Введение в эволюционные вычисления

Конспект лекций по современным интеллектуальным информационным технологиям за 2013 год

Акира Имада
Переводчик: Мокин Александр

Брестский государственный технический университет, Беларусь

(Дата последнего изменения)
Май 8, 2013

1 Что представляет собой Генетический Алгоритм?

Пусть имеется набор из n переменных. Задача опредиль оптимальный набор из n переменных. Для это был разработан генетичиский алгоритм.

1.1 Особь представляется в виде хромосомы

Совокупность последовательностей переменных x_i образуют популяцию. Каждая
такая последовательность называется хромосомой (особью).

x_1	x_2	x_3	x_4	x_5	- - - - - - - - -	x_n

Начинаем вычисления, предполагая, что результат будет улучшаться при переходе от предыдущей популяции к следующей.

1. **(Инициализация)** Создадим начальную популяцию из p случайно сгенерированных особей.
2. **(Оценка приспособленности)** Оценим приспособленность (fitness) каждой хромосомы и отсортируем хромосомы по убыванию их приспособленности.
3. **(Селекция)** Выбираются две хромосомы
   – В нашем случае выбираем две любые хромособы из лучшей половины популяции (Селекция отсечением - Truncate Selection)
4. **(Скрещивание)** Нового потомка создадим при помощи следущих двух операций
   – Равномерное скрещивание (Uniform Crossover)
   – Мутация
5. Создадим следующую популяцию повторяя шаг 3 и 4 n раз
6. Повторяем шаи 2 – 5 пока не будет полученино оптимальне или близкое к оптимальному занчение.

1.2 Как выбираются родительские особи?

Чем выше приспособленность особи, тем больше ероятность ее выбора.

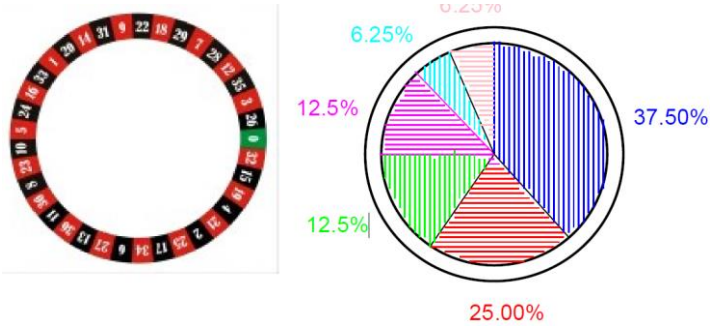
1.2.1 Три разных версии селекции

– *Селекция отсечением (Truncation Selection)* – самый простой из трех: выбираем родителей из лучшей части популяции, составляющей определеную долю популяции.

– *Селекция по принципу «Рулетки» (выбор пропорционально приспособленности)*: При селекции отсечением наименее приспосоленные особи не имеют шанса на скрещивание. В природе, однако,

Хорошие дети, иногда, получаются от плохих родителей. Тогда давайте дадим шанс плохим хромосомам, но с меньшей вероятностью выбора, пропорциональной их приспособленности. Такой метод назыается *выбор пропорционально приспособленности*, или *селекция по принципу «Рулетки».*

Выбираем хромосому таким образом, что вероятность выбора пропорциональна приспособленности.



Такую селекцию можно выполнить следующим образом: отсортировать имеющиеся
хромосомы в порядке убывания значения их оценки приспособленности и последовательно пересчитать для каждой хромосомы значение оценки накопительным
методом. Затем сгенерировать случайное число r от 0 до 1 и если r < 0.3750 тогда выбирем #1, иначе если r < 0.6250, тогда выберем #2, иначе если r < 0.7500, тогда выберем #3 и т.д.

Особи после сортировки	Оценка приспособленности	Оценка накопительным расчитанная методом
#1	0.3750	0.3750
#2	0.2500	0.6250
#3	0.1250	0.7500
#4	0.1250	0.8750
#5	0.0625	0.9375
#6	0.0625	1.0000

Таблица 1. Особи выбираются, используя случайные числа от 0 до 1, в соответствии с их значением приспособленности, расчитанной накопительным мтеодом.

– *Турнирная слекция (Tournament Selection)*: предположим, что имеется μ родительских особей, и μ потомков. Значение оценки приспособленности каждой из 2μ особей сравнивается с оценками приспособленности q особей, которые случайным образом выбираются из множества 2μ для каждой операции

сравнения. Затем 2μ особей упорядочиваются согласно количеству побед, и лучшие μ образцов выживают (q-турнирная селекция).

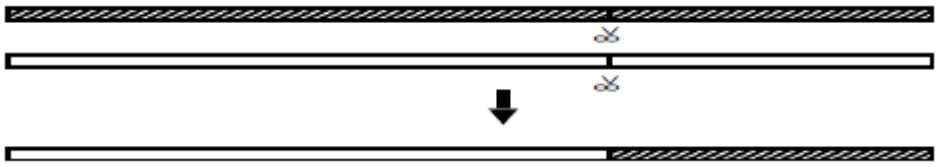
Обратите внимание, что при селекции по принципу рулетки, существует вероятность попадания наихудших представителей текущей популяции в следующую популяцию. А в случае селекции отсечением такое в принципе не возможно. При турнирной селекции так же существует вероятность выбора плохих особей, помимо самых худших q представителей популяции. Подбирая параметр q можно контролировать численность плохих особей в новой популяции.

1.3 Как родители создают потомство?

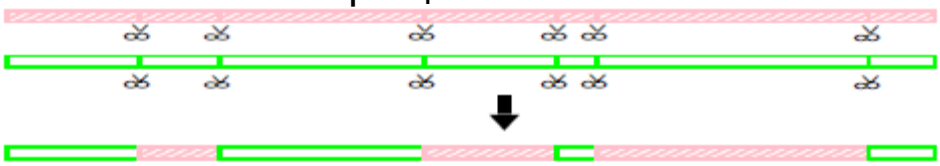
1.3.1 Кроссовер (Cross-over)

Так называемый кроссовер используется для производства потомков. Рассмотрим три разных версии срещивания.

Одноточечное скрещивание.



Многоточечное скрещивание



Равномерное срещивание

1.3.2 Мутация

Мутация используется для получения новых генов. Это необходимо, чтобы избежать
ситуации, когда особи популяции сходятся в локальном минимуме функции приспособленности. Вероятность мутации невелика – приблизительно равна

1/(количество_генов). Например, если если случайно сгенерированное значение от 0 до 1 меньше чем коэффициент мутации, тогда делаем мутацию иначе нет.

2 All One Problem

Чтобы рассмотреть как работает генетический алгоритм, напишем небольшую программу. Прдположим хромосома состоит из 20 генов. Каждый ген соответствует одной особенности, такой как голубые или карие глаза. Гены принимают значени 0 или
Также, предопложим, что 1 это хороший ген, а 0 плохой. Оценка приспособленности это число единиц в хромосоме.

Начнем с популяции из случайных хромосом, понаблюдаем за эволюцией. Пример хромосомы:

1 0 0 1 0 1 1 1 0 0 1 0 1 0 0 0 0 1 1 0.

Количество единиц это оценка приспособленности.
Если твая программа работает правильно, то мы получим лучшу хромосому

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

после нескольких операций.

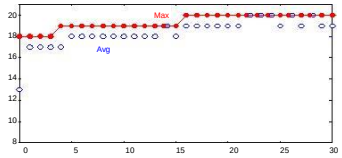


Рисунок 2: Зависимость приспособленности от поколения для случая с 20 генами.

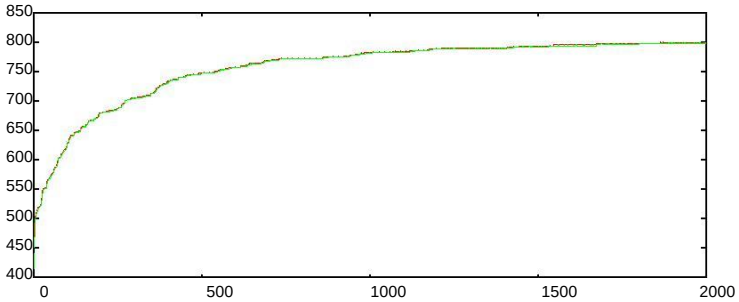


Рисунок 3: Зависимость приспособленности от поколения для случая с 800 генами.

3 Наиболее часто используемые функции приспособленности

Здесь описываются две многомерные функции, глобальные минимумы которых уже известны. Тем не менее, полезно изучить эти функции, для того чтобы понять каким образом эволюция работает внутри ПК.

3.1 Сферическая Модель (Sphere model)

3.1.1 Сферическая модель для 20 переменных

Одна из простейших функций

$$y = x_1^2 + x_2^2 + x_3^2 + \cdots + x_{20}^2.$$

или иначе:

$$y = \sum_{i=1}^{20} x_i^2$$

Которая определена, например, на интервале $x_i \in [-5.12, 5.12] \quad (i = 1, 2, \cdots, 20)$. Эта функция представление хорошо известной функции y = x^2 d 20-мерном пространстве. Единственный глобальный минимум расположен в начале координат, а локальные минимумы отсутствуют. Поэтому эта функция хорошо подходит для начала изучения генетического алгоритма.

Пример хромосомы:

3.2 4.4 -2.1 0.5 -3.8 -2.5 1.7 5.1 -0.3 -2.1 -3.8 5.0 0.4 4.2 -5.0 -1.3 3.3 4.0 -1.4 -3.9

Полагая, что i-й ген это i-я переменная x_i, тогда мы получим все значения $x_1, \cdots x_{20}$ и сможем и сможем вычислить значение y, которое является показателм приспособленности хромосомы.

3.1.2 Сферическая модель для одной переменной

Чтобы представить функцию графически разберем ее двумерную версию.

$$y = x^2$$

Тогда мы получим параболу (рис. 4).

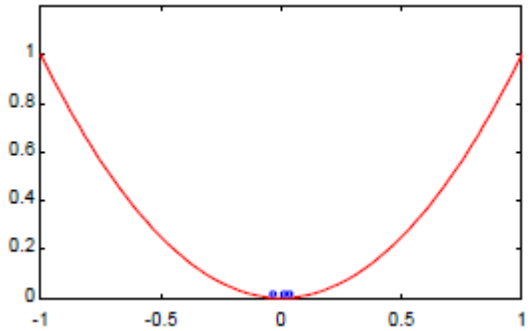


Рис. 4: Двумерная версия сферической модели

3.2 Функция Растригина

3.3 Функция Растригина для 20 переменных

Рассмотрим функцию Растригина. Она похожа на параболу, но имеет много

локальных минимумов:

$$y = \{x_i^2 - \cos(2\pi x_1)\} + \{x_2^2 - \cos(2\pi x_2)\} + \cdots + \{x_{20}^2 - \cos(2\pi x_{20}\}).$$

или иначе:

$$y = nA + \sum_{i=1}^{n} (x_i^2 - A\cos(2\pi x_i)), \qquad x_i \in [-5.12 - 5.12]$$

Волнистость может быть изменена значением A.

