Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Лабораторная работа №5. Многоагентные системы

Дисциплина: Интеллектуальные системы

Выполнил студент гр. 13541/3 _____ Д.В. Круминьш
(подпись)

Руководитель _____ Е.Н. Бендерская
(подпись)

”__”_____ 2017 г.

Санкт-Петербург
2017

Содержание

Лабораторная работа №5

Лабораторная работа №5

5.1 Этапы работы

Работа состоит из следующих этапов:

1. Общая теоретическая часть
2. Вариативная практическая часть с двумя типами заданий
3. Общая заключительная часть

5.1.1 Общая теоретическая часть

1. Дайте определение и краткую классификацию многоагентных интеллектуальных систем. В чем преимущества и в чем недостатки многоагентного подхода?
2. Приведите наиболее полную классификацию таких систем, кратко поясните по какому признаку дана эта классификация.
3. Какие задачи решаются с помощью многоагентного подхода. Приведите не менее ДВУХ(2) примеров задач, с кратким описанием (желательно сопроводить рисунками, диаграммами для наглядности)
4. Проанализируйте преимущества и недостатки многоагентного подхода на примере задач
5. Сформулируйте общие проблемы/недостатки и общие преимущества/достоинства такого подхода.

5.1.2 Практическая часть

ВАРИАНТ А

В среде Matlab v14.0+ (или любой другой по согласованию с руководителем) реализовать один из следующих алгоритмов с использованием многоагентного подхода:

1. Алгоритм имитации отжига (оптимизация функции)
2. Алгоритм оптимизации подражанием муравьиной колонии, англ. ant colony optimization, ACO)

3. Генетический алгоритм (оптимизация функции)
4. Метод роя частиц (оптимизация функции)
5. Муравьиный алгоритм поиска минимального пути (феромоны)
6. Алгоритм поиска минимума косяком рыб, стаи и т.д. (пример был на лекции)
7. Другие алгоритмы (по согласованию с преподавателем)

По результатам работы подготовить отчет в формате DOC, включающим листинг, скриншоты работы, а лучше видео-демонстрацию.

ВАРИАНТ Б

Проанализировать основные платформы для разработки многоагентных систем: выбрать одну из платформ (Например, NetLogo, StarLogo, Repast Simphony, Eclipse AMP, JADE, Jason либо другую по согласованию с руководителем) и провести обзор основных функциональных возможностей. Минимально должно быть описано:

1. Процесс установки ПО, ссылки на сайты ссылки на необходимы драйвера и т.д. (если нужно)
2. Процесс создания простого проекта
3. Анализ одного примера (ссылка на пример, описание алгоритма работы и процесса моделирования и т.д.)
4. При желании в качестве доп. баллов подготовить решение с использованием данной платформы одного индивидуального задания из ВАРИАНТА А, см. ранее.
5. Опишите основные возможности данного ПО применительно для создания многоагентных систем. Опишите замеченные недостатки или наоборот опишите достоинства данного ПО.

5.1.3 Общая заключительная часть

Написать выводы. В выводах отразить, помимо своих мыслей, возникших в ходе работы, ответы на приведенные ниже вопросы:

1. В чем Плюсы и минусы многоагентного подхода?
2. Какие еще среды и/или языки программирования использует для создания многоагентных систем?
3. Как по-вашему стоит ли развивать данное направление, если нет, то почему, если да, то в какую сторону?

4. Корректно ли по-вашему моделирование многоагентных систем на одной вычислительной машине (рассмотреть два варианта, итеративный перебор агентом в цикле, и создание для каждого агента своего процесса)

5. Приведите области/примеры в которых применение многоагентного подхода дает максимально положительные результаты.

5.2 Выполнение работы

5.2.1 Общая теоретическая часть

Дайте определение и краткую классификацию многоагентных интеллектуальных систем. В чем преимущества и в чем недостатки многоагентного подхода?
Мультиагентные системы — это системы, состоящие из автономных интеллектуальных агентов, взаимодействующих друг с другом и пассивной среды, в которой агенты существуют и на которую также могут влиять. В первую очередь речь идёт о программных системах или моделях, описывающих процесс их работы, их поведение.[1]

Важным основанием для классификации служит наличие (отсутствие) у агентов характеристик обучаемости или адаптивности. У обучаемых агентов поведение основано на предыдущем опыте. Еще одним основанием для классификации искусственных агентов служит принятие либо психологической, либо биологической метафоры при рассмотрении природы их действий.

Рис. 5.1: Классификация агентов

Интеллектуальные агенты обладают хорошо развитой и пополняемой символьной моделью внешнего мира, что достигается благодаря наличию у них базы знаний, механизмов решения и анализа действий. Небольшое различие между этими типами интеллектуальных агентов связано с расстановкой акцентов на тех или иных интеллектуальных функциях: либо на получении знаний о среде, либо на рассуждениях о возможных действиях.

