Генетические алгоритмы

Предисловие

Среди множества проблем, которые возникают перед исследователями как в области теории, так и в многочисленных практических приложениях значительную долю составляют так называемые оптимизационные проблемы. Понятие оптимальности, по-видимому, знакомо почти каждому и вошло в практику большинства предметных областей. С оптимизационной проблемой мы сталкиваемся каждый раз, когда возникает необходимость выбора из некоторого множества возможных решений наилучшего по определенным критериями и, как правило, удовлетворяющего заданным условиям и ограничениям. Само понятие оптимальности получает совершенно строгое толкование в математических теориях, однако в других областях оно может интерпретироваться скорее содержательно. Тем не менее различие между строгим и содержательным понятиями оптимальности, как правило, очень незначительно.

Существуют классы оптимизационных задач, решение которых удается находить с помощью достаточно эффективных методов, вполне приемлемых по трудоемкости. Вместе с тем имеются и такие классы оптимизационных задач (так называемые NP- полные задачи), решение которых невозможно найти без полного перебора вариантов. В частности, к числу последних относятся многие разновидности задач многокритериальной оптимизации. Известно, что при большой размерности этих задач реализация перебора вариантов практически невозможна из-за чрезвычайно больших временных затрат.

В этой ситуации альтернативным походом к решению упомянутых задач является применение методов, базирующихся на методологии эволюционных вычислений. Предлагаемое пособие содержит изложение основ эволюционных вычислений и их приложений к решению различных проблем, включая проблемы экономики, прогнозирования финансовых рынков, инвестиций, бизнеса, комбинаторной оптимизации, сложных задач в различных технических разработках и т.п. Эффективность различных методов в рамках эволюционного подхода подтверждается многочисленными данными, касающимися достигаемым реальным эффектом. При этом хотя объем вычислений может оказаться большим, но скорость, с которой он растет при увеличении размерности задачи, обычно меньше, чем у остальных известных методов. Отметим, что после того как компьютерные системы стали достаточно быстродействующими и недорогими, эволюционные методы превратились в важный инструмент поиска близких к оптимальным решений задач, которые до этого считались неразрешимыми.

Основной целью учебного пособия является систематическое изложение перспективного направления в области теории и практики искусственного интеллекта. Пособие рассчитано на студентов и аспирантов вузов, обучающихся по направлениям, связанным с прикладной информатикой и математикой, компьютерными и программными системами, с интеллектуальными системами обработки информации.

Освоение представленного в пособии материала предполагает знакомство читателя с математикой и информатикой в объеме первых двух курсов технического вуза, основами дискретной математики и методов оптимизации.

Введение

В настоящее время оформилось и успешно развивается новое направление в теории и практике искусственного интеллекта – эволюционные вычисления (ЭВ). Этот термин обычно используется для общего описания алгоритмов поиска, оптимизации или обучения, основанных на некоторых формализованных принципах естественного эволюционного отбора. Особенности идей эволюции и самоорганизации заключаются в том, что они являются плодотворными и полезность их применения не только для биологических систем перманентно подтверждается. Эти идеи в настоящее время с успехом используются при разработке многих технических и, в особенности, программных систем.

Высокая согласованность и эффективность работы элементов биологических систем приводила целый ряд исследователей к естественной мысли о возможности использования принципов биологической эволюции для оптимизации важных для приложений систем, природа которых отлична от биологической. Так, в 1966 году Фогель Л., Оуенс С. и Уолш М. в [[1](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.1)] подобную идею использовали для построения схемы эволюции логических автоматов, решающих задачи прогноза. В 1975 году была опубликована основополагающая работа Дж. Холланда [[2](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.2)], в которой был предложен генетический алгоритм, развивающий ту же идею. Д.Гольдберг, ученик Дж. Холланда, в работе [[3](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.3)] , выполненной в Мичиганском университете, успешно развил и расширил области его применения.

В 60-х годах прошлого века в Германии Рохенберг И., Швефель Г.-П., и др. [[4](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.4)] начали разработку так назывемой эволюционной стратегии. Перечисленные работы послужили толчком и основой развития прикладного направления, которое можно назвать эволюционными алгоритмами. К их числу, помимо упомянутых генетических алгоритмов и эволюционных стратегий, относятся также эволюционное программирование, ориентированное на оптимизацию функций без использования рекомбинаций, и генетическое программирование, использующее эволюционные идеи для оптимизации компьютерных программ. Основополагающая работа по эволюционному программированию принадлежит Дж. Коза [[5](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.5)] из Массачусетского технологического института.

Отметим, что аналогичные исследования успешно проводились отечественными учеными еще в Советском Союзе. Так, значительный вклад в развитие указанных направлений внесли Ивахненко А.Г. [[6](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.6)], Цыпкин Я.З. [[7](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.7)], Расстригин Л.А. [[8](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.8)] и др. В настоящее время исследования по ЭВ активно ведутся Букатовой [[9](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.9),[10](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.10)], Курейчиком В.М. [[11](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.11),[12](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.12),[13](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.13)], их сотрудниками и учениками, а также многими другими исследователями.

Учебное пособие состоит из пяти частей.

В первой части "Основы генетических алгоритмов" (разделы пособия 1-5) изложены идейная сторона простых генетических алгоритмов, их математические основы, подробно описаны примеры построения генетических алгоритмов для решения конкретных задач комбинаторной оптимзации и, наконец, представлены современные модификации и обобщения этих алгоритмов.

Во второй части "Генетическое программирование и машинное обучение" (разделы 6-7) описана концепция компьютерного синтеза программ (с различными структурами их представления – линейными, древовидными, графоподобными) с использованием генетических алгоритмов. Изложены два основных подхода при машинном обучении - Мичиганский и Питтсбургский.

В третьей части "Вероятностные генетические алгоритмы"(раздел 8) приведен другой – вероятностный подход в эволюционных вычислениях, где популяция представляется вектором вероятностей.

В четвертой части "Эволюционные стратегии" (раздел 9) представлена оригинальная парадигма, основанная на эволюции популяции потенциальных решений. Ее принципиалное отличие состоит в том, что генетические операторы используются здесь на уровне фенотипа, а не генотипа, как это было в генетических алгоритмах.

В пятой части "Эволюционное программирование" (раздел 10) содержится описание основанного Фогелем Л.Дж. гибкого подхода в эволюционных вычислениях. В нем форма представления потенциального решения и генетические операторы адаптируются к решаемой проблемы в достаточно широких пределах. В частности, в качестве особи в процессе эволюции Фогелем используются конечные автоматы и целью эволюции является способность решения задач прогнозирования.

В шестой части "Роевой интеллект" (разделы 11-12) описаны принципы и методы оптимизации, базирующиеся на коллективном поведении децентрализованных самоорганизующихся систем. Такие системы представляются, например, в виде графа, содержащего множество вершин (агентов), локально взаимодействующих между собой и с окружающей средой. Имеющиеся экспериментальные данные подверждают эффективность роевого интеллекта (муравьиных, пчелиных, роевых и т.п. алгоритмов) для решения многих оптимизационных задач.

Заканчивая введение заметим, что в пособии представлены не все аспекты быстро развивающихся эволюционных вычислений. Мы ограничились здесь расмотрением только наиболее важных и принципиальных с нашей точки зрения моментов эволюционных вычислений, хотя по этому поводу могут быть и другие мнения.

1. Введение в генетические алгоритмы

В этом разделе описывается концепция простого генетического алгоритма (ГА), ориентированного на решение различных оптимизационных задач. Вводятся и содержательно описываются понятия, используемые в теории и приложениях ГА. Приводится фундаментальная теорема ГА и излагается теория схем, составляющие теоретическую базу ГА. Обсуждаются концептуальные вопросы, касающиеся преимуществ и недостатков ГА.

1.1. Простой генетический алгоритм

Основы теории генетических алгоритмов сформулированы Дж. Г.Холландом в основополагающей работе [[2](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.2)] и в дальнейшем были развиты рядом других исследователей. Наиболее известной и часто цитируемой в настоящее время является монография Д.Голдберга [[3](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.3)], где систематически изложены основные результаты и области практического применения ГА.

ГА используют принципы и терминологию, заимствованные у биологической науки – генетики. В ГА каждая особь представляет потенциальное решение некоторой проблемы. В классическом ГА особь кодируется строкой двоичных символов – хромосомой, каждый бит которой называется геном. Множество особей – потенциальных решений составляет популяцию. Поиск оптимального или субоптимального решения проблемы выполняется в процессе эволюции популяции, т.е. последовательного преобразования одного конечного множества решений в другое с помощью генетических операторов репродукции, кроссинговера и мутации. ЭВ используют механизмы естественной эволюции, основанные на следующих принципах:

1. Первый принцип основан на концепции выживания сильнейших и естественного отбора по Дарвину, который был сформулирован им в 1859 году в книге "Происхождение видов путем естественного отбора". Согласно Дарвину особи, которые лучше способны решать задачи в своей среде, чаще выживают и чаще размножаются (репродуцируют). В генетических алгоритмах каждая особь представляет собой решение некоторой проблемы. По аналогии с этим принципом особи с лучшими значениями целевой (фитнесс) функции имеют большие шансы выжить и репродуцировать. Формализацию этого принципа, как мы увидим далее, реализует оператор репродукции.
2. Второй принцип обусловлен тем фактом, что хромосома потомка состоит из частей, полученных из хромосом родителей. Этот принцип был открыт в 1865 году Г.Менделем. Его формализация дает основу для оператора скрещивания (кроссинговера).
3. Третий принцип основан на концепции мутации, открытой в 1900 году де Вре. Первоначально этот термин использовался для описания существенных (резких) изменений свойств потомков и приобретение ими свойств, отсутствующих у родителей. По аналогии с этим принципом генетические алгоритмы используют подобный механизм для резкого изменения свойств потомков и, тем самым, повышают разнообразие (изменчивость) особей в популяции (множестве решений).

Эти три принципа составляют ядро ЭВ. Используя их, популяция (множество решений данной проблемы) эволюционирует от поколения к поколению.

Эволюцию искусственной популяции – поиск множества решений некоторой проблемы, формально можно описать в виде алгоритма, который представлен на [рис.1.1.](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=2#image.1.1)

ГА получает множество параметров оптимизационной проблемы и кодирует их последовательностями конечной длины в некотором конечном алфавите (в простейшем случае в двоичном алфавите "0" и "1").

Предварительно простой ГА случайным образом генерирует начальную популяцию хромосом (стрингов). Затем алгоритм генерирует следующее поколение (популяцию) с помощью трех следующих основных генетических операторов:

1. оператора репродукции (ОР);
2. оператора скрещивания (кроссинговера, ОК);
3. оператора мутации (ОМ).

Генетические алгоритмы – это не просто случайный поиск, они эффективно используют информацию, накопленную в процессе эволюции.

В процессе поиска решения необходимо соблюдать баланс между "эксплуатацией" полученных на текущий момент лучших решений и расширением пространства поиска. Различные методы поиска решают эту проблему по-разному.

Например, градиентные методы практически основаны только на использовании лучших текущих решений, что повышает скорость сходимости с одной стороны, но порождает проблему локальных экстремумов с другой. В полярном подходе случайные методы поиска используют все пространство поиска, но имеют низкую скорость сходимости. В ГА предпринята попытка объединить преимущества этих двух противоположных подходов. При этом операторы репродукции и кроссинговера делают поиск направленным. Широту поиска обеспечивает то, что процесс ведется на множестве решений – популяции и используется оператор мутации.

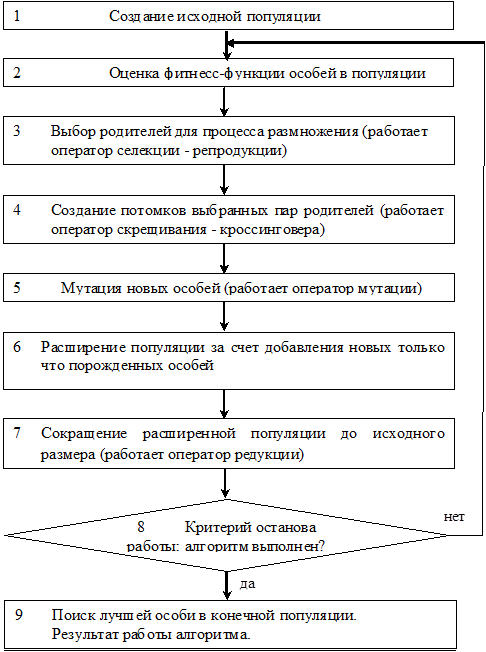


Рис. 1.1. Простой генетический алгоритм

В отличие от других методов оптимизации ГА оптимизируют различные области пространства решений одновременно и более приспособлены к нахождению новых областей с лучшими значениями целевой функции за счет объединения квазиоптимальных решений из разных популяций.

Упомянутые генетические операторы являются математической формализацией приведенных выше трех основополагающих принципов Ч.Дарвина, Г.Менделя и де Вре естественной эволюции. Каждый из функциональных блоков ГА на [рис. 1.1.](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=2#image.1.1) может быть реализован различными способами. Но сначала на простом примере мы рассмотрим основные моменты классического ГА.

1.2. Генетические операторы

Рассматриваемый ниже пример заимствован из популярной монографии Голдберга [[3](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.3)] и состоит в поиске целочисленного значения (для простоты) x на отрезке от [0,31], при котором функции y=x^2 принимает максимальное значение.

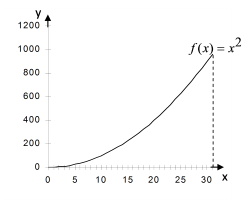


Рис. 1.2. Пример функции

Начальный этап работы ГА для данного примера приведен в верхней таблице (репродукция) [рис.1.3](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=3#image.1.3) Здесь особи начальной популяции (двоичные коды значений переменных x- столбец 2) сгенерированы случайным образом. Двоичный код значения х называется хромосомой (она представляет генотип). Популяция образует множество потенциальных решений данной проблемы. В третьем столбце представлены их десятичные значения (фенотип). Далее на этом примере проиллюстрируем работу трех основных генетических операторов.

1.2.1. Репродукция

Репродукция – это процесс, в котором хромосомы копируются в промежуточную популяцию для дальнейшего "размножения" согласно их значениям целевой (фитнесс-) функции. При этом хромосомы с лучшими значениями целевой функции имеют большую вероятность попадания одного или более потомков в следующее поколение.

Очевидно, оператор репродукции (ОР) является искусственной версией естественной селекции – выживания сильнейших по Ч. Дарвину. Этот оператор представляется в алгоритмической форме различными способами (подробнее различные варианты ОР будут рассмотрены далее). Самый простой (и популярный) метод реализации ОР – построение колеса рулетки, в которой каждая хромосома имеет сектор, пропорциональный по площади значению ее целевой функции. Для нашего примера "колесо рулетки" имеет вид, представленный на [рис.1.4.](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=3#image.1.4)

Для селекции хромосом используется случайный поиск на основе колеса рулетки. При этом колесо рулетки вращается и после останова ее указатель определяет хромосому для селекции в промежуточную популяцию (родительский пул).

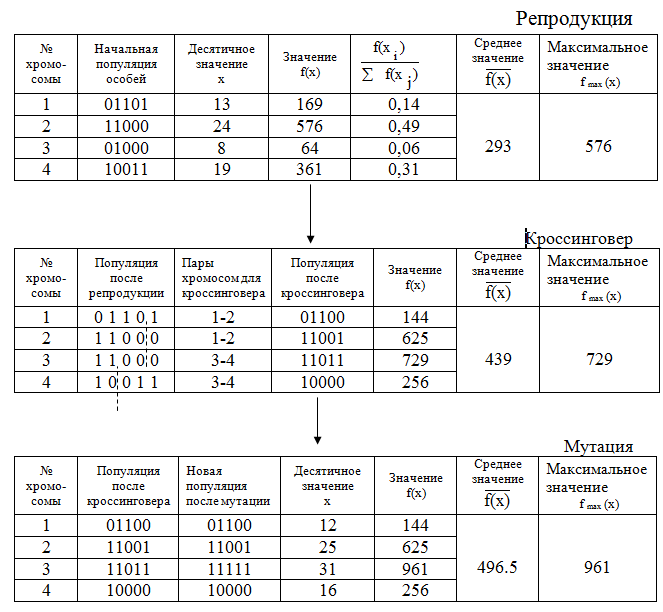


Рис. 1.3. Эволюция популяции

Очевидно, что хромосома, которой соответствует больший сектор рулетки, имеет большую вероятность попасть в следующее поколение. В результате выполнения оператора репродукции формируется промежуточная популяция, хромосомы которой будут использованы для построения поколения с помощью операторов скрещивания.

В нашем примере выбираем хромосомы для промежуточной популяции, вращая колесо рулетки 4 раза, что соответствует мощности начальной популяции. Величину \frac{f(x_i)}{\sum f(x_j)} обозначим как P(x_i), тогда ожидаемое количество копий i-ой хромосомы определяется значением M=P(x_i)*N, где N-мощность популяции. Число копий хромосомы, переходящих в следующее поколение, иногда определяется и так: \tilde M=\frac{f(x_i)}{\bar f(x)}, где \bar f(x)- среднее значение хромосомы в популяции.

