ОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Кафедра Прикладной и Вычислительной Математики

Разработка и анализ точности эволюционных алгоритмов

Дипломная работа
студента гр. М–64
Борисовского П.А.

____
(подпись)

Научный руководитель
к.ф.-м.н.
Еремеев А.В.

____
(подпись)

Омск - 2001

Оглавление

1 Введение

Эволюционные алгоритмы (ЭА) – сравнительно новые эвристические методы поиска, которые успешно применяются при решении оптимизационных задач различных типов. В основе ЭА лежат принципы теории эволюции Дарвина, такие как наследственность, изменчивость и селекция (отбор). Отличительной особенностью ЭА является организация многоточечного поиска, который позволяет избегать быстрой сходимости в относительно слабых локально оптимальных точках. Другая привлекательная черта ЭА – широкие возможности комбинирования их с другими эвристиками. Разработанные таким способом гибридные алгоритмы позволяют быстро находить качественные рашения. По сути ЭА представляет собой модель развития популяции особей в дискретные моменты времени. Особи определенным способом соответствуют точкам исходного пространства поиска и характеризуются пригодностью, которая отражает их качество с точки зрения прикладной области и полезность их участия в дальнейшем развитии популяции. Построение новых особей происходит на основе уже имеющихся случайным образом путем комбинирования и изменения их признаков. Ключевым элементом ЭА является механизм отбора особей с лучшей пригодностью и отсеивание малопригодных.

Рассмотрим подробнее схему ЭА. Популяция Π^t в момент времени $t = 0, 1, \ldots$ определяется просто как определенный набор особей (популяции возникающие в промежуточных вычислениях обозначены Π' и Π''). Предполагается, что имеется алгоритм Init для построения начальной популяции. Механизм отбора реализуется операторами PSelect для выбора родительских особей и Survive для набора в новую популяцию потомков, полученных оператором Reproduce. Реализация этих операторов может быть произвольной, предполагается только лишь выполнение условий $\mathrm{PSelect}(\Pi^t) \subset \Pi^t$ и $\mathrm{Survive}(\Pi^t, \Pi'') \subset \Pi^t \cup \Pi''$.

Общая схема ЭА

1. Построить $\Pi_0 = Init$
2. Положить $t = 0$;
3. Пока не $\mathrm{Terminate}(\Pi, t)$ выполнять:
    3.1 Вычислить пригодность всех особей Π^t
    3.2 $\Pi' = \mathrm{PSelect}(\Pi^t)$
    3.3 $\Pi'' = \mathrm{Reproduce}(\Pi')$
    3.4 $\Pi^{t+1} = \mathrm{Survive}(\Pi^t, \Pi'')$
    3.5 Положить $t = t + 1$

4 Ответ : лучшее из найденных решений.

Одними из широко известных типов ЭА являются генетические алгоритмы (ГА) и эвлюционные стратегии (ЭС). ГА были введены Голландом в 1975г. Их вид был наиболее приближен к природной или искусственной селекции. Существенным понятием в ГА является *генотип* – строка символов с закодированной информацией о соответствующем решении. Именно с генотипами работают все операторы ГА. Поэтому в практических задачах в принципе достаточно подобрать подходящую кодировку и применять ГА. Такой подход, однако, является устаревшим, значительно более качественные результаты получаются при использовании операторов, разработанных специально для данной задачи с учетом ее свойств и особенностей.

Исходя из этих принципов и был построен представленный в этой работе ГА для известной задачи о вершинном покрытии графа.

ЭС предложены Реченбергом и чаще всего используются в непрерывной оптимизации. Основное их отличие от ГА в особом операторе селекции. Также специфической особенностью ЭС считается механизм самонастройки, то есть подгонки внутренних параметров алгоритма к условиям конкретной задачи. По-видимому многие разработки, связанные с ЭС могут быть полезны в области дискретной оптимизации.

В разделе 3 рассматриваются модели двух простейших алгоритмов класса ЭС. Надо сказать, что эволюционные алгоритмы оказываются очень сложными для теоретического исследования. Достаточно точные математические модели ЭА удается построить только для упрощенных алгоритмов и задач, которые, как правило, не имеют практической ценности. Это не означает бесперспективности теоретического описания ЭА. Важность построения и исследования моделей объясняется необходимостью сравнивать различные алгоритмы и выбирать наиболее эффективные в зависимости от условий их применения.

В разделе 4 приводятся результаты экспериментального исследования разработанного ГА и его сравнения с другим ГА подобного типа. Кроме того реализован способ визуализации работы ЭА, который может быть полезен при теоретических исследованиях. Также предпринята попытка экспериментально проверить и выяснить рамки применения на практике результатов раздела 3 на примере одной тестовой задачи.

2 Разработка генетического алгоритма для задачи о вершинном покрытии графа.

В этой части рассматриваются некоторые свойства известной задачи о вершинном покрытии, которые могут оказаться полезными при разработке алгоритмов.

2.1 Определения.

Покрытием графа $G = (V, E)$ называется множество $C \subset V$, такое, что $\forall e = (v_1, v_2) \in E$ либо $v_1 \in C$, либо $v_2 \in C$. Пусть на множестве вершин определена весовая функция $w : V \to R$. Задача вершинного покрытия (ЗВП) состоит в нахождении покрытия C^* наименьшего веса. Дополнение к покрытию называется независимым множеством. Вершины из независимого множества попарно несмежны. При переходе к дополнительному графу они становятся попарно смежными, то есть порождают полный подграф (клику). Таким образом, ЗВП эквивалентна задаче нахождения наибольшего независимого множества и задаче о наибольшей клике.

Всякому покрытию можно сопоставить булев вектор $x = (x_1, ..., x_n)$, где $x_i = 1$ тогда и только тогда, когда $v_i \in C$. Вектор x^*, соответствующий наименьшему покрытию, является решением задачи целочисленного линейного программирования

$$\begin{cases} w(x) = \sum_{i=1}^{n} w_i x_i \to \min \\ x_i + x_j \geq 1, \quad (i,j) \in E, \\ x_i \in \{0,1\}, \quad i = \overline{1,n} \end{cases} \tag{1}$$

Задача является NP-трудной даже в случае, когда $w(v_i) = 1$ для всех $i = \overline{1,n}$ (см [4]).

2.2 Метод ЛП-релаксации

ЛП-релаксацией для ЗВП будем называть задачу (1), в которой ограничения целочисленности заменены условиями неотрицательности $x_i \geq 0, \quad i = \overline{1,n}$. Рассмотрим также следующую задачу ЛП:

$$\begin{cases} w'(x', x'') = \sum_{i=1}^{n} w_i (x'_i + x''_i) \to \min \\ x'_i + x''_j \geq 1, \quad x''_i + x'_j \geq 1, \quad (i,j) \in E, \\ x'_i \geq 0, \quad x''_i \geq 0, \qquad i = \overline{1,n} \end{cases} \tag{2}$$

Для каждого допустимого решения ЛП-релаксации x можно определить класс эквивалентности

$$\{(x', x'') : x_i = \frac{1}{2}(x'_i + x''_i), i = \overline{1,n}\} \tag{3}$$

во множестве допустимых решений задачи (2). При этом для всякого (x', x'') из такого класса выполняется $w'(x', x'') = 2w(x)$, откуда следует эквивалентность этих двух задач . Матрица ограничений (2) является транспонированной матрицей инциденций двудольного графа $G' = (V', E')$, построенного следующим образом: $V' = A \cup B$, $A = \{a_1 ... a_n\}$, $B = \{b_1 ... b_n\}$, $E' = \{(a_i, b_j) \quad | (v_i, v_j) \in E\} \cup \{(a_j, b_i) \quad | (v_i, v_j) \in E\}$. Как показано в [4] всегда существует оптимальное целочисленное решение задачи с такой матрицей. Это означает, что ЛП-релаксация решается нахождением оптимального покрытия C'^* в графе G'. Для этого известен полиномиальный алгоритм:

Алгоритм решения ЗВП двудольного графа (ВПДГ).

Шаг 1. Добавить в V' вершины s и t, в E' -дуги вида $(s, a_i), (b_i, t)$ с пропускными способностями $w(v_i), \quad i = \overline{1, n}$.

Шаг 2. Заменить ребра из E' дугами, ориентированными из A в B, и приписать им бесконечные пропускные способности.

Шаг 3. В полученной сети найти минимальный разрез (S, T) и положить $C'^* = (S \cap B) \cup (T \cap A)$.

Отметим, что множество C', полученное указанным на шаге 3 способом из некоторого (произвольного) разреза (S, T) конечной пропускной способности, является покрытием, так как множества $A \cap S$ и $B \cap T$ независимы, а дуг вида (a, b), $a \in A \cap S, \quad b \in B \cap T$ быть не может, иначе они имели бы пропускную способность ∞ и входили бы в разрез. Пропускная способность разреза складывается из пропускных способностей дуг вида $(s, a), \quad a \in A \cap T$ и $(b, t), \quad b \in B \cap S$, то есть равна весу соответствующего покрытия. Это доказывает оптимальность покрытия C'^*, полученного алгоритмом.

Из построения (3) следует, что существует оптимальное решение ЛП-релаксации, в котором $x_i \in \{0, \frac{1}{2}, 1\}$, для всех $i = \overline{1, n}$. Всякому такому решению поставим в соответствие разбиение множества V:

$$P = \{v_i | x_i = 1\}, \quad Q = \{v_i | x_i = \frac{1}{2}\}, \quad R = \{v_i | x_i = 0\} \tag{4}$$

Пусть H – подграф, порожденный Q.

Теорема. Если C_H^* оптимальное покрытие H, то $C_H^* \cup P$ – оптимальное покрытие G.

Доказательство [1] Очевидно, что $P \cup C_H^*$ покрытие G. Предполжим, что оно не является оптимальным и найдем какое-нибудь оптимальное покрытие C. Обозначим $C_P = C \cap P, C_Q = C \cap Q$, $C_R = C \cap R$. Тогда

$$w(C_P) + w(C_R) + w(C_Q) < w(C_H^*) + w(P).$$

Так как C_Q –покрытие H, то $w(C_Q) \geq w(C_H^*)$. Отсюда $w(P) \geq w(C_P) + w(C_R)$, или

$$w(P \setminus C_P) > w(C_R).$$

Положим $P' = C_P, \quad Q' = Q \cup (P \setminus C_P) \cup C_R, \quad R' = R \setminus C_R$. Легко проверить, что решение ЛП-релаксации соответствующее разбиению $\{P', Q', R'\}$ допустимо. Но это противоречит оптимальности решения, соответсвующего разбиению $\{P, Q, R\}$, поскольку

$$w(P') + \frac{1}{2}w(Q') = w(C_P) + \frac{1}{2}(w(Q) + w(P \setminus C_P) + w(C_R)) <$$

$$< w(C_P) + \frac{1}{2}w(Q) + \frac{1}{2}w(P \setminus C_P) + \frac{1}{2}w(P \setminus C_P) =$$

[1] Эта теорема доказана в [19], здесь предлагается более простой вариант доказательства.

$$= w(P) + \frac{1}{2}w(Q). \qquad \blacksquare$$

Таким образом, ЗВП сводится к нахождению оптимального покрытия графа H.

