Генетические алгоритмы

План доклада

1. Введение
2. История
3. Классический ГА
4. Теория
5. Настройка ГА
6. Различные модификации ГА
7. Некоторые модели ГА
8. Факторы, создающие сложность для ГА

Введение

Формальное определение

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

«Мягкие вычисления" (Soft computing techniques)

- Термин был введен Лофти Заде в 1994 году.
- Объединяет такие области как:
  - нечёткая логика
  - нейронные сети
  - вероятностные рассуждения
  - сети доверия
  - эволюционные вычисления
- Используются для создания гибридных интеллектуальных систем в различных комбинациях или самостоятельно.

Эволюционные вычисления

- Объединяют различные варианты использования эволюционных принципов для достижения поставленной цели
- Выделяют следующие направления:
  - Генетические алгоритмы
  - Эволюционные стратегии
  - Генетическое программирование
  - Эволюционное программирование

Зачем нужны ГА?

- Генетические алгоритмы применяются для решения следующих задач:
  - Оптимизация функций
  - Разнообразные задачи на графах (задача коммивояжера, раскраска, нахождение паросочетаний)
  - Настройка и обучение искусственной нейронной сети
  - Составление расписаний
  - Игровые стратегии
  - Аппроксимация функций
  - Искусственная жизнь
  - Биоинформатика

История

Несколько открытий в биологии

- В 1859 году Чарльз Дарвин опубликовал "Происхождение видов", где были провозглашены основные принципы эволюционной теории:
  - наследственность
  - изменчивость
  - естественный отбор
- Баричелли Н.А. - первые публикации, относящиеся к ГА:
  - "Symbiogenetic evolution processes realised by artificial methods" (1957)
  - "Numerical testing of evolution theories" (1962)

  Работы были направлены прежде всего на понимание природного феномена наследственности.

Ключевые работы

- Родителем современной теории генетических алгоритмов считается ***Д.Х. Холланд (J. Holland).*** Однако сначала его интересовала, прежде всего, способность природных систем к адаптации, а его мечтой было создание такой системы, которая могла бы приспосабливаться к любым условиям окружающей среды.
- В ***1975*** году Холланд публикует свою самую знаменитую работу ***«Adaptation in Natural and Artificial Systems».***В ней он впервые ввёл термин «генетический алгоритм» и предложил схему классического генетического алгоритма (*canonical GA*). В дальнейшем понятие «генетические алгоритмы» стало очень широким, и зачастую к ним относятся алгоритмы, сильно отличающиеся от классического ГА.
- Ученики Холланда - Кеннет Де Йонг (Kenneth De Jong) и Дэвид Голдберг (David E. Goldberg) - внесли огромный вклад в развитие ГА. Наиболее известная работа Голдберга - ***«Genetic algorithms in search optimization and machine learning» (1989).***

Классический ГА

Постановка задачи и функция приспособленности

- Пусть перед нами стоит задача оптимизации.
- Переформулируем её как задачу нахождения максимума некоторой функции $f(x1, x2, \ldots, xn)$, называемой ***функцией приспособленности*** (fitness function). Она должна:
  - быть определена на ограниченной области определения
  - принимать неотрицательные значения
  - 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