3.2.2 Функция Растригина для одной переменной.

Двумернуй пример (*n* = 1):

$$y = 3 + x^2 - 3\cos(2\pi x)$$

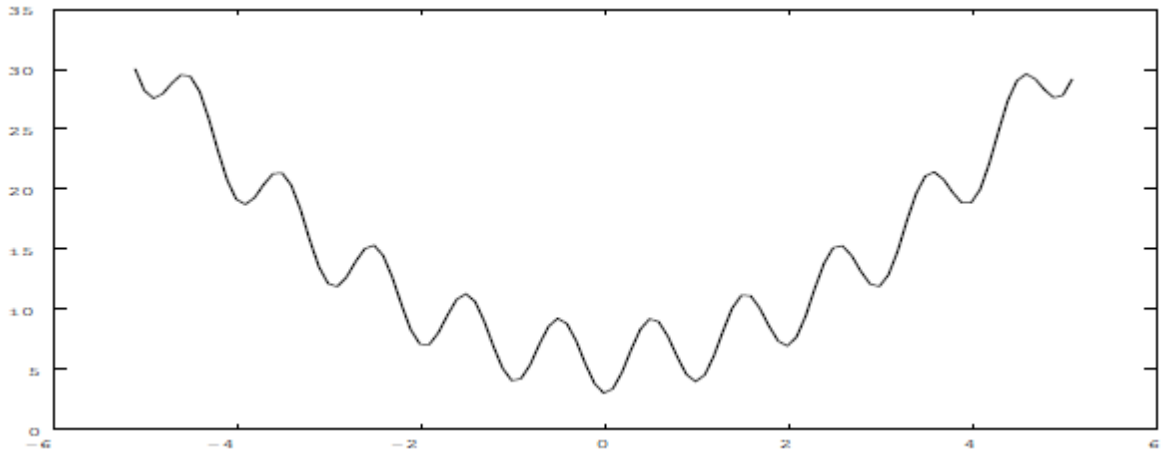


Рисунок 5. Двумерная версия функции Растригина.

4 Нейронная сеть для решения even-parity problem

Здесь мы определим кофигурацию весовых коэффициентов нейронной сети. Теперь подумайте о нейронной сети для решения логической функции even-N-parity. Если номер единицы четный, то результаты должен быть 1 иначе ноль. Возьмем нейронную сеть прямого распространения со структурой N-N-1, для неё мы должны определить $N^2 + N$ весовых коэффициентов так, чтобы любой двоичных вход преобразовывался в корректный результат.

Наша хромосома в данном случае имеет $N^2 + N$ генов. Для вычсления приспособленности (fitness) подаем на вход каждой сети (одна сеть соответствует одной хромосоме) все возможные 2^N входных, количество совпавших результатов будет оценкой приспособленности.

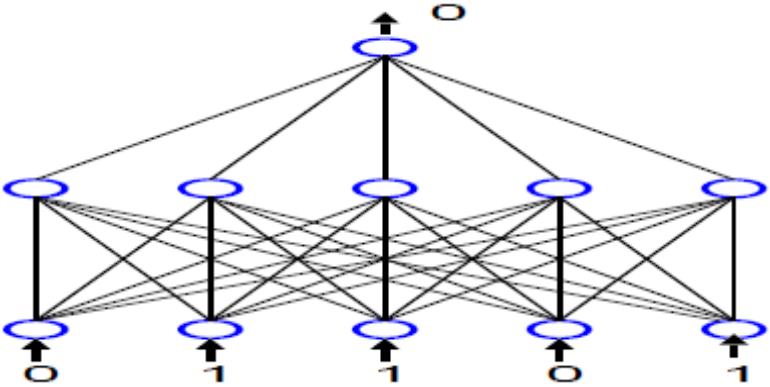


Рисунок 6. Структура сети прямого распространения (5-5-1)

Ниже представлен набор входов и выходов для even-5-parity Boolean function.

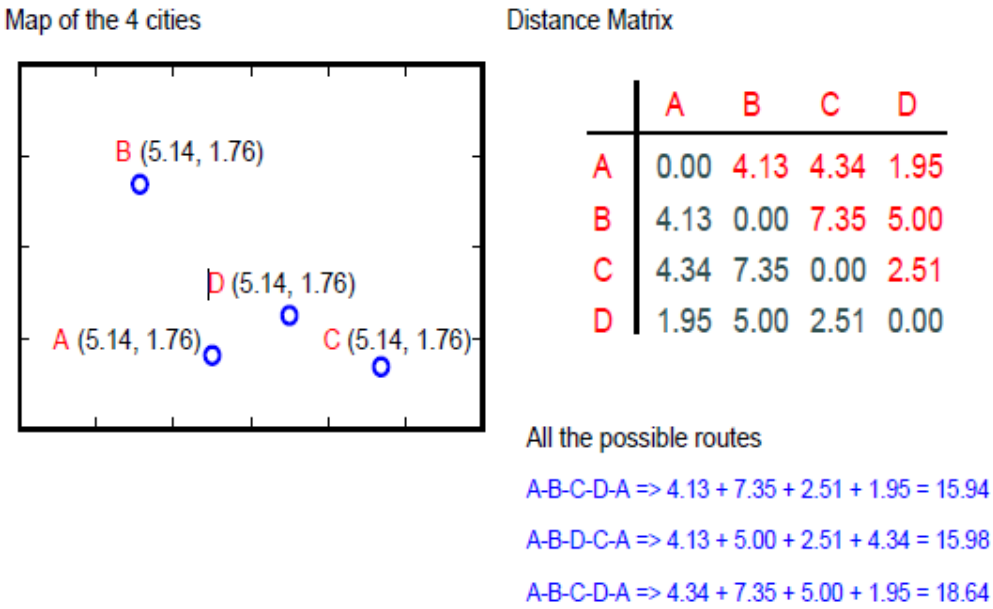
вход	выход
00000	1
00001	0
00010	0
00011	1
00100	0
00101	1
00110	1
00111	0
01000	0
...	...
11110	1
11111	0

Table 7: Входы и выходы для логической функции even-5-parity.

5 Задача коммивояжера (Traveling Sales-person Problem)

Допустим нам известны координаты N городов. Задача коммивояжера заключается в том, что торговец должен посетить каждый город только один раз, при этом его путь должен быть минимальным.
Торговец должен вернуться в начальный город после посещения всех городов, тогда если количество городов N, то количество всех возможных маршрутов (N - 1)! / 2, если мы не считаем маршруты с точно противоположным порядком городов уже посчитанного маршрута.

Давайте разберем задачу коммивояжера с 4 городами. Расположим города в случайно выбранных координатах x-y, где $0 < x, y < 10$. Затем рассчитаем расстояния между всеми возможными комбинациями из двух городов. Расстояния могут представлены в виде матрицы расстояний. Так как число маршрутов всего 3! / 2 = 3, то мы можем вычислить растояние для каждого маршрута. Пример изображен на рисунке 8.



	A	B	C	D
A	0.00	4.13	4.34	1.95
B	4.13	0.00	7.35	5.00
C	4.34	7.35	0.00	2.51
D	1.95	5.00	2.51	0.00

Рисунок 8 Пример для 4-х городов. Карта, матрица расстояний и все возможные маршруты с рассчитаным расстоянием.

Проблема в том, что число возможных маршрутов быдет очень быстро возрастать с увеличением числа городов N. Как много маршрутов по вашем может быть в случае когда N = 1000? Теперь мы применим генетический алгоритм для решения этой задачи[1].

[1]Мы рассмотрим решение задачи коммивояжера генетическим алгоритмом. Но существует казалось бы более простой метод вычисления, называемый муравьиный алгоритм (Ant Colony Optimization or ACO). ACO это техника оптимизации, которую мы заимстовали у умного муравьиного собщества, отражающая их коллективное поведение. Муравьи хорошо находят кратчайший путь между их гнездом и едой, когда их колега находит его, он сообщает его другим при помощи феромонов. Если у нас будет время мы изучим ACO позже.

5.1 Решение задачи коммивояжера при помощи генетического алгоритма

5.1.1 Как представить маршрут, чтобы скрещивание было корректным?

Как представить хромосому задающую путь? Представим предполагаемое решение как список городом, которые нужно посетить в определенном порядке. Тогда хромосома пути A-C-F-D-G-E-B имеет вид

(ACFDGEB)

Возможен ли корректный рельтат скрещивания и мутации? Ответ – нет! Например возможный ребенок двух родителей (ACFDGEB) и (AGBFCDE) после одноточечного скрещивания может быть (ACFFCDE) и это не корректный путь потому что C и F посещаются дважды, а не посещается вообще. Или если выполнить стандартную операцию мутации к (ACFDGEB), например, заменой 4-го гена на случайно выбранный город - (ACFAGEB), то такой резльтат не является корректным.

Тогда как правильно представить хромосому, чтобы результат скрещивания и мутации был корректным?

Один из вариантов:

Алгоритм 1 (Задача коммивояжера) Корректный путь может быть представлен в хромосоме, состоящей из любых целых чисел.

Шаг 1. Установим i = 1.

Шаг 2. Если i-й ген равен n, тогда n-й город в списке является городом, который уже посещен.

Шаг 3. Удаляем город из списка.

Шаг 4. Установим i = i + 1 и повторим шаги 2-4 пока i ≤ n.

Например, если список городов

{A, B, C, D, E, F, G, H, I}

хромосома: (112141311 задает путь:

A-B-D-C-H-E-I-F-G.

Попробуйте выполнить операцию одноточечного скрещивания на двух родителях (112141311) и (515553321).

5.1.2 Мутация

Следующее, как выполнить мутацию? Как насчет выбрать две случайные точки и высроить гены между ними в обратном порядке?

5.1.3 Когда остановить алгоритм?

И, наконец, когда нам остановить алгоритм? Ведь мы не знаем оптимальное

значение. Ответ - наблюдать за изменением оценки приспособленности. Мы можем предположить, что результат будет сходиться к оптимальному или близкому к оптимальному значению.

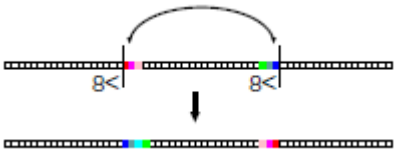


Рисунок 9. Возможная мутация в задаче коммивояжера.

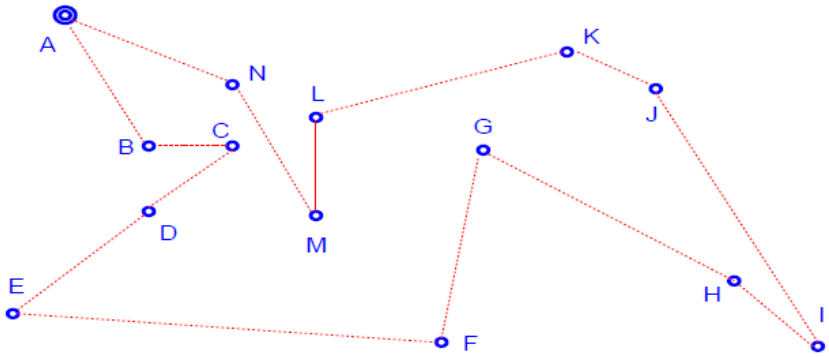


Рисунок 10. Пример с 14-ю городами и один возможный маршрут

5.1.4 TSP with a huge number of cities

Я нашел 13,509 настоящих городов в США с их координатами на сайте http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95. Почему мы не пытаемся решить эту сложную проблему сами. Графическое представление всех этих гордов предствалена на рисунке 11.

