NP-трудных задач дискретной оптимизации и трудоемкость разработанных точных алгоритмов ее решения экспоненциально растет с ростом числа переменных и ограничений. Для нахождения точного решения ЗНП были предложены алгоритмы, основанные на вычислительных схемах методов ветвей и границ, отсечения, перебора L-классов [3, 4]. Особенно плодотворным оказался гибридный подход, в частности, комбинации метода ветвей и границ с лагранжевой релаксацией [6] и с отсечениями [2, 7, 10] позволили значительно повысить размерность решаемых задач.

Наряду с точными методами для ЗНП разработан ряд приближенных алгоритмов полиномиальной трудоемкости с гарантированными оценками точности (см., например, [12]). В то же время, получение полиномиального алгоритма с гарантированной оценкой точности $(1-\varepsilon)\ln(m)$ при $\varepsilon > 0$ представляется проблематичным (см. [16]).

Для поиска приближенных решений задач большой размерности широкое распространение получили эвристические алгоритмы, в частности, методы лагранжевой релаксации [11], генетические алгоритмы [5, 9], алгоритмы муравьиной колонии [1], нейронные сети [18] и другие. Большинство из них основано на использовании эффективных эвристических процедур с учетом некоторой статистической информации, накопленной в процессе решения задачи.

В настоящей работе для ЗНП предлагается вариант генетического алгоритма (ГА), использующий недвоичное представление решений и новый оператор кроссинговера, основанный на поиске оптимальной комбинации генотипов родителей. В §2 приводится общее описание генетических алгоритмов и определяются используемые термины. В §3 рассматривается ГА с недвоичным представлением и используемые в нем операторы кроссинговера и мутации. Эвристики жадного типа, применяемые для локальной корректировки решений описываются в §4. В §5 приводится описание вычислительных экспериментов на задачах со случайными данными и на сериях задач, имеющих комбинаторную постановку.

2 Генетические алгоритмы

В основе ГА лежит идея моделирования развития популяции живых организмов в процессе эволюции или при искусственной селекции. Пусть оптимизационная задача состоит в максимизации функционала f на области допустимых решений D в пространстве решений X. Поиск решений задачи ведется с помощью популяции особей, причем каждая особь рассматривается как пара, состоящая из *генотипа* g и *фенотипа* $x(g)$, которая представляет в ГА некоторую "пробную" точку пространства X. Генотип g - строка из l символов некоторого алфавита, используемого для кодировки решений из X в алгоритме. Элементы этой строки принято называть *генами.* Функция $x(g)$ отображает генотип g в пространство X, определяя тем самым *представление* решений в ГА.

Процесс поиска идет с учетом значений функции *пригодности* $\Phi(g)$, которая определяет степень "приспособленности" особей, и при $x(g) \in D$ имеет вид $\Phi(g) = \phi(f(x(g)))$, где $\phi : R \to R$ - некоторая монотонная функция, используемая для повышения эффективности поиска. Если $x(g) \notin D$, то Φ может действовать как функция штрафа.

Перенумеровав особи популяции, ее можно представить как вектор из генотипов $G = (g_1, g_2, ..., g_s)$, где s-численность популяции. На практике в ГА достаточно хранить только генотипы особей, т.к. соответствующие им фенотипы всегда могут быть восстановлены.

Генотипы начальной популяции генерируются случайным образом в соответствии с заранее выбранным распределением вероятностей. Рассматриваемый здесь ГА (как и алгоритм, предложенный в [9]) реализует *стационарную* схему воспроизводства, при которой на каждой итерации создается одна новая особь. Новый генотип замещает в популяции особь с наименьшим значением функции пригодности и численность популяции остается постоянной. В качестве критерия остановки, как правило, используют ограничение по числу итераций t_{max}. Результатом работы ГА является наилучшее из найденных решений.

Построение генотипа новой особи (потомка) начинается с выбора пары родительских особей из текущей популяции G с помощью вероятностного оператора *селекции*, отдающего предпочтение более пригодным особям. Затем к выбранным генотипам применяется оператор *кроссинговер*, который заменяет часть генов одного генотипа генами другого. При этом, как правило, выбор замещаемых генов производится с элементом случайности. Далее полученная строка подвергается мутации, когда некоторое подмножество генов заменяется случайным образом. В алгоритме, предлагаемом в данной работе, мутируемые гены выбираются по схеме Бернулли с фиксированной вероятностью мутации p_m.

Для применения описанной схемы ГА при решении конкретной задачи необходимо определить способ представления решения в виде генотипа, функцию пригодности и операторы селекции, мутации и кроссинговера. Большое влияние на эффективность алгоритма оказывают настраиваемые параметры, такие как вероятность мутации p_m, численность популяции s и другие. Чаще всего эти величины подбираются эмпирически для конкретного класса задач. Во многих успешных реализациях ГА генотип перед добавлением в популяцию обрабатывается с помощью какой-либо эвристики локальной корректировки, или с помощью оператора "исправления", делающего решение допустимым, если это необходимо.

Достаточно эффективный вариант ГА для ЗНП был предложен в [9]. Этот алгоритм использует двоичное представление решений в виде строк из нулей и единиц длины n. Столбец j при этом входит в покрытие $x(g)$, если j-й бит в g равен 1. Сложность использования данного представления связана с возможностью получения недопустимых решений после мутации и кроссинговера. Для восстановления допустимости в [9] применяется оператор "исправления", представляющий собой эвристику жадного типа. Предлагаемый в данной работе ГА использует недвоичное представление решений, благодаря которому обеспечивается допустимость фенотипа $x(g)$ при любом генотипе g.

