**Определения и понятия классического генетического алгоритма**

**1.** ***Генетические алгоритмы*** – это новая область исследований, которая появилась в результате работ Д.Холланда и его коллег. Генетические алгоритмы, описанные Д.Холландом, заимствуют в своей терминологии многое из естественной генетики. Далее будут приведены технические толкования терминов из биологии и генетики, которые используются в теории и практике генетических алгоритмов. Впервые генетические алгоритмы были применены к таким научным проблемам, как распознавание образов и оптимизация. Генетический алгоритм представляет собой адаптивный поисковый метод, основанный на селекции лучших элементов в популяции, подобно эволюционной теории Ч. Дарвина.

Основой для возникновения генетических алгоритмов послужили модель биологической эволюции и методы случайного поиска. Л. Растригин отмечал, что методы случайный поиск возник как реализация простейшей модели эволюции, когда случайные мутации моделировались случайными шагами оптимального решения, а отбор – «устранением» неудачных вариантов.

***Эволюционный поиск*** с точки зрения преобразования информации – это последовательное преобразование одного конечного нечеткого множества промежуточных решений в другое. Само преобразование можно назвать алгоритмом поиска, или генетическим алгоритмом. Генетические алгоритмы – это не просто случайный поиск. Они эффективно используют информацию, накопленную в процессе эволюции.

Цель генетических алгоритмов состоит в том, чтобы:

* абстрактно и формально объяснять адаптацию процессов в естественной системе и интеллектуальной исследовательской системе;
* моделировать эволюционные процессы для эффективного решения оптимизационных задач науки и техники.

В настоящее время используется новая парадигма решений оптимизационных задач на основе генетических алгоритмов и их различных модификаций. Генетические алгоритмы осуществляют поиск баланса между эффективностью и качеством решений за счет «выживания сильнейших альтернативных решений» в неопределенных и нечетких условиях.

Генетические алгоритмы отличаются от других оптимизационных и поисковых процедур следующим:

* работают в основном не с параметрами задачи, а с закодированным множеством параметров;
* осуществляют поиск не путем улучшения одного решения, а путем использования сразу нескольких альтернатив на заданном множестве решений;
* используют ***целевую функцию***, а не её различные приращения для оценки качества принятия решений;
* применяют не детерминированные, а вероятностные правила анализа оптимизационных задач.

Для работы генетических алгоритмов выбирают множество натуральных параметров оптимизационной проблемы и кодируют их в последовательность конечной длины в некотором алфавите. Они работают до тех пор, пока не будет выполнено заданное число генераций (итераций алгоритма) или на некоторой генерации будет получено решение определенного качества, или когда найден локальный оптимум, т.е. возникла преждевременная сходимость и алгоритм не может найти выход из этого состояния. В отличие от других методов оптимизации эти алгоритмы, как правило, анализируют различные области пространства решений одновременно и поэтому они более приспособлены к нахождению новых областей с лучшими значениями целевой функции.

**2.** Приведем некоторые понятия и определения из теории генетических алгоритмов. Все генетические алгоритмы работают на основании начальной информации, в качестве которой выступает популяция альтернативных решений *Р*. ***Популяция*** *Рt*={*P1, P2,…, Рi,…, РNp*} есть множество элементов *Рi* , *t*=0,1,2… - номер генерации генетического алгоритма, Np – размер популяции. Каждый элемент *Рi* этой популяции, как правило, представляет собой одну или несколько хромосом или особей, или индивидуальностей (альтернативных упорядоченных или неупорядоченных решений). Хромосомы состоят из генов *Рt*={*g1, g2,…, gv*} (элементы, части закодированного решения), и позиции генов в хромосоме называются лоции или локус для одной позиции, т.е. ген – подэлемент (элемент в хромосоме), ***локус*** – позиция в хромосоме, ***аллель*** – функциональное значение гена.

Гены могут иметь числовые или функциональные значения. Обычно эти числовые значения берутся из некоторого алфавита. Генетический материал элементов обычно кодируется на основе двоичного алфавита {0,1}, хотя можно использовать буквенные, а также десятичные и другие алфавиты. Примером закодированной хромосомы длины девять на основе двоичного алфавита может служить хромосома *Рt*=(001001101).

Элементы в генетических алгоритмах часто называют родителями. ***Родители*** выбираются из популяции на основе заданных правил, а затем смешиваются (скрещиваются) для производства «***детей***» (***потомков***). Дети и родители в результате генерации, т.е. одного цикла (подцикла) эволюции, создают новую популяцию. Генерация, то есть процесс реализации одной итерации алгоритма, называется поколением.

По аналогии с амии, происходящими в живой природе, в технике считают, что ***эволюция популяции*** – это чередование поколений, в которых хромосомы изменяют свои значения так, чтобы каждое новое поколение наилучшим способом приспосабливалось к внешней среде. Тогда общая генетическая упаковка называется ***генотипом***, а организм формируется посредством связи генетической упаковки с окружающей средой и называется ***фенотипом***.

Каждый элемент в популяции имеет определенный уровень качества, который характеризуется значением целевой функции (в литературе иногда называется функция полезности, приспособляемости или пригодности (fitness)). Эта функция используется в генетических алгоритмах для сравнения альтернативных решений между собой и выбора лучших.

