Нгуен Минь Ханг

Вычислительный центр им. А. А. Дородницына РАН, Москва

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА ДЛЯ ЗАДАЧИ НАХОЖДЕНИЯ ПОКРЫТИЯ МНОЖЕСТВА*

1. Введение

Задача о покрытии множества (ЗПМ) системой его подмножеств [1–3] является математической моделью для многих важный приложений, таких как составление расписания [4], планирование сервиса, логического анализа данных, упрощение булевских выражений [3] и т. д. ЗПМ относится к классу NP-полных задач [5], поэтому построение алгоритма нахождения оптимального или приближенного решения является весьма сложной задачей. Однако структура представления реальной задачи предоставляет дополнительную информацию, позволяющую решать ЗПМ довольно больших размерностей (несколько сотен строк и несколько миллион столбцов) с результатом, отличающимся от оптимального решения примерно на 1 %, за приемлемой времени счета [3]. Практические методы решения ЗПМ опираются на различные подходы и могут быть распределены по классам таким, как класс алгоритмов, основанных на теории линейного программирования, класс эвристических алгоритмов и класс алгоритмов ветвей и границ. Далее будет дано формальное представление задачи, предложен алгоритм решения ЗПМ, основанный на принципах генетического алгоритма (см., например [6–9]), и экспериментально оценена эффективность предложенного подхода на множестве тестовых задач из электронной библиотеки [10].

2. Постановка задачи о покрытии множества

Задача о покрытии множества системой его подмножеств может быть формально определена следующим образом (см., например [11]).

* Работа выполнена при финансовой поддержке РФФИ (код проекта № 08–01–00619).

Пусть имеется конечное множество $S = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ и система его подмножеств $S_j \subset S$, $j = 1, 2, \ldots, n$, такие что

$$\bigcup_{j=1}^{n} S_j = S.$$

Каждому из подмножеств S_j поставлен в соответствие вес $c_j > 0$, таким образом рассматривается задача о взвешенном покрытии. Требуется найти минимальный по числу подмножеств набор S_j, такой, что каждый элемент множества S принадлежит хотя бы одному из подмножеств этого набора.

Введем матрицу $A = (a_{ij})_{m \times n}$ следующим образом:

$$a_{ij} = \begin{cases} 1, & \text{если } \sigma_i \in S_j, \\ 0, & \text{если } \sigma_i \notin S_j. \end{cases}$$

Предполагается, что каждый элемент σ_i входит хотя бы в одно из подмножеств S_j. Введем булевы переменные x_j, $j = 1, 2, \ldots, n$:

$$x_j = \begin{cases} 1, & \text{если } S_j \text{ входит в покрытие}, \\ 0, & \text{если } S_j \text{ не входит в покрытие}. \end{cases}$$

Тогда задача о покрытии множества состоит в нахождении покрытия с минимальной стоимостью по следующей математической модели:

$$\sum_{j=1}^{n} c_j x_j \to \min \tag{1}$$

на множестве ограничений:

$$\sum_{j=1}^{n} a_{ij} x_j \geqslant 1, \quad i = 1, 2, \ldots, m \tag{2}$$

$$x_j \in \{0, 1\}, \quad j = 1, 2, \ldots, n \tag{3}$$

В следующей части будет изложен подход решения задачи (1)-(3), основанный на принципах генетического алгоритма.

3. Подход, основанный на генетическом алгоритме, для нахождения минимального покрытия

Неформально генетический алгоритм (ГА) может быть представлен следующим образом:

Шаг 1: Выбор начальной популяции.

Шаг 2: Оценка приспособленности каждого индивида в популяции.

Шаг 3: Выбор родительских индивидов по степени приспособленности. Применение генетических операторов (кроссовер или мутации) над родительскими индивидами для получения потомков.

Шаг 4: Оценивание степени приспособленности полученных индивидов-потомков. Заменить некоторых (или всех) индивидов в популяции индивидами-потомки.