3 Общая схема ГА с недвоичным представлением решений

Обозначим через N_i множество столбцов, покрывающих строку i. В ГА с недвоичным представлением (GANP) генотип g состоит из m генов $g^{(1)}, g^{(2)}, ..., g^{(m)}$, причем $g^{(i)} \in N_i$, где $i = 1, 2, ..., m$, т.е. каждый ген кодирует один из столбцов, покрывающих соответствующую строку. Генотипу g сопоставляется вектор покрытия $x(g)$, где $x_i(g) = 1$ при условии, что столбец j представлен хотя бы в одном из генов g. В таком случае нет необходимости в операторе "исправления", однако для эффективной работы ГА требуется оператор локальной корректировки решений, исключающий после кроссинговера и мутации некоторые "лишние" столбцы. С этой целью в предложенном алгоритме используются жадные эвристики, находящие приближенное решение подзадачи наименьшего покрытия строк матрицы A посредством столбцов из $J(g) = \{j | x_j(g) = 1\}$, где g-полученный после мутации генотип. Модифицированная особь добавляется в популяцию при условии, что там отсутствуют особи с тем же фенотипом, что и у нее. Генотипы начальной популяции порождаются независимо, и каждый ген $g^{(i)}$ равномерно распределен на множестве "перспективных" столбцов $N_i(\alpha) \subseteq N_i$. Здесь $N_i(\alpha)$ вводится по аналогии с подходом, изложенным в [9], и содержит α столбцов из N_i с наименьшей суммарной стоимостью если $|N_i| \geq \alpha$; в противном случае $N_i(\alpha) = N_i$. Параметр α фиксирован и подбирается эмпирически с учетом свойств решаемой задачи.

Функцию пригодности $\Phi_t(g)$ на итерации t предлагается задавать следующим образом:

$$\Phi_t(g) = cx(g_{l(t)}) - cx(g) + c_n,$$

где $l(t)$ - номер особи, имеющей наибольшую стоимость покрытия в популяции на данной итерации.

Отбор родителей реализован по принципу *пропорциональной* селекции, при которой вероятность выбора особи с генотипом g_k $(k = 1, 2, ..., s)$ из популяции G на шаге t равна $\Phi_t(g_k) / \sum_{i=1}^{s} \Phi_t(g_i)$.

Операторы кроссинговера, мутации и процедуры локальной корректировки решений Lmin, Grd и DGrd, используемые в предлагаемом алгоритме, будут описаны после общей схемы GANP.

Алгоритм GANP

Шаг 1. Пока не заполнена начальная популяция выполнять:
1.1. Построить случайным образом генотип g.
1.2. Применить эвристику Lmin для исключения "лишних" столбцов из g.
1.3. Полученную особь добавить в популяцию.

Шаг 2. Для t от 1 до t_{max} выполнять:
2.1. Выбрать генотипы родителей g_u, g_v с помощью пропорциональной селекции.
2.2. Получить генотип g с помощью кроссинговера из g_u и g_v.
2.3. Применить к g оператор мутации.
2.4. Подать g на вход эвристик Grd и DGrd. Пусть g' - лучший по пригодности генотип из полученных эвристиками Grd и DGrd.
2.5. Если в G нет особей с фенотипом $x(g')$, то

заменить генотип особи с наименьшей пригодностью в G на g'.

2.5.1. Иначе заменить генотип особи с наименьшей пригодностью генотипом g, полученным на шаге 2.3.

Условие на шаге 2.5 предназначено для предотвращения преждевременной сходимости популяции к локальным оптимумам. Эта проверка отсутствует при добавлении особей на шагах 1.3 и 2.5.1, ввиду того что, как показал эксперимент, в этих случаях повторение уже имеющихся в популяции решений происходит редко.

3.1 Кроссинговер и мутация

В данной работе рассматриваются два варианта оператора кроссинговера.

Равномерный кроссинговер строит генотип потомка, в котором каждый ген копируется из соответствующей позиции генотипа одного или другого родителя с равными вероятностями.

ЛП-кроссинговер предлагается специально для ЗНП. Действие этого оператора основано на решении задачи поиска оптимальной комбинации генотипов родителей, которая решается с использованием линейного программирования.

Для подробного описания данного оператора сформулируем задачу оптимального кроссинговера (ЗОК), как подзадачу исходной ЗНП. Она состоит в отыскании наименьшего покрытия множества M посредством покрывающих подмножеств с индексами из $N' = J(g_u) \cup J(g_v)$: найти

$$\min_{K \subseteq N'} \sum_{j \in K} c_j$$

при условии

$$\bigcup_{j \in K} M_j = M.$$

К ЗОК применяется следующая редукция. Обозначим $S = \{i \in M : |N_i \cap N'| = 1\}$. Тогда каждая строка с индексом $i \in S$ в рассматриваемой подзадаче может быть покрыта единственным столбцом $j(i)$. Пусть $Q = \{j : j = j(i), i \in S\}$ - множество фиксированных столбцов. Каждому гену $g^{(i)}$, где $i \in \bigcup_{j \in Q} M_j$ назначается индекс некоторого столбца из Q, покрывающего строку i. Таким образом ЗОК сводится к задаче наименьшего покрытия нефиксированными столбцами всех строк из $M \setminus \bigcup_{j \in Q} M_j$. Обозначим редуцированную задачу через ЗОК'.