Назовем алгоритм для ЗВП *r-приближенным* , если для любого графа и любого покрытия C, полученного этим алгоритмом, выполняется $w(C) \leq rw(C_*)$. Такое покрытие также будем называть *r-приближенным* решением ЗВП, а число r – оценкой точности алгоритма. Известно, что нахождение приближенного решения ЗВП на произвольном графе с оценкой точности меньше $\frac{7}{6}$ является NP-трудной задачей.

Если C_H - покрытие графа H находится некоторым приближенным алгоритмом то, учитывая, что $w(C^*) \geq w(P) + \frac{1}{2}w(H)$ получаем

$$w(C) = w(C_H) + w(P) \leq w(C_H) + w(P) \leq 2w(C^*).$$

Решение ЛП-релаксации можно рассматривать как шаг предварительной подготовки, в результате которого, возможно, понижается размерность задачи и гарантируется оценка точности не хуже, чем 2. В [15] высказано предположение о том, что, если $P \neq NP$, то не существует полиномиального алгоритма с оценкой точности $2 - \delta$, где $\delta > 0$ не зависит от графа.

2.3 Жадный алгоритм и его рандомизация

Хорошие результаты на практике покпзывает следующий эвристический алгоритм жадного типа: [14, 15]

Алгоритм Grd

Шаг 1. Положить $C := \emptyset$.

Шаг 2. Пока $V \neq \emptyset$ выполнять:

2.1. Выбрать $v \in V$, для которой $\frac{d(v)}{w(v)} = \min_{v \in V} \frac{d(v)}{w(v)}$.

2.2. $V := V \backslash (N(v) \cup \{v\})$.

2.3. $C := C \cup N(v)$.

Лемма 1 *Пусть есть полиномиальный алгоритм, который находит в произвольном графе упаковку S, весом $w(S) \geq \rho w(S^*)$, где S^* – оптимальная упаковка. Тогда существует полиномиальный $(2-\rho)$- приближенный алгоритм для ЗВП.*

Доказательство. Найдем S^*_H упаковку подграфа H со свойством $w(S_H) \geq \rho w(S^*_H)$. Определим $C = P \cup (Q \setminus S_H)$. Тогда $w(C) \leq w(P) + w(Q) - \rho w(S^*_H) = w(P) + w(Q) - \rho(w(Q) - w(C^*_H)) = w(P) + (1-\rho)w(Q) + \rho w(C^*_H) \leq w(P) + 2(1-\rho)w(C^*_H) + \rho w(C^*_H) \leq (2-\rho)w(C^*)$.$\blacksquare$

Обозначим через $\bar{d} = \frac{1}{n}\sum_{k=1}^{n} deg(v_i)$-среднюю степень вершин графа. В [14] показано, что упаковка невзвешенного графа S , полученная жадным алгоритмом, удовлетворяет $|S| \geq \frac{1}{1+\bar{d}})|S^*|$. Тогда, применяя лемму 1 получаем:

Теорема 1 *Жадный алгоритм в сочетании с методом ЛП-релаксации является $2(1 - \frac{5}{4\bar{d}_H+6})$ приближенным алгоритмом решения невзвешенной ЗВП (здесь $\bar{d}_H$ – средняя степень вершин подграфа H).*

На шаге 2.1 алгоритма можно выбирать вершину для удаления случайным образом, отдавая предпочтение вершинам с меньшим значением $d(v)/w(v)$. Это можно реализовать, например, используя схему так называемой s-турнирной селекции, когда выбирается независимо с равномерным распределением s вершин, затем среди них выбирается вершина с наименьшим значением $d(v)/w(v)$. Назовем такой алгоритм жадным рандомизированным.

Алгоритм RGrd

Шаг 1. Положить $C := \emptyset$.

Шаг 2. Пока $V \neq \emptyset$ выполнять:

2.1. Построить $\{v_1, ...v_s\}$– выборку с возвращением из множества V

2.2. Выбрать v_i, для которой $\frac{d(v_i)}{w(v_i)} = \min\{\frac{d(v_k)}{w(v_k)} : k = \overline{1, s}\}$.

2.3. $V := V \backslash (N(v_i) \cup \{v_i\})$.

2.4. $C := C \cup N(v_i)$.

Вероятность того, что RGrd найдет оптимальное решение всегда ненулевая (хотя она может быть очень мала). Разумно запускать этот алгоритм несколько раз и в качестве результата брать лучшее найденное решение. Параметр s подбирается из опыта на тестовых задачах. Результаты экспериментов при $s = 0.7|V|$ приведены в разделе 6. Здесь стоит напомнить, что по ходу работы алгоритма число $|V|$ уменьшается (см. шаг 2.3.), соответственно в данной реализации изменяется значение s.

2.4 Генетические алгоритмы

Генетический алгоритм(ГА) – метод случайного поиска, основанный на механизмах природной эволюции. Пусть задача математического программирования сформулирована в виде:

$$\begin{cases} f(x) \to \min(\max) \\ x \in D \end{cases}$$

где $D \subset X$ – допустимое множество в пространстве X. Поиск решений данной задачи в ГА ведется с помощью популяции особей, причем каждая особь рассматривается как пара, состоящая из *генотипа* g и *фенотипа* $x(g)$, которая соответствует некоторой пробной точке пространства X. Генотип g - строка фиксированной длины, состоящая из l *генов* $g^1, g^2, \ldots, g^l$, где $g^i \in \Xi_i$ - символы некоторых алфавитов $\Xi_1, \Xi_2, ..., \Xi_l$ (чаще всего в ГА используется один алфавит $\{0, 1\}$). Функция $x = x(g)$ отображает генотип g в пространство X, задавая тем самым *представление* решений в ГА. Разграничение генотипа и фенотипа в ГА позволяет исследовать поведение алгоритма независимо от специфики конкретных задач и разрабатывать универсальные схемы алгоритмов. В некоторых случаях применения ГА генотип и фенотип практически совпадают, и разграничение генотипа и фенотипа оказывается формальным.

Для генотипа g определена функция *пригодности* $\Phi(g)$, которая связана со значением целевой функции $f(x(g))$ и характеризует "качество" особи в популяции. Как правило при $x(g) \in D$ имеет вид $\Phi(g) = \phi(f(x(g)))$, где $\phi : \mathbf{R} \to \mathbf{R}$ - некоторая монотонная (в случае задачи максимизации возрастающая, иначе – убывающая) функция, используемая для повышения эффективности поиска. Если $x(g) \notin D$, то Φ может действовать как функция штрафа (см., например, [8]).

Перенумеровав особи популяции на шаге t, ее можно представить в виде *вектора генотипов популяции* $\Pi^{(t)} = (g_1^{(t)}, g_2^{(t)}, ..., g_N^{(t)})$, где N-число особей популяции. На практике при реализации ГА достаточно хранить только генотипы особей, т.к. соответствующие им фенотипы, как правило, могут быть легко восстановлены. Генотипы начальной популяции $\Pi^{(0)}$ генерируются случайным образом в соответствии с некоторым заранее выбранным распределением вероятностей.

Один из наиболее распространенных вариантов организации ГА - *популяционная* схема воспроизводства, когда на каждой итерации ГА все генотипы новой популяции $\Pi^{(t+1)}$ порождаются в соответствии с одним и тем же распределением вероятностей, определяемым текущей популяцией $\Pi^{(t)}$ и распределениями вероятностей операторов *селекции*, *мутации* и *кроссинговера*, которые будут описаны ниже.

Другой способ организации ГА - *стационарная* схема воспроизводства, когда на каждой итерации создается только одна новая особь, замещающая в популяции особь с наименьшим значением функции пригодности, или другую особь, выбранную по некоторому эвристическому правилу. Как показывает опыт использования ГА, стационарная схема воспроизводства во многих случаях оказывается предпочтительнее популяционной с точки зрения среднего времени поиска решений приемлемого качества.

Как правило, в ГА построение генотипа новой особи (потомка) начинается с выбора пары родительских особей из $\Pi^{(t)}$ при помощи вероятностного оператора селекции, который отдает предпочтение более пригодным особям. Выбор одной родительской особи не зависит от выбора другой (и одна особь может быть выбрана в обоих случаях).

Среди наиболее распространенных операторов отбора - операторы *пропорциональной* и *турнирной* селекции. При пропорциональной селекции вероятность $P_{ch}(g_k^{(t)})$ выбора особи $g_k^{(t)}$ из $\Pi^{(t)}$ пропорциональна ее пригодности:

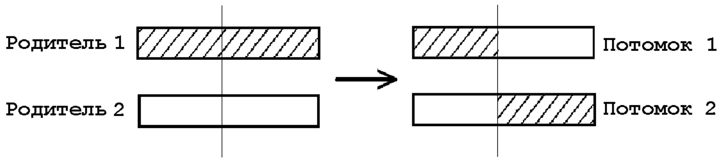
$$P_{ch}(g_k^{(t)}) = \frac{\Phi(g_k^{(t)})}{\sum\limits_{j=1}^{N} \Phi(g_j^{(t)})}.$$

При использовании данного оператора предполагается, что функция пригодности принимает только положительные значения. Основной недостаток пропорциональной селекции состоит в том, что, как правило, после достаточно большого числа итераций все особи популяции получают близкие значения пригодности, и вероятности их выбора становятся почти одинаковыми. К тому же, распределение вероятностей выбора особи сильно зависит от вида функции пригодности.

Оператор турнирной селекции с размером турнира s (или оператор s-турнирной селекции) при построении очередного решения из текущей популяции извлекает s особей с равномерным распределением и выбирает лучшую из них (одна особь может извлекаться до s раз в турнире). Данный способ отбора более экономичен по вычислительным затратам, чем предыдущий. Кроме того, его избирательность не изменяется в процессе работы алгоритма и не зависит от вида функции пригодности (здесь ее можно отождествить с целевой функцией, принимая, что в случае задачи минимизации "лучшая" особь – та, "пригодность" которой меньше). Посредством выбора размера турнира s можно регулировать интенсивность селекции.

К выбранным генотипам с фиксированной вероятностью p_{cross} применяется оператор кроссинговера, который заменяет часть генов одного генотипа генами другого.

Оператор *одноточечного* кроссинговера действует по следующей схеме:



Сначала случайным образом выбирается точка разрыва j от 1 до $l-1$. Каждый родительский генотип делится на две части (первая – с номерами генов не превосходящими j, вторая – оставшаяся). Генотипы потомков получаются присоединением второй части одного из родителей к первой части другого.

Двухточечный кроссинговер выбирает две точки разрыва, и родительские генотипы "обмениваются" той частью, которая находится между этими точками.

При использовании *униформного (равномерного)* кроссинговера обмен присходит каждым геном независимо с вероятностью $\frac{1}{2}$.

В работе [11] применяется так называемый оператор слияния, который, действуя на родительские генотипы g и h, формирует единственного потомка q, назначая для каждого $k=\overline{1,l}$ независимо от остальных $q_k = g_k$ с вероятностью $\frac{\Phi(g)}{\Phi(g)+\Phi(h)}$, и $q_k = h_k$ с вероятностью $\frac{\Phi(h)}{\Phi(g)+\Phi(h)}$.

Существует множество других вариантов рекомбинации, учитывающих структуру задачи или обладающих особыми статистическими свойствами.

