Генетические алгоритмы

Генетический алгоритм — это алгоритм, который позволяет найти удовлетворительное решение к аналитически неразрешимым или сложнорешаемым проблемам через последовательный подбор и комбинирование искомых параметров с использованием механизмов, напоминающих биологическую эволюцию.

Генетические алгоритмы являются частью более общей группы методов, называемой эволюционными вычислениями, которые объединяют различные варианты использования эволюционных принципов для достижения поставленной цели.

Также в ней выделяют следующие направления:

•Эволюционные стратегии

Метод оптимизации, основанный на идеях адаптации и эволюции. Степень мутации в данном случае меняется со временем – это приводит к, так называемой, самоадаптации.

•Генетическое программирование

Применение эволюционного подхода к популяции программ.

•Эволюционное программирование

Было впервые предложено Л.Дж. Фогелем в 1960 году для моделирования эволюции как процесса обучения с целью создания искусственного интеллекта. Он использовал конечные автоматы, предсказывающие символы в цифровых последовательностях, которые, эволюционируя, становились более приспособленными к решению поставленной задачи.

Пусть перед нами стоит задача оптимизации, например:

Задача наилучшего приближения

Если рассматривать систему n линейных уравнений с m неизвестными

$$Ax = b$$

в случае, когда она переопределена ($n > m$), то иногда оказывается естественной задача о нахождении вектора x, который "удовлетворяет этой системе наилучшим образом", т. е. из всех "не решений" является лучшим.

Задача о рационе.

Пусть имеется n различных пищевых продуктов, содержащих m различных питательных веществ. Обозначим через a_{ij} содержание (долю) j-го питательного вещества в i-ом продукте, через b_j — суточную потребность организма в j-ом питательном веществе, через c_i — стоимость единицы i-го продукта. Требуется составить суточный рацион питания минимальной стоимости, удовлетворяющий потребность во всех питательных веществах

Транспортная задача.

Эта задача — классическая задача линейного программирования. К ней сводятся многие оптимизационные задачи. Формулируется она так. На m складах находится груз, который нужно развезти n потребителям. Пусть a_i (i = 1, ..., n) — количество груза на i-ом складе, а b_j (j = 1, ..., m) — потребность в грузе j-го потребителя, c_{ij} — стоимость перевозки единицы груза с i-го склада j-му потребителю. Требуется минимизировать стоимость перевозок.

Задачи о распределении ресурсов.

Общий смысл таких задач — распределить ограниченный ресурс между потребителями оптимальным образом. Рассмотрим простейший пример — *задачу о режиме работы энергосистемы.* Пусть *m* электростанций питают одну нагрузку мощности *p*. Обозначим через *xj* активную мощность, генерируемую *j*-ой электростанцией. Техническими условиями определяются возможный минимум m*j* и максимум *Mj* вырабатываемой *j*-ой электростанцией мощности. Допустим затраты на генерацию мощности *x* на *j*-ой электростанции равны *ej*(*x*). Требуется сгенерировать требуемую мощность *p* при минимальных затратах.

Переформулируем задачу оптимизации как задачу нахождения максимума некоторой функции $f(x1, x2, \ldots, xn)$, называемой *функцией приспособленности* (fitness function).

Она должна принимать неотрицательные значения на ограниченной области определения (для того, чтобы мы могли для каждой особи считать её приспособленность, которая не может быть отрицательной), при этом совершенно не требуются непрерывность и дифференцируемость.

Каждый параметр функции приспособленности кодируется строкой битов.

Особью будет называться строка, являющаяся конкатенацией строк упорядоченного набора параметров:

1010 10110 101 … 10101

| x1 | x2 | x3 | … | xn |

Универсальность ГА заключается в том, что от конкретной задачи зависят только такие параметры, как функция приспособленности и кодирование решений. Остальные шаги для всех задач производятся одинаково.