Шаг 5: Если достигнуто желаемое решение, то остановиться. Иначе перейти к шагу 3.

Для того, чтобы можно было применить ГА для решения задачи нахождения минимального покрытия, опишем шаги выше представленного алгоритма в терминах ЗПМ. Не теряя общности, можем предположить, что подмножества в ЗПМ упорядочены по мере возрастания стоимости, для подмножеств с одинаковой стоимостью упорядочение будет производиться в соответствии с уменьшением количества элементов, содержащихся в подмножествах.

3.1. Представление задачи и функции приспособленности

Первым этапом формирования ГА является выбор подходящего представления (способа кодирования) для заданной задачи. Для решения данной задачи выбрано двоичное представление. Каждый индивид k представлен хромосомой, являющейся n-мерным вектором x^k, у которого j-й элемент x_j^k принимает значение 1, если подмножество S_j входит в покрытие, и принимает значение 0, если иначе. С таким представлением степень приспособленности f_k индивида x^k может быть рассчитан следующим образом

$$f_k = \sum_{j=1}^{n} c_j x_j^k, \tag{4}$$

где c_j — стоимость подмножества S_j.

3.2. Выбор родительских индивидов

Выбор родительских пар для скрещивания дает возможность индивидам возрождаться. Был предложен ряд методов выбора родительских пар. В данной работе применяется метод пропорционального выбора. Согласно этому методу каждому индивиду будет присвоена вероятность выбора в соответствии с его степенью приспособленности и выбор будет производиться на основе этих вероятностей. Вероятность выбора для каждого индивида рассчитывается по следующей формуле

$$p_k = \frac{\dfrac{1}{f_k}}{\displaystyle\sum_{k=1}^{N} \frac{1}{f_k}}, \tag{5}$$

где f_k — степень приспособленности k-го индивида в популяции и N — размер популяции. Нужно отметить, что ЗПМ является задачей минимизации, поэтому вероятность выбора у каждого индивида обратно пропорциональна его степени приспособленности.

3.3. Оператор скрещивания (кроссовер)

Обычно в генетических алгоритмах используется одноточечный или двухточечный кроссовер. Тогда точки кроссовера, созданные случайным образом, будут делить хромосому на несколько кусков гена. Куски гена родительских пар будут обменены для получения потомков. Оператор кроссовер может быть формально определен следующим образом:

Пусть x^{P_1} и x^{P_2} — родительские хромосомы, представленные в виде векторов $\left(x_1^{P_1}, \ldots, x_n^{P_1}\right)$ и $\left(x_1^{P_2}, \ldots, x_n^{P_2}\right)$. Тогда имея точку кроссовера v, $1 \leqslant v < n$, оператор кроссовер воспроизведет 2 индивида-потомка x^{Ch_1} и x^{Ch_2} таких, что

$$x^{Ch_1} = \left(x_1^{P_1}, \ldots, x_v^{P_1}, x_{v+1}^{P_2}, \ldots, x_n^{P_2}\right),$$
$$x^{Ch_2} = \left(x_1^{P_2}, \ldots, x_v^{P_2}, x_{v+1}^{P_1}, \ldots, x_n^{P_1}\right).$$

Заметим, что такие классические операторы кроссовера не зависят от степени приспособленности родительских индивидов. Поэтому они не могут моделировать доминирующие и рецессивные характеры соответствующих пар генов родителей. Для устранения этого ограничения мы предлагаем новый оператор кроссовер, создающий только 1 потомок от каждого пара родителей и позволяющий потомку унаследовать доминирующие гены от родителей. Если гены в одном месте одинаковы для обеих родительских хромосом, то потомок просто копирует этот ген. Иначе, если соответствующие гены у родителей неодинаковы, то доминирующий ген определяется в зависимости от степени приспособленности индивида и от частоты появления этого гена в популяции. Степень приспособленности индивида отражает «абсолютную доминантность» комплексного отношения между генами, создающими хромосому индивида, над хромосомам других индивидов. Частота появления гена в популяции отражает «относительную доминантность» гена при отдельном рассмотрении. Высокая частота появления гена в популяции подтверждает его важность в создании рассматриваемой популяции. Другими словами частота появления гена в популяции говорит о его «чистокровность» с точки зрения гена. Исходя из этой идеи предлагается следующий новый оператор кроссовера.

