Конспект лекций по курсу «Современные интеллектуальные информационне технологии»

Акира Имада
Брестский государственный технический университет, Беларусь

Последние изменения:10 мая, 2013
Перевод: Мокин Илья

1 All One Problem (Задача «всех единиц»)

Перед тем как приступить к изучению математических эволюционных моделей проведем простейший экперимент.

Сгенерируем начальную популяцию из 100 хромосом, каждая из которых состоит из 40 генов (все генерируются случайно). Fitness(пригодность) – количество единичных значений в хромосоме – чем больше, тем лучше. Наша цель – хромосома в которой все гены примут значение «1».

Применим стандартные алгоритмы эволюционирования (i) one-point-crossover (скрещивание через произвольную точку) и (ii) uniform-crossover (равномерное скрещивание), с частотой мутации 1/*N* где *N* – число генов в хромосоме.

Algorithm 1 (All-One-Problem)

1. *Создаем, например, 100 бинарных хромосом со случайно проинициализированными, скажем, 40 генами .*
2. *Рассчитаем fitness. Fitness (пригодность) – количество единичных генов в хромосоме. Чем больше, тем лучше.*
3. *Возьмем 2 произвольные хромомсомы из лучшей половины популяции.*
4. *Сгенерируем новую хромосому скрещиванием 2-ух.*

   Сравниете 2 варианта: 1)one-point-crossover 2)uniform-crossover.

5. *Произведем мутацию в новой хромосоме с вероятностью 1/40 = 0.025.*
6. *Повторять шаги 2-5 40 раз, до создания новой популяции.*
7. *Повторять шаг 6 до тех пор пока фитнес не станет равным 40.*
8. *Требуется показать :*

   (1) Лучшие хромосомы в каждой популяции.

   (2) Графики: 1)популяция/лучший фитнес 2)популяция/средний фитнес.

2 Простейшая тестовая функция – Сферическая модель

Первая задача этого задания получить минимум многомерной функции.

Чтобы быть более конкретным, предположим, что следующая функция определена в 20-ти-мерном пространстве.

$$y = x_1^2 + x_2^2 + x_3^2 + \cdots + x_{20}^2. \quad (1)$$

Затем нужно получить точки (x_1; x_2; x_3;…; x_{20}), которые должны давать как можно меньшее значение функции y. Для решения этой задачи, реализуйте следующий алгоритм.

Algorithm 2 (минимизация простейшей многомерной функции)

1. *Создаем, например, 100 бинарных хромосом.*

   Количество генов - 20.

   Тиким образом наши хромомсомы имеют вид (x_1; x_2; x_3;… ; x_{20}).

   Предположим что каждый x_i *принимает значения от -1 до 1*

   $-1 < x_i < 1$.

2. *Рассчитаем fitness хромосомы как* $y = x^2_1 + x^2_2 + x^2_3 + \ldots + x^2_{20}$*: Чем меньше, тем лучше.*
3. *Возьмем 2 произвольные хромосомы из лучшей половины популяции.*
4. *Сгенерируем потомков методом скрещивания.*
5. *Выполним мутацию потомка.*
6. *Повторяем 2-5 100 раз для создания новой популяции.*
7. *Повторяем 6 пока фитнес максимально не приблизится к 0.*

Требуется выполнить:

Задание 1 (Нахождение глобального минимума)

Построить графики: (1)популяция/средний fitness. (2)итерация/минимальное значение fitness.

3 Функция посложнее

Попробуем чуть более сложную функцию. Например функцию Растригина.

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

Размерность n произвольна, но должен быть хорошо виден график функции, так она выглядит при *n* = 1.

