

Лабораторный практикум по дисциплине «Теория принятия решений»

Крутиков А.К., Ростовцев В.С.

2025 г.

Лабораторная работа №1

Разработка фрагмента экспертной системы с помощью оболочки ANIES

К курсу лабораторных работ подготовлен список источников дополнительной литературы [1-16].

Цель: освоение технологии и методики построения экспертных систем на примере разработки учебной экспертной системы. Студент выступает в роли одновременно эксперта и инженера по знаниям.

Работа содержит следующие этапы:

- Знакомство с инструментальным программным обеспечением (ANIES) для построения экспертных систем. В инструментальную систему входят помимо описанной выше экспертной оболочки программа-редактор баз знаний и программа логического вывода.
- Выбор задачи и предметной области для реализации учебной экспертной системы. Согласование с преподавателем задание на разработку учебной экспертной системы
- Реализация 1-ой версии базы знаний. Консультации у преподавателя о пути развития базы знаний.
- Реализация и тестирование базы знаний. Отладка экспертной системы. Тестирование базы знаний учебной экспертной системы;
- Демонстрация работы учебной экспертной системы преподавателю.
- Подготовка отчета по лабораторной работе в электронном виде и защита выполненной работы.

В отчет включить титульный лист, цель выполнения лабораторной работы, задание, текст программы на языке ANIES, дерево логического вывода, расчет коэффициентов уверенности по одной из веток и их сравнение с полученными коэффициентами при выполнении программы. Анализ методов логического вывода (прямой, обратный) по критерию времени выполнения. В разделе выводы указать достоинства и недостатки системы ANIES.

Составить в текстовом редакторе описание учебной экспертной системы (файл с расширением *.ies). Пример приведен в приложении А. Количество гипотез – не менее 7, количество параметров – не менее 7, число переменных – не менее 2, количество правил определяется студентом из расчета количества используемых ключевых слов IF (не менее 20). Рекомендуется составить не менее 3-х правил.

- Запустить инструментальную систему ANIES в различных режимах логического вывода(прямой и обратный в глубину и в ширину) .
- Демонстрационный прототип экспертной системы предъявить преподавателю.
- Составить в текстовом редакторе Microsoft Word XP (и выше) отчет по лабораторной работе, в которой привести следующие данные:
 - Описание задания
 - Файл с расширением *.ies
 - Дерево логического вывода
 - Расчет коэффициентов уверенности для вершин дерева логического вывода и их проверка с помощью системы ANIES
 - Выводы по результатам выполнения лабораторной работы (результаты работы в различных режимах логического вывода, достоинства и недостатки системы ANIES).

Инструментальная экспертная система «ANIES» является обучающей программой, предназначенной для демонстрации возможностей, которые предоставляют продукционные правила при логическом выводе.

Для работы программы необходим процессор Intel Core 5 и выше, русифицированная версия Windows 7 или выше. Программное обеспечение включает в себя выполняемый файл ANIES.EXE.

В процессе работы программы образуются файлы баз знаний *.ies, хранящие ЭС пользователя. Все файлы хранятся в текстовом формате.

Взаимодействие пользователя с инструментальной экспертной системой осуществляется посредством интерфейса пользователя. Одним из основных управляющих элементов интерфейса является главное меню программы, которое состоит из горизонтального меню, содержащего имена основных групп команд, и выпадающих подменю, позволяющих выбрать конкретную команду или режим работы. Такие пункты горизонтального меню, как “Файл”, “Правка”, являются стандартными для программ. Они содержат набор команд для работы с файловой системой, облегчения редактирования

текста. При помощи текстового редактора либо используя режим вставки при помощи пункта меню “Ввод данных” и панели ключевых слов, специалист по ИИ создает структуру БЗ, с использованием продукционных правил “IF-THEN-ELSE”, которая в последствии будет участвовать в обработке данных в режиме интерпретации.

После запуска ЭС пользователь вводит ответы на запрашиваемые системой вопросы с указанием коэффициента уверенности в диапазоне $[-1;1]$. Отвечая, на один вопрос пользователь может указать несколько ответов или ни одного. Отсутствие ответа интерпретируется как коэффициент равный нулю. Также возможно остановить процесс обучения в любой момент. Система, используя машину логического вывода, производит подсчет всех заключений и отображает перечень гипотез. При желании пользователь может просмотреть ход срабатывания правил экспертной системы в виде протокола решения.

Некоторые пункты меню продублированы кнопками быстрого управления. Контекстно-зависимую помощь по многим элементам интерфейса программы можно получить, подведя курсор мыши к соответствующему элементу.

Разработчику экспертной системы предлагается использовать панели инструментов: гипотез, параметров, ключевых слов при составлении правил. Что позволяет значительно сократить количество ошибок. Также при написании правил можно использовать обычный режим - режим редактирования.

Структура базы знаний предусматривает использование продукционных правил “IF-THEN-ELSE”. В приведенном ниже примере Правило1 может быть реализовано с помощью правил “IF-THEN-ELSE” .

NAME Правило1

IF на_занятиях неусидчив

THEN темперамент холерик [0,5]

ELSE

IF на_занятиях энергичен

THEN темперамент сангвиник [0,5]

ELSE

IF на_занятиях спокоен

THEN темперамент флегматик [0,5]

ELSE темперамент меланхолик [0,5]

END

1. Все правила “IF-THEN-ELSE” и ”CASE” должны заканчиваться ключевым словом “END”.
2. Все правила “IF-THEN-ELSE” должны обязательно содержать ключевое слово ELSE.
3. При написании гипотез, параметров, наименований правил вместо пробелов необходимо использовать знаки подчёркивания, с целью распознавания окончания. Например, на_ занятиях спокоен.
4. При составлении циклов “IF-THEN-ELSE” необходимо проставлять все ключевые слова: IF, THEN, ELSE, даже в случае отсутствия последующих операторов.

Работа с программой состоит из двух этапов: создание, редактирование файла ЭС пользователя и режим выполнения работы ЭС.

Для начала работы необходимо создать новый проект с помощью команды системы меню **Файл | Новый проект**, затем создать новый файл базы знаний (БЗ) пользователя или загрузить ранее сохраненный файл БЗ с помощью команды системы меню **Файл | Открыть существующую базу знаний (БЗ)**. Пункты меню **Файл | Сохранить** и **Файл | Сохранить как** предназначены для сохранения БЗ пользователя.

При создании проекта пользователь может разбить всю программу на несколько разделов. Например, раздел ДЕРЕВЬЯ и раздел ЦВЕТЫ, приведенные ниже в примере, сформированы при создании проекта. Если требуется разбить на разделы после создания проекта, то можно использовать команду системы меню **Файл | Коррекция данных проекта** или выполнить редактирование текста программы.

Слияние нескольких ранее созданных баз знаний с помощью команды системы меню **Файл | Объединение файлов в проект**. Файлы с других компьютеров должны быть переписаны в текущий каталог системы.

Редактор базы знаний представляет собой стандартный текстовый редактор. Он позволяет создавать БЗ, которая в последствии будет участвовать в обработке данных интерпретатором. Редактор предусматривает несколько режимов работы:

- Работа с текстом;

- Копирование, удаление, вставка, перемещение блоков;
- Подготовка баз знаний с помощью данного текстового редактора заключается в последовательном выполнении ряда этапов:

1. набор БЗ;
2. редактирование БЗ;
3. открытие ранее разработанного файла БЗ;
4. сохранение файла БЗ на магнитном диске.

Все операции для работы с текстом базы знаний можно осуществлять либо выбрав один из пунктов главного меню программы, либо путем выбора соответствующей пиктограммы на инструментальной панели редактора, либо путем выбора соответствующего пункта всплывающего меню окна текстового редактора.

Подготовка баз знаний с помощью режима вставки, не только значительно сокращает количество ошибок, но и сокращает временные затраты на создание базы знаний. Он заключается в последовательном выполнении ряда этапов:

1. Открытие ранее разработанного файла БЗ;
2. Ввод гипотез и их значений, параметров и их значений с помощью команды системы меню ГИПОТЕЗЫ. Для ввода классов и элементов класса необходимо использовать клавишу Insert, а для записи в раздел программы базы знаний необходимо использовать кнопку ОК.

3. Для набора разделов программы можно использовать панель инструментов: гипотез, параметров, ключевых слов. Выбор гипотез и значений параметров производится правой кнопкой мыши в поле инструментов ГИПОТЕЗЫ, ПАРАМЕТРЫ.

4. Редактирование БЗ.
5. Сохранение БЗ.

После разработки всех разделов программы ЭС пользователь может запустить ЭС на выполнение. При запуске происходит чтение файла БЗ пользователя.

Данный режим работы программы заключается в последовательном выполнении ряда этапов:

1. Указание пути исполнения правил с помощью команды системы меню **Настройки | Путь исполнения правил, | Установка заданного пути,**

так же можно установить последовательный путь с помощью команды системы меню “Настройки|Путь исполнения правил|Установка последовательного пути”;

2. Выбор метода вывода и метода поиска решений с помощью команды системы меню “**Настройки | Выбор метода**”;

3. Запуск ЭС с помощью команды системы меню **Запуск | Автоматический запуск**;

4. Проверка правильности работы системы с помощью команды системы меню **Запуск | Протокол решения**.

Пример №1:

ГИПОТЕЗЫ:

дерево {ёлка, сосна, берёза}

цветок {ромашка, роза}

ПАРАМЕТРЫ:

оперение {листья, иголки}

ветки {вверх, вниз}

хвоя {короткая, длинная}

кора {белая, чёрно_белая, чёрная}

цвет {белый, розовый}

шипы {есть, нет}

РАЗДЕЛ ДЕРЕВЬЯ

NAME Правило1

IF оперение листья AND кора чёрно_белая

THEN дерево берёза [0,9]

ELSE дерево берёза [-0,9]

END

NAME Правило2

IF оперение иголки AND NOT хвоя короткая

THEN дерево сосна [0,9]

ELSE дерево сосна [-0,9]

END

NAME Правило3

IF ветки вверх

THEN дерево берёза [0,5], дерево сосна [0,5], дерево ёлка [-0,5]

ELSE дерево берёза [-0,5], дерево сосна [-0,5], дерево ёлка [0,5]

END

КОНЕЦ РАЗДЕЛА ДЕРЕВЬЯ

NAME Правило5

IF цвет белый

THEN IF шипы есть

THEN цветок роза [0,8]

ELSE цветок ромашка [0,8]

ELSE

END

КОНЕЦ РАЗДЕЛА ЦВЕТЫ

Пример №2

В базу знаний (БЗ) в качестве гипотез внесены пять студентов, обладающих следующими характеристиками:

Разряд {1, 1ю, 2, 3, КМС}

Вид_спорта {Гири, легкая_атлетика}

Группа {подгруппа_1, подгруппа_2}

Рост {175, 177, 180, 190}

Дата_рождения {1991, 1992, 1993}

Режим {да, нет}

Соревнования {учавствует, нет}

Параметры для каждого студента представлены в таблице 1.

Таблица 1 – Параметры группы студентов

	Разряд	Вид_спорта	Группа	Рост	Дата_Рождения	Режим	Соревнования	ГТО
Сучок	КМС	Гири	подгруппа_2	180	1993	да	учавствует	гто5
Першин	1	легкая_атлетика	подгруппа_2	190	1991	нет	учавствует	гто1
Крутиков	КМС	легкая_атлетика	подгруппа_1	190	1993	нет	учавствует	гто4
Куропаткин	2	легкая_атлетика	подгруппа_1	175	1992	нет	нет	гто2
Смирнов	3	легкая_атлетика	подгруппа_1	177	1993	да	учавствует	гто3
Белобородов	КМС	акробатика	подгруппа_1	122	1995	нет	нет	гто3
Волков	КМС	Лыжи	подгруппа_2	181	1994	да	учавствует	гто3

ГИПОТЕЗЫ

Студент {Куропаткин, Першин, Крутиков, Сучок, Смирнов, Белобородов, Волков}

ПАРАМЕТРЫ:

Разряд {1, 1ю, 2, 3, КМС}

Вид_спорта {Гири, легкая_атлетика, лыжи, акробатика}

Группа {подгруппа_1, подгруппа_2}

Рост {175, 177, 180, 190, 122, 181}

Дата_рождения {1991, 1992, 1993, 1994, 1995}

Режим {да, нет}

Соревнования {участвует, нет}

ГТО {гто1, гто2, гто3, гто4, гто5}

ПЕРЕМЕННЫЕ:

NAME 1

IF Рост 175

THEN Студент Куропаткин [0,8], Студент Крутиков [-1,0], Студент Смирнов [-1,0], Студент Сучок [-1,0], Студент Першин [-1,0], Студент Волков [-1,0], Студент Белобородов [-1,0]

ELSE

END

NAME 2

IF Рост 177

THEN Студент Смирнов [0,8], Студент Крутиков [-1,0], Студент Куропаткин [-1,0], Студент Сучок [-1,0], Студент Першин [-1,0], Студент Волков [-1,0], Студент Белобородов [-1,0]

ELSE

END

NAME 3

IF Рост 180

THEN Студент Сучок [0,8], Студент Крутиков [-1,0], Студент Смирнов [-1,0], Студент Куропаткин [-1,0], Студент Першин [-1,0], Студент Волков [-1,0], Студент Белобородов [-1,0]

ELSE

END

NAME 4

IF Рост 190 AND Группа подгруппа_2

THEN Студент Першин [0,8], Студент Крутиков [-1,0], Студент Смирнов [-1,0], Студент Сучок [-1,0], Студент Куропаткин [-1,0], Студент Волков [-1,0], Студент Белобородов [-1,0]

ELSE

END

NAME 5

IF Рост 190 AND Группа подгруппа_1

THEN Студент Крутиков [0,8], Студент Куропаткин [-1,0], Студент Смирнов [-1,0], Студент Сучок [-1,0], Студент Першин [-1,0], Студент Волков [-1,0], Студент Белобородов [-1,0]

ELSE

END

NAME 6

IF Разряд 2

THEN Студент Куропаткин [0,8], Студент Сучок [-1,0], Студент Крутиков [-1,0], Студент Смирнов [-1,0], Студент Першин [-1,0]

ELSE IF Разряд 3

THEN Студент Смирнов [0,8], Студент Сучок [-1,0], Студент Крутиков [-1,0], Студент Куропаткин [-1,0], Студент Першин [-1,0]

ELSE IF Разряд 1

THEN Студент Першин [0,8], Студент Сучок [-1,0], Студент Крутиков [-1,0], Студент Куропаткин [-1,0], Студент Смирнов [-1,0]

ELSE Студент Крутиков [0,8], Студент Сучок [0,8], Студент Крутиков [-1,0], Студент Куропаткин [-1,0], Студент Першин [-1,0], Студент Волков [0,8], Студент Белобородов [0,8]

END

NAME 7

IF Дата_рождения 1991

THEN Студент Першин [0,8], Студент Сучок [-1,0], Студент Крутиков [-1,0], Студент Куропаткин [-1,0], Студент Смирнов [-1,0]

ELSE IF Дата_рождения 1993

THEN Студент Смирнов [0,8], Студент Сучок [0,8], Студент Крутиков [0,8], Студент Куропаткин [-1,0], Студент Першин [-1,0]

ELSE Студент Куропаткин [0,8]

END

NAME 8

IF Группа подгруппа_2

THEN Студент Першин [0,8], Студент Сучок [0,8]

ELSE

END

NAME 9

IF Группа подгруппа_2 AND Дата_рождения 1991

THEN Студент Першин [0,8]

ELSE

END

NAME 10

IF Вид_спорта Гири

THEN Студент Сучок [0,8], Студент Куропаткин [-1,0], Студент Крутиков [-1,0], Студент Смирнов [-1,0], Студент Першин [-1,0]

ELSE

END

NAME 11

IF Вид_спорта легкая_атлетика

THEN Студент Сучок [-1,0], Студент Куропаткин [0,8], Студент Крутиков [0,8], Студент Смирнов [0,8], Студент Першин [0,8]

ELSE

END

NAME 12

IF Группа подгруппа_2 AND Дата_рождения 1993

THEN Студент Сучок [0,8]

ELSE

END

NAME 13

IF Группа подгруппа_1 AND Дата_рождения 1993

THEN Студент Крутиков [0,8], Студент Смирнов [0,8]

ELSE

END

NAME 14

IF Группа подгруппа_1 AND Дата_рождения 1992

THEN Студент Куропаткин [0,8]

ELSE

END

NAME 15

IF Соревнования участвует AND Режим да

THEN Студент Сучок [0,8], Студент Смирнов [0,8], Студент Волков [0,8]

ELSE

END

NAME 16

IF Соревнования участвует AND Режим нет

THEN Студент Крутиков [0,7], Студент Першин [0,6]

ELSE

END

NAME 17

IF Соревнования нет AND Режим нет

THEN Студент Куропаткин [0,8], Студент Белобородов [0,8]

ELSE

END

NAME 18

IF Группа подгруппа_2 AND Дата_рождения 1994

THEN Студент Волков [0,8]

ELSE

END

NAME 19

IF Вид_спорта лыжи

THEN Студент Волков [0,8], Студент Куропаткин [-1,0], Студент Крутиков [-1,0], Студент Смирнов [-1,0], Студент Першин [-1,0], Студент Белобородов [-1,0]

ELSE

END

NAME 20

IF Вид_спорта акробатика

THEN Студент Белобородов [0,8]

ELSE

END

NAME 21

IF Группа подгруппа_1 AND Дата_рождения 1995

THEN Студент Белобородов [0,8]

ELSE

END

NAME 22

IF Рост 122

THEN Студент Белобородов [0,8], Студент Крутиков [-1,0], Студент Смирнов [-1,0], Студент Сучок [-1,0], Студент Першин [-1,0], Студент Волков [-1,0], Студент Куропаткин [-1,0]

ELSE

END

NAME 23

IF Рост 181

THEN Студент Волков [0,8], Студент Крутиков [-1,0], Студент Смирнов [-1,0], Студент Сучок [-1,0], Студент Першин [-1,0], Студент Белобородов [-1,0], Студент Куропаткин [-1,0]

ELSE

END

Дерево логического вывода на примере правила 11 представлено на рисунке 1.

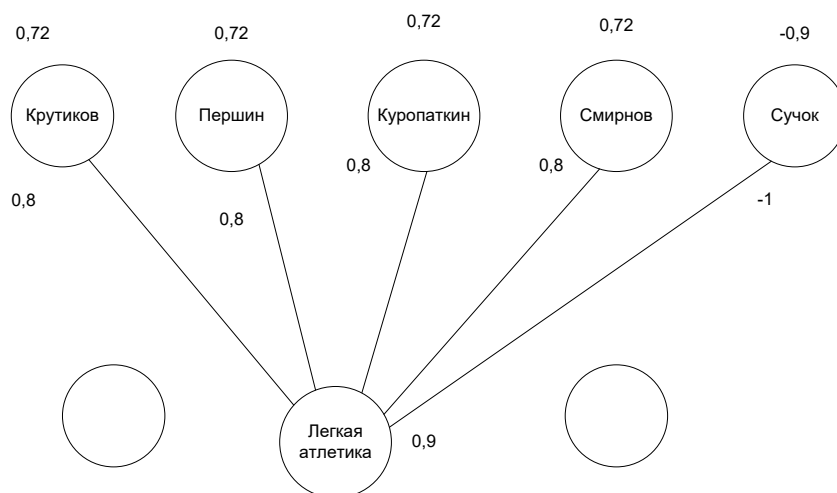


Рисунок 1 – Пример дерева логического вывода

Экранные формы применения правила 11 (поиск прямой в ширину) представлены на рисунках 2-4.