Реактивные агенты не имеют ни сколько-нибудь развитого представления внешней среды, ни механизма многошаговых рассуждений, ни достаточного количества собственных ресурсов.

Отсюда вытекает еще одно существенное различие между интеллектуальными и реактивными агентами, связанное с возможностями прогнозирования изменений внешней среды и, как следствие, своего будущего.

Результаты сравнительного анализа реактивных и когнитивных агентов представлены в таблице 5.1.

Характеристики	**Когнитивные агенты**	**Реактивные агенты**
Внутренняя модель внешнего мира	Развитая	Примитивная
Рассуждения	Сложные и рефлексивные рассуждения	Простые одношаговые рассуждения
Мотивация	Развитая система мотивации, включающая убеждения, желания, намерения	Простейшие побуждения, связанные с выживанием
Память	Есть	Нет
Реакция	Медленная	Быстрая
Адаптивность	Малая	Высокая
Модульная архитектура	Есть	Нет
Состав многоагентной системы	Небольшое число автономных агентов	Большое число зависимых друг от друга агентов

Таблица 5.1: Сравнение агентов

Преимущества систем, построенных на основе многоагентного подхода:

1. распределение вычислительной нагрузки между множеством агентов;
2. гибкость и масштабируемость за счет децентрализованности;
3. повышение качества выполнения функций за счет поиска оптимальных вариантов при переговорах агентов;
4. применение знаний и вывода на знаниях.

Недостатком является невозможность описания алгоритма работы системы в целом, что рождает некоторую неопределенность.

Приведите наиболее полную классификацию таких систем, кратко поясните по какому признаку дана эта классификация.

Далее под агентом понимается аппаратная или программная сущность, способная действовать в интересах достижения целей, поставленных перед ним владельцем или пользователем.

Свойства агента (объекта) описываются исходной системой, а правила поведения – порождающей системой.

Характеристики	Типы агентов			
	Простые	**Смышленые (smart)**	**Интеллектуальные (intelligent)**	**Действительно интеллектуальные (truly)**
Автономное выполнение	✔	✘	✔	✔
Взаимодействие с другими агентами и/или пользователями	✔	✔	✔	✔
Слежение за окружением	✔	✔	✔	✔
Способность использования абстракций	✘	✔	✔	✔
Способность использования предметных знаний	✘	✔	✔	✘
Возможность адаптивного поведения для достижения целей	✘	✘	✔	✔
Обучение из окружения	✘	✘	✔	✔
Толерантность к ошибкам и/или неверным входным сигналам	✘	✘	✔	✘
Real-time исполнения	✘	✘	✔	✘
ЕЯ-взаимодействие	✘	✘	✔	✘

Таблица 5.2: Классификации агентов

Исходя из таблицы выше целесообразное поведение появляется лишь на уровне интеллектуальных агентов. Для него необходимо не только наличие целей функционирования, но и возможность использования достаточно сложных знаний о среде, партнерах и о себе.

Какие задачи решаются с помощью многоагентного подхода. Приведите не менее ДВУХ(2) примеров задач, с кратким описанием (желательно сопроводить рисунками, диаграммами для наглядности)

Многие известные онлайн-системы бронирования билетов используют многоагентновые си-

стемы для обработки информации, поступающей из других систем.

Также многоагентновые системы используются в компьютерных играх, для решения внутриигровых задач. Например задание логики действий бота(атака, передвижение, диалог и т.д.)

Проанализируйте преимущества и недостатки многоагентного подхода на примере задач. Сформулируйте общие проблемы/недостатки и общие преимущества/достоинства такого подхода.

В примере с бронированием билетов, с помощью многоагентного подхода решается задача когда между точкой отправление и точкой назначения неимеется прямых путей. Это сильно упрощает для пользователя планирование нужного маршрута. В данном случае минусов нет, так как в таком случае можно гораздо быстрее оценить временные и денежные затраты, нежели самостоятельно искать путь из пункта **A** в пункт **B**.

В примере с игрой, целесообразность такого подхода зависит от того, насколько функциональным необходимо сделать бота, разумеется чем больше функциональность тем больше затраты. Как минус боты будут все похожи друг на друга, для решения этой проблемы опять таки необходимо потратить больше времени.

В целом необходимость использование многоагентного подхода зависит от решаемой задачи, а также имеющихся средств.

5.2.2 Практическая часть

Вариант - A
Алгоритм - Генетический алгоритм (оптимизация функции)
Средство разработки - Python 3.6.3

Генетический алгоритм — алгоритм в основе которого лежит скрещивание (комбинирование). Путем перебора генов в конечном итоге получается правильная «комбинация».