Генетические алгоритмы оперируют совокупностью особей (популяцией), которые представляют собой строки, кодирующие одно из решений задачи. Этим ГА отличается от большинства других алгоритмов оптимизации, которые оперируют лишь с одним решением, улучшая его.

С помощью функции приспособленности среди всех особей популяции выделяют:

- наиболее приспособленные (более подходящие решения), которые получают возможность скрещиваться и давать потомство
- наихудшие (плохие решения), которые удаляются из популяции и не дают потомства

Таким образом, приспособленность нового поколения в среднем выше предыдущего.

Алгоритм делится на три этапа:

Скрещивание
Селекция (отбор)
Формирования нового поколения

Если результат нас не устраивает, эти шаги повторяются до тех пор, пока результат нас не начнет удовлетворять или произойдет одно из ниже перечисленных условий:

Количество поколений (циклов) достигнет заранее выбранного максимума
Исчерпано время на мутацию

Более подробно о шагах

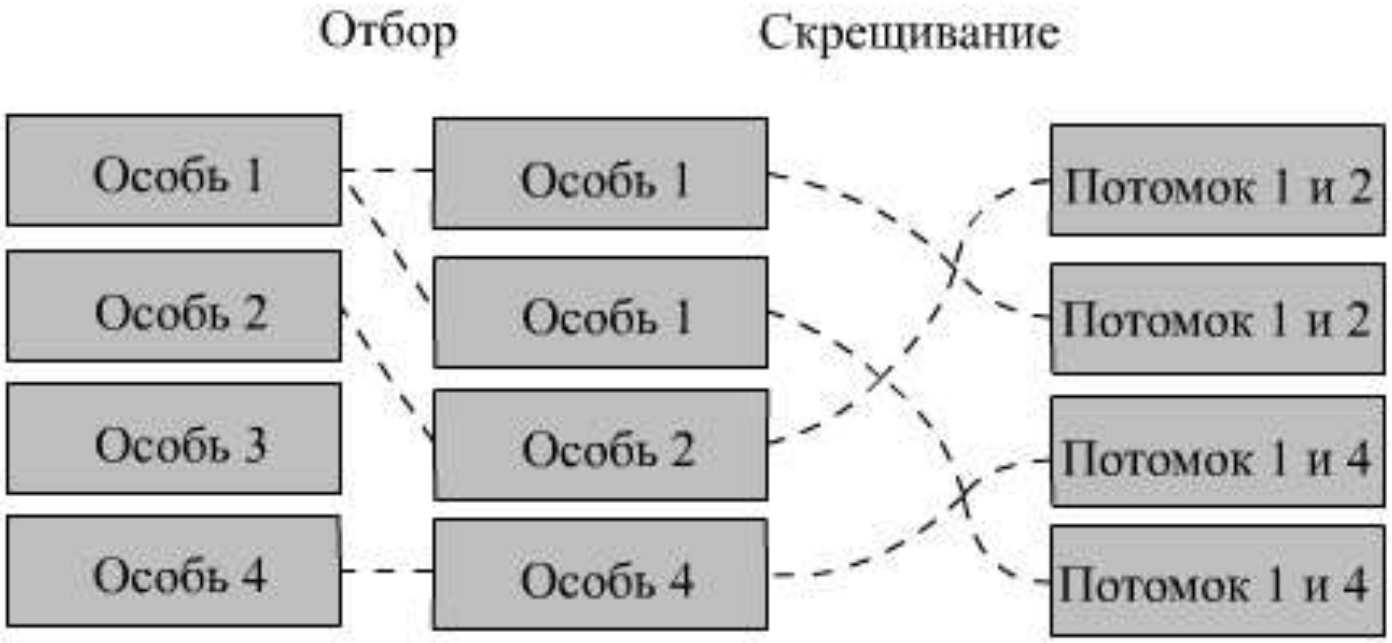
Создание новой популяции. На этом шаге создается начальная популяция, которая, вполне возможно, окажется не кошерной, однако велика вероятность, что алгоритм эту проблему исправит. Главное, чтобы они соответствовали «формату» и были «приспособлены к размножению».
Размножение. Ну тут все как у людей, для получения потомка требуется два родителя. Главное, чтобы потомок (ребенок) мог унаследовать у родителей их черты. При это размножаются все, а не только выжившие (эта фраза особенно абсурдна, но так как у нас все в сферическом вакууме, то можно все), в противном случае выделится один альфа самец, гены которого перекроют всех остальных, а нам это принципиально не приемлемо.
Мутации. Мутации схожи с размножением, из мутантов выбирают некое количество особей и изменяют их в соответствии с заранее определенными операциями.
Отбор. Тут начинается самое сладкое, мы начинаем выбирать из популяции долю тех, кто «пойдет дальше». При этом долю «выживших» после нашего отбора мы определяем заранее руками, указывая в виде параметра. Как ни печально, остальные особи должны погибнуть.