Пусть $f_{x^{P_1}}$ и $f_{x^{P_2}}$ есть степени приспособленности родительских индивидов x^{P_1} и x^{P_2} соответственно и определяются по (5), x^{Ch} есть индивид-потомок, полученный в результате применения оператора кроссовера над x^{P_1} и x^{P_2}. Так как гены принимают только значения из набора $\{0, 1\}$, обозначим через $p_0(j)$ частоту появления значения 0 и через $p_1(j)$ частоту появления значения 1 для j-го гена в популяции. Тогда для всех j,

$1 \leqslant j < n$, мы определим x_j^{Ch} по следующему правилу

1. Если $x_j^{P_1} = x_j^{P_2}$, присвоим $x_j^{Ch} = x_j^{P_1} = x_j^{P_2}$. (6)

2. Если $x_j^{P_1} \neq x_j^{P_2}$:

(a) При $x_j^{P_1} = 0$, присвоим $x_j^{Ch} = x_j^{P_1}$, если $f_{x^{P_1}} \cdot p_0(j) \geqslant f_{x^{P_2}} \cdot p_1(j)$.

Иначе присвоим $x_j^{Ch} = x_j^{P_2}$. (7)

(b) При $x_j^{P_1} = 1$, присвоим $x_j^{Ch} = x_j^{P_1}$, если $f_{x^{P_1}} \cdot p_1(j) \geqslant f_{x^{P_2}} \cdot p_0(j)$.

Иначе присвоим $x_j^{Ch} = x_j^{P_2}$. (8)

Заметим, что при $p_0(j) = 1$ (или $p_1(j) = 1$) все индивиды в популяции имеют одинаковое значение j-го гена и это значение гарантированно перейдет по наследству потомкам по (6). В литературе совокупность таких точек называют *схемой* популяции. *Схема* передается из поколения в поколению и делит хромосому на отдельные участки. Эта особенность позволяет применить параллельно генетический алгоритм на этих отдельных участках для увеличения эффективности в отношения времени вычисления алгоритма. *Схема* также гарантирует определенность последующего оператора мутации.

3.4. Переменная частота мутации

Многие авторы предлагают постоянную частоту мутации, равную 1/n. Тогда мутация будет происходить на случайно выбранном гене. Эта частота оказывается слишком мала при сходимости генетического алгоритма, особенно когда алгоритм приближается к локальному экстремуму. В локальных экстремумах индивиды в популяции могут быть рассмотрены как «единокровные». Поэтому для улучшения этой популяции мы должны уменьшить «единокровность» и увеличить «биологическую разнообразию» популяции. Использование переменной частоты мутации позволяет осуществить эту идею. Приведем метод определения переменной частоты мутации.

Для каждого j-го гена, $1 \leqslant j \leqslant n$, рассчитываем соответствующую энтропию

$$H_j = -p_0(j) \log p_0(j) - p_1(j) \log p_1(j), \tag{9}$$

где $p_0(j)$ и $p_1(j)$ — выше определенные частоты появления значения 0 и 1 в j-ом гене в популяции.