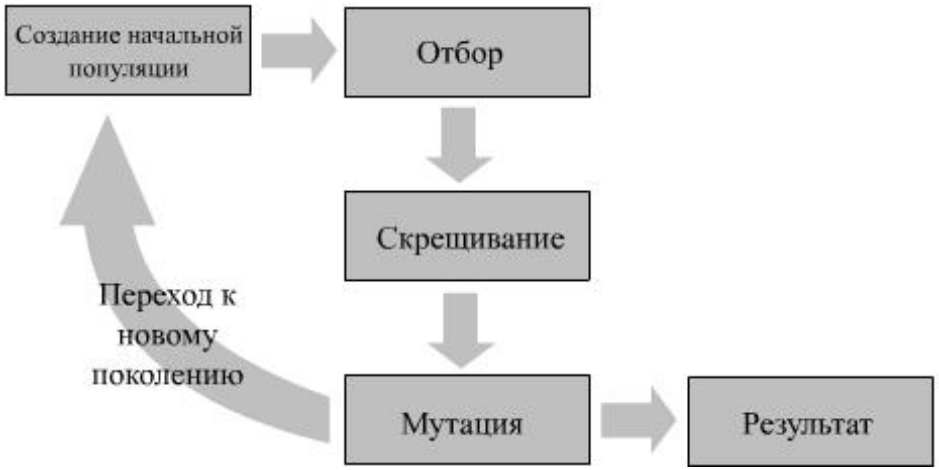
  | x1 | x2 | x3 | … | xn |

Принцип работы ГА

- Популяция – совокупностью всех «особей», представляющих собой строки, кодирующие одно из решений задачи.
- С помощью функции приспособленности:
  - наиболее приспособленные (более подходящие решения) получают возможность скрещиваться и давать потомство
  - наихудшие (плохие решения) удаляются из популяции и не дают потомства
- Таким образом, приспособленность нового поколения в среднем выше предыдущего.

- 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

Схема работы любого ГА

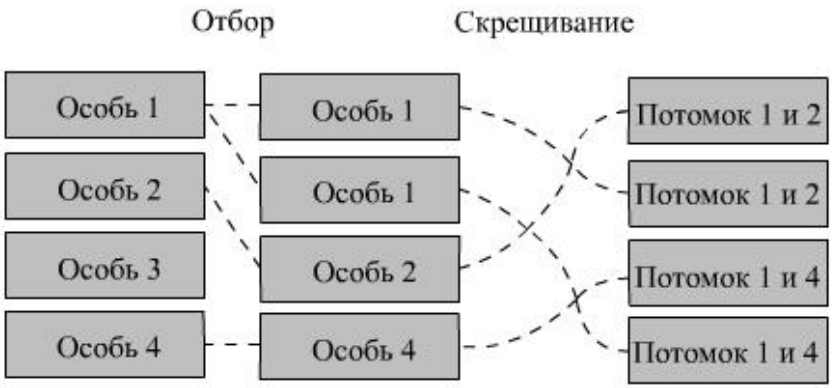


Шаг алгоритма состоит из трех стадий:

1. генерация промежуточной популяции (*intermediate generation*) путем отбора (*selection*) текущего поколения
2. скрещивание (*recombination*) особей промежуточной популяции путем *кроссовера* (*crossover*), что приводит к формированию нового поколения
3. мутация нового поколения

Отбор

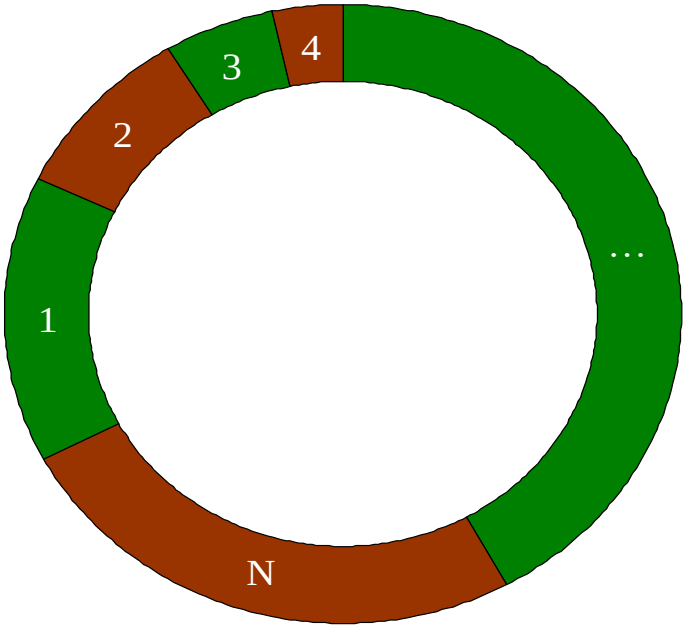
- На рисунке изображены отбор и скрещивание:

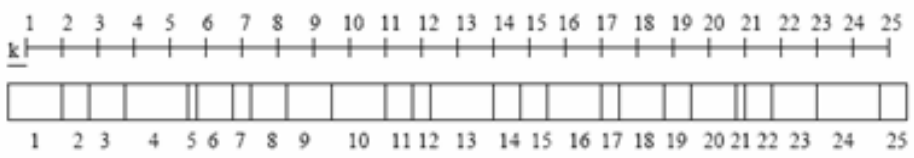


- Промежуточная популяция — набор особей, получивших право размножаться.
- В классическом ГА вероятность каждой особи попасть в промежуточную популяцию пропорциональна ее приспособленности, т.е. работает ***пропорциональный отбор*** (proportional selection).

Пропорциональный отбор

- Существует несколько способов реализации пропорционального отбора:
- ***stochastic sampling*.** 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*.** Особи располагаются на рулетке так же, как и раньше. Но теперь у рулетки не одна стрелка, а *N*, причем они отсекают одинаковые сектора. За один запуск рулетки выбираем сразу все *N* особей.





Скрещивание

- Особи промежуточной популяции случайным образом разбиваются на пары, которые с некоторой вероятностью
  - скрещиваются, в результате чего получаются два потомка, которые записываются в новое поколение
  - 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 => 011010.10011101001

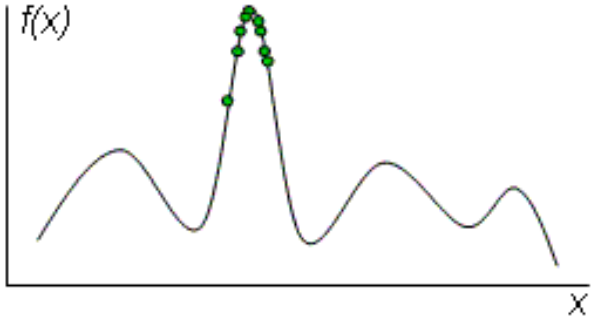
Мутация

- 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 малой вероятностью (обычно меньше 1%) инвертируется

1110001**0**10110 -> 1110001**1**10110

Критерии останова

- Критерием останова может служить заданное количество поколений или ***схождение*** (*convergence*) популяции.
- Схождением называется состояние популяции, когда все строки находятся в области некоторого экстремума и почти одинаковы.
- Таким образом, схождение популяции означает, что достигнуто решение близкое к оптимальному.
- Итоговым решением задачи может служить наиболее приспособленная особь последнего поколения.



Теория

Шаблоны

- ***Шаблоном*** (*schema*) называется строка длины *L* из символов 0, 1 и *(«don't care» символ). Строка является представителем данного шаблона, если все символы кроме * совпадают. Например, у шаблона 1*0*0 есть 4 представителя:
  - 1**0**0**0**
  - 1**0**0**1**0
  - 1**1**0**0**0
  - 1**1**0**1**0
- ***Порядком*** шаблона называется количество фиксированных в нём битов.
- ***Определяющей длиной*** шаблона называется расстояние между его крайними фиксированными битами.
- Например, для шаблона *H* = *0**10*:
  - порядок *o(H)* = 3
  - определяющая длина Δ*(H)* = 4
- Очевидно, что количество представителей шаблона *H* равно $2^{L\text{-}o(H)}$, а количество шаблонов равно 3^L.

Шаблоны

- Приспособленностью шаблона называется средняя приспособленность строк из популяции, являющихся его представителями.
  - Зависит от популяции, и поэтому меняется со временем.