В классическом ГА:
начальная популяция формируется случайным образом
размер популяции (количество особей N) фиксируется и не изменяется в течение работы всего алгоритма
каждая особь генерируется как случайная L-битная строка, где L — длина кодировки особи
длина кодировки для всех особей одинакова


Создание начальной популяции
Отбор
Скрещивание
Мутация
Результат
Переход к новому поколению

Шаг алгоритма состоит из трех стадий:
генерация промежуточной популяции (*intermediate generation*) путем отбора (*selection*) текущего поколения
скрещивание (*recombination*) особей промежуточной популяции путем *кроссовера* (*crossover*), что приводит к формированию нового поколения
мутация нового поколения

Первые две стадии (отбор и скрещивание):



Отбор

Промежуточная популяция — это набор особей, получивших право размножаться. Наиболее приспособленные особи могут быть записаны туда несколько раз, наименее приспособленные с большой вероятностью туда вообще не попадут.

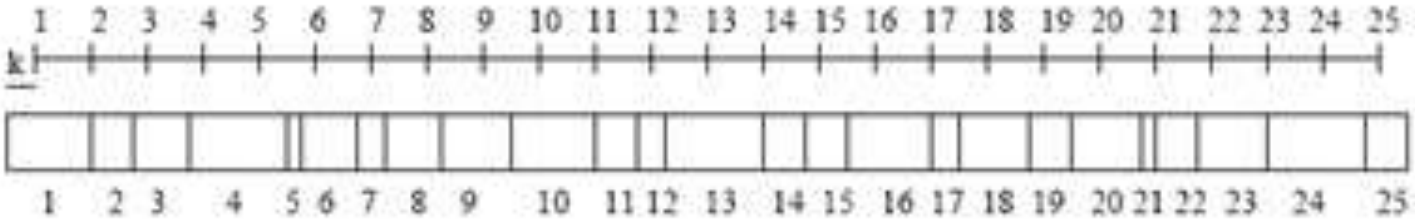
В классическом ГА вероятность каждой особи попасть в промежуточную популяцию пропорциональна ее приспособленности, т.е. работает *пропорциональный отбор* (*proportional selection*).

Существует несколько способов реализации данного отбора:

- *stochastic sampling*.

Пусть особи располагаются на колесе рулетки так, что размер сектора каждой особи пропорционален ее приспособленности. N раз запуская рулетку, выбираем требуемое количество особей для записи в промежуточную популяцию.

▪*remainder stochastic sampling*. Для каждой особи вычисляется отношение ее приспособленности к средней приспособленности популяции. Целая часть этого отношения указывает, сколько раз нужно записать особь в промежуточную популяцию, а дробная показывает её вероятность попасть туда ещё раз. Реализовать такой способ отбора удобно следующим образом: расположим особи на рулетке так же, как было описано. Теперь пусть у рулетки не одна стрелка, а *N*, причем они отсекают одинаковые сектора. Тогда один запуск рулетки выберет сразу все *N* особей, которые нужно записать в промежуточную популяцию. Такой способ иллюстрируется следующим рисунком:



Скрещивание

Особи промежуточной популяции случайным образом разбиваются на пары, потом с некоторой вероятностью скрещиваются, в результате чего получаются два потомка, которые записываются в новое поколение, или не скрещиваются, тогда в новое поколение записывается сама пара.