Следовательно, основная задача генетических алгоритмов состоит в оптимизации целевой функции. Другими словами, генетические алгоритмы анализируют популяцию хромосом, представляющих комбинацию элементов из некоторого множества, оптимизируют целевую функцию, оценивая каждую хромосому. Генетические алгоритмы анализируют и преобразовывают популяции хромосом на основе механизма натуральной эволюции. Каждая популяция обладает наследственной изменчивостью. Это означает наличие возможностей случайных отклонений от наиболее вероятного среднего значения целевой функции. Отклонения описываются нормальным законом распределения случайных величин. При этом наследственные признаки закрепляются, если они имеют приспособительный характер, т.е. обеспечивают популяции лучшие условия существования и размножения.

Так же как процесс эволюции начинается с начальной популяции, так и алгоритм начинает свою работу с создания начального множества конкурирующих между собой решений оптимизационной задачи. Затем эти «родительские» решения создают «потомков» путем случайных и направленных изменений. После этого оценивается эффективность этих решений, и они подвергаются селекции. Аналогично естественным системам здесь действует принцип «выживания сильнейших», наименее приспособленные решения «погибают», а затем процесс повторяется вновь и вновь.

Традиционные оптимизационные алгоритмы для нахождения лучшего решения используют большое количество допущений, что расширяет класс задач, которые можно решать с помощью генетических алгоритмов. Согласно существующим исследованиям можно сказать, что генетические алгоритмы позволяют решать те проблемы, решение которых традиционными алгоритмами затруднительно.

Генетический алгоритм дает преимущества при решении практических задач. Одно из них – это адаптация к изменяющейся окружающей среде. В реальной жизни проблема, которая была поставлена для решения изначально, может претерпеть огромные изменения в процессе своего решения. При использовании традиционных методов все вычисления приходится начинать заново, что приводит к большим затратам машинного времени. При эволюционном подходе популяцию можно анализировать, дополнять и видоизменять применительно к изменяющимся условиям, для этого не требуется полный перебор. Другое преимущество генетических алгоритмов для решения задач состоит в способности быстрой генерации достаточно хороших решений.

**3.** При решении практических задач с использованием генетических алгоритмов обычно выполняют четыре предварительных этапа:

* выбор способа представления решения;
* разработка операторов случайных изменений;
* определение способов «выживания» решений;
* создание начальной популяции альтернативных решений.

Рассмотрим некоторые особенности выполнения этих этапов.

*На первом этапе* для представления решения в формальном виде требуется такая структура, которая позволит кодировать любое возможное решение и производить его оценку. Математически доказано, что не существует идеальной структуры представления, так что для создания хорошей структуры требуется анализ, перебор и эвристические подходы. Возможный вариант представления должен позволять проведение различных перестановок в альтернативных решениях. Для оценки решений необходимо определить способ вычисления целевой функции.

*На втором этапе* достаточно сложным является выбор случайного оператора (или операторов) для генерации потомков. Существует огромное число таких операторов. Существуют два основных типа размножения: половое и бесполое. При половом размножении два родителя обмениваются генетическим материалом, который используется при создании потомка. Бесполое размножение – это фактически клонирование, при котором происходят различные мутации при передаче информации от родителя к потомку. Модели этих типов размножения играют важную роль в генетических алгоритмах. В общем случае можно применить модели размножения, которые не существуют в природе. Например, использовать материал от трех или более родителей, проводить голосование при выборе родителей и т.п. Фактически нет пределов в использовании различных моделей, и поэтому при решении технических задач нет смысла слепо копировать законы природы и ограничиваться только ими.

Успех генетических алгоритмов во многом зависит от того, как взаимодействуют между собой схема представления, методы случайных изменений и способ определения целевой функции. Поэтому для определенного класса задач целесообразно использовать направленные методы.

В качестве примера рассмотрим два способа представления перестановок при решении оптимизационных задач. В первом случае будем использовать одного родителя (альтернативное решение) и получать одного потомка. Во втором случае используем двух родителей, случайно выберем точку перестановки и для образования потомка возьмем первый сегмент у первого родителя, а второй сегмент – у второго. Первый метод похож на бесполое размножение, а второй –на половое размножение. Стоить отметить, что если первый метод всегда генерирует реальное решение, то второй может генерировать недопустимые решения. При этом требуется «восстанавливать» допустимые решения перед их оценкой.

*На третьем из рассматриваемых этапов* задаются правила выживания решений для создания потомства. Существует множество способов проведения селекции альтернативных решений. Простейшее правило – это «выживание сильнейших», т.е. остаются только лучшие решения с точки зрения заданной целевой функции, а все остальные устраняются. Такое правило часто оказывается малоэффективным при решении сложных технических проблем. Иногда лучшие я могут происходить от худших, а не только от самых лучших. Тем не менее, логично использовать принцип:

***Чем лучше решение, тем больше вероятность его выживания***

Отметим, что принцип (от латинского «начало») – это:

* основное исходное положение какой-либо теории;
* внутренняя убежденность в чем-либо;
* основная особенность работы механизма, устройства и т.п.

На последнем предварительном этапе создается начальная популяция. При неполноте исходных данных о проблеме решения могут случайным образом выбираться из всего множества. Это реализуется генерацией случайных внутрихромосомных перестановок, каждая из которых представляет собой определенное решение. При создании начальной популяции рекомендуется использовать знания о решаемой задаче. Например, эти знания могут быть получены из опыта разработчика, существующих стандартов и библиотек алгоритмов решения задач данного класса.