В данной работе, алгоритм будет заниматься поиском наибольшего значения функции, а хромосомами будут выступать бить чисел.

То есть индивидом представляется натуральное число, которое передается в функцию, чем большей результат вернет функция, тем более правильная комбинация хромосом у индивида.

Для реализации подобного алгоритма, на языке **python** были написана программа, алгоритм которой:

1. Создание популяции, заполненной случайными значениями(в заданных пределах);
2. Отбор индивидов для скрещивания, в моем алгоритме для того чтобы индивид имел возможность скрещиваться, его результат полученный из функции должен быть лучше чем 25% индивидов его популяции;
3. Отобранные индивиды(натуральные числа) преобразуются в двоичный код(011011);

4. Двоичный код двух случайных индивидов, случайным образом скрещивается(часть бит от одного, остальная от другого);

5. С вероятностью в 10% происходят мутации(инверсия одного случайного бита);

6. Двоичное число преобразуется в натуральное число;

7. добавление полученного индивида в новую популяцию;

8. вышенаписанный цикл повторяется ровно столько раз, сколько задано размером популяции.

Возможности написанной программ:

1. Вывод в консоль количества бит индивида, количество возможных значений, информации(среднее значение x и f(x) о последнем поколении;

2. сохранение в файл графика оригинальной функции;

3. сохранение в файл графика зависимости среднего значения результата функции от поколения;

4. сохранение в файлы значения и результата каждого индивида, для каждого поколения;