Полученный после кроссинговера генотип g подвергается мутации, которая вносит случайные изменения в особь. Это делается для того, чтобы избежать ситуации, когда популяция заполняется похожими генотипами, что сильно ограничивает пространство поиска. Чаще всего в ГА используется оператор мутации, который в соответствии со схемой Бернулли с фиксированной вероятностью мутации p_m изменяет значения генов, подставляя на их место случайные символы соответствующих алфавитов. В случае, когда после мутации $x(g) \notin D$, к нему можно применить оператор "исправления", который разрабатывается с учетом специфики задачи ([11, 3]). Также полезным оказывается разработка способов возможного повышения пригодности потомка.

В качестве критерия остановки ГА используют ограничение по числу итераций, времени работы или достижение оптимума, когда это можно установить. Результатом работы ГА является лучшее из найденных решений.

Для применения описанной схемы алгоритма при решении конкретной задачи необходимо определить способ представления решения в виде генотипа, функцию пригодности, операторы селекции, кроссинговера и мутации. Большое влияние на эффективность алгоритма оказывают настраиваемые параметры, такие как вероятности мутации и кроссинговера, размер селекции, численность популяции. Чаще всего эти величины подбираются эмпирически для рассматриваемого класса задач с известными свойствами и параметрами.

2.5 Генетический алгоритм для задачи о вершинном покрытии

Существует несколько способов кодирования особей в случае ЗВП. Наиболее естественно считать генотипом бинарный вектор $x = (x_1, ..., x_n)$, являющийся решением (планом) задачи ЦЛП (1), а фенотипом – соответствующее ему множество $C \subset V$. При таком подходе обычно приходится бороться с тем, генотип потомка полученного в результате операторов кроссинговера и мутации будет соответствовать недопустимому решению. В [11] для восстановления допустимости разработана жадная эвристика. В [8] предлагается считать особью любой бинарный вектор, а в функцию пригодности добавить штраф за каждое невыполненное ограничение (непокрытое ребро).

Здесь предлагается в качестве особей рассматривать только неизбыточные (минимальные по включению) покрытия. Из любого покрытия C можно получить неизбыточное следующим алгоритмом:

Алгоритм выделения неизбыточного покрытия (ВНП)

Шаг 1. Сгенерировать случайную перестановку $(\pi_1, ..., \pi_n)$.

Шаг 2. Для i=1 до i=n выполнять

$j := \pi_i$

Если $C \setminus \{v_j\}$ является покрытием, то $C := C \setminus \{v_j\}$.

Шаг 3. Получено неизбыточное покрытие C.

Обозначим C_1 и C_2 - родительские особи, C - особь потомка, $P = C_1 \cap C_2$, $Q = (C_1 \setminus C_2) \cup (C_2 \setminus C_1)$ Легко заметить,что все варианты кроссинговера, описанные в предыдущем параграфе, возвращают множество P в объединении с некоторым подмножеством Q, выбранным случайным образом. Оптимизированный кроссинговер (ОК) [5] также включает в новую особь множество P, а подмножество из Q выбирается как оптимальное покрытие подграфа H, порожденного Q. Поскольку Q - есть объединение независимых множеств, то H - двудольный граф, и его оптимальное покрытие C_H^* можно найти за полиномиальное время (например алгоритмом ВПДГ). Множество $C_H^* \cup P$ явлется покрытием, а его минимальность достигается применением алгоритма корректировки. Вариант генетического алгоритма, использующий ОК, предложен в [5] усовершенствован в [10], и является одним из хорошо зарекомендовавших себя эвристических методов решения ЗВП.

ОК всегда находит лучшее в смысле допустимости и значения целевой функции решение, по сравнению с рассмотренными ранее. Однако, очень часто ОК возвращает копию одного из родителей, что снижает эффективность поиска. Особенно это проявляется в задачах для графов с большой плотностью ребер, где множество Q обычно представляет собой незначительную часть вершин, и кроссинговер работает в сравнительно малой части пространства решений.

Пример 1.[4, 8] Рассмотрим граф G, множество вершин V которого состоит из трех непересекающихся частей $V_1 = \{a_i\}_{i=1}^{k}$, $V_2 = \{b_j\}_{j=1}^{k+2}$, $V_3 = \{c_j\}_{j=1}^{k+2}$, множество ребер $E = \{a_i b_j : i = \overline{1,k}, j = \overline{1,k+2}\} \cup \{b_j c_j : j = \overline{1,k+2}\}$. Единственное оптимальное покытие такого графа - V_2. Любое другое неизбыточное покрытие C содержит V_1, остальные вершины входят в него так, что $c_j \in C \Leftrightarrow b_j \notin C$. Если родительские особи не являются оптимальными, то полученный из них подграф H порожден множеством V_1 и некоторым подмножеством V_2. C_H^* - любая из долей H. В таком случае ОК возвращает копию одного из родителей, иначе говоря ОК для такой задачи просто не работает.

Замечание 1. Алгоритм ВНП, примененный ко всему множеству вершин данного графа находит оптимальное покрытие с вероятностью $p \geq \frac{1}{3}$.

Доказательство. Оптимальное покрытие находится, на шаге 1 алоритма в случайной перестановке все вершины из V_2 окажутся правее хотя бы какой нибудь одной вершины из V_1. По формуле полной вероятности

$$p \geq \frac{k}{3k+4} + \frac{k+2}{3k+4}\frac{k}{3k+2} + \frac{k+2}{3k+4}\frac{k+1}{3k+2}\frac{k}{3k} \geq \frac{13k^2+14k}{3(9k^2+18k+8)}.$$

Обозначим через $f(k)$ правую часть второго неравенства. Так как $f(2) > \frac{1}{3}$ и $f'(k) > 0$, то $f(k)$ возрастает, следовательно $f(k) > \frac{1}{3}$ для всех $k \geq 2$ ■.

Замечание 2. Алгоритм из [8] находил оптимальное покрытие данного графа с частотой 65% при $k = 32$, с частотой 60% при $k = 66$.

Чтобы расширить область поиска кроссинговером нового решения предлагается решать вспомогательную подзадачу для нахождения вершинного покрытия графа G с дополнительным условием, что искомое покрытие должно содержаться во множестве вершин $C_1 \cup C_2$ [1, 3]. Рассматриваемая задача сводится к решению ЗВП на графе $G((C_1 \cup C_2)\backslash W(C_1, C_2))$, где $W(C_1, C_2) = \{v \in V : N(v) \nsubseteq (C_1 \cup C_2)\}$ - множество фиксируемых вершин, которые просто необходимо включить в результирующее покрытие. Результатом работы кроссинговера будет найденное покрытие, в объединении с $W(C_1, C_2)$. В общем случае поставленная задача является NP-трудной. Представляется перспективным для ее решения использовать приближенные методы. Назовем варианты кроссинговера, построенные на основе алгоритмов Grd и RGrd, соответственно жадным и рандомизированным жадным.

Замечание 3. Пригодность потомка, полученного жадным кроссинговером, может оказаться хуже пригодности родителей.

Пример 2. Рассмотрим граф G, образованный кликой K размером k, независимыми множествами $S_1, ..., S_n$ размером $m \leq k$ каждое и вершинами $a_1, ..., a_n$. Любая вершина из $S_i, i = \overline{1,n}$ соединяется с a_i и со всеми вершинами из K. Пусть C_1 состоит из $\{a_1, ..., a_t\}, S_{t+1}, ..., S_n$ и K. C_2 - из $\{a_{t+1}, ..., a_n\}, S_1, ..., S_t$ и K, где $t < n$. C_1 и C_2 являются неизбыточными. Легко видеть, что $C_1 \cup C_2 = V$, жадный алгоритм построит покрытие, состоящее из K и всех S_i. Мощность этого покрытия больше, чем мощности C_1 и C_2.

Рандомизированный жадный кроссинговер позволяет получать значительно более широкий диапазон решений.

В основе оператора мутации лежит свойство, что, если C – неизбыточное покрытие, то для любой $v \in V$ существует альтернатива: либо $v \in C$, либо $N(v) \subseteq C$. Выбрав вершину $v \in C$ оператор мутации с вероятностью p_m удаляет ее из C, добавляя вместо нее в C множество $N(v)$. После этого все вершины v и $N(v)$ фиксируются и больше не рассматриваются. С вероятностью $1 - p_m$ фиксируется только вершина v. Процесс продолжается, пока не останется незафиксированных вершин. Допустимость покрытия C все время сохраняется. Если в результате в C окажутся "лишние" вершины, то необходимо применить процедуру ВНП.

Алгоритм MutVCP(g)

Шаг 1. Положить $C := \emptyset, V' := V$.

Шаг 2. Пока $C(g) \cap V' \neq \emptyset$ выполнять:

2.1. Выбрать $v \in C(g) \cap V'$ с равномерным распределением.

2.2 С вероятностью p_m положить

$$C := C \cup N(v),\ V' := V' \backslash N(v).$$

иначе положить $C := C \cup \{v\}$

2.3 $V' := V' \backslash \{v\}$.

Шаг 3. Получен генотип $g = g(C)$.

ГА использующий жадный кроссинговер и указанный оператор мутации описан в [3]. Здесь предлагается вариант ГА на основе рандомизированного жадного кроссинговера.

Новый алгоритм реализован по стационарной схеме воспроизводства. Особи для начальной популяции строятся применением алгоритма ВНП ко всему множеству V. Такой подход позволяет получить любое неизбыточное покрытие с ненулевой вероятностью. Каждый из двух родителей выбирается с помощью 5-турнирной селекции. Одновременно выбирается худшая особь среди попавших в оба турнира, которая впоследствии заменяется потомком. В качестве кроссинговера используется оператор, основанный на алгоритме RGrd. Мутацию оказалось выгоднее осуществлять над одним из родительских генотипов, так как это предотвращает заполнение популяции похожими локально оптимальными решениями и при этом не "портит" хорошие решения, полученные кроссинговером.

Описанный ГА далее обозначается через GA-RGrd. Его общая схема имеет следующий вид.

Алгоритм GA-RGrd

Шаг 1. Пока не заполнена начальная популяция $\Pi^{(0)}$ выполнять:

1.1. Построить с помощью оператора ВНП генотип g.

1.2. Полученную особь добавить в популяцию.

Шаг 2. Для t от 1 до t_{max} выполнять:

2.1. С помощью 5-турнирной селекции выбрать родительские генотипы g, h и генотип w для замены.

2.2. Применить к h оператор мутации.

2.3. Получить генотип g' с помощью рандомизированного жадного кроссинговера из g и h.

2.4. Если в $\Pi^{(t)}$ нет особей с фенотипом $C(g')$, то заменить генотип w на g'.

Результаты экспериментов с GA-RGrd приведены в разделе 4.1.

3 Исследование некоторых эволюционных алгоритмов.