Найдено решение методом: Прямой в ширину время выполнения: 0:00:11 /100

Гипотеза	Значение гипотезы	Козфициент уверенности
Студент	Куропаткин	0,72
Студент	Крутиков	0,72
Студент	Смирнов	0,72
Студент	Першин	0,72
Студент	Сучок	-0,9

Рисунок 2 – Результат применения правила 11 (поиск прямой в ширину)

Козфициенты фактов

Параметр	Значение	Козфициент
Вид_спорта	легкая_атлетика	0,9

Рисунок 3 – Ветка решения

Правило	Решение	Коэффициент
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
Гипотеза	Студент Сучок	-0,9
Гипотеза	Студент Куропа...	0,72
Гипотеза	Студент Крутик...	0,72
Гипотеза	Студент Смирнов	0,72
Гипотеза	Студент Першин	0,72
12		
13		
14		
15		
16		
17		
18		
19		
20		

Рисунок 4 – Протокол решения

Подсчитываются коэффициенты.

Студент Крутиков

$$K=0,8*0,9=0,72$$

Студент Смирнов

$$K=0,8*0,9=0,72$$

Студент Першин

$$K=0,8*0,9=0,72$$

Студент Куропаткин

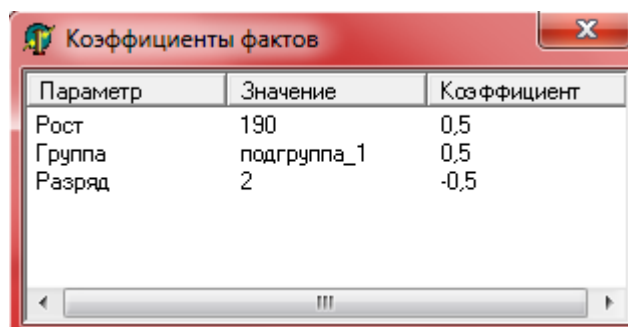
$$K=0,8*0,9=0,72$$

Студент Сучок

$$K=-1*0,9=-0,9$$

В данной задаче не используются другие правила кроме 11, поэтому общие коэффициенты доверия будут равны подсчитанным по правилу 11.

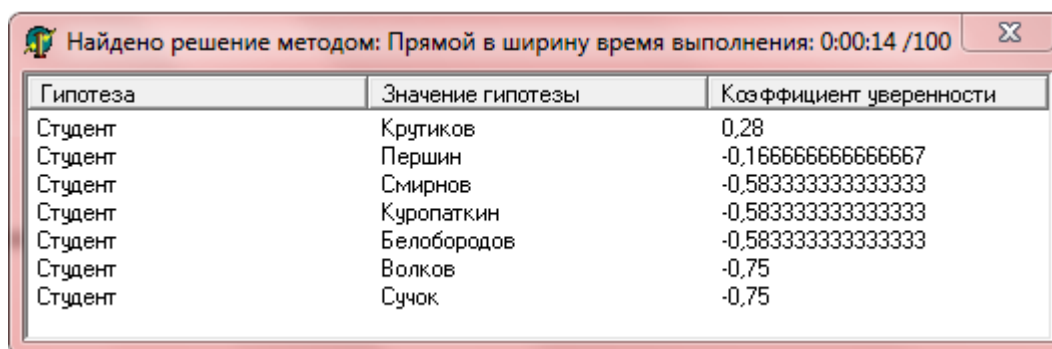
Производится решение по ветке, представленной на рисунке 5.



Параметр	Значение	Коэффициент
Рост	190	0,5
Группа	подгруппа_1	0,5
Разряд	2	-0,5

Рисунок 5 – Ветка решения

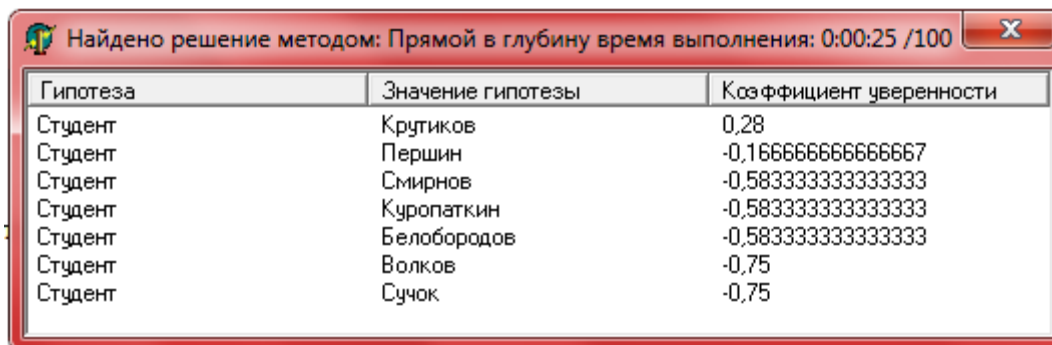
Решение методом прямой в ширину составило 14 секунд (Рисунок 6)



Гипотеза	Значение гипотезы	Коэффициент уверенности
Студент	Крутиков	0,28
Студент	Першин	-0,166666666666667
Студент	Смирнов	-0,583333333333333
Студент	Куропаткин	-0,583333333333333
Студент	Белобородов	-0,583333333333333
Студент	Волков	-0,75
Студент	Сучок	-0,75

Рисунок 6 – Решение методом прямой в ширину

Решение методом прямой в глубину составило 25 секунд (Рисунок 7)



Гипотеза	Значение гипотезы	Коэффициент уверенности
Студент	Крутиков	0,28
Студент	Першин	-0,166666666666667
Студент	Смирнов	-0,583333333333333
Студент	Куропаткин	-0,583333333333333
Студент	Белобородов	-0,583333333333333
Студент	Волков	-0,75
Студент	Сучок	-0,75

Рисунок 7 – Решение методом прямой в глубину

Решение методом обратный в ширину составило 12 секунд (Рисунок 8)

Найдено решение методом: Обратный в ширину время выполнения: 0:00:12 /...

Гипотеза	Значение гипотезы	Коэффициент уверенности
Студент	Крутиков	0,28
Студент	Першин	-0,166666666666667
Студент	Смирнов	-0,583333333333333
Студент	Куропаткин	-0,583333333333333
Студент	Белобородов	-0,583333333333333
Студент	Волков	-0,75
Студент	Сучок	-0,75

Рисунок 8 – Решение методом обратный в ширину

Решение методом обратный в глубину составило 16 секунд (Рисунок 9)

Найдено решение методом: Обратный в глубину время выполнения: 0:00:18 /...

Гипотеза	Значение гипотезы	Коэффициент уверенности
Студент	Крутиков	0,28
Студент	Першин	-0,166666666666667
Студент	Смирнов	-0,583333333333333
Студент	Куропаткин	-0,583333333333333
Студент	Белобородов	-0,583333333333333
Студент	Волков	-0,75
Студент	Сучок	-0,75

Рисунок 9 – Решение методом обратный в глубину

Решение методом до первого совпадения 2 секунды (Рисунок 10)

Найдено решение методом: До первого заключения время выполнения: 0:00:02 /100

Гипотеза	Значение гипотезы	Коэффициент уверенности
Студент	Першин	0,4
Студент	Крутиков	-0,5
Студент	Смирнов	-0,5
Студент	Сучок	-0,5
Студент	Куропаткин	-0,5
Студент	Волков	-0,5
Студент	Белобородов	-0,5

Рисунок 10 – Решение методом до первого совпадения

Менее всего времени потребовалось при решении задачи методом до первого совпадения (2 секунды), но поскольку учитывается только первое совпадение, не все правила при решении были проверены. Методы обратный в ширину и прямой в ширину занимают примерно одинаковое количество времени, метод обратный в глубину занимает большее количество времени. Самое большое количество времени занимает метод прямой в глубину.

Экранная форма оболочки ANIES приведена на рисунке 11.

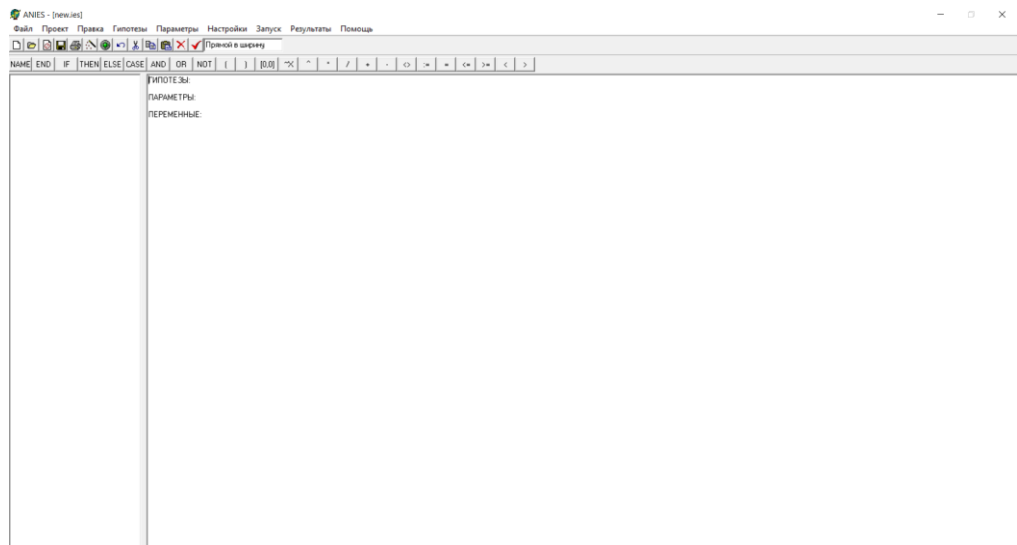


Рисунок 11 - Экранная форма оболочки ANIES

Лабораторная работа №2

Прогнозирование численного результата с применением искусственных нейронных сетей

Цель: освоение нейросетевой технологии для решения задач классификации и прогнозирования с помощью программы NeuroPro 0.25, пакета MATLAB и специализированных библиотек языка Python.

Работа содержит следующие этапы:

1. Знакомство с инструментальным программным обеспечением ((NeuroPro 0.25) для решения задач прогнозирования. Описание программы NeuroPro 0.25 приведено ниже.
2. Получение у преподавателя задания по задаче прогнозирования. Создать обучающую выборку с помощью программы Excel в формате *.dbf.
3. Задачу моделирования результатов выполнить для разных вариантов структуры нейросети (не менее 5). Например, вариант 1 - 3 слоя по 10 нейронов в слое; вариант 2 - 3 слоя по 5-7 нейронов в слое и т. п.
4. Зафиксировать результаты, время прогнозирования и значимость входов для каждого из 5-х вариантов.
5. Выбрать и обосновать наилучший вариант нейронной сети.
6. Выполнить упрощение нейронной сети для оптимального варианта структуры. Показать в отчете, какие методы упрощения целесообразно проводить для выбранной структуры нейронной сети.
7. Выбрать и обосновать метод оптимизации для варианта нейронной сети.
8. Привести в отчете анализ значимости входов для выбранного варианта нейронной сети.
9. Демонстрация результатов прогнозирования преподавателю.
10. *Дополнительное задание №1.* Выполнение аналогичного прогнозирования в среде MATLAB, с применением нейросетевых моделей: линейная нейронная сеть (LNN), нейронная сеть прямого распространения (FNN).
11. *Дополнительное задание №2.* Выполнение аналогичного прогнозирования с применением языка Python и специализированных библиотек, с применением нейросетевых моделей: линейная нейронная сеть (LNN), нейронная сеть прямого распространения (FNN).

12. Подготовка отчета по лабораторной работе в электронном виде и защита выполненной работы.

Программа NeuroPro 0.25 является свободно распространяемой альфа-версией разрабатываемого программного продукта для работы с нейронными сетями и производства знаний из данных с помощью обучаемых искусственных нейронных сетей.

В данной программе реализованы только сети слоистой архитектуры. В слоистой сети все нейроны сгруппированы в несколько слоев, нейроны внутри одного слоя могут работать параллельно. Каждый нейрон в слое принимает все выходные сигналы нейронов предыдущего слоя, а его выходной сигнал рассылается всем нейронам следующего слоя.

Каждый нейрон сети имеет адаптивный сумматор, вычисляющий взвешенную сумму приходящих на нейрон сигналов, и следующий за ним нелинейный элемент.

Веса адаптивных связей при создании сети принимают случайные значения и при обучении сети могут изменяться в диапазоне $[-1, 1]$. Число слоев в сети задается пользователем. Обычно не стоит задавать больше трех слоев нейронов. Число нейронов в слоях может быть различным и не зависеть от числа входных и выходных сигналов сети.

В качестве файлов данных (содержащих обучающую выборку для нейронных сетей) используются файлы форматов DBF (форматы пакетов Dbase, FoxBase, FoxPro, Clipper) и DB (Paradox). Возможно чтение и редактирование этих файлов и сохранение измененных файлов на диске. Программа не накладывает ограничений на число записей (строк) в файле данных.

Файлы нейропроекта имеют уникальный формат, поддерживаемый только настоящей программой. В следующих версиях возможно изменение этого формата без сохранения совместимости с настоящей версией.

Файлы вербального описания сети имеют стандартный ASCII-формат текстовых файлов данных и могут читаться всеми программами-редакторами текстов.

Меню программы содержит следующие пункты, относящиеся к нейронным сетям и работе с ними:

Файл - базовые операции с файлами:

Создать - создает новый файл проекта.

Открыть - открывает существующий на диске файл проекта.

Сохранить - сохраняет файл. Возможно сохранение файлов проекта, файлов данных и файлов вербального описания сети.

Сохранить как - сохраняет файл под другим именем. Возможно сохранение файлов проекта, файлов данных и файлов вербального описания сети. См. Окно редактирования файла данных и Окно вербального описания сети.

Выход - завершение работы программы.

Нейросеть - операции с нейронными сетями. Операция выполняется над активной в данный момент в нейропроекте нейросетью.

Обучение - обучение нейронной сети.

Тестирование - тестирование нейронной сети.

Сокращение числа входных сигналов - удаление наименее значимых входных сигналов.

Сокращение числа синапсов - удаление наименее значимых синапсов сети.

Сокращение числа неоднородных входов - удаление наименее значимых неоднородных входов нейронов сети.

Равномерное упрощение сети - сокращение числа приходящих на нейроны сети сигналов до задаваемого пользователем. Бинаризация синапсов сети - приведение значений весов синапсов и неоднородных входов нейронов к значениям -1 и 1.

Вербализация - генерация вербального описания нейронной сети.

Значимость входов - подсчет и отображение значимости входных сигналов нейронной сети.

Возмущение весов синапсов - добавление случайных поправок к весам синапсов сети.

Настройка - операции по настройке. Настройки действуют в пределах нейропроекта, сохраняются в файле нейропроекта и восстанавливаются при его чтении программой.

Метод оптимизации - выбор метода оптимизации для обучения сети. Из трех реализованных в настоящее время в программе методов (градиентный спуск, модифицированный ParTan метод и метод сопряженных градиентов) при создании нейропроекта автоматически предлагается метод «Сопряжённые градиенты».

Норма накопления значимости - выбор нормы накопления градиента при подсчете показателей значимости. При создании нейропроекта автоматически выбирается норма в виде суммы модулей.

Работа с нейронными сетями возможна только в рамках некоторого нейропроекта. Для того, чтобы создать нейропроект, необходимо выбрать пункт меню Файл/Создать или нажать кнопку Создать в панели кнопок. После создания нейропроекта в него можно вставлять нейронные сети и работать с последними. Созданный нейропроект может быть сохранен на диске при помощи команд меню Файл/Сохранить, Файл/Сохранить как или нажатием на кнопку Сохранить.

В дальнейшем возможна работа с сохраненными файлами нейропроекта - для этого необходимо выбрать пункт меню Файл/Открыть или нажать кнопку Открыть и далее выбрать в диалоговом окне имя нужного нейропроекта.

Большинство операций с нейронными сетями требуют присутствия подключенного к нейропроекту файла данных. Для подключения файла данных или его замены необходимо нажать кнопку. Открыть файл данных в окне нейропроекта и далее выбрать имя необходимого файла данных. Открытый файл данных отображается в собственном окне, где предоставляется возможность его редактирования. При закрытии окна файла данных подключение к нейропроекту завершается. При подключенном файле данных можно проводить операции создания новых нейросетей, их обучения, тестирования и упрощения.

Диалог создания нейронной сети предназначен для задания спецификаций для создаваемой нейронной сети. Элементы диалога:

Входы и выходы - лист для определения использования нейронной сетью имеющихся в файле данных полей. Поля в файле данных - список полей в файле данных. Использование поля - использование текущего поля нейронной сетью. Варианты:

Поле не числовое и недоступно сети - поле не является числовым и не может обрабатываться нейронной сетью.

Поле не используется сетью - данное числовое поле не используется сетью.

Поле является входным для сети - значения данного поля подаются на входы сети.

Поле является выходным для сети - нейросеть обучается прогнозировать значения этого поля.

Диапазон изменения значений поля - минимальное и максимальное значение поля в файле данных.

Оценивание поля - способ оценивания выходного поля при обучении сети.

Число входных полей - число полей в файле данных, используемых сетью в качестве входных.

Число входов сети - число входных сигналов сети.

Число выходных полей - число полей в файле данных, используемых сетью в качестве выходных.

Число выходов сети - число выходных сигналов сети.

Структура сети - лист для задания структуры нейронной сети.

Монотонность - создание сети монотонной архитектуры.

Число слоев нейронов - число слоев нейронов в сети. Изменяется от 1 до 10. Дополнительно после последнего слоя нейронов создается слой выходных сумматоров с числом сумматоров, равным числу выходных сигналов сети. По умолчанию предлагается 3 слоя нейронов. Для каждого слоя нейронов возможно задание следующих характеристик:

Число нейронов - число нейронов в слое. Изменяется от 1 до 100. По умолчанию предлагается 10 нейронов в слое.

Нелинейность - вид нелинейного преобразователя нейронов данного слоя. На данный момент реализована только сигмоидная нелинейность вида $f(A)=A/(c+|A|)$, где c - характеристика нейрона.

Характеристика - значение не обучаемой константы, используемой нелинейным преобразователем. Может изменяться в диапазоне от 0,0001 до 1 для описанной выше сигмоидной нелинейности. По умолчанию предлагается значение « c », равное 0,1. Чем больше значение характеристики, тем лучше интерполяционные и экстраполяционные способности обученной сети, но, как правило, для этого требуется более длительное обучение.

