ПОВЫШЕНИЕ ВЫЧИСЛИТЕЛЬНОЙ ЭФФЕКТИВНОСТИ ФОРМАЛЬНЫХ ЭВОЛЮЦИОННО-ГЕНЕТИЧЕСКИХ ПРЕОБРАЗОВАНИЙ

Слепцов Николай Владимирович,
к.т.н., доцент кафедры ЭиОП
Пензенского государственного университета
05.13.17
nbs_nbs@km.ru

Аннотация. ***Рассмотрено применение в генетических алгоритмах инкрементного преобразования объединения, обобщенного преобразования разбиения для оптимального с точки зрения вычислений представления данных. Преобразования упрощают обобщение ряда существующих теорий ГА и дают возможность использования с помощью алгоритмов быстрых вычислений.***

Ключевые слова: Генетические алгоритмы, моделирование, представление данных, преобразования.

THE INCREASING OF COMPUTABILITY FOR THE FORMAL GENETIC TRANSFORMATIONS

Sleptsov Nikolai Vladimirovich
Ph.D. of technical Sciences,
assistant professor of department EiOP,
Penza State University

Abstract. ***The possibilities of multary and incremental transforms in genetic algorithms are discussed for the best in terms of computing data. Transforms simplifies compilation of some existing theories of GA and give the possibility to use with fast algorithms for computing.***

Keywords: Genetic algorithms, simulation, data representation, transforms.

Введение

В ряде приложений, характеризующихся сочетанием значительных вычислительных затрат с нестрогой формулировкой конечной цели вычислений и/или границ процесса, активно используются средства, относящиеся к классу эволюционных вычислений и эволюционной информатики [1,2].

Наиболее известными из группы эволюционных вычислений являются генетические алгоритмы (ГА) – поисковые алгоритмы, основанные на механизмах натуральной селекции и натуральной генетики. Они реализуют принцип «выживания наиболее приспособленных» среди анализируемых структур, формируя и изменяя поисковый алгоритм на основе моделирования эволюции поиска. В очередной генерации новое множество особей - искусственных последовательностей, кодирующих решение исследуемой проблемы, создается путем использования части старых и добавления новых частей со свойствами, являющихся по каким – то параметрам предпочтительными. Базисным отличием ГА от алгоритмов случайного поиска является эффективное использование информации, накопленной в процессе эволюции.

Формальное описание ГА

Формально генетический алгоритм можно представить как

$$ГА = (P^0, \lambda, l, s, p, f, t),$$ где

$P^0 = (\alpha_1^0, .., \alpha_\lambda^0)$ – исходная популяция, α_i^0 – решение задачи, представленное в виде хромосомы; λ – размер популяции; l – длина каждой хромосомы популяции, целое число; s – оператор отбора; p – отображение, определяющее рекомбинацию (кроссовер, мутация, инверсия, сегрегация…); $\boldsymbol{f}$ – функция пригодности (оценки); $\boldsymbol{t}$ – критерий остановки.

Оператор отбора $\boldsymbol{s}$ порождает промежуточную популяцию $\hat{P}^t$ из популяции P^t посредством отбора и генерации новых копий элементов P^t : $\hat{P}^t = s(P^t)$. Функция пригодности f используется для отбора конкурентоспособных особей популяции и обеспечивает обратную связь от результатов, полученных в течение поколения t.

Поскольку в большинстве реализаций ГА изменения зависят только от состава имеющейся популяции и не зависят от того, каким образом данная популяция получена, для моделирования ГА принципиально очень удобен формализм марковских цепей. Но такое представление не обеспечивает возможности эффективной организации эволюционного поиска, в частности, помимо очевидной затратности представления матриц преобразования в явном виде, при таком подходе проблематично описание целей эволюционного моделирования и не приходится говорить о конструктивности процедуры поиска решения. Более того, при этом скрываются существенные моменты, обеспечивающие эффективность эволюционно-генетического поиска [4].

Чем обеспечивается эффективный поиск при случайных базовых манипуляциях ГА – случайном обмене и изменении подстрок? Рассмотрим не строки в целом, кодирующие конкретные решения, а способ сохранения внутри строк отображений аллелей. Отображения аллелей называются шаблонами, они - просто строки, которые обрабатывает ГА, но с дополнительным отличительным признаком “#”, разрешенным в каждой позиции. Это групповой символ, означающий «не имеет значения». Так, шаблонами для задачи представления <3,2,2> являются # # #, # # 0, # 1 #, 2 # #, # 0 0, 0 # 1 и 2 1 0.