3.1 Эволюционные стратегии

Эволюционные стратегии (ЭС) – класс эвристических алгоритмов случайного поиска, которые подобно ГА имитируют развитие популяции особей. ЭС применяются преимущественно для решения задач непрерывной оптимизации, хотя они вполне применимы и для дискретных задач. Мы будем рассматривать самую общую постановку задачи максимизации целевой функции $f : D \to \mathbf{R}$ на множестве $D \subset \mathbf{R^n}$. Понятие кодировки решения для ЭС не столь важно, как для ГА. Обычно в реализациях генотип, представляющий решение x либо просто равен x, либо содержит еще некоторые дополнительные параметры, которые используются при мутации. Пригодность как правило отождествляется с целевой функцией. Поиск основан на построении новых особей из уже имеющихся с помощью мутации и кроссинговера. Здесь нас будут интересовать ЭС без кроссинговера. Отметим, что традиционно мутация считается наиболее важным инструментом поиска для ЭС. Прежде чем дать описание алгоритмов необходимо определить характерный для них оператор селекции. Пусть популяция представлена вектором генотипов $\Pi = (g_1, \dots, g_\lambda)$. Упорядочим вектор по неубыванию пригодностей особей, то есть $\Phi(g_{i_1}) \geq \Phi(g_{i_2}) \geq \dots \Phi(g_{i_\lambda})$, и для $0 < \mu < \lambda$ определим вектор $s_\mu(g_1, \dots, g_\lambda) = (g_{i_1}, \dots, g_{i_\mu})$. В отличие от вероятностных операторов пропорциональной и турнирной селекции оператор s_μ детерминировано выбирает μ лучших потомков из λ возможных.

Общая схема алгоритмов (μ, λ)-ES и $(\mu + \lambda)$-ES

1. Построить $\Pi^0 = (g_1^0, \dots, g_\mu^0)$
2. Для $t = 0$ до $t_{max} - 1$ выполнять
   2.1 Вычислить $\Phi(g_1^t), \dots, \Phi(g_\mu^t)$
   2.2 Для $i = 1$ до λ выполнять
      2.2.1 u – случайная величина из $\{1, 2, , \dots, \mu\}$ с равномерным распределением
      2.2.1 $g_i' = Mut(g_u^t)$
   2.3 $\Pi^{t+1} = \begin{cases} s_\mu(g_1', \dots, g_\lambda') \text{ для } (\mu, \lambda)\text{-ES} \\ s_\mu(g_1^t, \dots, g_\mu^t, g_1', \dots, g_\lambda') \text{ для } (\mu + \lambda)\text{-ES} \end{cases}$
   2.4 $t = t + 1$

Оператор PSelect реализован здесь на шаге 2.2.1, и устроен так, что все генотипы популяции имеют равные шансы стать родительскими. Reproduce соответствует шагу 2.2.2 и является довольно типичным для многих видов ЭА. Оператор Survive (шаг 2.3) просто уничтожает малопригодные особи.

3.2 Описание модели.

Модель исследования оператора мутации, используемая далее, была введена в [13]. Информация о пригодности различных генотипов не присутствует здесь в явном виде. Вместо этого используются определенные известные заранее вероятностные характеристики оператора мутации, которые и будут описаны в этой части.

Предположим, что d линий уровня функции пригодности заданы таким образом, что $0 = \Phi_0 < \Phi_1 < \Phi_2 \ldots < \Phi_d$. Количество линий может быть выбрано произвольно, нужно только стремиться к тому, чтобы для данного оператора мутации получилась наиболее точная модель. Каждой линии уровня сопоставим подмножество

$$H_i = \{g \in A^n : \Phi(g) \geq \Phi_i\}, \quad i = 0, \ldots, d.$$

Очевидно $H_{i+1} \subseteq H_i$. Благодаря неотрицательности функции пригодности, в H_0 попадают все генотипы. Также, для удобства, определим множество $H_{d+1} = \emptyset$. Предположим, что для всех $i = 0, ..., d$ и $j = 1, ..., d$ известны нижние и верхние оценки вероятностей перехода в результате мутации из $H_i \backslash H_{i+1}$ в H_j, то есть для каждого $g \in H_i \backslash H_{i+1}$

$$\alpha_{ij} \leq P\{Mut(g) \in H_j\} \leq \beta_{ij},$$

где $P\{Mut(g) \in H_j\} = \sum_{g' \in H_j} P\{Mut(g) = g'\}$. Пусть $\mathbf{A}$ обозначает матрицу с элементами α_{ij} где $i = 0, ..., d$, и $j = 1, ..., d$. Аналогичная матрица верхних оценок β_{ij} обозначается $\mathbf{B}$. Если для всех $i = 0, \ldots, d$, и $j = 1, \ldots, d$ вероятность $P\{Mut(g) \in H_j\}$ не зависит от выбора $g \in H_i \backslash H_{i+1}$ то существуют такие матрицы оценок, что $\mathbf{A} = \mathbf{B}$. В этом случае оператор мутации называется *ступенчатым* относительно набора подмножеств $H_0, H_1, \ldots, H_d$ (или просто ступенчатым). Матрицу $\mathbf{A} = \mathbf{B}$ будем называть матрицей пороговых переходных вероятностей. Следует отметить, что ступенчатые операторы мутации скорее всего почти не встречаются в прикладных задачах, однако, они оказываются полезными для дальнейшего анализа, как класс "идеальных" операторов.

Будем называть произвольную матрицу M с элементами $m_{ij}, i = 0, \ldots, d$, и $j = 1, \ldots, d$ *монотонной*, если $m_{i-1,j} \leq m_{ij}$ для всех i, j от 1 до d. Другими словами, матрица оценок переходных вероятностей монотонна, если для любого $j = 1, \ldots, d$, генотипы из произвольного подмножества H_i имеют оценки вероятностей перехода в H_j не меньшие, чем аналогичные оценки для генотипов из $H_{i'}$, для любого $i' < i$. Далее мы будем работать в основном с монотонными матрицами оценок. Очевидно, они существуют всегда. (Например $\mathbf{A} = \mathbf{0}$ где $\mathbf{0}$ – нулевая матрица, и $\mathbf{B} = \mathbf{U}$ где $\mathbf{U}$ – матрица, все элементы которой равны 1.)

В работе [13] в рамках этой модели получены оценки среднего количества генотипов с заданным значением пригодности на произвольной итерации одного упрощенного варианта ГА с турнирной селекцией

ГА с мутацией и s-турнирной селекцией.

1. Построить начальную популяцию Π^0.
2. Для $t = 0$ до $t_{max} - 1$
   2.1 для $k = 1$ до N
      Выбрать родительский генотип g', используя s-турнирную селекцию.
      Построить $g_k^{(t+1)} = Mut(g')$ и добавить его в популяцию Π^{t+1}.

Если $z_i^{(t)}$ – доля генотипов из H_i в популяции $\Pi^{(t)}$, то

$$\mathbf{E}[z_j^{(t+1)}] \geq \alpha_{dj} - \sum_{i=1}^{d} (\alpha_{i,j} - \alpha_{i-1,j}) \mathbf{E}[(1 - z_i^{(t)})^s]. \quad (5)$$

Наибольший интерес вызывает вопрос: является ли ГА лучшим среди подобных алгоритмов, и если нет, то какие могут быть сделаны улучшения. Конечно в общем случае однозначного ответа дать нельзя, но решение этого вопроса хотя бы в условиях монотонности и ступенчатости также может быть полезным. В качестве первого наблюдения отметим, что, если матрица $\mathbf{A}$ монотонна, то правая част выражения (5) является неубывающей функцией от s, то есть с возрастанием размера турнира, видимо, следует ожидать большего количества генотипов с высокой пригодностью, а значит и более эффективной работы алгоритма. Это дает основания устремить s в бесконечность. Турнирная селекция при $s = \infty$ с вероятностью 1 выбирает генотип с наибольшей пригодностью, поэтому такой вариант ГА реализуется очень просто, и его поведение аналогично поведению алгоритма $(1, \lambda)$-ES при значении $\lambda = N$. Тогда $b^{(t)}$ будет соответствовать лучшему генотипу в популяции $\Pi^{(t)}$, а λ его потомков – особям популяции $\Pi^{(t+1)}$. Разница может быть только в тех случаях, когда в популяции ГА есть несколько лучших генотипов с одинаковой пригодностью. Тогда для построения потомков в ГА могут выбираться разные родительские особи, а в $(1, \lambda)$-ES фиксируется одна. Однако, как будет видно из дальнейшего, вид вероятностного распределения для родительской особи не используется при построении оценок, поэтому наша модель в подобным различиям нечувствительна.

Исследуя различные алгоритмы, мы пытаемся ответить на вопрос: с какой вероятностью можно получить достаточно "хорошее" решение. Если генотип b построен случайным образом, то его качество характеризуется d-мерным *вектором вероятностей*:

$$P_b = (P\{b \in H_1\}, \dots, P\{b \in H_d\})$$

Для популяции $\Pi = (g_1, \dots, g_N)$ определим $b = Argmax(\Phi(g_1), \dots, \Phi(g_N))$ и будем считать вектором вероятности $P_\Pi = P_b$. Для некоторого эволюционного алгоритма A также определим вектор P_A – вектор вероятности генотипа (популяции), который возвращается алгоритмом A по окончании работы.

Легко видеть, что значение $\mathbf{E}[(1 - z_i^{(t)})^s]$ при $s \to \infty$ стремиться к величине $1 - P\{b^{(t)} \in H_i\}$. Иначе говоря, вектор $\mathbf{E}[z^{(t+1)}]$ выражается через вектор вероятностей для $b^{(t)}$, который мы обозначим через $P^{(t)}$.

3.3 Оценки функции распределения пригодностей генотипов на заданной итерации.

3.3.1 Анализ (1,λ)-ES.

Наша задача оценить $P\{b^{(t)} \in H_j\}$ для всех j при заданном t. Заметим, что всегда существует такой номер $0 \leq i \leq d$, что $b^{(t-1)} \in H_i \setminus H_{i+1}$, поэтому по формуле полной вероятности получаем

$$p_j^{(t)} = \sum_{i=0}^{d} P\{b^{(t)} \in H_j | b^{(t-1)} \in H_i \setminus H_{i+1}\} P\{b^{(t-1)} \in H_i \setminus H_{i+1}\}.$$

Определим для удобства $p_0^{(t-1)} = 1$ и $p_{d+1}^{(t-1)} = 0$. Тогда можно написать $P\{b^{(t-1)} \in H_i \setminus H_{i+1}\} = p_i^{(t-1)} - p_{i+1}^{(t-1)}$ для всех $i = 0, ..., d$. Следовательно

$$P\{b^{(t)} \notin H_j | b^{(t-1)} \in H_i \setminus H_{i+1}\} = \sum_{b \in H_i \setminus H_{i+1}} \frac{P\{b^{(t)} \notin H_j | b^{(t-1)} = b\} P\{b^{(t-1)} = b\}}{P\{b^{(t-1)} \in H_i \setminus H_{i+1}\}} \leq$$

$$\sum_{b \in H_i \setminus H_{i+1}} \frac{(1-\alpha_{ij})^\lambda P\{b^{(t-1)} = b\}}{P\{b^{(t-1)} \in H_i \setminus H_{i+1}\}} = (1-\alpha_{ij})^\lambda.$$

Предложение 1 *Компоненты вектора $P^{(t)}$ оцениваются следующим образом:*

$$p_j^{(t)} \geq 1 - (1-\alpha_{0j})^\lambda + \sum_{i=1}^{d} ((1-\alpha_{i-1,j})^\lambda - (1-\alpha_{ij})^\lambda) p_i^{(t-1)} \tag{6}$$

для всех $j = 1, \ldots, d$.