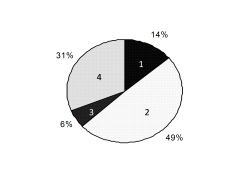


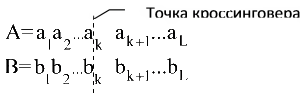
Рис. 1.4. Колесо рулетки

Расчетные числа копий хромосом по приведенной формуле следующие: хромосома 1 - 0,56; хромосома 2 - 1,97; хромосома 3 – 0,22; хромосома 4 – 1,23. В результате, в промежуточную популяцию 1-я хромосома попадает в одном экземпляре, 2-я – в двух, 3-я – совсем не попадает, 4-я – в одном экземпляре. Полученная промежуточная популяция является исходной для дальнейшего выполнения операторов кроссинговера и мутации.

1.2.2 Оператор кроссинговера (скрещивания)

Одноточечный или простой оператор кросинговера (ОК) с заданной вероятностью P_c выполняется в три этапа:

1-й этап. Две хромосомы (родители) выбираются случайно (или одним из методов, рассмотренных далее) из промежуточной популяции, сформированной при помощи оператора репродукции (ОР).



2-й этап. Случайно выбирается точка скрещивания - число k из диапазона [1,n-1], где n– длина хромосомы (число бит в двоичном коде).

Две новых хромосомы A', B' (потомки) формируются из A и B путем обмена подстрок после точки скрещивания:

A'=a_1 a_2\dots a_k\ b_{k+1}\dots b_L\\B'=b_1 b_2\dots b_k\ a_{k+1}\dots a_L

Например, рассмотрим выполнение кроссинговера для хромосом 1 и 2 из промежуточной популяции:

A=0\ 1\ 1\ 0\ 1\\B=1\ 1\ 0\ 0\ 0\\1\le k\le 4, k=4\\A'=0\ 1\ 1\ 0\ 0\\B'=1\ 1\ 0\ 0\ 1\\

Следует отметить, что ОК выполняется с заданной вероятностью P_c (отобранные два родителя не обязательно производят потомков). Обычно величина вероятности полагается равной P_c\approx 0,5.

Таким образом, операторы репродукции и скрещивания очень просты – они выполняют копирование особей и частичный обмен частей хромосом. Продолжение нашего примера представлено на [рис.1.3](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=3#image.1.3) во второй таблице (кроссинговер).

Сравнение с предыдущей таблицей показывает, что в промежуточной популяции после скрещивания улучшились все показатели популяции (среднее и максимальное значения целевой функции - ЦФ).

1.2.3 Мутация

Далее согласно схеме классического ГА с заданной вероятностью P_M выполняется оператор мутации. Иногда этот оператор играет вторичную роль. Обычно вероятность мутации мала - P_m\approx 0,001

Оператор мутации (ОМ) выполняется в два этапа:

1-й этап. В хромосоме A=a_1 a_2\dots a_L случайным образом выбирается k-ая позиция (бит) (1\le k\le n).

2-й этап. Производится инверсия значения гена в k-й позиции: a'_k=\bar a_k

Например, для хромосомы 11011 выбирается k=3 и после инверсии значения третьего бита получается новая хромосома – 11111. Продолжение нашего примера представлено в третьей таблице (мутация) [рис.1.3](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=3#image.1.3) образом, в результате применения генетических операторов найдено оптимальное решение x=31.

В данном случае, поскольку пример искусственно подобран, мы нашли оптимальное решение за одну итерацию. В общем случае ГА работает до тех пор, пока не будет выполнен критерий окончания процесса поиска и в последнем полученном поколении определяется лучшая особь, которая и принимается в качестве решения задачи.

1.3. Представление вещественных решений в двоичной форме

В предыдущем примере мы рассматривали только целочисленные решения. Обобщим ГА на случай вещественных чисел на примере функции f(x)=(1,85-x)*\cos(3,5x-0,5), представленной на [рис.1.5](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=4#image.1.5) [[14](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.14)]. Рассматривается та же задача: необходимо найти вещественное x\in [-10,+10], которое максимизирует f(x), т.е. такое x_0, для которого f(x_0)\ge f(x) для всех x\in [-10,+10].

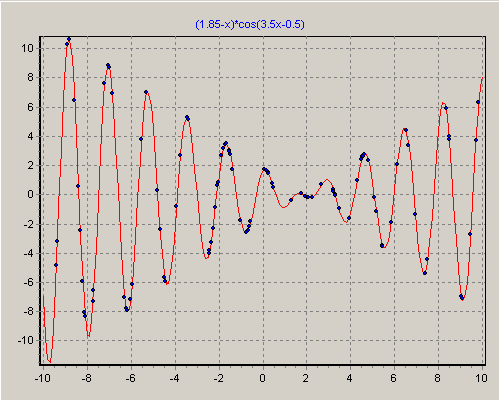


Рис. 1.5. Пример функции с популяцией особей в начале эволюции

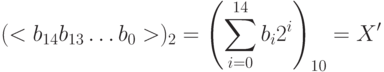
Для решения этой задачи с помощью ГА будем использовать представления вещественного решения (хромосомы) в виде двоичного вектора [[15](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.15)], который применяется в классическом простом ГА. Его длина зависит от требуемой точности решения, которую в данном случае положим, например, равной 3 знакам после запятой.

Поскольку отрезок области решения имеет длину 20, для достижения заданной точности отрезок [a,c]=[-10,+10] должен быть разбит на равные части (маленькие отрезки), число которых должно быть не менее 20\*1000. В качестве двоичного представления используем двоичный код номера (маленького) отрезка. Этот код позволяет определить соответствующее ему вещественное число, если известны границы области решения. Отсюда следует, что двоичный вектор для кодирования вещественного решения должен иметь 15 бит, поскольку

16384=2^{14}<20000\le2^{15}=32768

Отсюда следует, что для обеспечения необходимой точности требуется разбить отрезок [-10,+10] на 32768 частей. Отображение из двоичного представления (b_{14} b_{13}\dots b_0)\ (b_i\in \{0,1\} в вещественное число из отрезка [a,c]=[-10,+10] выполняется в два шага.

1. Перевод двоичного числа в десятичное:



1. Вычисление соответствующего вещественного числа x:

x=a+x'\cdot\frac{(c-a)}{2^{15}-1}=-10+x'\cdot\frac{20}{2^{15}-1},

, где (– 10) левая граница области решения. Естественно хромосомы (000000000000000) и (111111111111111) представляют границы отрезка –10 и +10 соответственно.

Очевидно, при данном двоичном представлении вещественных чисел можно использовать классический простой ГА. На [рис.1.6](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=4#image.1.6),[рис.1.7](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=4#image.1.7),[рис.1.8](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=4#image.1.8) представлено расположение особей (потенциальных решений) на различных этапах ГА в процессе поиска решения[[14](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.14)]. На [рис.1.5](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=4#image.1.5) показана начальная популяция потенциальных решений, которая равномерно покрывает область поиска решения. Далее явно видно, как постепенно с увеличением номера поколения особи "конденсируются" в окрестностях экстремумов и, в конечном счете, находится лучшее решение.

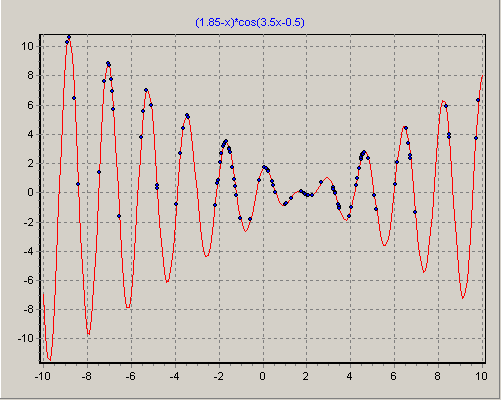


Рис. 1.6. Начальная "конденсация" особей популяции в окрестностях экстремумов

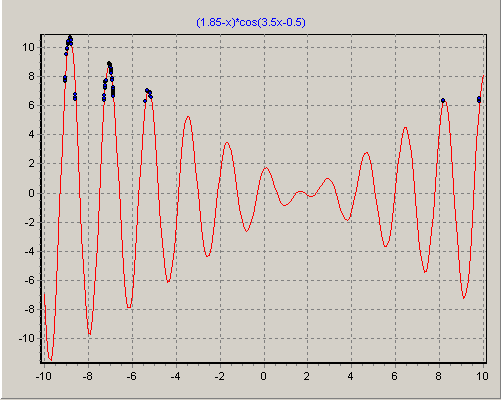


Рис. 1.7. "Конденсация" особей в окрестностях экстремумов

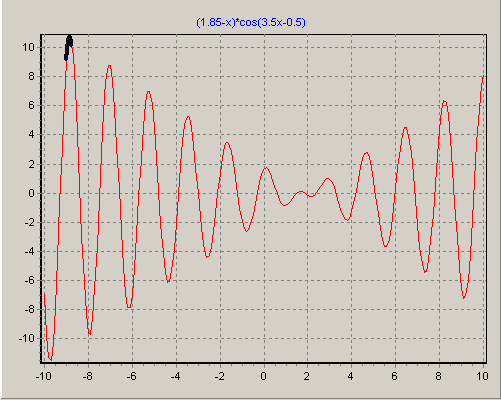


Рис. 1.8. Положение особей популяции в конце эволюции

Для сокращения длины хромосом иногда применяют логарифмическое кодирование, при котором первый бит (a) кодовой последовательности используется для знака показательной функции, второй бит (b) – для знака степени этой функции, и остальные биты (str) представляют значение самой степени [[16](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.16)]. Таким образом, двоичный код <a\ b\ str> представляет вещественное число (-1)^a e^{(-1)^b [str]_{10}}. Здесь [str]_{10} означает десятичное число, представленное двоичным кодом str.

Например, двоичный код <10101> представляет вещественное число (-1)^0 e^{(-1)^1 [101]_{10}} =e^{-6}=0,002478752. Следует отметить, что при таком кодировании пять битов позволяет кодировать вещественные числа из интервала [-e^7,e^7], что значительно больше, чем это позволяет, например, метод кодирования, представленный ранее.

1.4. Использование кода Грея в ГА

Рассмотренное двоичное представление вещественного числа имеет существенный недостаток: расстояние между вещественными числами (на числовой оси) часто не соответствует расстоянию (по Хеммингу) между их двоичными представлениями. Поэтому желательно получить двоичное представление, где близкие расстояния между хромосомами (двоичными представлениями) соответствовали близким расстояниям в проблемной области (в данном случае расстоянию на числовой оси) [[3](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.3),[14](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.14),[15](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.15)]. Это можно сделать, например, с помощью кода Грея. В [таблице 1.1](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=5#table.1.1) приведен для примера код Грея для 4-х битовых слов.

|  |  |
| --- | --- |
| **Таблица 1.1.** | |
| **Двоичный код** | **Код Грея** |
| 0000 | 0000 |
| 0001 | 0001 |
| 0010 | 0011 |
| 0011 | 0010 |
| 0100 | 0110 |
| 0101 | 0111 |
| 0110 | 0101 |
| 0111 | 0100 |
| 1000 | 1100 |
| 1001 | 1101 |
| 1010 | 1111 |
| 1011 | 1110 |
| 1100 | 1010 |
| 1101 | 1011 |
| 1110 | 1001 |
| 1111 | 1000 |

Заметим, что в коде Грея соседние двоичные слова отличаются на один бит (расстояние по Хеммингу равно 1).

Рассмотрим алгоритмы преобразования двоичного числа \bar b=<b_1,\dots,b_m> в код Грея \bar g=<g_1,\dots,g_m> и наоборот.

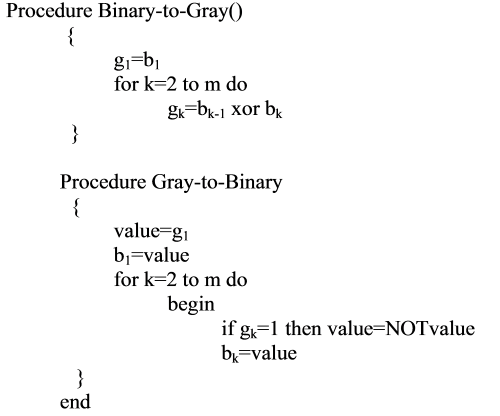
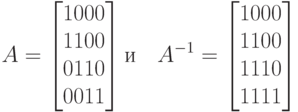


Рис. 1.9. Преобразование в код Грея

Здесь параметр m определяет разрядность двоичного числа. Существует и другая, матричная, процедура преобразования в код Грея. Например, для m=4 матрицы



позволяют выполнять следующие преобразования:

\bar g=A\cdot\bar b\ \mbox{и}\ \bar b=A^{-1}\bar g

где умножение матриц выполняются в арифметике по mod 2.

Отметим, что применение кода Грея прежде всего оправдано при использовании операторов мутации.

1.5. Фитнесс-функция

Как видно из основной блок-схемы ГА на [рис.1.1](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=2#image.1.1), каждая особь популяции (потенциальное решение проблемы) оценивается путем вычисления значения фитнесс-функции. Следует отметить, что в общем случае целевая функция (ЦФ) и фитнесс-функция могут различаться. Целевая функция предназначена для оценки характеристик особи относительно конечной цели (например, экстремумов). Фитнесс-функция предназначена, прежде всего, для отбора особей для дальнейшей репродукции и здесь важны характеристики качества одной особи относительно других особей. После декодирования хромосомы, где выполняется преобразование "генотип \to фенотип" (например, двоичный код преобразуется в вещественное число), полученное значения далее используется в качестве аргумента для фитнесс-функции. Далее для каждой особи популяции вычисляются значения фитнесс-функции, которые ранжируют эти особи относительно друг друга в смысле перспективности получения из них хорошего решения.

Определение соответствующей фитнесс-функции является решающим для корректной работы ГА. В частности, вид фитнесс-функции может зависеть от накладываемых ограничений при решении оптимизационных задач. Отметим, что операторы кроссинговера и мутации не учитывают, попадают ли вновь построенные особи-потомки в область допустимых решений, которая обусловлена накладываемыми ограничениями.

В ГА используются четыре основных метода для учета накладываемых ограничений при решении оптимизационных задач. Вероятно, простейшим способом является метод отклонения (отбрасывания), где недопустимые хромосомы (не удовлетворяющие ограничениям) исключаются из дальнейшей эволюции. Второй метод основан на использовании процедуры восстановления, которая преобразует полученное недопустимое решение в допустимое. Другой альтернативой является применение проблемно-ориентированных генетических операторов, которые порождают только допустимые решения.

Рассмотренные методы не строят недопустимых решений. Но это не всегда дает хорошие результаты. Например, в том случае, когда оптимальные решения лежат на границе допустимой области, указанные методы могут давать неоптимальные решения. Одним из возможных вариантов преодоления этой проблемы является выполнение процедуры восстановления только для некоторого подмножества решений (например, 10% особей).

Для решения оптимизационных задач со сложными ограничениями иногда позволяют вести поиск решения и в недопустимых областях. Реализуется это подход часто с помощью метода штрафных функций, что позволяет расширить пространство поиска решений. Следует отметить, что часто недопустимая точка, близкая к оптимальному решению, содержит больше полезной информации, чем допустимая точка, далекая от оптимума. С другой стороны, построение штрафных функций является достаточно сложной проблемой, которая сильно зависит от решаемой задачи. Обычно нет априорной информации о расстоянии до оптимальных точек, есть только расстояние до границы области допустимых решений. Поэтому, как правило, штрафные функции используют расстояние до границ допустимой области. Штрафы, основанные на нарушении отдельных ограничений, работают обычно не очень хорошо.

Разработаны два основных способа построения штрафных функций со штрафным термом: аддитивная и мультипликативная формы. В первой форме функция представляется в виде g(x)=f(x)+p(x), где при максимизации для допустимых точек p(x)=0 и в противном случае p(x)<0. Максимум значения p(x) по абсолютной величине не может быть больше, чем минимальное значение f(x) по абсолютной величине для любой генерации, чтобы избежать отрицательных фитнесс-значений. Мультипликативная форма представляет функцию в виде g(x)=f(x)\cdot p(x), где при максимизации p(x)=1 для допустимых точек и 0\le p(x)<1 в противном случае.

При этом штрафной терм должен изменяться не только в зависимости от степени нарушения ограничения, но и от номера поколения ГА. Наряду с нарушением ограничения, штрафной терм обычно содержит штрафные коэффициенты (по одному для каждого ограничения). На практике большую роль играют значения этих коэффициентов. Маленькие значения коэффициентов могут привести к недопустимым значениям решения, в то время как большие значения полностью отвергают недопустимые подпространства. В среднем абсолютные значения целевой и штрафной функции должны быть соизмеримы. При таком подходе параметры штрафной функции можно включить в параметры ГА, что позволяет разработать адаптивный метод, где значения коэффициентов регулируются в процессе поиска решения.

В целом на выбор (построение) фитнесс-функции оказывают влияние следующие факторы:

* тип задачи – максимизация или минимизация;
* содержание шумов окружающей среды в фитнесс-функции;
* возможность динамического изменения фитнесс-функции в процессе решения задачи;
* объем допустимых вычислительных ресурсов – допускается ли использовать более точные методы и значительные ресурсы, или возможны только приближенные аппроксимации, не требующие больших ресурсов;
* насколько различные значения для особей должна давать фитнесс-функция для облегчения отбора родительских особей;
* должна ли она содержать ограничения решаемой задачи;
* может ли она совмещать различные подцели (например, для многокритериальных задач).

В ГА фитнесс-функция используется в виде черного ящика: для данной хромосомы она вычисляет значение, определяющее качество данной особи. Внутри она может быть реализована по-разному: в виде математической функции, программы моделирования (в том числе имитационного), нейронной сети, или даже экспертной оценки.

Для некоторых задач оценку значений фитнесс-функции можно выполнять с помощью объектно-ориентированной имитационной модели. Взаимодействие такой модели с ГА показано на [рис.1.10](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=6#image.1.10).