Полученная подзадача может решаться любым точным или приближенным методом, разработанным для ЗНП. Наиболее эффективным из опробованных нами вариантов ГА оказался предлагаемый ниже алгоритм, основанный на использовании симплекс-метода.

Рассмотрим линейную релаксацию ЗОК', где условия целочисленности заменены неравенствами $x_j \geq 0$ для всех переменных подзадачи. Данная задача решается с помощью двойственного симплекс-метода. Если полученное решение x' оказывается целочисленным, то на его основе назначаются оставшиеся неопределенными при редукции гены генотипа g, которые соответствуют строкам из $M \setminus \bigcup_{j \in Q} M_j$. В случае, если решение x' содержит дробные компоненты, кроссинговер не выполняется и оператор возвращает в качестве результата копию генотипа одного из родителей.

Необходимо отметить, что в случае дробного решения информация, содержащаяся в оптимальном решении линейной релаксации, может быть использована для приближенного решения ЗОК'. С этой целью нами была опробована эвристика редуцированных стоимостей, успешно использованная авторами работы [6]. Как показал эксперимент, данная эвристика позволяет улучшить эффективность ГА на задачах относительно небольшой размерности, но не дает устойчивого повышения качества решений на задачах с большим числом переменных и ограничений. Далее в этой работе будет рассматриваться ЛП-кроссинговер, не учитывающий информации о дробных решениях.

С целью предотвращения больших вычислительных затрат поиск оптимума линейной задачи отменяется, если число ограничений в ЗОК' превышает порог μ, или если выполнено более установленного числа ν итераций симплекс-метода (в наших экспериментах $\mu = 150$, $\nu = 300$). В этих случаях оператор также возвращает копию генотипа одного из родителей.

В проведенных вычислительных экспериментах ЛП-кроссинговер оказался значительно более эффективным по сравнению с равномерным кроссинговером. Кроме того, эксперименты показали, что ГА с равномерным кроссинговером при решении некоторых серий тестовых задач получает решения с бо́льшим отклонением от оптимума в среднем, чем алгоритм без какого-либо кроссинговера вообще. В связи с этим далее в данной работе будет рассматривается только GANP с использованием ЛП-кроссинговера.

Оператор мутации организован по схеме аналогичной пропорциональной селекции. Он применяется к каждому гену с заданной вероятностью мутации p_m. Предположим, мутация происходит в гене $g^{(i)}$. Тогда новое значение гена выбирается из множества индексов $N_i(\alpha)$ и вероятность назначения столбца $j \in N_i(\alpha)$ равна

$$p_i(j) = \frac{\frac{1}{c_j}}{\sum\limits_{k \in N_i(\alpha)} \frac{1}{c_k}}.$$

Легко видеть, что при $\alpha \geq \max\limits_i |N_i|$ и $p_m > 0$ в результате мутации любое наперед заданное покрытие может быть получено с ненулевой вероятностью. Пусть q_t^* - вероятность того, что оптимум не будет найден к шагу t. Тогда при $\alpha \geq \max\limits_i |N_i|$ и $p_m > 0$ для любой задачи существует такое $\varepsilon < 1$, что $q_t^* \leq \varepsilon^t$. Тем не менее, в задачах с большим числом переменных такое ε может оказаться очень близко к единице.

Кроме предложенного оператора рассматривались и другие варианты мутации. В частности, был опробован оператор, в котором мутируемый ген $g^{(i)}$ равномерно распределен на множестве индексов $N_i(\alpha)$, а так же вариант мутации с запретом на использование тех столбцов, которые уже имеются в решении. Предлагаемый оператор мутации в ходе экспериментов показал себя лучше других.

4 Локальная корректировка решений

Для исключения "лишних" столбцов из покрытия, соответствующего генотипу, полученному после мутации, предлагается использовать следующие эвристики жадного типа.

Алгоритм поиска лексикографического минимума Lmin - наименее трудоемкая из трех рассматриваемых эвристик. Lmin начинает работу с данного на вход покрытия $J(g)$, и, просматривая его столбцы в порядке возрастания номеров, исключает те из них, удаление которых не нарушает допустимости решения.

Алгоритм Lmin

Шаг 1. Положить $J := J(g)$.

Шаг 2. Для всех j из J в порядке возрастания индексов выполнять:

2.2. Если $J \backslash \{j\}$ - покрытие, то положить $J := J \backslash \{j\}$.

Шаг 3. Для всех $i \in M$ положить $h^{(i)} := j$, где $j \in N_i \cap J$.

Не трудно показать, что вектор покрытия, полученный в результате работы Lmin, будет лексикографически минимальным среди всех допустимых решений задачи покрытия строк матрицы A столбцами из $J(g)$.

Жадный алгоритм Grd - алгоритм подъема (см. [12]), который, начиная с пустого множества, строит покрытие $J \subseteq J(g)$ множества M. Пусть M'- множество еще не покрытых строк. На каждой итерации алгоритма в качестве очередного элемента покрытия выбирается столбец, для которого отношение стоимости к числу покрываемых строк из M' минимально. Полученное таким образом покрытие J может не быть минимальным по включению. В связи с этим к покрытию J применяется алгоритм Lmin, обеспечивающий минимальность по включению. Результатом работы Grd является модифицированный генотип h.