После последнего слоя нейронов строится слой выходных сумматоров с числом сумматоров, равным числу выходных сигналов сети. Имя сети - имя нейронной сети в списке сетей нейропроекта. **Создать/Изменение** - создание нейронной сети или внесение изменений в ее параметры.

Отменить - отмена создания или изменения параметров нейронной сети. После создания нейронной сети она появляется в списке сетей нейропроекта и становится активной.

Для обучения активной в данный момент в нейропроекте нейронной сети необходимо выбрать пункт меню **Нейросеть/Обучение**. Если подключенный к нейропроекту файл данных не содержит необходимых полей (а это возможно, когда создается сеть по одному файлу данных, а далее проверяется ее обучение, тестирование или упрощение по данным из другого файла), то выдается сообщение о несовместимости нейросети и файла данных. Если же в файле данных имеются все необходимые поля и он не пустой, то запускается процесс обучения сети. При этом на экран выводится Окно обучения и упрощения сети, где пользователь имеет возможность наблюдать процесс обучения и при необходимости самостоятельно завершить обучение нажатием кнопки **Завершить**.

Обучение прекращается при достижении нулевого значения средней оценки на задачнике, в случае невозможности дальнейшего улучшения оценки либо при аварийных ситуациях (нулевой или бесконечный шаг в направлении оптимизации).

Имея нейронную сеть, можно посмотреть, насколько точно она прогнозирует значения выходных полей в файле данных. Для тестирования нейронной сети необходимо выбрать пункт меню **Нейросеть/Тестирование**. Результат тестирования сети выводится в Окно тестирования сети.

Для вычисления показателей значимости входных сигналов нейронной сети необходимо выбрать пункт меню **Нейросеть/Значимость входов**. Вычисленные показатели значимости выводятся в Окно значимости входов.

Если окно значимости входов отображено на экране и для данной нейронной сети запускается процесс упрощения, то окно значимости динамически изменяет отображаемые данные после каждого перерасчета показателей значимости входных сигналов (при каждом шаге упрощения - при исключении входного сигнала или контрастировании/бинаризации синапса).

Упрощение нейронной сети выполняется на основе вычисленных показателей значимости. Показатели значимости вычисляются в пяти точках в пространстве адаптивных параметров сети и усредняются в одной из норм.

Процедура упрощения строится как последовательное исключение из сети наименее значимого входного сигнала или элемента сети, дальнейшее дообучение сети и исключение другого сигнала или элемента, если удалось дообучить сеть до заданной пользователем точности. В противном случае процедура упрощения останавливается.

Предоставляются следующие возможности по упрощению сети:

- сокращение числа входных сигналов;

- сокращение числа синапсов сети;
- сокращение числа неоднородных входов нейронов сети;
- равномерное упрощение сети, чтобы на каждый нейрон сети приходило не более n сигналов.

- Бинаризация весов синапсов и неоднородных входов сети. Бинаризованные синапсы и неоднородные входы в дальнейшем не обучаются. Нейрон сети считается отконтрастированным, если у него удалены все выходы или его сигнал не используется нейронами следующего слоя. Входной сигнал считается отконтрастированным, если отконтрастированы все синапсы, по которым он подавался сети. Отконтрастированные элементы физически остаются в нейронной сети, но при генерации вербального описания сети не вносятся в вербальное описание.

При генерации вербального описания в тексте перечисляются используемые поля файла данных, правила их предобработки для подачи сети, описание нелинейных функций нейронов, функционирование нейронной сети послойно и понейронно, правила нормировки выходных сигналов сети в диапазон истинных значений. Сигналам, генерируемым нейронами сети, присваиваются некоторые имена и в дальнейшем пользователь при анализе сети может именовать эти сигналы в терминах проблемной области.

Пример:

Выполнить прогнозирование цены садового участка **4500** при конкретных значениях 10 факторов (входов), используя инструментальную систему NeuroPro. Исходный данные для задания представлены в таблице 2.

Таблица 2 – Исходные данные

Число соток	Состояние земли	Наличие водопровода	Наличие колодца	электричество	теплица	парник	дом	удаленность от города(км)	рейсовые маршруты	Цена участка
6	1	0	1	1	1	0	1	20	2	4000
6	2	0	1	1	1	0	1	20	2	6000
6	2	1	1	1	2	1	2	10	1	10000
6	0	1	1	1	1	1	3	20	2	7000
6	1	1	0	1	0	1	3	10	2	8000
3	2	1	1	1	2	1	3	10	2	6000

10	2	1	1	1	2	1	3	30	2	10000
5	2	0	1	1	1	0	3	140	2	6000
4	2	0	1	1	0	2	2	60	3	2000
7	0	0	1	0	1	2	3	10	1	3000
3	2	1	1	1	2	1	3	80	2	3700
5	2	1	1	1	1	1	2	50	2	4500
20	1	1	1	1	0	1	3	120	1	10000
5	2	0	1	1	1	1	1	15	1	5000
6	1	1	0	0	0	2	3	20	2	7500
3	2	1	1	1	1	0	2	10	1	4200
15	1	0	1	1	3	2	3	70	2	8200
5	0	1	1	0	1	1	1	8	1	5300
6	2	1	0	1	1	0	2	5	1	7000
3	1	1	1	0	1	1	2	3	1	6500

Входы

-Число соток.

-Состояние земли.

-Наличие водопровода.

-Наличие колодца.

-Наличие электричества.

-Наличие теплиц.

-Наличие парников.

-Дом.

-Расстояние до города (км).

-Автобусные маршруты и электрички.

Выход: цена участка (руб.).

Выполнение задания

В ходе выполнения лабораторной работы обучено 6 нейронных сетей, с разным количеством нейронов в слоях. Данные по нейронным сетям представлены в таблице 3.

Таблица 3 – Нейронные сети

Название сети	Число слоев	Число нейронов	Число циклов обучения	Кол-во обученных примеров	Макс. Ошибка	Средняя ошибка	Прогноз	Ошибка прогноза
Network 2		15-15-5	48	20	476.0568	767.796	4642.588	142.588
Network 11		15-18-11	38	20	79.44629	52.5181	3838.763	661.237
Network 8		22-22-7	18	20	99.99123	56.4868	3696.177	803.823
Network 5		21-30-13	36	20	97.98291	57.1892	4102.544	397.456
Network 31		9-10-5	44	20	79.70215	59.3313	3896.391	603.609
Network 33		10-10-10	39	20	79.22119	43.0672	4476.918	23.082

По результатам тестирования выбирается сеть Network 33, имеющая наименьшую ошибку прогноза, среднюю ошибку и максимальную ошибку.

Значимость входных сигналов для выбранной сети представлена на рисунке 12.

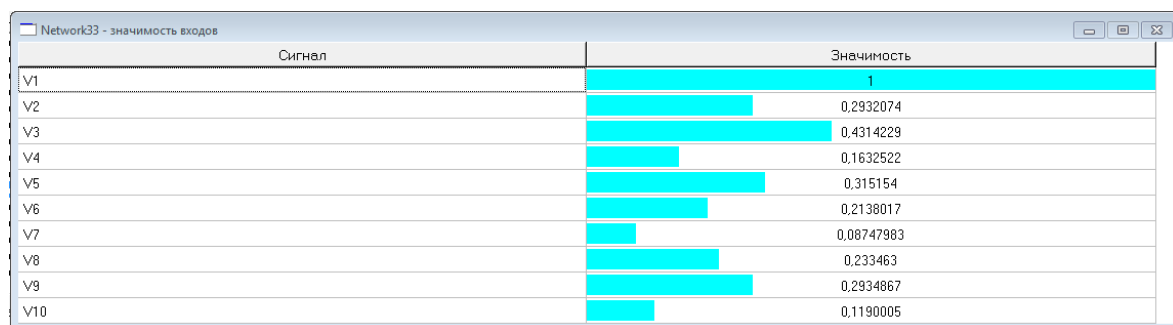


Рисунок 12- Значимость входных сигналов

Тестирование проведенное для выбранной сети представлено на рисунке 13.

Тестирование Network33			
№	OUT	Прогноз сети	Ошибка
1	4000	3989,655	10,34546
2	6000	5994,301	5,699219
3	10000	9926,934	73,06641
4	7000	7046,747	-46,74658
5	8000	8058,213	-58,21338
6	6000	5946,855	53,14453
7	10000	10073,42	-73,41602
8	6000	6048,721	-48,7207
9	2000	1989,078	10,92151
10	3000	2956,289	43,71094
11	3700	3745,75	-45,74951
12		4476,918	
13	10000	9930,789	69,21094
14	5000	5035,134	-35,13379
15	7500	7467,665	-67,66468

Рисунок 13 – Тестирование

Результаты прогнозирования полученные при упрощениях сети представлены в таблице 4.

Таблица 4 – Методы упрощения сети

Метод упрощения сети	Прогноз	Средняя ошибка	Максимальная ошибка
Сокращение числа входных сигналов	7746.321	47.3983	79.7373
Сокращение числа нейронов	3601.72	61.06355	79.95117
Равномерное упрощение сети	6795.997	64.70992	79.87207
Сокращение числа синапсов	6135.079	43.68476	79.63672
Сокращение числа неоднородных входов	7761.41	47.129	78.12678
Бинаризация весов синапсов и неоднородных входов	4646.955	39.00327	79.79736

Экранные формы при выборе упрощений представлены на рисунках 14-25.

Упрощение Network51

Число циклов обучения: 432
 Шаг: Zero step
 Средняя оценка: 0,0042025
 Правильно решенных примеров: 18 из 20

Удалено входов: 3 (3 из 10)
 Удалено нейронов: 0 (0 из 30)
 Удалено синапсов: 30 (30 из 341)
 Бинаризовано синапсов: 0 (0 из 311)

Готово

Рисунок 14 – Сокращение входных сигналов

№	OUT	Прогноз сети	Ошибка
1	4000	3969,258	30,7417
2	6000	6028,479	-28,47852
3	10000	9923,289	76,71094
4	7000	7021,294	-21,29395
5	8000	7932,079	67,9209
6	6000	6046,396	-46,39551
7	10000	10079,74	-79,7373
8	6000	5930,359	69,64063
9	2000	2063,153	-63,15332
10	3000	2924,865	75,13525
11	3700	3676,998	23,00171
12		7746,321	
13	10000	9950,634	49,36621
14	5000	5030,336	-30,33643
15	7500	7450,010	-49,70000

Рисунок 15 – Тестирование (Network 51 – Копия Network 33)

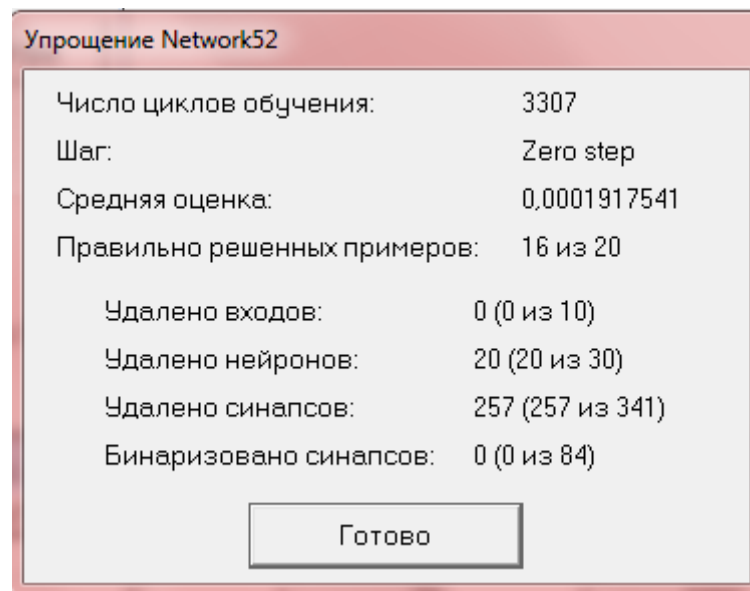


Рисунок 16 – Сокращение числа нейронов

Тестирование Network52			
№	OUT	Прогноз сети	Ошибка
1	4000	3934,053	65,94727
2	6000	6066,194	-66,19385
3	10000	9920,729	79,27051
4	7000	6972,641	27,35889
5	8000	7931,423	68,57715
6	6000	6070,198	-70,19775
7	10000	10028,13	-28,12891
8	6000	5925,906	74,09424
9	2000	2034,805	-34,80481
10	3000	3043,982	-43,98242
11	3700	3772,377	-72,37671
12		3601,72	
13	10000	10079,95	-79,95117
14	5000	5072,492	-72,49219
15	7500	7500,407	-0,40704

Рисунок 17 – Тестирование (Network 52 – Копия Network 33)

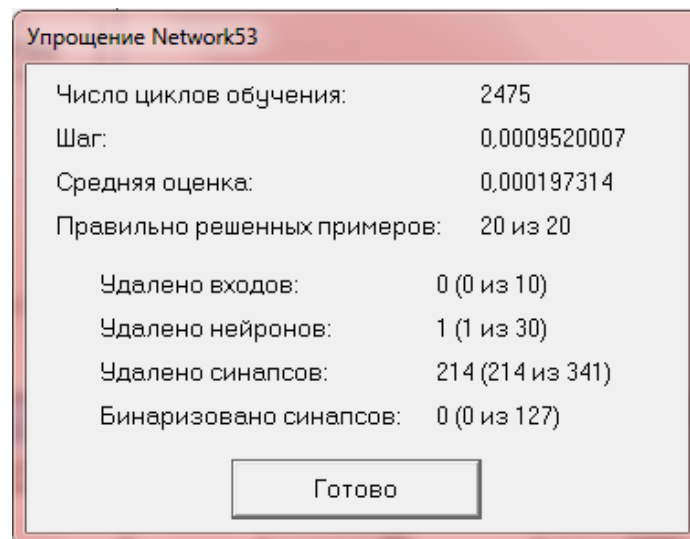


Рисунок 18 – Равномерное упрощение сети

Тестирование Network53			
№	OUT	Прогноз сети	Ошибка
1	4000	3948,888	51,11206
2	6000	6073,652	-73,65186
3	10000	9930,23	69,76953
4	7000	7076,316	-76,31592
5	8000	7948,223	51,77734
6	6000	5920,293	79,70703
7	10000	10073,06	-73,0625
8	6000	5920,128	79,87207
9	2000	1927,894	72,10583
10	3000	3020,496	-20,49561
11	3700	3779,031	-79,03076
12		6795,997	
13	10000	9937,035	62,96484
14	5000	5070,242	-70,24219
15	7500	7544,000	-44,00000

Рисунок 19 – Тестирование (Network 53 – Копия Network 33)

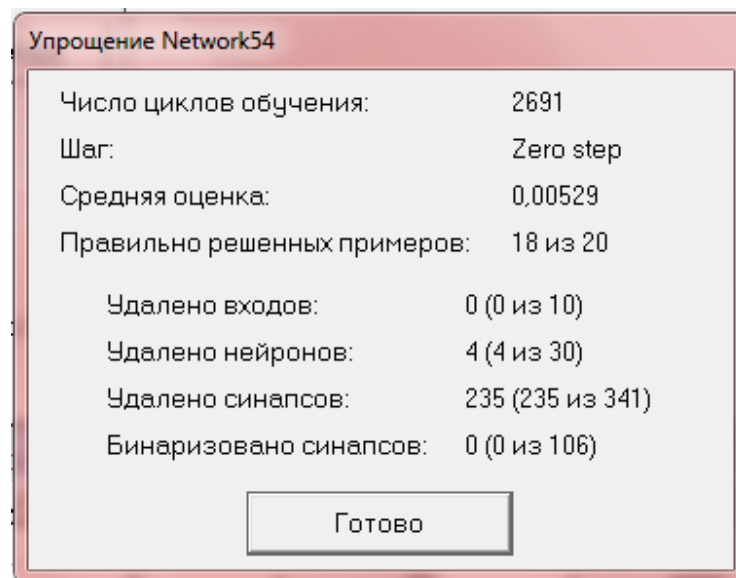


Рисунок 20 – Сокращение числа синапсов

Тестирование Network54			
№	OUT	Прогноз сети	Ошибка
1	4000	3973,85	26,15039
2	6000	5967,023	32,97656
3	10000	9920,363	79,63672
4	7000	7073,078	-73,07813
5	8000	7960,191	39,80859
6	6000	6051,785	-51,78516
7	10000	10078,58	-78,57617
8	6000	6041,115	-41,11523
9	2000	2029,552	-29,55151
10	3000	2925,408	74,5918
11	3700	3690,654	9,346436
12		6135,079	
13	10000	10029,32	-29,31738
14	5000	5062,726	-62,72607
15	7500	7400,000	-10,00000

Рисунок 21 – Тестирование (Network 54 – Копия Network 33)

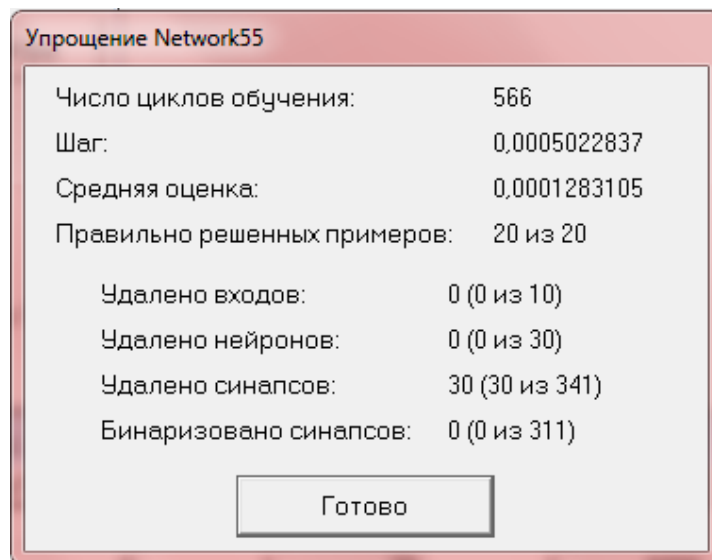


Рисунок 22 – Сокращение числа неоднородных входов

Тестирование Network55			
№	OUT	Прогноз сети	Ошибка
1	4000	3960,265	39,73535
2	6000	5944,055	55,94531
3	10000	9928,296	71,7041
4	7000	7031,596	-31,5957
5	8000	7949,039	50,96094
6	6000	6070,69	-70,68994
7	10000	9921,873	78,12695
8	6000	5947,402	52,59814
9	2000	2041,18	-41,17993
10	3000	3041,034	-41,03418
11	3700	3767,738	-67,73755
12		7761,41	
13	10000	10077,34	-77,33887
14	5000	5012,458	-12,45752
15	7500	7504,404	-4,40410

Рисунок 23 – Тестирование (Network 55 – Копия Network 33)

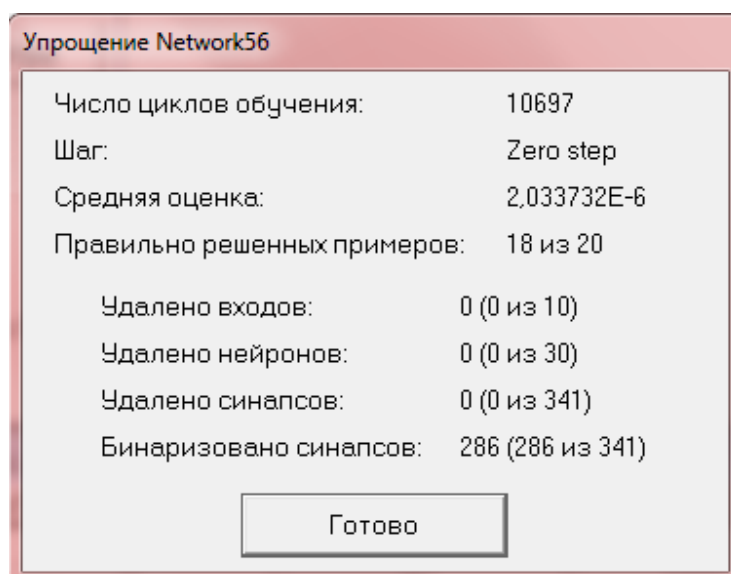


Рисунок 24 – Бинаризация синапсов сети

Тестирование Network56			
№	OUT	Прогноз сети	Ошибка
1	4000	3944,841	55,15869
2	6000	6032,73	-32,72998
3	10000	9937,438	62,56152
4	7000	6933,978	66,02246
5	8000	7951,933	48,06689
6	6000	6008,931	-8,931152
7	10000	10062,56	-62,55957
8	6000	6014,459	-14,45947
9	2000	1967,085	32,91516
10	3000	2950,983	49,01733
11	3700	3761,841	-61,84131
12		4646,955	
13	10000	9995,737	4,262695
14	5000	5000,132	-0,1318359
15	7500	7510,000	-10,00000

Рисунок 25 – Тестирование (Network 56 – Копия Network 33)

Наиболее приемлемым вариантом упрощения сети является вариант бинаризации синапсов сети, при котором сеть имеет минимальную ошибку прогнозирования и минимальную среднюю ошибку. Значимость входных сигналов для упрощенной сети представлена на рисунке 26.