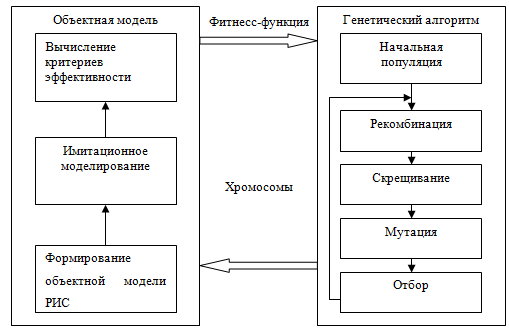


Рис. 1.10. Схема взаимодействия объектной модели с ГА.

1.6. Теория схем

Теоретические основы ГА составляют двоичное стринговое представление решений (хромосом) и понятие схемы (schema) или шаблона. Это понятие было ведено Холландом для определения множества хромосом, которые обладают некоторыми общими свойствами, то есть в каком- то смысле подобны друг другу.Термин "схема" согласно Холланду [[2](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.2)] есть шаблон, описывающий подмножество стрингов, имеющих одинаковые значения в некоторых позициях. Для этого вводится новый троичный алфавит {0, 1, \*}, где \* означает неопределенное значение (0 или 1, т.е. неизвестно что именно). Например, схема (0\*0001) соответствует двум стрингам {000001, 010001}, а (\*0110\*) описывает подмножество из 4-х стрингов {001100, 101100, 001101, 101101}. Очевидно, схема с m неопределенными позициями "\*" представляет 2^m стрингов. Для стрингов длины n, существует 3^n схем (возможны 3 символа {0, 1, \*} в каждой из n позиций).

В простом ГА основная идея заключается в объединении хромосом со значениями целевой функции (ЦФ) выше среднего. Например, пусть 1 в хромосоме соответствует наличию признака, способствующего выживанию (значение целевой функции больше среднего). Допустим, что имеются подстринги вида 11\*\*\* и \*\*111. Тогда, применяя к ним ОК можно получить хромосому 11111 с признаками, способствующими наилучшим значениям фитнесс-функции. Рассмотрим также полезное понятие строительного блока. Например, в шаблоне \*\*\*\*1 строительным блоком является элемент 1. В шаблоне 10\*\*\* строительным блоком будет составной элемент 10. Очевидно, вид строительных блоков должен выбираться из знаний о решаемой задаче и отражать полезные свойства, чтобы далее из строительных блоков как из "кирпичиков" собрать "здание", т.е. решение с лучшим значением ЦФ. Желательно, чтобы в ГА выражались условия, которые разрешают разрыв строительных блоков только в крайних случаях, указанных пользователем

Не все схемы являются одинаковыми. Некоторые более специфичны, чем другие. Для количественной оценки вводятся две характеристики:

1. порядок схемы O(H), который определяется числом фиксированных позиций (не равных \*). Например, для H_1= 0\*1\*\*\* ее порядок O(H_1)= 2.
2. определенная длина L(H)– это расстояние между первой и последней определенной (не равной \*) позицией. Например для шаблона H_2= 0111\*1\*\* L(H_2)= 6 – 1 = 5. А для H_3= 0\*\*\*\* L(H_3)= 1 – 1 = 0.

Обозначим через m(H,t)– число стрингов, содержащихся в популяции A(t)(t – шаг итерации или время), которые отображаются (покрываются) схемой H.

Пусть f(H) означает среднее значение фитнесс-функции для хромосом, покрываемых данной схемой H, а \overline {f(x)}=\frac{\sum_{j=1}^N f(x_j)}{N}\\ – среднее значение фитнесс-функции для всей популяции (N – размер популяции). Очевидно, что для схемы, которая представляет хорошее решение, было бы желательным, чтобы количество хромосом, соответствующих этой схеме, возрастало в процессе эволюции с ростом номера поколения. Далее мы рассмотрим, как ведет себя m(H,t) при выполнении основных операторов ГА.

Фундаментальная теорема ГА

Влияние репродукции

Напомним, что в процессе репродукции хромосомы копируются в промежуточную популяцию согласно их значениям фитнесс-функции – хромосома x_i со значением f(x_i) выбирается с вероятностью P(x_i)=\frac{f(x_i)}{\sum f(x_j)}.

После репродукции мы ожидаем на следующем шаге получить m(H,t+1) двоичных стрингов, отображаемых схемой H. Известно, что

|  |  |
| --- | --- |
| m(H,t+1)=\frac{m(H,t)*N*f(H)}{\sum f(x_j)} | ( 1.1) |

Это обусловлено тем, что:

1. в среднем вероятность выбора стрингов, покрываемых схемой H, определяется величиной \frac{f(H)}{\sum f(x_j)},
2. число стрингов, представляемых H, равно m(H,t),
3. число стрингов в популяции равно N.

Мы можем переписать эту формулу с учетом обозначения

\overline{f(x)}=\frac{\sum_{j=1}^N f(x_j)}{N}

и получим следующее выражение для числа особей, покрываемых схемой в промежуточной популяции :

|  |  |
| --- | --- |
| m(H,t+1)=\frac{m(H,t)*f(H)}{\overline{f(x)} | ( 1.2) |

Другими словами, схемы "растут" как отношение среднего значения фитнесс-функции схемы к среднему значению фитнесс-функции популяции. Схема со значением фитнесс-функции выше средней в популяции имеет больше возможностей для копирования и наоборот. Правило репродукции Холланда гласит: "схема со значением фитнесс-функции выше среднего живёт и размножается, а со значением фитнесс-функции ниже среднего умирает".

Предположим, что схема H имеет значение выше среднего фитнесс-функции на величину c*f, где c– константа. Тогда последнее выражение (1.2) можно модифицировать:

m(H,t+1)=\frac{m(H,t)*(\bar f+c*\bar f)}{\bar f}=(1+c)m(H,t)

Начиная с t=0 и предполагая, что c– величина постоянная, получаем следующее выражение числа особей промежуточной популяции, покрываемых схемой,

|  |  |
| --- | --- |
| m(H,t)=m(H,0)*(1+c)^t | ( 1.3) |

Это равенство описывает геометрическую прогрессию. Очевидно, что при c>0 схема "размножается" а при c<0 схема умирает.

Далее рассмотрим влияние оператора кроссинговера на число особей в популяции, покрываемых схемой.

Влияние кроссинговера

Рассмотрим конкретный стринг А=011|1000 длины n=7 и две схемы, представляющие этот стринг: H_1 = \*1\*| \*\*\*0, H_2= \*\*\*| 10\*\*

Здесь символ "|" , как обычно, обозначает точку кроссинговера k=3.

Очевидно, что схема H_1 после кроссинговера с точкой k=3, скорее всего, будет уничтожена потому, что "1" в позиции 2 и "0" в позиции 7 попадут в разные новые стринги после кроссинговера. С другой стороны, ясно, что схема H_2 будет выживать, так как "10" в позициях 4,5 будут содержаться вместе в одном новом стринге. Хотя мы взяли точку скрещивания ОК случайно, ясно, что схема H_1 менее приспособлена к выживанию, чем схема H_2. Если точка скрещивания ОК выбирается случайным образом среди n-1=7-1=6 возможных позиций, то ясно, что схема H_1 разрушается с вероятностью

P(d)=\frac{L(H_1)}{(n-1)}=\frac{5}{6}

Очевидно, что эта же схема выживает с вероятностью

P(S)=1-P(d)=\frac{1}{6}

Аналогично, схема H_2 имеет длину L(H_2) = 1 и вероятность её уничтожения P(d) = 1/6, а вероятность выживания схемы после применения ОК P(S) = 5/6. Очевидно, что нижняя граница вероятности выживания схемы после применения ОК может быть вычислена для любой схемы. Так как схема выживает, когда точка ОК попадает вне "определенной длины" , вероятность выживания для простого ОК определяется по формуле P_s(OK) = 1 – L(H)/(n-1).

Если ОК выполняется посредством случайного выбора, например, с вероятностью P_c, то вероятность выживания схемы определяется так: P(S) \ge 1 – P_c*L(H)/(n-1).

Очевидно, что это выражение уменьшается при P_c\to 1. Теперь мы можем асимптотически оценить совместный эффект операторов репродукции и кроссинговера. При независимости выполнения OP и OK можно получить следующее выражение:

|  |  |
| --- | --- |
| m(H,t+1)\ge m(H,t)\frac{f(H)}{\bar f}[1-P_c\frac{L(H)}{n-1}] | ( 1.4) |

Таким образом, число схем H в новой популяции зависит от двух факторов:

1. значение фитнесс-функции схемы выше или ниже значения ЦФ популяции;
2. схема имеет "длинную" или "короткую" L(H) (определенную длину).

Видно, что схемы со значением ЦФ выше средней и с короткой длиной L(H) имеют возможность экспоненциального роста в новой популяции.

Далее рассмотрим влияние оператора мутации на число особей в популяции, покрываемых схемой.

Влияние мутации

Напомним, что оператор мутации (ОМ) есть случайное изменение элемента в стринге с вероятностью P_m. Очевидно, что для того чтобы схема H выжила, все определенные позиции должны выжить. Поскольку один ген выживает с вероятностью (1-P_m), то данная схема выживает, когда каждая из O(H) фиксированных позиций схемы выживает. Умножая вероятность выживания (1-P_m) O(H) раз, получим, что вероятность выживания при ОМ равна (1-p_m)^{O(H)}. Для малых величин P_m\ll 1 это выражение может быть аппроксимировано как P_S(OM) = 1 – О(H)\cdot P_m.

Из этого следует, что схема H дает ожидаемое число особей в следующей популяции после выполнения всех генетических операторов ОР, ОК и ОМ согласно следующей формуле:

|  |  |
| --- | --- |
| m(H,t+1)>m(H,t)\cdot \frac{f(H)}{\bar f}[1-P_m\frac{L(H)}{1-n}-O(H)\cdot P_m] | ( 1.5) |

Этот важный результат известен какфундаментальная теорема ГА. Учитывая (1.3) ее можно также сформулировать следующим образом.

Теорема 1.1. Схемы малого порядка, малой определенной длины и со значением фитнесс-функции выше средней формируют растущее по показательному закону число своих представителей в последующих поколениях генетического алгоритма.

На основании приведенных результатов была выдвинута гипотеза о строительных блоках.

Гипотеза 1.1. Генетический алгоритм стремится достичь близкого к оптимальному результата за счет комбинирования хороших схем (со значением фитнесс-функции выше средней, малого порядка и определенной длины).

Такие схемы называются строительными блоками. В соответствии с этим оптимальное решение строится путем объединения наилучших из полученных на текущий момент частичных решений. Оператор скрещивания на двоичных стрингах не слишком часто уничтожает схемы малой определенной длины, однако ликвидирует схемы большой длины. Однако, не смотря на губительность операторов скрещивания и мутации для схем большого порядка и определенной длины, число обрабатываемых схем настолько велико, что даже в относительно малом числе особей в популяции генетический алгоритм дает неплохие результаты.

Несмотря на то, что для доказательства этой гипотезы были предприняты значительные усилия, строгого доказательства получено не было, и в большинстве нетривиальных приложений опираются на эмпирические результаты.

Далее проиллюстрируем приведенные результаты. Вернемся к примеру Голдберга определения \mbox{max}\ f(x)=x^2. В дополнение к имеющимся таблицам [рис 1.3](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=3#image.1.3) пусть имеется 3 конкретные схемы H_1=1\*\*\*\*\*, H_2=\*10\*\* и H_3=1\*\*\*0, описанные в [табл.1.2](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=9#table.1.2).

Рассмотрим сначала схему H_1. При репродукции стринги копируются с вероятностью, определенной согласно величине их фитнесс-функции. Из [табл. 1.1](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=5#table.1.1) видно, что стринги 2,4 покрываются схемой H_1. После выполнения репродукции мы видим в [табл. 1.3](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=9#table.1.3) (строка H_1), что получены три копии и стринг 3 также вошел в популяцию.

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица 1.2.** | | | |
| **Схема** | **Перед репродукцией** | **Представители стрингов** | **Среднее значение ЦФ схемы f(H)** |
| H_1 | 1\*\*\*\* | 2,4 | 469 |
| H_2 | \*10\*\* | 2,3 | 320 |
| H_3 | 1\*\*\*0 | 2 | 576 |

Проверим, соответствует ли это число фундаментальной теореме схем. Согласно теории мы ожидаем m*f(H)/\bar f копий. Вычисляя среднее значение ЦФ f(H_1), получаем \bar f(H_1)=(576+361)/2=468,5. Разделив это число на среднее значение ЦФ популяции \bar f=293 и умножив его на число m(H_1,t)=2 стрингов, покрываемых H_1 на шаге t, получаем ожидаемое число стрингов, покрываемых H_1 на шаге t+1, т.е. m(H_1,t+1)=2\cdot 468,5 / 293 = 3,20.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Таблица 1.3.** | | | | | | |
| **Схема** | **После репродукции** | | | **После всех операторов** | | |
|  | **Ожидаемое число стрингов** | **Действительное число стрингов** | **Представители стрингов** | **Ожидаемое число стрингов** | **Действительное число стрингов** | **Представители стрингов** |
| H_1 | 3,20 | 3 | 2, 3, 4 | 3,20 | 3 | 2, 3, 4 |
| H_2 | 2,18 | 2 | 2, 3 | 1,64 | 2 | 2, 3 |
| H_3 | 1,97 | 2 | 2, 3 | 0,0 | 1 | 4 |

Сравнивая это число с реальным числом копий 3, видим, что округление рассчитанного значения копий 3,2 дает их реальное число 3. Дальнейший анализ показывает, что в данном случае ОК не оказывает влияние на число стрингов, покрываемых схемой, так как определенная длина L(H_1)=0 предотвращает разрыв схемы. Далее при мутации с вероятностью P_{OM}=0,001 мы должны ожидать уменьшение числа стрингов на m*О(H_1)*P(OM)=3*1*0,001=0,003, что практически не оказывает влияния на окончательный результат. В итоге для схемы H_1 получаем ожидаемое увеличение числа особей до 3, что соответствует формуле (1.2).

Рассмотрим теперь схемы H_2 и H_3 с двумя фиксированными позициями. Схема H_2 имеет также два представителя в начальной популяции (2,3) и две копии в следующей промежуточной популяции. Вычисляем m(H_2)=2\cdot 320/293=2,18. Здесь f(H_2)=320– среднее значение ЦФ схемы, а \bar f=293– среднее значение ЦФ популяции. Для H_3 получаем только одно стринговое представление и m(H_3)=1\cdot 576/293=1,97. Здесь 576 – среднее значение ЦФ схемы.

Заметим, что для конкретной схемы H_2 представительство двух копий в популяции является хорошим результатом и он может случиться только один раз из 4-х возможных случаев (n-1=5-1=4). Схема H_2=*|1|0|*|* имеет реальную возможность дать новую копию. В результате, схема H_2 выживает с высокой вероятностью. Действительно, число стрингов, покрываемых H_2, равно m(H_2,t+1)=2,18\cdot0,75=1,64\approx 2.

Для схемы H_3 при высокой определенной длине L(H_3)=4 оператор ОК обычно разрушает эту схему. Поэтому ожидаемое число m(H_3,t+1)=0 соответствует числу, получаемому по формуле (1.4).

В соответствие с полученными результатами видно, что важнейшим аспектом является кодирование особей, которое должно обеспечить построение схем малого порядка малой определенной длины и со значениями фитнесс-функции выше среднего

Следующий простой пример показывает важность построения генома с учетом теорем схем.

Рассмотрим поиск максимума (для простоты при целочисленных значениях x и y) функции f(x,y)=x^2-y+17. Можно показать, что максимум f=66 достигается при значениях x=(111)_2=(7)_{10} и y=(0000)_2=(0)_{10}. Пусть x=x_2x_1x_0 и y= y_3y_2y_1y_0, где x_i,y_j\in \{0,1\}. Рассмотрим различное строение геномов. В первом случае пусть генотип представляет y_3x_2y_2x_1y_1x_0y_0, во втором генотип определим как y_3y_2y_1y_0\ x_2x_1x_0. Отметим, что в первом случае старшие разряды x и y расположены в генотипе близко, а во втором варианте наоборот – достаточно далеко. Поскольку желательна короткая определенная длина, генетический алгоритм с геномом, в котором старшие разряды расположены близко, должен быть лучше по сравнению с геномом, где эти разряды стоят далеко друг от друга. Для первого случая \frac{L(H)}{1-n}=\frac{1}{6} для схемы H_1=01\*\*\*\*\*, где содержатся значения, наиболее важные для искомого решения. Для второго случая схема при тех же значениях имеет вид H_2=0\*\*\*\*\*1, для которой \frac{L(H)}{1-n}=1 и поэтому необходимые значения старших битов x и y, скорее всего, при выполнении оператора кроссинговера будут "разорваны". Поэтому первый вариант генома предпочтительней.

1.7. Параметры генетических алгоритмов

Эффективность ГА зависит от ряда параметров, к которым относятся: мощность популяции, структура представления решения, вид генетических операторов кроссинговера и мутации, вероятности кроссинговера и мутации P_c и P_m и т.п..

Мощность популяции N является важнейшим параметром ГА, который критичен во многих приложениях. Чем больше N, тем больше разнообразие потенциальных решений (при хорошейсхеме инициализации, обеспечивающей однородное распределение частиц). Большое число особей позволяет покрыть большую часть пространства поиска за одну итерацию. С другой стороны, большое число особей повышает вычислительную сложность итерации и при этом ГА может выродиться в случайный параллельный поиск. Если N мало, то ГА работает быстро, но при этом увеличивается опасность преждевременной сходимости к локальному экстремуму. Большая мощность популяции увеличивает генофонд, но процесс поиска замедляется. Обычно полагают N\in [30;200].