Алгоритм Grd

Шаг 1. Положить $M' := M$, $J := \emptyset$.

Шаг 2. Пока $M' \neq \emptyset$ выполнять:

2.1. Выбрать $k \in J(g) \backslash J$ такой, что $\frac{c_k}{|M_k \cap M'|} \leq \frac{c_j}{|M_j \cap M'|}$ для любого $j \in J(g) \backslash J$.

2.2. Положить $J := J \cup \{k\}$, $M' := M' \backslash (M_k \cap M')$.

Шаг 3. Применить к J эвристику Lmin.

Шаг 4. Для всех $i \in M$ положить $h^{(i)} := j$, где $j \in N_i \cap J$.

Как показано в [12], данный алгоритм имеет оценку точности $1 + \ln m$. Легко видеть, что его трудоемкость не превышает $O(q_g \cdot |J|)$, где $q_g = \sum_{j \in J(g)} |M_j|$.

Двойственный жадный алгоритм DGrd представляет собой вариант жадного алгоритма спуска с адаптивным пересчетом оценок столбцов. Пусть M' обозначает множество нефиксированных строк, J'-множество оставшихся столбцов. В начале работы алгоритма множество фиксированных строк пусто. На каждой итерации из дальнейшего рассмотрения исключается один столбец. Пусть строка с индексом i называется "критической", если она может быть покрыта только одним столбцом $k(i)$ из J'. В случае, если среди нефиксированных строк имеется "критическая" строка с номером i, всем генам, соответствующим нефиксированным строкам, покрываемым столбцом $k(i)$, присваивается индекс $k(i)$. После этого данный столбец исключается, а нефиксированные строки, покрытые им, становятся фиксированными. Если же "критических" строк не оказалось, исключается тот столбец из J', для которого максимально отношение $\frac{c_j}{|M_j \cap M'|}$.

Алгоритм DGrd

Шаг 1. Положить $N'_i := N_i \cap J(g)$ для всех $i = 1, ..., m$; $M' := M, J' := J(g)$.

Шаг 2. Пока $M' \neq \emptyset$ выполнять

2.1. Если есть такие $i \in M'$, что $|N'_i| = 1$, то
Взяв $k \in N'_i$, для всех $j \in M_k \cap M'$ положить $h^{(j)} := k$.
Положить $M' := M' \backslash M_k$.

2.2. Иначе выбрать $k \in J'$ такой, что $\frac{c_k}{|M_k \cap M'|} \geq \frac{c_j}{|M_j \cap M'|}$ для всех $j \in J'$, и
Для всех $j \in M_k \cap M'$ положить $N'_j := N'_j \backslash \{k\}$.

2.3. Положить $J' := J' \backslash \{k\}$.

В результате работы DGrd получен новый генотип h, представляющий некоторое минимальное по включению покрытие. Трудоемкость данного алгоритма не превышает $O(q_g \cdot |J(g)|)$.

Нетривиальных оценок точности предложенного алгоритма в общем случае нам не известно. В частном случае, когда ЗНП на входе эвристики представляет собой задачу вершинного покрытия некоторого графа, т.е. $|N_i| = 2$ для $i = 1, ..., m$ и $c_j = 1$ для всех $j \in J(g)$, алгоритм DGrd действует аналогично жадному алгоритму Эрдеша для задачи о наибольшем устойчивом множестве вершин графа (см., например, [19]). Пусть $\delta = \sum_{j \in J(g)} \frac{|M_j|}{|J(g)|}$ - среднее число единиц в столбце, а $J^*(g)$ - оптимальное решение рассматриваемой подзадачи. Тогда [19]

$$\frac{|J(g)| - |J(h)|}{|J(g)| - |J^*(g)|} \geq \frac{2}{\delta + 2}.$$

При экспериментальном сравнении процедур корректировки выяснилось, что отказ от использования Grd или DGrd приводит к понижению эффективности GANP (в смысле качества полученных решений) для многих тестовых задач. Кроме того, при решении отдельных задач использование Grd или DGrd для корректировки особей начальной популяции вызывает слишком быструю сходимость популяции к локально оптимальным решениям, что в дальнейшем затрудняет поиск решений с лучшими значениями целевой функции. В связи с этим при построении начальной популяции в GANP используется алгоритм Lmin, позволяющий получить большее разнообразие особей на этом этапе.

5 Вычислительный эксперимент

5.1 Тестовые задачи

GANP с ЛП-кроссинговером был запрограммирован на языке Borland Pascal и тестировался на ПК Pentium-100. В экспериментах использовались серии тестовых задач **4-6**, **A-H**, **CLR** и **Stein** из электронной библиотеки OR-library [8].

Серии **4-6** и **A-H** состоят из ЗНП, построенных случайным образом со стоимостями $c_j \in [1, 100]$ и матрицами, доля единиц в которых составляет от 2% до 20 %. Серии **CLR** и **Stein** содержат задачи с единичными стоимостями, имеющие комбинаторную постановку: серия **CLR** связана с одним известным вопросом Эрдеша [14], а **Stein** состоит из задач о покрытии систем троек Штейнера [17] (описание этих задач будет приведено в пп. 5.4).