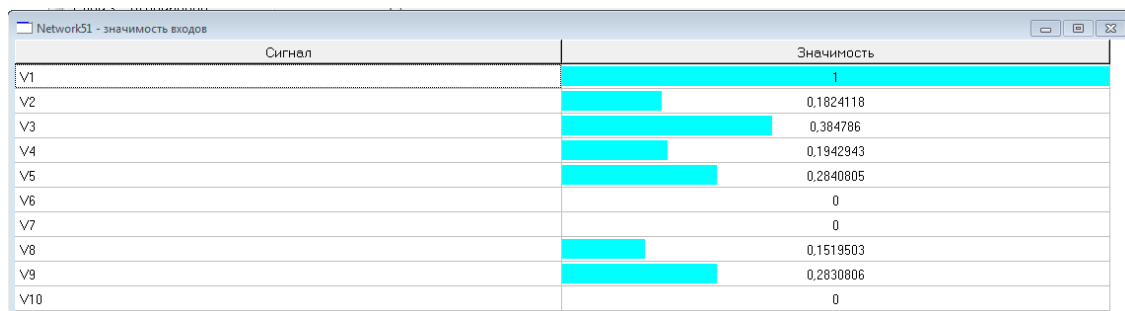


Рисунок 26 – Значимость входных сигналов

Рассмотрены различные методы оптимизации. Данные полученные при различных методах оптимизации представлены в таблице

Таблица 5 – Методы оптимизации

Метод	Число циклов обучения	Кол-во обученных примеров	Макс. Ошибка	Средняя ошибка	Прогноз	Ошибка прогноза
Градиентный спуск	330	20	79.99609	64.9741	4468.179	31.821
Модифицированный Par Tan	1222	20	79.99609	61.8168	4019.398	480.602
Сопряженные градиенты	39	20	79.22119	43.0672	4476.918	23.082
BFGS	37	20	77.28711	50.6696	4960.694	460.694

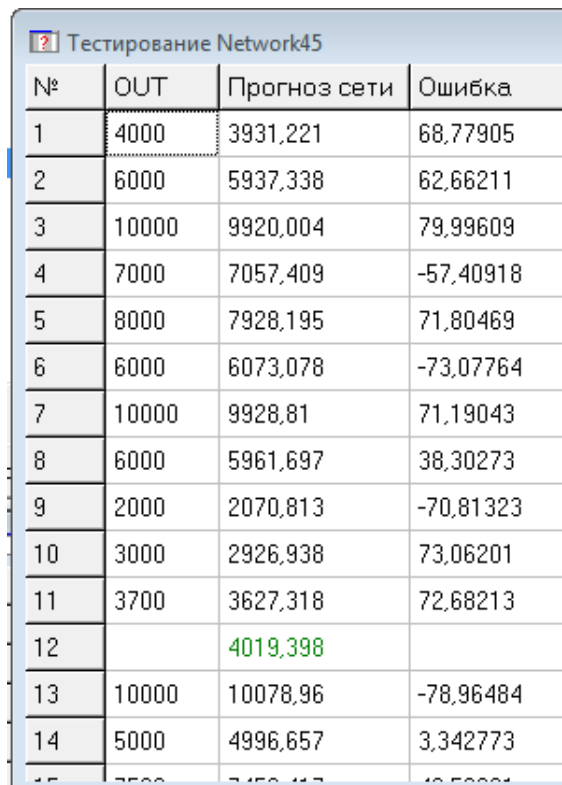
Результаты тестирования при различных метода оптимизации представлены на рисунках 27-29.

Тестирование Network38			
№	OUT	Прогноз сети	Ошибка
1	4000	4072,932	-72,93237
2	6000	5920,004	79,99609
3	10000	9922,658	77,3418
4	7000	6928,044	71,95605
5	8000	7927,26	72,73975
6	6000	6073,939	-73,93896
7	10000	9928,452	71,54785
8	6000	5931,677	68,32275
9	2000	2006,572	-6,571777
10	3000	3074,615	-74,6145
11	3700	3771,834	-71,83398
12		4468,179	
13	10000	10078,63	-78,625
14	5000	5073,567	-73,56689
15	7500	7548,887	-58,48888

Рисунок 27 – Тестирование при оптимизации методом градиентного спуска

Тестирование Network42			
№	OUT	Прогноз сети	Ошибка
1	4000	4050,766	-50,76587
2	6000	5975,57	24,42969
3	10000	10034,67	-34,66797
4	7000	6985,592	14,4082
5	8000	7950,471	49,52881
6	6000	6018,208	-18,20752
7	10000	9922,859	77,14063
8	6000	5922,713	77,28711
9	2000	1943,377	56,62256
10	3000	3040,425	-40,42505
11	3700	3763,524	-63,52441
12		4960,694	
13	10000	10074,28	-74,28027
14	5000	4948,547	51,45264
15	7500	7448,488	-58,58788

Рисунок 28 – Тестирование при BFGS



№	OUT	Прогноз сети	Ошибка
1	4000	3931,221	68,77905
2	6000	5937,338	62,66211
3	10000	9920,004	79,99609
4	7000	7057,409	-57,40918
5	8000	7928,195	71,80469
6	6000	6073,078	-73,07764
7	10000	9928,81	71,19043
8	6000	5961,697	38,30273
9	2000	2070,813	-70,81323
10	3000	2926,938	73,06201
11	3700	3627,318	72,68213
12		4019,398	
13	10000	10078,96	-78,96484
14	5000	4996,657	3,342773
15	7500	7450,417	49,58583

Рисунок 29 - Тестирование при методе модифицированный Par Tan

Наилучшей оптимизацией оказался метод сопряженных градиентов, имеющий более точный прогноз и наименьшие ошибки. Анализ обучающего множества представлен на рисунке 30.

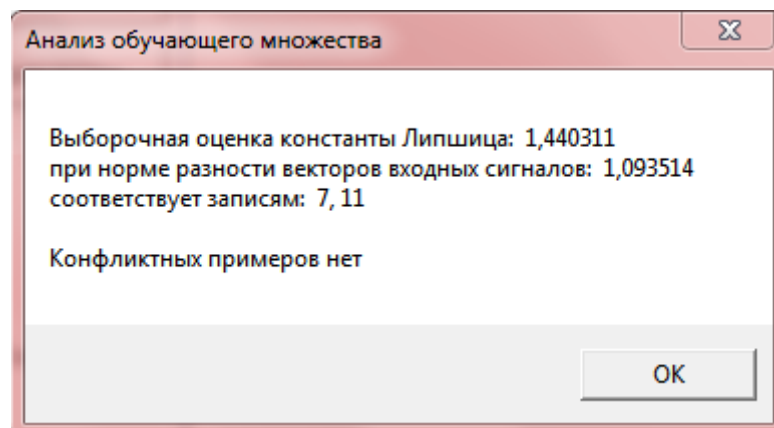


Рисунок 30 – Анализ обучающего множества

Для данной задачи оптимальной является сеть имеющая три слоя нейронов (в каждом из которых по 10 нейронов), не имеющая упрощений. Метод оптимизации – метод сопряженных градиентов.

Задание

Самостоятельно подготовить обучающую выборку (не менее 100 значений), с необходимостью спрогнозировать одно из неизвестных значений. Произвести действия с нейронной сетью, описанные выше в этапах работы.

Дополнительное задание №1 (выполняется по согласованию с преподавателем):

Вторая часть задания заключается в выполнении аналогичных действий (обучении сетей на аналогичной обучающей выборке, см. задание 1) для двух простых моделей нейронных сетей: линейной нейронной сети и нейронной сети прямого распространения. Использовать пакет MATLAB и утилиту NNTool.

Отчет включает в себя краткие теоретические сведения, исходный код, заполненную таблицу, для сравнения разных экземпляров сетей (параметры для включения в таблицу выбираются самостоятельно, обязателен параметр выхода сети, и оценки ошибки/точности), выводы.

MATLAB — это мощная среда программирования и интерактивных вычислений, созданная компанией MathWorks, предназначенная для численного анализа, моделирования, визуализации данных и разработки алгоритмов. Широко применяется в инженерии, науке и исследованиях благодаря удобному языку программирования, обширным библиотекам и инструментам для решения задач в области математики, статистики, обработки сигналов и изображений, а также построения моделей динамических систем. Несмотря на высокую цену лицензий, MATLAB остается популярным выбором среди профессионалов благодаря своей эффективности и простоте использования.

Работу целесообразно начинать с практики проектирования нейронных сетей посредством графического интерфейса пакета NNT, включающего утилиту NNTool (Neural Network Toolbox). Данный интерфейс удобен тем, что позволяет создать, обучить, смоделировать, импортировать и экспортировать нейронные сети и данные без обращения к командному режиму MATLAB. Однако его функциональность ограничена работой с простыми однослойными и двухслойными структурами, поэтому он подходит преимущественно начинающим пользователям.

Опытным специалистам рекомендуется использовать прямое взаимодействие с системой MATLAB через командное окно, поскольку этот способ открывает доступ ко всему функционалу пакета NNT. Помимо этого, важно ознакомиться с интеграцией инструмента NNT с пакетом Simulink, обеспечивающим наглядное представление архитектуры нейронных сетей и их моделирование как в статическом, так и динамическом режиме.

Руководство NNTool подробно описано в [10].

Знакомство со средой MATLAB описано в [16].

Линейная нейронная сеть представляет собой простейшую структуру искусственной нейронной сети, состоящую из одного слоя нейронов, каждый из которых производит линейную комбинацию входных сигналов с последующим применением активационной функции (линейная функция активации). Эта архитектура используется преимущественно для решения простых задач классификации и регрессии, поскольку её возможности ограничены способностью моделировать лишь линейные зависимости между признаками и целевыми переменными. Линейные сети легко интерпретируются и быстро обучаются, однако неспособны эффективно обрабатывать сложные нелинейные взаимосвязи, характерные для большинства реальных задач машинного обучения.

Линейная сеть, показанная ниже, имеет один слой нейронов S , соединенных с входными параметрами R через матрицу весов W .

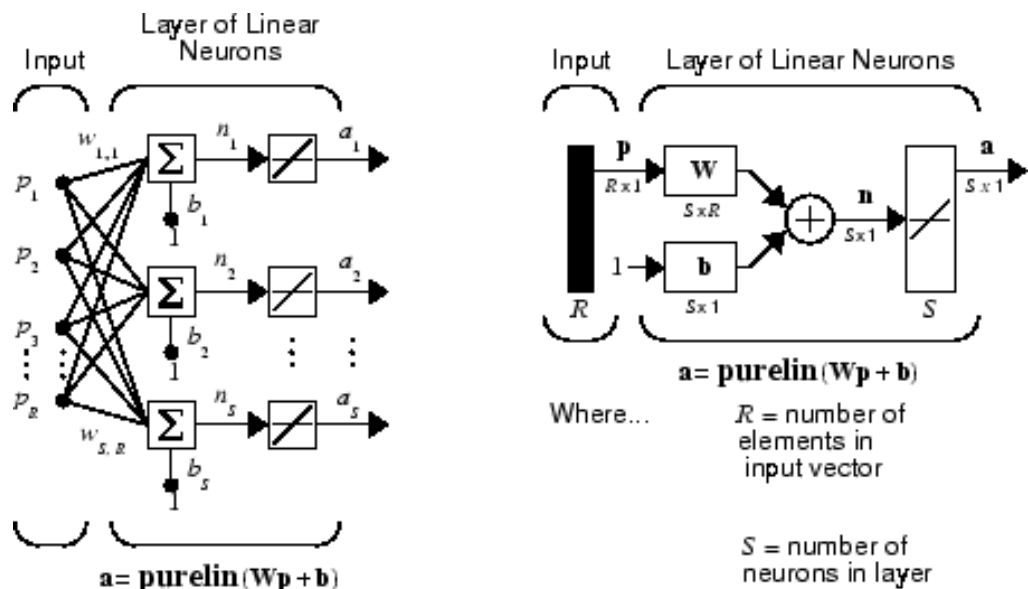


Рисунок 31 – Структура линейного нейрона

Эта сеть имеет ту же базовую структуру как perceptron. Единственная разница - то, что линейный нейрон использует линейную передаточную функцию *purelin*.

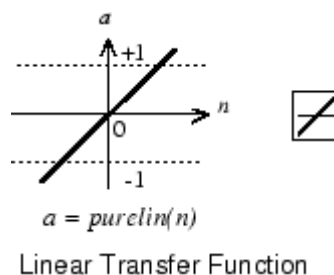


Рисунок 32 – Линейная функция активации

Линейная передаточная функция вычисляет, выход нейрона путем простого возвращения значения передал ей.

$$a = \text{purelin}(n) = \text{purelin}(\mathbf{Wp} + b) = \mathbf{Wp} + b$$

Линейные слои являются единственными слоями линейных нейронов. Они могут быть статическими, с входными задержками 0, или динамическими с входными задержками, больше, чем 0. Они могут быть обучены на простых линейных проблемах временных рядов, но часто используются адаптивно, чтобы продолжить учиться, в то время как развернуто, таким образом, они могут настроить к изменениям в отношении между вводами и выводами, будучи используемым.

`linearlayer(inputDelays, widrowHoffLR)` берет эти аргументы и возвращает линейный слой.

- `inputDelays` Вектор-строка из увеличения 0 или положительных задержек (значение по умолчанию = 0).

- `widrowHoffLR` - Скорость обучения Видроу-Хофф (значение по умолчанию = 0.01)

Если скорость обучения слишком будет мала, изучение будет происходить очень медленно. Однако большая опасность состоит в том, что это может быть слишком большим, и изучение станет нестабильным получившийся в больших изменениях в векторах веса и ошибках, увеличивающихся вместо уменьшения.

Пример создания линейного слоя:

Здесь линейный слой обучен на простой проблеме временных рядов.

```
x = {0 -1 1 1 0 -1 1 0 0 1};  
t = {0 -1 0 2 1 -1 0 1 0 1};  
net = linearlayer(1:2,0.01);  
[Xs,Xi,Ai,Ts] = preparets(net,x,t);  
net = train(net,Xs,Ts,Xi,Ai);  
view(net)  
Y = net(Xs,Xi);  
perf = perform(net,Ts,Y)  
perf = (нажать Enter)
```

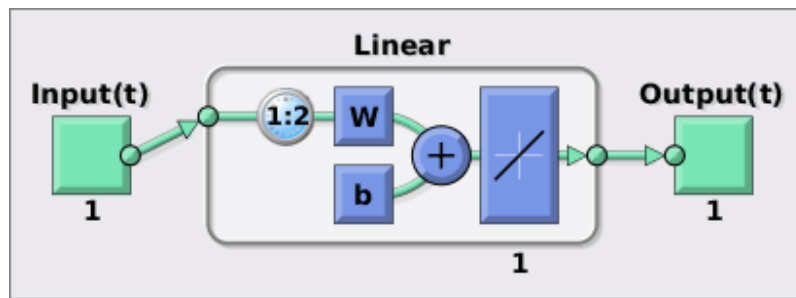


Рисунок 33 – Линейный слой

Матрица W веса в этом случае имеет только одну строку. Сетевой выход

$$\alpha = \text{purelin}(n) = \text{purelin}(Wp + b) = Wp + b$$

или

$$\alpha = w_{1,1}p_1 + w_{1,2}p_2 + b$$

Как перцептрон, линейная сеть имеет *контур решения*, который определяется входными векторами, для которых сетевой вход n является нулем. Для $n = 0$ уравнение $Wp + b = 0$ задает такой контур решения, как показано ниже:

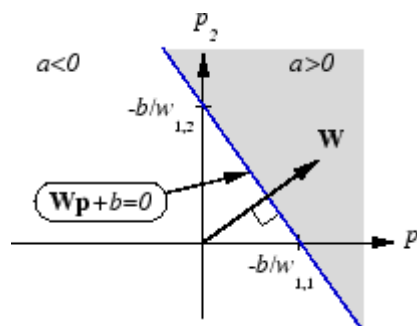


Рисунок 34 – Контур решения

Пример №12

Спрогнозировать численный результат на подготовленной выборке. Выборка имеет набор параметров (12 шт.) и один «эталонный» параметр (ранее в данной лабораторной работе мы обозначили его как выход, фактически значение, которое необходимо спрогнозировать).