Шаблоны определены двумя величинами. Порядок шаблона - число определенных в нем аллелей (число отличных от # символов). Разрешение (длина разрешения) - расстояние между первой и последней определенной аллелью. Данные величины определяют поведение шаблона при рекомбинации. Порядок определяет, насколько вероятно шаблон будет разрушен при мутации: большее количество аллелей означает большую вероятность мутации одной из них. Разрешение определяет, с какой вероятностью шаблон может быть разрушен операцией кроссовера: чем ни больше размер шаблона, тем больше вероятность того, что кроссовер произойдет в шаблоне.

Для эффективной организации генетического поиска было бы существенным выделение некоторого основного множества шаблонов, обработка которых вызывает неявную обработку других шаблонов. Такие шаблоны носят название ключевых шаблонов и они являются точно теми шаблонами, которые *должны* быть обработаны для всех обрабатываемых шаблонов. Обработка множества ключевых шаблонов может привести к свободной обработке всех других шаблонов. В качестве примера возьмем простейшую проблему представления <2>. Если известны средние оценки пригодности популяции, наблюдаемая пригодность шаблона 1 и удельный вес шаблона 1 в популяции, тогда мы можем вычислить ожидаемый удельный вес шаблона 1 в следующем поколении. Это вычисление – основа обработки шаблона, которую ГА делает неявно. Существенным является то, что обработка шаблона 1 – также обработка шаблона 0 в качестве побочного эффекта.

Применение ключевых шаблонов означает, что имеется поддающееся трактовке представление об обработке шаблонов, поскольку в этом случае можно ограничиться рассмотрением ключевых шаблонов; остальные ситуации обрабатываются автоматически как побочный эффект. Вследствие важности этого момента встает вопрос о методе, позволяющем обеспе-

чить рассмотрение разных аспектов ГА с точки зрения различных ключевых шаблонов.

Преобразования в ГА

Рассмотрим преобразования, которые являются методами повышения эффективности для изменения целей от с*трокового представления,* где ключевые шаблоны являются только строками к *представлению шаблонов,* при котором максимизируется число # в наборе ключевых шаблонов.

Двоичное *кодирование* или *представление* задачи для ГА традиционно означает, что решение задачи представляется в виде двоичных строк фиксированной длины. Недвоичное кодирование предполагает представление задачи строками фиксированной длины, содержащих аллели из дискретных генов, из которых все или часть не являются двоичными, т.о. двоичное представление не является подмножеством недвоичного. Термин «недвоичное кодирование» применяется для того, чтобы подчеркнуть отличие данного представления от представления двоичных генов. Для точной характеристики вводится термин «обобщенное», или «унифицированное представление», оно применяется для описания генов, являющихся множествами величин строк фиксированной длины и задач, основанных на представлении таких генов.

Поскольку нет явных причин предпочесть один способ кодирования другому, для обеспечения единой основы представления можно предложить введение единого унифицированного способы кодирования задачи для реализации ее средствами ГА.

Произведение Кронекера. Произведение Кронекера (прямое или тензорное произведение, ПК) применяется для построения матрицы из двух меньшего размера путем использования правой матрицы в качестве «строительного блока» и масштабирования ее элементами левой матрицы.

$$A \otimes B = \begin{bmatrix} A_{1,1}B & A_{1,2}B & A_{1,n_A^{\downarrow}}B \\ A_{2,1}B & A_{2,2}B & A_{2,n_A^{\downarrow}}B \\ & & \\ A_{n_A^{\rightarrow},2}B & A_{n_A^{\rightarrow},2}B & A_{n_A^{\rightarrow},n_A^{\downarrow}}B \end{bmatrix}$$

где A – матрица $n_A^{\rightarrow} \times n_A^{\downarrow}$, B – матрица $n_B^{\rightarrow} \times n_B^{\downarrow}$, $A \otimes B = n_A^{\rightarrow} n_B^{\rightarrow} \times n_A^{\downarrow} n_B^{\downarrow}$. Из определения произведения Кронекера следует, что применив индексирование от 0, а не от 1, имеем:

$$(A \otimes B)_{in_B^{\rightarrow}+i', in_B^{\uparrow}+j'} = A_{i,j} B_{i',J}$$

r* – матрицы, *r* – вектора и операции над ними.** Представлением задачи или просто представлением назовем вектор ***c целых чисел, такой, что $\forall_i c_i \geq 2$. Размер представления задачи – произведение его элементов $\prod \langle \ \rangle = 1$.