Считаем, что чем меньше H_j, тем больше степень «единокровности» индивидов в популяции в контексте j-го гена. Например, если для какого-то j-го гена его значения в паре родительских индивидов одинаковы, тогда после описанного выше оператора кроссовер j-й ген индивида-потомка будет иметь то же значения, что и у родителей. В результате генетический алгоритм может остановиться в некотором локальном экстремуме. Ясно,

что «чистокровность» и «единокровность» с какой-то точки зрения являются понятиями с противоположными эффектами, но не противоречат, а дополняют друг другу. В генетическом алгоритме операторы кроссовер и мутации частично отображают это свойство. Оператор мутации помогает расширить пространство поиска, другими словами, создает «биологическое разнообразие», позволяет алгоритму выйти из локальных экстремумов. В процессе создания популяции операторы кроссовер и мутации используются подходящим образом для балансировки между «чистокровностью» и «биологическим разнообразием».

Если популяция не имеет *схему*, то для всех j, $1 \leqslant j \leqslant n$, $H_j \neq 0$. В случае существования *схемы* мы можем разделить хромосому на отдельные участки, в которых нет *схемы*. Это разбиение гарантирует, что энтропии генов, лежащих в каждой участке, отличны от 0, и оператор мутации будет применен отдельно на каждой участке. Таким образом, не теряя общности, для удобства описания мы всегда можем считать $H_j \neq 0$ для всех j, $1 \leqslant j \leqslant n$. Тогда определим вероятность мутации j-го гена, $1 \leqslant j \leqslant n$, по следующей формуле:

$$p_{\text{mutation}}(j) = \frac{\dfrac{1}{H_j}}{\displaystyle\sum_{j'=1}^{n} \frac{1}{H_{j'}}} \tag{10}$$

Оператор мутации будет применен над индивидами-потомками, созданными оператором кроссовер. Гены, имеющие вероятность мутации больше $1/n$, будут выбраны для мутации по следующему правилу замены битов:

1. Если $p_{\text{mutation}}(j) > \dfrac{1}{n},\ x_j^{Ch} = 1,\ p_1(j) > p_0(j),$ то $x_j^{Ch} = 0.$ (11)

2. Если $p_{\text{mutation}}(j) > \dfrac{1}{n},\ x_j^{Ch} = 0,\ p_0(j) > p_1(j),$ то $x_j^{Ch} = 1.$ (12)

3.5. Восстановление допустимости решения

При применении операторов кроссовер и мутации над индивидами допустимость решения может быть нарушена (т. е. полученный набор не является покрытием множества). Для устранения этого явления предлагается эвристический алгоритм восстановления допустимости решения, одновременно являющийся локальным оптимизационным шагом для увеличения общей эффективности генетического алгоритма.

Когда рожденный индивид не удовлетворяет условиям допустимости, назовем M' — подмножество элементов S, в которых нарушено условие допустимости (т. е. элементы не входят ни в одно из подмножеств

покрытия). Тогда подмножества S_j, не входящие в решения, будут выбраны, чтобы дополнить покрытие, по порядку возрастания следующего отношения

$$\frac{c_j}{\|S_j \cap M'\|}, \tag{13}$$

где c_j — стоимость подмножества S_j и $\|S_j \cap M'\|$ — мощность множества $S_j \cap M'$. Процесс дополнения заканчивается, когда все элементы M' покрыты.

Такой процесс дополнения восстанавливает допустимость решения, но иногда может создать избыток. Подмножество называется *избыточным*, если удаления этого подмножества из решения не нарушает допустимость решения. Следующий эвристический алгоритм позволяет одновременно восстановить допустимость и исключить избыточность решения. Результат алгоритма является локальным экстремумом задачи.

Пусть:

- $K_i = \{j \mid \sigma_i \in S_j\}, \quad i = 1, 2, \ldots, m$;
- R — множество всех j, такие что S_j входит в покрытие;
- M' — множество всех непокрытых элементов σ_i;
- w_i — количество подмножеств в решении, содержащих элемент σ_i, $i = 1, 2, \ldots, m$.

Шаги алгоритма:

Шаг 1: Инициализировать $w_i := \|R \cap K_i\|, \ i = 1, 2, \ldots, m$.