Формируется исходный код в среде MATLAB.

```

nets2 = newlin([160 195; 110 120.5; 60 67; 269 304; 14.78 15.48; 11.69 11.8; 54 85.5],1)
nets2.trainParam.epochs = 1000;
nets2.trainParam.goal = 1;
[nets2,trs] = trainlm(nets2, P, T);
Yns=sim(nets2,P);
%Построение графика функций
figure('NumberTitle','off','Name','Функция ','ToolBar','none','MenuBar','none');
plot(P,T,'--k','LineWidth',1);hold on;
A=nets2(P);
Yns=sim(nets2,P);
plot(P,Yns,'--k','LineWidth',1);hold on;
E=T-A;
SSE=sumsq(E)
MSE=mse(E)

```

В результате работы получена следующая модель ИНС

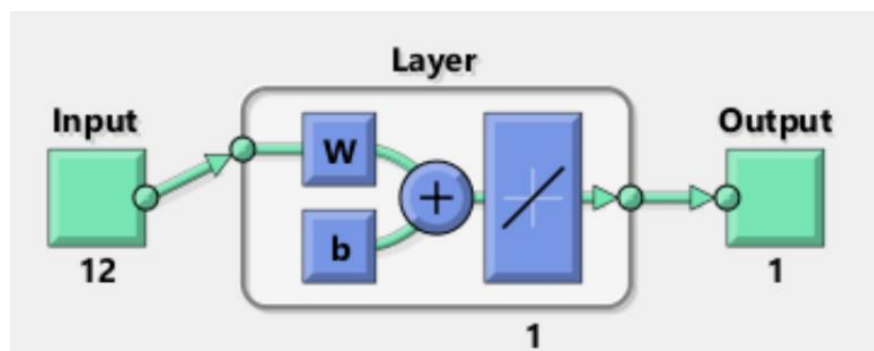


Рисунок 35 – Модель линейной нейронной сети

Результаты работы (оценка MSE – минимальная среднеквадратичная ошибка обучения) в зависимости от метода обучения приведены в таблице 6.

Таблица 6 – Ошибка обучения

Структура сети	Метод обучения Левенберга	Метод обучения с использованием	Метод обучения с использованием
----------------	------------------------------	---------------------------------------	---------------------------------------

	Маркварда (trainlm)	Байесовской регуляризации (trainbr)	градиентного спуска (traingd)
12, 1	MSE=0,0330 (4 эпохи)	MSE= 36,6 (7 эпох)	Сеть не обучилась

После обучения сети построен совмещенный график работы линейного нейронного слоя и эталонных значений обучающей выборки.

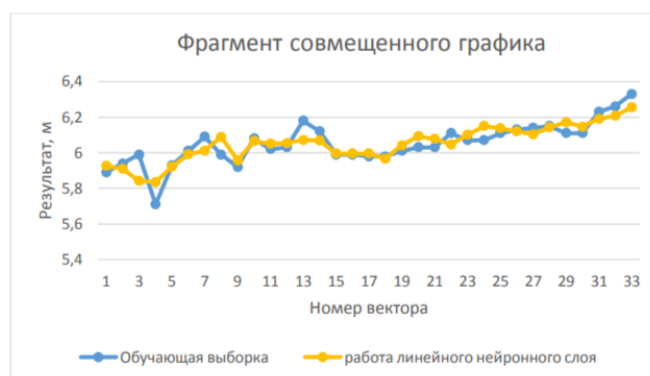


Рисунок 36 – Фрагмент совмещенного графика

Линейная нейронная сеть выявляет лишь часть зависимостей прогнозируемых значений от остальных параметров выборки.

Задание

Построить несколько экземпляров линейной нейронной сети. Обучить на основе выборки из задания 1. Привести исходный код, заполнить таблицу.

Структура сети	Метод обучения 1	Метод обучения 2	Метод обучения 3
?, ?	MSE	MSE	MSE

Нейронная сеть прямого распространения (Feedforward Neural Network) — это тип нейронной сети, в которой передача сигнала осуществляется исключительно в одном направлении, от входного слоя к выходному через промежуточные скрытые слои. Она не имеет обратных связей и способна эффективно решать задачи классификации, регрессии и распознавания образов. Обучение такой сети производится методом обратного

распространения ошибки (Backpropagation), позволяющим автоматически настраивать веса соединений между нейронами для минимизации ошибки предсказания.

Feedforward сети часто имеют один или более скрытые уровни sigmoid нейронов, сопровождаемые слоем выходных линейных нейронов. Многоуровневые нейронные сети с нелинейными функциями передачи позволяют получать нелинейные и линейные отношения между векторами ввода и вывода. Линейный уровень вывода позволяет сети производить значения вне диапазона от -1 до +1. С другой стороны, если Вы хотите ограничить выводы сети (типа между 0 и 1), тогда уровень вывода должен использовать функцию передачи типа logsig. Для сетей с несколькими слоями мы используем номер уровня, чтобы определить верхний индекс на весе матрицы. Соответствующий пример показан в tansig/purelin сети с двумя уровнями, показанной ниже.

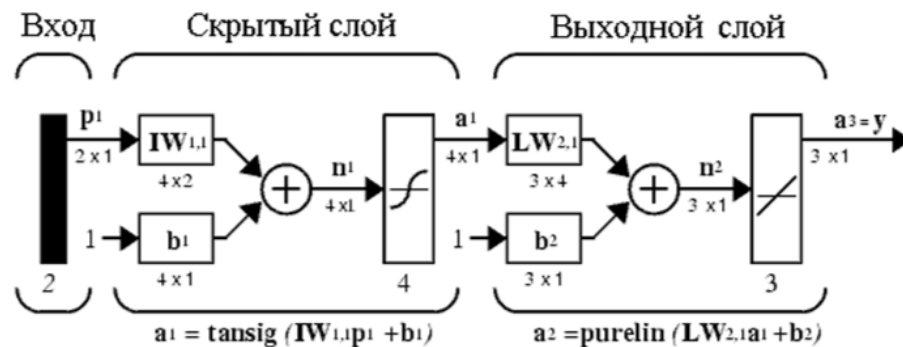


Рисунок 37 - Укрупненная структура FNN- сети

Эта сеть может использоваться как общая функция аппроксимации. Это может приближать любую функцию с конечным числом discontinuities, произвольно хорошо, учитывая нейроны в скрытом уровне.

Создание Сети (newff).

Первый шаг в обучении feedforward сети должен создать сетевой объект. Функция newff создает feedforward сеть. Это требует четырех вводов и возвращает сетевой объект. Первый ввод - R 2 матрицей минимальных и максимальных значений для каждого из R элементов входного вектора. Второй ввод - массив, содержащий размеры каждого уровня. Третий ввод - массив ячеек, содержащий названия функций передачи, которые нужно использовать в каждом уровне. Заключительный ввод содержит название функции обучения, которую нужно использовать. Например, следующая команда создает сеть с двумя уровнями. Имеется один входной вектор с двумя элементами. Значения для первого элемента входного векторного диапазона между -1 и 2, значения

второго элемента входного векторного диапазона между 0 и 5. Имеются три нейрона в первом уровне и одном нейроне во втором уровне вывода. Функция передачи в первом уровне тангенциально - sigmoid, и функция передачи уровня вывода линейна. Функция обучения - traingd.

```
net=newff([-1 2; 0 5],[3,1],{'tansig','purelin'},'traingd');
```

Эта команда создает сетевой объект и также инициализирует веса и уклоны сети; поэтому сеть готова к обучению. В любое время можно повторно инициализировать веса, или исполнить заказную инициализацию. Следующий раздел объясняет детали процесса инициализации. Инициализация Весов (init). Перед обучением feedforward сети, веса и уклоны должны быть инициализированы. Newff команда автоматически инициализирует веса, но Вы можете повторно инициализировать их. Это может быть сделано командой init. Эта функция берет сетевой объект как вход и возвращает сетевой объект со всеми весами и смещает инициализированный. Например повторная инициализация сети: `net = init(net);`

Пример №3

Спрогнозировать численный результат на подготовленной выборке. Выборка имеет набор параметров (7 шт.) и один «эталонный» параметр (ранее в данной лабораторной работе мы обозначили его как выход, фактически значение, которое необходимо спрогнозировать).

Формируется исходный код в среде MATLAB.

```
%Создание многослойных нейронных сетей для аппроксимации
nets2 = newff([160 195; 110 120.5; 60 67; 269 304; 14.78 15.48; 11.8 11.69; 54
85.5],[25,15,1],{'tansig','poslin','poslin'},'trainbr');

%Тренировка сети и получение результатов работы сети
[nets2,trs] = trainbr(nets2, P, T);
Yns=sim(nets2,P);

%Построение графика функций
figure('NumberTitle','off','Name','Функция ','ToolBar','none','MenuBar','none');
plot(P,T,'--k','LineWidth',1);hold on;
A=nets2(P);
```

```

Yns=sim(nets2,P);
plot(P,Yns,'--k','LineWidth',1);hold on;
E=T-A;
SSE=sumsq(E)
MSE=mse(E)

```

В результате работы получена следующая модель ИНС

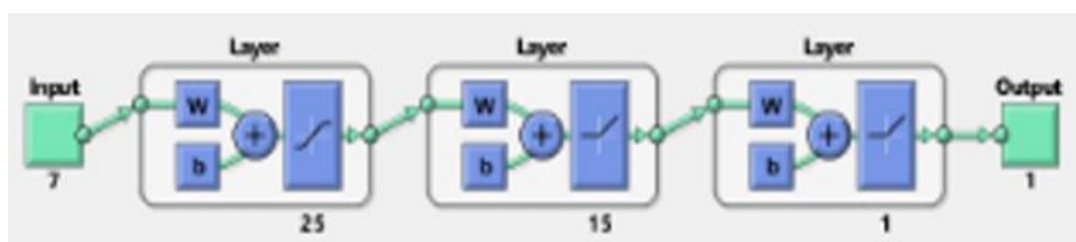


Рисунок 38 – Модель нейронной сети прямого распространения

Результаты работы (оценка MSE – минимальная среднеквадратичная ошибка обучения) в зависимости от метода обучения приведены в таблице 7.

Таблица 7 – Ошибка обучения

Структура сети	Метод обучения Левенберга Маркварда (trainlm)	Метод обучения с использованием Байесовской регуляризации (trainbr)	Метод обучения с использованием градиентного спуска (traingd)
5,1	MSE= 0.000388 (1000 эпох обучения)	MSE= 0.000239 (223 эпохи обучения)	MSE= 0.0131 (116 эпох обучения)
15,5,1	MSE=0.00567 (37 эпох обучения)	MSE=0.00180 (1000 эпох обучения)	MSE= 0.0131 (1000 эпох обучения)
25,15,1	MSE= 0.0189 (61 эпоха обучения)	MSE= 0.00947 (339 эпох обучения)	MSE= 0.0131 (27 эпох обучения)
35,25,1	MSE= 0.0199 (94 эпохи обучения)	MSE= 0.00268 (1000 эпох обучения)	MSE= 28.6 (22 эпохи обучения)

После обучения сети построен совмещенный график работы нейронной сети прямого распространения и эталонных значений обучающей выборки.

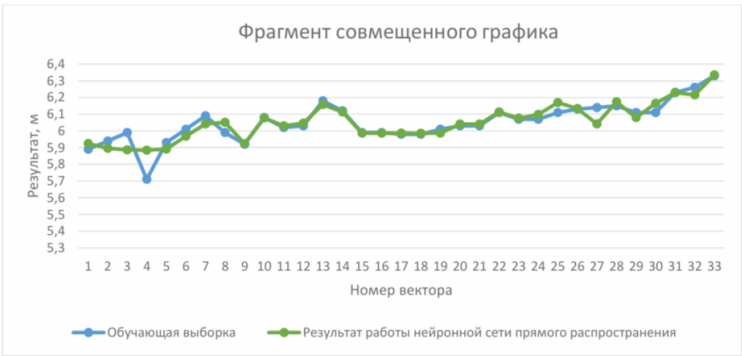


Рисунок 39 – Фрагмент совмещенного графика

Нейронная сеть прямого распространения демонстрирует меньшую MSE, чем любой из экспериментов с линейной нейронной сетью.

Задание

Построить несколько экземпляров нейронной сети прямого распространения. Обучить на основе выборки из задания 1. Привести исходный код, заполнить таблицу.

Структура сети	Метод обучения 1	Метод обучения 2	Метод обучения 3
?, ?	MSE	MSE	MSE
...
?, ?	MSE	MSE	MSE

Дополнительное задание №2 (выполняется по согласованию с преподавателем):

Третья часть задания заключается в выполнении аналогичных действий (обучении сетей на аналогичной обучающей выборке, см. задание 1) для двух простых моделей нейронных сетей: линейной нейронной сети и нейронной сети прямого распространения. Использовать специализированные библиотеки для языка Python.

Отчет включает в себя краткие теоретические сведения, исходный код, заполненную таблицу, для сравнения разных экземпляров сетей (параметры для включения в таблицу выбираются самостоятельно, обязателен параметр выхода сети, и оценки ошибки/точности), выводы.

Пример №4

```
1  import numpy as np
2
3  # Определение класса для линейной нейронной сети
4  class LinearNeuralNetwork:
5      def __init__(self, input_size, output_size):
6          # Инициализация весов случайными значениями
7          self.weights = np.random.rand(input_size, output_size)
8
9      def forward(self, X):
10         # Прямой проход: умножение входных данных на веса
11         return np.dot(X, self.weights)
12
13 # Пример использования
14 if __name__ == "__main__":
15     # Размерность входа и выхода
16     input_size = 3
17     output_size = 1
18
19     # Создание экземпляра модели
20     model = LinearNeuralNetwork(input_size, output_size)
21
22     # Входные данные (пример)
23     inputs = np.array([[1, 2, 3], [4, 5, 6]])
24
25     # Прогон прямого прохода
26     outputs = model.forward(inputs)
27
28     print("Выходные значения:", outputs)
```

Класс LinearNeuralNetwork: содержит инициализацию весов (weights) и метод прямого прохода (forward), который вычисляет выход путем матричного умножения входных данных на веса.

Создается экземпляр модели с заданной размерностью ввода-вывода, задаются входные данные, выполняется прямой проход и выводятся полученные выходные значения.

Пример №5

```
import torch

import torch.nn as nn

from torch.utils.data import DataLoader, TensorDataset

import numpy as np

# Генерация случайных данных для примера
np.random.seed(42)
X = np.linspace(-10, 10, 100).reshape(-1, 1) # Входные данные
y = X * 2 + 1 + np.random.normal(scale=0.5, size=(100, 1)) # Выходные данные с шумом

# Преобразование данных в PyTorch-тензоры
X_tensor = torch.tensor(X, dtype=torch.float32)
y_tensor = torch.tensor(y, dtype=torch.float32)

# Создание датасета и загрузчика данных
dataset = TensorDataset(X_tensor, y_tensor)
dataloader = DataLoader(dataset, batch_size=8, shuffle=True)

# Определение структуры линейной нейронной сети
class LinearModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear_layer = nn.Linear(in_features=1, out_features=1)

    def forward(self, x):
        return self.linear_layer(x)
```

```

model = LinearModel()

criterion = nn.MSELoss() # Функция потерь - среднеквадратичная ошибка

optimizer = torch.optim.SGD(model.parameters(), lr=0.01) # Оптимизатор градиентного
спуска

# Обучение модели

epochs = 100

for epoch in range(epochs):

    for inputs, targets in dataloader:

        optimizer.zero_grad()

        outputs = model(inputs)

        loss = criterion(outputs, targets)

        loss.backward()

        optimizer.step()

    if epoch % 10 == 0:

        print(f'Epoch {epoch}, Loss: {loss.item():.4f}')

# Тестирование модели

test_input = torch.tensor([[5]], dtype=torch.float32)

predicted_output = model(test_input)

print(f'Прогнозируемое значение для ввода 5: {predicted_output.item():.2f}')

```

Создаем искусственные данные для демонстрации работы сети. Это простая задача регрессии вида $y=2x+1$ с добавлением шума.

Преобразуем наши массивы NumPy в объекты типа Tensor библиотеки PyTorch, поскольку эта библиотека работает именно с такими объектами.

Создаем объект класса DataLoader, который автоматически загружает мини-пакеты данных (batch).

Реализована простая однослойная линейная сеть (nn.Linear), принимающая одну переменную на входе и выдающая одну переменную на выходе.

Используем метод обратного распространения ошибок (backpropagation) и оптимизацию методом стохастического градиентного спуска (SGD) для минимизации средней квадратичной ошибки.

Проверяем работу модели на новых данных.

Задание

Создать экземпляры сетей на разных размерностях. Результаты выходных значений зафиксировать в таблице. Допускаются различные модификации сети.

Пример №6

```
1  import tensorflow as tf
2  from tensorflow.keras import layers, models
3
4  # Загрузка датасета MNIST
5  mnist = tf.keras.datasets.mnist
6  (x_train, y_train), (x_test, y_test) = mnist.load_data()
7
8  # Нормализация данных (приведение значений пикселей к диапазону [0, 1])
9  x_train, x_test = x_train / 255.0, x_test / 255.0
10
11 # Создание модели FNN
12 model = models.Sequential([
13     # Флэттенг слоя для преобразования двумерных изображений в одномерный вектор
14     layers.Flatten(input_shape=(28, 28)),
15
16     # Полносвязный слой с 128 нейронами и функцией активации ReLU
17     layers.Dense(128, activation='relu'),
18
19     # Выходной слой с softmax активацией для многоклассовой классификации
20     layers.Dense(10, activation='softmax')
21 ])
22
23 # Компилирование модели
24 model.compile(optimizer='adam',
25               loss='sparse_categorical_crossentropy', # Функция потерь для категориальной классификации
26               metrics=['accuracy'])
27
28 # Обучение модели
29 history = model.fit(x_train, y_train, epochs=5)
30
31 # Оценка точности модели на тестовом наборе
32 test_loss, test_acc = model.evaluate(x_test, y_test)
33 print(f"Точность на тестовых данных: {test_acc*100:.2f}%")
```

Используется встроенный датасет MNIST из библиотеки TensorFlow.

Преобразование пиксельных значений от 0 до 255 в диапазон от 0 до 1 для ускорения обучения.

- Первый слой (layers.Flatten) преобразует изображение размером 28×28 пикселей в одномерный массив длиной 784.
- Второй слой (Dense) — полносвязный слой с 128 нейронами и функцией активации ReLU.
- Третий слой (Dense) — выходной слой с 10 нейронами (по количеству классов) и функцией активации SoftMax для вывода вероятностей принадлежности каждого класса.

Используются оптимизатор Adam, функция потерь sparse categorical cross entropy и метрика accuracy.

Модель учится на тренировочных данных в течение пяти эпох.

Проверяется точность модели на тестовом наборе данных.