На разных этапах работы ГА оптимальное значение N может быть различным. На начальном этапе N должно быть большим, а на заключительном N можно уменьшить. Большую роль играют также вид генетических операторов, которые представлены в следующем разделе. Кроме этого, не менее важны значения вероятностей кроссинговера и мутации P_c и P_m.

Для оптимизации, особенно мультимодальных функций, наиболее существенными являются две характеристики ГА:

* способность сходиться к оптимуму (локальному или глобальному) после нахождения области, содержащей этот оптимум;
* способность находить новые области в пространстве решений в поисках глобального оптимума.

Баланс между этими характеристиками ГА в значительной степени определяется значениями вероятности P_c и P_m, типом используемых генетических операторов (прежде всего кроссинговера). Увеличение значений P_c и P_m ведет к расширению пространства поиска. Обычно используют следующие значения вероятностей: P_c\in [0,5;1] и P_m\in [0,001;0,05]. В адаптивных генетических алгоритмах изменяются, прежде всего, вероятности кроссинговера и мутации P_c и P_m.

1.8. Преимущества и недостатки генетических алгоритмов

1.8.1. Концептуальная простота

Основным преимуществом ГА является их концептуальная простота. Рассмотрим снова блок-схему ГА, представленную на [рис.1.1](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=2#image.1.1) Основными шагами алгоритма являются: инициализация, оценка качества решения с помощью фитнесс-функции, итеративное изменение популяции путем отбора особей и применения генетических операторов. Отметим, что здесь не важна высокая точность оценки качества потенциальных решений – необходимо знать прежде всего их ранг (номер позиции по качеству решения). Информация о значениях градиента целевой функции здесь не нужна. Через ряд итераций (поколений) популяция (множество потенциальных решений проблемы) может сойтись асимптотически к оптимальному решению. Эффективность ГА зависит от способа кодирования решения, используемых генетических операторов, включая отбор особей, и начальной инициализации популяции.

1.8.2. Широкая применимость

ГА могут быть использованы при решении любой проблемы, которая может быть сформулирована как задача оптимизации. Они требуют разработки (или выбора) структуры данных для представления потенциального решения, показателя качества для оценки потенциального решения и генетических операторов, порождающих новые решения из старых. Пространство возможных решений может быть разбито на различные области, которые могут включать недостижимые зоны и зоны возможных изменений. Способ представления решения обычно выбирается инженером-разработчиком на основании его интуиции и опыта. В этом смысле процедура выбора кодирования потенциального решения независима в отличие от обычных численных методов, где, как правило, допускаются только непрерывные значения из определенного диапазона. Представление решения должно позволять генетическим операторам при изменении решений сохранять поведенческую связь между родителями и потомками. Небольшие изменения в структуре данных родителя должны вести к небольшим изменениям свойств потомков и наоборот, большие изменения у родителей должны вызывать значительные изменения свойств потомков, что должно способствовать эффективному поиску решения в пространстве поиска. Множество возможных изменений может регулироваться с помощью эффективного размера шага изменения в пространстве поиска, который может регулироваться разработчиком (или даже адаптироваться автоматически). Такой гибкий подход позволяет использовать по сути одну и ту же процедуру при решении задач как численной, так и комбинаторной оптимизации (в частности, целочисленной оптимизации и т.п.).

1.8.3. Менее жесткие требования при решении реальных задач

Реальные задачи оптимизации часто:

1. накладывают нелинейные ограничения;
2. требуют платежной функции, не связанной с наименьшей квадратичной ошибкой;
3. включают наличие шумов или случайных выбросов, которые не позволяют применять классические методы оптимизации [[17](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.17)].

Целевые функции для реальных проблем часто мультимодальны и градиентные методы сходятся быстро к локальным экстремумам, которые могут давать неудовлетворительные решения. Для простых задач, где поверхность отклика, например, является строго выпуклой, генетические алгоритмы проигрывают классическим по эффективности. Эксперименты показали, что для мультимодальных функций ГА дают лучшие результаты. В случае нелинейных ограничений классические методы даже при выпуклой поверхности могут давать некорректные результаты. Напротив, эволюционные методы могут непосредственно учитывать произвольные линейные и нелинейные ограничения.

1.8.4. Потенциальное использование априорных знаний и гибридизация с другими методами

При решении конкретной проблемы всегда целесообразно учесть в алгоритме проблемно-ориентированные априорные знания [[17](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.17)]. Специализированные алгоритмы, учитывающие такую информацию (но имеющие ограниченную область применения), как правило, существенно превосходят по характеристикам неспециализированные методы. Эволюционные алгоритмы по своей структуре легче позволяют учитывать априорные знания. Это может быть выражено, например, в виде специальной структуры данных для представления решений или специальных проблемно-ориентированных генетических операторов. Часто такая информация включается в фитнесс-функцию (например, физические или химические свойства вещества), что позволяет сфокусировать поиск решения в пространстве поиска.

Генетические алгоритмы могут комбинироваться с другими более традиционными методами. Известны работы, где на первом этапе оптимизации используются генетические алгоритмы совместно с градиентным методом, который применяется на заключительном этапе, когда уже найдена "зона интереса". Эти алгоритмы могут применяться совместно и параллельно. Отметим, что начальная популяция потенциальных решений может быть получена путем применения, например, жадных алгоритмов, а не эволюционных методов. Генетические алгоритмы часто используются для оптимизации и обучения искусственных нейронных сетей или нечетких продукционных систем. В этом случае часто удается преодолеть ограничения, связанные с традиционными подходами.

1.8.5. Параллелизм

Эволюция является высоко параллельным процессом, поскольку популяция состоит из множества особей, которые развиваются параллельно. Это позволяет расширить возможности применения эволюционных вычислений для решения все более сложных задач. Отметим, что основные вычислительные ресурсы в генетических алгоритмах используются при оценке значений фитнесс-функций, которые для различных особей могут выполняться параллельно, а последовательной является только процедура отбора. Поэтому эволюционные методы естественно реализуются в многопроцессорных распределенных компьютерных системах, которые в настоящее время все шире применяются в различных областях науки и техники и даже на бытовом уровне в многоядерных процессорах. В действительности при решении некоторых задач на распределенных вычислительных системах время решения в идеале может быть обратно пропорционально числу используемых задач. Это создает благоприятные условия для решения сложных задач высокой размерности за разумное время с помощью генетических алгоритмов.

1.8.6. Устойчивость к динамическим изменениям

Традиционные методы оптимизации неустойчивы к динамическим изменениям окружающей среды и часто требуют полного рестарта при таких изменениях для получения адекватного решения. Напротив, эволюционные алгоритмы могут быть использованы для адаптации потенциальных решений к изменившимся условиям. Полученная на момент изменения популяция дает базис для дальнейшего улучшения решений и в большинстве случаев нет необходимости проводить случайную реинициализацию.

1.8.7. Способность к самоорганизации

Большинство классических методов требуют начальной установки соответствующих параметров алгоритмов. Это также относится и к генетическим алгоритмам, которые зависят от множества параметров, таких как мощность популяции, вероятности кроссинговера и мутации, шаг мутации и т.п. Однако в эволюционных алгоритмах легче ввести самоадаптацию, когда в процессе поиска решения указанные параметры оптимизируются.

1.8.8. Решение проблем, для которых отсутствует опыт решений

Возможно, самым большим преимуществом генетических алгоритмов является их способность исследовать проблемы, для которых нет экспертов и соответствующего опыта решений. Следует отметить, что экспертные оценки достаточно часто используются при решении трудно формализуемых задач, но они иногда дают менее адекватные решения, чем автоматизированные методы. Существуют определенные проблемы с получением знаний у экспертов: они могут не согласиться на это, могут быть неквалифицированными, могут быть несовместимыми и просто ошибаться.

Исследования по искусственному интеллекту в настоящее время дали ряд интересных результатов, каждый из которых позволяет эффективно решать свой класс задач (например, хорошо играть в шахматы, или распознавать изображения символов и т.п.). Но большинство этих узких приложений требуют участия человека. Эти методы могут эффективно решать некоторые сложные проблемы, требующие высокого быстродействия, но они не могут конкурировать с человеческим интеллектом - "Они решают проблемы, но они не решают проблему как решать проблемы". Напротив, эволюционные алгоритмы дают метод решения проблемы, как решать проблемы при отсутствии экспертов (человеческого опыта) [[17](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.17)].

1.8.9. Недостатки ГА

Естественно ГА не свободны от недостатков. К ним можно отнести прежде всего следующие. Конфигурация ГА для решения сложных реальных задач не очевидна. Для решения конкретной задачи необходимо выбрать или разработать представление (кодирование) потенциального решения. Существует также проблема определения фитнесс-функции. Есть проблема выбора параметров ГА, таких как мощность популяции, вероятности генетических операторов и т.д. Нет эффективных критериев окончания работа алгоритма. ГА не могут использовать информацию о градиентах, что уменьшает их эффективность для классических задач. ГА не эффективны для гладких унимодальных (с одним экстремумом) функций. ГА не эффективны при поиске локальных экстремумов. ГА требуют достаточно больших вычислительных ресурсов. При решении мультимодальных задач бывают случаи преждевременной сходимости к локальным экстремумам и поэтому в общем случае не гарантируют нахождение глобального экстремума.

1.9. No Free Lunch теорема

Возникает естественный вопрос – существует ли некоторый лучший эволюционный алгоритм, который дает всегда лучшие результаты при решении всевозможных проблем? Например, можно ли выбрать генетические операторы и их параметры так, чтобы алгоритм давал лучшие результаты независимо от решаемой проблемы? К сожалению, ответ отрицательный – такого лучшего эволюционного алгоритма не существует! Это следует из известной No Free Lunch (NFL) теоремы (бесплатных завтраков не бывает), которая доказана относительно недавно, в 1996г.[[18](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.18),[19](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.19)] и вызвала оживленную дискуссию. Пусть P(d_m^y|f,m,a)- условная вероятность получения частного решения d_m после m итераций работы алгоритма a при целевой функции f. В этих терминах Вольперт и Макреди доказали так называемую No Free Lunch (NFL) теорему:

NFL теорема. Для любой пары алгоритмов a_1 и a_2 имеет место равенство

\sum_f P(d_m^y|f,m,a_1)=\sum_f P(d_m^y|f,m,a_2).

Таким образом, сумма условных вероятностей посещения в пространстве решений каждой точки d_m одинакова для множества всевозможных целевых функций независимо от используемого алгоритма. Из этого результата непосредственно следует, что при любой мере \Phi(d_m^y) производительности (характеристик сложности) алгоритма в среднем для всевозможных целевых функций f вероятность P(\Phi(d_m^y)|f,m,a) не зависит от алгоритма a. Другими словами, не существует лучшего алгоритма (эволюционного или любого другого) для решения всех проблем. Если алгоритм выигрывает по своим характеристикам при решении некоторого класса задач, то это неминуемо компенсируется проигрышем (худшими характеристиками) для остальных задач.

Эта теорема вызвала оживленную дискуссию у специалистов по эволюционным вычислениям и некоторое неприятие. Дело в том, что в семидесятых годах были предприняты значительные усилия по поиску лучших значений параметров и генетических операторов ГА. Исследовались различные генетические операторы, значения вероятностей выполнения кроссинговера и мутации, мощности популяции и т.д. Большинство этих исследований апробировалось на сложившихся в каждой проблемной области тестовых задачах. Но из NFL-теоремы следует, что полученные результаты справедливы только на использованных тестовых задачах, а не для произвольных задач. Все усилия найти лучший оператор кроссинговера или мутации, оптимальные значения их параметров при отсутствии ограничений для исследуемого класса задач не имеют смысла!

Каждому эволюционному алгоритму присуще некоторое представление, которое позволяет манипулировать с потенциальными решениями. NFL теорема утверждает, что не существует лучшего эволюционного алгоритма для решения всех проблем.

Для того чтобы разрабатываемый алгоритм решал поставленную задачу лучше, чем случайный поиск (который с точки зрения NFL теоремы является просто другим алгоритмом) необходимо в нем использовать (отразить) структуру (априорные знания) этой проблемы. Из этого следует, что такой алгоритм может не соответствовать структуре другой проблемы (и покажет для нее плохие результаты). Следует отметить, что недостаточно просто указать, что проблема имеет некоторую структуру - такая структура должна соответствовать разрабатываемому алгоритму. Более того, структура должна быть определена. Недостаточно, как это иногда бывает, сказать "Мы имеем дело с реальными проблемами, а не с всевозможными, поэтому NFL теорема не применима". Что значит структура реальной проблемы? Очевидно, что формальное описание такой структуры проблематично. Например, реальные проблемы нашего времени и столетней давности могут сильно отличаться. Следует отметить, что простое сужение области возможных проблем без идентификации соответствия между рассматриваемым множеством проблем и алгоритмом недостаточно для получения преимущества данного метода решения этих проблем по сравнению с другими.

NFL-теорема подтверждает, что разные алгоритмы имеют различную эффективность при решении разных задач. Например, классические методы оптимизации, как правило, более эффективны при решении линейных, квадратичных, строго выпуклых, унимодальных, разделяемых и других специальных классов проблем. С другой стороны, генетические алгоритмы часто успешно решают задачи там, где классические методы не работают – там, где целевые функции терпят разрывы, не дифференцируемы, мультимодальны (имеют много экстремумов), зашумлены и т.п. Обычно их эффективность и устойчивость выше там, где целевые функции имеют сложный (не стандартный) вид, что более характерно для решения реальных практических задач. Конечно, лучшим способом подтверждения эффективности алгоритма является доказательство его сходимости и оценки вычислительной сложности. Но, как правило, это возможно только в случае упрощенной постановки задачи. Другой альтернативой является проверка алгоритмов на тестовых задачах (benchmarks) данной проблемной области. К сожалению, в настоящее время не существует согласованного каталога таких задач для оценки старых или новых алгоритмов решения, хотя для многих типовых задач они уже сложились и широко используются.

Контрольные вопросы

1. Каковы "источники" ГА?
2. Какие генетические операторы используются в ГА?
3. Какую роль в ГА играет оператор репродукции (ОР)?
4. Опишите реализацию ОР в виде колеса рулетки и приведите пример его работы.
5. Придумайте другую реализацию ОР.
6. Опишите одноточечный оператор кроссинговера (ОК) и приведите пример его работы.
7. Предложите другую реализацию ОК.
8. Какую роль играет оператор мутации (ОМ)?
9. Опишите ОМ и приведите пример его работы.
10. Предложите другую реализацию ОМ.
11. Каковы основные параметры ГА?

Упражнения

1. Выполните программную реализацию простого ГА на одном из языков программирования для поиска экстремума заданной по варианту функции одной переменной ([табл. 1.5](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=14#table.1.5)).

Вид экстремума:

|  |  |
| --- | --- |
| **Таблица 1.4.** | |
| **Вариант** | **Вид экстремума** |
| \le 15 | Максимум |
| > 15 | Минимум |

1. Исследовать зависимость времени поиска, числа поколений (генераций), точности нахождения решения от основных параметров генетического алгоритма:
   * число особей в популяции
   * вероятность кроссинговера, мутации.
2. Вывести на экран график данной функции с указанием найденного экстремума для каждого поколения
3. Сравнить найденное решение с действительным.

|  |  |  |
| --- | --- | --- |
| **Таблица 1.5.** | | |
| **Вариант** | **Вид функции** | **Промежуток поиска решения** |
| 1 | (1,85-x)*\cos(3,5x-0,5) | x\in [-10,10] |
| 2 | \cos(\exp(x))/\sin(\ln(x)) | x\in [2,4] |
| 3 | \sin(x)/x^2 | x\in [3.1,20] |
| 4 | \sin(2x)/x^2 | x\in [-20,-3.1] |
| 5 | \cos(2x)/x^2 | x\in [-20,-2.3] |
| 6 | (x-1)\cos(3x-15) | x\in [-10,10] |
| 7 | \ln(x)\cos(3x-15) | x\in [1,10] |
| 8 | \cos(3x-15)/|x|=0 | x\in [-10,-0.3),(0.3,10]\\x\in[-0.3,0.3] |
| 9 | \cos(3x-15)*x | x\in [-9.6,9.1] |
| 10 | \sin(x)/(1+\exp(-x)) | x\in [0.5,10] |
| 11 | \cos(x)/ (1+\exp(-x) | x\in [0.5,10] |
| 12 | (\exp(x)-\exp(-x))\cos(x)/(\exp(x)+\exp(-x)) | x\in [-5,5] |
| 13 | (\exp(-x)-\exp(x))\cos(x)/(\exp(x)+\exp(-x)) | x\in [-5,5] |
| 14 | \cos(x-0,5)/|x| | x\in [-10,0),(0,10],\min |
| 15 | \cos(2x)/|x-2| | x\in [-10,2),(2,10],\max |

Краткие итоги:

* представлено описание простого ГА;
* введены основные генетические операторы – репродукции, кроссинговера и мутации;
* описан концептуальный смысл фитнесс-функции и обсуждено ее отличие от целевой функции;
* представлены теоретические основы ГА (теория схем, фундаментальная теорема ГА);
* обсуждены преимущества и недостатки ГА.
* Просмотр
* [Редактировать](https://moodle.bgpu.ru/mod/lesson/edit.php?id=71572)
* [Отчеты](https://moodle.bgpu.ru/mod/lesson/report.php?id=71572)
* [Оценить эссе](https://moodle.bgpu.ru/mod/lesson/essay.php?id=71572)

Генетические алгоритмы

Предисловие

Среди множества проблем, которые возникают перед исследователями как в области теории, так и в многочисленных практических приложениях значительную долю составляют так называемые оптимизационные проблемы. Понятие оптимальности, по-видимому, знакомо почти каждому и вошло в практику большинства предметных областей. С оптимизационной проблемой мы сталкиваемся каждый раз, когда возникает необходимость выбора из некоторого множества возможных решений наилучшего по определенным критериями и, как правило, удовлетворяющего заданным условиям и ограничениям. Само понятие оптимальности получает совершенно строгое толкование в математических теориях, однако в других областях оно может интерпретироваться скорее содержательно. Тем не менее различие между строгим и содержательным понятиями оптимальности, как правило, очень незначительно.