***4. Эффективность генетического алгоритма*** – степень реализации запланированных действий алгоритма и достижение требуемых значений целевой функции. Эффективность во многом определяется структурой и составом её популяции. При создании начального множества решений происходит формирование популяции на основе четырех основных принципов:

* «одеяло» - генерируется полная популяция, включающая все возможные решения в некоторой заданной области;
* «дробовик» - подразумевает случайный выбор допустимых альтернатив из всей области решения данной задачи;
* «фокусировка» - реализует случайный выбор допустимых альтернатив из заданной области решения данной задачи;
* «комбинирование» - состоит в различных совместных реализациях первых трех принципов.

Отметим, что популяция обязательно является конечным множеством.

**Генетические операторы**

**1.** В каждой генерации генетического алгоритма хромосомы являются результатом применения некоторых генетических операторов.

***Оператор* –** это языковая конструкция, представляющая один шаг из последовательности действий или набора описаний алгоритма.

Генетический алгоритм состоит из набора генетических операторов.

***Генетический оператор*** по аналогии с оператором алгоритма – средство отображения одного множества на другое. Другими словами, это конструкция, представляющая один шаг из последовательности действий генетического алгоритма.

2. Рассмотрим основные операторы генетических алгоритмов.

***Оператор репродукции (селекция)*** – это процесс, посредством которого хромосомы (альтернативные решения), имеющие более высокое значение целевой функции (с «лучшими» признаками), получает большую возможность для воспроизводства (репродукции) потомков, чем «худшие» хромосомы. Элементы, выбранные для репродукции, обмениваются генетическим материалом, создавая аналогичных или различных потомков.

Существует большое число видов операторов репродукции. К ним относятся следующие:

* *Селекция на основе рулетки* – это простой и широко используемый в простом генетическом алгоритме метод. При его реализации каждому элементу в популяции соответствует зона на колесе рулетки, пропорционально соразмерная с величиной целевой функции. Тогда при повороте колеса рулетки каждый элемент имеет некоторую вероятность выбора для селекции. Причем элемент с большим значением целевой функции имеет большую вероятность для выбора.
* *Селекция на основе заданной шкалы.* Здесь популяция предварительно сортируется от «лучшей» к «худшей» на основе заданного критерия. Каждому элементу назначается определенное число и тогда селекция выполняется согласно этому числу.
* *Элитная селекция*. В этом случае выбираются лучшие (элитные) элементы на основе сравнения значений целевой функции. Далее они вступают в различные преобразования, после которых снова выбираются элитные элементы. Процесс продолжается аналогично до тех пор, пока продолжают появляться элитные элементы.
* *Турнирная селекция*. При этом некоторое число элементов (согласно размеру «турнира») выбираются случайно или направленно из популяции, и лучшие элементы в этой группе на основе заданного турнира определяются для дальнейшего эволюционного поиска.

Оператор репродукции считается эффективным, если он создает возможность перехода из одной подобласти альтернативных решений области поиска в другую. Это повышает вероятность нахождения глобального оптимума целевой функции. Выделяют два основных типа реализации оператора репродукции:

* случайный выбор хромосом;
* выбор хромосом на основе значений целевой функции.

При случайном выборе хромосом частота *R* образования родительских пар не зависит от значения целевой функции хромосом ** и полностью определяется численностью популяции *N:*

,

где β – коэффициент селекции, принимающей в зависимости от условий внешней среды значение 1÷4.

Другой способ реализации оператора репродукции связан с использованием значений целевой функции. Существует две основные стратегии. ***Стратегия*** – это оптимальный набор правил и приемов, которые позволяют реализовать общую цель, достигнуть глобальных и локальных целей решаемой задачи. В первой предпочтение отдается хромосомам с близкими и «лучшими» (наибольшими при максимизации и наименьшими – при минимизации) значениями целевой функции. Во второй – хромосомам, со значениями целевой функции, сильно различающимися между собой.

Для реализации первой стратегии с максимизацией целевой функции с вероятностью

 ()

выбирают разные хромосомы. Здесь ЦФ – целевая функция, ОР – это оператор репродукции, моделирующий естественный процесс селекции, Pr(ОР) – вероятность выбора хромосом для репродукции.

Вторая стратегия реализуется так: часть хромосом выбирается случайным образом, а вторая – с вероятностью на основе выражения ().

Если  < ЦФср, где ЦФср – среднеезначение целевой функции в популяции, то оператор репродукции моделирует естественный отбор. Выбор случайных и сильно различающихся хромосом повышает генетическое разнообразие популяции, что повышает скорость сходимости генетического алгоритма на начальном этапе оптимизации и позволяет в некоторых случаях выходить из локальных оптимумов.

**3.** Кроме описанных, существует большое число других методов селекции, которые можно условно классифицировать на три группы. К первой группе отнесем вероятностные методы. Ко второй – детерминированные методы. К третьей – различные комбинации методов из первой и второй групп. Построение новых операторов репродукции непрерывно продолжается.

**4.** Основной трудностью решения инженерных оптимизационных задач с большим количеством локальных оптимумов является ***предварительная сходимость алгоритмов***. Другими словами, попадание решения в один, далеко не самый лучший, локальный оптимум при наличии их большого количества. Различные методы селекции и их модификации как раз и позволяют в некоторых случаях решать проблему предварительной сходимости алгоритмов. Следует отметить, что исследователи генетических алгоритмов все более склоняются к мысли применять комбинированные методы селекции с использованием предварительных знаний о решаемых задачах и предварительных результатах.