В случае ступенчатого оператора мутации (6) обращается в равенство, и тогда $P^{(t)}$ можно считать функцией $P^{(t)} = \mathbf{P}(P^{(t-1)})$.

Определим для $(d \times d)$-матрицы $\mathbf{W}$ матричную норму $||\mathbf{W}|| = \max_j \sum_{i=1}^d |w_{ij}|$. Чтобы представить рекуррентную оценку (6) в терминах последовательности $\{P^{(t)}\}$ потребуется результат, полученный в [13] и сформулированный здесь в несколько более общем виде.

Лемма 2 *Пусть $\mathbf{W}$ – $(d \times d)$-матрица с неотрицательными элементами, такая что $||\mathbf{W}|| < 1$. Предположим, что $\alpha \in \mathbf{R^d}$, $\beta \in \mathbf{R^d}$ и $\{\zeta^{(t)}\}, t = 0, 1, \ldots$ – векторы из $\mathbf{R^d}$.*
(a) Если $\zeta^{(t+1)} \geq \zeta^{(t)}\mathbf{W} + \alpha$ для всех $t \geq 0$, то

$$\zeta^{(t)} \geq \zeta^{(0)}\mathbf{W}^t + \alpha(\mathbf{I} - \mathbf{W})^{-1}(\mathbf{I} - \mathbf{W}^t). \tag{7}$$

(b) If $\zeta^{(t+1)} \leq \zeta^{(t)}\mathbf{W} + \beta$ для всех $t \geq 0$, то

$$\zeta^{(t)} \leq \zeta^{(0)}\mathbf{W}^t + \beta(\mathbf{I} - \mathbf{W})^{-1}(\mathbf{I} - \mathbf{W}^t), \tag{8}$$

где $\mathbf{I}$ – единичная матрица.

Доказательство. Случай (a). Из свойств линейных операторов следует, что матрица $(\mathbf{I} - \mathbf{W})^{-1}$ существует. Так как элементы $\mathbf{W}$ неотрицательны, то значение $\zeta^{(t)}\mathbf{W} + \alpha$ не возрастет, если вектор $\zeta^{(t)}$ заменить на $\zeta^{(t-1)}\mathbf{W} + \alpha$. Таким образом, мы получаем утверждение леммы индукцией по t. Доказательство случая (b) аналогично Заметим, что правая часть выражения (7) стремиться к фиксированной точке $\alpha(\mathbf{I} - \mathbf{W})^{-1}$ при $t \to \infty$, независимо от начального вектора $\zeta^{(0)}$.

Применим теперь лемму 2 к матрице $\mathbf{W}'$ с элементами $w'_{ij} = (1-\alpha_{i-1,j})^\lambda - (1-\alpha_{i,j})^\lambda$ и вектору $\alpha' = (1-(1-\alpha_{01})^\lambda, \ldots, 1-(1-\alpha_{0d})^\lambda)$.

Теорема 2 *Если $\mathbf{A}$ монотонна и $\alpha_{dj} - \alpha_{0j} < 1$ для $j = 1, \ldots, d$, то $P^{(t)} \geq \zeta^{(t)}$ где $\zeta^{(t)}$ удовлетворяет (7) при $\zeta^{(0)} = P^{(0)}, \mathbf{W} = \mathbf{W}'$ и $\alpha = \alpha'$.*

Доказательство. Неотрицательность величин w'_{ij} следует из монотонности $\mathbf{A}$. Осталось только проверить, что $||\mathbf{W}|| < 1$. Действительно, $||\mathbf{W}|| = \max_j \sum_{i=1}^d |w_{ij}| = \max_j \sum_{i=1}^d w_{ij} = \max_j\{(1-\alpha_{0j})^\lambda - (1-\alpha_{dj})^\lambda\}$. Полученное выражение меньше 1 тогда, и только тогда, когда $\alpha_{0j} > 0$ или $\alpha_{dj} < 1$ для всех $j = 1, \dots, d$, что эквивалентно $\alpha_{dj} - \alpha_{0j} < 1$.■

Условие $\alpha_{dj} - \alpha_{0j} < 1$ обеспечивает существование и единственность предела $\lim_{t\to\infty} P^{(t)}$, подобно тому, как неразложимость и непериодичность цепи Маркова (грубо говоря, возможность попасть из произвольного состояния в любое другое с ненулевой вероятностью) обеспечивает сходимость к стационарному распределению. Надо сказать, что наше условие не является необходимым, так как рассматривалась только одна из возможных матричных норм. Тем не менее, частный случай, когда $\alpha_{0j} > 0, j = 1, \dots, d$, означает, что генотип из произвольного уровня под действием мутации может оказаться в любом другом уровне (это следует из монотонности). Эта ситуация типична для многих практически используемых реализаций мутации, поэтому такое условие не представляется сильно ограничительным.

Заметим также, что компоненты предельного вектора вероятностей для (1,λ)-ES не обязаны равняться 1. При этом вероятность найти оптимум может стремиться к 1 при $t \to \infty$. Дело здесь в том, что в соответствии с определением ЭА, результатом работы считается лучший генотип, найденный за все итерации. Рассмотренные же здесь модели в целях упрощения вычислений описывают состояние только конечной популяции. Для $(\mu + \lambda)$-ES это не важно, так как лучший генотип всегда присутствует в конечной популяции, но для ГА и (μ,λ)-ES разница существенна. В принципе есть смысл рассмотреть (1,λ)-ES и с этой точки зрения, но это не входит в данную работу.

Верхние оценки компонентов $P^{(t)}$ получаются из матрицы $\mathbf{B}$ таким же способом. Определим матрицу $\widehat{\mathbf{W}}'$ с элементами $\hat{w}'_{ij} = (1 - \beta_{i-1,j})^\lambda - (1 - \beta_{i,j})^\lambda$, $i = 1, ..., d$, $j = 1, ..., d$, и вектор $\beta' = (1 - (1 - \beta_{01})^\lambda, \dots, 1 - (1 - \beta_{0d})^\lambda)$.

Теорема 3 *Если* $\mathbf{B}$ *монотонна и* $\beta_{dj} - \beta_{0j} < 1$ *для* $j = 1, \dots, d$, *то* $P^{(t)} \le \zeta^{(t)}$ *где* $\zeta^{(t)}$ *удовлетворяют* (8) *при* $\zeta^{(0)} = P^{(0)}, \mathbf{W} = \widehat{\mathbf{W}}'$ *и* $\beta = \beta'$.

Переход от ГА к (1,λ)-ES был сделан путем неограниченного увеличения размера турнира – параметра s оператора турнирной селекции. Обоснованность такого перехода при монотонных оценках уже пояснялась. Следующая теорема формально утверждает превосходство (1,λ)-ES над ГА, построенном на основе мутации и селекции произвольного типа (не обязательно турнирной).

Наша цель – сравнить производительности алгоритмов при работе в один и тот же промежуток времени. Время работы считается по количеству применений мутации. Критерием производительности алгоритма будем считать его вектор вероятностей, который для ГА будем обозначать через $R^{(t)}$, где t – номер итерации. Для (1,λ)-ES и (1+1)-ES сохраним прежнее обозначение $P^{(t)}$.

Теорема 4 *Пусть монотонный ступенчатый оператор мутации используется в ГА с произвольной селекцией и (1,λ)-ES, при $\lambda = N$. Если $P^{(0)} \ge R^{(0)}$, то $P^{(t)} \ge R^{(t)}$ для всех t.*

Доказательство. Достаточно доказать утверждение для $t = 1$, тогда общий случай легко доказывается индукцией по t.

(a)Пусть $r^{(t)}$ – лучший генотип в популяции Π^t, $r^{(1,k)}$ – лучший среди первых k генотипов, включенных в Π^1, $1 \le k \le N$. Аналогично, $b^{(1,k)}$ – лучший среди первых k потомков $b^{(0)}$. Рассмотрим случай, когда $r^{(0)} \in H_i \setminus H_{i+1}$ и $b^{(0)} \in H_i \setminus H_{i+1}$ для любого фиксированного i. Из свойства монотонности и ступенчатости оператора следует, что для всех j

$$P\{Mut(g') \notin H_j\} \ge P\{Mut(b^{(0)}) \notin H_j\}, \tag{9}$$

где g' выбран в результате селекции. Предположим по индукции, что $P\{r^{(1,k)} \notin H_j\} \ge P\{b^{(1,k)} \notin H_j\}$ для всех $j = 1, \dots, d$, при $2 \le k \le N-1$ (для $k = 1$ это следует непосредственно из (9)). Тогда

$$P\{r^{(1,k+1)} \notin H_j\} = P\{r^{(1,k)} \notin H_j\}P\{Mut(g') \notin H_j\} \ge$$

$$P\{b^{(1,k)} \notin H_j\}P\{Mut(b^{(0)}) \notin H_j\} = P\{b^{(1,k+1)} \notin H_j\} \tag{10}$$

(b) Докажем неравенство $P^{(1)} \ge R^{(1)}$ при условии $P^{(0)} \ge R^{(0)}$. Для этого воспользуемся формулой полной вероятности и результатом пункта (a):

$$P\{r^{(1)} \notin H_j\} = \sum_{i=0}^{d} P\{r^{(1)} \notin H_j | r^{(0)} \in H_i \setminus H_{i+1}\} P\{r^{(0)} \in H_i \setminus H_{i+1}\} \ge$$

$$\sum_{i=0}^{d} P\{b^{(1)} \notin H_j | b^{(0)} \in H_i \setminus H_{i+1}\} P\{b^{(0)} \in H_i \setminus H_{i+1}\} = P\{b^{(1)} \notin H_j\} \tag{11}$$

(c) Наконец, в общем случае при $P^{(0)} \ge R^{(0)}$ вспомним, что $P^{(t)}$ можно рассматривать как функцию $P^{(t)} = \mathbf{P}(P^{(t-1)})$. Тогда, благодаря неотрицательности коэффициентов при $p_1^{(t-1)}, ..., p_d^{(t-1)}$ в формуле (6), получаем $\mathbf{P}(P^{(0)}) \ge \mathbf{P}(R^{(0)}) \ge R^{(1)}$.■

3.3.2 Анализ (1+1)-ES.

Рассматривая ход работы $(1, \lambda)$-ES в случае монотонного ступенчатого оператора мутации, можно отметить два его серьезных недостатка. Во-первых: если при построении потомков нашелся генотип с пригодностью большей, чем $\Phi(b^{(t)})$, то выгоднее будет прекратить мутации $b^{(t)}$ и сразу использовать найденного потомка в качестве родителя. Во-вторых: если пригодности всех потомков оказались меньшими, чем $\Phi(b^{(t)})$, то не стоит менять родительский генотип $b^{(t)}$ на менее пригодный, лучше оставить его прежним. Это замечание ведет к исследованию (1+1)-ES и его формальному сравнению с (1,λ)-ES. Определим *вектор вероятностей для (1+1)-ES* следующим образом:

$$Q^{(t)} = (q_1^{(t)}, ..., q_d^{(t)}) = (P\{g^{(t)} \in H_1\}, \dots, P\{g^{(t)} \in H_d\}).$$

Применяя формулу полной вероятности получаем:

$$q_j^{(t)} \ge \sum_{i=0}^{j-1} \alpha_{ij} P\{g^{(t)} \in H_i \setminus H_{i+1}\} + q_j^{(t-1)} = \sum_{i=0}^{j-1} \alpha_{ij} (q_i^{(t-1)} - q_{i+1}^{(t-1)}) + q_j^{(t-1)}.$$

Предложение 2 *Компоненты вектора $Q^{(t)}$ оцениваются следующим образом:*

$$q_j^{(t)} \geq \alpha_{0j} + \sum_{i=1}^{j-1} (\alpha_{ij} - \alpha_{i-1,j}) q_i^{(t-1)} + (1 - \alpha_{j-1,j}) q_j^{(t-1)}, \tag{12}$$

для $j = 1, \ldots, d$.