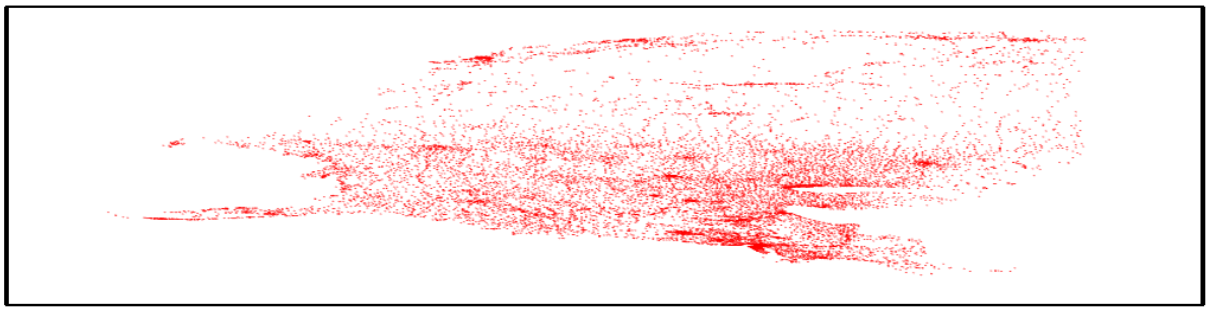


Рисунок 11. Пример 13509 настоящих городов расположенных в США. Нарисовано на основе данных взятых с сайта http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.

6 Навигация робота в сетке (Robot Navigation in GridWorld)

Предположим, что мы хотим запрограммировать агента или робота в сетке, тогда хромосома может быть составлена из генов, примающих целочисленные занчения от 1 до 4, где 1, 2, 3 и 4 оответствуют перемещению агента на одну позицию на север, юг, восток и запад. Взлянем на пример ниже.

Наша хромосома составлена из 4-х разных разных генов: движение вверх(i), вниз (ii), вправо (iii) и влево (iv). Взглянем на пример ниже.

(3111133233223333131442411141)

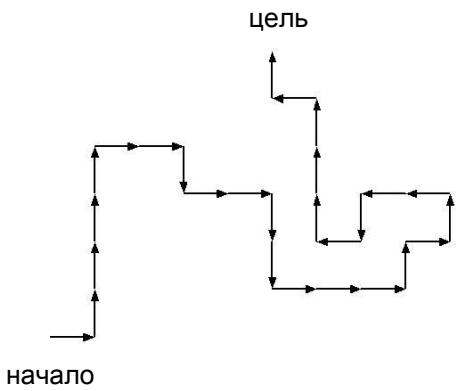


Рисунок 12. Пример хромосомы и предстваленного ею пути.

Рассмотрим две такие задачи более подробно в следующих подразделах.

6.1 Исследование сетки с ограниченной энергией (Exploration of a gridworld with a limited energy).

Поиск пути минимального манхэттенского расстояния

.

Поиск пути к начальной точке после максимального исследования.

Для человека не так сложно найти такой маршрут. Но как начет компьютера? Главный вопрос это, «Как мы представим фукцию приспособленности?»

Мы вернёмся к теме «нескольких функций приспособленности» позже.

6.2 Задача Джипа - От «Верблюда в пустыне» до «масохода»

Предположим на краю пустыни находится джип. бак джипа может быть наполнен только одной порцией топлива,

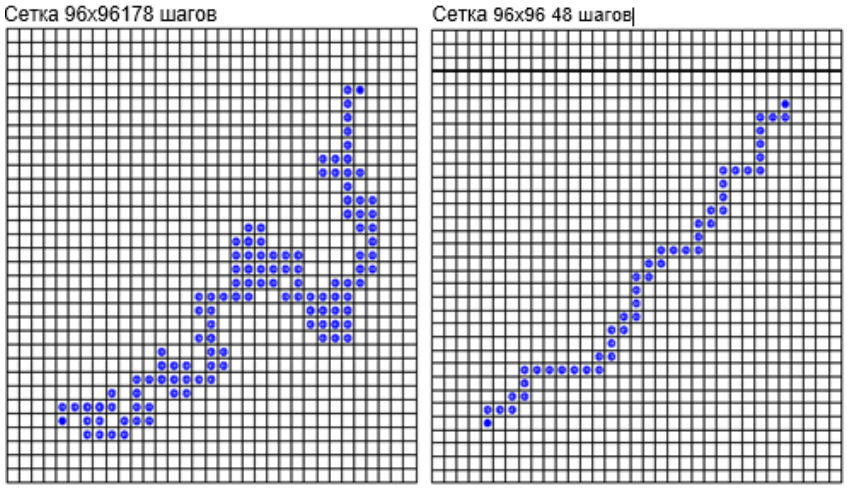


Рисунок 13. В пространстве 96x96, начиная с позиции (24,24) робот двигается к цели в точке (72,72), о которой роботу заранее ничего не известно. Левый рисунок: путь минимально длины спустя 100 случайных шагов. Справа: Минимаьлный путь, который робот нашел после эволюционного обучения, как показано на рисунке 3. (крайние области не оущены)

на которой он может проехать только одну единицу дистанции. Джип может заправляться только на базе. Джип может перевозить канистры с топливом для последующего использования. Допустим джип двигается по прямой.[2]

Вопрос: «Как далеко может в пустыне может заехать джип, двигаясь по пряой дроге, если в запасе у него n единиц топлива».

Допустим $n = 2$, лучшей стратегие будет выехать из базы с одной единицей топлива в баке и проядя 1/3 единицы расстояния (к этому моменту он потратит 1/3 единицы топлива в баке) оставить 1/3 топлива на точке и вернуться назад на базу. Конечно, когда джип вернётся на базу все топливо полученное на старте будет израсходовано. Затем джип заполнит вторую единицу топлива, пройдет 1/3 пути, зальет оставленное на точке топливо в бак. Теперь у него полный бак. Затем он поедет дальше пока все топливо не будет израсходовано. Таким образом максимальное расстояние, которое может пройти джип это 4/3 единицы расстояния

Мы знаем максимальное расстоение, если n=2. Тогда мы можем узнать максимальную дистанцию для n единиц топлива D_n, которая выражается через рекурсивное выражение $D_n=D_{n-1}+1/(2n-1)$.

Нам интересно могут ли эволюционные вычисления найти близкую к лучшей стратегию, скажем, если n=5, для которой максимальное расстояние 1323/945=1.4. (Если я все правильно посчитал, можете проверить).

---

[2]Первоначально задача появилась под названием “Верблюд, перевозящий зерно в пустыне». Это была 52 задача в сборнике “*Propositions ad acuendos inventes"* (латынь) автор Алкуин (Alcuin) из Йорка (около 732{804) до н.э.). *Затем это была задача джипа в пустыне, а потом задача марсохода.*

7 Эволюционнирующая стратегия – итеративная Дилемма Заключенного

Итак, Быть или не быть? – Вот в чем вопрос. Не только Шекспир, но и многие другие авторы посвящали свои рабты данной дилемме. Опера Пуччини (Puccini) «Тоска» (1899) является типичным примером[2].

.

7.1 Когда возникает дилемма?

Допустим, *n* человек являются участниками следующей игры. Каждый из них находится в изолированной кабине, откуда они не могут общаться и видеть друг друга. Каждая кабина оборудована кнопкой. Все участники находится в кабине в течении одной минуты. Если никто не нажмет кнопку, каждый получит по 100$. С другой стороны, если кто-нибудь нажмет кнопку или несколько человек нажмут кнопку, то первый кто нажмет кнопку получит 10$, а остальные не получат ни цента. Что вы будете делать, если вы будете одним из этих людей?

7.1.1 Условия наличия дилеммы

Что если деньги, которые вам дадут, если вы не нажмете кнопку это 10$, а тот кто нажмет кнопку первым получит 100$? В этом случае никакой дилеммы нет. Нажимайте кнопку незамедлительно без колебаний.

7.2 Дилемма заключенного

В теории игр существует задача называемая дилеммой заключенного[4], которая формулируется следующим образом:

Задача (Дилемма заключенного) Двум арестантам А и В предлагают сделку:

---

[3]Matt Ridley однажды написал книгу “Происхождение альтруизма и добродетели. От инстинктов к сотрудничеству”. В Penguin Books (1996) про эту книгу сказано следующее “Героиня оперы Пуччини “Тоска” столкнулась со сложнейшей дилеммой. Ее возлюбленный Каварадосси был приговорен к смертной казни шерифом по имени Скарпиа. Однако шериф предложил ей сделку. Если Тоска переспит с ним, то он сохранит жизнь возлюбленного, приказав использовать холостые патроны. Тоска решает обмануть Скарпиа. Она соглашается на его предложение, но надеется убить его, после того как будет дано распоряжение об использовании холостых патронов. Она выполняет свой план, но только после узнает, что Скарпиа тоже обманул ее. Каварадосси мертв, распоряжения об использовании холостых патронов так и не последовало. Тоска кончает жизнь самоубийством. В итоге все три героя оперы мертвы”. Эту историю можно рассмотреть с позиции Теории Игр. Автор продолжает: “Что если бы все было иначе? Тоска и Скарпиа являются участниками одной игры, самой распространенной в теории игр – особой области математики, представляющей связь между биологией и экономикой. Эта игра является предметом особого интереса научных исследований последнего времени: ничего более, чем просто понимание того, почему люди нравятся друг другу. Кроме того, Тоска и Скарпиа сыграли свои партии в этой игре так, как это и должно было получиться в соответствии с Теорией игр, несмотря на печальный исход игры. Как это понимать?

[4]Предложен Меррил Флудом (Merrill Flood) и Мелвин Драшером (Melvin Dresher) в 1950-х

Если А сознается, а В нет, то А будет освобожден, а В получит 5 пять лет тюрьмы, и наоборот

Если оба сознаются, то оба получат по 4 года тюрьмы

Если оба молчат, то оба получат по 2 года.

Очевидно, что 0 лет тюрьмы это наилучшиий исход для каждого в отдельности. Если мы подумаем о выгоде обоих заключенных, то случай “оба получат 2 года” будет лучше чем “оба получат 4 года тюрьмы”, только если вы не считаете тюрьмы “бесплатным отелем”.

7.2.1 Условия наличия дилеммы

Сейчас позвольте мне процитировать описание с сайта[5], где параметры следующие: R это награда за совместные действия. Таким образом, если обо игрока игроки кооперируются, то они получают награду R (например, 3 очка). Если один предаёт, а другой нет, тогда тот кто предает получает T (например, 5 очков) – соблазн предать (Tempation to defects), а другой игрок полчает M (например, 0 очков). Если оба игрока предают, то они оба получат P (например 1 очко) – наказание за взаимное предательство (Punishment for mutual defection).