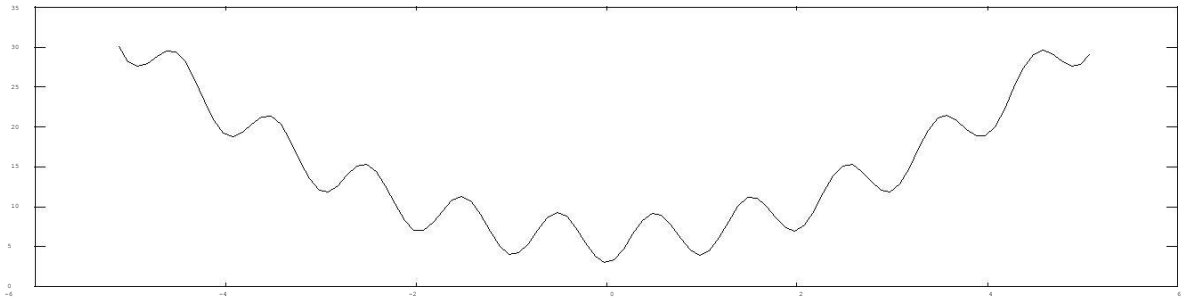

Рисунок 1: 2-D версия функции Растригина

Задание 2 (Нахождение глобального минимума)

(0) При n = 20

(1) *Построить графики: (1)популяция/средний fitness.*
(2)Итерация/Минимальное значение fitness.

(3) Проведите эксперимент с различным значением коэффициента вероятноти мутации.

4 2-D Функция

Теперь опопробуем эволюционный алгоритм на 2-D функции. Попробуйте найти точку минимума на следующей функции

$$y = x^4 - 5x^3 - 6x^2 + 8x + 15$$

График рассмотрим на $x \in [\ 2; 5]$

$$y = x^4 - 5x^3 - 6x^2 + 8x + 15 \text{ with } x \in [-2, 5].$$

Какой дизайн хромосомы использовать для решения этой задачи?

Как и в предыдущей задаче, число генов-n, если функция определена на n-мерном пространстве. Тогда хромосома имеет только один ген? Как тогда возможно скрещивание двух хромосом?

Решением этих проблем будет ипользование бинарных хромосом, как в следюующем примере, ...

5 Neural Network for Even-n-Parity

Even-*n*-Parity – это логическая функция для проверки n-разрядного бинарного множества на истинность. (true/false или 1/0)

Предположим теперь, что n=4, для простоты. Снова бинарное представление состоит из -1 и 1 вместо 0 и 1, для удобства. Следовательно, как и в предыдущем разделе передаточная функция:

$$y_i = 2 \cdot \mathrm{sgn}(\sum_{j=1}^{N} w_{ij} x_j - \theta_j) - 1,$$

где y_i – выход i-го нейрона, w_{ij}-вес синапса от j-го нейрона к *i-тому*, x_j - состояние *j-го нейрона*, θ - *пороговое значение j-го нейрона*, и N – количество нейронов свзанных с i-тым нейроном. Зададим здесь θ_j = 0.5 для всех j.

x_1	x_2	x_3	x_4	y
−1	−1	−1	−1	+1
−1	−1	−1	+1	−1
−1	−1	+1	−1	−1
−1	−1	+1	+1	+1
−1	+1	−1	−1	+1
−1	+1	−1	+1	−1
−1	+1	+1	−1	−1
−1	+1	+1	+1	+1
+1	−1	−1	−1	+1
+1	−1	−1	+1	−1
+1	−1	+1	−1	−1
+1	−1	+1	+1	+1
+1	+1	−1	−1	+1
+1	+1	−1	+1	−1
+1	+1	+1	−1	−1
+1	+1	+1	+1	+1

Теперь реализуем нейронную сеть прямого распространения с 4 входными нейронами, 4 скрытых нейрона, и один выходной. Что ж, это значит, что мы имеем 20 синапсов, которые соответствуют 20 генам.Создадим 100 хромосом со случайными весами от -1 до 1. Fitness оценивается путем подсчета правильных ответов всех возможных 16 случаев, подавая образы на вход нейронной сети из 4 нейронных элементов.

Задание 3 (Neural Network for Even-4-Parity)

Построить графики:
(1)популяция/средний fitness.(2)популяция/максимальный fitness.
(3) Продемонстрировать работу нейронной сети подавая на вход значения с клавиатуры.