Существуют классы оптимизационных задач, решение которых удается находить с помощью достаточно эффективных методов, вполне приемлемых по трудоемкости. Вместе с тем имеются и такие классы оптимизационных задач (так называемые NP- полные задачи), решение которых невозможно найти без полного перебора вариантов. В частности, к числу последних относятся многие разновидности задач многокритериальной оптимизации. Известно, что при большой размерности этих задач реализация перебора вариантов практически невозможна из-за чрезвычайно больших временных затрат.

В этой ситуации альтернативным походом к решению упомянутых задач является применение методов, базирующихся на методологии эволюционных вычислений. Предлагаемое пособие содержит изложение основ эволюционных вычислений и их приложений к решению различных проблем, включая проблемы экономики, прогнозирования финансовых рынков, инвестиций, бизнеса, комбинаторной оптимизации, сложных задач в различных технических разработках и т.п. Эффективность различных методов в рамках эволюционного подхода подтверждается многочисленными данными, касающимися достигаемым реальным эффектом. При этом хотя объем вычислений может оказаться большим, но скорость, с которой он растет при увеличении размерности задачи, обычно меньше, чем у остальных известных методов. Отметим, что после того как компьютерные системы стали достаточно быстродействующими и недорогими, эволюционные методы превратились в важный инструмент поиска близких к оптимальным решений задач, которые до этого считались неразрешимыми.

Основной целью учебного пособия является систематическое изложение перспективного направления в области теории и практики искусственного интеллекта. Пособие рассчитано на студентов и аспирантов вузов, обучающихся по направлениям, связанным с прикладной информатикой и математикой, компьютерными и программными системами, с интеллектуальными системами обработки информации.

Освоение представленного в пособии материала предполагает знакомство читателя с математикой и информатикой в объеме первых двух курсов технического вуза, основами дискретной математики и методов оптимизации.

Введение

В настоящее время оформилось и успешно развивается новое направление в теории и практике искусственного интеллекта – эволюционные вычисления (ЭВ). Этот термин обычно используется для общего описания алгоритмов поиска, оптимизации или обучения, основанных на некоторых формализованных принципах естественного эволюционного отбора. Особенности идей эволюции и самоорганизации заключаются в том, что они являются плодотворными и полезность их применения не только для биологических систем перманентно подтверждается. Эти идеи в настоящее время с успехом используются при разработке многих технических и, в особенности, программных систем.

Высокая согласованность и эффективность работы элементов биологических систем приводила целый ряд исследователей к естественной мысли о возможности использования принципов биологической эволюции для оптимизации важных для приложений систем, природа которых отлична от биологической. Так, в 1966 году Фогель Л., Оуенс С. и Уолш М. в [[1](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.1)] подобную идею использовали для построения схемы эволюции логических автоматов, решающих задачи прогноза. В 1975 году была опубликована основополагающая работа Дж. Холланда [[2](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.2)], в которой был предложен генетический алгоритм, развивающий ту же идею. Д.Гольдберг, ученик Дж. Холланда, в работе [[3](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.3)] , выполненной в Мичиганском университете, успешно развил и расширил области его применения.

В 60-х годах прошлого века в Германии Рохенберг И., Швефель Г.-П., и др. [[4](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.4)] начали разработку так назывемой эволюционной стратегии. Перечисленные работы послужили толчком и основой развития прикладного направления, которое можно назвать эволюционными алгоритмами. К их числу, помимо упомянутых генетических алгоритмов и эволюционных стратегий, относятся также эволюционное программирование, ориентированное на оптимизацию функций без использования рекомбинаций, и генетическое программирование, использующее эволюционные идеи для оптимизации компьютерных программ. Основополагающая работа по эволюционному программированию принадлежит Дж. Коза [[5](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.5)] из Массачусетского технологического института.

Отметим, что аналогичные исследования успешно проводились отечественными учеными еще в Советском Союзе. Так, значительный вклад в развитие указанных направлений внесли Ивахненко А.Г. [[6](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.6)], Цыпкин Я.З. [[7](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.7)], Расстригин Л.А. [[8](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.8)] и др. В настоящее время исследования по ЭВ активно ведутся Букатовой [[9](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.9),[10](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.10)], Курейчиком В.М. [[11](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.11),[12](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.12),[13](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.13)], их сотрудниками и учениками, а также многими другими исследователями.

Учебное пособие состоит из пяти частей.

В первой части "Основы генетических алгоритмов" (разделы пособия 1-5) изложены идейная сторона простых генетических алгоритмов, их математические основы, подробно описаны примеры построения генетических алгоритмов для решения конкретных задач комбинаторной оптимзации и, наконец, представлены современные модификации и обобщения этих алгоритмов.

Во второй части "Генетическое программирование и машинное обучение" (разделы 6-7) описана концепция компьютерного синтеза программ (с различными структурами их представления – линейными, древовидными, графоподобными) с использованием генетических алгоритмов. Изложены два основных подхода при машинном обучении - Мичиганский и Питтсбургский.

В третьей части "Вероятностные генетические алгоритмы"(раздел 8) приведен другой – вероятностный подход в эволюционных вычислениях, где популяция представляется вектором вероятностей.

В четвертой части "Эволюционные стратегии" (раздел 9) представлена оригинальная парадигма, основанная на эволюции популяции потенциальных решений. Ее принципиалное отличие состоит в том, что генетические операторы используются здесь на уровне фенотипа, а не генотипа, как это было в генетических алгоритмах.

В пятой части "Эволюционное программирование" (раздел 10) содержится описание основанного Фогелем Л.Дж. гибкого подхода в эволюционных вычислениях. В нем форма представления потенциального решения и генетические операторы адаптируются к решаемой проблемы в достаточно широких пределах. В частности, в качестве особи в процессе эволюции Фогелем используются конечные автоматы и целью эволюции является способность решения задач прогнозирования.

В шестой части "Роевой интеллект" (разделы 11-12) описаны принципы и методы оптимизации, базирующиеся на коллективном поведении децентрализованных самоорганизующихся систем. Такие системы представляются, например, в виде графа, содержащего множество вершин (агентов), локально взаимодействующих между собой и с окружающей средой. Имеющиеся экспериментальные данные подверждают эффективность роевого интеллекта (муравьиных, пчелиных, роевых и т.п. алгоритмов) для решения многих оптимизационных задач.

Заканчивая введение заметим, что в пособии представлены не все аспекты быстро развивающихся эволюционных вычислений. Мы ограничились здесь расмотрением только наиболее важных и принципиальных с нашей точки зрения моментов эволюционных вычислений, хотя по этому поводу могут быть и другие мнения.

1. Введение в генетические алгоритмы

В этом разделе описывается концепция простого генетического алгоритма (ГА), ориентированного на решение различных оптимизационных задач. Вводятся и содержательно описываются понятия, используемые в теории и приложениях ГА. Приводится фундаментальная теорема ГА и излагается теория схем, составляющие теоретическую базу ГА. Обсуждаются концептуальные вопросы, касающиеся преимуществ и недостатков ГА.

1.1. Простой генетический алгоритм

Основы теории генетических алгоритмов сформулированы Дж. Г.Холландом в основополагающей работе [[2](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.2)] и в дальнейшем были развиты рядом других исследователей. Наиболее известной и часто цитируемой в настоящее время является монография Д.Голдберга [[3](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.3)], где систематически изложены основные результаты и области практического применения ГА.

ГА используют принципы и терминологию, заимствованные у биологической науки – генетики. В ГА каждая особь представляет потенциальное решение некоторой проблемы. В классическом ГА особь кодируется строкой двоичных символов – хромосомой, каждый бит которой называется геном. Множество особей – потенциальных решений составляет популяцию. Поиск оптимального или субоптимального решения проблемы выполняется в процессе эволюции популяции, т.е. последовательного преобразования одного конечного множества решений в другое с помощью генетических операторов репродукции, кроссинговера и мутации. ЭВ используют механизмы естественной эволюции, основанные на следующих принципах:

1. Первый принцип основан на концепции выживания сильнейших и естественного отбора по Дарвину, который был сформулирован им в 1859 году в книге "Происхождение видов путем естественного отбора". Согласно Дарвину особи, которые лучше способны решать задачи в своей среде, чаще выживают и чаще размножаются (репродуцируют). В генетических алгоритмах каждая особь представляет собой решение некоторой проблемы. По аналогии с этим принципом особи с лучшими значениями целевой (фитнесс) функции имеют большие шансы выжить и репродуцировать. Формализацию этого принципа, как мы увидим далее, реализует оператор репродукции.
2. Второй принцип обусловлен тем фактом, что хромосома потомка состоит из частей, полученных из хромосом родителей. Этот принцип был открыт в 1865 году Г.Менделем. Его формализация дает основу для оператора скрещивания (кроссинговера).
3. Третий принцип основан на концепции мутации, открытой в 1900 году де Вре. Первоначально этот термин использовался для описания существенных (резких) изменений свойств потомков и приобретение ими свойств, отсутствующих у родителей. По аналогии с этим принципом генетические алгоритмы используют подобный механизм для резкого изменения свойств потомков и, тем самым, повышают разнообразие (изменчивость) особей в популяции (множестве решений).

Эти три принципа составляют ядро ЭВ. Используя их, популяция (множество решений данной проблемы) эволюционирует от поколения к поколению.

Эволюцию искусственной популяции – поиск множества решений некоторой проблемы, формально можно описать в виде алгоритма, который представлен на [рис.1.1.](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=2#image.1.1)

ГА получает множество параметров оптимизационной проблемы и кодирует их последовательностями конечной длины в некотором конечном алфавите (в простейшем случае в двоичном алфавите "0" и "1").

Предварительно простой ГА случайным образом генерирует начальную популяцию хромосом (стрингов). Затем алгоритм генерирует следующее поколение (популяцию) с помощью трех следующих основных генетических операторов:

1. оператора репродукции (ОР);
2. оператора скрещивания (кроссинговера, ОК);
3. оператора мутации (ОМ).

Генетические алгоритмы – это не просто случайный поиск, они эффективно используют информацию, накопленную в процессе эволюции.

В процессе поиска решения необходимо соблюдать баланс между "эксплуатацией" полученных на текущий момент лучших решений и расширением пространства поиска. Различные методы поиска решают эту проблему по-разному.

Например, градиентные методы практически основаны только на использовании лучших текущих решений, что повышает скорость сходимости с одной стороны, но порождает проблему локальных экстремумов с другой. В полярном подходе случайные методы поиска используют все пространство поиска, но имеют низкую скорость сходимости. В ГА предпринята попытка объединить преимущества этих двух противоположных подходов. При этом операторы репродукции и кроссинговера делают поиск направленным. Широту поиска обеспечивает то, что процесс ведется на множестве решений – популяции и используется оператор мутации.

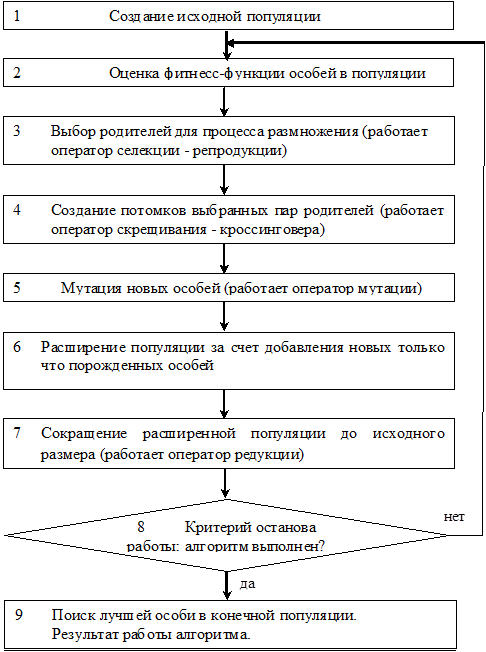


Рис. 1.1. Простой генетический алгоритм

В отличие от других методов оптимизации ГА оптимизируют различные области пространства решений одновременно и более приспособлены к нахождению новых областей с лучшими значениями целевой функции за счет объединения квазиоптимальных решений из разных популяций.

Упомянутые генетические операторы являются математической формализацией приведенных выше трех основополагающих принципов Ч.Дарвина, Г.Менделя и де Вре естественной эволюции. Каждый из функциональных блоков ГА на [рис. 1.1.](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=2#image.1.1) может быть реализован различными способами. Но сначала на простом примере мы рассмотрим основные моменты классического ГА.

1.2. Генетические операторы

Рассматриваемый ниже пример заимствован из популярной монографии Голдберга [[3](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.3)] и состоит в поиске целочисленного значения (для простоты) x на отрезке от [0,31], при котором функции y=x^2 принимает максимальное значение.

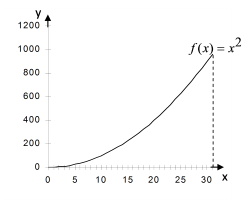


Рис. 1.2. Пример функции

Начальный этап работы ГА для данного примера приведен в верхней таблице (репродукция) [рис.1.3](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=3#image.1.3) Здесь особи начальной популяции (двоичные коды значений переменных x- столбец 2) сгенерированы случайным образом. Двоичный код значения х называется хромосомой (она представляет генотип). Популяция образует множество потенциальных решений данной проблемы. В третьем столбце представлены их десятичные значения (фенотип). Далее на этом примере проиллюстрируем работу трех основных генетических операторов.

1.2.1. Репродукция

Репродукция – это процесс, в котором хромосомы копируются в промежуточную популяцию для дальнейшего "размножения" согласно их значениям целевой (фитнесс-) функции. При этом хромосомы с лучшими значениями целевой функции имеют большую вероятность попадания одного или более потомков в следующее поколение.

Очевидно, оператор репродукции (ОР) является искусственной версией естественной селекции – выживания сильнейших по Ч. Дарвину. Этот оператор представляется в алгоритмической форме различными способами (подробнее различные варианты ОР будут рассмотрены далее). Самый простой (и популярный) метод реализации ОР – построение колеса рулетки, в которой каждая хромосома имеет сектор, пропорциональный по площади значению ее целевой функции. Для нашего примера "колесо рулетки" имеет вид, представленный на [рис.1.4.](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=3#image.1.4)

Для селекции хромосом используется случайный поиск на основе колеса рулетки. При этом колесо рулетки вращается и после останова ее указатель определяет хромосому для селекции в промежуточную популяцию (родительский пул).

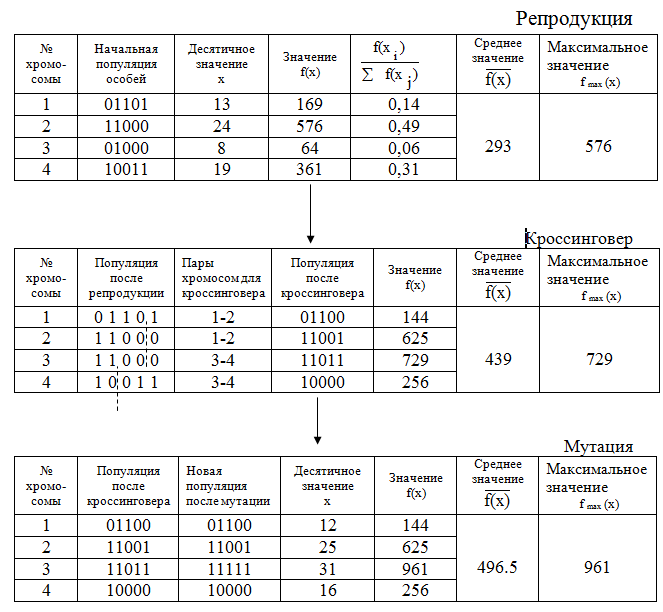


Рис. 1.3. Эволюция популяции

Очевидно, что хромосома, которой соответствует больший сектор рулетки, имеет большую вероятность попасть в следующее поколение. В результате выполнения оператора репродукции формируется промежуточная популяция, хромосомы которой будут использованы для построения поколения с помощью операторов скрещивания.

В нашем примере выбираем хромосомы для промежуточной популяции, вращая колесо рулетки 4 раза, что соответствует мощности начальной популяции. Величину \frac{f(x_i)}{\sum f(x_j)} обозначим как P(x_i), тогда ожидаемое количество копий i-ой хромосомы определяется значением M=P(x_i)*N, где N-мощность популяции. Число копий хромосомы, переходящих в следующее поколение, иногда определяется и так: \tilde M=\frac{f(x_i)}{\bar f(x)}, где \bar f(x)- среднее значение хромосомы в популяции.