Если один игрок предает, а другой нет, то первый получает награду за предательство (5 в нашем случае), а другой игрок получает награду простока (0 в нашем случае). Если оба игрока предают, тогда они оба получают нагрда за взаимное предательство (1 в нашем случае).

Возникает вопрос: что же мы должны делать в такой игре?

Наверно вы думаете, что другой игрок будет сотрудничать. Если вы сотрудничаете, тогда вы получите награду за взаимное сотрудничество (3 в нашем случае). Если вы предаете, то вы получите награду равную 5, за предательство. Следовательно, если вы думает, что другой игрок будет сотрудничать, тогда вы будете предавать, чтобы получить нагруду равную 5.

А если вы думаете, что другой игрок будет предавать? Если вы сотрудничаете, тогда вы получаете награду простока (0 в нашем случае). Если вы предаёте, то оба получите награду за взаимное предательство равную 1 очко.
Следовательно, вы думаете ,что если другие предают, то и вы будете.

Таким образом, вы должны предавать незавимо от того, что выберете ваш оппонент

Конечно, такого же мнения придерживается ваш оппонент. И если вы оба придаете, то получаете по 1-му очку, тогда как лучшим исходом будет взаимное сотрудничество с наградой 3 очка. Если каждый будет думать только о себе, то они получать меньшую награду, чем если бы они сотрудничали. Таким образом дилемма и иследование проблемы поиска стратегии способствуют взаимному сотрудничеству.

---

[5]http://www.prisoners-dilemma.com/whatisit.html. Author's name is not visible in the page.

В определении дилеммы заключонного должны сохраняться определнные условия. Значения, которые мы использовали выше, чтобы продемострировать игру, не являются единственными значенимями, которые могут быть использованы, но они соответствуют условиям приведенным ниже.

Во-первых важен порядок выплат. Лучшее что может получить игрок это T (соблазн предать). Худшее что может получить игрок это награда простака, S (sucker payoff). Если Оба игрока сотрудничают, тогда они получают награду за взаимное сотрудничество, R, которое должно быть лучше чем награда за взаимное предательство, P. Следовательно следующее условие должно сохраняться.

$$T > R > P > S$$

Во-вторых, игроков не должно быть возможности выйти из ситуации, таким образом, чтобы эксплуатировать друга друга по очереди. Иначе говоря, игроки не должны играть так, чтобы первый половину времени эксплутировал второй, а второй половину времени первого. Другими словами, одинковая вероятность быть эксплуатированным или эксплуатировать не должна быть больше среднего для предательства(T) и награды простака (S). Тогда следующее условие должно сохраняться.

$$R > (S + T)/2$$

Однако дилема не такая серьезная как у Гамлета: “Быть или не быть, вот в чем вопрос.” Лучше всегда признаваться.

С другой стороны, если игра итерационная, тогда мы смотрим на игру иначе. Как любые переговоры, вероятно, могут иметь дилемму – сотрдничать или предавать?

7.2.2 Итеративная дилемма заключенного

Но как быть, если игра повторится и, возможно, несколько раз? Такая ситуация известнакак Итеративная Дилемма Заключенного (Itereted Prisoner’s Dilemma - IPD). В такой игре выбирается стратегия, которая в результате позволит получить наибольшую выгоду (награду). Но какое действие выбрать на определенном этапе игры? Например, можно остановиться на стратегии типа «Всегда Предательство» или стратегии «Услуга за Услугу», в которой игроки сотрудничают в первой игре, а в дальнейшем просто повторяют шаги оппонента в предыдущей игре.

Рассмотрим ситуацию, когда стратегия выбирается на основании трех предшествующих шагов двух игроков. Количество всевозможных вариантов трех предшествующих игр составляет 26 = 64 — 64 комбинаций Сотрудничества и Предательства. Все эти возможные комбинации 6 предыдущих шагов могут быть представлены 64-битной хромосомой. Например, пусть история 6 шагов некоторого игрока и его оппонента задана в виде C-d-Dd-C-d. Тогда в бинарном

представлении эта последовательность задается следующим двоичным числом - 100010, где “C” и “c” соответствуют 1, “D” и “d” соответствуют 0, прописные “C” и “D” - действия оппонента, строчные “c” и “d” - действия игрока, C – “Сотрудничество”, D – “Предательство”.

Таким образом, первая бинарная последовательность задается в виде (000000). Для получения следующей последовательности устанавливаем первый бит в 1, что означает Сотрудничество. Теперь, когда имеется история (000001), меняем второй бит – и получаем (000010), третий бит и т.д. Повторяем подобные действия пока не будет получен последний вариант – (111111). Используя этот нехитрый алгоритм можно без особого труда перечислить все 64 возможные комбинации.

Далее для каждой такой последовательности (хромосомы) случайным образом выбираются p хромосом, и подсчитывается количество побед рассматриваемой хромосомы над отобранными. Полученное значение количества одержанных побед используется для отбора оптимальной стратегии (т.н. p- турнирная селекция).

7.2.3 Пример хромосомы

Допусти, например, что игра имеет вид:

A	0	1	1	Моя очерь. Что я должен выбрать 1 или 0?
B	0	1	0	

История в данном случае:

0, 1, 1, 1, 0, 0

Тогда бинарное представление 011100 переводится в десятичное 28.

Хромосома имеет вид:

001010111100101010101001101010010101110100001111011110000011111101

Действие дожно совпадать с 28-м битом (считаем с младшего бита, т.е. справа) в хромосоме, это 1. Следовательно сотрудничаем.

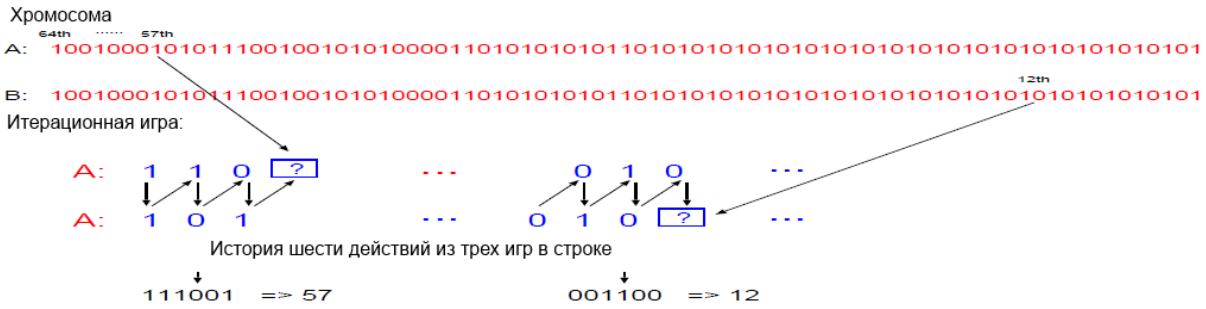


Рисунок 14. Двумерное графическое представление функции Shcwefel

7.2.4 Турнирная селекция

Предположим популяция состоит из N игроков(например 40). Каждый игрок выбирает T (например 5), предать других игроков. Приспосоленность каждого игрока это число побед, или количество очков, полученное во время T игр.

8 Схема сортировки
- Каково число элементарных операций сравнения?

Кто умнее? – Человек или Компьютер?

При написании алгоритма, часто возникает необходимость отсортировать набор элементов, упорядочив их в соответствии с некоторым критерием. Как, например, выполнить задачу сортировки 16 целочисленных значений в порядке возрастания? Мы выберем и сравним два элемента массива. В случае, если порядок не удовлетворяет условию сортировки, поменяем их местами и т.д

Алгоритм 1 (Алгоритм Сортировки) Предположим, необходимо отсортировать N числовых элементов в порядке от меньшего к большему.

Для i = 1 до N-1

⋆ *Для j = i+1 до N*

если item(i) < item (j) тогда меняем местами item(i) и item(j)

Теперь представим вышеописанную сортировку графически:

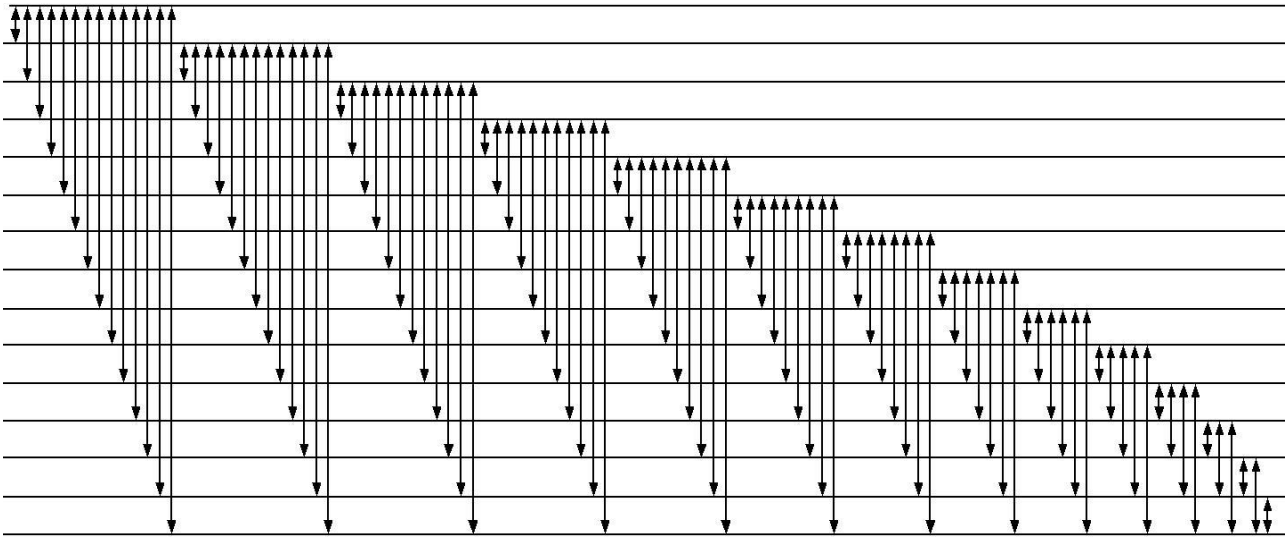


Рисунок 15. Типичная схема сортировки, но недостаточно эффективная.

Общее количество сравнений в данном случае 120, впрочем, это не самый эффективный метод. Поэтому вопрос заключается в том, чтобы подсчитать минимальное число операций сравнения, при котором любой произвольный набор из 16 значений будет правильно отсортирован.

