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

5.1 Этапы работы . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

5.1.1 Общая теоретическая часть . . . . . . . . . . . . . . . . . . . . . . . . . 3 5.1.2 Практическая часть . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

5.1.3 Общая заключительная часть . . . . . . . . . . . . . . . . . . . . . . . . 4 5.2 Выполнение работы . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

5.2.1 Общая теоретическая часть . . . . . . . . . . . . . . . . . . . . . . . . . 5 5.2.2 Практическая часть . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

5.3 Вывод 20

Список литературы 21

# Лабораторная работа №5

## Этапы работы

Работа состоит из следующих этапов:

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

## 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.

## 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 мо- делирования и т.д.)
4. При желании в качестве доп. баллов подготовить решение с использованием данной плат- формы одного индивидуального задания из ВАРИАНТА А, см. ранее.
5. Опишите основные возможности данного ПО применительно для создания многоагент- ных систем. Опишите замеченные недостатки или наоборот опишите достоинства дан- ного ПО.

## 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 многоагент- ных 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 вычислитель- ной 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 макси- мально положительные результаты.

## Выполнение работы

* + 1. **Общая теоретическая часть**

### Дайте определение и краткую классификацию многоагентных интеллектуальных си- стем. В чем преимущества и в чем недостатки многоагентного подхода?

Мультиагентные системы — это системы, состоящие из автономных интеллектуальных аген- тов, 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 моде- лях, описывающих процесс их работы, их поведение.[1]

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



Рис. 5.1: Классификация агентов

**Интеллектуальные агенты** обладают хорошо развитой и пополняемой символьной моде- лью внешнего мира, что достигается благодаря наличию у них базы знаний, механизмов реше- ния и анализа действий. Небольшое различие между этими типами интеллектуальных агентов связано с расстановкой акцентов на тех или иных интеллектуальных функциях: либо на полу- чении 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 след- ствие, своего будущего.

Результаты сравнительного анализа реактивных и когнитивных агентов представлены в таб- лице 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 рож- дает 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 – порож- дающей 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 интеллек- туальных агентов. Для него необходимо не только наличие целей функционирования, но и воз- можность 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, диаграмма- ми 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 внут- риигровых задач. Например задание логики действий бота(атака, передвижение, диалог и т.д.) **Проанализируйте преимущества и недостатки многоагентного подхода на примере за- дач. Сформулируйте общие проблемы/недостатки и общие преимущества/достоинства та-**

### кого подхода.

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

В примере с игрой, целесообразность такого подхода зависит от того, насколько функциональ- ным 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 необ- ходимо потратить больше времени.

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

## Практическая часть

### Вариант - А

**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 воз- можность 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 популя- ции.

Возможности написанной программ:

1. Вывод в консоль количества бит индивида, количество возможных значений, информа- ции(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 поко- ления;
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 / b i n / env python*

2

1. **from** p y l a b **import** \*
2. **import** numpy as np
3. **import** m a t p l o t l i b . p y p l o t as p l t
4. **from** m a t p l o t l i b **import** a n i m a t i o n , r c
5. **from** I P y t h o n . d i s p l a y **import** HTML, Image 8

9 r c ( ’ a n i m a t i o n ’ , html = ’ html 5 ’ ) 10

1. *# Functions*
2. *# Define t h e x f u n c t i o n*
3. **def** x \_ f u n c t i o n ( x ) :

14 *# \_ f u n c t i o n = ( ( s i n ( 300 ) \*\*2) / ( 1 + ( x / 10* − *500 ) \*\*2) \*50 ) +( s i n ( x / 1 0 )*

*'*→ *\*5 )*

15 \_ f u n c t i o n = 8000\* x − 0 . 25 \* x\*\*2

16 **return** \_ f u n c t i o n 17

18

19 *# Convert decimal t o a b i nary s t r i n g*

|  |  |  |
| --- | --- | --- |
| 20 | **def** | den 2 b i n ( f ) : |
| 21 |  | b S t r = ’ ’ |
| 22 |  | n = **i n t** ( f ) |
| 23 |  | **i f** n < 0 : **r a i s e** |
| 24 |  | **i f** n == 0 : **return** ’ 0 ’ |
| 25 |  | **while** n > 0 : |
| 26 |  | b S t r = **s t r** ( n % 2 ) + b S t r |
| 27 |  | n = n >> 1 |
| 28 |  | **return** b S t r |
| 29 |  |  |

1. *# Convert decimal t o a b i nary s t r i n g o f d e s i r e d s i z e o f b i t s*
2. **def** d2b ( f , b ) :
3. n = **i n t** ( f )
4. ba se = **i n t** ( b )
5. r e t = ” ”
6. **f o r** y **in range** ( base −1 , −1 , −1) : 36 r e t += **s t r** ( ( n >> y ) & 1 )