- ***Неявный параллелизм*** (*implicit parallelism*)
  - Генетический алгоритм при отборе выбирает строку, но при этом неявным образом происходит и выборка шаблонов, представителем которых она является. То есть на каждом поколении количество представителей шаблона изменяется в соответствии с его текущей приспособленностью. Одна строка может являться представителем сразу многих шаблонов, поэтому при отборе одной строки отбирается сразу целое множество шаблонов.
- Сколько шаблонов могут иметь своим представителем данную строку?
- 2^L - на каждой позиции мы либо оставляем бит строки, либо заменяем его на *.

Теорема шаблонов

- ***Теорема шаблонов*** (*The Schema Theorem*) показывает, как изменяется доля представителей шаблона в популяции.
- Она верна только для классического ГА с пропорциональным отбором и одноточечным кроссовером.

- Пусть $M(H, t)$ — число представителей шаблона H в поколении t.
- Количество представителей шаблона H в промежуточном поколении:

$$M(H, t+ intermediate) = M(H, t)\frac{f(H, t)}{< f(t) >}$$

где $f(H, t)$ — приспособленность шаблона H в поколении t, а $<f(t)>$ — средняя приспособленность поколения t.

Теорема шаблонов

- Одноточечный кроссовер может разрушить шаблон из промежуточной популяции.
- Вероятность разрушения шаблона меньше, чем

$$\frac{\Delta(H)}{(L-1)}\left(1 - P(H, t)\frac{f(H, t)}{< f(t) >}\right)$$

- где *P(H, t)* — доля представителей шаблона *H* в поколении *t*. Первый множитель произведения равен вероятности попадания точки раздела между фиксированными битами шаблона, а второй — вероятности выбрать в пару представителя другого шаблона.
- Но даже в случае, когда второй родитель не является представителем данного шаблона, и точка раздела попадает между фиксированными битами, шаблон не обязательно разрушается.
- Например
  - рассматриваем шаблон 11***
  - точка раздела попадает между первыми двумя битами

    1.1011 => 1.**1011**

    1.0100 **1**.0100
- В этой ситуации шаблон не разрушается.

Теорема шаблонов

- Переходя от количества представителей к их доле, получаем следующее неравенство:

$$P(H, t+1) \geq P(H, t) \frac{f(H, t)}{< f(t) >} \left(1 - p_c \frac{\Delta(H)}{(L-1)} \left(1 - P(H, t) \frac{f(H, t)}{< f(t) >} \right) \right)$$

- Учтём влияние мутации. В шаблоне $o(H)$ фиксированных битов, и каждый не будет инвертирован с вероятностью $(1 - p_m)$.
- **Итоговая формула теоремы шаблонов:**

$$P(H, t+1) \geq P(H, t) \frac{f(H, t)}{< f(t) >} \left(1 - p_c \frac{\Delta(H)}{(L-1)} \left(1 - P(H, t) \frac{f(H, t)}{< f(t) >} \right) \right) (1 - p_m)^{o(H)}$$

  - Полученное выражение не слишком удачно для анализа работы генетического алгоритма, т.к. в нём присутствует знак неравенства.
  - Мы не учитывали случаи, когда рассматриваемый шаблон получается в результате кроссовера пары строк, не являющихся его представителями.
  - Приспособленность шаблона и средняя приспособленность популяции быстро изменяются от поколения к поколению, поэтому полученное неравенство хорошо описывает ситуацию только для следующего поколения.

- На данный момент существуют более точные версии этой теоремы и другие рассуждения, доказывающие целесообразность использования генетических алгоритмов.