5. сохранение в видео-файл, развития популяции от начального до конечного поколения.

```
1  #! /usr/bin/env python
2
3  from pylab import *
4  import numpy as np
5  import matplotlib.pyplot as plt
6  from matplotlib import animation, rc
7  from IPython.display import HTML, Image
8
9  rc('animation', html='html5')
10
11 # Functions
12 # Define the x function
13 def x_function(x):
14     #_function = ((sin(300)**2)/(1+(x/10-500)**2)*50)+(sin(x/10)
   ↪ *5)
15     _function = 8000*x - 0.25*x**2
16     return _function
17
18
```

```
19 # Convert decimal to a binary string
20 def den2bin(f):
21     bStr = ''
22     n = int(f)
23     if n < 0: raise
24     if n == 0: return '0'
25     while n > 0:
26         bStr = str(n % 2) + bStr
27         n = n >> 1
28     return bStr
29
30 #Convert decimal to a binary string of desired size of bits
31 def d2b(f, b):
32     n = int(f)
33     base = int(b)
34     ret = ""
35     for y in range(base-1, -1, -1):
36         ret += str((n >> y) & 1)
37     return ret
38
39
40 #Invert Chromosome
41 def invchr(string, position):
42     if int(string[position]) == 1:
43         string = string[:position] + '0' + string[position+1:]
44     else:
45         string = string[:position] + '1' + string[position+1:]
46     return string
47
48
49 #Random Wheel
50 def roulette(values, fitness):
51     n_rand = random()*fitness
52     sum_fit = 0
53     for i in range(len(values)):
54         sum_fit += values[i]
55         if sum_fit >= n_rand:
56             break
57     return i
58
```

```
59 # Genetic Algorithm Code to find the Maximum of F(X)
60
61
62 #Range of Values
63 x_max = 32000
64 x_min = 0
65
66 #GA Parameters
67 pop_size = 100
68 mutation_probability = 0.05
69 number_of_generations = 50
70
71 #Variables & Lists to be used during the code
72 gen_1_xvalues = []
73 gen_1_fvalues = []
74 generations_x = []
75 generations_f = []
76 generations_x_values = []
77 generations_f_values = []
78 avgFValues = []
79 avgXValues = []
80 fitness = 0
81
82
83
84 #Size of the string in bit
85 x_size = int(len(den2bin(x_max)))
86
87
88 print ("Maximum chromosome size of x is" + repr(x_size) + "bits,
      ↪ i.e.,"+ repr(pow(2,x_size)) + "variables.")
89
90
91 #first population - random values
92 for i in range(pop_size):
93     x_tmp = int(round(randint(x_max-x_min)+x_min))
94     gen_1_xvalues.append(x_tmp)
95
96     f_tmp = x_function(x_tmp)
97     gen_1_fvalues.append(f_tmp)
```

```
98
99      #Create total fitness
100     fitness += f_tmp
101 #print 'gen 1', gen_1_xvalues
102
103 #Getting maximum value for initial population
104 max_f_gen1 = 0
105 for i in range(pop_size):
106         if gen_1_fvalues[i] >= max_f_gen1:
107             max_f_gen1 = gen_1_fvalues[i]
108             max_x_gen1 = gen_1_xvalues[i]
109
110
111 #Starting GA loop
112
113 for i in range(number_of_generations):
114     #Reseting list for 2nd generation
115     gen_2_xvalues = []
116     gen_2_fvalues = []
117     selected = []
118
119     # Getting avg and min results
120     avgFValue=0
121     avgXValue=0
122     minXValue=9999999999
123     for j in range(pop_size):
124         avgFValue+=gen_1_fvalues[j]
125         avgXValue+=gen_1_xvalues[j]
126         if(gen_1_fvalues[j]<minXValue):
127             minXValue=gen_1_fvalues[j]
128     avgFValue=avgFValue/pop_size
129     avgXValue=avgXValue/pop_size
130     avgFValues.append(avgFValue)
131     avgXValues.append(avgXValue)
132
133     #Selecting individuals to reproduce, to repoduce u must have
    ↪ result value bigger at least than 25% of other individuals
134     k=0
135     while(k<pop_size):
136         ind_sel = roulette(gen_1_fvalues,fitness)
```

```
137             if(gen_1_fvalues[ind_sel]>((avgFValue+minXValue)/2)):
138                 selected.append(gen_1_xvalues[ind_sel])
139                 k+=1
140
141
142         #Crossing the selected members