37 **return** r e t 38

39

1. *# I n v e r t Chromosome*
2. **def** i n v c h r ( s t r i n g , p o s i t i o n ) :
3. **i f i n t** ( s t r i n g [ p o s i t i o n ] ) == 1 :
4. s t r i n g = s t r i n g [ : p o s i t i o n ] + ’ 0 ’ + s t r i n g [ p o s i t i o n + 1 : ]

### e l s e :

1. s t r i n g = s t r i n g [ : p o s i t i o n ] + ’ 1 ’ + s t r i n g [ p o s i t i o n + 1 : ]
2. **return** s t r i n g 47

48

1. *# Random Wheel*
2. **def** r o u l e t t e ( v a l u e s , f i t n e s s ) :
3. n\_ rand = random ( ) \* f i t n e s s
4. s u m \_ f i t = 0
5. **f o r** i **in range** ( **l e n** ( v a l u e s ) ) :
6. s u m \_ f i t += v a l u e s [ i ]
7. **i f** s u m \_ f i t >= n\_ rand :

### break

1. **return** i 58

|  |  |  |  |
| --- | --- | --- | --- |
| 59 | *# Genetic Algorithm Code t o f i n d t h e Maximum o f F( X)* |  | |
| 60 |  |
| 61 |  |
| 62 | *# Range o f Values* |
| 63 | x\_max = 32000 |
| 64 | x\_min = 0 |
| 65 |  |
| 66 | *#GA Parameters* |
| 67 | p o p \_ s i z e = 100 |
| 68 | m u t a t i o n \_ p r o b a b i l i t y = 0 . 0 5 |
| 69 | n u m b e r \_ o f \_ g e n e r a t i o n s = 50 |
| 70 |  |
| 71 | *# V a r i a b l e s & L i s t s t o be used during t he code* |
| 72 | g e n \_ 1 \_ x v a l u e s = [ ] |
| 73 | g e n \_ 1 \_ f v a l u e s = [ ] |
| 74 | g e n e r a t i o n s \_ x = [ ] |
| 75 | g e n e r a t i o n s \_ f = [ ] |
| 76 | g e n e r a t i o n s \_ x \_ v a l u e s = [ ] |
| 77 | g e n e r a t i o n s \_ f \_ v a l u e s = [ ] |
| 78 | avg FValues = [ ] |
| 79 | avg XValues = [ ] |
| 80 | f i t n e s s = 0 |
| 81 |  |
| 82 |  |
| 83 |  |
| 84 | *# S i z e o f t h e s t r i n g i n b i t* |
| 85 | x \_ s i z e = **i n t** ( **l e n** ( den 2 b i n ( x\_max ) ) ) |
| 86 |  |
| 87 |  |
| 88 | **p r i n t** ( ”Maximum chromosome s i z e of x i s ” + **repr** ( x \_ s i z e ) | + | ” b i t s , |
| 89 | *'*→ i . e . , ”+ **repr** ( **pow** ( 2 , x \_ s i z e ) ) + ” v a r i a b l e s . ” ) |  |  |
| 90 |  |  |  |
| 91 | *# f i r s t p o p u l a t i o n* − *random v a l u e s* |  |  |
| 92 | **f o r** i **in range** ( p o p \_ s i z e ) : |  |  |
| 93 | x\_tmp = **i n t** ( **round** ( r a n d i n t ( x\_max−x\_min ) + x\_min ) ) |  |  |
| 94 | g e n \_ 1 \_ x v a l u e s . append ( x\_tmp ) |  |  |
| 95 |  |  |  |
| 96 | f\_ tmp = x \_ f u n c t i o n ( x\_tmp ) |  |  |
| 97 | g e n \_ 1 \_ f v a l u e s . append ( f\_ tmp ) |  |  |

98

* 1. *# Create t o t a l f i t n e s s*
  2. f i t n e s s += f\_ tmp
  3. *# p r i n t ’ gen 1 ’ , gen\_ 1 \_ xvalues*

102

1. *# G e t t i n g maximum value f o r i n i t i a l p o p u l a t i o n*
2. max\_ f\_ gen 1 = 0
3. **f o r** i **in range** ( p o p \_ s i z e ) :
4. **i f** g e n \_ 1 \_ f v a l u e s [ i ] >= max\_ f\_ gen 1 :
5. max\_ f\_ gen 1 = g e n \_ 1 \_ f v a l u e s [ i ]
6. max\_x\_gen1 = g e n \_ 1 \_ x v a l u e s [ i ] 109