6 Navigation in gridworld (Навигация на координатной плоскости)

Предположим, что мы хотим создать агента или робота, который будет уметь перемещаться в какой-либо координатной плоскости. Мы можем создать хромосому из целочисленных генов от 1 до 4, где цифры 1,2,3,4 соответствуют направлению движения агента от текущей точки (верх, низ, право, лево), как на следующем примере:

(31111332332233313144241114 1)

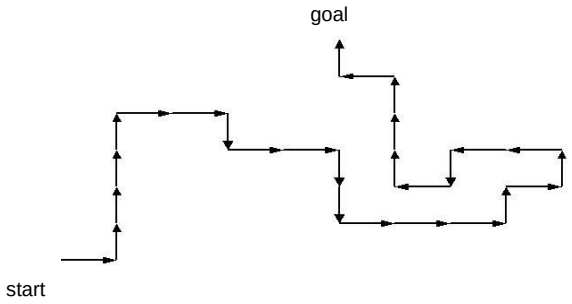

Рисунок 3: Пример хромосомы и результат ее работы.

Поиск максимальной Манхэттенской протяженности (Manhattan distance)
Начнем в центре огромной двухмерной сетки. Робот перемещается на следующий шаг по средствам хромосомы, как было описано выше. Для примера, длина хромосомы 40, таким образом, мы можем исследовать сетку в 40 шагов.

В начале, робот при исследовании перемещается случайно, т.к. его хромосома задана случайным образом.

Некоторые роботы просто будут блуждать вокруг начальной точки, пример такого робота:

(12121212121212121212121212121212121212121212)

Цель состоит в том, чтобы найти агента, с максимальной (40) манхэттенской протяженностью.

Поиск минимальной Манхэттенской протяженности (Manhattan distance)
В этой задаче начальное положение робота и цель которую он должен достичь задается предварительно.

Цель состоит в том, чтобы найти робота, который достигнет цели с минимальной манхэттенской протяженностью. Смотрите рисунок в качестве примера

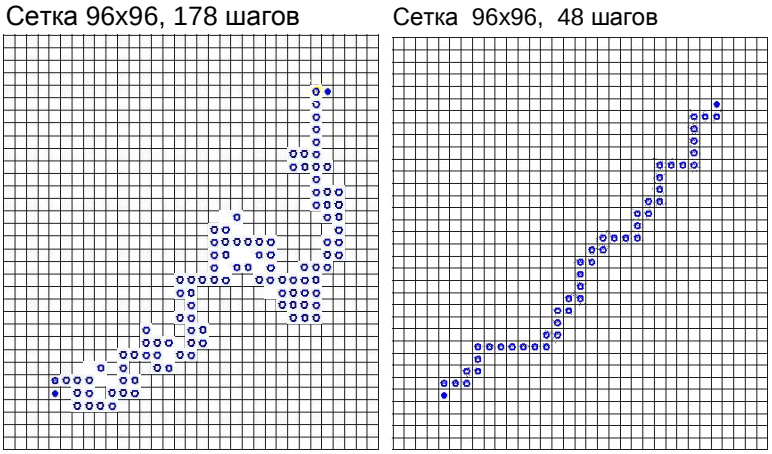

Рисунок 4: В пространстве размерностью 96x96 робот перемещается из клетки в позиции(24,24) в клетку (72,72), о которой ему заранее ничего не известно. Слева: представлен путь минимальной длины, сгенерированный случайным образом и отобранный из 100 экспериментов. Справа: путь минимальной длины, найденный роботом в результате применения эволюционного обучения, как показано на Figure 3. (Крайние области не показаны.)

7 Traveling Salesperson Problem (TSP) Задача о коммивояжере

Предположим, что заданы координаты N городов. Задача коммивояжера (ЗК) заключается в том, что торговец должен посетить каждый из этих городов по одному разу, при этом его путь должен быть минимален.

Рассмотрим 4 города A, B, C и D, в качестве простейшего примера. Предположим, города располагаются следующим образом:

	(*x; y*)
A	(0:83*;* 7:79)
B	(3:28*;* 8:32)
C	(1:52*;* 4:48)
D	(7:65*;* 3:46)

Тогда евклидовое расстояние между всеми возможными парами городов рассчитываются по формуле:

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad (2)$$

где r_{ij} расстояние между городом *i* и городом *j* и (x_i; y_i) и (x_j; y_j) координаты городов *i* и *j*, соответственно. Рассчитаем дистанции:

	A	B	C	D
A	0.000	2.505	3.382	8.074
B	2.505	0.000	4.232	6.539
C	3.382	4.232	0.000	6.214
D	8.074	6.539	6.214	0.000

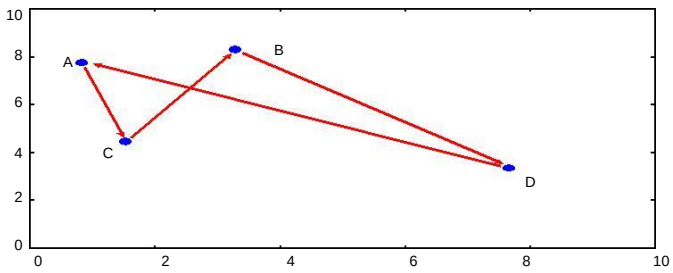

Рисунок 5: Пример с 4 городами и возможными путями в нем.