Заметим, что значения $\alpha_{dj}, j = 1, \ldots, d$ нигде не встречаются в выражениях 12. Это означает, что, если например, монотонность нарушается только в последней строке, то для (1+1)-ES такая матрица остается монотонной. Исходя из этого назовем матрицу $\mathbf{A}$ слабо монотонной, если $\alpha_{i-1,j} \leq \alpha_{ij}$ для всех $i = 1, \ldots, d-1, j = 1, \ldots, d$. Оператор Mut назовем слабо ступенчатым $\alpha_{ij} = \beta ij$ для всех $i = 0, \ldots, d-1, j = 1, \ldots, d$. В случае слабо ступенчатого оператора мутации (12) обращается в равенство, поэтому мы можем рассматривать $Q^{(t)}$ как функцию от $Q^{(t)} = \mathbf{Q}(Q^{(t-1)})$.

Также мы можем применить лемму 2, определив матрицу $\mathbf{W}''$ и вектор α'' как:

$$w''_{ij} = \begin{cases} \alpha_{i,j} - \alpha_{i-1,j}, & i < j \\ 1 - \alpha_{j-1,j}, & i = j \\ 0, & i > j, \end{cases} \qquad \alpha'' = (\alpha_{01}, \ldots, \alpha_{0d}).$$

Теорема 5 *Если $\mathbf{A}$ монотонна и $\alpha_{0j} > 0$ для $j = 1, \ldots, d$, то $Q^{(t)} \geq \zeta^{(t)}$ где $\zeta^{(t)}$ удовлетворяют (7) при $\zeta^{(0)} = Q^{(0)}, \mathbf{W} = \mathbf{W}''$ и $\alpha = \alpha''$.*

Легко убедиться, что $\mathbf{1} = \alpha'' + \mathbf{1W}''$, где $\mathbf{1} = (1, \ldots, 1)$, откуда следует $P\{g^{(t)} \in H_d\} \to 1$ при $t \to \infty$.

Аналогично построим верхние оценки. Определим

$$\hat{w}''_{ij} = \begin{cases} \beta_{i,j} - \beta_{i-1,j}, & i < j \\ 1 - \beta_{j-1,j}, & i = j \\ 0, & i > j, \end{cases} \qquad \beta'' = (\beta_{01}, \ldots, \beta_{0d}).$$

Теорема 6 *Если $\mathbf{B}$ монотонна и $\beta_{0j} > 0$ для $j = 1, \ldots, d$, то $Q^{(t)} \leq \zeta^{(t)}$ где $\zeta^{(t)}$ удовлетворяет (8) при $\zeta^{(0)} = Q^{(0)}, \mathbf{W} = \widehat{\mathbf{W}}''$ и $\beta = \beta''$.*

Подобно приведенному выше сравнению ГА и (1,λ)-ES можно сформулировать превосходство (1+1)-ES над (1,λ)-ES, как это и было сделано в [**BorEr**]. Однако, здесь можно доказать более общий результат о том, что (1+1)-ES при сделанных предположениях оказывается лучше любого алгоритма, основанного только на использовании мутации. Это тем более важно потому, что, как было отмечено разделе 3.3.2, наша модель (1,λ)-ES описывает только генотип на конечной итерации, а не лучший генотип вообще.

Пусть A – некоторый алгоритм, который удовлетворяет свойствам: задана начальная популяция $\Pi^{(0)} = (a^{(0,1)}, \ldots, a^{(0,N)})$, и на каждой итерации очередное решение $a^{(t)}$ получается однократным применением мутации к одному из имеющихся $a^{(0,1)}, \ldots, a^{(0,N)}, a^{(1)}, \ldots, a^{(t-1)}$ (как видно, ГА, (1,λ)-ES и (1+1)-ES также удовлетворяют этой схеме) Обзначим $P_A^{(t)}$ – вектор вероятностей для генотипа возвращаемого алгоритмом A за t итераций.

Теорема 7 *Пусть алгоритмы A и (1+1)-ES используют один и тот же слабо ступенчатый и монотонный оператор мутации Mut. Тогда, если $P_{\Pi^{(0)}} \le Q^{(0)}$, то $P_A^{(t)} \le Q^{(t)}$.*

Доказательство. Итерации алгоритма A строят последовательность генотипов $a^{(1)}, \ldots, a^{(t)}$, пусть $b^{(t)}$ - лучший из них. Без уменьшения общности можно считать, что на итерации t алгоритм A возвращает генотип $b^{(t)}$, т.к. иначе вектор $P_A^{(t)}$ может только уменьшиться.

Доказательство проведем индукцией по t. Предположим, что $P^{(t-1)} \le Q^{(t-1)}$. На t-ой итерации $a^{(t)} = Mut(a')$, где a' – генотип выбранный для мутации. Тогда $b^{(t)} = s_1(Mut(a'), b^{(t-1)})$. (Напомним, что оператор s_1 в соответствии с пунктом 3.1 осуществляет выбор одного лучшего генотипа).

Сначала докажем, что $P\{s_1(Mut(a'), b^{(t-1)}) \in H_j\} \le P\{s_1(Mut(b^{(t-1)}), b^{(t-1)}) \in H_j\}$ для всех j.

$$P\{Mut(a') \in H_j \text{ или } b^{(t-1)} \in H_j\} =$$
$$P\{b^{(t-1)} \in H_j\} + P\{Mut(a') \in H_j, b^{(t-1)} \notin H_j\}$$

Второе слагаемое разложим по формуле полной вероятности при гипотезах $\{b^{(t-1)} \in H_i \setminus H_{i+1}\}$, причем достаточно взять $i = 1, \ldots, j-1$. Получится

$$P\{Mut(a') \in H_j, b^{(t-1)} \notin H_j\} = \sum_{i=0}^{j-1} P\{Mut(a') \in H_j | b^{(t-1)} \in H_i \backslash H_{i+1}\} P\{b^{(t-1)} \in H_i \backslash H_{i+1}\}$$

Из слабой монотонности матрицы пороговых переходных вероятностей следует

$$P\{Mut(a') \in H_j | b^{(t-1)} \in H_i \setminus H_{i+1}\} \le P\{Mut(b^{(t-1)}) \in H_j | b^{(t-1)} \in H_i \setminus H_{i+1}\}$$

В результате чего получается

$$P\{s_1(Mut(a'), b^{(t-1)}) \in H_j\} \le$$
$$P\{b^{(t-1)} \in H_j\} + \sum_{i=0}^{j-1} P\{Mut(b^{(t-1)}) \in H_j | b^{(t-1)} \in H_i \setminus H_{i+1}\} =$$
$$P\{s_1(Mut(b^{(t-1)}), b^{(t-1)}) \in H_j\}$$

Выражение $s_1(Mut(b^{(t-1)}), b^{(t-1)})$ означает не что иное, как применение одной итерации (1+1)-ES к генотипу $b^{(t-1)}$. Вспомним, что Mut слабо ступенчатый, и мы можем рассматривать $Q^{(t)}$ как функцию $Q^{(t)} = \mathbf{Q}(Q^{(t-1)})$. Поэтому полученное неравенство можно записать в виде:

$$P_A^{(t)} \le \mathbf{Q}(P_{b^{(t-1)}}).$$

Из неотрицательности коэффициентов при $q_j^{(t)}$ в формуле (12) следует $\mathbf{Q}(P_{b^{(t-1)}}) \le \mathbf{Q}(Q^{(t-1)}) = Q^{(t)}$. ■

В [13] было показано, что оператор мутации, который с фиксированной вероятностью изменяет каждый элемент бинарной строки на противоположный независимо от других, является монотонным и ступенчатым для задачи максимизации количества единиц в строке. Аналогичный результат получен для ЗВП на графах специальной структуры. Конечно, для многих известных операторов мутации, маловероятно, что они окажутся монотонными и ступенчатыми при решении трудных задач оптимизации. Например, если множество линий уровня $\Phi_1, ..., \Phi_d$ состоит из всех возможных значений функции пригодности, то при наличии локальных, но не глобальных, оптимумов в смысле метрики Хэмминга, упомянутый только что оператор мутации не будет монотонным и ступенчатым. Тем не менее, ступенчатый оператор может быть полезен в качестве теоретического вспомогательного средства. Для данного оператора Mut может быть построен монотонный ступенчатый оператор Mut' с пороговыми вероятностями перехода, равными монотонным нижним оценкам для Mut. Тогда Mut' можно рассматривать как наихудшую реализацию действия Mut. При таком подходе теорема 7 утверждает, что гарантированная нижняя оценка для $P^{(t)}$ не лучше, чем гарантированная нижняя оценка для $Q^{(t)}$.

3.4 О вероятностной сложности оператора мутации

В рамках теории вычислительной сложности было получено значительное число пессимистических результатов относительно алгоритмов оптимизации(см. [2]). В этом разделе исследуются нижние оценки переходных вероятностей αij в ситуации, когда нахождение приближенного решения с определенной оценкой точности является NP-трудной задачей (такие примеры можно найти в [6, 4]). Будем исползовать далее обозначения, следуя [7]. Напомним, что Σ обозначает алфавит $\{0, 1\}$, а Σ^* – множество всех конечных строк алфавита Σ.

Определение 1 *Задача NP-максимизации P_{max} –это тройка $P_{max} = (I, Sol, f_x)$, где*

1. $I \subseteq \Sigma^*$ – множество примеров (индивидуальных задач) P_{max} распознаваемое за полиномиальное время.

2. Для данного примера $x \in I$, $Sol(x) \subseteq \Sigma^*$ обозначает множество допустимых решений x. Для $x, y \in \Sigma^*$ ответ на вопрос о принадлежности y множеству $Sol(x)$ может быть дан за полиномиальное время, и существует такой полином h, что $|y| = h(|x|)$ [2] для всех $x \in I$ и $y \in Sol(x)$.

3.Для $x \in I$ и $y \in Sol(x)$, $f_x(y)$ – положительная целочисленная целевая функция, которая вычисляется за полиномиальное время.

Оптимальное значение целевой функции на примере x обозначим через $f_x^* = \max_{y \in Sol(x)} f_x(y)$. Мы рассматриваем только задачу максимизации, хотя определение и дальнейшие результаты легко переносятся на случай задачи на минимум.