Элементы представления задачи - характеристики генов. Они определяют число допустимых символов в каждом локусе строки, т.е. характеристику гена в этом локусе. Далее считаем множество $G_i = \{0,1,2,...c_i - 1\}$ геном для локуса i и таким образом представление задачи становится эквивалентно набору строк.

ПК можно применить для формирования строк и представлений, используя операцию конкатенации **<>:** $П(c <> c') = ПcПc'$.

При дальнейшем рассмотрении было бы желательно иметь возможность индексировать матрицы не числами, а строками. Для этой цели с матрицей свяжем представление задачи, которое точно определяет, какие строки могут быть использованы в качестве индекса матрицы. Такая пара матрица – представление называется *r-матрицей*.

Определение О1. ***r*-матрица** – двойка $< M,c >$, где M – матрица размером $n \times n$ и представление задачи c размера n, называемое основным или базовым представлением

Определение О2. ***r*-вектор** – это двойка $< \nu,c >$, включающая вектор ν, вида $n \times 1$ или $1 \times n$, и базовое представление задачи ***c*** размера ***n***.

Для индексации строк или столбцов r-матрицы (r-вектора) с помощью отдельных строк применяется функция отображения строк в числа *idx()*.

Индексация r-матриц строками. Значение r-матрицы *(M,c)*, индексированной строками *s* и *t*, определяется как

$$(M,c)_{s,t} = M_{idx(s,c),idx(t,c)}$$

Определим *idx()*. Для преобразования строки в число упорядочим все строки, положение строки в упорядоченной последовательности даст уникальное значение.

Определение О3.

$$idx(\langle\ \rangle\langle\ \rangle) = 0$$
$$idx(s_* :: s, c_* :: c) = s_* \Pi c + idx(s,c)$$

Определение дает возможность проводить декомпозицию векторов.

Рассмотрим два преобразования – преобразование объединения - инкрементной комбинации (IC) и преобразование обобщенного разбиения – MP. Они являются методами изменения «перспективы» рассмотрения изменением рассматриваемых множеств ключевых шаблонов. Например, мы можем рассматривать оценку пригодности или пропорции изменения популяции, либо же некоторую комбинацию этих двух элементов. Для абстрагирования от таких деталей можно предположить, что имеется некоторая величина, связанная с каждой строкой, которую мы будем называть *значением строки*. С точки зрения шаблонов нас будет интересовать значение шаблона, которое является величиной значения строк в строках шаблонов.

Одним из способов изменить перспективу рассмотрения является переход от значений строк к значениям шаблонов ключевого шаблона. Это можно осуществить следующим образом: матрицы, преобразующие значения строк в значения ключевых шаблонов, не содержащих нулей, являются ***k*** – матрицами с матрицами – ядрами следующего вида:

$$K_{<C_*>} = \begin{bmatrix} 1/c_* & 1/c_* & ... & 1/c_* \\ 0 & 1 & ... & 0 \\ : & : & ... & : \\ 0 & 0 & ... & 1 \end{bmatrix}$$

Вместо рассмотрения абсолютных значений шаблонов следует рассматривать относительные комбинации. Шаблон #12# является комбинацией шаблонов #1## и ##2#. Первым приближением было бы вычислить значение шаблона #12# как z(#1##)+z(##2#). Естественно, между этими двумя одноаллельными шаблонами могут наблюдаться взаимные воздействия. Относительное значение шаблона #12# - это дополнительная величина, которую нужно добавить к указанному первому приближению, чтобы получить действительное значение шаблона #12#, таким образом, использование относительных величин уменьшает уровень взаимодействия между двумя одноаллельными шаблонами.

Значения одноаллельных шаблонов берутся относительно значения ####, так действительное значение шаблона #12# будет $e_{\#\#\#\#} + e_{\#1\#\#} + e_{\#\#2\#} + e_{\#12\#}$, где e_s – относительное значение шаблона, которое можно назвать коэффициентом разбиения.