Шаг 2: Инициализировать $M' := \{\sigma_i \mid w_i = 0, \ i = 1, 2, \ldots, m\}$.

Шаг 3: Для каждого элемента σ_i по порядку нахождения в M':

a) найти первое подмножество S_j по порядку возрастания в K_i, минимизирующее (13);

b) дополним S_j в R и присвоим

$$w_i := w_i + 1 \quad \forall\, \sigma_i \in S_j;$$
$$M' := M' \backslash S_j.$$

Шаг 4: Для каждого j по порядку убывания в R, если $w_i > 1$ для всех $\sigma_i \in S_j$, присвоим

$$R := R \backslash j;$$
$$w_i := w_i - 1 \quad \forall\, \sigma_i \in S_j.$$

Шаг 5: R является допустимым решением, не содержащим избыточных подмножеств. Алгоритм закончен.

В выше предложенном алгоритме шаги 1 и 2 определяют непокрытые элементы. Шаги 3 и 4 являются «жадной» эвристикой, на шаге 3

подмножества с низкой стоимостью будут иметь преимущество быть первыми занесенными в решение, а шаг 4 удаляет избыточные подмножества с наибольшей стоимостью (так как в начале предположили упорядочение подмножеств по мере возрастания стоимости).

3.6. Размер и модель смены популяции

Созданные индивиды-потомки будут использоваться для смены старой популяции по правилу стационарной замены (Steady-State Replacement). Индивиды со степенью приспособленности выше среднего являются кандидатами на замену индивидами с меньшей степенью приспособленности по формуле (4).

В процессе смены популяции по правилу стационарной замены мы должны избежать повторного появления индивида в популяции.

Размер популяции N выбирается пользователем. В тестовых задачах был выбран $N = 100$.

Генетический алгоритм для решения задачи нахождения минимального покрытия

Исходя из вышеприведенных рассуждений разработан генетический алгоритм нахождения минимального покрытия со следующими шагами:

Шаг 1: Создание начальной популяции из N случайных индивидов.

Шаг 2: Выбрать 2 индивида x^{P_1} и x^{P_2} из популяции методом турнирного выбора (tournament selection method), используя вероятность выбора, определенную по (5).

Шаг 3: Применить оператор кроссовер (6)–(7)–(8) над индивидами x^{P_1} и x^{P_2} для создания нового индивида x^{Ch}.

Шаг 4: Мутировать x^{Ch} оператором мутации, определенным по (10)-(12).

Шаг 5: Применить алгоритм восстановления допустимости решения на x^{Ch} для получения допустимого и не избыточного индивида $x^{Ch'}$. Если $x^{Ch'}$ совпадает с каким-либо индивидом в популяции, то вернуться к шагу 2. Иначе перейти к следующему шагу.

Шаг 6: Заменить на $x^{Ch'}$ случайно выбранный индивид со степенью приспособленности выше среднего (средней степени приспособленности популяции) из популяции.

Шаг 7: Повторить шаги 2–6 до того момента, когда больше не рождаются новые индивиды в популяции. Тогда решением задачи является индивид с наименьшей степенью приспособленности в популяции.

Таблица 1

Класс задач	Кол-во строк	Кол-во столбцов	Кол-во задач
4	200	1000	10
5	200	2000	10
6	200	1000	5
A	300	3000	5
B	300	3000	5
C	400	4000	5
D	400	4000	5
E	500	5000	5
F	500	5000	5
G	1000	10000	5
H	1000	10000	5

4. Результаты вычисления

Описанный выше алгоритм был реализован на C++, далее будут представлены результаты численных расчетов на компьютере P-IV, 2.0 Ghz., 512 Mb. Для оценки эффективности предложенного алгоритма были использованы тестовые задачи из библиотеки OR Library [10]. Эта библиотека включает в себя 65 задач различной размерности, разделенных на 11 классов с обозначением 4, 5, 6, A, B, C, D, E, F, G и H. Детали задач приведены в табл. 1.