Пусть для простоты кодировка допустимого решения задачи NP-максимизации используется в качестве генотипа, и функция пригодности на множестве $Sol(x)$ совпадает с целевой функцией и равна 0 для остальных генотипов из $\Sigma^{h(|x|)}$, хотя

[2] Обычно используется неравенство $|y| \leq h(|x|)$, требование равенства не ограничивает общности и нужно только для упрощения вычислений. [7]

аналогично можно рассматривать и более широкий класс представлений, которые реализуются отображениями, вычисляемыми за полиномиальное время.

Определение 2 *Для примера x задачи NP-максимизации и $y \in Sol(x)$, оценкой точности y относительно x назовем число $R(x,y) = f_x^*/f_x(y)$.*

Определение 3 *Пусть P_{max} – задача NP-максимизации и T – алгоритм, который для любого примера x из P_{max} такого, что $Sol(x) \neq \emptyset$, находит допустимое решение $T(x)$ за полиномиальное время. При заданной функции $r : \mathbf{Z}^+ \to [1,\infty)$, T называется r-приближенным алгоритмом для P_{max} если оценка точности $T(x)$ относительно x такова, что $R(x,T(x)) \leq r(|x|)$. Само решение $T(x)$ называется $r(|x|)$-приближенным решением.*

Получение для любого примера $x \in P_{max}$ такого решения y, что $f_x(y) \geq f_x^*/r(|x|)$ (при условии, что $Sol(x) \neq \emptyset$) называется *аппроксимацией P_{max} с оценкой r*. Например, известно, что задача максимального разреза NP-трудна для аппроксимации с оценкой 1.012. Аппроксимация задачи о наибольшей клике с оценкой $|V|^{1-\varepsilon}$ (где $|V|$ – число вершин графа и $\varepsilon > 0$ –произвольная константа) за полиномиальное время также не представляется возможной. Более подробное описание результатов о трудности аппроксимации дано в [6].

Нам также потребуется формальные определения вероятностного алгоритма и класса языков BPP, распознаваемых с ограниченной вероятностью за полиномиальное время (см. [18]). Под вероятностным алгоритмом мы подразумеваем алгоритм, который может быть реализован на *вероятностной машине Тьюринга*, то есть такой машине, у которой есть специальное состояние для "подбрасывания монетки". Когда машина оказывается в этом состоянии, она получает случайное число равное 0 с вероятностью 1/2 и равное 1 с вероятностью 1/2. *Полиномиальная вероятностная машина Тьюринга* – это такая машина, которая останавливается за полиномиальное число шагов.

Определение 4 *BPP –это класс языков $L \subseteq \Sigma^*$ для которых существует такая полиномиальная вероятностная машина Тьюринга M, что:*
1) *для всех* $x \in L$ $P\{M$ *возвращает* $1\} \geq 2/3$.
2) *для всех* $x \notin L$ $P\{M$ *возвращает* $0\} \geq 2/3$.

Изучая сложность оператора мутации с определенными нижними оценками переходных вероятностей, мы будем рассматривать его как полиномиальный вероятностный алгоритм. Нам потребуется следующий результат:

Лемма 3 *Пусть P_{max} – задача NP-максимизации и аппроксимация P_{max} с оценкой r NP-трудна. Тогда, если $NP \not\subseteq BPP$, то не существует полиномиального вероятностного алгоритма, который находит r-приближенные решения для всех примеров x из P_{max} с вероятностью более, чем $1/\mathrm{poly}(|x|)$, где $\mathrm{poly}(|x|)$ – полином от длины входа x.*

Доказательство. Так как r-аппроксимация P_{max} NP-трудна, то существует полиномиальная машина Тьюринга M которая решает задачу распознавания некоторого NP-полного языка $L \in NP$, используя запросы к оракулу для аппроксимации P_{max} с оценкой r. Количество таких запросов ограничено некоторым полиномом $\mathrm{poly}_1(|x'|)$, где $|x'|$ – длина входа x' задачи распознавания. Построим вероятностную машину Тьюринга M', в которой каждый запрос машины M к оракулу заменяется k независимыми запусками вероятностного алгоритма, определенного в формулировке (число k будет указано позже). Лучшее найденное за k запусков решение считается результатом запроса.

Пусть $p(x)$ – вероятность того, что вероятностный алгоритм находит r-приближенное решение на входе x за один запуск. Тогда вероятность получить требуемое r-приближенное решение хотя бы один раз – не меньше, чем $1-(1-p(x))^k$. Кроме того $p(x) \geq 1/\mathrm{poly}_2(|x'|)$ для некоторого полинома $\mathrm{poly}_2(|x'|)$, так как длина входа x полиномиально ограничена от $|x'|$. Тогда легко проверить, что $p(x) \geq 1-\exp\{-1/\mathrm{poly}_2(|x'|)\}$. Пусь p' обозначает вероятность того, что во всех запросах r-приближенные решения были получены. Тогда

$$p' \geq \left(1-\exp\left\{-\frac{k}{\mathrm{poly}_2(|x'|)}\right\}\right)^{\mathrm{poly}_1(|x'|)} \geq \exp\left\{\frac{-O(1)\cdot \mathrm{poly}_1(|x'|)}{\exp(k/\mathrm{poly}_2(|x'|))}\right\}.$$

Выбирая подходящее полиномиально ограниченное значение k, можно добиться выполнения $p' \geq \frac{2}{3}$. Таким образом машина M' правильно распознает любой вход x' за полиномиальное время с вероятностью не менее, чем $\frac{2}{3}$, поэтому $L \in BPP$. ■

Предположение $NP \not\subseteq BPP$ эквивалентно $NP \neq RP$, и принято считать его верным [17]

Полагая $r = 1$ в лемме 3 заключаем, что для для любой NP-трудной задачи NP-максимизации по-видимому нет вероятностных полиномиальных алгоритмов, которые находили бы оптимальное решение с вероятностью оцененной снизу положительной константой.

С помощью леммы 3 можно получить пессимистические оценки вероятности получения "хороших" решений полиномиальным оператором мутации, применяя его к генотипу, представляющему относительно "слабое" решение.

Теорема 8 *Пусть P_{max} – задача NP-максимизации, аппроксимация P_{max} с оценкой r NP-трудна, но существует r'-приближенный алгоритм. Тогда, если $NP \not\subseteq BPP$, то не существует полиномиального оператора мутации с монотонными нижними оценками, удовлетворяющими $\alpha_{ij} \geq 1/\mathrm{poly}(|x|)$ при $\Phi_i \leq f^*_x/r'$, $\Phi_j \geq f^*_x/r$, для некоторого полинома $\mathrm{poly}(|x|)$ и для всех $x \in P_{max}$*

Доказательство. Рассмотрим полиномиальный вероятностный алгоритм который сначала находит решение y r'-приближенным алгоритмом, и применяет оператор мутации к генотипу $g = y$. Так как $g \in H_i$ то должен существовать номер k, $i \leq k \leq d$, что $g \in H_k \backslash H_{k+1}$. Благодаря монотонности переходных оценок, $\alpha_{kj} \geq \alpha_{ij} \geq 1/\mathrm{poly}(|x|)$, и требуемое утверждение следует из леммы 3. ■

Опять полагая $r = 1$,заключаем, что при решении NP-трудной задачи величина α_{id}, где $\Phi_i \leq f_x^*/r'$, $\Phi_в = f_x^*$, будет убывать быстрее чем любая функция вида $1/poly(|x|)$.

4 Вычислительные эксперименты.

В этом параграфе приведены результаты тестирования рассмотренных ранее алгоритмов и сравнение с другими известными алгоритмами подобного типа. Также рассматривается способ визуального представления развития эволюционного процесса на основе построенных моделей.

4.1 Эксперименты на тестовых задачах.

Алгоритмы RGrd и GA-RGrd были реализованы на языке Borland C++ 5.0 и тестировались на ЭВМ Celeron-400 (тактовая частота процессора 400 МГц). Эксперимент поставлен в соответствии со схемой, описанной в [10], где приведены результаты исследования детерминированной эвристики CLQMRG и генетических алгоритмов GA-SS [10] и GA-OCH [5]. CLQMRG использует подход, основанный на построении наибольшей клики в объединении двух данных клик при помощи алгоритма поиска максимального паросочетания. В GA-SS и GA-OCH данная эвристика применяется для реализации оператора оптимизированного кроссинговера. В качестве тестовых задач используются невзвешенные ЗВП полученные из задач о максимальной клике библиотеки DIMACS доступных по адресу *ftp://dimacs.rutgers.edu.* Время счета для каждой задачи ограничивается временем работы CLQMRG. Для сопоставления быстродействия Celeron-400 и применяемой в [10] ЭВМ HP-9000 Apollo была использована тестовая программа "dfmax.c", предложенная в [16]. Сравнение показало, что Celeron-400 примерно в 5.72 раз быстрее HP-9000 Apollo. В течение отведенного времени на каждой задаче алгоритм GA-RGrd стартовал λ раз и работал одинаковые промежутки времени. Были опробованы различные настройки параметров алгоритма. Удовлетворительные результаты были получены при $p_m = 0.01, N = 200, \lambda = 10$. Алгоритм RGrd запускался многократно в течение такого-же времени. Также были проведены расчеты с алгоритмом Grd.

Результаты экспериментов приведены в таблице 1. Для каждой задачи указана ее размерность (число вершин и ребер графа), мощность оптимального или наилучшего известного покрытия и время счета T в секундах ЭВМ Celeron-400. Затем для каждого алгоритма приводится лучшее найденное им решение. В скобках дается число случаев его нахождения, далее – число случаев нахождения решения со значением целевой функции большим на 1, и затем – на 2. Например в задаче **C500.9** алгоритм GA-RGrd за 10 запусков нашел покрытие весом 443 три раза, весом 444 – пять раз, и весом 446 – два раза. Для алгоритма RGrd также указано через точку с запятой число запусков, которое успел сделать алгоритм в течение заданного времени. Это позволяет судить о вероятности нахождения тех или иных решений, а также о скорости работы алгоритма. Жирным шрифтом отмечены оптимальные решения.

Из результатов видно, что алгоритм RGrd при многократном запуске способен находить решения значительно лучшие, чем Grd. При этом вероятность найти оптимальное решение алгоритмом RGrd за один запуск крайне мала. GA-RGrd всегда работал не хуже, чем RGrd, причем, если на задачах серий **Brock** и **p-hat** эти алгортмы находят похожие по качеству решения, то на задачах серии **gen** обнару-

Таблица 2: Сравнение известных эвристик с GA-RGrd

Количество найденных решений	CLQMRG	GA-SS	GA-OCH	GA-RGrd
оптимальных	19	20	18	19
лучших, чем GA-RGrd	2	2	1	-
худших, чем GA-RGrd	1	2	2	-
такого же качества	22	21	22	-

жилось значительное преимущество GA-RGrd.

Сравнение с известными эвристиками представлено в таблице 2, где приведено число задач, для которых соответствующий алгоритм находил оптимальные, лучшие, худшие по сравнению с GA-RGrd решения и одинаковые с ним по целевой функции. Как видно, GA-RGrd не уступает по производительности ранее разработанным ГА. При этом нет оснований выделить один доминирующий над остальными алгоритм.

4.2 Графическое представление.