В классическом ГА применяется одноточечный оператор кроссовера (*1-point crossover*): для родительских строк случайным образом выбирается точка раздела, потомки получаются путём обмена отсечёнными частями.

011010.01010001101 -> 111100.**01010001101**

111100.10011101001 0**11010**.10011101001

Двухточечный кроссовер: выбираются 2 точки раздела, и родители обмениваются промежутками между ними:

010.1001.1011 -> **010.**1011.**1011**

110.1011.0100 110.**1001**.0100

Мутация

К полученному в результате отбора и скрещивания новому поколению применяется оператор мутации, необходимый для "выбивания" популяции из локального экстремума и способствующий защите от преждевременной сходимости. Каждый бит каждой особи популяции с некоторой вероятностью инвертируется. Эта вероятность обычно очень мала, менее 1%.

10110011**0**101101 -> 10110011**1**101101

Можно выбирать некоторое количество точек в хромосоме для инверсии, причем их число также может быть случайным. Также можно инвертировать сразу некоторую группу подряд идущих точек. Среди рекомендаций по выбору вероятности мутации нередко можно встретить варианты 1/L или 1/N.

Критерии останова

Такой процесс эволюции, вообще говоря, может продолжаться до бесконечности. Критерием останова может служить заданное количество поколений или *схождение* (*convergence*) популяции. Схождением называется состояние популяции, когда все строки популяции находятся в области некоторого экстремума и почти одинаковы. То есть кроссовер практически никак не изменяет популяции, а мутирующие особи склонны вымирать, так как менее приспособлены. Таким образом, схождение популяции означает, что достигнуто решение близкое к оптимальному.

В данной модели используется специфичная стратегия отбора. На каждом шаге только *одна* пара случайных родителей создает только *одного* ребенка. Этот ребенок заменяет не родителя, а одну из худших особей популяции.

Таким образом, на каждом шаге в популяции обновляется только одна особь.

Исследования показали, что поиск гиперплоскостей происходит лучше, а сходимость быстрее, чем у классического ГА.

Для нового поколения выбираются *N* лучших *различных* особей среди родителей и детей. Дублирование строк не допускается.

Для скрещивания все особи разбиваются на пары, но скрещиваются только те пары, между которыми расстояние Хэмминга больше некоторого порогового (также возможны ограничения на минимальное расстояние между крайними различающимися битами).

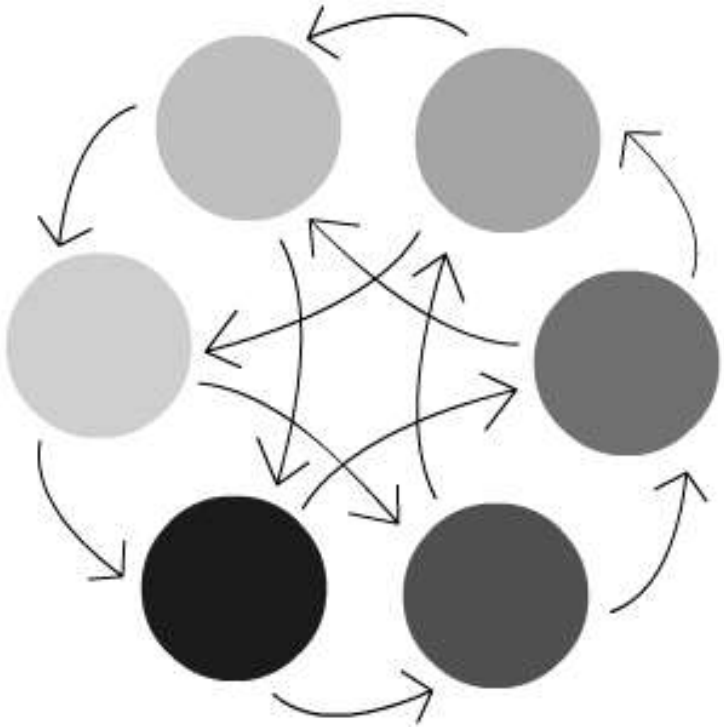
При скрещивании используется так называемый HUX-оператор (Half Uniform Crossover), разновидность однородного кроссовера – каждому потомку переходит ровно половина битов каждого родителя.