Настройка ГА

Настройка ГА

- Генетический алгоритм производит поиск решений с помощью:
  - ***отбора гиперплоскостей*** (*hyperplane sampling*) путём кроссовера
  - метода *hill-climbing* путём мутации
- Исследования показали, что на простых задачах с малым размером популяции ГА с мутацией (и без кроссовера) находят решение быстрее, а на сложных многоэкстремальных функциях лучше использовать ГА с кроссовером, поскольку этот метод более надежен.
- С точки зрения теоремы шаблонов, мутация только вредит росту количества представителей хороших шаблонов, лишний раз разрушая их.
- Но мутация необходима для ГА с малым размером популяции, потому что для них свойственна ***преждевременная сходимость*** (*premature convergence*) – ситуация, когда в некоторых позициях все особи имеют один и тот же бит, не соответствующий глобальному экстремуму.

Настройка ГА

- ***Давление отбора*** (*selection pressure*) — мера того, насколько различаются шансы лучшей и худшей особей популяции попасть в промежуточную популяцию. Для пропорционального отбора эта величина с увеличением средней приспособленности популяции уменьшается, стремясь к 1.
- Для эффективной работы генетического алгоритма необходимо поддерживать тонкое равновесие между ***исследованием и использованием***:
  - При увеличении вероятностей скрещивания или мутации и уменьшении давления отбора (за счет использования других стратегий отбора) размножение представителей приспособленных шаблонов замедляется, но зато происходит интенсивный поиск других шаблонов.
  - Уменьшение вероятностей скрещивания или мутации и увеличение давления отбора ведет к интенсивному использованию найденных хороших шаблонов, но меньше внимания уделяется поиску новых.
- Необходимость сбалансированной сходимости ГА:
  - быстрая сходимость может привести к схождению к неоптимальному решению
  - медленная сходимость часто приводит к потере найденной наилучшей особи.
- Методология управления сходимостью классического ГА до сих пор не выработана.

Различные модификации ГА

Алфавит

- Аргументы в пользу кодирования бинарным алфавитом:
  - обеспечивает лучший поиск с помощью гиперплоскостей, т. к. предоставляет максимальное их количество.
    - Например, при кодировании 2^L значений для бинарного алфавита количество гиперплоскостей будет 3^L, а при использовании, четырехзначного алфавита – $5^{L/2}$.
  - для встречаемости каждого символа в каждой позиции требуется меньший размер популяции
    - Даже для двух строк, есть вероятность, что на каждой позиции в популяции есть и 0, и 1. Если же алфавит большей мощности, то до применения мутации большая часть пространства поиска будет недоступна с точки зрения кроссовера, после применения мутации станет недоступна другая часть.
- Однако небинарные алфавиты зачастую обеспечивают более наглядное представление решений задачи.

Кодирование параметров

- Для большинства функций ГА будут работать лучше при кодировании параметров кодом Грея, а не прямым бинарным кодом. Это связано с тем, что расстояние Хэмминга не всегда является критерием близости – например, числа 7 и 8 различаются на 4 бита. Бинарное кодирование добавляет дополнительные разрывы, что осложняет поиск.
  - Пример: пусть требуется минимизировать функцию $f(x) = x^2$
    - Если в начальной популяции преобладали хорошие отрицательные решения, то скорее всего мы придём к решению −1 = 11…1. Но достигнуть глобального минимума 00…0 будет практически невозможно, поскольку изменение любого бита будет приводить к ухудшению решения. При кодировании кодом Грея такой проблемы не возникает.
- Иногда применяется кодирование с плавающей точкой, которое тоже является более удачным, чем прямое бинарное.

Стратегии отбора

- ***Ранковый отбор*** (*rank selection*):для каждой особи ее вероятность попасть в промежуточную популяцию пропорциональна ее порядковому номеру в отсортированной по возрастанию приспособленности популяции. Такой вид отбора не зависит от средней приспособленности популяции.
- ***Турнирный отбор*** (*tournament selection*): из популяции случайным образом выбирается *t* особей, и лучшая из них помещается в промежуточную популяцию. Этот процесс повторяется *N* раз, пока промежуточная популяция не будет заполнена. Наиболее распространен вариант при *t* = 2. Турнирный отбор является более агрессивным, чем пропорциональный.
- ***Отбор усечением*** (*truncation selection*): популяция сортируется по приспособленности, затем берется заданная доля лучших, и из них случайным образом *N* раз выбирается особь для дальнейшего развития.