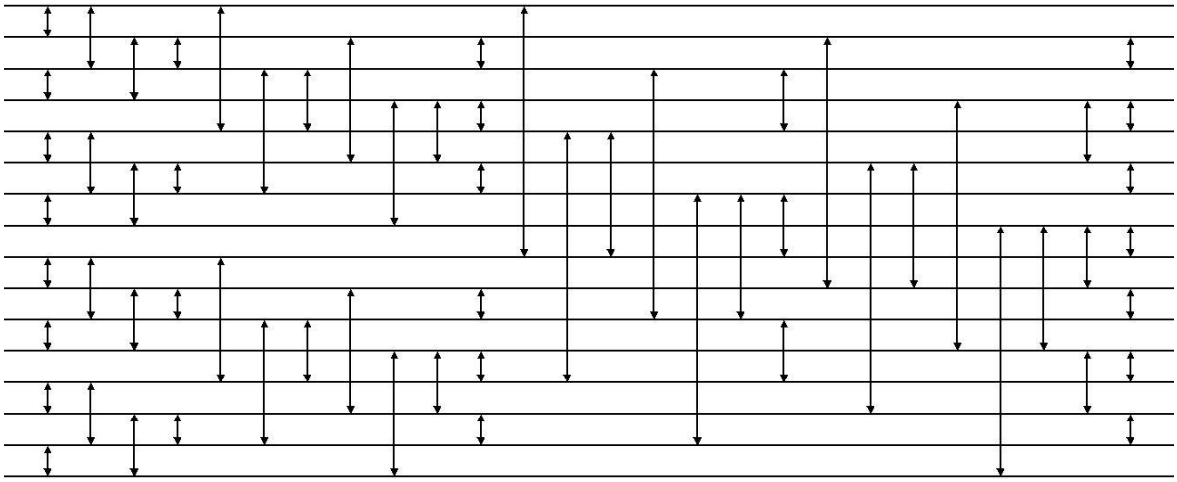
Задача (Схема Сортировки). Нужно отсортировать n элементов. Для этого сравниваются значения i-го и j-го элемента и, при необходимости, элементы меняются местами. Цель заключается в поиске алгоритма, способного за минимальное количество операций сравнения отсортировать исходное множество из n элементов.

Возможно, будет интересно ознакомиться с предысторией этого вопроса. В 1960-х было проведено соревнование на предмет поиска алгоритма, способного за минимальное количество операций сравнения обработать некоторый массив элементов (например, когда их количество равно 16, n=16). Были получены следующие результаты:

- 65 сравнений (Bose and Nelson, 1962).
- 63 сравнения (Batcher, Floyd and Knuth, 1964).
- 62 сравнения (Shapiro, 1969)
- 60 сравнений (Green 1969)

Смотрите рисунок ниже.

Batcher сортировка: 63 сравнения (Knuth 1973):



Сравнения в одинаковых колонках могут быть выполнены параллельно

Рисунок 16. Предложенная Batcher'ом схема сортировки с 63 сравнениями (1964)

Однако, до сих пор не существует доказательства оптимальности этого решения. Тогда, давайте, попробуем применить Эволюционные Вычисления для поиска этого минимального числа операций. Окажется ли результат применения такого подхода эффективнее результата предложенного человеком? В 1992 году Hillis проводил свои исследования в этой области. Он использовал особую хромосому Диплоидия (Diploidy Chromosome), детальное описание которой дадим позже. Здесь же рассмотрим упрощенный вариант применения генетического алгоритма.

Предположим одна хромосома, соответствующая определенной схеме сортировки, состоит из 140 генов, каждый из которых принимает целочисленное значение от 1 до 16, допуская повторения, например:

(12 01 05 04 16 12 04 14 01 02 06 ..... 07 15 08 10)

здесь гены с нечетными номерами образуют пары с соседними справа генами с четными номерами, как показано ниже:

12 <=> 01; 05 <=> 04; 16 <=> 12; .....; 08 <=> 10)

Таким образом, одна хромосома описывает максимум 70 операций сравнения. Почему максимум? Да потому что одни и те же пары могут встречаться несколько раз. Отсюдаследует, что минимальное количество операций сравнения равняется единице, хотя такая ситуация маловероятна.

9 Визуализация многомерного пространства

Порой очень важно наглядно отобразить информацию, заданную в многомерном пространстве. Вполне возможно вы уже знакомы с такими методами как Самоорганизующиеся Карты Кохонена (Kohonen's Self Organizing Map - SOM) или Метод Главных Компонент (Principal Component Analysis - PCA).

9.1 Почему необходимо понижать размерность?

Люди не в состоянии представить мир в более чем трехмерном пространстве. Однако во многих сферах науки особенно важно воспринимать картину пространства высокой размерности. Но помимо науки такая задача часто возникает и в повседневной жизни.

Приведем пример. Допустим, необходимо определить специализацию для каждого вновь прибывшего солдата в зависимости от его способностей к Математике и Английскому языку.

	A	B	C	D	E	F	G	H	I	J	K	L	M
Математика	95	32	89	52	12	20	3	99	42	91	26	95	60
Английский	92	90	21	48	14	5	11	97	50	92	89	13	55

Таблица 17. Результаты новобранцев по 2-м экзаменам

Задача классификации солдат станет проще, если полученные на экзаменах результаты отобразить графически

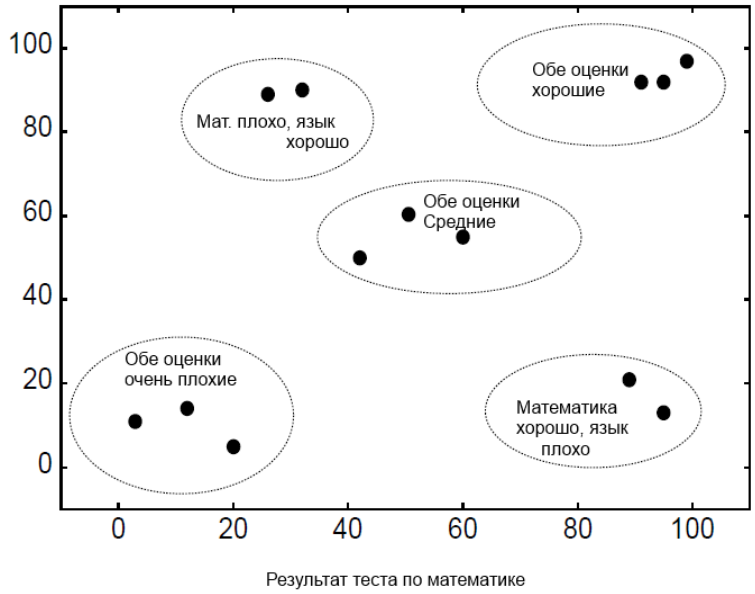


Рисунок 18. Результат визуализации. Очень легко классифицировать солдат на 5 групп

Но как быть, если у нас имеются оценки еще по одной дисциплине, например, по физической подготовке. Так для разведки хорошая физическая подготовка

является оченьполезным качеством. В таком случае нам необходимо рассматривать трехмерные данные, и отображать их придется в трехмерном пространстве. Усложним задачу, представив, что оценки выставлены по 10 экзаменам. Визуализировать в обычном виде такой набор данных уже невозможно.

Таким образом, визуализация многомерного пространства, а также задача сокращения размерности очень важны. Было предложено большое количество методов по этой тематике, и Самоорганизующиеся Карты Кохонена является одним из самых широко распространенных.

9.2 Sammon Mapping при помощи генетического алгоритма

В данном разделе рассматривается т.н. Отображение Sammon. Отображение Sammon – это отображение набора точек многомерного пространства в двумерное пространство с насколько это возможным сохранением соотношения расстояний между элементами исходного пространства. Другими словами, задача состоит в аппроксимации расстояний исходного n-мерного пространства соответствующими расстояниями в 2-мерном пространстве с минимально возможными потерями.

Метод был предложен в 1980-х в качестве задачи оптимизации, к которой был применен не самый простой алгоритм Наискорейшего Спуска из области Исследования Операций. С другой стороны применить Эволюционные Вычисления в этой ситуации гораздо проще. Разберемся теперь, что представляет собой Sammon Mapping:

Алгоритм (Sammon Mapping)

1. *Допустим, заданы N точек в n-мерном пространстве*

2. *Рассчитаем матрицу расстояний R (N X N), где элемент в позиции i-j – Евклидово расстояние между i-ой и j-ой точкой.*

3. *Определим также N точек в двумерном пространстве и для начала распределим их случайным образом.*

4. *Высчитаем матрицу расстояний Q, аналогично матрице R.*

5. *Рассчитывается матрица ошибок, как $P = R - Q$.*

6. *Осуществляется поиск позиций N точек в двумерном пространстве таким образом, чтобы минимизировать суммарное значение элементов матрицы P.*

Таким образом, мы имеем дело с задачей оптимизации, которая, как нам известно, может быть легко решена при помощи Эволюционных Вычислений.

Наша задача заключается в поиске N точек в 2-мерном пространстве, которые соответствовали бы N точкам в n-мерном пространстве. При чем при отображении необходимо сохранить, на сколько это возможно, все соотношения расстояний между точками исходного n-мерного пространства, т.е. выполнить аппроксимацию с минимальной ошибкой.

При реализации генетического алгоритма для решения поставленной задачи хромосомы должны включать n генов, каждый из которых соответствует искомой координате x-y точки в 2-мерном пространстве. Применяется операция равномерного скрещивания, а время от времени – мутация, замещающая один ген случайной координатой x-y (см. Рисунок 2, см. Рисунок представленный ниже).
Пример для 492 = 2401 мерного пространства:

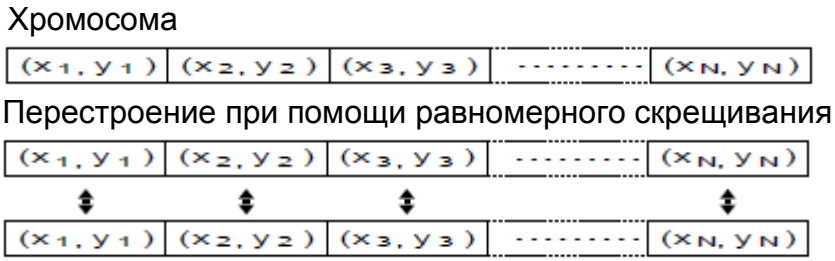


Рисунок 19. Представление хромосомы и равномерное скрещивание

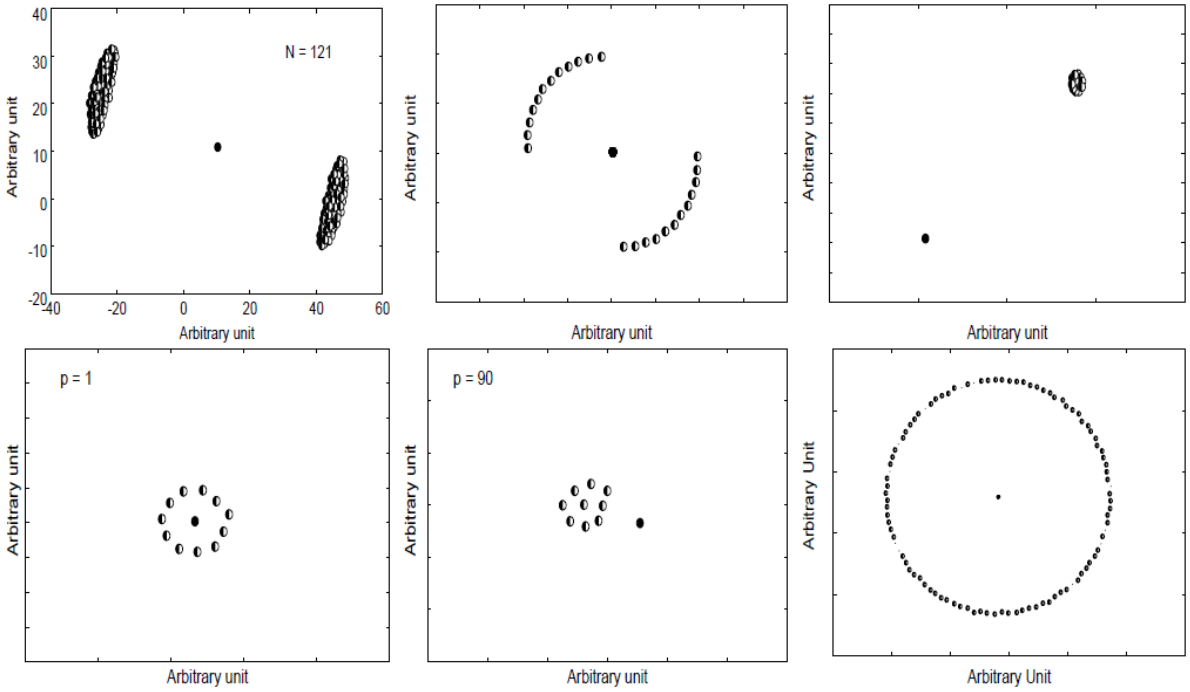


Рисунок 20. Шесть прмеров Отображения из 241-мероного пространства в 2-мероное пространство. Разъяснения приведены в тексте.

10 Мультимодальная проблема

Как быть, если у нас имеется множество значимых решений?

В задаче Коммивояжера нам было важно найти только самый короткий путь из множества полученных решений. Однако иногда нас интересуют все возможные решения, полученные в процессе вычислений. Например, когда мы хотим получить множество нечетких правил для того, чтобы спроектировать нечеткий контроллер. Тема данного раздела посвящена этой проблеме. Давайте начнем с простых математических функций.

10.1 Еще одна функция

Двумерная функция с большим количеством пиков

Вопрос в том, как сформировать хромосому. В случае многомерной функции $y = f(x_1, x_2, , x_n)$ каждый ген может соответствовать переменной x_i (i = 1, 2, n) функции, таким образом, мы имеем n генов в хромосоме. Из-за того, что у нас имеется достаточное количество генов n, проблем не возникает. С другой стороны, как нам поступить в ситуации, когда в наличии только один ген. Использовать хромосому из одного гена? А как тогда выполнить операцию скрещивания?

Хорошо, в такой ситуации мы обычно используем хромосомы в бинарном представлении. Любое (десятичное) реальное значение переменной $x_i \in [a, b]$ может быть закодировано n-битной двоичной последовательностью, где a и b представлены соответственно (00 ⋯ 0) и (11 ⋯ 1). Точность вычислений при использовании такого подхода можно оценить по формуле $(b-a) / (2^n-1)$. Например, если наше значение принадлежит диапазону $x \in [0, 1]$, то при использовании 10-ти бит можно задать битовые строки от 0000000000 и до 1111111111. В этом случае точность десятичных чисел будет составлять 1/1024. Кроме того, вы можете воспользоваться Кодом Грея, где код-грея a1a2…an получается из двоичного числа b1b2…bn, по следующему правилу

$$a_i = \begin{cases} b_i & \text{if} \quad i = 1 \\ b_{i-1} \oplus b_i & \text{otherwise} \end{cases} \tag{3}$$

где $\oplus$ - сумма по модулю 2. В коде Грея расстояние Хемминга для пары смежных десятичных чисел отличается на 1, что в общем случае не выполняется для стандартной бинарной кодировки.

Функции, рассмотренные нами в разделе 4, особенно хорошо подходят для выполнения эволюционных вычислений, так как мы можем легко применить их к данным большой размерности, просто увеличив количество генов в хромосоме. Но как быть в случае, когда задана 2-мерная функция? Например,

$$y = \sin^6(5\pi x) \tag{4}$$

или

$$y = -2((x - 0.2)/0.8)^2 \sin^6(5\pi x) \qquad (5)$$

Эти функции представляют для нас особый интерес, поскольку мы можем исследовать на их примере поведение популяции хромосом, эволюционирующих в направлении поиска пиков (см. Рисунок 2).

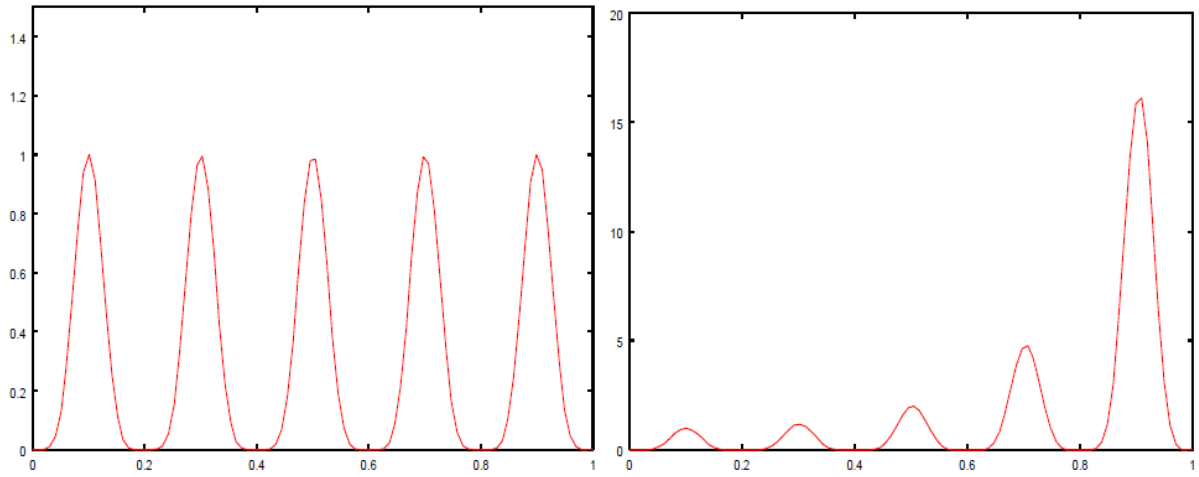


Рисунок 21. Мультипиковая 2-мерная функция и ее разновидность.

10.2 Мультимодальная оптимизация

Порой мы имеем множество решений. Но эволюционные вычисления зачастую сходятся только в одном из них. Поэтому для получения всех решений нам приходится выполнять алгоритм несколько раз. В этом подразделе будет рассмотрен метод, в котором формируются своеобразные ниши для каждого вида особей, представляющих отдельное решение.

Для этого, особи популяции разделяются на несколько видов. Каждый вид независимо от особей других видов выполняет поиск пиковых значений (в надежде обнаружить оптимальное решение), формирует свою нишу около некоторого пика и придерживается ее на протяжении остального времени выполнения программы.

Некоторое время назад, как правило, использовались следующие три метода. Это:

- Fitness sharing (Goldberg & Richardson, 1987)

  - Схожие особи делят оценку приспособленности друг с другом.

- Crowding (De Jong, 1975)

  - Схожие особи замещаются случайными особями

- Species Method.

  - Скрещивание ограничивается кругом схожих особей.

На сегодняшний день, по моему мнению, наиболее популярны следующие два метода:.

- Detterministic Crowding (Mahfoud, 1992)
- Sequential Niching

Рассмотрим более подробно каждый из этих методов.

Fitness sharing: Приспособленность каждой особи снижается в зависимости от количества схожих особей в популяции. Это означает, что долевая оценка приспособленности $F_s(i)$ для i-ой особи рассчитывается по формуле F(i)

$$F_s(i) = \frac{F(i)}{\sum_{j=1}^{\mu} s(d_{ij})}$$

где F(i) – оценка приспособленности i-ой особи; d_{ij} – расстояние между i-ой и j-ой особью; обычно dij задается расстоянием Хемминга (в случае генотипического пространства) или Евклидовым расстоянием (в случае фенотипического пространства) и s(·) называется функцией разделения, которая задается:

$$s(d_{ij}) = \begin{cases} 1 - (d_{ij}/\sigma_{\text{share}})^{\alpha} & \text{Если } d_{ij} < \sigma_{\text{share}} \\ 0 & \text{Иначе} \end{cases}$$

где σ_{share} - интерпретируется как размер ниши, а α - определяет форму функции. Этот знаменатель называют еще отсчетом ниши. На Рисунке 10.2 можно видеть, как форма $s(d_{ij})$ зависит от значения α.

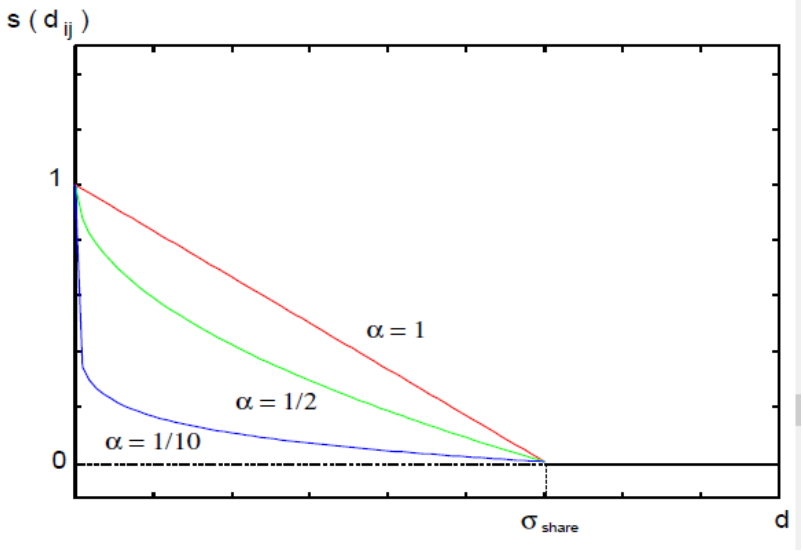


Рисунок 22. Зависимость формы $s(d_{ij})$ от α.

Проще говоря, схожие особи делят оценку приспособленности. Количество особей, которые концентрируются около некоторой вершины (ниши), ограничено. Теоретически их количество должно быть пропорционально высоте пика.
Deterministic Crowding: в этом случае вопрос о замещении родителей потомками решается в зависимости от значения расстояния между ними.

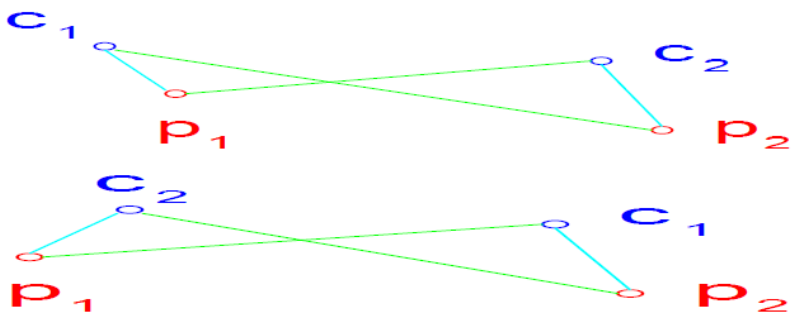


Рисунок 23. Два типичных примера расстояний между родителями и потомками.