В случае ступенчатого оператора выражения 6 позволяют построить последовательность векторов $P^{(1)}, \ldots, P^{(t)}$ для всех t при заданном $P^{(0)}$. При $d = 2$ вектор $P^{(t)}$ можно представить точкой на плоскости координат (p_1, p_2). Иначе говоря, при заданной матрице перехода каждая точка $P^{(0)}$ опеделяет единственную траекторию.

Другой способ визуализации процесса работы алгоритма отражает скорость сходимости при разных начальных условиях. Показано, что в определенных условиях существует предел $P = \lim_{t\to\infty} P^{(t)}$. Задавшись уровнем точности $\varepsilon > 0$ и выбрав векторную норму $\|\cdot\|$ (мы пользуемся эвклидовой), можно для каждой точки $P^{(0)}$ посчитать значение $l(P^{(0)}) = \min\{t : \|P^{(t)} - P\| < \varepsilon\}$ – показатель удаленности (число итераций) от неподвижной точки. Затем самые дальние точки экрана окрашиваются в белый цвет, самые близкие – в черный, а остальные – в промежуточные цвета. Полученная картина может помочь при выборе наилучших начальных условий.

Переменные p_1 и p_2 имеют вероятностный смысл только в треугольнике $\{0 \le p_1 \le 1, 0 \le p_2 \le p_1\}$. Однако и вне этой области можно формально воспользоваться выражениями 6. Это позволяет более широко увидеть картину поведения переходного оператора. Аналогично исследуются последовательности $Q^{(t)}$ на плоскости (q_1, q_2), используя формулы 12.

Вспомним упоминавшийся уже оператор мутации, который с вероятностью p_m и независимо от других изменяет каждый элемент x_i бинарной строки $x \in \{0,1\}^n$. В [13] получены выражения для элементов матрицы пороговых вероятностей этого оператора в применении к задаче ONEMAX: $\{\max \sum_{i=1}^n x_i | x \in \{0,1\}^n\}$. На рисунках 1-4 показаны раскраски и некоторые траектории векторов вероятности алгоритмов (1,2)-ES и (1+1)-ES при значениях $p_m = 0.1$ и $p_m = 0.9$.

При $p_m = 0.1$ матрица пороговых переходных вероятностей является монотонной. Любопытно, что соответствующие графики (рис. 1 и 3) оказались очень похожими, в том смысле, что, судя по наклонным полосам, на скорость сходимости в равной степени влияют начальные данные по первой и второй координате.

Рис. 2 иллюстрирует то, как $(1,\lambda)$-ES может терять хорошие решения. Высокая вероятность мутации $p_m = 0.9$ заставляет алгоритм переходить поочередно от хороших точек к плохим. Начиная с самой "непригодной" особи алгоритм способен уже на второй итерации находить оптимум с высокой вероятностью, но далее эта вероятность уменьшается и стремится к сравнительно малому значению.

Для (1+1)-ES немонотонность здесь означает то, что с увеличением q_1 при фиксированном q_2 возрастает число итераций до нахождения оптимума (рис. 4). Лучшими начальными условиями здесь будут точки, в которых $q_1 = q_2$.

Особый интерес вызывает возможность применения полученных теоретических результатов к трудным задачам оптимизации. В качестве примера возьмем используемый в GA-RGrd оператор мутации MutVCP. Хотя не удалось теоретически построить нетривиальные оценки переходных вероятностей, для задачи BR200_2 оказалось возможно найти их экспериментально, путем многократного применением мутации. Применяя хорошо известный метод ветвей и границ, установлено, что граф задачи BR200_2 имеет одно покрытие размера 188 (оптимальное), два неизбыточных покрытия размера 189 и 171 неизбыточное покрытие размера 190. Все они были найдены и выписаны в явном виде, и только их мы будем считать генотипами. Остальные покрытия в целях уменьшения вычислительных затрат не рассматриваются. Результат оператора мутации – генотип, который всегда представляет неизбыточное покрытие, но его вес может быть больше 190. Пусть в этом случае MutVCP возвращает старый генотип без изменения. Определим $H_0 = \{g : \Phi(g) \leq 190\}, H_1 = \{g : \Phi(g) \leq 189\}, H_2 = \{g : \Phi(g) = 188\}$. Для каждого генотипа $g \in H_i \subset H_{i+1}$ вероятность $P\{MutVCP(g) \in H_j\}$ находилась осуществлением 50000 независимых запусков MutVCP. Получены следующие числовые значения:

$$\mathbf{A} = 10^{-4} \begin{pmatrix} 1.4 & 1.3 \\ 4.1 & 3.8 \\ 0.2 & 0.06 \end{pmatrix} \quad , \qquad \mathbf{B} = 10^{-4} \begin{pmatrix} 7.5 & 7.1 \\ 5 & 4.6 \\ 0.2 & 0.06 \end{pmatrix}$$

Применяя метод статистической оценки доверительного интервала, можно показать, что с вероятностью 0.95 наибольшее отклонение от реальных значений не превышает $3 \cdot 10^{-4}$. Заметим, что матрица $\mathbf{A}$ является слабо монотонной. На рис.5 и 6 изображены траектории для нижних и верхних оценок вектора $Q^{(t)}$ и его точного значения, полученного статистически. Выделенными точками отмечены оценки и точное значение $Q^{(500)}$ на рис. 5 и $Q^{(6000)}$ на рис. 6. Вертикальные и горизонтальные линии, проведенные из этих точек, наглядно показывают соотношения между ними.

На рис. 5 особого интереса заслуживают вертикальные полосы, которые указывают на слабую чувствительность к начальным данным по q_2 , особенно вблизи точки (0,0). Раскраска для немонотонной матрицы $\mathbf{B}$ на рис. 6 похожа на раскраску рис. 4. Интересно также отметить, что при большом значении t реальное значение $Q^{(t)}$ проходит ближе к верхней оценке, чем к нижней.

В связи с этим, возвращаясь также к результатам в Таблице 1, можно утверждать, что мутация вполне может быть хорошим самостоятельным средством поиска решений, а не только дополнением к кроссинговеру для поддержания разнообразия в популяции.

5 Заключение.

В данной работе представлены результаты исследования эволюционных алгоритмов по двум важным направлениям: разработка алгоритмов для практического применения и теоретический анализ.

Предложенный ГА для задачи о вершинном покрытии показывает хорошие результаты на тестовых задачах и по крайней мере не уступает одному наиболее успешному варианту ГА, разработанного Балашем. Представляется перспективным для дальнейшего усовершенствования ГА использовать технику самонастройки внутренних параметров, которая широко используется в зволюционных стратегиях. Более того, известно, что ЭС в области непрерывной оптимизации показывают более высокую эффективность чем ГА, поэтому определенный интерес может представлять построение алгоритмов ЭС для задач дискретной оптимизации и сравнение их с ГА.

В теоретической части исследуются два простейших алгоритма из класса ЭС. Построены верхние и нижние оценки вероятности нахождения решения с заданным значением целевой функции. Основной результат этой части – теорема о которая достаточно полно характеризует слабо ступенчатый монотонный оператор мутации. Применение некоторых результатов теории сложности показывает, что такой оператор в принципе может существовать и для NP-трудных задач, но с определенными ограничениями на переходные вероятности. Важнейшей задачей для дальнейшего развития этой теории является выяснение возможностей применения результатов на практике. Некоторую помощь в этом может оказать реализованная здесь методика графического изображения вероятностных состояний алгоритма. Кроме этого удачным продолжением могло бы быть детальное исследование мутации с немонотонными оценками.

Литература

[1] **Борисовский П.А.** Разработка генетического алгоритма для задачи о вершинном покрытии. Тезисы докладов XXIII научной студенческой конференции, посвященной 25-летию ОмГУ.-Омск: ОмГУ, 1999.

[2] Гэри М., Джонсон Д.: Вычислительные машины и труднорешаемые задачи. М.:Мир, 1982.

[3] **Еремеев А.В.** Разработка и анализ генетических и гибридных алгоритмов для решения задач дискретной оптимизации: Автореф. дис. канд. физ.-мат. наук. - Омск, 2000.

[4] **Пападимитриу Х.,Стайглиц К.** Комбинаторная оптимизация. Алгоритмы и сложность. М.Мир,1985.

[5] **Aggarval, C.C., J.B. Orlin, and R.P. Tai** An Optimized Crossover for Maximum Independent Set. *Operation Research* 45:226-234,1997.

[6] Arora, S., Lund, C.: Hardness of approximations. Approximation Algorithms for NP-Hard Problems. Ed. by S.D.Hochbaum. PWS Publishing Company, (1995) 399–446

[7] Ausiello, G., Protasi, M.: Local search, reducibility and approximability of NP-optimization problems. Information Processing Letters, **54** (1995) 73–79

[8] **T.Bäck and S.Khuri** An evolutionary heuristic for the minimum vertex cover problem. In J.Kunze and H.Stoyan, editors, KI'94 Workshops - Extended Abstracts, pages 83-84. Gesellschaft for Informatik e.V., Bonn, 1994.

[9] Bäck, T., Schwefel H.-P.: An overview of evolutionary algorithms for parameter optimization, Evolutionary Computation, **1** (1) (1993) 1–23

[10] **E. Balas and W. Niehaus** Optimized Crossover-Based Genetic Algorithms for the Maximum Cardinality and Maximum Weight Clique Problems // Journ. of Heuristics. - 1998. - Vol. 4, N 4, - P.107-122.

[11] J.E. Beasley, P.C. Chu: A Genetic Algorithm for the Set Covering Problem. *European J. Oper. Res.* V.94, N 2, 293-300,1996.

[12] P.A.Borisovsky, A.V. Eremeev.: On Performance Estimates for Two Evolutionary Algorithms. In E.J.W.Boers et al. (Eds.) Applications of evolutionary computing: Proceedings of EvoWorkshops 2001. LNCS Vol. 2037, 2001. pp. 161-171. c Springer-Verlag.

[13] Eremeev A.V.: Modeling and Analysis of Genetic Algorithm with Tournament Selection. Proc. of Artificial Evolution Conference (AE'99). Lecture Notes in Computer Science, Vol. 1829. Springer Verlag (2000) 84–95

[14] M.M. Halldórsson, J. Radhakrishnan: Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica*, 18:145-163, 1997.

[15] D.S. Hochbaum: Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Applied Mathematics*, 6:243-254, 1983.

[16] Johnson D.S., Trick M.A. Introduction to the Second DIMACS Challenge: Cliques, Coloring, and Satisfiability// Cliques, Coloring, and Satisfiability / Ed. by D.S. Johnson and M.A. Trick. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Vol. 26. - AMS, 1996. -P.1-10.

[17] K.Ko: Some observations on the probabilistic algorithms and NP-hard problems, Information Processing Letters, **14** (1982) 39–43

[18] Motwani, R. and Raghavan, P.: Randomized Algorithms. Cambridge University Press (1995)

[19] G.L. Nemhauser, J.L.E. Trotter: Vertex packing: structural properties and algorithms. *Math. Programming*, 8:232-248,1975.

[20] Rechenberg I.: Evolutionsstrategie'94. Formann-Holzboog Verlag, Stuttgart (1994)

[21] Rudolph, G.: Finite Markov Chain Results in Evolutionary Computation: A Tour d'Horizon, Fundamenta Informaticae **35** (1-4) (1998) 67–89