110

111 *# S t a r t i n g GA l oop*

112

1. **f o r** i **in range** ( n u m b e r \_ o f \_ g e n e r a t i o n s ) :
2. *# R e s e t i n g l i s t f o r 2 nd g e n e r a t i o n*
3. g e n \_ 2 \_ x v a l u e s = [ ]
4. g e n \_ 2 \_ f v a l u e s = [ ]
5. s e l e c t e d = [ ] 118
6. *# G e t t i n g avg and min r e s u l t s*
7. avg FValue =0
8. avgXValue =0

122 minXValue =9999999999

1. **f o r** j **in range** ( p o p \_ s i z e ) :
2. avg FValue += g e n \_ 1 \_ f v a l u e s [ j ]
3. avgXValue+= g e n \_ 1 \_ x v a l u e s [ j ]
4. **i f** ( g e n \_ 1 \_ f v a l u e s [ j ] < minXValue ) :
5. minXValue= g e n \_ 1 \_ f v a l u e s [ j ]
6. avg FValue= avg FValue / p o p \_ s i z e
7. avgXValue= avgXValue / p o p \_ s i z e
8. avg FValues . append ( avg FValue )
9. avg XValues . append ( avgXValue ) 132

133 *# S e l e c t i n g i n d i v i d u a l s t o reproduce , t o repoduce u must have*

*'*→ *r e s u l t value b i gger a t l e a s t t han 25% o f o t h e r i n d i v i d u a l s*

134 k=0

1. **while** ( k< p o p \_ s i z e ) :
2. i n d \_ s e l = r o u l e t t e ( g e n \_ 1 \_ f v a l u e s , f i t n e s s )
3. **i f** ( g e n \_ 1 \_ f v a l u e s [ i n d \_ s e l ] > ( ( avg FValue+ minXValue ) / 2 ) ) :
4. s e l e c t e d . append ( g e n \_ 1 \_ x v a l u e s [ i n d \_ s e l ] )

139 k+=1

140

141

1. *# Crossing t h e s e l e c t e d members*
2. **f o r** j **in range** ( 0 , po p\_ s i ze , 2 ) :
3. s e l \_ i n d \_ A = d2b ( s e l e c t e d [ j ] , x \_ s i z e )
4. s e l \_ i n d \_ B = d2b ( s e l e c t e d [ j + 1 ] , x \_ s i z e ) 146
5. *# s e l e c t p o i n t t o c r o s s over*
6. c u t \_ p o i n t = r a n d i n t ( 1 , x \_ s i z e ) 149
7. *# new i n d i v i d u a l AB*
8. ind\_AB = s e l \_ i n d \_ A [ : c u t \_ p o i n t ] + s e l \_ i n d \_ B [ c u t \_ p o i n t : ] 152
9. *# mutation AB*
10. ran\_ mut = random ( )
11. **i f** ran\_ mut < m u t a t i o n \_ p r o b a b i l i t y :
12. g e n e \_ p o s i t i o n = r a n d i n t ( 0 , x \_ s i z e )
13. ind\_ mut = i n v c h r ( ind\_AB , g e n e \_ p o s i t i o n )
14. ind\_AB = ind\_ mut 159
15. *# new i n d i v i d u a l BA*
16. ind\_BA = s e l \_ i n d \_ B [ : c u t \_ p o i n t ] + s e l \_ i n d \_ A [ c u t \_ p o i n t : ] 162

163

1. *# mutation BA*
2. ran\_ mut = random ( )
3. **i f** ran\_ mut < m u t a t i o n \_ p r o b a b i l i t y :
4. g e n e \_ p o s i t i o n = r a n d i n t ( 0 , x \_ s i z e )
5. ind\_ mut = i n v c h r ( ind\_BA , g e n e \_ p o s i t i o n )
6. ind\_BA = ind\_ mut 170
7. *# Creating Generation 2*
8. new\_AB = **i n t** ( ind\_AB , 2 )
9. g e n \_ 2 \_ x v a l u e s . append ( new\_AB ) 174
10. new\_f\_AB = x \_ f u n c t i o n ( new\_AB )
11. g e n \_ 2 \_ f v a l u e s . append ( new\_f\_AB )

177

1. new\_BA = **i n t** ( ind\_BA , 2 )
2. g e n \_ 2 \_ x v a l u e s . append ( new\_BA ) 180
3. new\_f\_BA = x \_ f u n c t i o n ( new\_BA )
4. g e n \_ 2 \_ f v a l u e s . append ( new\_f\_BA )
5. *# p r i n t ’ gen ’ , i +2 , gen\_ 2 \_ xvalues*