Алгоритм. Пред+положим, что операции скрещивания, мутации, а также функция приспособленности уже заданы.

1. *Случайным образом выбирается пара родительских особей, p_1 и p_2, при чем ни один родитель не может быть выдран более одного раза.*
2. *Генерируются два потомка c'_1 и c'_2*
3. *Выполняется мутация и скрещивание потомственных особей. В результате получаем c_1 и c_2*
4. *Замещение родительской особи потомком происходит по следующим правилам:*

- Если $d(p_1, c_1) + d(p_2, c_2) > d(p_1, c_2) + d(p_2, c_1)$
  - * Если $f(c_1) > f(p_1)$ тогда меняем p_1 на c_1
  - * Если $f(c_2) > f(p_2)$ тогда меняем p_2 на c_2
- Иначе
  - * Если $f(c_2) > f(p_1)$ тогда меняем p_1 на c_2
  - * Если $f(c_1) > f(p_2)$ тогда меняем p_2 на c_1

где $d(\zeta_1, \zeta_2)$ – расстояние Хемминга между двумя особями (ζ_1, ζ_2). Формирование потомков продолжается до тех пор, пока все особи популяции не примут участие в этом процессе. Далее те же действия по воспроизводству новой популяции и поиску циклически повторяются до тех пор, пока не будет найдено оптимальное решение или не будет превышено заданное количество популяций.

Sequential Niching: На каждом цикле выполнения алгоритма выбирается наилучшая особь.

Algorithm

1. *Определяется ниша радиусом r.*
2. *Составляется модифицированная функция приспособленности m(x), в*

результате присваивания исходной функции приспособленности f(x) (см. выше).

3. *После выполнения генетического алгоритма выбирается наилучшая особь.*
4. *Пересчитывается значение m(x) по формуле*[6]

$$m_{n+1}(\mathbf{x}) = m_n(\mathbf{x}) \cdot g(\mathbf{x}, \mathbf{s}_n) \tag{7}$$

где n - число выполненных циклов алгоритма, sn - лучшая особь, отобранная после n-ого цикла и

$$g(\mathbf{x}, \mathbf{s}_n) = \begin{cases} (d_{xs}/r)^{\alpha} & \text{Если } d_{xs} < r \\ 1 & \text{Иначе} \end{cases} \tag{8}$$

является функцией понижения значения, в которой d_{xc} – расстояние между x и sn. Под m_0 понимается исходная не модифицированная функция приспособленности каждой особи.

5. *Выполняется генетический алгоритм с использованием модифицированной функции приспособленности и выбирается наилучшая особь.*
6. *Обновляется модифицированная функция приспособленности.*
7. *Если не модифицированная оценка приспособленности выбранной особи превосходит заданный порог (см. ниже), то получено окончательное решение.*
8. *Если не все решения найдены, то возвращаемся к шагу 5 алгоритма.*

• *Порог решения (Solution threshold) это*

- *В том случае, когда известно количество вершин, можно поэкспериментировать, понизив предел оценки приспособленности. В ином случае устанавливайте порог равным 0.*

Задание Подобно рисунку 10.2, постройте график функции $y = (x/r)^{\alpha}$, где r = 1 и α = 0.5, 1, 2, 4, 8. Это позволит продемонстрировать, как выглядит $g(x, s_n)$.

Так же будет интересно поэкспериментировать с мультимодальными эволюционными вычислениями для следующих двух функций:

---

[6]Это называется Степенная функция понижения. Альтернативой данной функции может служить Экспоненциальная Функция Понижения.

$$g_e(x, s) = \begin{cases} \exp((\log m(x, s)) \cdot (r - d_{xs}/r)) & \text{Если } d_{xs} < r \\ 0 & \text{Иначе} \end{cases}$$

(1) $y = \sin^6(5\pi x)$

(2) $y = -2((x - 0.2)/0.8)^2 \sin^6(5\pi x)$

11 Многоцелевой Генетический Алгоритм

До сих пор мы рассматривали как получить возможное решение(-ия) для одной целевой функции, допустим, максимизируя значение функции приспособленности. Однако в реальных задачах, как правило, нас интересуют несколько целей или критериев одновременно.

Часто цели противоречат друг другу. Например “время” и “деньги”: чем больше мы хотим зарабатывать, тем меньше остается времени тратить деньги; или ”надежность” продукции и ее “цена” при фабричном производстве. Или, предположим, необходимо подобрать певца для партии сопрано в опере. Критерием выбора являются: красота голоса, стройность фигуры, владение языками (итальянский, немецкий и т.д.). Увы, Бог не сделал нас талантливыми во всем.

Когда мы имеем дело с многоцелевой функцией, то в первую очередь должны разработать саму стратегию получения оптимального решения удовлетворяющего множеству функций или, другими словами, “недоминируемого решения”.

Определение: под “недоменируемым” или “Паретто-оптимальным” решением понимается такое решение, которое является наилучшим с учетом всех целевых функций.

Предположим, мы имеем n целевых функций:

$$f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}), \cdots f_n(\mathbf{x})$$

где x - возможное решение. Если новое возможное решение y окажется лучше решения x, т.е.,

$$f_i(\mathbf{y}) > f_i(\mathbf{x}) \text{ for } \forall i$$

мы можем сказать

“y доминирует над x”

Если же такое y отсутствует, мы скажем

“x недоминируем” или “Паретто-оптимален”

Небольшой пример: Допустим, у нас имеются две целвые функции.

$$\begin{cases} f_1(x) = x^2 \\ f_2(x) = (x-2)^2 \end{cases}$$

· x=0 – является оптимальным решением функции f1, но не для f2.

· x=2 – является оптимальным решением функции f2, но не для f1.

	test-1	test-1	test-1	test-1	test-1	доминируем	доминирует	ранг
A	9	9	9	8	7	0	3	1
B	5	4	5	3	6	1	2	2
C	3	3	4	2	3	2	1	3
D	2	2	3	1	2	3	0	4
E	1	1	1	1	9	0	0	4

Таблица 24. Пример того, кто доминирует над кем и как считается ранг.

· Любое значение между двумя указанными будет компромиссным решением или Паретто-оптимальным.

· Однако решение x=3 не является Паретто-оптимальным, поскольку эта точка не лучше решения x=2 для заданных целевых функций.

· Если мы построим график в пространстве решений f_1-f_2, то увеличение f1 на некотором интервале повлечет уменьшение f_2, и, наоборот, что означает, что решения в этом интервале будут Паретто-оптимальными. В другом же интервале значений функции f_1 увеличение f_1 приведет к росту значения функции f_2 (и наоборот). См. Рисунок 32. Пространство f_1-f_2 еще называется Пространством Компромиссного Решения (Trade-off Space).

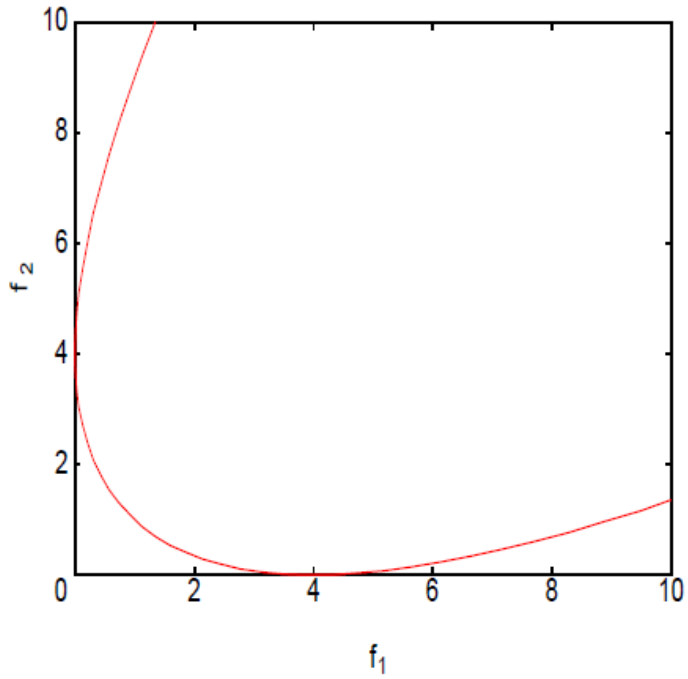


Рисунок 25. Пространство Компромиссного Решения функций $f_1(x) = x^2$ и $f_2(x) = (x - 2)^2$

Тема для размышления: Интересно попробовать графически отобразить все особи популяции 0 и, допустим, популяции 100?

Как использовать несколько целевых функций? Эволюционный процесс в этом случае в чем-то похож на применение генетического алгоритма с одной функцией приспособленности. Единственное отличие в том, что мы имеем несколько целевых функций. Далее мы объединяем множество целевых функций в единую функцию приспособленности. Существует множество способов решения этой задачи. Перечислим некоторые их них:

- Использование взвешенной суммы.

  Приспособленность расчитывается по формуле:

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i f_i(\mathbf{x}) \qquad (9)$$

  где w_i характеризует важность i-й целевой функции.

  Обратите внимание, что для любого множества весов > 0, оптимально е решение «недоминирумео», однако обратное утверждение не всегда верно.

- Минимаксный подход

  Приспособленность расчитывается как результат минимизации максимума n целевых функций.

- Использование вектора целей

  Функция приспособленности рассчитывается посредством минимизации вектора

$$(f_1, f_2, f_3, \ldots, f_n)$$

  от заранее заданной цели.

- Подход усредненного ранжирования

  В данном случае рассчитывается ранг $r(x_i)$ для каждой особи x_i в популяции с учетом i-ой целевой функции. Оценка приспособленности определяется как усредненное значение всех r_i (i=1,…,n).

- Подход ранжирования Паретто

  Ранжирование выполняется согласно “количеству особей в популяции, которые над ними доминируют”.

Рассмотрим общий алгоритм применения многоцелевого генетического алгоритма.

Алгоритм (многоцелевой генетический алгоритм)

1. *Создаем начальную полпуляцию.*
2. *Выбор особей из популяции.*
3. *Рвыполнение скрещивания и мутации для создания потомка.*
4. *Расчет ранга для полученного потомка.*
5. *Поиск особи в популяции максимально похожей на сформированного потомка. Замещение этой особи потомком, в случае если ранг потомка выше или если потомок доминирует*[7]
6. *Если потомок был принят, то пересчитываем ранги в популяции.*
7. *Повторяем шаги 2-6 в соответствии с размером популяции*
8. *Если результат не был достигнут, то вернуться к шагу 2 и начать новую итерацию.*

---

[7] На 5-ом шаге алгоритма добавление нового потомка в популяцию происходит только при условии, что он доминирует над наиболее схожей с ним особью, или если она имеет более низкий ранг, т.е. ниже степень доминирования.