Так как результат применения генетического алгоритма на практике зависит от создания начальной популяции [12], для каждой задачи мы будем экспериментировать с 15 различными случайно выбранными начальными популяциями размерности 80.

Таблица 2

Номер задачи	Оптимальное решение	$F_{\min}$	$F_{\max}$	$F_{\text{сред}}$	Среднее отклонение %
4,1	429	429	432	430	0,14
4,2	512	512	512	512	0,0

Продолжение таблицы 2

Номер задачи	Оптимальное решение	F_{min}	F_{max}	$F_{сред}$	Среднее отклонение %
4,3	516	516	517	516	0,02
4,4	494	494	494	494	0,0
4,5	512	512	512	512	0,0
4,6	560	560	561	560	0,02
4,7	430	430	432	431	0,08
4,8	492	492	494	492	0,08
4,9	641	641	643	642	0,23
4,10	514	514	515	514	0,03
5,1	253	253	255	253	0,17
5,2	302	302	305	304	0,71
5,3	226	226	229	227	0,61
5,4	242	242	245	243	0,56
5,5	211	211	212	212	0,41
5,6	213	213	214	213	0,02
5,7	293	293	294	293	0,03
5,8	288	288	289	289	0,31
5,9	279	279	279	279	0,0
5,10	265	265	266	265	0,02
6,1	138	138	138	138	0,0
6,2	146	146	147	146	0,14
6,3	145	145	146	145	0,03
6,4	131	131	131	131	0,0
6,5	161	161	162	162	0,56
A,1	253	253	254	253	0,06

Окончание таблицы 2

Номер задачи	Оптимальное решение	$F_{\min}$	$F_{\max}$	$F_{\text{сред}}$	Среднее отклонение %
A,2	252	252	253	252	0,01
A,3	232	232	233	233	0,20
A,4	234	234	234	234	0,0
A,5	236	236	237	236	0,01
B,1	69	69	69	69	0,0
B,2	76	76	76	76	0,0
B,3	80	80	81	80	0,01
B,4	79	79	79	79	0,0
B,5	72	72	72	72	0,0
C,1	227	227	229	227	0,07
C,2	219	219	221	220	0,39
C,3	243	243	247	246	1,12
C,4	219	219	220	219	0,11
C,5	215	215	216	215	0,05
D,1	60	60	60	60	0,0
D,2	66	66	66	66	0,0
D,3	72	72	72	72	0,0
D,4	62	62	63	62	0,09
D,5	61	61	61	61	0,0

Результаты вычисления для задач классов 4–6 и A–D, имеющих известные оптимальные решения [13], представлены в табл. 2. В столбцах таблицы введены следующие обозначения:

- $F_{\min}$ — минимальное значение целевой функции (1) среди полученных решений;
- $F_{\max}$ — максимальное значение целевой функции (1) среди полученных решений;

- $F_{\text{сред}}$ — среднее значение целевой функции (1) среди полученных решений;
- Среднее отклонение $= \sum_{i=1}^{15} \dfrac{(F_i - F_{\text{сред}})}{15} \cdot 100\%$.

Таблица 3

Номер задачи	Предполагаемое решение	$F_{\min}$	$F_{\max}$	$F_{\text{сред}}$	Среднее отклонение %
E,1	29	29	29	29	0,0
E,2	30	30	31	30	1,32
E,3	27	27	28	27	1,45
E,4	28	28	28	28	0,0
E,5	28	28	29	28	0,24
F,1	14	14	14	14	0,0
F,2	15	15	15	15	0,0
F,3	14	13	14	14	−2,16
F,4	14	14	14	14	0,0
F,5	14	13	14	14	−2,02
G,1	179	176	179	178	−0,56
G,2	158	155	158	156	−0,71
G,3	169	166	169	168	−0,56
G,4	172	168	172	170	−0,89
G,5	168	168	170	169	0,67
H,1	64	64	64	64	0,0
H,2	64	63	64	64	0,09
H,3	60	59	60	59	−1,42
H,4	59	58	60	59	−0,13
H,5	55	55	56	55	0,11