143         for j in range(0, pop_size, 2):
144             sel_ind_A = d2b(selected[j],x_size)
145             sel_ind_B = d2b(selected[j+1],x_size)
146
147         #select point to cross over
148             cut_point = randint(1,x_size)
149
150         #new individual AB
151             ind_AB = sel_ind_A[:cut_point] + sel_ind_B[cut_point:]
152
153         #mutation AB
154             ran_mut = random()
155             if ran_mut < mutation_probability:
156                 gene_position = randint(0,x_size)
157                 ind_mut = invchr(ind_AB, gene_position)
158                 ind_AB = ind_mut
159
160         #new individual BA
161             ind_BA = sel_ind_B[:cut_point] + sel_ind_A[cut_point:]
162
163
164         #mutation BA
165             ran_mut = random()
166             if ran_mut < mutation_probability:
167                 gene_position = randint(0,x_size)
168                 ind_mut = invchr(ind_BA, gene_position)
169                 ind_BA = ind_mut
170
171         #Creating Generation 2
172             new_AB = int(ind_AB,2)
173             gen_2_xvalues.append(new_AB)
174
175             new_f_AB = x_function(new_AB)
176             gen_2_fvalues.append(new_f_AB)
```

```
177
178         new_BA = int(ind_BA,2)
179         gen_2_xvalues.append(new_BA)
180
181         new_f_BA = x_function(new_BA)
182         gen_2_fvalues.append(new_f_BA)
183     #print 'gen',i+2, gen_2_xvalues
184
185
186     #Getting maximum value
187     max_f_gen2 = 0
188     for j in range(pop_size):
189         if gen_2_fvalues[j] >= max_f_gen2:
190             max_f_gen2 = gen_2_fvalues[j]
191             max_x_gen2 = gen_2_xvalues[j]
192
193     #Transform gen2 into gen1
194     gen_1_xvalues = gen_2_xvalues
195     gen_1_fvalues = gen_2_fvalues
196     max_x_gen1 = max_x_gen2
197     max_f_gen1 = max_f_gen2
198     generations_x.append(max_x_gen2)
199     generations_f.append(max_f_gen2)
200
201     generations_x_values.append(gen_1_xvalues)
202     generations_f_values.append(gen_1_fvalues)
203
204     #Creating new fitness
205     fitness = 0
206     for j in range(pop_size):
207         f_tmp = x_function(gen_1_xvalues[j])
208         fitness += f_tmp
209
210
211 print ("AVG f(x) value of last generation: " + repr(avgFValues[
      ↪ number_of_generations-1]))
212 print ("AVG x value of last generation: " + repr(avgXValues[
      ↪ number_of_generations-1]))
213
214 #Ploting
```

```
215
216 #Ploting  Function
217 x = arange(x_min,x_max,0.01)
218 y = x_function(x)
219
220 #figure(1)
221 figure()
222 plot(x,y)
223 xlabel('x')
224 ylabel('F(x)')
225 title(r'$F(x)$')
226 savefig('orginalFunction.png')
227
228
229 #Ploting data for maximum values for each generation
230 figure()
231 plot(range(number_of_generations),avgFValues, 'ro')
232 xlabel('Generations')
233 ylabel('F(x) avg')
234 title(r'$F(x)$')
235 savefig('resultsOfGeneration.png')
236
237
238
239 def _update_plot(i, fig, scat):
240     preapred = []
241     for j in range(pop_size):
242         item=[]
243         item.append(int(round(generations_x_values[i][j])))
244         item.append(int(round(generations_f_values[i][j])))
245         preapred.append(item)
246     scat.set_offsets((preapred))
247     return scat,
248
249 fig =  plt.figure()
250
251 x = [0, 50]
252 y = [0, 0]
253
254 ax = fig.add_subplot(111)
```

```
255 ax.grid(True, linestyle = '-', color = '0.75')
256 ax.set_xlim([0, 30000])
257 ax.set_ylim([0, 70000000])
258
259 scat = plt.scatter(x, y, c = 'b')
260
261 anim = animation.FuncAnimation(fig, _update_plot, fargs = (fig,
        ↪ scat), frames = number_of_generations, interval = 200)
262 anim.save('animation.mp4')
263
264 for i in range(number_of_generations):
265     figure()
266     plt.plot(generations_x_values[i], generations_f_values[i], 'bo
        ↪ ')
267     plt.axis([0, 30000, 0, 70000000])
268     savefig('step'+repr(i)+'.png')
```