Все возможные пути в этом примере
(A-B-C-D-A), (A-B-D-C-A), (A-C-B-D-A), (A-C-D-B-A), (A-D-B-C-A), and (A-D-C-B-A).

Заметьте, что длины маршрутов некоторых пар идентичны, например (A-B-C-D-A) и (A-D-C-B-A). Т.е. в общей сложности, мы имеем 3!/2 = 3 маршрута в этом примере.

Давайте теперь рассмотрим маршрут A-C-B-D-A, изображенный на Figure 5.

Длина маршрута показанного на рисунке рассчитывается по формуле:

$$r_{A-C-B-D-A} = 3{:}382 + 4{:}232 + 6{:}539 + 8{:}074 = 22{:}227$$

Так же мы можем рассчитать два других маршрута:

$$r_{A-B-D-C-A} = 2{:}505 + 6{:}539 + 6{:}214 + 3{:}382 = 15{:}640$$

$$r_{A-B-C-D-A} = 2{:}505 + 4{:}232 + 6{:}214 + 8{:}074 = 21{:}025$$

Мы видим, что маршрут с минимальной длиной это A-B-D-C-A (или A-C-D-B-A).

Но что если мы имеем большее число городов? Даже если мы имеем всего 10 городов, это 9!/2 = 181440 возможных различных маршрутов. Вы хотите рассчитать дистанции для всех возможных маршрутов? Конечно же нет! А что если их еще больше, скажем, около 1000 городов?

Так давайте применим эволюционный алгоритм. Однако следует отметить, что хромосома, такая как

(B D C)

маршрута A-B-D-C-A и

(D C B)

маршрута A-D-C-B-A, не будет работать, потому что возможные потомки после *one-point скрещивания между* 1-ой и 2-ой хромосомой могут быть следующими:

(B C B) and (D D C)

Это не возможно, т.к. получаются неправильные маршруты – посещая один из городов дважды, мы пренебрегаем одним из городов.

Рассмотрим такой вариант решения этой проблемы:
Шаг-1. Установим i = 1.
Шаг-2. Если i-й ген равен n, то n-й город является посещенным городом.
Шаг-3. Удаляем город из списка.
Шаг-4. Установим i = i + 1 и повторяем пункты от Шага-2 до Шага-4 пока i ≤ n

Например, Если в списке городов есть, кроме стартового города A

{B, C, D}

хромосома: (121) представляет следующий маршрут:

A-B-D-C-A.

Следует отметить, что гены могут быть любым целым числом и мутация может постой заменой гена с другим случайным числом. Вероятность мутации может быть например 1/количество генов (хотя вы можете его выбрать по своему усмотрению)

Задание 4 (TSP)

(1) *Создайте 14 городов со случайными координатами* (x_i; y_i).

(2) *Рассчитайте дистанции между всеми возможными парами городов.*

(3) *Затем применяйте эволюционный алгоритм.*

(4) *Повторяйте (3) пока значение fitness (оно же длина маршрута) сходится.*

Результаты, которые вы должны показать мне.

Координаты всех городов.

Матрицу дистанций между любыми городами.

График расположения всех городов и маршрут.

8 Knapsack Problem (Задача о рюкзаке)

Предположим, что имеется n предметов, которые можно положить в рюкзак. Каждый предмет имеет вес wi и коэффициент полезности pi. Далее для каждого i-го предмета подбираются неотрицательные целые значения xi, где i=1,2,…,n. Цель заключается в поиске максимума для выражения

$$\sum_{i=1}^{n} x_i p_i. \tag{3}$$

при чем так, чтобы

$$\sum_{i=1}^{n} x_i w_i < C \tag{4}$$

где *C* максимально возможный вес рюкзака.