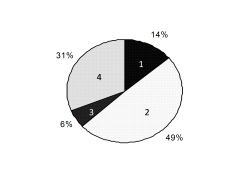


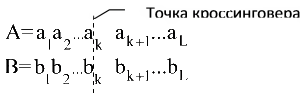
Рис. 1.4. Колесо рулетки

Расчетные числа копий хромосом по приведенной формуле следующие: хромосома 1 - 0,56; хромосома 2 - 1,97; хромосома 3 – 0,22; хромосома 4 – 1,23. В результате, в промежуточную популяцию 1-я хромосома попадает в одном экземпляре, 2-я – в двух, 3-я – совсем не попадает, 4-я – в одном экземпляре. Полученная промежуточная популяция является исходной для дальнейшего выполнения операторов кроссинговера и мутации.

1.2.2 Оператор кроссинговера (скрещивания)

Одноточечный или простой оператор кросинговера (ОК) с заданной вероятностью P_c выполняется в три этапа:

1-й этап. Две хромосомы (родители) выбираются случайно (или одним из методов, рассмотренных далее) из промежуточной популяции, сформированной при помощи оператора репродукции (ОР).



2-й этап. Случайно выбирается точка скрещивания - число k из диапазона [1,n-1], где n– длина хромосомы (число бит в двоичном коде).

Две новых хромосомы A', B' (потомки) формируются из A и B путем обмена подстрок после точки скрещивания:

A'=a_1 a_2\dots a_k\ b_{k+1}\dots b_L\\B'=b_1 b_2\dots b_k\ a_{k+1}\dots a_L

Например, рассмотрим выполнение кроссинговера для хромосом 1 и 2 из промежуточной популяции:

A=0\ 1\ 1\ 0\ 1\\B=1\ 1\ 0\ 0\ 0\\1\le k\le 4, k=4\\A'=0\ 1\ 1\ 0\ 0\\B'=1\ 1\ 0\ 0\ 1\\

Следует отметить, что ОК выполняется с заданной вероятностью P_c (отобранные два родителя не обязательно производят потомков). Обычно величина вероятности полагается равной P_c\approx 0,5.

Таким образом, операторы репродукции и скрещивания очень просты – они выполняют копирование особей и частичный обмен частей хромосом. Продолжение нашего примера представлено на [рис.1.3](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=3#image.1.3) во второй таблице (кроссинговер).

Сравнение с предыдущей таблицей показывает, что в промежуточной популяции после скрещивания улучшились все показатели популяции (среднее и максимальное значения целевой функции - ЦФ).

1.2.3 Мутация

Далее согласно схеме классического ГА с заданной вероятностью P_M выполняется оператор мутации. Иногда этот оператор играет вторичную роль. Обычно вероятность мутации мала - P_m\approx 0,001

Оператор мутации (ОМ) выполняется в два этапа:

1-й этап. В хромосоме A=a_1 a_2\dots a_L случайным образом выбирается k-ая позиция (бит) (1\le k\le n).

2-й этап. Производится инверсия значения гена в k-й позиции: a'_k=\bar a_k

Например, для хромосомы 11011 выбирается k=3 и после инверсии значения третьего бита получается новая хромосома – 11111. Продолжение нашего примера представлено в третьей таблице (мутация) [рис.1.3](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=3#image.1.3) образом, в результате применения генетических операторов найдено оптимальное решение x=31.

В данном случае, поскольку пример искусственно подобран, мы нашли оптимальное решение за одну итерацию. В общем случае ГА работает до тех пор, пока не будет выполнен критерий окончания процесса поиска и в последнем полученном поколении определяется лучшая особь, которая и принимается в качестве решения задачи.

1.3. Представление вещественных решений в двоичной форме

В предыдущем примере мы рассматривали только целочисленные решения. Обобщим ГА на случай вещественных чисел на примере функции f(x)=(1,85-x)*\cos(3,5x-0,5), представленной на [рис.1.5](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=4#image.1.5) [[14](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.14)]. Рассматривается та же [задача](https://moodle.bgpu.ru/mod/assign/view.php?id=109703): необходимо найти вещественное x\in [-10,+10], которое максимизирует f(x), т.е. такое x_0, для которого f(x_0)\ge f(x) для всех x\in [-10,+10].

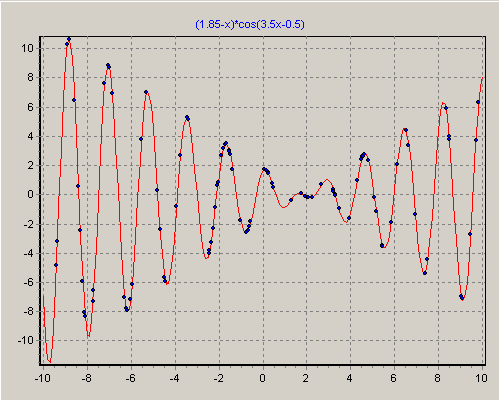


Рис. 1.5. Пример функции с популяцией особей в начале эволюции

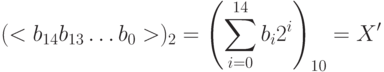
Для решения этой задачи с помощью ГА будем использовать представления вещественного решения (хромосомы) в виде двоичного вектора [[15](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.15)], который применяется в классическом простом ГА. Его длина зависит от требуемой точности решения, которую в данном случае положим, например, равной 3 знакам после запятой.

Поскольку отрезок области решения имеет длину 20, для достижения заданной точности отрезок [a,c]=[-10,+10] должен быть разбит на равные части (маленькие отрезки), число которых должно быть не менее 20\*1000. В качестве двоичного представления используем двоичный код номера (маленького) отрезка. Этот код позволяет определить соответствующее ему вещественное число, если известны границы области решения. Отсюда следует, что двоичный вектор для кодирования вещественного решения должен иметь 15 бит, поскольку

16384=2^{14}<20000\le2^{15}=32768

Отсюда следует, что для обеспечения необходимой точности требуется разбить отрезок [-10,+10] на 32768 частей. Отображение из двоичного представления (b_{14} b_{13}\dots b_0)\ (b_i\in \{0,1\} в вещественное число из отрезка [a,c]=[-10,+10] выполняется в два шага.

1. Перевод двоичного числа в десятичное:



1. Вычисление соответствующего вещественного числа x:

x=a+x'\cdot\frac{(c-a)}{2^{15}-1}=-10+x'\cdot\frac{20}{2^{15}-1},

, где (– 10) левая граница области решения. Естественно хромосомы (000000000000000) и (111111111111111) представляют границы отрезка –10 и +10 соответственно.

Очевидно, при данном двоичном представлении вещественных чисел можно использовать классический простой ГА. На [рис.1.6](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=4#image.1.6),[рис.1.7](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=4#image.1.7),[рис.1.8](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=4#image.1.8) представлено расположение особей (потенциальных решений) на различных этапах ГА в процессе поиска решения[[14](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.14)]. На [рис.1.5](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=4#image.1.5) показана начальная популяция потенциальных решений, которая равномерно покрывает область поиска решения. Далее явно видно, как постепенно с увеличением номера поколения особи "конденсируются" в окрестностях экстремумов и, в конечном счете, находится лучшее решение.

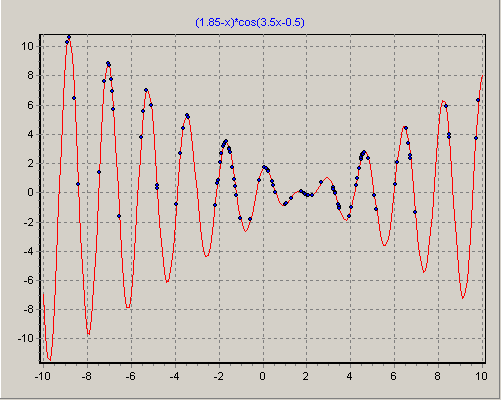


Рис. 1.6. Начальная "конденсация" особей популяции в окрестностях экстремумов

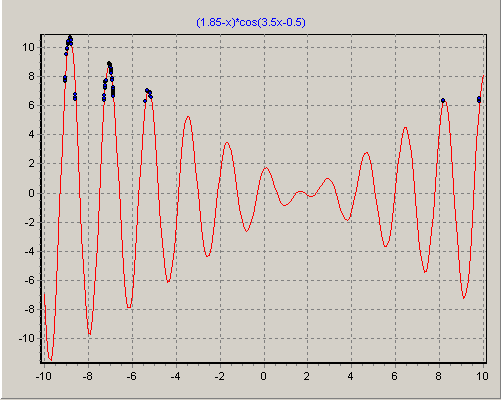


Рис. 1.7. "Конденсация" особей в окрестностях экстремумов

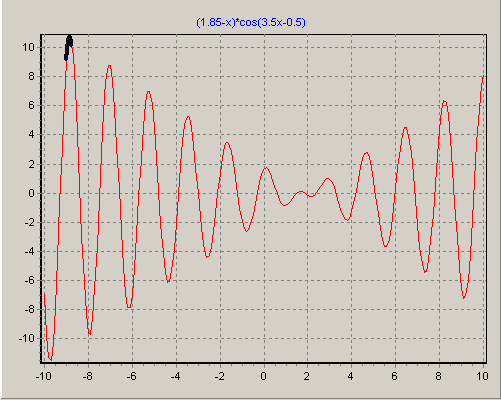


Рис. 1.8. Положение особей популяции в конце эволюции

Для сокращения длины хромосом иногда применяют логарифмическое кодирование, при котором первый бит (a) кодовой последовательности используется для знака показательной функции, второй бит (b) – для знака степени этой функции, и остальные биты (str) представляют значение самой степени [[16](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.16)]. Таким образом, двоичный код <a\ b\ str> представляет вещественное число (-1)^a e^{(-1)^b [str]_{10}}. Здесь [str]_{10} означает десятичное число, представленное двоичным кодом str.

Например, двоичный код <10101> представляет вещественное число (-1)^0 e^{(-1)^1 [101]_{10}} =e^{-6}=0,002478752. Следует отметить, что при таком кодировании пять битов позволяет кодировать вещественные числа из интервала [-e^7,e^7], что значительно больше, чем это позволяет, например, метод кодирования, представленный ранее.

1.4. Использование кода Грея в ГА

Рассмотренное двоичное представление вещественного числа имеет существенный недостаток: расстояние между вещественными числами (на числовой оси) часто не соответствует расстоянию (по Хеммингу) между их двоичными представлениями. Поэтому желательно получить двоичное представление, где близкие расстояния между хромосомами (двоичными представлениями) соответствовали близким расстояниям в проблемной области (в данном случае расстоянию на числовой оси) [[3](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.3),[14](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.14),[15](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.15)]. Это можно сделать, например, с помощью кода Грея. В [таблице 1.1](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=5#table.1.1) приведен для примера код Грея для 4-х битовых слов.

|  |  |
| --- | --- |
| **Таблица 1.1.** | |
| **Двоичный код** | **Код Грея** |
| 0000 | 0000 |
| 0001 | 0001 |
| 0010 | 0011 |
| 0011 | 0010 |
| 0100 | 0110 |
| 0101 | 0111 |
| 0110 | 0101 |
| 0111 | 0100 |
| 1000 | 1100 |
| 1001 | 1101 |
| 1010 | 1111 |
| 1011 | 1110 |
| 1100 | 1010 |
| 1101 | 1011 |
| 1110 | 1001 |
| 1111 | 1000 |

Заметим, что в коде Грея соседние двоичные слова отличаются на один бит (расстояние по Хеммингу равно 1).

Рассмотрим алгоритмы преобразования двоичного числа \bar b=<b_1,\dots,b_m> в код Грея \bar g=<g_1,\dots,g_m> и наоборот.

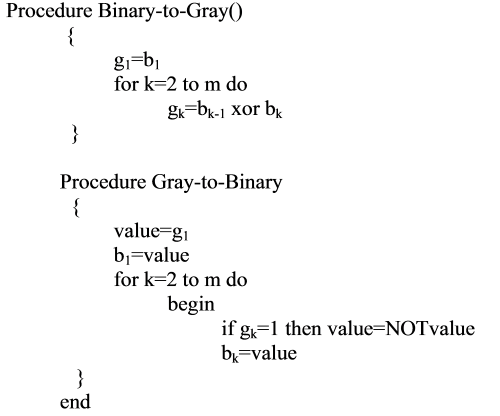
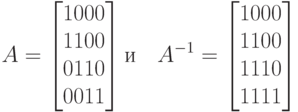


Рис. 1.9. Преобразование в код Грея

Здесь параметр m определяет разрядность двоичного числа. Существует и другая, матричная, процедура преобразования в код Грея. Например, для m=4 матрицы



позволяют выполнять следующие преобразования:

\bar g=A\cdot\bar b\ \mbox{и}\ \bar b=A^{-1}\bar g

где умножение матриц выполняются в арифметике по mod 2.

Отметим, что применение кода Грея прежде всего оправдано при использовании операторов мутации.

1.5. Фитнесс-функция

Как видно из основной блок-схемы ГА на [рис.1.1](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=2#image.1.1), каждая особь популяции (потенциальное решение проблемы) оценивается путем вычисления значения фитнесс-функции. Следует отметить, что в общем случае целевая функция (ЦФ) и фитнесс-функция могут различаться. Целевая функция предназначена для оценки характеристик особи относительно конечной цели (например, экстремумов). Фитнесс-функция предназначена, прежде всего, для отбора особей для дальнейшей репродукции и здесь важны характеристики качества одной особи относительно других особей. После декодирования хромосомы, где выполняется преобразование "генотип \to фенотип" (например, двоичный код преобразуется в вещественное число), полученное значения далее используется в качестве аргумента для фитнесс-функции. Далее для каждой особи популяции вычисляются значения фитнесс-функции, которые ранжируют эти особи относительно друг друга в смысле перспективности получения из них хорошего решения.

Определение соответствующей фитнесс-функции является решающим для корректной работы ГА. В частности, вид фитнесс-функции может зависеть от накладываемых ограничений при решении оптимизационных задач. Отметим, что операторы кроссинговера и мутации не учитывают, попадают ли вновь построенные особи-потомки в область допустимых решений, которая обусловлена накладываемыми ограничениями.

В ГА используются четыре основных метода для учета накладываемых ограничений при решении оптимизационных задач. Вероятно, простейшим способом является метод отклонения (отбрасывания), где недопустимые хромосомы (не удовлетворяющие ограничениям) исключаются из дальнейшей эволюции. Второй метод основан на использовании процедуры восстановления, которая преобразует полученное недопустимое решение в допустимое. Другой альтернативой является применение проблемно-ориентированных генетических операторов, которые порождают только допустимые решения.

Рассмотренные методы не строят недопустимых решений. Но это не всегда дает хорошие результаты. Например, в том случае, когда оптимальные решения лежат на границе допустимой области, указанные методы могут давать неоптимальные решения. Одним из возможных вариантов преодоления этой проблемы является выполнение процедуры восстановления только для некоторого подмножества решений (например, 10% особей).

Для решения оптимизационных задач со сложными ограничениями иногда позволяют вести поиск решения и в недопустимых областях. Реализуется это подход часто с помощью метода штрафных функций, что позволяет расширить пространство поиска решений. Следует отметить, что часто недопустимая точка, близкая к оптимальному решению, содержит больше полезной информации, чем допустимая точка, далекая от оптимума. С другой стороны, построение штрафных функций является достаточно сложной проблемой, которая сильно зависит от решаемой задачи. Обычно нет априорной информации о расстоянии до оптимальных точек, есть только расстояние до границы области допустимых решений. Поэтому, как правило, штрафные функции используют расстояние до границ допустимой области. Штрафы, основанные на нарушении отдельных ограничений, работают обычно не очень хорошо.

Разработаны два основных способа построения штрафных функций со штрафным термом: аддитивная и мультипликативная формы. В первой форме функция представляется в виде g(x)=f(x)+p(x), где при максимизации для допустимых точек p(x)=0 и в противном случае p(x)<0. Максимум значения p(x) по абсолютной величине не может быть больше, чем минимальное значение f(x) по абсолютной величине для любой генерации, чтобы избежать отрицательных фитнесс-значений. Мультипликативная форма представляет функцию в виде g(x)=f(x)\cdot p(x), где при максимизации p(x)=1 для допустимых точек и 0\le p(x)<1 в противном случае.

При этом штрафной терм должен изменяться не только в зависимости от степени нарушения ограничения, но и от номера поколения ГА. Наряду с нарушением ограничения, штрафной терм обычно содержит штрафные коэффициенты (по одному для каждого ограничения). На практике большую роль играют значения этих коэффициентов. Маленькие значения коэффициентов могут привести к недопустимым значениям решения, в то время как большие значения полностью отвергают недопустимые подпространства. В среднем абсолютные значения целевой и штрафной функции должны быть соизмеримы. При таком подходе параметры штрафной функции можно включить в параметры ГА, что позволяет разработать адаптивный метод, где значения коэффициентов регулируются в процессе поиска решения.

В целом на выбор (построение) фитнесс-функции оказывают влияние следующие факторы:

* тип задачи – максимизация или минимизация;
* содержание шумов окружающей среды в фитнесс-функции;
* возможность динамического изменения фитнесс-функции в процессе решения задачи;
* объем допустимых вычислительных ресурсов – допускается ли использовать более точные методы и значительные ресурсы, или возможны только приближенные аппроксимации, не требующие больших ресурсов;
* насколько различные значения для особей должна давать фитнесс-функция для облегчения отбора родительских особей;
* должна ли она содержать ограничения решаемой задачи;
* может ли она совмещать различные подцели (например, для многокритериальных задач).

В ГА фитнесс-функция используется в виде черного ящика: для данной хромосомы она вычисляет значение, определяющее качество данной особи. Внутри она может быть реализована по-разному: в виде математической функции, программы моделирования (в том числе имитационного), нейронной сети, или даже экспертной оценки.