184

185

1. *# G e t t i n g maximum value*
2. max\_ f\_ gen 2 = 0
3. **f o r** j **in range** ( p o p \_ s i z e ) :
4. **i f** g e n \_ 2 \_ f v a l u e s [ j ] >= max\_ f\_ gen 2 :
5. max\_ f\_ gen 2 = g e n \_ 2 \_ f v a l u e s [ j ]
6. max\_x\_gen2 = g e n \_ 2 \_ x v a l u e s [ j ] 192
7. *# Transform gen 2 i n t o gen 1*
8. g e n \_ 1 \_ x v a l u e s = g e n \_ 2 \_ x v a l u e s
9. g e n \_ 1 \_ f v a l u e s = g e n \_ 2 \_ f v a l u e s
10. max\_x\_gen1 = max\_x\_gen2
11. max\_ f\_ gen 1 = max\_ f\_ gen 2
12. g e n e r a t i o n s \_ x . append ( max\_x\_gen2 )
13. g e n e r a t i o n s \_ f . append ( max\_ f\_ gen 2 ) 200
14. g e n e r a t i o n s \_ x \_ v a l u e s . append ( g e n \_ 1 \_ x v a l u e s )
15. g e n e r a t i o n s \_ f \_ v a l u e s . append ( g e n \_ 1 \_ f v a l u e s ) 203
16. *# Creating new f i t n e s s*
17. f i t n e s s = 0
18. **f o r** j **in range** ( p o p \_ s i z e ) :
19. f\_ tmp = x \_ f u n c t i o n ( g e n \_ 1 \_ x v a l u e s [ j ] )
20. f i t n e s s += f\_ tmp 209

210

1. **p r i n t** ( ”AVG f ( x ) v a l u e of l a s t g e n e r a t i o n : ” + **repr** ( avg FValues [

*'*→ n u m b e r \_ o f \_ g e n e r a t i o n s − 1 ]) )

1. **p r i n t** ( ”AVG x v a l u e of l a s t g e n e r a t i o n : ” + **repr** ( avg XValues [

*'*→ n u m b e r \_ o f \_ g e n e r a t i o n s − 1 ]) )

213

214 *# P l o t i n g*

215

1. *# P l o t i n g Function*
2. x = a r a n g e ( x\_min , x\_max , 0 . 0 1 )
3. y = x \_ f u n c t i o n ( x ) 219
4. *# f i g u r e ( 1 )*
5. f i g u r e ( )
6. p l o t ( x , y )
7. x l a b e l ( ’ x ’ )
8. y l a b e l ( ’ F ( x ) ’ )
9. t i t l e ( r ’ $F ( x ) $ ’ )
10. s a v e f i g ( ’ o r g i n a l F u n c t i o n . png ’ ) 227

228

1. *# P l o t i n g data f o r maximum v a l u e s f o r each g e n e r a t i o n*
2. f i g u r e ( )
3. p l o t ( **range** ( n u m b e r \_ o f \_ g e n e r a t i o n s ) , avg FValues , ’ ro ’ )
4. x l a b e l ( ’ G e n e r a t i o n s ’ )
5. y l a b e l ( ’ F ( x ) avg ’ )
6. t i t l e ( r ’ $F ( x ) $ ’ )
7. s a v e f i g ( ’ r e s u l t s O f G e n e r a t i o n . png ’ ) 236

237

238

1. **def** \_ u p d a t e \_ p l o t ( i , f i g , s c a t ) :
2. p r e a p r e d = [ ]
3. **f o r** j **in range** ( p o p \_ s i z e ) :
4. i t em = [ ]
5. i t em . append ( **i n t** ( **round** ( g e n e r a t i o n s \_ x \_ v a l u e s [ i ] [ j ] ) ) )
6. i t em . append ( **i n t** ( **round** ( g e n e r a t i o n s \_ f \_ v a l u e s [ i ] [ j ] ) ) )
7. p r e a p r e d . append ( i t e m )
8. s c a t . s e t \_ o f f s e t s ( ( p r e a p r e d ) )
9. **return** s c a t , 248

249 f i g = p l t . f i g u r e ( ) 250

251 x = [ 0 , 50 ]

252 y = [ 0 , 0 ]

253

254 ax = f i g . a d d \_ s u b p l o t ( 1 1 1 )

255

ax . g r i d ( True , l i n e s t y l e = ’−’ , c o l o r = ’ 0 . 7 5 ’ )

ax . s e t \_ x l i m ( [ 0 , 3 0 0 0 0 ] )

ax . s e t \_ y l i m ( [ 0 , 70000000 ] )

s c a t = p l t . s c a t t e r ( x , y , c = ’ b ’ )

anim = a n i m a t i o n . Func Animation ( f i g , \_ u p d a t e \_ p l o t , f a r g s = ( f i g ,