Таблица 1: Параметры задач

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	1000	10000	808-817	2	5	0.81
H	1000	10000	353-370	5	5	0.48

В последующих разделах обсуждается выбор некоторых настраиваемых параметров ГА, приводятся результаты экспериментов, в которых GANP сравнивается с двумя другими эвристиками, а также результаты работы данного алгоритма на задачах серий **CLR** и **Stein**, для которых в трех случаях получены решения, улучшающие известные ранее оценки оптимума.

С целью снижения вычислительных затрат до начала работы ГА размерность исходной задачи понижается путем исключения из матрицы A тех столбцов, которые заведомо не могут войти ни в одно решение, генерируемое ГА при данном параметре α. Для этого из исходной задачи выделяется "ядро", состоящее из столбцов с индексами из $\cup_{i=1}^{m} N_i(\alpha)$

В проведенных экспериментах при параметре $\alpha = 10$ за 10 запусков ГА нашел оптимум хотя бы один раз для всех задач серий **4-6, A-D,F**, а для всех задач из **E,G** и **H** кроме **H.1** и **H.2** были найдены наилучшие известные решения. Уменьшение параметра α приводит к задачам меньшей размерности, которые, как правило, решаются быстрее и с большей точностью, однако оптимальное решение исходной ЗНП в некоторых тестовых задачах бывает потеряно. Например, при $\alpha = 5$ оптимальное решение задачи 5.3 становится недопустимым после такой редукции. С другой стороны, при $\alpha = 5$ удается найти наилучшее известное решение для задачи **H.2**, а среднее время достижения оптимума на серии **4** уменьшается более чем в 3 раза. При решении задач с единичными стоимостями редукция не применяется и параметр α полагается равным n.

В табл.1 приведены исходные размерности задач, число столбцов n' после редукции с параметром $\alpha = 10$, а также процент единиц ρ в матрицах тестовых задач. В строке S указано число задач в серии, а в строке χ - доля успешно решенных подзадач линейной релаксации при выполнении ЛП-кроссинговера. В проведенных экспериментах частота срабатывания ЛП-кроссинговера значительно повышалась после нескольких сотен итераций ГА, когда генотипы родителей становились более близкими и размерность ЗОК' становилась меньше.

5.2 Вероятность мутации и размер популяции

Для настройки параметров ГА были проведены эксперименты при различных значениях p_m и s. На рис.1 для задач **4.1, C.1** и **CLR.10** изображены графики среднего отклонения от оптимума σ (в процентах от оптимального значения целевой функции) после выполнения 3000 итераций в зависимости от p_m. Из рис.1 видно, что характер этой зависимости для **4.1** и **C.1** аналогичен. Эксперименты с другими задачами серий **4-6, A-H** показали сходную зависимость. Необходимо отметить, что с ростом вероятности мутации средняя трудоемкость каждой итерации ГА растет. С учетом этого свойства была выбрана вероятность мутации равная 0.1, несмотря на то, что при $p_m = 0.3$ среднее отклонение от оптимума после 3000 итераций существенно меньше.

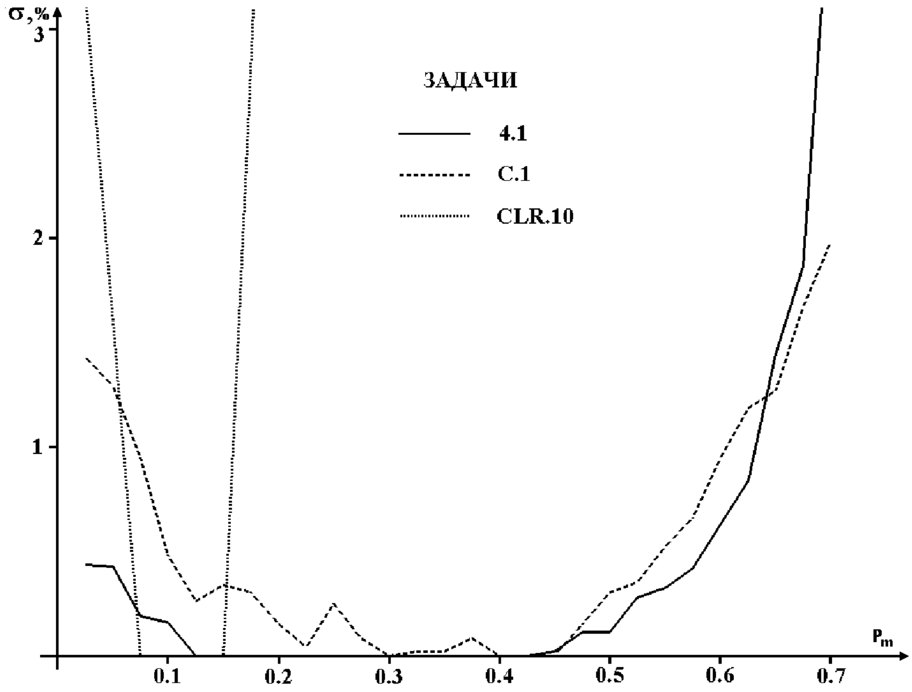

Рис.1. Зависимость средней погрешности GANP от вероятности мутации

При решении задач серии **CLR** алгоритм GANP оказался более чувствителен к параметру p_m, что и видно на примере **CLR 10**. Вероятно, это вызвано особенностями структуры этих задач, которые требуют дальнейшего исследования. В связи с большой чувствительностью этой серии задач к значению p_m, данная величина экспериментально подбиралась для каждой задачи из серии **CLR**.