**5.** Опишем теперь ***операторы кроссинговера (скрещивания). Оператор кроссинговера*** – это языковая конструкция, позволяющая на основе преобразования (скрещивания) хромосом родителей (или их частей) создавать хромосомы потомков. Существует огромное число операторов кроссинговера, так как их структура в основном и определяет эффективность генетических алгоритмов. Кратко рассмотрим основные операторы кроссинговера, известные в литературе, и их модификации.

**5.1. *Простой ( одноточечный)* *оператор кроссинговера.*** Перед началом работы одноточечного оператора кроссинговера определяется так называемая точка оператора кроссинговера, или разрезающая точка оператора кроссинговера, которая обычно определяется случайно. Эта точка определяет место в двух особях, где они должны быть «разрезаны». Например, пусть популяция *Р* состоит из хромосом *P1* и *P2*, которые выступают в качестве родителей, *Р=*{ *P1*, *P2*}. Пусть первый второй родители имеют вид *P1*: 11111, *P2*:00000. Выберем точку оператора кроссинговера между вторым и третьим генами в *P1*, *P2*. Тогда, меняя элементы после точки оператора кроссинговера между двумя родителями, можно создать два новых потомка. В нашем примере получим



Итак, одноточечный оператор кроссинговера выполняется в три этапа:

1. Две хромосомы А =  и В = ** выбираются случайно из текущей популяции.

2. Число *k* выбирается из{1,2,…,n -1} также случайно. Здесь L – длина хромосомы, *k*  - точка оператора кроссинговера (номер, значение или код гена, после которого выполняется разрез хромосомы).

3. Две новые хромосомы формируются из **А** и **В** путем перестановок элементов согласно правилу

*= ,*

* = .*

После применения оператора кроссинговера имеем две старые хромосомы и всегда получаем две новые хромосомы. Схематически простой оператор кроссинговера показывает преобразование двух хромосом и частичный обмен информацией между ними, использующий точку разрыва, выбранную случайно.

**5.2. *Двухточечный* *оператор кроссинговера.*** В каждой хромосоме определяются две точки оператора кроссинговер, и хромосомы обмениваются участками, расположенными между двумя точками оператора кроссинговер. Например:



Отметим, что точка оператора кроссинговер в двухточечном операторе кроссинговера также определяется случайно. Существует большое количество модификаций двухточечного оператора кроссинговера. Развитием двухточечного оператора кроссинговера является многоточечный или N-точечный оператор кроссинговера. Многоточечный оператор кроссинговера выполняется аналогично двухточечному, хотя большое число «разрезанных» точек может привести к потере «хороших» родительских свойств.

Пример трехточечного оператора кроссинговера:



Здесь точки оператора кроссинговера делят хромосому на ряд строительных блоков (в данном случае 4). Потомок  образуется из нечетных блоков родителя  и четких блоков родителя . Потомок  образуется соответственно из нечетных блоков родителя  и четких блоков родителя .

Тогда многоточечный оператор кроссинговера выполняется аналогичным образом.

**5.3. *Упорядоченный оператор кроссинговера.*** Здесь «разрезающая» точка также выбирается случайно. Далее происходит копирование левого сегмента  в . Остальные позиции в  берутся из  в упорядоченном виде слева направо, исключая элементы, уже попавшие в . Например:



Получение  может выполняться различными способами. Наиболее распространенный метод копирования левого сегмента из , а далее анализ  методом, указанным выше. Тогда имеем : (G A B E ⎢C D F H).

**5.4. *Частично-соответствующий оператор кроссинговера.*** Здесь также случайно выбирается «разрезающая» точка или точка оператора кроссинговера. Дальше анализируются сегменты (части) в обеих хромосомах и устанавливается частичное соответствие между элементами первого и второго родителей с формированием потомков. При этом правый сегмент  переносится в , левый сегмент  переносится в  с заменой повторяющихся генов на отсутствующие гены, находящиеся в частичном соответствии. Например:



Аналогично можно получить :



**5.5. *Циклический оператор кроссинговера.*** Циклический оператор кроссинговера выполняет рекомендации согласно циклам, которые существуют при установлении соответствия между генами первого и второго родителей. Например, пусть популяция **Р** состоит из двух хромосом: Р = { и }. Первый и второй родители и их потомок имеют вид:



При выполнении циклического оператора кроссинговера  заполняется, начиная с первой позиции, и копирует элемент с первой позиции . Элементу 1 в  соответствует элемент 5 в . Следовательно, сформирован первый путь в цикле (1,5). Элементу 5 в  соответствует элемент 4 в , откуда второй путь в первом цикле (1,5; 5,4). Продолжая далее, получим, что элементу 4 в  соответствует элемент 1 в . Следовательно, сформирован первый цикл (1,5; 5,4; 4,1). Согласно этому циклу элемент 5 переходит в пятую позицию , а элемент 4 – в четвертую позицию.

Сформируем теперь второй цикл. Элемент 2 в  соответствует элементу 3 в . Продолжая аналогично, получим второй (2,3; 3,9; 9,6; 6,8; 8,2) и третий (7,10; 10,7) циклы.