Пример №7

Создадим простую зависимость вида $y=ax^2+bx+c$.

```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt

# Случайные значения X
X = np.linspace(-10, 10, 100).reshape(-1, 1)
Y = 0.5 * X**2 + 2*X + 1 + np.random.randn(*X.shape)*2 # Добавляем небольшой шум

# Преобразовываем в PyTorch-тензоры
X_torch = torch.from_numpy(X.astype(np.float32))
Y_torch = torch.from_numpy(Y.astype(np.float32)).view(-1, 1)

plt.scatter(X, Y)
plt.title('Исходные данные')
```

```
plt.show()
```

Создаём класс, наследуя модуль `nn.Module`. Архитектуру выберем следующим образом:

- Входной слой: 1 нейрон (одна переменная).
- Скрытый слой: 10 нейронов с функцией активации ReLU.
- Выходной слой: 1 нейрон (регрессия).

```
class FeedForwardNN(nn.Module):  
    def __init__(self, input_dim, hidden_dim, output_dim):  
        super(FeedForwardNN, self).__init__()  
        self.fc1 = nn.Linear(input_dim, hidden_dim)  
        self.relu = nn.ReLU()  
        self.fc2 = nn.Linear(hidden_dim, output_dim)  
  
    def forward(self, x):  
        out = self.fc1(x)  
        out = self.relu(out)  
        out = self.fc2(out)  
        return out
```

цикл обучения модели

```
input_dim = 1 # Размерность входа  
hidden_dim = 10 # Количество скрытых нейронов  
output_dim = 1 # Регрессия (один выход)  
  
model = FeedForwardNN(input_dim, hidden_dim, output_dim)  
criterion = nn.MSELoss() # Критерий оценки MSE  
optimizer = optim.Adam(model.parameters(), lr=0.01) # Оптимизатор Adam  
  
num_epochs = 500  
losses = []
```

```
for epoch in range(num_epochs):
    optimizer.zero_grad()
    predictions = model(X_torch)
    loss = criterion(predictions, Y_torch)
    losses.append(loss.item())
    loss.backward()
    optimizer.step()

    if (epoch+1)%50==0:
        print(f'Эпоха [{epoch+1}/{num_epochs}], Потеря: {loss.item():.4f}')
```

Посмотрим график потерь и визуализируем предсказания нашей модели

```
plt.plot(range(len(losses)), losses)
plt.xlabel("Эпоха")
plt.ylabel("Потеря")
plt.title("График изменения потерь")
plt.show()

with torch.no_grad():
    predicted_values = model(X_torch)

plt.figure(figsize=(10, 6))
plt.scatter(X, Y, label='Истинные значения', color="blue")
plt.plot(X, predicted_values.numpy(), 'r-', lw=3, label='Предсказанные значения')
plt.legend()
plt.title('Результат моделирования')
plt.show()
```

Программа демонстрирует базовую реализацию нейронной сети прямого распространения для задачи регрессии

Задание

Создать экземпляры сетей на разных размерностях. Результаты выходных значений зафиксировать в таблице. Допускаются различные модификации сети, в том числе различные активационные функции и функции потерь.

Лабораторная работа №3

Разработка программного эмулятора работы Марковской цепи в качестве инструмента моделирования развития ситуации

Цель: освоение технологии применения Марковских цепей для моделирования развития ситуаций и решения задачи прогнозирования.

Этапы работы:

- Постановка задачи. Ситуация или процесс описывается студентом самостоятельно
- Теоретическое моделирование. Описывается Марковская цепь, моделирующая процесс.
- Написание программы на одном из языков программирования, эмулирующую работу Марковской цепи.
- Демонстрация полученной программы преподавателю.
- Интерфейс эмулятора формируется произвольно. Основной критерий – понятность демонстрации и изложения, а так же соответствие задаче.
- Формирование отчета.

Отчет должен содержать: цель, постановку задачи, описание ситуации, моделирование ситуации с применением цепи Маркова, описание конечного автомата и матрицы вероятностей переходов, исходный код программной эмуляции, экранные формы работы программной эмуляции, выводы.

Марковские цепи представляют собой математический инструмент, эффективно используемый в множестве сфер, включая машинное обучение, создание текстов и музыкальных композиций, шифрование данных и прочие направления. Данная концепция была предложена российским учёным Андреем Марковым в первые десятилетия XX столетия и впоследствии стала востребованной в разнообразнейших дисциплинах.

Андрей Андреевич Марков являлся видным представителем отечественной науки, специализировавшимся на изучении случайных процессов, численных методов и статистического анализа. Его наиболее значимым вкладом стало создание знаменитой математической структуры, известной ныне как «цепи Маркова».

Сам термин появился впервые в научной работе Андрея Маркова в 1906 году под названием «О предельном распределении одного класса зависимых

случайных величин», где исследовался статистический характер последовательностей связанных друг с другом случайных явлений.

Основное свойство этой модели состоит в возможности вычислять вероятности перехода системы из текущего состояния в следующее исключительно исходя из текущего положения. Следовательно, используя принципы Марковской цепи, можно создавать разнообразные последовательности объектов — будь то слова текста, музыкальные композиции, фрагменты программного кода и многое другое.

Одним из ярких примеров использования Марковских цепей служит область компьютерного моделирования и кинематографа, где они применяются для создания правдоподобных анимаций и визуальных эффектов. Не менее важной областью применения являются технологии обработки естественных языков и автоматическая генерация текстов.

Конечный автомат (КА) представляет собой математическую структуру, включающую фиксированное количество состояний, набор возможных входных сигналов, правила перехода между этими состояниями и определенное число выходных значений. Эта модель активно применяется в компьютерных науках для описания поведения отдельных элементов либо целых систем, отслеживая изменения их состояний.

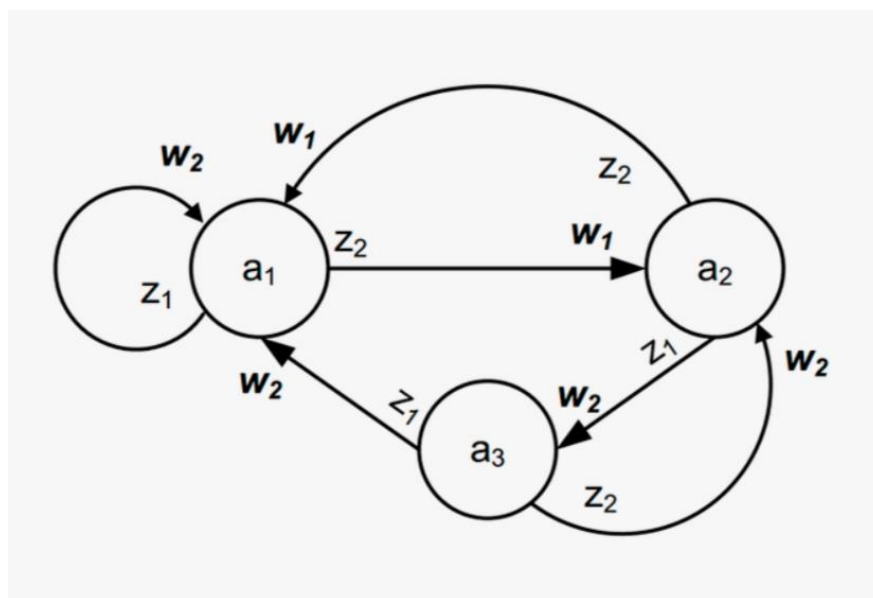


Рисунок 40 – Пример графа конечного автомата

В теории Марковский цепей КА реализует последовательность состояний.

Случайный процесс — это математический инструмент, применяемый для описания неопределенных событий. Типичными примерами служат случайное движение частицы и пуассоновский поток событий. Цепи Маркова относятся к числу случайных процессов, отличительной чертой которых является зависимость вероятности смены состояния лишь от текущего положения системы.

Стационарное распределение описывает равновесное состояние системы спустя неограниченно долгое время. Оно отражает тот факт, что вероятности перемещений перестают меняться, и система приходит к устойчивому состоянию. По стационарному распределению можно судить о доле времени, которую система проводит в каждом возможном состоянии.

Переходная вероятность характеризует шанс перемещения системы из начального состояния в другое за определенный промежуток времени. Вероятность такого события задаётся однозначно положением системы в рассматриваемый момент времени.

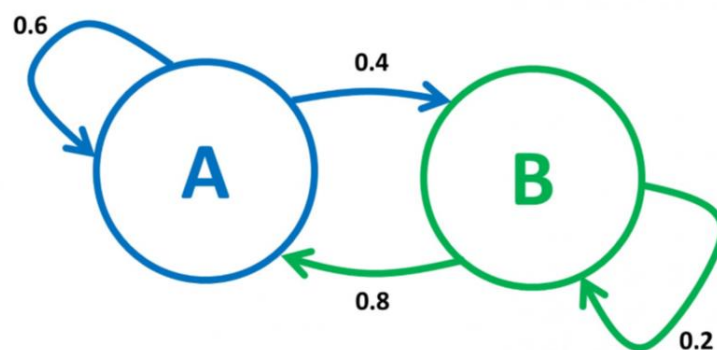


Рисунок 41 – Граф с вероятностью перехода (изображены на дугах, допускается не размещать вероятности на дугах а указывать только в виде матрицы)

Вероятности переходов между состояниями представляются в виде матрицы переходных вероятностей. Элементы этой матрицы показывают вероятность перехода из состояния i в состояние j за один шаг.

Обозначим:

- P_{ij} — вероятность перехода из состояния i в состояние j .
- $S = \{s_1, s_2, \dots, s_n\}$ — множество всех возможных состояний системы.

Матрица переходных вероятностей записывается следующим образом:

$$\begin{bmatrix} P_{11} & P_{12} & \dots & P_{1n} \\ P_{21} & P_{22} & \dots & P_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{n1} & P_{n2} & \dots & P_{nn} \end{bmatrix}$$

Рисунок 42 – Матрица переходных состояний

Каждая строка матрицы соответствует исходящему состоянию, каждый столбец — приходящему состоянию. Например, элемент P_{23} показывает вероятность перехода из второго состояния (s_2) в третье состояние (s_3).

Два свойства матриц переходных вероятностей: нормировка (сумма вероятностей переходов (смотреть по строкам) всегда будет равна 1, т.к. система должна куда то перейти) и ненегативность (каждая вероятность неотрицательна).

Допустим, мы имеем простую Марковскую цепь с тремя состояниями $\{A, B, C\}$, и заданную матрицу переходных вероятностей:

$$P = \begin{pmatrix} 0.2 & 0.5 & 0.3 \\ 0.8 & 0.1 & 0.1 \\ 0.4 & 0.3 & 0.3 \end{pmatrix}$$

Рисунок 43 – Матрица переходов вероятностей

Здесь, например, вероятность перехода из состояния А в В равна $P_{AB}=0.5$, из В в С — $P_{BC}=0.1$. Эти значения непосредственно считываются из соответствующих позиций матрицы.

Марковская цепь — это такая последовательность состояний, в которой вероятность следующего шага определяется исключительно текущим состоянием системы, игнорируя всю предшествующую историю изменений.

Такой подход удобен для моделирования ситуаций, где будущее развитие системы целиком обусловлено настоящим моментом, без учета прошлых этапов развития.

Основное качество Марковских цепей заключается в том, что вероятность перехода из одного состояния в другое определяется исключительно текущим состоянием, не учитывая предыдущие шаги. Другими словами, влияние прошлого на будущее отсутствует, и вероятности остаются постоянными для каждого текущего состояния. Благодаря данному свойству Марковские цепи широко применяют для моделирования процессов, представленных в виде серии случайных событий.

Другое значительное свойство Марковских цепей состоит в наличии единственного устойчивого распределения вероятностей. Независимо от начальных условий, любая Марковская цепь стремится достичь именно этого стабильного состояния, известного как стационарное распределение.

Третье свойство Марковских цепей называется свойством достижимости. Оно утверждает, что из любого состояния возможно попасть в любое другое состояние за некоторое конечное количество шагов. Наличие этого свойства важно для гарантии того, что построенная модель способна учитывать возможные переходы между любыми парами состояний, что обеспечивает точность и полноту описания изучаемого процесса.

Использование Марковских моделей также сопряжено с определёнными недостатками и трудностями, которые могут снижать эффективность их применения.

Одна из проблем связана с ограниченными возможностями построения моделей. Несмотря на свою полезность, Марковские цепи не способны учесть абсолютно все детали реальных процессов. Они предполагают, что вероятности переходов между состояниями неизменны и не зависят от начальной точки системы, что не всегда, верно, для сложных систем.

Ещё одним важным аспектом является ограничение сферы применимости Марковских моделей. Некоторые реальные процессы обладают сложной структурой зависимости между событиями, которую невозможно адекватно описать с помощью простых Марковских цепей. Помимо этого, результаты моделирования могут оказаться нестабильными, если вероятности переходов существенно различаются в зависимости от стартового состояния.

Иногда Марковские модели приводят к неверным прогнозам и ошибочным выводам. Если вероятность перехода между состояниями оказывается чрезмерно завышенной или заниженной, итоговые прогнозы будут некорректны. Игнорирование важных состояний системы также

способно вызывать неоправданную уверенность в точности полученных результатов.

Пример №1 (см. список дополнительной литературы)

Представьте себе небольшой сайт, состоящий всего из семи страниц, которым присвоены номера от 1 до 7. Структура гиперссылок между этими страницами представлена следующим графом.

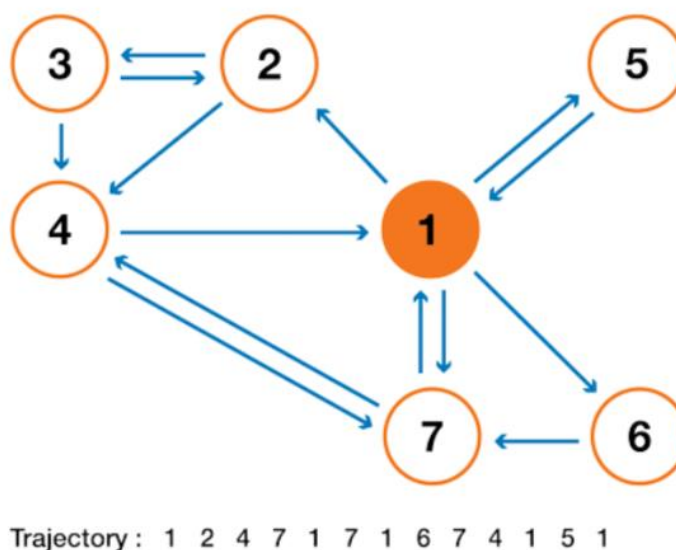


Рисунок 44 – Граф переходов

Ради простоты восприятия в приведённой выше анимации не отображены конкретные вероятности переходов. Тем не менее, предполагается, что навигация осуществляется случайно («случайное блуждание»). Исходя из этого принципа, можно восстановить вероятности переходов, следуя такому простому правилу: для вершины с K исходящими связями (страницы с K ссылками на другие страницы) вероятность перехода по каждой ссылке составляет ровно $1/K$. Соответственно, переходная матрица вероятностей выглядит следующим образом.

$$p = \begin{pmatrix} . & 0.25 & . & . & 0.25 & 0.25 & 0.25 \\ . & . & 0.5 & 0.5 & . & . & . \\ . & 0.5 & . & 0.5 & . & . & . \\ 0.5 & . & . & . & . & . & 0.5 \\ 1.0 & . & . & . & . & . & . \\ . & . & . & . & . & . & 1.0 \\ 0.5 & . & . & 0.5 & . & . & . \end{pmatrix}$$

Рисунок 45– Пример матрицы переходов

Заменяя нули на точки (".") для большей наглядности, перед началом дальнейших расчётов заметим, что данная Марковская цепь обладает свойствами неразложимости и апериодичности, что гарантирует существование стационарного распределения в долговременной перспективе. Ранее было показано, что данное стационарное распределение можно определить путём решения следующей левой собственной задачи векторного уравнения.

$$\pi = \pi p \quad \text{with} \quad \pi = (\pi_1 \quad \pi_2 \quad \pi_3 \quad \pi_4 \quad \pi_5 \quad \pi_6 \quad \pi_7)$$

Сделав так, мы получим следующие значения PageRank (значения стационарного распределения) для каждой страницы.

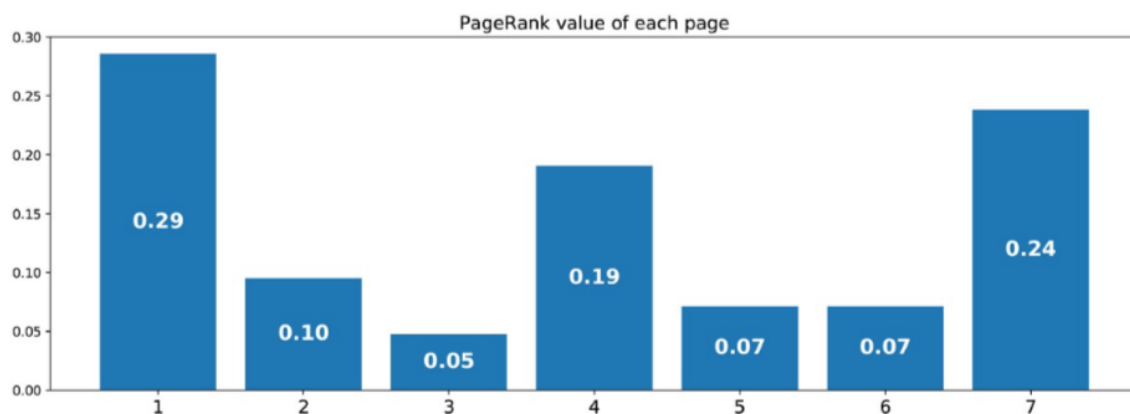


Рисунок 46– Ранжирование веб-сайта

Значения PageRank, вычисленные для нашего искусственного примера из 7 страниц. Тогда ранжирование PageRank этого крошечного веб-сайта имеет вид $1 > 7 > 4 > 2 > 5 = 6 > 3$.

Пример №2 (см. список дополнительной литературы)

У нас есть город, где может быть четыре состояния погоды: ясно, облачно, дождь и гроза. Состояние погоды на завтра зависит только от текущего состояния.

Если сегодня было «ясно», то завтра с вероятностью 0,6 будет «ясно», с вероятностью 0,2 будет «облачно», с вероятностью 0,1 — «дождь», и с вероятностью 0,1 — «гроза».

Если сегодня было «облачно», то завтра с вероятностью 0,1 будет «ясно», с вероятностью 0,5 будет «облачно», с вероятностью 0,2 — «дождь», и с вероятностью 0,2 — «гроза».

Если сегодня был «дождь», то завтра с вероятностью 0,3 будет «ясно», с вероятностью 0,3 будет «облачно», с вероятностью 0,2 — «дождь», и с вероятностью 0,2 — «гроза».

Если сегодня была «гроза», то завтра с вероятностью 0,4 будет «ясно», с вероятностью 0,3 будет «облачно», с вероятностью 0,1 — «дождь», и с вероятностью 0,2 — «гроза».

Мы хотим предсказать состояние погоды на завтра, опираясь на текущую погоду.

Исходный код на языке Python приведен ниже:

```
import numpy as np #numpy - мощнейшая библиотека для реализации Марковской цепи

# задаем вероятности переходов
transition_probabilities = [
    [0.6, 0.2, 0.1, 0.1], # из "ясно"
    [0.1, 0.5, 0.2, 0.2], # из "облачно"
    [0.3, 0.3, 0.2, 0.2], # из "дождь"
    [0.4, 0.3, 0.1, 0.2], # из "гроза"
]
```

```

# создаем массив состояний
states = ["ясно", "облачно", "дождь", "гроза"]

def predict_weather(today, days=1):
    # находим текущее состояние погоды
    current_state = states.index(today)

    # создаем вектор начальных вероятностей
    current_probabilities = np.zeros(len(states))
    current_probabilities[current_state] = 1

    # прогнозируем погоду на несколько дней
    for i in range(days):
        current_probabilities = np.dot(current_probabilities, transition_probabilities)

    # найдем индекс максимального элемента вектора вероятностей - это и будет предсказанное
    # состояние погоды
    predicted_state_index = np.argmax(current_probabilities)

    # вернем предсказанное состояние погоды
    return states[predicted_state_index]

# пример использования функции - предсказание погоды на завтра, если сегодня "ясно"
print(predict_weather("ясно")) # результат: "ясно"

```

В этом коде мы определяем матрицу вероятностей переходов (transitionprobabilities) и массив состояний (states).