Для серии **G** приведем графики величины σ (усредненной по 5 задачам) в зависимости от времени работы алгоритма при различных размерах популяции (рис.2).

Как видно, слишком малый размер популяции влечет раннюю сходимость ГА к решениям с относительно высоким значением целевой функции. С другой стороны, при слишком больших s сходимость алгоритма значительно замедляется из-за чрезмерной "инертности" популяции. На основании проведенных экспериментов был выбран размер популяции $s = 100$ для всех рассматриваемых далее задач.

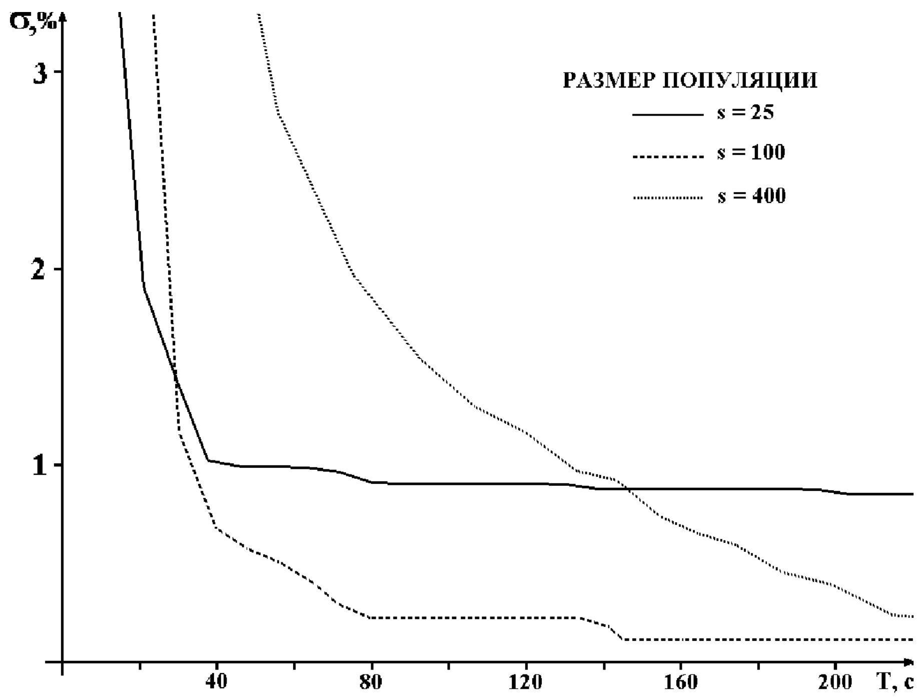

Рис.2. Зависимость средней погрешности GANP от времени работы

5.3 Сравнение с другими алгоритмами на задачах со случайными данными

В табл.2 и 3 GANP сравнивается с ГА из [9] (обозначен BC) и с алгоритмами лагранжевой релаксации [11] (обозначен CFT), а также с методом ветвей и границ (BaCa)[6]. В наших экспериментах проводилось 10 испытаний ГА для каждой задачи при параметрах $s = 100$, $p_m = 0.1$, $t_{max} = 10000$. Столбец σ содержит процент среднего отклонения от оптимума или наилучшего известного решения, т.е. $\sigma = \sum_{k=1}^{10} \frac{z_k - z^*}{10z^*} \cdot 100\%$, где z_k- стоимость решения, найденного при k-м испытании, а z^* - стоимость оптимального или наилучшего известного решения.

Пусть f_{bst} - частота получения наилучшего из найденных ГА решений при 10 испытаниях. Обозначим через T_{sol} среднее время последнего обновления решения; T_{ex} - среднее полное время работы алгоритма при 10000 итераций (в секундах ПК Pentium-100). Время для алгоритмов BC, CFT и BaCa получено из данных, приведенных в [9],[11] и [6] с использованием таблицы быстродействия ЭВМ [13]. В табл. 2 указанные величины усреднены по всем задачам для каждой серии **4-6** и **A-D**.

В соответствии с данными из [9] при 10 запусках алгоритм BC находил хотя бы раз оптимальное решение для всех задач **4-6** и **A-D** кроме задачи **5.3**. Алгоритм CFT для каждой задачи запускался однократно, при этом во всех 45 случаях был найден оптимум [11].

Для сравнения рассматриваемых эвристик с точными алгоритмами в последнем столбце табл.2 приведено среднее время решения задач с помощью одного из лучших известных нам алгоритмов ветвей и границ [6] для ЗНП. Как видно из таблицы, при малых размерностях задач, точный алгоритм имеет преимущество перед эвристиками, но при увеличении числа переменных в 4 раза, а ограничений - в 2 раза, время решения этим алгоритмом возросло более чем в 4000 раз. С другой стороны,

Таблица 2: Эксперименты с сериями 4-6 и A-D

Серия	CFT		BC				GANP				BaCa
	T_{sol}	T_{ex}	f_{bst}	σ	T_{sol}	T_{ex}	f_{bst}	σ	T_{sol}	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