Для некоторых задач оценку значений фитнесс-функции можно выполнять с помощью объектно-ориентированной имитационной модели. Взаимодействие такой модели с ГА показано на [рис.1.10](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=6#image.1.10).

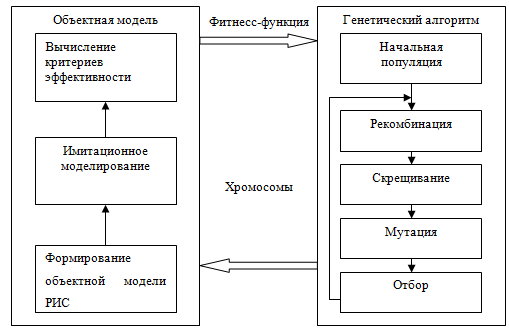


Рис. 1.10. Схема взаимодействия объектной модели с ГА.

1.6. Теория схем

Теоретические основы ГА составляют двоичное стринговое представление решений (хромосом) и понятие схемы (schema) или шаблона. Это понятие было ведено Холландом для определения множества хромосом, которые обладают некоторыми общими свойствами, то есть в каком- то смысле подобны друг другу.Термин "схема" согласно Холланду [[2](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.2)] есть шаблон, описывающий подмножество стрингов, имеющих одинаковые значения в некоторых позициях. Для этого вводится новый троичный алфавит {0, 1, \*}, где \* означает неопределенное значение (0 или 1, т.е. неизвестно что именно). Например, схема (0\*0001) соответствует двум стрингам {000001, 010001}, а (\*0110\*) описывает подмножество из 4-х стрингов {001100, 101100, 001101, 101101}. Очевидно, схема с m неопределенными позициями "\*" представляет 2^m стрингов. Для стрингов длины n, существует 3^n схем (возможны 3 символа {0, 1, \*} в каждой из n позиций).

В простом ГА основная идея заключается в объединении хромосом со значениями целевой функции (ЦФ) выше среднего. Например, пусть 1 в хромосоме соответствует наличию признака, способствующего выживанию (значение целевой функции больше среднего). Допустим, что имеются подстринги вида 11\*\*\* и \*\*111. Тогда, применяя к ним ОК можно получить хромосому 11111 с признаками, способствующими наилучшим значениям фитнесс-функции. Рассмотрим также полезное понятие строительного блока. Например, в шаблоне \*\*\*\*1 строительным блоком является элемент 1. В шаблоне 10\*\*\* строительным блоком будет составной элемент 10. Очевидно, вид строительных блоков должен выбираться из знаний о решаемой задаче и отражать полезные свойства, чтобы далее из строительных блоков как из "кирпичиков" собрать "здание", т.е. решение с лучшим значением ЦФ. Желательно, чтобы в ГА выражались условия, которые разрешают разрыв строительных блоков только в крайних случаях, указанных пользователем

Не все схемы являются одинаковыми. Некоторые более специфичны, чем другие. Для количественной оценки вводятся две характеристики:

1. порядок схемы O(H), который определяется числом фиксированных позиций (не равных \*). Например, для H_1= 0\*1\*\*\* ее порядок O(H_1)= 2.
2. определенная длина L(H)– это расстояние между первой и последней определенной (не равной \*) позицией. Например для шаблона H_2= 0111\*1\*\* L(H_2)= 6 – 1 = 5. А для H_3= 0\*\*\*\* L(H_3)= 1 – 1 = 0.

Обозначим через m(H,t)– число стрингов, содержащихся в популяции A(t)(t – шаг итерации или время), которые отображаются (покрываются) схемой H.

Пусть f(H) означает среднее значение фитнесс-функции для хромосом, покрываемых данной схемой H, а \overline {f(x)}=\frac{\sum_{j=1}^N f(x_j)}{N}\\ – среднее значение фитнесс-функции для всей популяции (N – размер популяции). Очевидно, что для схемы, которая представляет хорошее решение, было бы желательным, чтобы количество хромосом, соответствующих этой схеме, возрастало в процессе эволюции с ростом номера поколения. Далее мы рассмотрим, как ведет себя m(H,t) при выполнении основных операторов ГА.

Фундаментальная теорема ГА

Влияние репродукции

Напомним, что в процессе репродукции хромосомы копируются в промежуточную популяцию согласно их значениям фитнесс-функции – хромосома x_i со значением f(x_i) выбирается с вероятностью P(x_i)=\frac{f(x_i)}{\sum f(x_j)}.

После репродукции мы ожидаем на следующем шаге получить m(H,t+1) двоичных стрингов, отображаемых схемой H. Известно, что

|  |  |
| --- | --- |
| m(H,t+1)=\frac{m(H,t)*N*f(H)}{\sum f(x_j)} | ( 1.1) |

Это обусловлено тем, что:

1. в среднем вероятность выбора стрингов, покрываемых схемой H, определяется величиной \frac{f(H)}{\sum f(x_j)},
2. число стрингов, представляемых H, равно m(H,t),
3. число стрингов в популяции равно N.

Мы можем переписать эту формулу с учетом обозначения

\overline{f(x)}=\frac{\sum_{j=1}^N f(x_j)}{N}

и получим следующее выражение для числа особей, покрываемых схемой в промежуточной популяции :

|  |  |
| --- | --- |
| m(H,t+1)=\frac{m(H,t)*f(H)}{\overline{f(x)} | ( 1.2) |

Другими словами, схемы "растут" как отношение среднего значения фитнесс-функции схемы к среднему значению фитнесс-функции популяции. Схема со значением фитнесс-функции выше средней в популяции имеет больше возможностей для копирования и наоборот. Правило репродукции Холланда гласит: "схема со значением фитнесс-функции выше среднего живёт и размножается, а со значением фитнесс-функции ниже среднего умирает".

Предположим, что схема H имеет значение выше среднего фитнесс-функции на величину c*f, где c– константа. Тогда последнее выражение (1.2) можно модифицировать:

m(H,t+1)=\frac{m(H,t)*(\bar f+c*\bar f)}{\bar f}=(1+c)m(H,t)

Начиная с t=0 и предполагая, что c– величина постоянная, получаем следующее выражение числа особей промежуточной популяции, покрываемых схемой,

|  |  |
| --- | --- |
| m(H,t)=m(H,0)*(1+c)^t | ( 1.3) |

Это равенство описывает геометрическую прогрессию. Очевидно, что при c>0 схема "размножается" а при c<0 схема умирает.

Далее рассмотрим влияние оператора кроссинговера на число особей в популяции, покрываемых схемой.

Влияние кроссинговера

Рассмотрим конкретный стринг А=011|1000 длины n=7 и две схемы, представляющие этот стринг: H_1 = \*1\*| \*\*\*0, H_2= \*\*\*| 10\*\*

Здесь символ "|" , как обычно, обозначает точку кроссинговера k=3.

Очевидно, что схема H_1 после кроссинговера с точкой k=3, скорее всего, будет уничтожена потому, что "1" в позиции 2 и "0" в позиции 7 попадут в разные новые стринги после кроссинговера. С другой стороны, ясно, что схема H_2 будет выживать, так как "10" в позициях 4,5 будут содержаться вместе в одном новом стринге. Хотя мы взяли точку скрещивания ОК случайно, ясно, что схема H_1 менее приспособлена к выживанию, чем схема H_2. Если точка скрещивания ОК выбирается случайным образом среди n-1=7-1=6 возможных позиций, то ясно, что схема H_1 разрушается с вероятностью

P(d)=\frac{L(H_1)}{(n-1)}=\frac{5}{6}

Очевидно, что эта же схема выживает с вероятностью

P(S)=1-P(d)=\frac{1}{6}

Аналогично, схема H_2 имеет длину L(H_2) = 1 и вероятность её уничтожения P(d) = 1/6, а вероятность выживания схемы после применения ОК P(S) = 5/6. Очевидно, что нижняя граница вероятности выживания схемы после применения ОК может быть вычислена для любой схемы. Так как схема выживает, когда точка ОК попадает вне "определенной длины" , вероятность выживания для простого ОК определяется по формуле P_s(OK) = 1 – L(H)/(n-1).

Если ОК выполняется посредством случайного выбора, например, с вероятностью P_c, то вероятность выживания схемы определяется так: P(S) \ge 1 – P_c*L(H)/(n-1).

Очевидно, что это выражение уменьшается при P_c\to 1. Теперь мы можем асимптотически оценить совместный эффект операторов репродукции и кроссинговера. При независимости выполнения OP и OK можно получить следующее выражение:

|  |  |
| --- | --- |
| m(H,t+1)\ge m(H,t)\frac{f(H)}{\bar f}[1-P_c\frac{L(H)}{n-1}] | ( 1.4) |

Таким образом, число схем H в новой популяции зависит от двух факторов:

1. значение фитнесс-функции схемы выше или ниже значения ЦФ популяции;
2. схема имеет "длинную" или "короткую" L(H) (определенную длину).

Видно, что схемы со значением ЦФ выше средней и с короткой длиной L(H) имеют возможность экспоненциального роста в новой популяции.

Далее рассмотрим влияние оператора мутации на число особей в популяции, покрываемых схемой.

Влияние мутации

Напомним, что оператор мутации (ОМ) есть случайное изменение элемента в стринге с вероятностью P_m. Очевидно, что для того чтобы схема H выжила, все определенные позиции должны выжить. Поскольку один ген выживает с вероятностью (1-P_m), то данная схема выживает, когда каждая из O(H) фиксированных позиций схемы выживает. Умножая вероятность выживания (1-P_m) O(H) раз, получим, что вероятность выживания при ОМ равна (1-p_m)^{O(H)}. Для малых величин P_m\ll 1 это выражение может быть аппроксимировано как P_S(OM) = 1 – О(H)\cdot P_m.

Из этого следует, что схема H дает ожидаемое число особей в следующей популяции после выполнения всех генетических операторов ОР, ОК и ОМ согласно следующей формуле:

|  |  |
| --- | --- |
| m(H,t+1)>m(H,t)\cdot \frac{f(H)}{\bar f}[1-P_m\frac{L(H)}{1-n}-O(H)\cdot P_m] | ( 1.5) |

Этот важный результат известен какфундаментальная теорема ГА. Учитывая (1.3) ее можно также сформулировать следующим образом.

Теорема 1.1. Схемы малого порядка, малой определенной длины и со значением фитнесс-функции выше средней формируют растущее по показательному закону число своих представителей в последующих поколениях генетического алгоритма.

На основании приведенных результатов была выдвинута гипотеза о строительных блоках.

Гипотеза 1.1. Генетический алгоритм стремится достичь близкого к оптимальному результата за счет комбинирования хороших схем (со значением фитнесс-функции выше средней, малого порядка и определенной длины).

Такие схемы называются строительными блоками. В соответствии с этим оптимальное решение строится путем объединения наилучших из полученных на текущий момент частичных решений. Оператор скрещивания на двоичных стрингах не слишком часто уничтожает схемы малой определенной длины, однако ликвидирует схемы большой длины. Однако, не смотря на губительность операторов скрещивания и мутации для схем большого порядка и определенной длины, число обрабатываемых схем настолько велико, что даже в относительно малом числе особей в популяции генетический алгоритм дает неплохие результаты.

Несмотря на то, что для доказательства этой гипотезы были предприняты значительные усилия, строгого доказательства получено не было, и в большинстве нетривиальных приложений опираются на эмпирические результаты.

Далее проиллюстрируем приведенные результаты. Вернемся к примеру Голдберга определения \mbox{max}\ f(x)=x^2. В дополнение к имеющимся таблицам [рис 1.3](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=3#image.1.3) пусть имеется 3 конкретные схемы H_1=1\*\*\*\*\*, H_2=\*10\*\* и H_3=1\*\*\*0, описанные в [табл.1.2](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=9#table.1.2).

Рассмотрим сначала схему H_1. При репродукции стринги копируются с вероятностью, определенной согласно величине их фитнесс-функции. Из [табл. 1.1](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=5#table.1.1) видно, что стринги 2,4 покрываются схемой H_1. После выполнения репродукции мы видим в [табл. 1.3](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=9#table.1.3) (строка H_1), что получены три копии и стринг 3 также вошел в популяцию.

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица 1.2.** | | | |
| **Схема** | **Перед репродукцией** | **Представители стрингов** | **Среднее значение ЦФ схемы f(H)** |
| H_1 | 1\*\*\*\* | 2,4 | 469 |
| H_2 | \*10\*\* | 2,3 | 320 |
| H_3 | 1\*\*\*0 | 2 | 576 |

Проверим, соответствует ли это число фундаментальной теореме схем. Согласно теории мы ожидаем m*f(H)/\bar f копий. Вычисляя среднее значение ЦФ f(H_1), получаем \bar f(H_1)=(576+361)/2=468,5. Разделив это число на среднее значение ЦФ популяции \bar f=293 и умножив его на число m(H_1,t)=2 стрингов, покрываемых H_1 на шаге t, получаем ожидаемое число стрингов, покрываемых H_1 на шаге t+1, т.е. m(H_1,t+1)=2\cdot 468,5 / 293 = 3,20.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Таблица 1.3.** | | | | | | |
| **Схема** | **После репродукции** | | | **После всех операторов** | | |
|  | **Ожидаемое число стрингов** | **Действительное число стрингов** | **Представители стрингов** | **Ожидаемое число стрингов** | **Действительное число стрингов** | **Представители стрингов** |
| H_1 | 3,20 | 3 | 2, 3, 4 | 3,20 | 3 | 2, 3, 4 |
| H_2 | 2,18 | 2 | 2, 3 | 1,64 | 2 | 2, 3 |
| H_3 | 1,97 | 2 | 2, 3 | 0,0 | 1 | 4 |

Сравнивая это число с реальным числом копий 3, видим, что округление рассчитанного значения копий 3,2 дает их реальное число 3. Дальнейший анализ показывает, что в данном случае ОК не оказывает влияние на число стрингов, покрываемых схемой, так как определенная длина L(H_1)=0 предотвращает разрыв схемы. Далее при мутации с вероятностью P_{OM}=0,001 мы должны ожидать уменьшение числа стрингов на m*О(H_1)*P(OM)=3*1*0,001=0,003, что практически не оказывает влияния на окончательный результат. В итоге для схемы H_1 получаем ожидаемое увеличение числа особей до 3, что соответствует формуле (1.2).

Рассмотрим теперь схемы H_2 и H_3 с двумя фиксированными позициями. Схема H_2 имеет также два представителя в начальной популяции (2,3) и две копии в следующей промежуточной популяции. Вычисляем m(H_2)=2\cdot 320/293=2,18. Здесь f(H_2)=320– среднее значение ЦФ схемы, а \bar f=293– среднее значение ЦФ популяции. Для H_3 получаем только одно стринговое представление и m(H_3)=1\cdot 576/293=1,97. Здесь 576 – среднее значение ЦФ схемы.

Заметим, что для конкретной схемы H_2 представительство двух копий в популяции является хорошим результатом и он может случиться только один раз из 4-х возможных случаев (n-1=5-1=4). Схема H_2=*|1|0|*|* имеет реальную возможность дать новую копию. В результате, схема H_2 выживает с высокой вероятностью. Действительно, число стрингов, покрываемых H_2, равно m(H_2,t+1)=2,18\cdot0,75=1,64\approx 2.

Для схемы H_3 при высокой определенной длине L(H_3)=4 оператор ОК обычно разрушает эту схему. Поэтому ожидаемое число m(H_3,t+1)=0 соответствует числу, получаемому по формуле (1.4).

В соответствие с полученными результатами видно, что важнейшим аспектом является кодирование особей, которое должно обеспечить построение схем малого порядка малой определенной длины и со значениями фитнесс-функции выше среднего

Следующий простой пример показывает важность построения генома с учетом теорем схем.

Рассмотрим поиск максимума (для простоты при целочисленных значениях x и y) функции f(x,y)=x^2-y+17. Можно показать, что максимум f=66 достигается при значениях x=(111)_2=(7)_{10} и y=(0000)_2=(0)_{10}. Пусть x=x_2x_1x_0 и y= y_3y_2y_1y_0, где x_i,y_j\in \{0,1\}. Рассмотрим различное строение геномов. В первом случае пусть генотип представляет y_3x_2y_2x_1y_1x_0y_0, во втором генотип определим как y_3y_2y_1y_0\ x_2x_1x_0. Отметим, что в первом случае старшие разряды x и y расположены в генотипе близко, а во втором варианте наоборот – достаточно далеко. Поскольку желательна короткая определенная длина, генетический алгоритм с геномом, в котором старшие разряды расположены близко, должен быть лучше по сравнению с геномом, где эти разряды стоят далеко друг от друга. Для первого случая \frac{L(H)}{1-n}=\frac{1}{6} для схемы H_1=01\*\*\*\*\*, где содержатся значения, наиболее важные для искомого решения. Для второго случая схема при тех же значениях имеет вид H_2=0\*\*\*\*\*1, для которой \frac{L(H)}{1-n}=1 и поэтому необходимые значения старших битов x и y, скорее всего, при выполнении оператора кроссинговера будут "разорваны". Поэтому первый вариант генома предпочтительней.

1.7. Параметры генетических алгоритмов

Эффективность ГА зависит от ряда параметров, к которым относятся: мощность популяции, структура представления решения, вид генетических операторов кроссинговера и мутации, вероятности кроссинговера и мутации P_c и P_m и т.п..