*'*→ s c a t ) , f r a m e s = n u m b e r \_ o f \_ g e n e r a t i o n s , i n t e r v a l = 200 ) anim . s av e ( ’ a n i m a t i o n . mp4 ’ )

**f o r** i **in range** ( n u m b e r \_ o f \_ g e n e r a t i o n s ) :

f i g u r e ( )

p l t . p l o t ( g e n e r a t i o n s \_ x \_ v a l u e s [ i ] , g e n e r a t i o n s \_ f \_ v a l u e s [ i ] , ’ bo

*'*→ ’ )

p l t . a x i s ( [ 0 , 30000 , 0 , 70000000 ] )

s a v e f i g ( ’ s t e p ’+ **repr** ( i ) + ’ . png ’ )

Листинг 5.1: script1.py

256

257

258

259

260

261

262

263

264

265

266

267

268

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

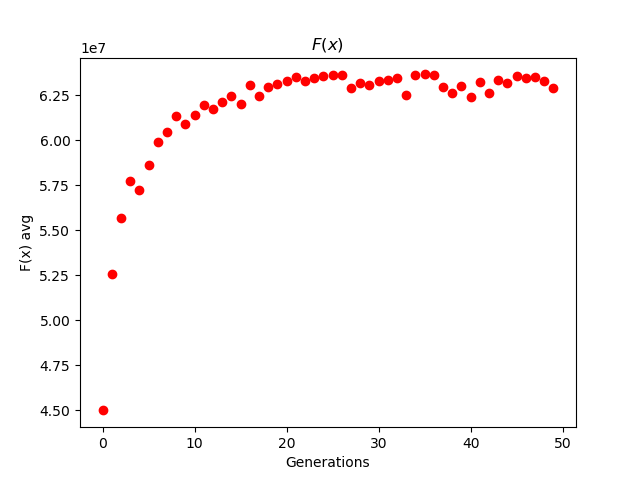
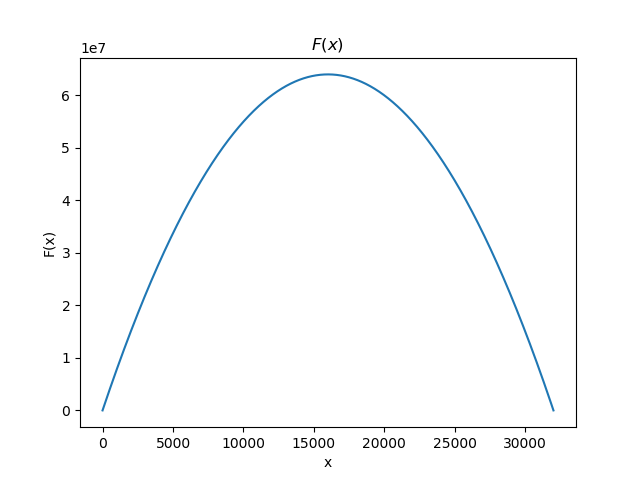