Листинг 5.1: script1.py

Запустим алгоритм со следующими параметрами:

Параметр	Описание	Значение
_function	Функция для нахождения максимума	$8000 * x - 0.25 * x^2$
x_max	максимальное значение индивида	32000
x_min	минимальное значение индивида	0
pop_size	размер популяции	100
mutation_probability	вероятность мутации	0.05
number_of_generations	количество поколений	50

П.С. Максимум функции достигается при x = 16000

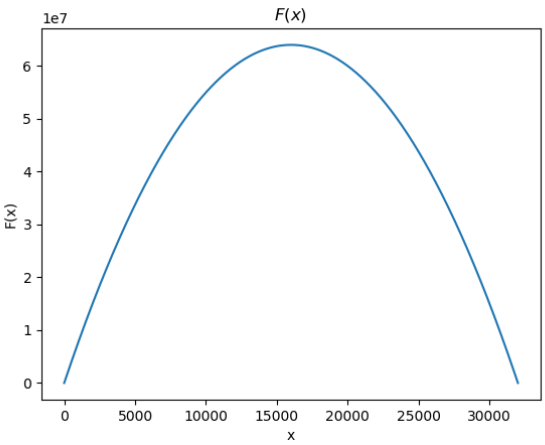

Рис. 5.2: Исходная функция

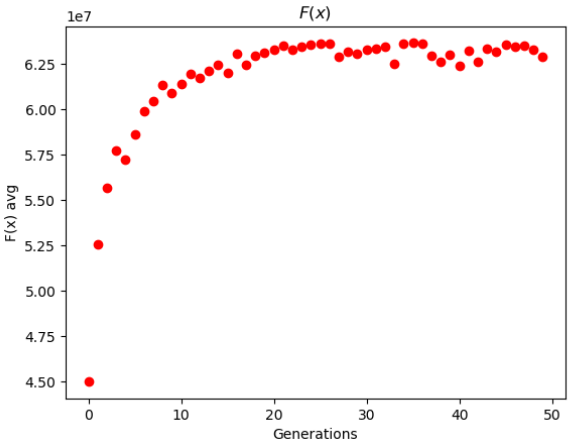

Рис. 5.3: Зависимость среднего значения от поколения

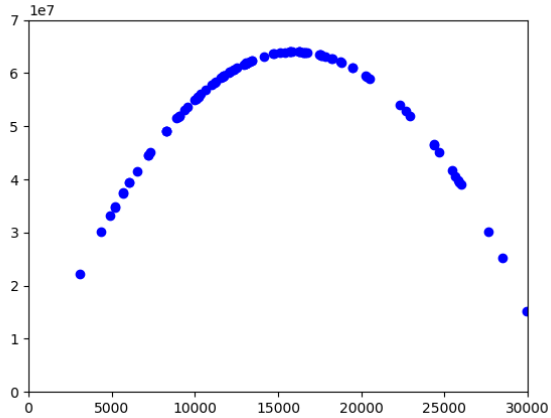

Рис. 5.4: 1 поколение

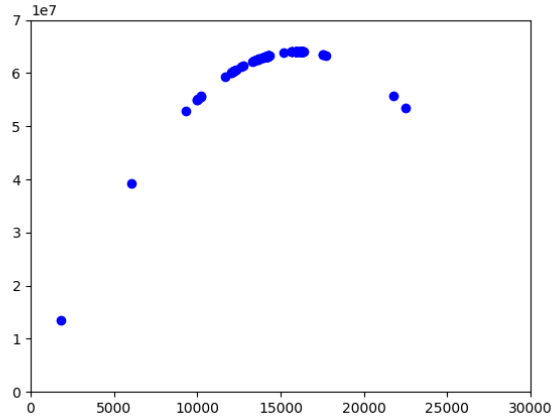

Рис. 5.5: 15 поколение

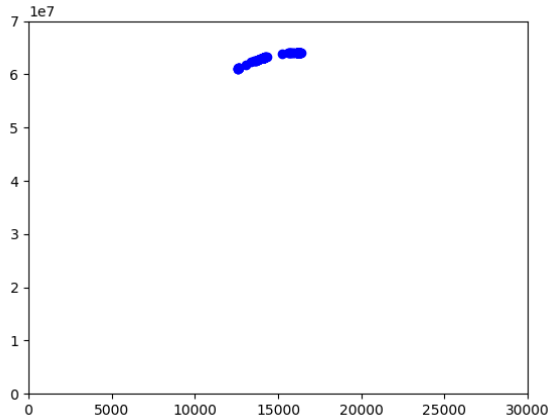

Рис. 5.6: 20 поколение

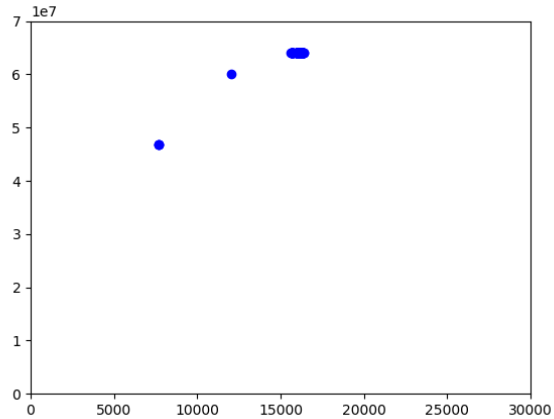

Рис. 5.7: 50 поколение

Из графика зависимости среднего значения популяция от поколения видно, что начиная

примерно с 20 поколения, существенных улучшений результата не возникало. Некоторое ухудшение результата происходило из-за случайных мутаций.

```
E:\study\s09ИС\\lab_05\script>python script.py
Maximum chromosome size of x is 15 bits, i.e., 32768 variables.
AVG f(x) value of last generation: 62591427.05
AVG x value of last generation: 16315.8
```

Листинг 5.2: Лог консоли

Изменим возможность случайных мутаций до 0%.

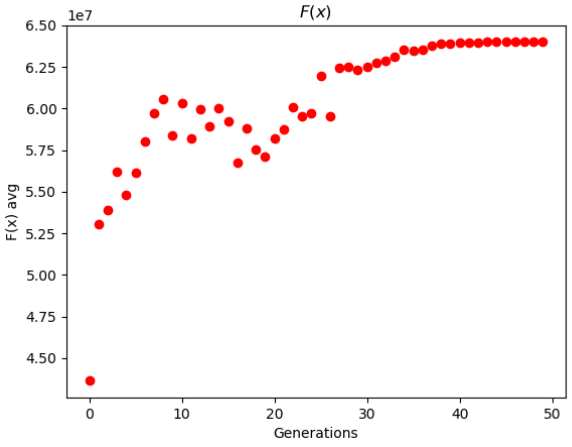

Рис. 5.8: Зависимость среднего значения от поколения

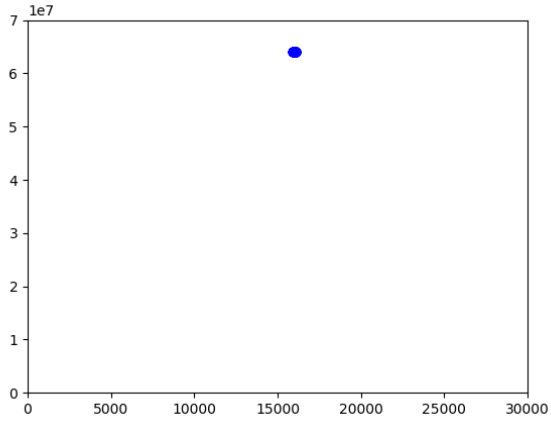

Рис. 5.9: 50 поколение

```
E:\study\s09ИС\\lab_05\script>python script1.py
Maximum chromosome size of x is 15 bits, i.e., 32768 variables.
AVG f(x) value of last generation: 63998377.0825
AVG x value of last generation: 16030.09
```

Листинг 5.3: Лог консоли

Отключив случайные мутации, былы получен более стабильный результат. Однако стоит отметить что среднее значение поколения колебалось примерно до 30 поколение, как вывод отсюда - мутации ускоряют нахождение необходимого результата, но после его нахождения имеется некоторая нестабильность.

Протестируем алгоритм для другой функции.