Применить генетический алгоритм в данном случае достаточно просто. Наша хромосома задается в форме:

$$(x_1 x_2 x_3 \ldots x_n) \tag{5}$$

Где каждый x_i количество *i*-ых предметов помещенных в рюкзак.

Удаление непригодных хромосом
Необходимо отметить что, если хромосома не удовлетворяет выражению (4), то такая хромосома удаляется, а процедура генерации хромосомы потомка (скрещивание, мутация и т.п.) повторяется снова до тех пор, пока не будет получена подходящая хромосома потомка.

Задание 5 (Задача о рюкзаке) *Предположим, что размер рюкзака 60.*

(1) Создайте, например, 100 предметов, присваивая, случайным образом, каждому из них полезность p_i и размер w_i в диапазоне от 0 до 1. Например:

Предмет	Полезность	Размер
1	0.37	0.62
2	0.52	0.45
3	0.95	0.38
…	…	…
100	0.72	0.32

(2) Создайте сорок хромосом, каждая из которых имеет 100 целочисленных генов. Например

(5, 7, 13,… 2)

Что значит: 5 первых предметов, 7 вторых, затем 13 третьих и т.д. до 100.

(3) *Пытайтесь сначала выбирать предметы, полезность которых ближе к 0.99, а размер к 0.01. Представьте, что это алмазы, драгоценные и небольшие. Таким образом, все предметы должны быть как можно ближе к таким.*

(4) Опробуйте эволюционный метод и постройте графики: максимальный фитнес / поколение; средний фитнес / поколение

(5) Визуализируйте содержимое рюкзака.

9 Sammon Mapping by GA (Отображение Sammon генетическим алгоритмом)

В данном разделе рассматривается 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 Отображение Sammon:

Алгоритм (Отображение Sammon)

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:

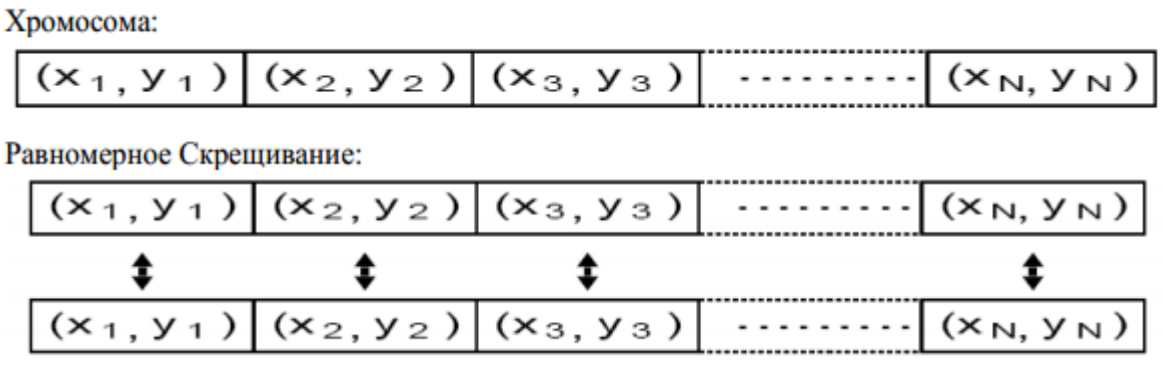

Рисунок 6: Представление хромосомы и равномерное скрещивание

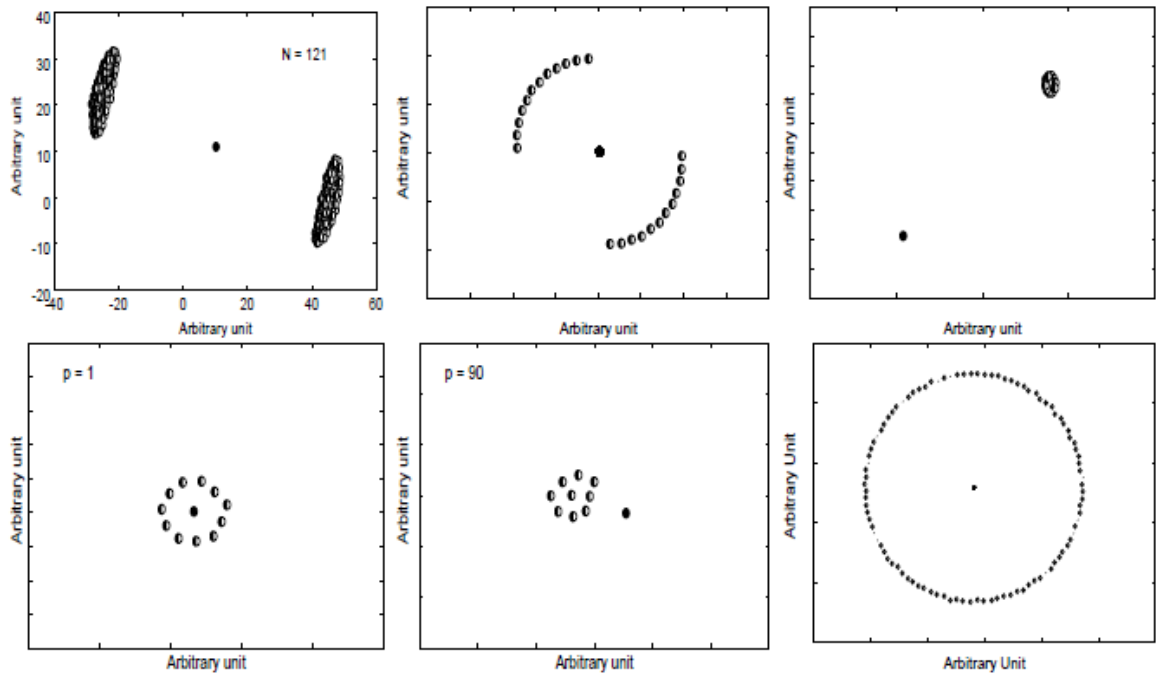

Рисунок 7: Шесть примеров Отображения из 2401-мерного пространства в 2-мерное пространство. Разъяснения приведены в тексте.

10 Multi Modal GA (Мультимодальная проблема) - Как быть, если у нас имеется множество значимых решений?

10.1 Целевые функции

Предположим, что нашей целью является максимизация, т.е. мы хотим знать при каких значениях x функция y принимает максимальное значение, опробуем 2 тестовые функции:

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

И

$$y = -2((x - 0.2)/0.8)^2 \sin^6(5\pi x) \tag{7}$$

Посмотрим как выглядят эти две функции.

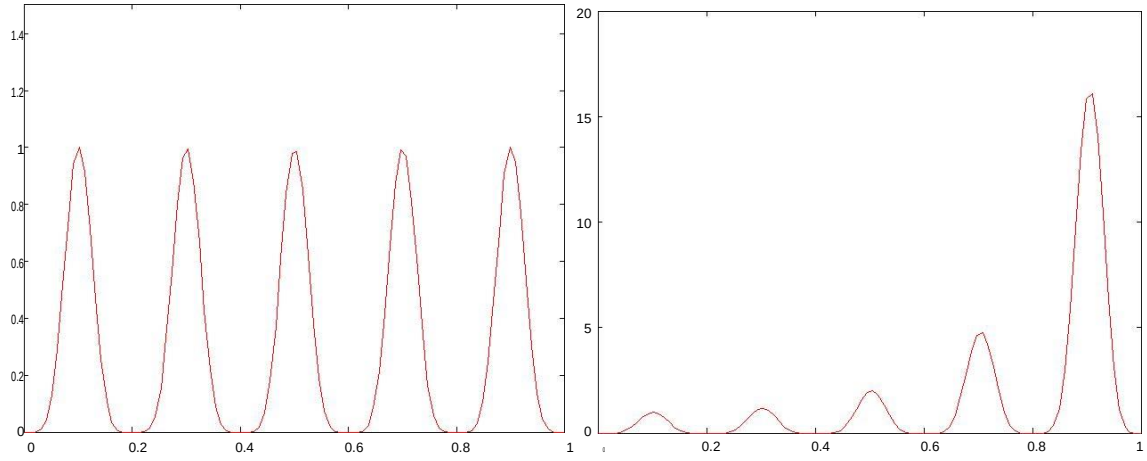

Рисунок 8: Мультипиковая 2-мерная функция и ее разновидность

Рассмотрим 2 алгоритма для решения мульти модальной задачи.