Функция predictweather() принимает в качестве аргументов текущее состояние погоды today и количество дней days, на которые нужно прогнозировать погоду. Функция возвращает предсказанное состояние погоды на days дней вперед от today.

Алгоритм работы функции заключается в следующем:

- 1) Определяем текущее состояние погоды (currentstate) на основе значения today.

2) Создаем вектор начальных вероятностей (`currentprobabilities`), в котором значение 1 на месте текущего состояния, а остальные элементы равны 0.

3) Производим умножение `currentprobabilities` на матрицу `transitionprobabilities` столько раз, сколько указано в `days`. Результат этого умножения будет являться вектором вероятностей для `days` дней вперед.

4) Находим индекс максимального элемента полученного вектора вероятностей и возвращаем соответствующее значение из массива состояний — это и будет предсказанное состояние погоды.

Пример №3 (см. список дополнительной литературы)

Предположим, что мы проводим эксперимент с монеткой, которая может выпасть на орла или решку. При каждом броске монетки мы записываем результат броска (0 — орёл, 1 — решка) и переходим к следующему броску. Мы хотим создать марковскую цепь, которая будет предсказывать вероятность выпадения орла или решки в следующем броске, в зависимости от количества орлов и решек в прошлых бросках.

Сначала создадим список всех возможных состояний марковской цепи. Каждое состояние будет представлять собой строку из двух символов, где первый символ обозначает количество орлов, а второй символ — количество решек в прошлых бросках. Таким образом, список состояний будет выглядеть следующим образом:

```
states = ['00', '01', '10', '11']
```

Теперь нужно создать матрицу переходных вероятностей. Матрица будет иметь размерность 4x4 (так как у нас 4 состояния), где каждый элемент матрицы будет представлять собой вероятность перехода из одного состояния в другое. Для простоты будем считать, что вероятность выпадения орла или решки на каждом броске равна 0.5.

```
transition_matrix = [  
    [0.5, 0.5, 0, 0],  
    [0, 0, 0.5, 0.5],  
    [0.5, 0.5, 0, 0],  
    [0, 0, 0.5, 0.5]  
]
```

Элемент `transition_matrix[0][1]` обозначает вероятность перехода из состояния '00' (0 орлов и 0 решек) в состояние '01' (0 орлов и 1 решка) и равен 0.5.

Осталось только написать функцию для нахождения вероятности выпадения орла или решки в следующем броске, в зависимости от состояния в текущем броске и матрицы переходных вероятностей.

```
import numpy as np

def predict_next_state(current_state, transition_matrix):
    # Находим индекс текущего состояния в списке состояний
    current_index = states.index(current_state)

    # Вычисляем вероятности перехода из текущего состояния в каждое другое состояние
    probabilities = transition_matrix[current_index]

    # Выбираем только те вероятности, которые соответствуют выбранному символу (0 - орел, 1 - решка)
    next_probabilities = []
    for i in range(len(states)):
        if states[i][0] == current_state[0]:
            next_probabilities.append(probabilities[i])

    # Нормируем вероятности и выбираем наивысшую
    next_probabilities = np.array(next_probabilities)
    next_probabilities /= sum(next_probabilities)

    # Выбираем наивысшую вероятность и возвращаем соответствующий символ (0 - орел, 1 - решка)
    if np.random.choice([0, 1], p=next_probabilities) == 0:
        return '0'
    else:
        return '1'
```

Например, вызов функции `predict_next_state('01', transition_matrix)` может вернуть символ '0' с вероятностью 0.5 или символ '1' с вероятностью 0.5, что соответствует вероятностям перехода из состояния '01' в состояния '00' и '01' в соответствующей матрице переходных вероятностей.

Пример №4

Простой пример реализации Марковской цепи (без привязки к ситуации) на языке Python.

```
import numpy as np

# Матрица переходных вероятностей
transition_matrix = np.array([
    [0.5, 0.5, 0],
    [0.3, 0.2, 0.5],
    [0, 0.7, 0.3]
])

# Начальное распределение (вероятности нахождения в каждом состоянии)
initial_state = np.array([1, 0, 0]) # Начинаем с первого состояния

def markov_chain_simulation(steps):
    current_state = initial_state.copy()

    for _ in range(steps):
        next_state = np.dot(current_state, transition_matrix)
        print(f"Шаг {_+1}: {next_state}")
        current_state = next_state

markov_chain_simulation(10)
```

Матрица переходных вероятностей (transition_matrix): Каждая ячейка [i][j] этой матрицы определяет вероятность перехода из состояния i в состояние j. Например, значение transition_matrix[0][1] = 0.5 означает, что из состояния 0 в состояние 1 система переходит с вероятностью 0.5.

Начальное распределение (initial_state): Здесь указано начальное состояние системы. В данном примере мы начинаем с первого состояния (первая строчка массива равна 1, остальные элементы — 0). Это значит, что изначально вся вероятность сосредоточена в первом состоянии.

Функция симуляции (markov_chain_simulation): Функция последовательно рассчитывает вероятности пребывания системы в каждом из

состояний на каждом шаге. Для этого используется операция умножения текущего состояния на матрицу переходных вероятностей.

При запуске программы, вы увидите, как меняется распределение вероятностей на каждом шаге. Через какое-то количество итераций распределение начнёт приближаться к своему стационарному значению.

Задание. Описать ситуацию. Произвести ее моделирование и эмуляцию в соответствии с этапами задачи, применяя механизм Марковских цепей.

Лабораторная работа №4

Исследование генетического алгоритма

Цель: изучение принципов работы генетического алгоритма и анализ влияния параметров генетического алгоритма на качество его работы. Работа выполняется с помощью программы Genetic.exe, разработанной на кафедре ЭВМ.

Вариант задания выдаётся студенту преподавателем.

Таблица - Варианты задания

Номер варианта	Вид функции	Диапазон изменения параметров	Экстремум
1	$F(x,y)=-x \cdot e^{-x^2-y^2}$	$x[0;10]$ $y[0;10]$	max
2	$F(x,y)=-x \cdot e^{-x^2-y^2}$	$x[0;10]$ $y[0;10]$	min
3	$F(x,y)=-x^2 \cdot e^{1-x^2-20 \cdot (x-y)^2}$	$x[0,01;0,5]$ $y[1;2]$	max
4	$F(x,y)=-x^2 \cdot e^{1-x^2-20 \cdot (x-y)^2}$	$x[0;10]$ $y[0;10]$	min
5	$F(x,y)=-x^2+0,5 \cdot y+(1-1,5 \cdot x)^2+(1-2 \cdot y)^2$	$x[0;10]$ $y[0;10]$	max
6	$F(x,y)=-x^2+0,5 \cdot y+(1-1,5 \cdot x)^2+(1-2 \cdot y)^2$	$x[0;10]$ $y[0;10]$	min
7	$F(x,y)=(x-2)^2+(y-5)^2+(z+2)^2-16-x+y-z$	$x[0;10]$ $y[0;10]$ $z[0;5]$	max
8	$F(x,y)=(x-2)^2+(y-5)^2+(z+2)^2-16-x+y-z$	$x[0;10]$ $y[0;10]$	min

		z[0;5]	
9	$F(x,y)=-x^2 - y^2 + 10 \cdot \cos(x) + 10 \cdot \sin(y)$	x[0;10] y[0;10]	max
10	$F(x,y)=-x^2 - y^2 + 10 \cdot \cos(x) + 10 \cdot \sin(y)$	x[0;10] y[0;10]	Min
11	$F(x,y)= x^2 + y^2 + 40 \cdot \sin(x) \cdot \sin(y)$	x[0;10] y[0;10]	max
12	$F(x,y)= x^2 + y^2 + 40 \cdot \sin(x) \cdot \sin(y)$	x[0;10] y[0;10]	Min
13	$F(x,y)= e^{(x+y)} / e^{(x^2+y^2)}$	x[0;2] y[0;2]	max
14	$F(x,y)= e^{(x+y)} / e^{(x^2+y^2)}$	x[0;2] y[0;2]	Min
15	$F(x,y)= (y+1)^2 - x^2 + y^2 + (z-1)^2 + z^2 - (x+1)^2$	x[0;10] y[0;10] z[0;5]	max
16	$F(x,y)= (y+1)^2 - x^2 + y^2 + (z-1)^2 + z^2 - (x+1)^2$	x[0;10] y[0;10] z[0;5]	Min
17	$F(x,y)= (x-1)^2 + (y+2)^2 + (z-2)^2 + (t+1)^2 + z - y + x - t$	x[0;10] y[0;10] z[0;5] t[0;10]	max
18	$F(x,y)= (x-1)^2 + (y+2)^2 + (z-2)^2 + (t+1)^2 + z - y + x - t$	x[0;10] y[0;10] z[0;5] t[0;10]	Min

19	$F(x_1, x_2, x_3, x_4, x_5, x_6) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2$	[-10; +10]	Min
20	$F(x_1, x_2, x_3) = x_1^2 + x_2^2 +$	[-5; +5]	Min
21	$F(x_1, x_2) = 20 + x_1 * x_1 + x_2 * x_2 - 10 * \cos(x_1) - 10 * \cos(x_2)$	[-10; +10]	max

Задание включает две задачи:

- исследование генетического алгоритма в задаче оптимизации многопараметрической функции;
- решение задачи коммивояжера для 6 городов. Матрицу расстояний (6×6) для задачи коммивояжера заполнить самостоятельно с нулевой главной диагональю.

Этапы работы:

- Найти экстремум функции в соответствии с заданием и решить задачу коммивояжера размерностью 6*6. Стоимость переезда между городами в данной матрице задать самостоятельно.

- Запустить программу Genetic.exe.

- Диапазон изменения параметра X для исследуемой функции устанавливается студентом в окне НАСТРОЙКИ/ ПАРАМЕТРЫ.

- Выполнить эксперименты (для различных параметров), приведенных в таблице, для четырёх критериев останова генетического алгоритма:

- амплитуда колебаний среднего значения;
- максимум равен среднему значению;
- стабилизация максимума;
- стабилизация среднего значения.

- Выбрать эффективный метод останова алгоритма и дальнейшие исследования проводить с выбранным методом останова генетического алгоритма.

Таблица 8 – Исследование одноточенного и двух точечного кроссовера

Эксперимент	Оператор отбора	Элитизм (%)	Размер популяции	Коэффициент размножения	Вероятность инверсии	Вероятность перестановки	Вероятность редукции, %	Количество поколений	Значение функции
1	Рулетка	0	5	70	10	10	60		
2	Рулетка	5	10	70	10	10	70		
3	Рулетка	10	20	80	20	20	80		
4	Рулетка	20	30	80	20	20	90		
5	Турнирный отбор	0	5	70	10	10	60		
6	Турнирный отбор	5	10	80	20	20	70		
7	Турнирный отбор	10	20	80	30	30	80		
8	Турнирный отбор	20	30	90	40	30	90		

По результатам исследования составить отчет, в котором привести:

- задание с видом исследуемой функции и матрицу расстояний для задачи коммивояжера;
- таблицу с экспериментами и результатами поиска экстремума для каждого из них (количество поколений и значение функции, при которых зафиксирован останов алгоритма);
- проверить полученные данные нахождения экстремума с помощью других программных средств (Excel, MathCAD и другие);
- указать критерий останова генетического алгоритма, выбранный для решения задачи оптимизации;
- для задачи коммивояжера привести оптимальный путь коммивояжера и вычисленную целевую функцию и данные по скрещиванию и формированию двух поколений;
- выполнить сравнительный анализ и сделать выводы по результатам поиска экстремума для заданной функции (какие параметры генетического

алгоритма обеспечивают высокую производительность и точность приближения к эталонному экстремуму функции).

Таблица 9 – Результат исследования

Эксперимент	Оператор отбора	Элитизм (%)	Размер популяции	Коэффициент размножения	Вероятность инверсии	Вероятность перестановки	Вероятность редукции, %	Количество поколений	Значение функции
1	Рулетка	0	5	70	10	10	60		
2	Рулетка	5	10	70	10	10	70		
3	Рулетка	10	20	80	20	20	80		
4	Рулетка	20	30	80	20	20	90		
5	Турнирный отбор	0	5	70	10	10	60		
6	Турнирный отбор	5	10	80	20	20	70		
7	Турнирный отбор	10	20	80	30	30	80		
8	Турнирный отбор	20	30	90	40	30	90		

Описание работы с установкой.

Выполнить настройки в соответствии с рисунком. Для этого в меню Задача выбрать команду настройки. При решении задачи коммивояжера сначала установить значение оператора кроссовера - изошренный, а затем на вкладке ПАРАМЕТРЫ щелкнуть по кнопке МАТРИЦА и ввести данные матрицы.

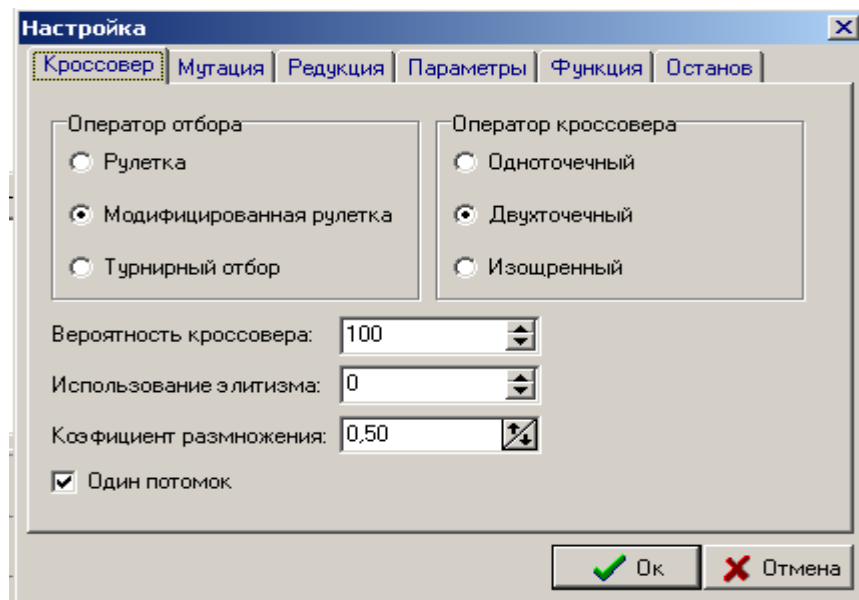


Рисунок 47 – Окно настроек

- выбрать одноточечный или двухточечный кроссовер;
- вероятность (например, 90%);
- Сколько элитных особей перемещать в следующее поколение;
- вероятность инверсии или перестановки (вкладка МУТАЦИЯ);
- размер популяции и вид поиска минимум или максимум (вкладка РЕДУКЦИЯ);
- типы аргументов создаваемой функции (вкладка ПАРАМЕТРЫ);
- введите функцию для анализа по ГА (вкладка ФУНКЦИЯ);
- . Для этого в поле ИМЯ ФУНКЦИИ введите, например, Func1, а в поле ФУНКЦИЯ. – вид функции, например, $X^2 - 5X + 2$. Если требуется проверить, щелкните по кнопке ПРОВЕРИТЬ. После завершения – кнопка ОК;
- Задайте критерий останова (вкладка ОСТАНОВ).

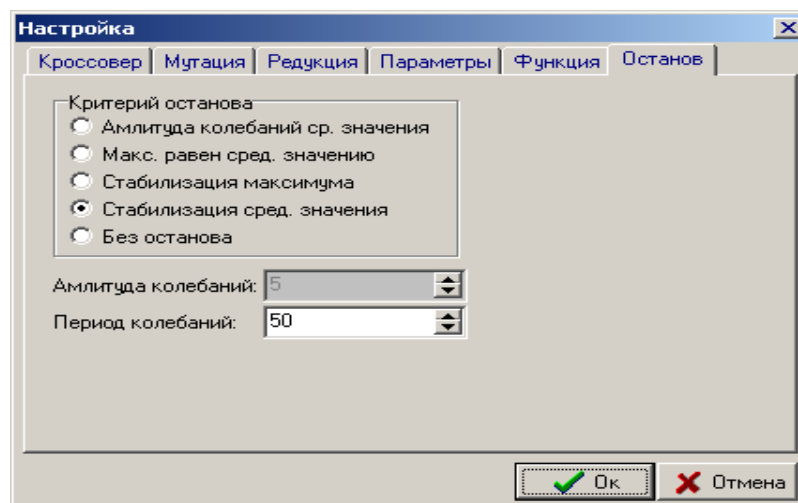


Рисунок 48 – Окно настроек

Для задания параметров работы программы выберите в меню ОПЦИИ команду ПАРАМЕТРЫ.

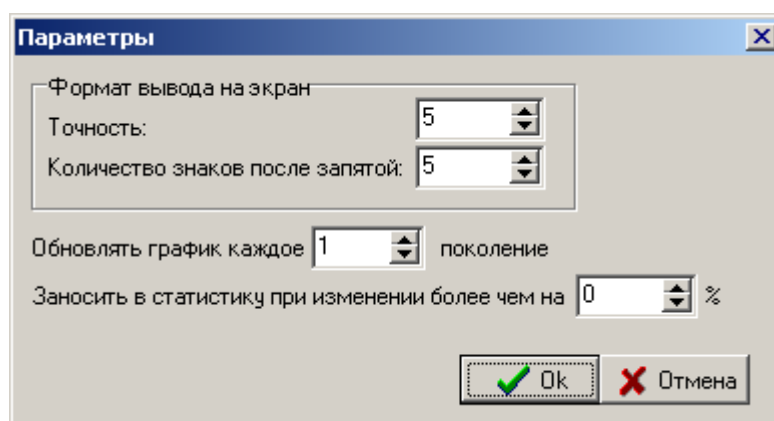


Рисунок 49 – Параметры

Для запуска генетического алгоритма в меню ЗАДАЧА выберите команду ЗАПУСК или ШАГ. После завершения работы на экране появится окно.