Параметр	Описание	Значение
_function	Функция для нахождения максимума	$((sin(300)^2)/(1+(x/10-500)^2)*50)+(sin(x/10)*5)$
x_max	максимальное значение индивида	5200

x_min	минимальное значение индивида	4800
pop_size	размер популяции	100
mutation_probability	вероятность мутации	0.05
number_of_generations	количество поколений	50

П.С. Максимум функции достигается при x = 4999.546

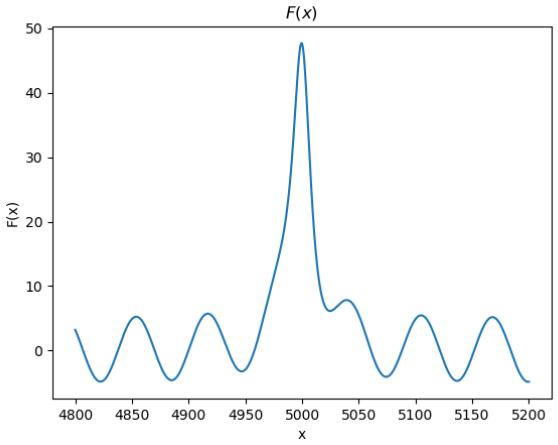

Рис. 5.10: Исходная функция

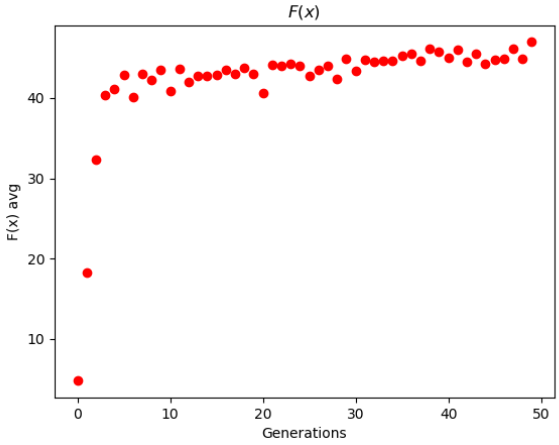

Рис. 5.11: Зависимость среднего значения от поколения

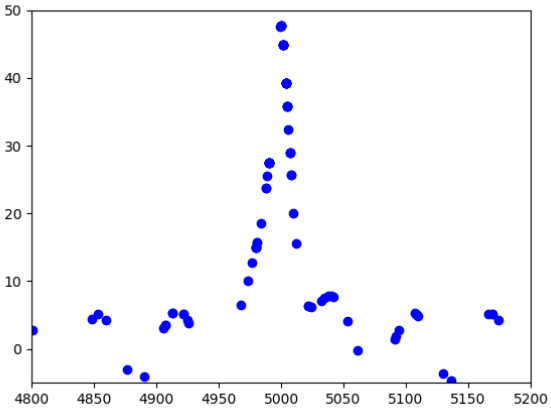

Рис. 5.12: 1 поколение

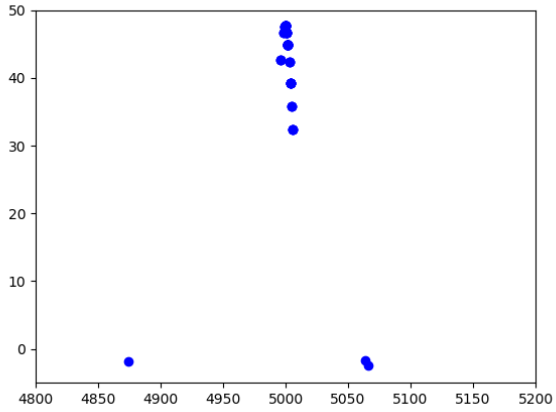

Рис. 5.13: 10 поколение

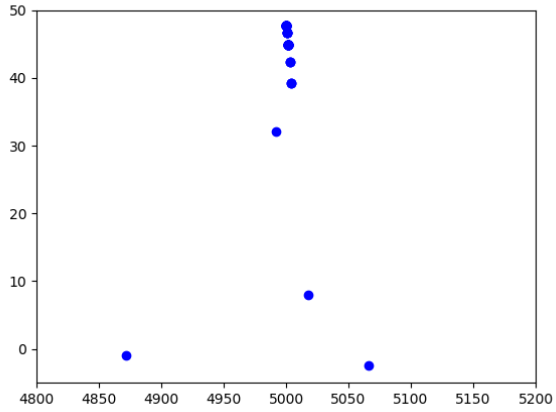

Рис. 5.14: 25 поколение

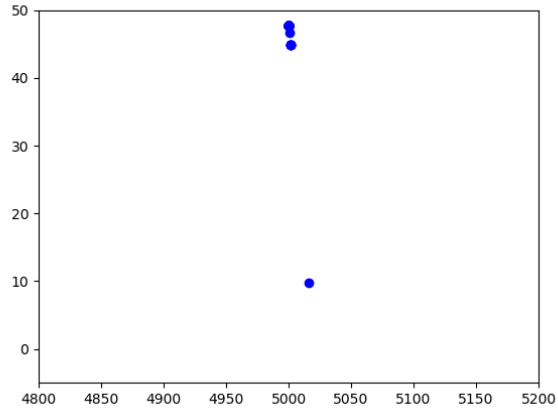

Рис. 5.15: 50 поколение

```
1 E:\study\s09ИС\\lab_05\script>python script2.py
2 Maximum chromosome size of x is 13 bits, i.e., 8192 variables.
3 AVG f(x) value of last generation: 46.943020176442943
4 AVG x value of last generation: 5000.25
```