Следовательно, в  элемент 3 расположен во втором локусе, т.е. на второй позиции, элемент 9 – в третьем, элемент 6 – в девятом, элемент 8 – в шестом, элемент 2 – в восьмом, элемент 10 – в седьмом и элемент 7 – в десятом локусах. Циклический оператор кроссинговера и его модификации эффективно применяются для решения комбинаторно-логических задач, задач на графах и гиперграфах и других оптимизационных задач.

**5.6. *Универсальный оператор кроссинговера.*** В настоящее время он популярен для решения различных задач из теории расписаний. Вместо использования разрезающей точки (точек) в универсальный оператор кроссинговера вводят двоичную маску, длина которой равна длине заданных хромосом. Первый потомок получается сложением первого родителя с маской на основе следующих правил: (0+0=0, 0+1=1, 1+1=0). Второй потомок получается аналогичным образом. Для каждого элемента в ,  гены меняются, как показано на следующем примере:

 - маска

Маска может быть задана или выбирается случайно с заданной вероятностью или на основе генератора случайных чисел. При этом чередование 0 и 1 в маске происходит с вероятностью ≈ 50%. В некоторых случаях используется параметризированный универсальный оператор кроссинговера, где маска может выбираться с вероятностью для 1 или 0 выше, чем 50%. Такой вид маски эффективен, когда хромосомы кодируются в двоичном алфавите.

**5.7.** Для решения многих оптимизационных задач можно использовать некоторые классы алгоритмов, называемых ***«жадными»***. Такой алгоритм делает на каждом шаге локально оптимальный выбор, в надежде, что итоговое решение также окажется оптимальным. Это не всегда так, но для многих задач такие эвристические алгоритмы дают оптимальный результат. Говорят, что к оптимизационной задаче применим принцип «жадного» выбора, если последовательность локально–оптимальных («жадных») выборов дает оптимальное решение.

***«Жадный» оператор -*** это языковая конструкция, позволяющая создавать новые решения на основе частичного выбора на каждом шаге преобразования локально оптимального значения целевой функции.

Рассмотрим ***«жадный» оператор кроссинговера***. Он может быть реализован на двух и более хромосомах, а в пределе – на всей популяции. Приведем алгоритм «жадного» оператора кроссинговера на примере нахождения пути с минимальной или максимальной стоимостью на графе:

1. Для всех хромосом популяции вычисляется целевая функция (стоимость пути между вершинами графа). Выбирается заданное число родительских хромосом и случайным образом на одной из хромосом определяется точка «жадного» оператора кроссинговера.
2. В выбранной хромосоме для i-го гена, расположенного слева от точки «жадного» оператора кроссинговера, определяется значение частичной целевой функции. Например, это стоимость пути от выбранного гена к ближайшему, находящемуся справа гену. Аналогичные действия выполняются по определению стоимости пути от i-го гена во всех остальных хромосомах, выбранных для «жадного» оператора кроссинговера.
3. В хромосому «потомок» выбирают тот ген, у которого значение целевой функции выше (ниже) при максимизации (минимизации) значений целевой функции.
4. Процесс продолжается аналогично, до построения хромосомы «потомок». Если в процессе реализации возникает цикл или тупик, то выбираются нерассмотренные гены с лучшим значением целевой функции.

Например, пусть задан граф *G = (X,U), X = {a,b,c,d,e}* в виде матрицы. Построим популяцию Р, состоящую из трех родительских хромосом *Р = {P1,P2,P3},* где *P1* : *abcde; P2: bdeca; P3 : ebadc.* Элементы определяют стоимость пути между любыми двумя вершинами графа, а каждый ген в хромосоме кодируется номером вершины графа:



Согласно алгоритму выберем точку «жадного» оператора кроссинговера между генами *b* и *c* в хромосоме *P1*. Теперь выбор (*b – c)* даетзначение целевой функции, равное 4 ( см. матрицу), выбор (*b – d*) (в хромосоме *P2*) определяет целевую функцию, со значением 3, а выбор (*b - a*) (в хромосоме *P3*) определяет целевую функцию, равную 15. При минимизации целевой функции выберем путь (*b – d*). Продолжая, получим путь реализации «жадного» оператора кроссинговера (рис. ). Итак, хромосома потомка *Р′:bdcae* имеет суммарную целевую функцию, равную 18: 3+1+6+8=18, а целевая функция родителей для *P1* равна 15+4+1+9=29, для *P2* равна 3+9+10+6=28 и для *P3* равна 2+15+7+1=25.



Рис. Пример реализации «жадного» оператора кроссинговера

Стратегию «жадного» оператора кроссинговера можно выполнять различными способами. Следует отметить, что поиск специальных и специализированных операторов кроссинговера, ориентированных на решение заданного класса задач, продолжается.

Рассмотрим различные варианты выбора пары хромосом для скрещивания. Вероятность скрещивания *Рл,х*(ОК) лучших хромосом с худшими по значению целевой функции, должна уменьшаться при эволюции поколений: ОК – оператор кроссинговера.

Вероятность скрещивания лучших хромосом должна увеличиваться на последних этапах оператора кроссинговера для раскрепления желаемых признаков в хромосомах.

**5.8.** Существуют и другие методы выбора пар хромосом для скрещивания. Например, «близкое» родство, «дальнее» родство, выбор на основе кода Грея и т.д.

«Близкое родство». Здесь вероятность выбора хромосом, подлежащих скрещиванию, запишется так:

- для первой хромосомы *Рi*

, ( )

где  - вероятность выбора хромосомы на основе близкого родства;

- для второй хромосомы P*j*вероятность  задается пользователем.