Кроссовер

- ***Двухточечный кроссовер**:* выбираются 2 точки раздела, и родители обмениваются промежутками между ними:
  - При этом определяющая длина измеряется в кольце – для шаблона 1*****1 при двухточечном кроссовере она будет равна 1, хотя при одноточечном была 6.
- ***Однородный кроссовер**:* один из детей наследует каждый бит с вероятностью p_0 у первого родителя и с $(1 - p_0)$ у второго, второй ребенок получает не унаследованные первым биты.
  Обычно $p_0 = 0.5$.
  - Однородный кроссовер в большинстве случаев разрушает шаблон, поэтому плохо предназначен для отбора гиперплоскостей, однако при малом размере популяции он препятствует преждевременному схождению.

Стратегии формирования нового поколения

- Два основных типа формирования нового поколения после кроссовера и мутации:
  - дети замещают родителей
  - новое поколение составляется из совокупности и детей, и их родителей
- Также применяется ***принцип элитизма***: в новое поколение включается заданное количество лучших особей предыдущего поколения (часто одна лучшая особь).
- Использование второй стратегии и элитизма не допускает потери лучших решений.
  - К примеру, если популяция сошлась в локальном максимуме, а мутация вывела одну из строк в область глобального, то при замещении родителей весьма вероятно, что эта особь в результате скрещивания будет потеряна, и решение задачи не будет получено. Если же используется элитизм, то полученное хорошее решение будет оставаться в популяции до тех пор, пока не будет найдено лучшее.

Некоторые модели ГА

Некоторые модели ГА

Genitor (Whitley)

- 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.

Некоторые модели ГА

CHC (Eshelman)

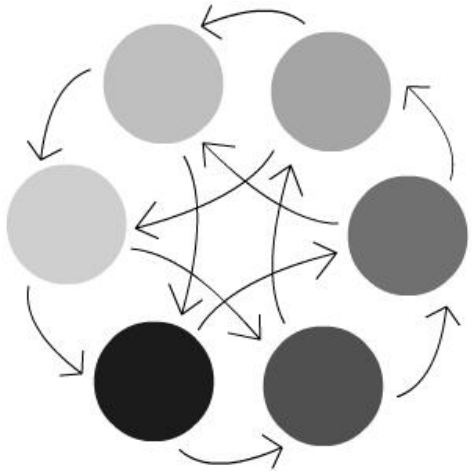
- *CHC* – это *Cross generational elitist selection, Heterogenous recombination, Cataclysmic mutation*.
- Для нового поколения выбираются *N* лучших *различных* особей среди родителей и детей. Дублирование строк не допускается.
- Для скрещивания все особи разбиваются на пары, но скрещиваются только те 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. В этом случае CHC применяет *cataclysmic mutation*: все строки, кроме самой приспособленной, подвергаются сильной мутации (изменяется около трети битов). Таким образом, алгоритм перезапускается и далее продолжает работу, применяя только кроссовер.

Некоторые модели ГА

Hybrid algorithm (Davis)

- Использование гибридного алгоритма позволяет объединить преимущества ГА с преимуществами классических методов.
- Дело в том, что ГА являются ***робастными*** алгоритмами, т.е. позволяют находить хорошее решение, но нахождение оптимального зачастую оказывается намного более трудной задачей в силу стохастичности принципов работы алгоритма. Поэтому возникла идея использовать ГА на начальном этапе для эффективного сужения пространства поиска вокруг глобального экстремума, а затем, взяв лучшую особь, применить один из "классических" методов оптимизации.
- Однако можно использовать "классические" методы (*hill-climbing,* например*)* и внутри самих ГА*.* На каждом поколении каждый полученный потомок оптимизируется этим методом, таким образом, каждая особь достигает локального максимума, вблизи которого она находится, после чего подвергается отбору, скрещиванию и мутации. Такой метод ухудшает способность алгоритма искать решение с помощью отбора гиперплоскостей, но зато возрастает вероятность того, что одна из особей попадет в область глобального максимума и после оптимизации окажется решением задачи.