Листинг 5.4: Лог консоли

В данном примере, максимальные изменения для популяции произошли в первых четырех поколениях, оставшиеся поколения она лишь ненамного улучшалась. Полученное максимальное значение в 5000.25, крайне близко к необходимому в 4999.546. По сути это оно и есть, так как работа происходила с целыми числами, без учета знаков после запятой

5.3 Вывод

Практическая часть работы оказалась довольно интересной, так как пронаблюдал как мои индивиды ”эволюционировали на глазах”, для поиска максимума функции. Это укрепило мои навыки программирование, а также познакомило с генетическим алгоритмом.

В чем Плюсы и минусы многоагентного подхода?
К положительным чертам отнести качество результата (при должной организации), простоту распараллеливание, гибкость работы. К минусам - для некоторых задач сложность организации процесса. Так как чем разностороннее процесс, тем больше разносторонних агентов, которые должны общаться между собой.

Какие еще среды и/или языки программирования использует для создания многоагентных систем? Сред достаточно много. Например:

- **Kiva(Amazon Robotics)** - распределенная система склада, при которой элементы хранения находятся на специальных модулях и перемещаются движущимися роботами. При

вводе заказа в базу данных системы, программа находит ближайший транспортный робот и направляет его к модулю хранения с помощью штрих-кодов нанесенных на полу склада.

- **CogniTAO** - платформа разработки автономных муль тиагентных систем, ориентированная на реальных роботов и виртуальных существ.
- **JADE** - широко используемая программная среда для создания мультиагентных си стем и приложений, поддерживающая FIPA-стандарты для интеллектуальных агентов.

Наиболее популярными языками программирования являются **C++** и **Java**.

Как по-вашему стоит ли развивать данное направление, если нет, то почему, если да, то в какую сторону?

Развивать данное направление разумеется стоит, а с приходом коммерческой составляющей развитие лишь ускорится. Например можно последовать примеру **Amazon Robotics** с оптимизацией работы склада.

Также на крупных мероприятиях уже начали использовать группы дронов, которые формируют определенные изображения в небе.

Возможно через **-дцать** лет в некоторых городах можно будет увидеть автоматизацию общественного транспорта, а потом и вовсе всего транспорта.

Корректно ли по-вашему моделирование многоагентных систем на одной вычислительной машине (рассмотреть два варианта, итеративный перебор агентом в цикле, и создание для каждого агента своего процесса)

Для простых задач вполне уместно, но для более сложных задач с сотнями, тысячими машин, потребуется уже целый кластер машин.

Приведите области/примеры в которых применение многоагентного подхода дает максимально положительные результаты.

- Amazon Robotics;
- интернет-порталы по продаже билетов;
- сложная и многофункциональная логистика.

Разумеется коммерческая область даст большие результаты чем научная область.

Литература

[1] И.Д.Зайцев. МНОГОАГЕНТНЫЕ СИСТЕМЫ В МОДЕЛИРОВАНИИ СОЦИАЛЬНО-ЭКОНОМИЧЕСКИХ ОТНОШЕНИЙ: ИССЛЕДОВАНИЕ ПОВЕДЕНИЯ И ВЕРИФИКАЦИЯ СВОЙСТВ С ПОМОЩЬЮ ЦЕПЕЙ МАРКОВА. — Новосибирск, 2014.

[2] Классификация агентов [Электронный ресурс]. — URL: http://www.aiportal.ru/articles/multiagent-systems/agent-classification.html (дата обращения: 2017-11-18).

[3] МНОГОАГЕНТНЫЕ СИСТЕМЫ И МНОГОАГЕНТНЫЙ ПОДХОД [Электронный ресурс]. — URL: http://lib.sale/informatsionnyiy-menedjment-knigi/mnogoagentnyie-sistemyi-mnogoagentnyiy-51882.html (дата обращения: 2017-11-18).

[4] ПРИМЕНЕНИЕ ТЕХНОЛОГИИ МНОГОАГЕНТНЫХ СИСТЕМ ДЛЯ ИНТЕЛЛЕКТУАЛЬНОЙ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЯ [Электронный ресурс]. — URL: http://systech.miem.edu.ru/2003/n1/Chekinov.htm (дата обращения: 2017-11-18).