Затем вычисляется ***Хэммингово расстояние*** dist(Pi,Pj) между выбранными хромосомами текущей популяции. Рассмотрим, например, две , каждая из которых задается конкретными значениями семи переменных: х1, х2, х3, х4, х5, х6, х7; пусть в первом случае ими будут 0, 1, 0, 1, 1, 0, 1, а во втором 1, 1, 0, 1, 0, 0, 0. Запишем данные совокупности значений х1, х2, … , х7 одну под другой:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| х1 | х2 | х3 | х4 | х5 | х6 | х7 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| -1 | 0 | 0 | 0 | 1 | 0 | 1 |

Вычислим разность между каждым элементом первой строки и соответствующим элементом второй строки. Запишем результаты под второй строкой. Теперь вычислим сумму «абсолютных значений» полученных результатов. Эта сумма равна 3. Говорят, что расстояние Хэмминга между указанными выше точками (или величинами) [0, 1, 0, 1, 1, 0, 1] и [1, 1, 0, 1, 0, 0, 0] равно 3:

⏐-1⏐+⏐1⏐+⏐1⏐=1+1+1=3

В общем случае рассмотрим две хромосомы (, , …, ) и (х1,х2, …, хn); обобщенным расстоянием Хэмминга между этими двумя хромосомами называется скаляр

π = | - х1| + | - х2| + …+| - хn |.

Хромосомы рекомендуются для скрещивания, если хэммингово расстояние между ними dist(Pi,Pj) > R, где R – радиус скрещивания, задаваемый лицом, принимающим решение, или определяется как ближайшее меньшее целое от разности значений целевых функций ЦФ(Pi) - ЦФ(Pj), деленное на два.

Вероятности  и  возрастают на конечных стадиях работы оператора кроссинговера.

«Дальнее» родство определяется условием

,

где  - вероятность выбора хромосом при «дальнем родстве на начальных стадиях работы генетических алгоритмов, с учетом особенностей вычисляется для первой и второй хромосомы.

Хромосомы Pi и Pj подлежат скрещиванию, если хэммингово расстояние между ними dist(Pi,Pj) > R. Вероятность  и  уменьшается на конечных стадиях поиска оптимального решения.

***Код Грея*** – это двоичный код, последовательные значения которого отличаются только одним двоичным разрядом. Код Грея может использоваться для хромосом, представленных в двоичном виде. Например:

0 – 0000

1 – 0001

2 – 0011

3 – 0010

4 – 0110

5 – 0111 и т.д.

Такое кодирование альтернативных решений позволяет решать вопросы «взбалтывания» популяции; оно эффективно на начальных стадиях генетического алгоритма.

Как следует из биологии, некоторые процессы преобразования популяции происходят толчками. Основой таких процессов являются точковые мутации. В генетическом алгоритме мутация необходима потому, что предотвращает потерю генетического материала. Точковые мутации не изменяют размера и строения хромосом, а изменяют расположение генов в хромосоме.

**5.9.** ***Оператор мутации*** - языковая конструкция, позволяющая на основе преобразования родительской хромосомы (или ее части) создавать хромосому потомка.

Оператор мутации обычно состоит из двух этапов:

1. В хромосоме *А=(a1, a2, a3, …, aL-2, aL-1, aL)* определяются случайным образом две позиции (например, *a2* и *aL-1*).
2. Гены, соответствующие выбранным позициям, переставляются, и формируется новая хромосома *А′* *=(a1, aL-1, a3, …, aL-2, a2 , aL)*.

Рассмотрим кратко основные операторы мутации. Простейшим оператором мутации является одноточечный. При его реализации случайно выбирают ген в родительской хромосоме и, обменивая его на рядом расположенный ген, получают хромосому потомка, например:



Здесь  - родительская хромосома ,  - хромосома-потомок после применения одноточечного оператора мутации.

При реализации ***двухточечного оператора мутации*** случайным или направленным образом выбираются две точки разреза. Затем производится перестановка генов между собой, расположенных справа от точек разреза, например:





Здесь  - родительская хромосома ,  - хромосома-потомок после применения двухточечного оператора мутации.

Развитием двухточечного оператора мутации является многоточечный (или n-точечный) оператор мутации. В этом случае происходит последовательный обмен генов, расположенных правее точек разреза друг с другом в порядке их расположения. Ген, расположенный правее последней точки разреза, переходит на место первого, например:





***Строительные блоки*** – это тесно связанные между собой гены (части альтернативных решений), которые нежелательно изменять при выполнении генетических операторов. Из строительных блоков (как из кирпичиков при построении дома) можно создавать альтернативные оптимальные или квазиоптимальные решения.

В частном случае, когда строительные блоки, расположенные между точками разреза, одинаковы, многоточечный оператор мутации выполняется следующим образом. При четном числе точек разреза меняются местами гены, расположенные справа и слева от выбранных точек, например:





При нечетном числе точек потомок получается после обмена участками хромосом, расположенных между четными точками разреза, например:





Часто используют операторы мутации, использующие знания о решаемой задаче. Реализация таких операторов заключается в перестановке местами любых выбранных генов в хромосоме, причем точка или точки мутации определяются не случайно, а направленно.