Мощность популяции N является важнейшим параметром ГА, который критичен во многих приложениях. Чем больше N, тем больше разнообразие потенциальных решений (при хорошейсхеме инициализации, обеспечивающей однородное распределение частиц). Большое число особей позволяет покрыть большую часть пространства поиска за одну итерацию. С другой стороны, большое число особей повышает вычислительную сложность итерации и при этом ГА может выродиться в случайный параллельный поиск. Если N мало, то ГА работает быстро, но при этом увеличивается опасность преждевременной сходимости к локальному экстремуму. Большая мощность популяции увеличивает генофонд, но процесс поиска замедляется. Обычно полагают N\in [30;200].

На разных этапах работы ГА оптимальное значение N может быть различным. На начальном этапе N должно быть большим, а на заключительном N можно уменьшить. Большую роль играют также вид генетических операторов, которые представлены в следующем разделе. Кроме этого, не менее важны значения вероятностей кроссинговера и мутации P_c и P_m.

Для оптимизации, особенно мультимодальных функций, наиболее существенными являются две характеристики ГА:

* способность сходиться к оптимуму (локальному или глобальному) после нахождения области, содержащей этот оптимум;
* способность находить новые области в пространстве решений в поисках глобального оптимума.

Баланс между этими характеристиками ГА в значительной степени определяется значениями вероятности P_c и P_m, типом используемых генетических операторов (прежде всего кроссинговера). Увеличение значений P_c и P_m ведет к расширению пространства поиска. Обычно используют следующие значения вероятностей: P_c\in [0,5;1] и P_m\in [0,001;0,05]. В адаптивных генетических алгоритмах изменяются, прежде всего, вероятности кроссинговера и мутации P_c и P_m.

1.8. Преимущества и недостатки генетических алгоритмов

1.8.1. Концептуальная простота

Основным преимуществом ГА является их концептуальная простота. Рассмотрим снова блок-схему ГА, представленную на [рис.1.1](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=2#image.1.1) Основными шагами алгоритма являются: инициализация, оценка качества решения с помощью фитнесс-функции, итеративное изменение популяции путем отбора особей и применения генетических операторов. Отметим, что здесь не важна высокая точность оценки качества потенциальных решений – необходимо знать прежде всего их ранг (номер позиции по качеству решения). Информация о значениях градиента целевой функции здесь не нужна. Через ряд итераций (поколений) популяция (множество потенциальных решений проблемы) может сойтись асимптотически к оптимальному решению. Эффективность ГА зависит от способа кодирования решения, используемых генетических операторов, включая отбор особей, и начальной инициализации популяции.

1.8.2. Широкая применимость

ГА могут быть использованы при решении любой проблемы, которая может быть сформулирована как задача оптимизации. Они требуют разработки (или выбора) структуры данных для представления потенциального решения, показателя качества для оценки потенциального решения и генетических операторов, порождающих новые решения из старых. Пространство возможных решений может быть разбито на различные области, которые могут включать недостижимые зоны и зоны возможных изменений. Способ представления решения обычно выбирается инженером-разработчиком на основании его интуиции и опыта. В этом смысле процедура выбора кодирования потенциального решения независима в отличие от обычных численных методов, где, как правило, допускаются только непрерывные значения из определенного диапазона. Представление решения должно позволять генетическим операторам при изменении решений сохранять поведенческую связь между родителями и потомками. Небольшие изменения в структуре данных родителя должны вести к небольшим изменениям свойств потомков и наоборот, большие изменения у родителей должны вызывать значительные изменения свойств потомков, что должно способствовать эффективному поиску решения в пространстве поиска. Множество возможных изменений может регулироваться с помощью эффективного размера шага изменения в пространстве поиска, который может регулироваться разработчиком (или даже адаптироваться автоматически). Такой гибкий подход позволяет использовать по сути одну и ту же процедуру при решении задач как численной, так и комбинаторной оптимизации (в частности, целочисленной оптимизации и т.п.).

1.8.3. Менее жесткие требования при решении реальных задач

Реальные задачи оптимизации часто:

1. накладывают нелинейные ограничения;
2. требуют платежной функции, не связанной с наименьшей квадратичной ошибкой;
3. включают наличие шумов или случайных выбросов, которые не позволяют применять классические методы оптимизации [[17](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.17)].

Целевые функции для реальных проблем часто мультимодальны и градиентные методы сходятся быстро к локальным экстремумам, которые могут давать неудовлетворительные решения. Для простых задач, где поверхность отклика, например, является строго выпуклой, генетические алгоритмы проигрывают классическим по эффективности. Эксперименты показали, что для мультимодальных функций ГА дают лучшие результаты. В случае нелинейных ограничений классические методы даже при выпуклой поверхности могут давать некорректные результаты. Напротив, эволюционные методы могут непосредственно учитывать произвольные линейные и нелинейные ограничения.

1.8.4. Потенциальное использование априорных знаний и гибридизация с другими методами

При решении конкретной проблемы всегда целесообразно учесть в алгоритме проблемно-ориентированные априорные знания [[17](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.17)]. Специализированные алгоритмы, учитывающие такую информацию (но имеющие ограниченную область применения), как правило, существенно превосходят по характеристикам неспециализированные методы. Эволюционные алгоритмы по своей структуре легче позволяют учитывать априорные знания. Это может быть выражено, например, в виде специальной структуры данных для представления решений или специальных проблемно-ориентированных генетических операторов. Часто такая информация включается в фитнесс-функцию (например, физические или химические свойства вещества), что позволяет сфокусировать поиск решения в пространстве поиска.

Генетические алгоритмы могут комбинироваться с другими более традиционными методами. Известны работы, где на первом этапе оптимизации используются генетические алгоритмы совместно с градиентным методом, который применяется на заключительном этапе, когда уже найдена "зона интереса". Эти алгоритмы могут применяться совместно и параллельно. Отметим, что начальная популяция потенциальных решений может быть получена путем применения, например, жадных алгоритмов, а не эволюционных методов. Генетические алгоритмы часто используются для оптимизации и обучения искусственных нейронных сетей или нечетких продукционных систем. В этом случае часто удается преодолеть ограничения, связанные с традиционными подходами.

1.8.5. Параллелизм

Эволюция является высоко параллельным процессом, поскольку популяция состоит из множества особей, которые развиваются параллельно. Это позволяет расширить возможности применения эволюционных вычислений для решения все более сложных задач. Отметим, что основные вычислительные ресурсы в генетических алгоритмах используются при оценке значений фитнесс-функций, которые для различных особей могут выполняться параллельно, а последовательной является только процедура отбора. Поэтому эволюционные методы естественно реализуются в многопроцессорных распределенных компьютерных системах, которые в настоящее время все шире применяются в различных областях науки и техники и даже на бытовом уровне в многоядерных процессорах. В действительности при решении некоторых задач на распределенных вычислительных системах время решения в идеале может быть обратно пропорционально числу используемых задач. Это создает благоприятные условия для решения сложных задач высокой размерности за разумное время с помощью генетических алгоритмов.

1.8.6. Устойчивость к динамическим изменениям

Традиционные методы оптимизации неустойчивы к динамическим изменениям окружающей среды и часто требуют полного рестарта при таких изменениях для получения адекватного решения. Напротив, эволюционные алгоритмы могут быть использованы для адаптации потенциальных решений к изменившимся условиям. Полученная на момент изменения популяция дает базис для дальнейшего улучшения решений и в большинстве случаев нет необходимости проводить случайную реинициализацию.

1.8.7. Способность к самоорганизации

Большинство классических методов требуют начальной установки соответствующих параметров алгоритмов. Это также относится и к генетическим алгоритмам, которые зависят от множества параметров, таких как мощность популяции, вероятности кроссинговера и мутации, шаг мутации и т.п. Однако в эволюционных алгоритмах легче ввести самоадаптацию, когда в процессе поиска решения указанные параметры оптимизируются.

1.8.8. Решение проблем, для которых отсутствует опыт решений

Возможно, самым большим преимуществом генетических алгоритмов является их способность исследовать проблемы, для которых нет экспертов и соответствующего опыта решений. Следует отметить, что экспертные оценки достаточно часто используются при решении трудно формализуемых задач, но они иногда дают менее адекватные решения, чем автоматизированные методы. Существуют определенные проблемы с получением знаний у экспертов: они могут не согласиться на это, могут быть неквалифицированными, могут быть несовместимыми и просто ошибаться.

Исследования по искусственному интеллекту в настоящее время дали ряд интересных результатов, каждый из которых позволяет эффективно решать свой класс задач (например, хорошо играть в шахматы, или распознавать изображения символов и т.п.). Но большинство этих узких приложений требуют участия человека. Эти методы могут эффективно решать некоторые сложные проблемы, требующие высокого быстродействия, но они не могут конкурировать с человеческим интеллектом - "Они решают проблемы, но они не решают проблему как решать проблемы". Напротив, эволюционные алгоритмы дают метод решения проблемы, как решать проблемы при отсутствии экспертов (человеческого опыта) [[17](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.17)].

1.8.9. Недостатки ГА

Естественно ГА не свободны от недостатков. К ним можно отнести прежде всего следующие. Конфигурация ГА для решения сложных реальных задач не очевидна. Для решения конкретной задачи необходимо выбрать или разработать представление (кодирование) потенциального решения. Существует также проблема определения фитнесс-функции. Есть проблема выбора параметров ГА, таких как мощность популяции, вероятности генетических операторов и т.д. Нет эффективных критериев окончания работа алгоритма. ГА не могут использовать информацию о градиентах, что уменьшает их эффективность для классических задач. ГА не эффективны для гладких унимодальных (с одним экстремумом) функций. ГА не эффективны при поиске локальных экстремумов. ГА требуют достаточно больших вычислительных ресурсов. При решении мультимодальных задач бывают случаи преждевременной сходимости к локальным экстремумам и поэтому в общем случае не гарантируют нахождение глобального экстремума.

1.9. No Free Lunch теорема

Возникает естественный вопрос – существует ли некоторый лучший эволюционный алгоритм, который дает всегда лучшие результаты при решении всевозможных проблем? Например, можно ли выбрать генетические операторы и их параметры так, чтобы алгоритм давал лучшие результаты независимо от решаемой проблемы? К сожалению, ответ отрицательный – такого лучшего эволюционного алгоритма не существует! Это следует из известной No Free Lunch (NFL) теоремы (бесплатных завтраков не бывает), которая доказана относительно недавно, в 1996г.[[18](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.18),[19](https://www.intuit.ru/studies/courses/14227/1284/literature#literature.1.19)] и вызвала оживленную дискуссию. Пусть P(d_m^y|f,m,a)- условная вероятность получения частного решения d_m после m итераций работы алгоритма a при целевой функции f. В этих терминах Вольперт и Макреди доказали так называемую No Free Lunch (NFL) теорему:

NFL теорема. Для любой пары алгоритмов a_1 и a_2 имеет место равенство

\sum_f P(d_m^y|f,m,a_1)=\sum_f P(d_m^y|f,m,a_2).

Таким образом, сумма условных вероятностей посещения в пространстве решений каждой точки d_m одинакова для множества всевозможных целевых функций независимо от используемого алгоритма. Из этого результата непосредственно следует, что при любой мере \Phi(d_m^y) производительности (характеристик сложности) алгоритма в среднем для всевозможных целевых функций f вероятность P(\Phi(d_m^y)|f,m,a) не зависит от алгоритма a. Другими словами, не существует лучшего алгоритма (эволюционного или любого другого) для решения всех проблем. Если алгоритм выигрывает по своим характеристикам при решении некоторого класса задач, то это неминуемо компенсируется проигрышем (худшими характеристиками) для остальных задач.

Эта теорема вызвала оживленную дискуссию у специалистов по эволюционным вычислениям и некоторое неприятие. Дело в том, что в семидесятых годах были предприняты значительные усилия по поиску лучших значений параметров и генетических операторов ГА. Исследовались различные генетические операторы, значения вероятностей выполнения кроссинговера и мутации, мощности популяции и т.д. Большинство этих исследований апробировалось на сложившихся в каждой проблемной области тестовых задачах. Но из NFL-теоремы следует, что полученные результаты справедливы только на использованных тестовых задачах, а не для произвольных задач. Все усилия найти лучший оператор кроссинговера или мутации, оптимальные значения их параметров при отсутствии ограничений для исследуемого класса задач не имеют смысла!

Каждому эволюционному алгоритму присуще некоторое представление, которое позволяет манипулировать с потенциальными решениями. NFL теорема утверждает, что не существует лучшего эволюционного алгоритма для решения всех проблем.

Для того чтобы разрабатываемый алгоритм решал поставленную задачу лучше, чем случайный поиск (который с точки зрения NFL теоремы является просто другим алгоритмом) необходимо в нем использовать (отразить) структуру (априорные знания) этой проблемы. Из этого следует, что такой алгоритм может не соответствовать структуре другой проблемы (и покажет для нее плохие результаты). Следует отметить, что недостаточно просто указать, что проблема имеет некоторую структуру - такая структура должна соответствовать разрабатываемому алгоритму. Более того, структура должна быть определена. Недостаточно, как это иногда бывает, сказать "Мы имеем дело с реальными проблемами, а не с всевозможными, поэтому NFL теорема не применима". Что значит структура реальной проблемы? Очевидно, что формальное описание такой структуры проблематично. Например, реальные проблемы нашего времени и столетней давности могут сильно отличаться. Следует отметить, что простое сужение области возможных проблем без идентификации соответствия между рассматриваемым множеством проблем и алгоритмом недостаточно для получения преимущества данного метода решения этих проблем по сравнению с другими.

NFL-теорема подтверждает, что разные алгоритмы имеют различную эффективность при решении разных задач. Например, классические методы оптимизации, как правило, более эффективны при решении линейных, квадратичных, строго выпуклых, унимодальных, разделяемых и других специальных классов проблем. С другой стороны, генетические алгоритмы часто успешно решают задачи там, где классические методы не работают – там, где целевые функции терпят разрывы, не дифференцируемы, мультимодальны (имеют много экстремумов), зашумлены и т.п. Обычно их эффективность и устойчивость выше там, где целевые функции имеют сложный (не стандартный) вид, что более характерно для решения реальных практических задач. Конечно, лучшим способом подтверждения эффективности алгоритма является доказательство его сходимости и оценки вычислительной сложности. Но, как правило, это возможно только в случае упрощенной постановки задачи. Другой альтернативой является проверка алгоритмов на тестовых задачах (benchmarks) данной проблемной области. К сожалению, в настоящее время не существует согласованного каталога таких задач для оценки старых или новых алгоритмов решения, хотя для многих типовых задач они уже сложились и широко используются.

Контрольные вопросы

1. Каковы "источники" ГА?
2. Какие генетические операторы используются в ГА?
3. Какую роль в ГА играет оператор репродукции (ОР)?
4. Опишите реализацию ОР в виде колеса рулетки и приведите пример его работы.
5. Придумайте другую реализацию ОР.
6. Опишите одноточечный оператор кроссинговера (ОК) и приведите пример его работы.
7. Предложите другую реализацию ОК.
8. Какую роль играет оператор мутации (ОМ)?
9. Опишите ОМ и приведите пример его работы.
10. Предложите другую реализацию ОМ.
11. Каковы основные параметры ГА?

Упражнения

1. Выполните программную реализацию простого ГА на одном из языков программирования для поиска экстремума заданной по варианту функции одной переменной ([табл. 1.5](https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=14#table.1.5)).

Вид экстремума:

|  |  |
| --- | --- |
| **Таблица 1.4.** | |
| **Вариант** | **Вид экстремума** |
| \le 15 | Максимум |
| > 15 | Минимум |

1. Исследовать зависимость времени поиска, числа поколений (генераций), точности нахождения решения от основных параметров генетического алгоритма:
   * число особей в популяции
   * вероятность кроссинговера, мутации.
2. Вывести на экран график данной функции с указанием найденного экстремума для каждого поколения
3. Сравнить найденное решение с действительным.

|  |  |  |
| --- | --- | --- |
| **Таблица 1.5.** | | |
| **Вариант** | **Вид функции** | **Промежуток поиска решения** |
| 1 | (1,85-x)*\cos(3,5x-0,5) | x\in [-10,10] |
| 2 | \cos(\exp(x))/\sin(\ln(x)) | x\in [2,4] |
| 3 | \sin(x)/x^2 | x\in [3.1,20] |
| 4 | \sin(2x)/x^2 | x\in [-20,-3.1] |
| 5 | \cos(2x)/x^2 | x\in [-20,-2.3] |
| 6 | (x-1)\cos(3x-15) | x\in [-10,10] |
| 7 | \ln(x)\cos(3x-15) | x\in [1,10] |
| 8 | \cos(3x-15)/|x|=0 | x\in [-10,-0.3),(0.3,10]\\x\in[-0.3,0.3] |
| 9 | \cos(3x-15)*x | x\in [-9.6,9.1] |
| 10 | \sin(x)/(1+\exp(-x)) | x\in [0.5,10] |
| 11 | \cos(x)/ (1+\exp(-x) | x\in [0.5,10] |
| 12 | (\exp(x)-\exp(-x))\cos(x)/(\exp(x)+\exp(-x)) | x\in [-5,5] |
| 13 | (\exp(-x)-\exp(x))\cos(x)/(\exp(x)+\exp(-x)) | x\in [-5,5] |
| 14 | \cos(x-0,5)/|x| | x\in [-10,0),(0,10],\min |
| 15 | \cos(2x)/|x-2| | x\in [-10,2),(2,10],\max |

Краткие итоги:

* представлено описание простого ГА;
* введены основные генетические операторы – репродукции, кроссинговера и мутации;
* описан концептуальный смысл фитнесс-функции и обсуждено ее отличие от целевой функции;
* представлены теоретические основы ГА (теория схем, фундаментальная теорема ГА);
* обсуждены преимущества и недостатки ГА.