В отличие от задачи классов 4–6 и A–D, задачи классов E–H являются задачами большой размерности с неизвестными оптимальными решениями. В рамках данной работы для оценки разработанного алгоритма были использованы предполагаемые решения задач из классов E–H, полученные в [14] на основе использования эвристического алгоритма искусственного отжига (Simulated annealing-based heuristic). Результаты расчетов приведены в табл. 3.

Результаты в таблицах показывают, что для задач классов 4–6 и A–D алгоритм дает оптимальное решение хотя бы в одном эксперименте и среднее отклонение не превышает 1,12 %. Для задач классов E–H алгоритм дает лучшее решение хотя бы в одном эксперименте. Среднее время расчетов составляет от несколько секунд для первой группы задач и достигает несколько минут для второй группы.

5. Заключение

В статье изложен эвристический алгоритм решения задачи нахождения минимального покрытия с неоднородной стоимостью. Алгоритм основан на генетическом алгоритме с операторами кроссовер и мутации, моделирующими различных свойств естественной эволюции, такие как «доминантность», «чистокровность» и «биологическое разнообразие». Результаты вычисления показывают, что алгоритм может дать оптимальное решение для задач средних размерностей и качественно лучше решение для задач больших размерностей.

Литература

1. *Ceria S. et al.* Set Covering Problem // Dell'Amico, F. Maffioli and S. Martello (eds.). Annotated Bibliographies in Combinatorial Optimization. Wiley and Sons, 1997. P. 415–428.
2. *Galinier P. and Heztz A.* Solution techniques for the large set covering problem // Discrete applied Mathematics. 2007. Vol. 155. Issue 3.
3. *Caprara A.*, *Fischetti M.*, *Toth P.* Algorithms for the set covering problem // Working Paper, DEIS, University of Bologna. 1998.
4. *Capara A. et al.* Algorithms for Railway Crew Management // Mathematical Programming. 1997. 79. P. 125–141.
5. *Гэри М.*, *Джонсон Д.* Вычислительные машины и труднорешаемые задачи: Пер. с англ. 1982.
6. *Goldberg D. E.* Genetic algorithms in search, optimization, and machine learning // MA.: Addison-Wesley, 1989.
7. *Батищев Д. И.* Генетические алгоритмы решения экстремальных задач: Учеб. пособие / Воронеж. гос. техн. ун-т; Нижегородский гос. ун-т. Воронеж, 1995.
8. *Нгуен М. Х.* Применение генетического алгоритма для решения одной задачи планирования производства // Динамика неоднородных систем. Том 29. Выпуск 11. М.: ЛКИ, 2007. С. 160–167.

9. *Iwamura K.*, *Okaday N.*, *Deguchiz Y.* Recent Advancements of a Genetic Algorithm to Solve the Set Covering Problem. v2004.
10. *Beasley J. E.* OR-Library: distributing test problems by electronic mail // Journal of the Operational Research Society. 1990. 41. P. 1069–1072.
11. *Сигал И. Х.*, *Иванова А. П.* Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы. М.: ФИЗМАТЛИТ, 2007.
12. *Lan G. and DePuy G. W.* On the effectiveness of incorporating randomness and memory into a multi-start metaheuristic with application to the Set Covering Problem // Computers and Industrial Engineering. 2006. Vol. 51. Issue 3.
13. *Beasley J. E. and Jornsten K.* Enhancing an algorithm for set covering problems // European Journal of Operational Research. 1992. 58. P. 293–300.
14. *Jacobs L. W. and Brusco M. J.* A simulated annealing-based heuristic for the set covering problem // Working paper, Operations Management and Infprmation Systems Department, Northern Illinois University, Dekalb, IL 60115, USA. 1993.