В позиционном операторе мутации две точки разреза выбираются случайно, а затем ген, соответствующий второй точке мутации, размещается в позицию перед геном, соответствующим первой точке, например:





**5.10.** Введем понятие оператора инверсии. ***Оператор инверсии*** – это языковая конструкция, позволяющая на основе инвертирования родительской хромосомы (или ее части) создавать хромосому потомка. При его реализации случайным образом определяется одна или несколько точек разреза (инверсии), внутри которых элементы инвертируются.

Генетический оператор инверсии состоит из следующих шагов.

1. Хромосома *В=(b1, b2, …, bL,)* выбираются случайным образом из текущей популяции.
2. Два числа и выбираются случайным образом из множества {0,1,2,…,L+1}, причем считается, что y1=min{, } и y2=max{, }.
3. Новая хромосома формируется из *В* путем инверсии сегмента, который лежит справа от позиции y1 и слева от позиции y2 в хромосоме *В*. Тогда после применения оператора инверсии получаем:

В = (b1, …, by1, by2-1, by2-2, …, by1+1, by2, …, bL).

Для одноточечного оператора инверсии запишем





Здесь Р2 - родительская хромосома,  - хромосома-потомок после применения оператора инверсии.

Например, для двухточечного оператора инверсии получим





Здесь  - родительская хромосома,  - хромосома-потомок после применения двухточечного оператора инверсии.

Часто применяется специальный оператор инверсии. В нем точки инверсии определяются с заданной вероятностью для каждой новой создаваемой хромосомы в популяции.

**5.11.** Рассмотрим оператор транслокации. ***Оператор транслокации*** - это языковая конструкция, позволяющая на основе скрещивания и инвертирования из пары родительских хромосом (или их частей) создавать две хромосомы потомков. Другими словами, он представляет собой комбинацию операторов кроссинговера и инверсии. В процессе его реализации случайным образом производится один разрез в каждой хромосоме. При формировании потомка  берется левая часть до разрыва из родителя Р1 и инверсия правой части до разрыва из Р2. При создании  берется левая часть Р2 и инверсия правой части Р1, например:





Существует большое число других видов оператора транслокации. Отметим, что до последнего времени оператор транслокации не применялся в генетических алгоритмах, а также при разработке интеллектуальной искусственной системы и решении оптимизационных задач.

**5.12.** ***Оператор транспозиции*** - языковая конструкция, позволяющая на основе преобразования и инвертирования выделяемой части родительской хромосомы создавать хромосому потомка. Например:





Здесь три точки разреза. Точки разреза выбираются случайным или направленным образом. В родительской хромосоме P1 строительный блок *DEF* инвертируется и вставляется в точку разреза между генами *А* и *В*. В результате получаем хромосому-потомок . Отметим, что существует большое количество модификаций оператора транспозиции.

**5.13.** Рассмотрим ***оператор сегрегации***. Это языковая конструкция, позволяющая на основе выбора строительных блоков из хромосом родителей (или их частей) создавать хромосомы потомков.

Приведем один их примеров его реализации. Отметим, что оператор сегрегации, как правило, реализуется на некотором наборе хромосом. Пусть имеется популяция *Р*, состоящая из четырех родительских:

Р = {P1, P2, P3, P4} P1 : (12345678); P2 : (24316587); P3 : (31425768); P4 : (87654321).

В каждой хромосоме выделим строительные блоки. Выделим по два строительных блока: в P1 - 23 и 67, в P2 - 1 и 4, в P3 – 768 и 425 и в P4 – 54 и 87. Тогда, например, потомок  можно сформировать, взяв первые строительные блоки из каждой родительской хромосомы популяции. Так, вариант  будет представлен последовательностью (23176854) и является реальным решением, а вариант  (67442587) является нереальным (недопустимым). Очевидно, этот можно реализовать различными способами в зависимости от выборки строительных блоков или генов из хромосом.

5.14. Опишем ***оператор удаления.*** Это языковая конструкция, позволяющая на основе удаления строительных блоков из хромосом родителей (или их частей) создавать хромосомы потомков.

При его реализации направленным или случайным образом определяется точка или точки разреза. Далее производится пробное удаление генов или их строительных блоков с вычислением изменения значения целевой функции. Элементы, расположенные слева от точки оператора удаления или между двумя точками, удаляются из хромосомы. При этом производится преобразование потомка таким образом, чтобы соответствующее альтернативное решение оставалось реальным.

**5.15.** Опишем ***оператор вставки.*** Это языковая конструкция, позволяющая на основе вставки строительных блоков в хромосомы родителей создавать хромосомы потомков.

При его реализации направленным или случайным образом создается хромосома (донор), состоящая из строительных блоков, которые желательно разместить в другие хромосомы популяции. После этого направленным или случайным образом определяется хромосома для реализации оператора вставки. В ней находится точка или точки разреза. Затем анализируются другие хромосомы в популяции для определения альтернативных вставок. Далее производится пробная вставка строительных блоков с вычислением изменения значения целевой функции и получений реальных решений. Новые строительные блоки вставляются в хромосому справа от точки оператора вставки или между его двумя точками. Отметим, что оператор удаления и оператор вставки могут изменять размер хромосом. Для сохранения постоянного размера хромосом эти операторы можно применять совместно.

На конечном этапе поиска целесообразно применять выбор близких решений в соответствии с определенным критерием, т.е. искать решение среди лучших .