Обобщение двоичных коэффициентов

Коэффициенты разбиения дают возможность вычислить значения строки или шаблона по вкладу аллелей [3]. Их удобно рассматривать как «вклад» отдельных аллелей, например:

$$z_{11} = e_{\#\#} + e_{1\#} + e_{\#1} + e_{11}$$

$e_{\#\#}$ - среднее значение всех строк, $e_{1\#}$ - дополнительное значение, которое в среднем строка дает при наличии 1 в первой аллели, $e_{\#1}$ - соответственно, во второй.

В двоичном случае налицо симметрия. Если строка не содержит в локусе 1, она должна содержать 0, т.е. вклад, который строка будет иметь от наличия 0, к примеру, в первом

локусе, равен потерям от отсутствия там 1, т.е. $e_{1\#.}$

$$z(10\#) = e\#_{\#\#} + e1\#_{\#} - e_{\#1\#} - e1_{1\#}$$

В недвоичном случае такая симметрия нарушается. Мы должны явно указать, какую аллель следует использовать или подразумевать при отсутствии других. Можно предположить, что подразумеваемой аллелью является 0, поскольку такой выбор объединяет выбор ключевых шаблонов. Используя такие значения для коэффициентов, можно записать уравнения, связывающие коэффициенты разбиения и значения строк для задачи, представленной геном с характеристикой 4:

$z_0 = e\# - e_1 - e_2 - e_3$

$z_1 = e\# + e_1$

$z_2 = e\# + e_2$

$z_3 = e\# + e_3$

где e_1 – вклад 1, e_2 -, соответственно, вклад 2, e_3 - 3, а вклад 0 является вкладом ситуации отсутствия 1,2,3.

Инверсное унифицированное преобразование разбиения. Обозначим матрицу обратного преобразования как E^{-1} и, следуя подходу, изложенному для *r*- матриц, свяжем ее с базовым представлением задачи. В качестве примера возьмем уравнения:

$$z_{<4>} = E^{-1}_{<4>e<4>}$$

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & -1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} e_\# \\ e_1 \\ e_2 \\ e_3 \end{bmatrix}$$

Каждая строка матрицы преобразования соответствует вычислению значения строки базового преобразования:

Любая матрица $E^{-1}_{<c_*>}$ для одного генома будет представлять матрицу размером $c_* \times c_*$ с 1 в первой столбце (поскольку каждое вычисление требует коэффициент «среднего значения»), 1 вдоль главной диагонали и –1 в верхнем ряду со второй позиции (вычитая коэффициенты для подразумеваемой аллели) Аналогично:

$$E^{-1}_{<c_*>} = \left[\begin{array}{c|c} 1 & \begin{matrix} -1 \\ I_{c_*-1} \end{matrix} \end{array}\right],$$

где $\boldsymbol{I}_n$ – единичная матрица n*n. Далее рассмотрим $E^{-1}_{<3,2,2>}$ и, анализируя ее структуру, выведем общую формулу для $E^{-1}_{<c>}$.

$E^{-1}_{<3,2,2>} =$

1	−1	−1	1	−1	1	1	−1	−1	1	1	−1
1	1	−1	−1	−1	−1	1	1	−1	−1	1	1
1	−1	1	−1	−1	1	−1	1	−1	1	−1	1
1	1	1	1	−1	−1	−1	−1	−1	−1	−1	−1
1	−1	−1	1	1	−1	−1	1	0	0	0	0
1	1	−1	−1	1	1	−1	−1	0	0	0	0
1	−1	1	−1	1	−1	1	−1	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0
1	−1	−1	1	0	0	0	0	1	−1	−1	1
1	1	−1	−1	0	0	0	0	1	1	−1	−1
1	−1	1	−1	0	0	0	0	1	−1	1	−1
1	1	1	1	0	0	0	0	1	1	1	1

Рис. 1

Для отображения структуры использовано выделение фоном. На «верхнем» уровне матрица состоит из 3 типов одинаковых блоков: все нули, блок со светло-серым фоном и блок с темно-серым фоном. Последний является полностью аналогичным блоку со светло-серым фоном, но для всех элементов изменен знак. Можно записать:

$$E^{-1}_{<3,2,2>} = \begin{bmatrix} B & -B & -B \\ B & B & 0 \\ B & 0 & B \end{bmatrix},$$

что с точки зрения структуры очень близко к $E^{-1}_{<3>}$. Аналогично проанализируем структуру блока *B*:

$$B = \left[\begin{array}{cc|cc} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ \hline 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{array}\right] = \begin{bmatrix} B_2 & -B_2 \\ B_2 & B_2 \end{bmatrix}$$

Рис. 2

что с точки зрения структуры подобно $E^{-1}_{<2>}$.