Некоторые модели ГА



Island Models

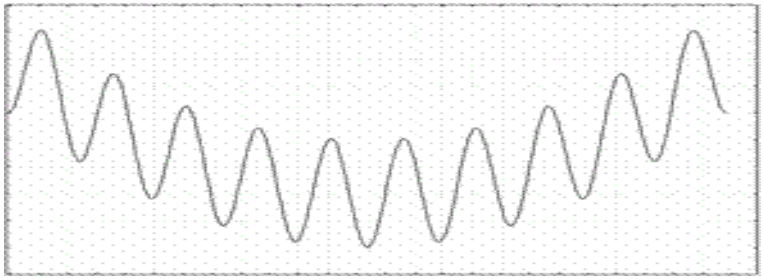
- ***Островная модель*** (*island model*) — модель параллельного генетического алгоритма. Разобьем популяцию на несколько подпопуляций. Каждая их них будет развиваться отдельно с помощью некого генетического алгоритма. Таким образом, можно сказать, что мы расселили особи по нескольким изолированным островам.
- 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
- Генетические алгоритмы стохастичны, поэтому при разных его запусках популяция может сходиться к разным хорошим решениям. Островная модель позволяет запустить алгоритм сразу несколько раз и совместить «достижения» разных островов для получения наилучшего решения.

Факторы, создающие сложность для ГА

Свойства функций приспособленности, создающие сложность для ГА

- ***Многоэкстремальность***: создается множество ложных аттракторов. Пример — функция Растригина:

$$f(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$$



- ***Обманчивость*** (*deception*): функция построена так, что шаблоны малого порядка уводят популяцию к локальному экстремуму.
  - Пример: пусть строка состоит из 10-ти четырехбитных подстрок. Пусть u_i равно количеству единиц в *i*-той подстроке. Зададим функцию *g*(*u*) следующей таблицей:

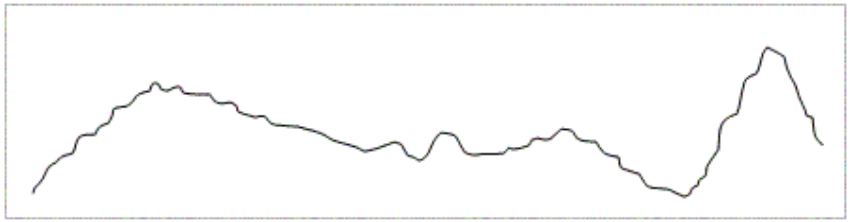
u	0 1 2 3 4
***g*(*u*)**	3 2 1 0 4

$$f = \sum_{i=1}^{10} g(u_i)$$

Свойства функций приспособленности, создающие сложность для ГА

- ***Изолированность*** («поиск иголки в стоге сена»): функция не предоставляет никакой информации, подсказывающей, в какой области искать максимум. Лишь случайное попадание особи в глобальный экстремум может решить задачу.

- ***Дополнительный шум*** (*noise*): значения приспособленности шаблонов сильно разбросаны, поэтому часто даже хорошие гиперплоскости малого порядка не проходят отбор, что замедляет поиск решения.



Заключение

Выводы

- Генетические алгоритмы являются универсальным методом оптимизации многопараметрических функций, что позволяет решать широкий спектр задач.
- Генетические алгоритмы имеют множество модификаций и сильно зависят от параметров. Зачастую небольшое изменение одного из них может привести к неожиданному улучшению результата.
- Следует помнить, что применение ГА полезно лишь в тех случаях, когда для данной задачи нет подходящего специального алгоритма решения.