**5.16. *Оператор редукции*** - языковая конструкция, позволяющая на основе анализа популяции после одной или нескольких поколений генетического алгоритма уменьшить ее размер до заданной величины. Рассмотрим способы реализации оператора редукции. Он выполняется для устранения неудачных решений. В некоторых генетических алгоритмах, в частности, в простом генетическом алгоритме, этот оператор применяется для сохранения постоянного размера начальной популяции. Основная проблема здесь – нахождение компромисса между разнообразием генетического материала и качеством решений. Сначала формируют репродукционную группу из всех решений, образовавшихся в популяции , затем выполняют отбор решений в следующую популяцию.

Численность новой популяции

 = ++++++++,

где

 - численность новой популяции,

 - численность популяции на предыдущем шаге (поколении) t,

, , , , , , ,  - потомки, полученные в результате применения операторов кроссинговера, мутации, инверсии, транслокации, транспозиции, сегрегации, удаления, вставки.

Отметим, что оператор редукции может применяться после каждого оператора или после всех в одной генерации генетического алгоритма. Выделяют две основных схемы редукции (иногда их называют схемы отбора).

1*. Элитная схема редукции*. В группу удаления из популяции включаются такие хромосомы, как  и только те потомки, для которых выполняется условие:

()()(ЦФ() > ЦФ()), где k = ,

где  - потомки (решения), полученные после применения генетического оператора (ГО).

2*. Последовательная схема редукции* позволяет варьировать методы выбора хромосом для удаления из популяции:

* случайный выбор,
* выбор «лучших» и «худших»,
* «близкое» родство,
* «дальне» родство,
* на основе кода Грея для бинарных хромосом,
* на основе расстояния Хэмминга,
* на основе «турнира».

Случайный выбор хромосом позволяет разнообразить генофонд на ранних этапах генетических алгоритмов. Вероятность этого выбора должна снижаться при эволюции поколений.

По аналогии с оператором репродукции известны следующие модификации оператора редукции:

1) равновероятностный отбор с вероятностью

(s) = .

где N – размер популяции;

2) пропорциональный отбор с вероятностью

(s) = .

Подведем итоги. С помощью операторов редукции на ранних стадиях работы генетических алгоритмов происходит выбор хромосом без учета значений их , т.е. случайный отбор. На заключительной стадии – определяющий фактор при отборе значения . Чем выше , тем выше вероятность отбора  в следующую популяцию. На заключительной стадии проводится уменьшение случайных операций и увеличивается процент направленных.

**5.17**. Рассмотрим теперь ***оператор рекомбинации.*** Оператор рекомбинации - языковая конструкция, которая определяет, как новая генерация хромосом будет построена из родителей и потомков. Другими словами, оператор рекомбинации - это технология анализа и преобразования популяции при переходе из одной генерации в другую. Существует много путей выполнения рекомбинации. Один из них состоит из перемещения родителей в потомки после реализации каждого генетического оператора. Другой путь заключается в перемещении некоторой части популяции после каждой генерации.

Часто в генетических алгоритмах задается параметр *W(P)*, который управляет этим процессом. Так, *Np(1- W(P))* элементов в популяции *Р*, выбранных случайно, могум «выжить» в следующей генерации. Здесь *Np* – размер популяции. Величина *W(P)* = 0 означает, что целая предыдущая популяция перемещается в новую популяцию в каждой генерации. При дальнейшей реализации алгоритма лучшие или отобранные элементы из родителей и потомков будут выбираться для формирования новой популяции.

В инженерных задачах используются различные механизмы и модели этого процесса. Приведем несколько из них:

* М1 – вытеснение (crowding factor). Этот механизм определяет способ и порядок замены родительских хромосом из генерации *t* хромосомами потомками после генерации *t+1*. Механизм реализован таким образом, что стремится удалять «похожие» хромосомы из популяции и оставлять отличающиеся.
* М2 – разделение (sharing). Этот механизм вводит зависимость значения целевой функции хромосомы от их распределения в популяции и поисковом пространстве. Это позволяет копиям родительских хромосом или близких к ним не появляться в популяции.
* М3 – введения идентификаторов (tagging). Специальным хромосомам присваиваются метки. Операторы генетических алгоритмов применяются только к помеченным хромосомам.

Отметим, что оператор редукции является частным случаем оператора рекомбинации.

**5.18**. Важным понятием при реализации генетических операторов является вероятность, которая определяет, какой процент общего числа генов в популяции изменяется каждой генерации. Для оптимизационных задач вероятность оператора кроссинговера обычно принимают в пределах (0,6÷0,99); вероятность оператора мутации – 0,6; инверсии – (0,1÷0,5); транслокации - (0,1÷0,5); транспозиции - (0,1÷0,5); сегрегации - (0,6÷0,99); удаления - (0,6÷0,99); вставки - (0,6÷0,99).

**Литература**

1. 1.Рутковская Д. И др.Нейронные сети, генетические алгоритмы и нечеткие системы:Пер. с польск. И.Д. Рудинского. М.:Горячая линия-Телеком, 2004.-452с.
2. Романов В.П. Интеллектуальные информационные системы в экономике: Учеб. пособие/ В.П. Романов.; Под ред. д.э.н., проф. Н.П. Тихомирова.-М.: Издательство Экзамен, 2003.-496 с.
3. Корнеев В.В. и др. Базы данных. Интеллектуальная обработка информации . – М.:”Нолидж”, 2000. – 352 с www/ Know ledge. ru
4. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы/Под ред. В.М. Курейчика.- 2-ое изд. испр. и доп.-М.:ФИЗМАТЛИТ. 2006-320 с.