Рис. 5.2: Исходная функция

Рис. 5.3: Зависимость среднего значения от по- коления

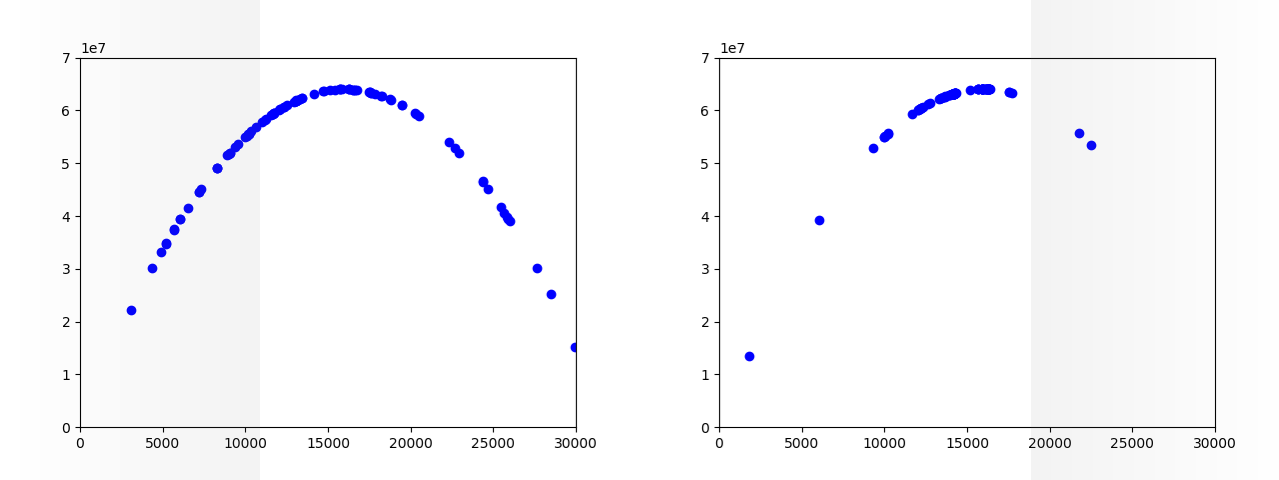


Рис. 5.4: 1 поколение Рис. 5.5: 15 поколение

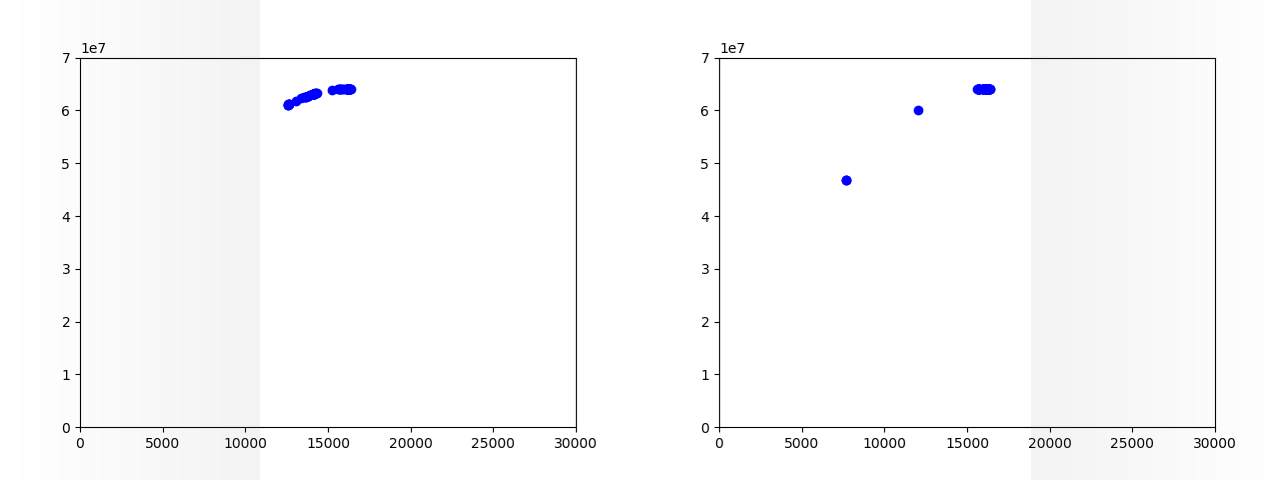


Рис. 5.6: 20 поколение Рис. 5.7: 50 поколение

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

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

E : \ s t u d y \ s 09 ИС \ \ l a b \_ 0 5 \ s c r i p t > python s c r i p t . py

Maximum chromosome s i z e of x i s 1 5 b i t s , i . e . , 3 2 7 6 8 v a r i a b l e s . AVG f ( x ) v a l u e of l a s t g e n e r a t i o n : 62591427 . 05

AVG x v a l u e of l a s t g e n e r a t i o n : 16315 . 8

Листинг 5.2: Лог консоли

1

2

3

4

Изменим возможность случайных мутаций до 0%.

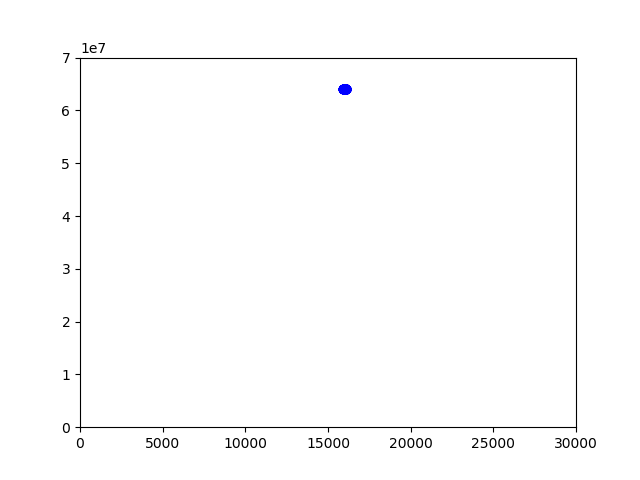
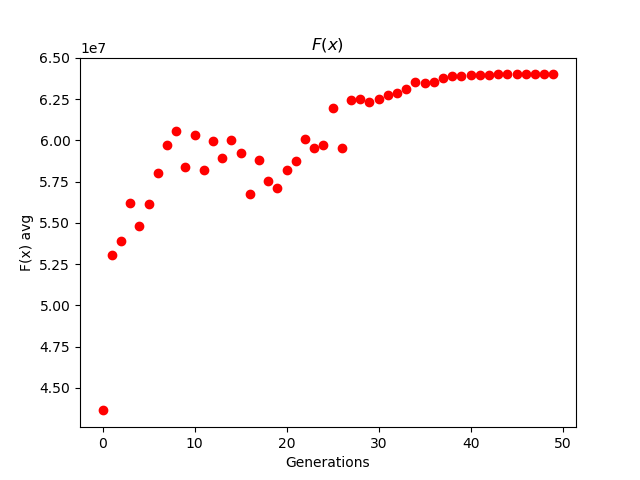