Иными словами, крайний левый ген (характеристика 3) управляет *глобальной* структурой матрицы, используя $E^{-1}_{<3>}$ как шаблон, в котором размещены структуры низкого уровня. Второй ген управляет промежуточными структурами, используя как шаблон $E^{-1}_{<2>}$, и крайний правый ген управляет локальной структурой, используя $E^{-1}_{<2>}$.

ПК строит матрицы точно так же. Используя его, можно записать:

$$E^{-1}_{<3,2,2>} = E^{-1}_{<3>} \otimes E^{-1}_{<2>} \otimes E^{-1}_{<2>},$$

или в общем случае

$$E^{-1}_{<l,m,n>} = E^{-1}_{<l>} \otimes E^{-1}_{<m>} \otimes E^{-1}_{<n>}$$

Сравнивая это равенство с определением *k*-матрицы, легко увидеть, что E^{-1} является *k*-матрицей.

Унифицированное преобразование разбиения. Матрица унифицированного преобразования разбиения E_c – это *k*-матрица, получаемая из матриц – ядер следующей формы:

$$E_{<c_*>} = \frac{1}{c_*}\begin{bmatrix} 1 & 1 & 1 & ... & 1 \\ -1 & c_*-1 & -1 & ... & -1 \\ -1 & & c_*-1 & ... & -1 \\ : & : & : & : & : \\ -1 & -1 & -1 & ... & c_*-1 \end{bmatrix}$$

Такая матрица-ядро подсчитывает коэффициенты для единого гена. Можно неформально рассмотреть этот процесс. Первая строка суммирует все величины и делит на c_*. – их число. Эта величина – коэффициент «средней величины» – $e_{\#}$. Все остальные строки являются отрицанием первой строки и содержат единственную дополнительную величину e_i. Она соответствует переопределению $z_i = e_{\#} + e_i$, что дает $e_i = z_i - e_{\#}$.

Быстрые преобразования. Выполнение преобразований разбиения и инкрементного объединения (IC) с помощью прямого умножения матриц требует порядка $O(n^2)$ операций (n – размер матриц преобразований). Если предположить, что можно реализовать выполнение матричных преобразований «на лету», без временных издержек, нам будет необходимо $2n$ устройств памяти: n для вектора значений строк и n для вектора коэффициентов.

Предлагаемый механизм обработки использует преимущества тождественных вычислений при матричных преобразованиях. Например, рассмотрим матрицу для $E^{-1}_{<2,2>}$:

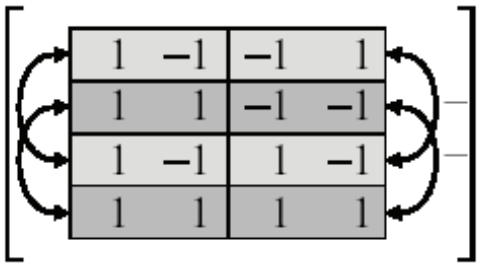

Рис. 3

Стрелки показывают идентичные вычисления. Вместо 3*4=12 операций сложение\вычитания необходимо две операции по столбцу и четыре операции на объединение столбцов, что дает восемь операций.

Идентичные вычисления реализуются с помощью ПК. Матрица, формируется из одинаковых блоков (прямоугольное масштабирование), таким образом для каждого столбца требуется выполнение вычислений только в одном блоке. Преобразование инкрементного объединения и преобразование разбиения в общем случае образуются из ряда операций ПК. Поэтому формируются матрицы X_i, для которых выполняются только необходимые вычисления для каждого столбца блоков на каждом «уровне» структуры в матрице. Подобная организация быстрого алгоритма может применяться к любой матрице, сформированной с применением ПК.