Такая стратегия привносит элементы элитизма в вычисления, поскольку недоминируемая особь

может быть замещена только потомком, который доминирует над ней.
Степень подобия двух особей рассчитывается через функцию расстояния

Приложение

I. Наиболее чато используемые фукнции приспособленности

Функция Shcwefel

Функция, представленная ниже, называется Shcwefel-функцией. Она характеризуется большим количеством локальных минимумов и одним глобальным минимумом в точке начала координат. Найдите глобальный минимум, когда функция задана, допустим, в интервале $x_i \in [-500, 500]$.

$$y = \sum_{i=1}^{n} (x_i \sin(|x_i|)) \tag{10}$$

Например, для начала можно опробовать эту функцию при n=20, т.е. когда поверхность определена в 20-мерном Евклидовом пространстве. Если вы все же хотите увидеть график этой функции, то можно воспользоваться ее двумерным представлением. График функции

$$y = x \sin(|x|) \tag{11}$$

показан на рисунке 26.

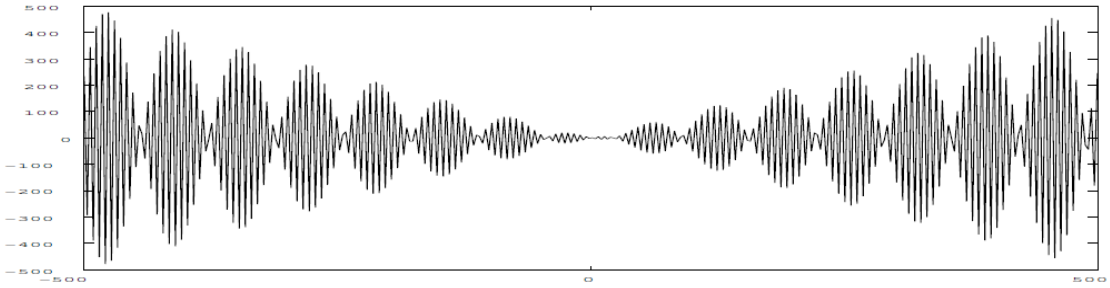


Рисунок 26: График двумерной версии функции Schwefel.

11.1 Функция Растригина

⋆ Функция Растригина

$$y = nA + \sum_{i=1}^{n} (x_i^2 - A\cos(2\pi x_i)), \quad x_i \in [-5.12 - 5.12]$$

• Величина неровностей может быть изменена значением A.

•

• Двумернуй пример (n = 1):

$$y = 3 + x^2 - 3\cos(2\pi x)$$

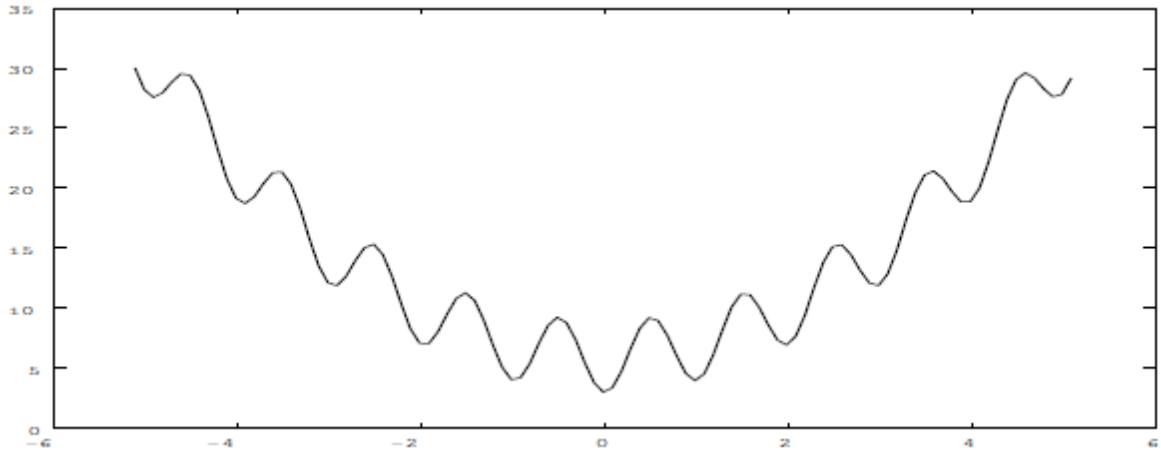


Рисунок 27. Двумерная версия функции Растригина.

⋆ Функция Griewangk F_8:

$$y = \sum_{i=1}^{n} x_i^2/4000 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1, \quad x_i \in [-600, 600].$$

• Двумерный приер:

$$y = x^2/4000 - \cos x + 1.$$

⋆ Функция Ackley F_9:

$$y = -20 \sum_{i=1}^{n} \exp|(-0.2\sqrt{x_i^2/n}) - \exp((\sum_{i=1}^{n} \cos 2\pi x_i)/n) + 20 + e,$$

$$x_i \in [-30, 30].$$

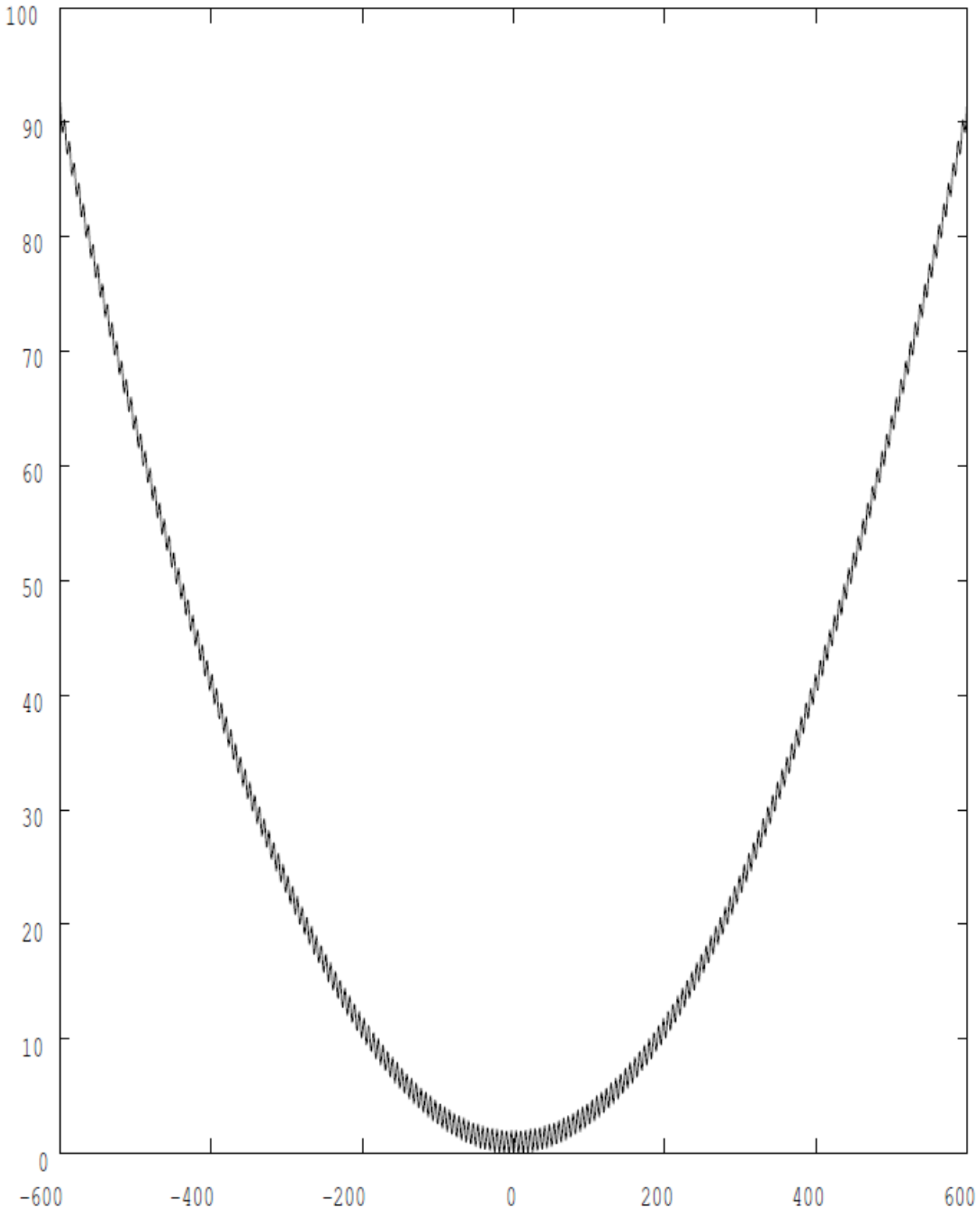


Рисунок 28. Двумерная версия функции Griewangk

• Двумерный приер:

$$y = -20\exp\left(-0.2\sqrt{x^2}\right) - \exp\left(\cos 2\pi x\right) + 20 + e.$$

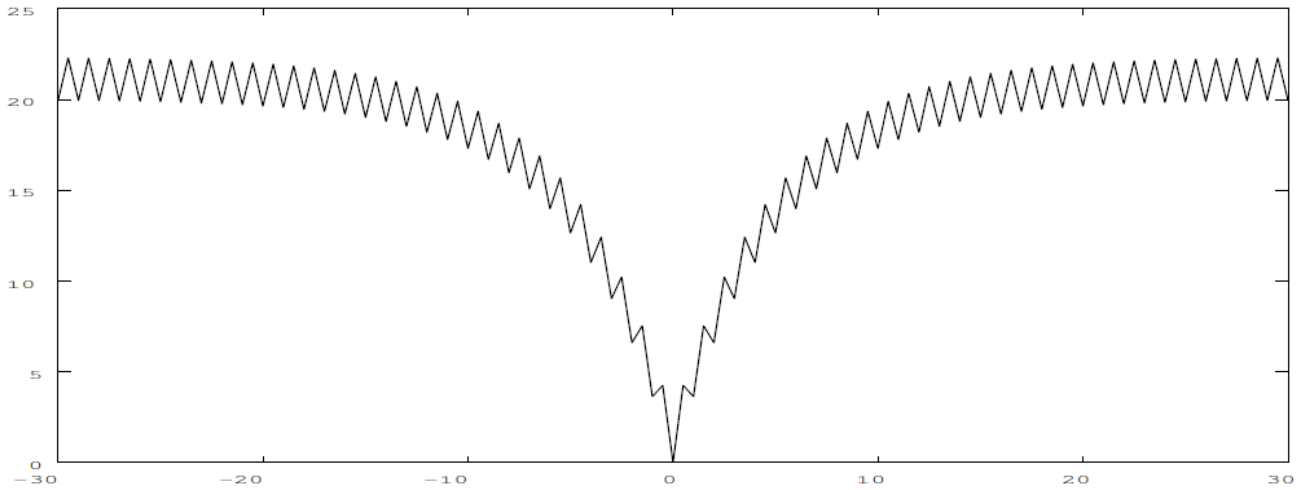


Рисунок 29. Двумерная версия функци Ackley