Размер популяции небольшой. Этим оправдано использование однородного кроссовера.

Данный алгоритм довольно быстро сходится из-за того, что в нем нет мутаций.

Островная модель (*island model*) — модель параллельного генетического алгоритма. Разобьем популяцию на несколько подпопуляций. Каждая их них будет развиваться отдельно с помощью некого генетического алгоритма. Таким образом, можно сказать, что мы расселили особи по нескольким изолированным островам.

Изредка (например, каждые 5 поколений) происходит миграция – острова обмениваются несколькими хорошими особями.



Так как населённость островов невелика, то подпопуляции будут склонны к преждевременной сходимости. Поэтому важно правильно установить частоту миграции:
чересчур частая миграция (или миграция слишком большого числа особей) приведет к смешению всех подпопуляций, и тогда островная модель будет несильно отличаться от обычного ГА
если миграция будет слишком редкой, то она не сможет предотвратить преждевременного схождения подпопуляций

Диофантовые уравнения (Уравнения с целочисленными корнями).

a+2b+3c+4d=30

корни данного уравнения лежат на отрезке [1;30], поэтому мы берем 5
случайных значений a,b,c,d. И так, у нас есть первое поколение:

(1,28,15,3)
(14,9,2,4)
(13,5,7,3)
(23,8,16,19)
(9,13,5,2)

Для того чтобы вычислить коэффициенты выживаемости, подставим каждое решение в выражение. Расстояние от полученного значения до 30 и будет нужным значением.

|114-30|=84

|54-30|=24

|56-30|=26

|163-30|=133

|58-30|=28

Среднее 59

Меньшие значения ближе к 30, соответственно они более желанны. Получается, что большие значения будут иметь меньший коэффициент выживаемости. Для создания системы вычислим вероятность выбора каждой (хромосомы). Но решение заключается в том, чтобы взять сумму обратных значений коэффициентов, и исходя из этого вычислять проценты.

0.135266 — сумма обратных коэффициентов

(1/84)/0.135266 = 8.80%
(1/24)/0.135266 = 30.8%
(1/26)/0.135266 = 28.4%
(1/133)/0.135266 = 5.56%
(1/28)/0.135266 = 26.4%

Далее будем выбирать пять пар родителей, у которых будет ровно по одному ребенку. Давать волю случаю мы будем давать ровно пять раз, каждый раз шанс стать родителем будет одинаковым и будет равен шансу на выживание.

3-1, 5-2, 3-5, 2-5, 5-3

Как было сказано ранее, потомок содержит информацию о генах отца и матери. Это можно обеспечить различными способами, но в данном случае будет использоваться «кроссовер». (| = разделительная линия)

Х.-отец: a1 | b1,c1,d1 **Х.-мать:** a2 | b2,c2,d2 **Х.-потомок:** a1,b2,c2,d2 or a2,b1,c1,d1

Х.-отец: a1,b1 | c1,d1 **Х.-мать:** a2,b2 | c2,d2 **Х.-потомок:** a1,b1,c2,d2 or a2,b2,c1,d1

Х.-отец: a1,b1,c1 | d1 **Х.-мать:** a2,b2,c2 | d2 **Х.-потомок:** a1,b1,c1,d2 or a2,b2,c2,d1

А теперь сделаем тоже самое с потомками:

Х.-отец: (13 | 5,7,3) **Х.-мать:** (1 | 28,15,3) **Х.-потомок:** (13,28,15,3)
Х.-отец: (9,13 | 5,2) **Х.-мать:** (14,9 | 2,4) **Х.-потомок:** (9,13,2,4)
Х.-отец: (13,5,7 | 3) **Х.-мать:** (9,13,5 | 2) **Х.-потомок:** (13,5,7,2)
Х.-отец: (14 | 9,2,4) **Х.-мать:** (9 | 13,5,2) **Х.-потомок:** (14,13,5,2)
Х.-отец: (13,5 | 7, 3) **Х.-мать:** (9,13 | 5, 2) **Х.-потомок:** (13,5,5,2)

Теперь вычислим коэффициенты выживаемости потомков.

(13,28,15,3) — |126-30|=96(9,13,2,4) — |57-30|=27

(13,5,7,2) — |57-30|=22

(14,13,5,2) — |63-30|=33

(13,5,5,2) — |46-30|=16

Теперь вычислим коэффициенты выживаемости потомков.

(13,28,15,3) — |126-30|=96
(9,13,2,4) — |57-30|=27
(13,5,7,2) — |57-30|=22
(14,13,5,2) — |63-30|=33
(13,5,5,2) — |46-30|=16

Печально так как средняя приспособленность (fitness) потомков оказалась 38.8, а у родителей этот коэффициент равнялся 59.
Именно в этот момент целесообразнее использовать мутацию, для этого заменим один или более значений на случайное число от 1 до 30.
Алгоритм будет работать до тех, пор, пока коэффициент выживаемости не будет равен нулю. Т.е. будет решением уравнения.

Задача коммивояжера является классической оптимизационной задачей.

Суть ее заключается в следующем. Дано множество из n городов и матрица расстояний между ними или стоимостей переезда (в зависимости от интерпретации).
Цель коммивояжера – объехать все эти города по кратчайшему пути или с наименьшими затратами на поездку. Причем в каждом городе он должен побывать один раз и свой путь закончить в том же городе, откуда начал.
Для решения предлагается следующая задача: имеется пять городов, стоимость переезда между которыми представлена следующей матрицей:

	Г1	Г2	Г3	Г4	Г5
Город 1	0	4	6	2	9
Город 2	4	0	3	2	9
Город 3	6	3	0	5	9
Город 4	2	2	5	0	8
Город 5	9	9	9	8	0

Для решения задачи применим следующий генетический алгоритм. Решение представим в виде перестановки чисел от 1 до 5, отображающей последовательность посещения городов. Значение целевой функции будет равно стоимости всей поездки, вычислений в соответствии с вышеприведенной матрицей. Одним из решений задачи является последовательность **12345** стоимостью 29. Из первого города во второй стоимость равна 4 из второго в третий – 3, из третьего в четвертый - 5, из четвертого в пятый – 8, из пятого в пятый - 9. Таким образом, 4+3+5+8+9=29.

34521 = 5+8+9+4+6=32

В качестве оператора скрещивания выберем двухточечный оператор скрещивания.
Пусть есть две родительские перестановки (1/234/5) и (3/452/1). Случайно и равновероятно выбираются две точки разрыва. Для примера возьмем ситуацию, когда первая точка разрыва находится между первым и вторым элементами перестановки, а вторая точка – между четвертым и пятым: (1/234/5), (3/452/1). На первом этапе перестановки обмениваются фрагментами, заключенными между точками разрыва: (*/452/*), (*/234/*). На втором этапе вместо звездочек вставляются соответствующие числа из исходной родительской перестановки, начиная со второго числа. В данном случае в первой перестановке (1/234/5) таким начальным числом является 3, а за ним идет 4, которое есть в новой перестановке, и мы его пропускаем, также пропускаем число 5, переходим на начало перестановки и выбираем число 1. В итоге вместо (*/452/*) получаем (34521), аналогично из (/3/452/1) и (*/234/*) получаем (52341).

34521 = 5+ 8+ 9+ 4+ 6 = 32

52341 = 9+ 3+ 5+ 2+ 9 = 28