10.2 Fitness Sharing

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

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

где F(i) – оценка приспособленности i-ой особи; dij – 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:

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

где σ_{share} - интерпретируется как размер ниши, а α - определяет форму функции. Этот знаменатель называют еще отсчетом ниши. На Рисунке 26 можно видеть, как форма s(dij) зависит от значения α

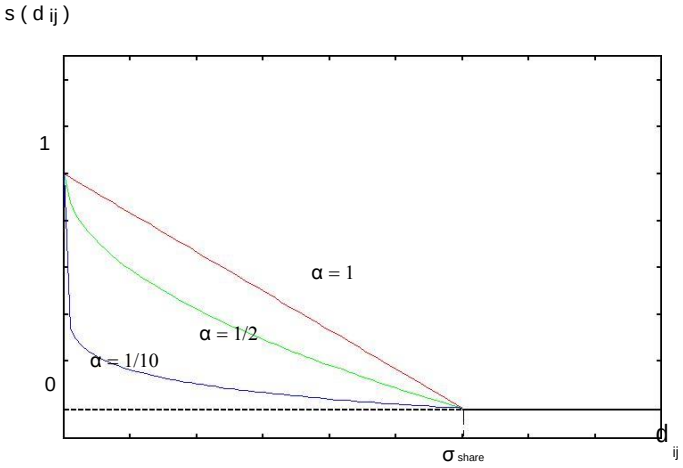

Рисунок 9: Зависимость формы *s*(d_{ij}) от α. .

Проще говоря, схожие особи делят оценку приспособленности. Количество особей, которые концентрируются около некоторой вершины (ниши), ограничено. Теоретически их количество должно быть пропорционально высоте пика.

10.2.1 Результаты, которые вы должны показать.

(1) Постройте таблицу содержащую (i) хромосому, (ii) с ее значениями x, (iii) и *y*, (iv) ее конкретным фитнесом *f*, и (iv) общим фитнесом *F*. Покажите три таких таблицы: в первом поколении, промежуточном и последнем. Смотрите таблицу, приведенную ниже. (2) Так же постройте график фитнесс/поколение.

No.	chromosome	x	y	f	F
#01	(0 1 1 0 1 0 ... 1)	0.34	0.62	0.62	0.48
	. . . . .				
	. . . . .				
#40	(1 0 1 0 0 1 ... 0)	0.86	0.13	0.13	0.23

Рисунок 10: Таблица содержащая (i) хромосому, (ii) с ее значениями x, (iii) и *y*, (iv) ее конкретным фитнесом *f*, и (iv) общим фитнесом *F*.

10.3 Deterministic Crowding

В этом случае вопрос о замещении родителей потомками решается в зависимости от значения расстояния между ними.

Алгоритм. Предположим, что операции скрещивания, мутации, а также функция приспособленности уже заданы.

1. Случайным образом выбирается пара родительских особей, p1 и p2, 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 c1 и c2.
4. Замещение родительской особи потомком происходит по следующим правилам:

- $IF \quad d(p_1, c_1) + d(p_2, c_2) > d(p_1, c_2) + d(p_2, c_1)$
  - $IF \quad f(c_1) > f(p_1) \ THEN \quad replace \ p_1 \ with \ c_1$
  - $IF \quad f(c_2) > f(p_2) \ THEN \quad replace \ p_2 \ with \ c_2$
- $ELSE$
  - $IF \quad f(c_2) > f(p_1) \ THEN \quad replace \ p_1 \ with \ c_2$
  - $IF \quad f(c_1) > f(p_2) \ THEN \quad replace \ p_2 \ with \ c_1$

где d (ζ1, ζ2) – расстояние Хемминга между двумя особами (ζ1, ζ2). 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.

Будем надеяться, что следующие 2 фигуры помогут вам понять почему.

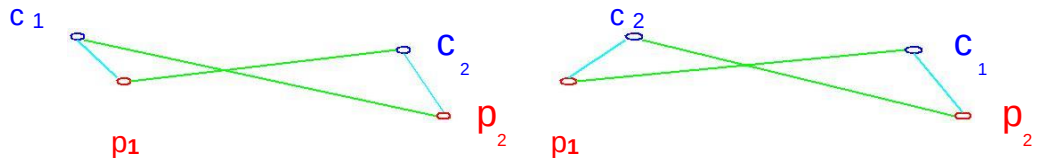

Рисунок 11: Два примера расстояний между родителями потомками.