Таблица 3: Эксперименты с сериями E-H

	F'	LP	CFT			BC					GANP				
			F	T_{sol}	T_{ex}	F	f_{bst}	σ	T_{sol}	T_{ex}	F	f_{bst}	σ	T_{sol}	T_{ex}
E1	29	21.3	29	11.5	2212	29	1.0	0.0	16.9	3014	29	1.0	0.0	1.0	562
E2	30	22.4	30	180.5	2212	30	0.4	2.0	266.9	3102	30	1.0	0.0	94.8	1768
E3	27	20.5	27	41.7	2212	27	0.3	2.6	85.1	3110	27	1.0	0.0	23.1	1345
E4	28	21.4	28	11.6	2212	28	1.0	0.0	238.5	3094	28	1.0	0.0	11.0	1234
E5	28	21.3	28	16.2	2212	28	1.0	0.0	15.5	3122	28	1.0	0.0	2.1	1067
F1	14*	8.8	14	14.7	2212	14	1.0	0.0	33.8	5026	14	1.0	0.0	7.9	1699
F2	15*	10.0	15	13.8	2212	15	1.0	0.0	34.5	5536	15	1.0	0.0	1.3	1549
F3	14*	9.5	14	110.0	2212	14	1.0	0.0	117.9	5318	14	1.0	0.0	55.4	1765
F4	14*	8.5	14	13.7	2212	14	1.0	0.0	92.6	5710	14	1.0	0.0	20.4	5686
F5	13*	7.8	13	89.0	2212	13	0.3	5.4	67.1	5760	13	0.3	5.4	151.2	8290
G1	176	159.9	176	65.0	2212	176	0.2	1.0	451.3	3222	176	0.7	0.3	96.0	795
G2	154	142.1	154	346.6	2212	155	0.5	1.5	159.3	3628	154	0.5	0.65	226.6	837
G3	166	148.3	166	432.7	2212	166	0.1	1.1	312.1	3152	166	0.1	0.8	319.1	973
G4	168	148.9	168	167.5	2212	168	0.2	1.4	665.4	3504	168	0.4	0.7	172.5	885
G5	168	148.2	168	105.0	2212	168	0.2	0.8	242.6	3696	168	0.7	0.2	170.4	952
H1	63	48.1	63	642.1	2212	64	1.0	1.6	743.0	5383	64	1.0	1.6	90.5	1574
H2	63	48.7	63	392.5	2212	64	1.0	1.6	234.3	5750	64	1.0	1.6	34.7	9650
H3	59	45.2	59	690.4	2212	59	0.9	0.2	796.6	5704	59	1.0	0.0	493.2	5620
H4	58	44.0	58	105.1	2212	58	0.4	1.6	962.9	5535	58	1.0	0.0	218.2	3458
H5	55	42.4	55	68.8	2212	55	0.9	0.2	198.6	5610	55	1.0	0.0	25.2	1049

для CFT и GANP явной зависимости погрешности или времени поиска решения от размерности задач этих серий не наблюдается.

В табл. 3 приведена более подробная информация по экспериментам с задачами серий **E-H**. Здесь F' - стоимость наилучшего известного решения ; LP - оценка оптимума снизу, полученная решением линейной релаксации ЗНП; F-наилучшее значение целевой функции, найденное эвристикой. В [11] установлено, что наилучшие известные решения для задач серии **F** являются оптимальными (эти решения помечены символом '*' в столбце F'); об оптимальности решений для серий **E,G,H** нам не известно.

На основании проведенного вычислительного эксперимента можно отметить, что при указанных параметрах оба рассмотренных ГА показали близкие результаты на сериях **4-6** и **A-D** и бо́льшее время поиска оптимальных решений по сравнению с CFT. На серии **E** алгоритм GANP превосходит два других алгоритма; на части задач серий **F** и **H** он уступает CFT по времени решения или по качеству найденных решений, а на части задач более эффективен по сравнению с ним. В целом по качеству решений GANP уступает алгоритму CFT только на двух из 65 рассмотренных

Таблица 4: Эксперименты с сериями CLR и Stein

Задача	m	n	F'	GANP	f_{bst}	T_{sol}	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	26	**23**	0.2	979.7	0.02	0.7
CLR.13	4095	715	26	**23**	0.3	4615.4	0.01	0.6
Stein.27	117	27	18^*	18	1	1.2	0.1	0.01
Stein.45	330	45	30^*	30	1	12.6	0.1	0.03
Stein.81	1080	81	61^*	61	1	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

случайных задач. К сожалению, у нас нет информации о том, применялись ли алгоритмы BC и CFT к задачам с единичными стоимостями, которые будут рассмотрены ниже.

5.4 Эксперимент с задачами, имеющими комбинаторную постановку

Примеры серии **CLR** связаны с известной задачей П.Эрдеша о раскраске гиперграфа [14]. *Найти $m_d(r)$, равное минимальному числу сочетаний по r элементов, которые достаточно выбрать из базового d-элементного множества V, чтобы при любой раскраске в 2 цвета элементов V, хотя бы одно из выбранных r-элементных сочетаний оказывалось монохроматическим. Предполагается, что $d \geq 2r-1$.* При фиксированных r и d эта задача может быть сведена к ЗНП [18], в которой M- множество всевозможных раскрасок базового множества, N-множество всех сочетаний по r элементов из V. Полагаем, что сочетание покрывает те раскраски, при которых оно монохроматично. С учетом симметрии половину раскрасок можно не рассматривать. Тогда ЗНП имеет размерность $n = \binom{d}{r}, m = 2^{d-1}-1$, и $m_d(4)$ совпадает с мощностью оптимального покрытия. Здесь рассматривается случай $r = 4$. Даже для этого случая оптимальные решения задачи не известны (см. [18]). В табл. 4 приведены размерности получаемых ЗНП, наилучшие известные из литературы верхние оценки оптимума (столбец F'), а также число элементов покрытий, найденных с помощью GANP. Номера задач в серии **CLR** соответствуют значениям параметра d. Столбцы f_{bst} и T_{sol} имеют тот же смысл, что и в табл. 3. Как и в экспериментах, описанных выше, $s = 100$, $t_{max} = 10000$. Частота срабатывания ЛП-кроссинговера χ и вероятность мутации p_m при решении этих задач также приведены в табл. 4.