Рис. 5.8: Зависимость среднего значения от по- коления

Рис. 5.9: 50 поколение

1

E : \ s t u d y \ s 09 ИС \ \ l a b \_ 0 5 \ s c r i p t > python s c r i p t 1 . py

Maximum chromosome s i z e of x i s 1 5 b i t s , i . e . , 3 2 7 6 8 v a r i a b l e s . AVG f ( x ) v a l u e of l a s t g e n e r a t i o n : 63998377 . 0825

AVG x v a l u e of l a s t g e n e r a t i o n : 16030 . 09

Листинг 5.3: Лог консоли

2

3

4

Отключив случайные мутации, былы получен более стабильный результат. Однако стоит отме- тить 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

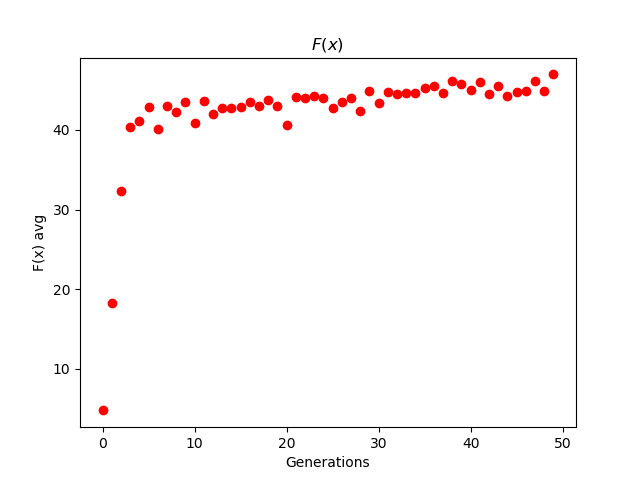
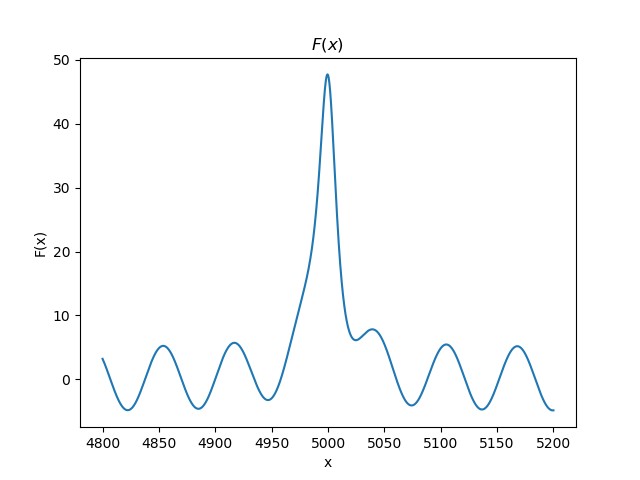