Оценка числа операций. Матричное умножение включает операции умножения и суммирования, при правильной организации одно умножение исключает проведение этих операций для нулевых элементов в M и исключает проведение умножения для элементов со значениями 1 и –1. Поэтому для простоты будем оценивать операции сложения и вычитания, учитывая наличие нулевых элементов. Получены следующие оценки:

- для инверсного преобразования

$$ops(E_c^{-1}) \approx \prod_{i=1}^{l} 3c_i = O(3^{\log_{\hat{c}} n_s} n_s$$

- для быстрого преобразования:

$$ops(\prod X_i = O(n_s \log_{\hat{c}} n_s)$$

- для унифицированного преобразования:

$$ops(E_{<c_*>}) = 2c_* - 2.$$

Таким образом, унифицированное преобразование разбиения и его инверсия имеют одинаковую оценку сложности для организации вычислений.

Осуществление быстрого преобразования

Для осуществления быстрого преобразования мы должны обеспечить умножение матриц $\boldsymbol{X_i}$.

Каждая из матриц $\boldsymbol{X_i}$ может быть жестко кодирована. Рассмотрим структуру матриц $\boldsymbol{X_i}$. В качестве примера рассмотрим $\boldsymbol{X_2}$ для инверсии преобразования разбиения для представления (задачи) <3,2,2>. Исходная формула:

$$X_2 = I_{\gamma(1,1)} \otimes E^{-1}_{<2>} \otimes I_{\gamma(3,3)} = I_3 \otimes E^{-1}_{<2>} \otimes I_2,$$

а матрица имеет вид:

1	0	−1	0	0	0	0	0	0	0	0	0
0	1	0	−1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	−1	0	0	0	0	0
0	0	0	0	0	1	0	−1	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	−1	0
0	0	0	0	0	0	0	0	0	1	0	−1
0	0	0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	0	0	1	0	1

Рис. 4

В формуле для X_2 левая единичная матрица (I_3) управляет общей структурой и, соответственно, вдоль главной диагонали матрицы X_2. расположены 3 блока. Правая единичная матрица управляет локальной структурой, что можно показать как «вырезание» матрицы $E^{-1}_{<2>}$ и копирование ее вдоль главной диагонали (вырезанные матрицы $E^{-1}_{<2>}$ показаны затененными прямоугольниками). По формуле для X_i мы должны ожидать $g(1, i-1)$ блоков, содержащих $g(i+1, l)$ вырезанных матриц $E^{-1}_{<c_i>}$. Расстояние между элементами каждой вырезанной матрицы равно $g(i+1, l)$.

При реализации процедур, обеспечивающих умножение матриц $E_{<c>}, E^{-1}_{<c>}, V_{<c>}, V^{-1}_{<c>}$ и учитывающих фактор «вырезания» и сдвига, умножение X_i вызовет циклически соответствующую процедуру с требуемыми параметрами. Эти параметры достаточно генерируются в процессе выполнения «на лету» и поэтому матрицу X_i нет необходимости явно вычислять, либо хранить.

Заключение

Рассмотрены возможности применения в генетических алгоритмах родственных недвоичных преобразований - инкрементного преобразования объединения (IC), обобщенного преобразования разбиения, и их инверсий.

IC – преобразование упрощает обобщение ряда существующих теорий ГА для унифицированного представления. Преобразование обобщенного разбиения вычисляет коэффициенты разбиения, необходимые для понимания свойств строк анализируемого шаблона, игнорируя избыточные подразумеваемые коэффициенты. Преобразование унифицированного представления дает возможность его применения в быстром преобразовании. Алгоритм быстрого преобразования с оценкой затратности *O(nlog n)* является критическим для использования преобразований, что несколько ограничивает применение коэффициентов разбиения в ГА-приложениях, несмотря на возможность их легкого интуитивного понимания.

Список литературы:

1. Букатова И.Л. Эвоинформатика: теория и практика эволюционного моделирования/ Букатова И.Л., Михасев Ю.И., Шаров А.М. – М, Наука, 1991
2. Редько В.Г. Эволюционная кибернетика. М.: Наука, 2001.
3. Слепцов Н.В. Математические модели генетических алгоритмов ПГУ,, тр. Международной конференции «Надежность и качество – 2007», Пенза-2007
4. Chambers Practical handbook of genetic algorithms v 3 Complex coding systems 2 ed, 2001
5. Koza J. Genetic programming: a paradigm for genetically breeding computer population of computer programs to solve problems. Cambridge, MA: MIT Press, 1992.