10.3.1 Результаты, которые вы должны показать.

Надеюсь, вы примените оба алгоритма к двум функциям. Кроме привычного графика фитнесс/поколение, попробуйте визуализировать изменения, которые происходят из поколения в поколение.

11 Multi Objective Genetic Algorithm (MOGA) 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.

Таким образом, имея несколько целевых функций, мы должны ввести определение недоминируемых или Парето-оптимальных решений.

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

Предположим, мы имеем n целевых функций:

$$f_1(x); f_2(x); f_3(x); \ldots f_n(x)$$

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 это не так, мы скажем

“x недоминируем” или “Паретто-оптимален”

Простейший пример: допустим, у нас имеются две целевые функции

$$\begin{cases} f_1(x-2) = x^2 \\ f_2(x-4) = (x-2)^2 \end{cases}$$

- x=2 является оптимальным для f1, но не для f2

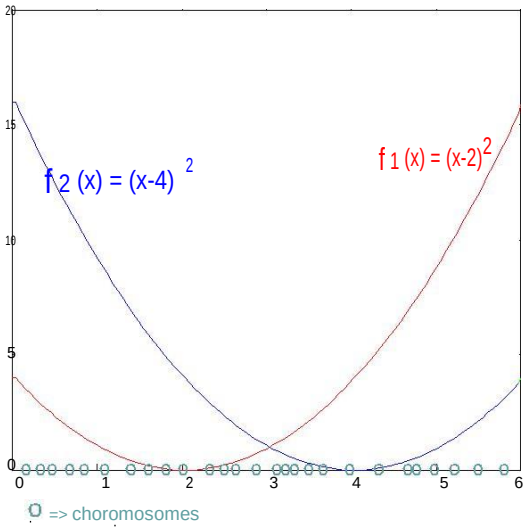

Рисунок 12: Две фитнесс-функции.

- x=4 является оптимальным для. f_2 но не для. f_1.

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

- Однако решение x=5 не является Парето-оптимальным, т.к. не лучше решения x = 3 для заданных целевых функций.

- Если мы построим график в пространстве решений f1-f2, то увеличение f1 на некотором интервале повлечет уменьшение f2, и, наоборот, что означает, что решения в этом интервале будут Паретто-оптимальными. В другом же интервале значений функции f1 увеличение f1 приведет к росту значения функции f2 (и наоборот). См. Рисунок 32. Пространство f1-f2 еще называется Пространством Компромиссного Решения (Trade-off Space).

Рассмотрим общий алгоритм применения многоцелевого генетического алгоритма.

Algorithm

1. Генерирование популяции.
2. Выбор особей в популяции.
3. Выполнение скрещивания и мутации для генерации 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[5]

[5] 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

6. Если потомок был принят, то пересчет рангов в популяции.
7. Повтор шагов 2-6 в соответствии с размером популяции.
8. Если желаемый результат не был достигнут, вернуться к шагу 2 и сформировать новую популяцию.

Задание 6 (Парето-оптимальное решение)
Опробуйте алгоритм со следующими целевыми функциями $y = (x\text{-}2)^2$ *и* $y = (x\text{-}4)^2$. *Покажите возможные Парето-оптимальные решения, которые вы нашли.*

11.1 Результаты, которые вы должны показать.

(1) Постройте таблицу содержащую (i) хромосому, (ii) с ее значениями x, (iii) *y*, (iv) и кол-во точек доминирующих над текущей точкой (ранг). Покажите три таких таблицы: в первом поколении, промежуточном и последнем. Смотрите таблицу, приведенную ниже. (2) Также постройте график кривой ранг/популяция.

No.	хромосома	x	y_1	y_2	ранг
#01	(0 1 1 0 1 0 ... 1)	1.32	2.62	4.58	13
	. . . . .				
	. . . . .				
#40	(1 0 1 0 0 1 ... 0)	7.86	4.13	9.13	5

Рисунок 13: Таблица, содержащую хромосому, с ее значениями x, *y*, и ранг.