Рис. 5.10: Исходная функция

Рис. 5.11: Зависимость среднего значения от по- коления

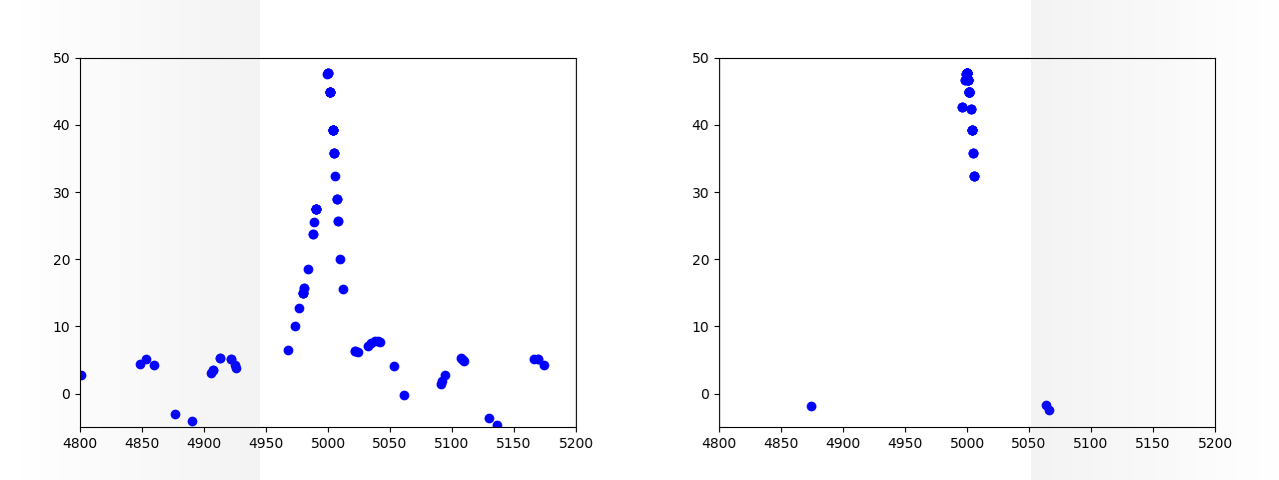


Рис. 5.12: 1 поколение Рис. 5.13: 10 поколение

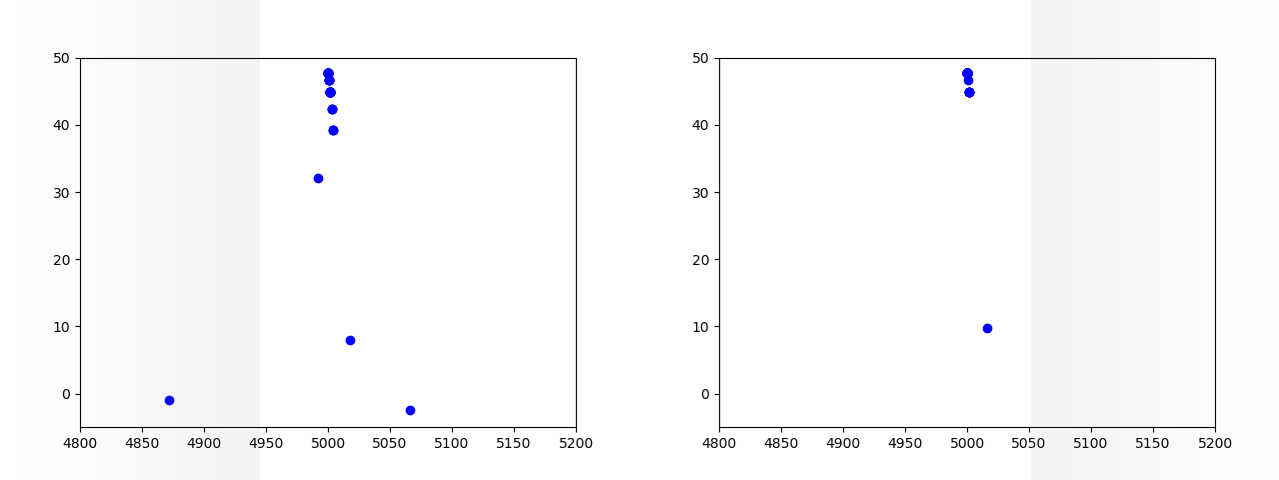


Рис. 5.14: 25 поколение Рис. 5.15: 50 поколение

1

E : \ s t u d y \ s 09 ИС \ \ l a b \_ 0 5 \ s c r i p t > python s c r i p t 2 . py

Maximum chromosome s i z e of x i s 1 3 b i t s , i . e . , 8 1 9 2 v a r i a b l e s . AVG f ( x ) v a l u e of l a s t g e n e r a t i o n : 46 . 943020176442943

AVG x v a l u e of l a s t g e n e r a t i o n : 5000 . 25

Листинг 5.4: Лог консоли

2

3

4

В данном примере, максимальные изменения для популяции произошли в первых четырех по- колениях, 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

## 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 многоагент- ных систем?** Сред достаточно много. Например:

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

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

* **CogniTAO** - платформа разработки автономных муль тиагентных систем, ориентирован- ная 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 оптими- зацией работы склада.

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

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

### Корректно ли по-вашему моделирование многоагентных систем на одной вычислитель- ной машине (рассмотреть два варианта, итеративный перебор агентом в цикле, и созда- ние 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 мак- симально 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 ВЕРИФИКА- ЦИЯ 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 ИНТЕЛЛЕКТУ- АЛЬНОЙ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЯ [Электронный ресурс]. – URL:

<http://systech.miem.edu.ru/2003/n1/Chekinov.htm> (дата обращения: 2017-11-18).