Для случаев $d = 12$ и $d = 13$ известные оценки улучшены с помощью GANP, а для $d = 9, 10, 11$ - найдены решения с такими же значениями целевой функции, что и известные ранее (см., например [18]).

Решения из двадцати трех покрывающих элементов могут быть получены для произвольного $d > 11$ и без использования ГА. Достаточно заметить, что если набор сочетаний $\{S_1, ..., S_k\}$ (где $|S_i| = r, i = 1, ..., k$) удовлетворяет условиям задачи при $d = d'$, то этот же набор удовлетворяет условиям и при любом $d > d'$.

Табл.4 содержит также результаты работы GANP на задачах серии **Stein**. Как было отмечено в [17], эти задачи представляют большую трудность для решения с помощью точных алгоритмов. Система троек Штейнера состоит из $m = \frac{n(n-1)}{6}$ трехэлементных подмножеств базового n-элементного множества, где $n \equiv 1, 3(mod\ 6)$, и любая пара элементов базового множества содержится ровно в одной тройке Штейнера. Задача состоит в поиске наименьшего покрытия всех троек элементами базового множества (предполагается, что каждый элемент покрывает все тройки системы, содержащие его). В экспериментах с этими примерами найдены оптимальные решения для всех задач, в которых известен оптимум (соответствующие решения помечены символом '*' в столбце F'), а для задачи **Stein.243** найдены решения с меньшей стоимостью, чем лучшее решение, известное нам из публикаций (см. [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 [15]).

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

Список литературы

[1] **Александров А.Д.** Алгоритм муравьиной колонии для задачи о минимальном покрытии // Труды 11-й Байкальской международной 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. N 4. С. 134-140.

[3] **Заозерская Л.А.** Об одном алгоритме перебора L-классов для решения задачи о покрытии множества // Труды 11-й Байкальской международной школы-семинара "Методы оптимизации и их приложения", Иркутск, 1998, Т. 1. С. 139-142.

[4] **Колоколов А.А.** Регулярные разбиения и отсечения в целочисленном программировании // Сиб. журн. исслед. операций, 1994. Т. 1, N 2. С. 18-39.

[5] **Al-Sultan K., Hussain M., J.Nizami M.** A Genetic Algorithm for the Set Covering Problem // J. Oper. Res. Soc. 1996. V. 47. P. 702-709.

[6] **Balas E., Carrera M.C.** A Dynamic Subgradient-Based Branch and Bound Procedure for Set Covering // Oper. Res. 1996. V. 44. P. 875-890.

[7] **Balash E., Ho A.** Set Covering Algorithms Using Cutting Planes, Heuristics and Subgradient Optimization: A Computational Study // Math. Programming. 1980. V. 12. P. 37-60.

[8] **Beasley J.E.** OR-Library: Distributing Test Problems by Electronic Mail // J. Oper. Res. Soc. 1990. V. 41, N 11. P. 1069-1072.

[9] **Beasley J.E., Chu P.C.** A Genetic Algorithm for the Set Covering Problem // European J. Oper. Res. 1996. V. 94, N 2. P. 394-404.

[10] **Beasley J.E., Jörnsten K.** Enhancing an Algorithm for Set Covering Problems // European J. Oper. Res. 1992. V. 58. P.293-300.

[11] **Caprara A., Fischetti M., Toth P.** Algorithms for the Set Covering Problem // DEIS-Operations Research Group. 1998. Technical Report No. OR-98-3.

[12] **Chvátal. V.** A Greedy Heuristic for the Set Covering Problem // Mathematics of Oper. Res. 1979. V. 4, N 3. P. 233-235.

[13] **Dongarra J.J.** Performance of Various Computers Using Standard Linear Equations Software // Computer Science Department, University of Tennessee. 1998. Technical Report No. CS-89-85.

[14] **Erdös P.** On a Combinatorial Problem // Nordisk Mat. Tidskrift. 1963. V. 11. P. 5-10.

[15] **Eremeev A.V., Kolokolov A.A.** On Some Genetic and L-class Enumeration Algorithms in Integer Programming // Proc. 1st International Conference on Evolutionary Computation and Its Applications. Moscow. 1996. P. 297-303.

[16] **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.

[17] **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.

[18] **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.

[19] **Halldorsson M.M., Radhakrishnan J.** Greed is good: Approximating Independent Sets in Sparse and Bounded-Degree Graphs. // Algorithmica. 1997. V. 18. P. 143-163.

[20] **Mannino C., Sassano A.** Solving Hard Set Covering Problems // Oper. Res. Letters. 1995. V. 18. P. 1-5.