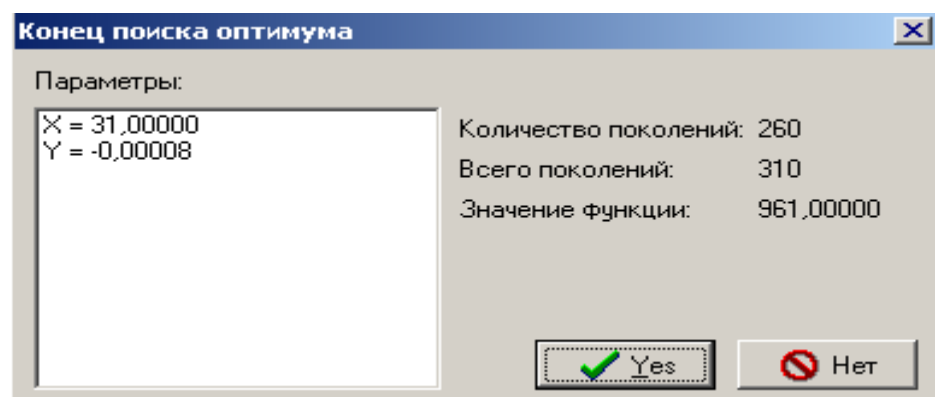


Рисунок 50 – Конец поиска оптимума

Меню файл. Данный пункт позволяет сохранять и загружать настройки проекта. Также имеется возможность сделать один из проектов “по умолчанию”, т.е. настройки будут загружаться при старте программы.

Описание пункта *меню правка*. Имеется возможность добавлять и удалять особь в популяцию, а также редактировать параметры особи.

Описание пункта *меню просмотр*. Позволяет настроить формат вывода параметров хромосомы на экран. Имеется два способа вывода: битовые двоичные строки, десятичные параметры. Здесь же имеется возможность просмотреть статистику проекта.

Описание пункта *меню задача*. Позволяет запускать проект на исполнение, прерывать, трассировать и принудительно завершать. Кроме того, здесь же возможно настроить параметры проекта.

Описание *настроек проекта*. Форма настроек включает в себя следующие закладки: “кроссовер”, “мутация”, “редукция”, “параметры”, “функция”, “останов”.

В закладке “**кроссовер**” возможна настройка оператора отбора и оператора кроссовера.

Возможен выбор следующих видов оператора отбора: рулетка, модифицированная рулетка и турнирный отбор. Вероятность участия особи в размножении по методу «рулетка» определяется формулой

$$p = \frac{100}{S} \quad \text{при Max=Min поиск минимума}$$

Где p - вероятность участия особи в размножении;

S – размер популяции;

$$p = 100 * K * \frac{Max - F}{Max - Min} \quad \text{при поиске максимума}$$

$$p = 100 * K * \frac{F - Min}{(Max - Min)} \quad \text{при поиске минимума}$$

K- коэффициент размножения.

Max – максимальное значение функции в текущей популяции;

Min – минимальное значение функции в текущей популяции;

F – значение функции в текущей популяции для текущей хромосомы;

Вероятность участия особи в размножении по методу «модифицированная рулетка» определяется формулой (при поиске минимума)

$$p = \frac{100}{S}$$

$$p = 21.7147242 * K * \ln\left(1 + 100 * \frac{Max - F}{Max - Min}\right) \quad \text{при поиске максимума}$$

Как видно из приведенных формул, используется не стандартная рулетка, в которой вероятность определяется отношением значения функции текущей особи к сумме значений функций для всех особей. В данной реализации можно заметить, что множество всех возможных значений

$$p = 21.7147242 * K * \ln\left(1 + 100 * \frac{F - Min}{(Max - Min)}\right) \quad \text{при поиске минимума}$$

функций в интервале от минимума до максимума равномерно (при стандартной рулетке) и логарифмически (при модифицированной рулетке) проецируется на множество от 0 до 100, т.е. вероятность размножения. Такой подход, позволяет резко повысить рождаемость.

Турнирный отбор. В этом методе выбираются две произвольные особи, и победителем является особь с наилучшими параметрами.

Возможен выбор оператора кроссовера: одноточечный, двухточечный и изошренный.

При одноточечном кроссовере случайным образом выбирается единственная точка разрыва В двухточечном кроссовере случайным образом выбираются две точка разрыва.

Изоцранный кроссовер реализован для задачи коммивояжера и является модификацией двухточечного кроссовера.

Здесь же возможно задать вероятность кроссовера, использование стратегии элитизма, задать коэффициент размножения и использование только одного потомка.

Вероятность кроссовера (задается в процентах) ограничивает число воспроизводимых потомков.

Параметр использование элитизма (задается в процентах) определяет процент элитных особей от размера популяции. На элитные особи не распространяется оператор редукции.

При малой вероятности редукции возможна ситуация, когда размер популяции будет неограниченно расти. Для предотвращения этого используется коэффициент размножения, который ограничивает рождаемость.

При использовании одного из двух потомков после скрещивания выбирается только один с вероятностью 50%.

В закладке “мутация” возможна настройка оператора мутации и оператора инверсии.

Имеется возможность задать вероятность инверсии, вероятность перестановки, вероятность оператора инверсии (задается в процентах).

Количество мутаций над особью неограниченно, хотя и зависит от вероятности мутации.

$$P = p_m^k$$

где P_m – вероятность мутации;

k – количество мутаций.

В закладке “редукция” возможна настройка оператора редукции.

Возможен выбор оператора редукции:

- поиск минимума функции;
- поиск максимума функции.

Здесь же возможно задать размер поколения, а также вероятность редукции.

При вероятности редукции меньше 100%, размер популяции является динамическим и меняется случайным образом. Однако меньше “размера популяции” опускаться не может.

В закладке “параметры” возможна настройка параметров функции.

Имеется два типа параметров: Integer, Float.

Каждый параметр имеет имя, диапазон (в пределах которого он может изменяться), количество бит, выделяемое под параметр. Каждый параметр может интерпретироваться двояко: как двоичная строка и как строка из кодов Грея.

В закладке “функции” необходимо задать функцию.

Функция включает в себя имя, и собственно саму функцию.

Основная функция – функция значение, которой будет интерпретироваться как значение особи. Остальные функции могут входить в параметры основной.

В закладке “останов” возможно, задать способ останова.

Также можно задать амплитуду колебаний (в процентах) и период колебаний (в поколениях).

Имеются следующие критерии останова:

- амплитуда колебаний среднего значения. Останов происходит, когда в течение периода колебаний не происходит изменения среднего значения более чем на амплитуду колебаний;
- максимум равен среднему значению. Останов происходит, когда в течение периода колебаний среднее значения становится приблизительно равным максимуму (в пределах амплитуды колебаний);
- стабилизация максимума. Останов происходит, когда в течение периода колебаний максимальное значение не меняется;
- стабилизация среднего значения. Останов происходит, когда в течение периода колебаний среднее значение не меняется;

- без останова;
- останов происходит по прерыванию от пользователя.

Пример №1

На примере данной функции $F(x) = \sin X/X^2$ провести по восемь экспериментов для каждого критерия останова, исследовать максимум одноточечного одноточечного и двухточечного кроссовера.

Критерии останова:

- Амплитуда колебаний среднего значения;
- Максимум равен среднему значению;
- Стабилизация максимума;
- Стабилизация среднего значения.

Решить задачу коммивояжера.

Данные по проводимым экспериментам представлены в таблице 10.

Таблица 10 – Данные по экспериментам

Эксперимент	Оператор отбора	Элитизм (%)	Размер популяции	Коэффициент размножения	Вероятность инверсии	Вероятность перестановки	Вероятность редукции, %
1	Рулетка	0	5	70	10	10	60
2	Рулетка	5	10	70	10	10	70
3	Рулетка	10	20	80	20	20	80
4	Рулетка	20	30	80	20	20	90
5	Турнирный отбор	0	5	70	10	10	60
6	Турнирный отбор	5	10	80	20	20	70
7	Турнирный отбор	10	20	80	30	30	80
8	Турнирный отбор	20	30	90	40	30	90

Исследование одноточенного кроссовера по четырем критериям останова (на каждый из критериев останова проведено восемь экспериментов) представлено в таблице 11.

Таблица 11 – Исследование одноточенного кроссовера

№	Количество поколений	Параметр X	Значение функции
1	113	1	0,84150
2	143	1	0,84150
3	38	1	0,84150
4	13	1	0,84150
5	200	1	0,84150
6	126	1	0,84150
7	249	1	0,84150
8	93	1	0,84150
9	288	1	0,84150
10	195	1	0,84150
11	416	1	0,84150
12	16206	-	-
13	16101	-	-
14	14002	-	-
15	13099	-	-
16	13111	-	-
17	71	0,99940	0,84120
18	183	1	0,84150
19	183	1	0,84150
20	79	1	0,84150
21	219	1	0,84150
22	67	0,99990	0,84140
23	154	1	0,84150
24	1	0,99060	0,83640
25	10087	-	-

26	11345	-	-
27	2195	1	0,84150
28	1277	1	0,84150
29	6878	-	-
30	5110	-	-
31	6033	-	-
32	7122	-	-

Некоторые экранные формы, полученные при проведении экспериментов представлены на рисунках 51-55.

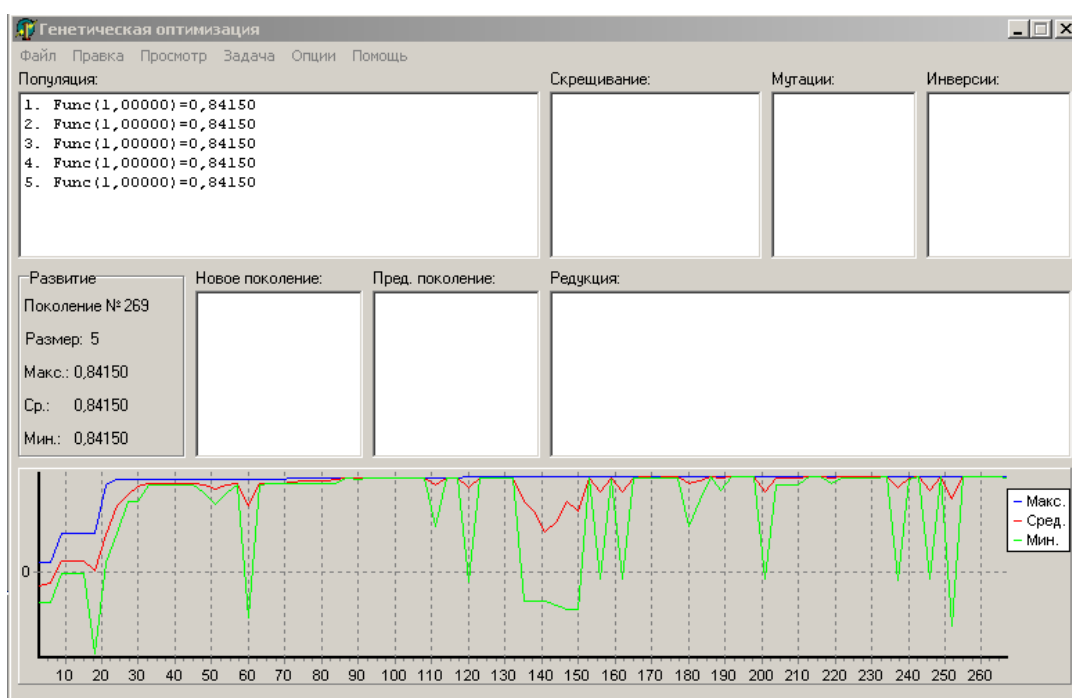


Рисунок 51 – Исследование однотоочечного кроссовера при стабилизации максимума (эксперимент 1)

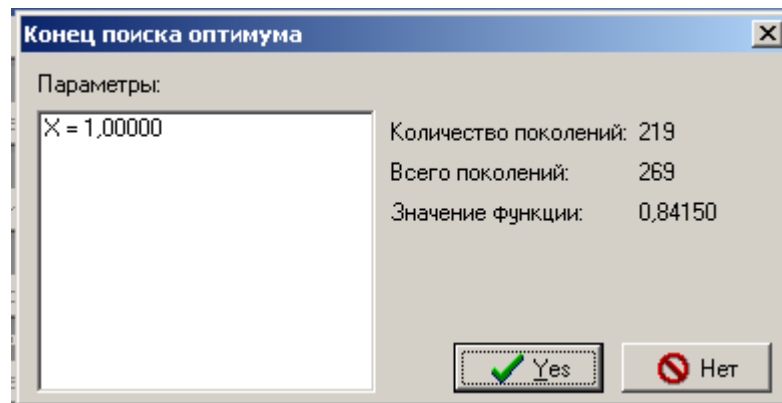


Рисунок 52 – Результат эксперимента

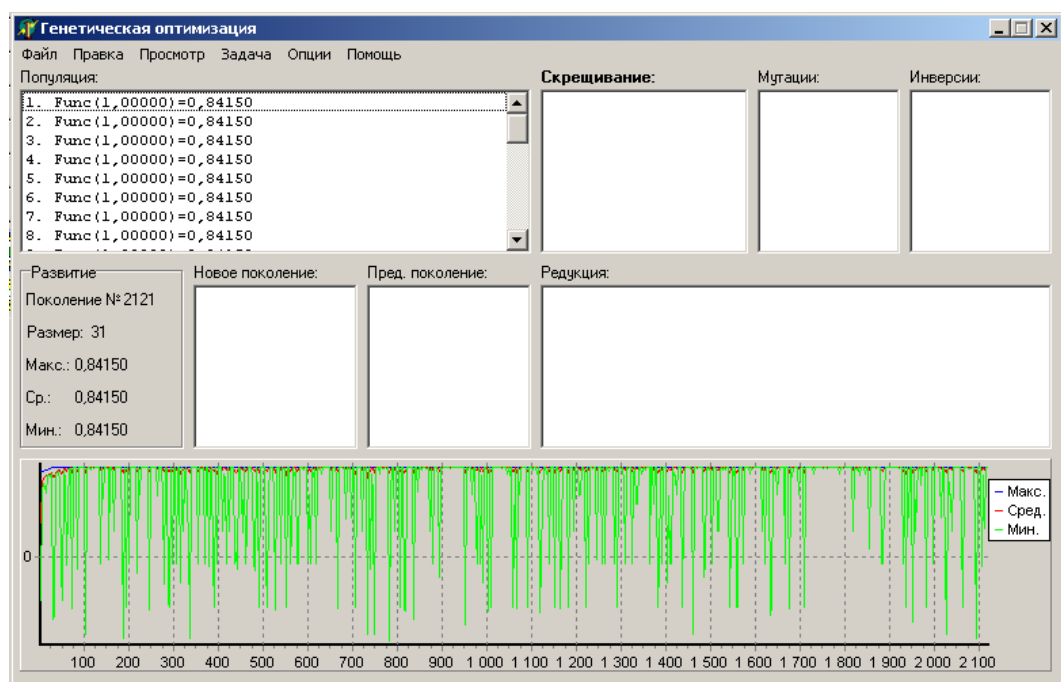


Рисунок 53 - Исследование одноточечного кроссовера при максимуме равном среднему значению (эксперимент 8)

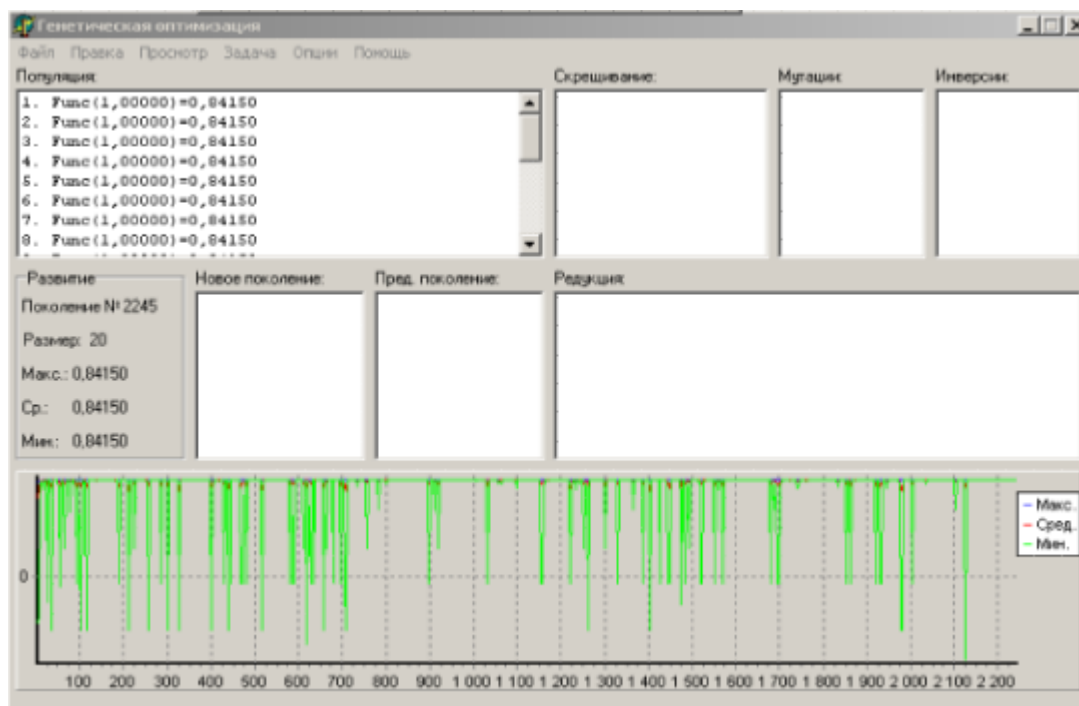


Рисунок 54 - Исследование одноточечного кроссовера при стабилизации среднего значения (эксперимент 3)

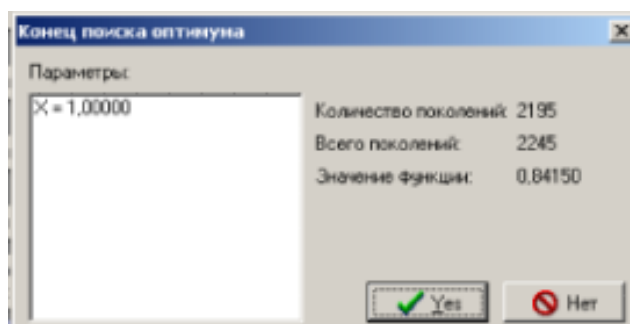


Рисунок 55 - Результат

Исследование двухточечного кроссовера по четырем критериям останова (на каждый из критериев останова проведено восемь экспериментов) представлено в таблице 12.

Таблица 12 – Исследование двухточечного кроссовера

№	Количество поколений	Параметр X	Значение функции
1	394	1	0,84150
2	47	0,99210	0,83720

3	100	1	0,84150
4	10	1	0,84150
5	228	1	0,84150
6	231	1	0,84150
7	205	1	0,84150
8	28	0,99990	0,84140
9	1176	1	0,84150
10	887	1	0,84150
11	600	1	0,84150
12	6818	-	-
13	5884	-	-
14	6671	-	-
15	12306	-	-
16	13911	-	-
17	109	1	0,84150
18	91	0,97590	0,82820
19	142	1	0,84150
20	118	1	0,84150
21	71	0,99680	0,83980
22	363	1	0,84150
23	125	1	0,84150
24	81	1	0,84150
25	132	0,66990	0,62090
26	714	1	0,84150
27	134	1	0,84150
28	855	1	0,84150
29	14034	-	-
30	12721	-	-
31	12987	-	-
32	13137	-	-

Некоторые экранные формы, полученные при проведении экспериментов представлены на рисунках 56-63.

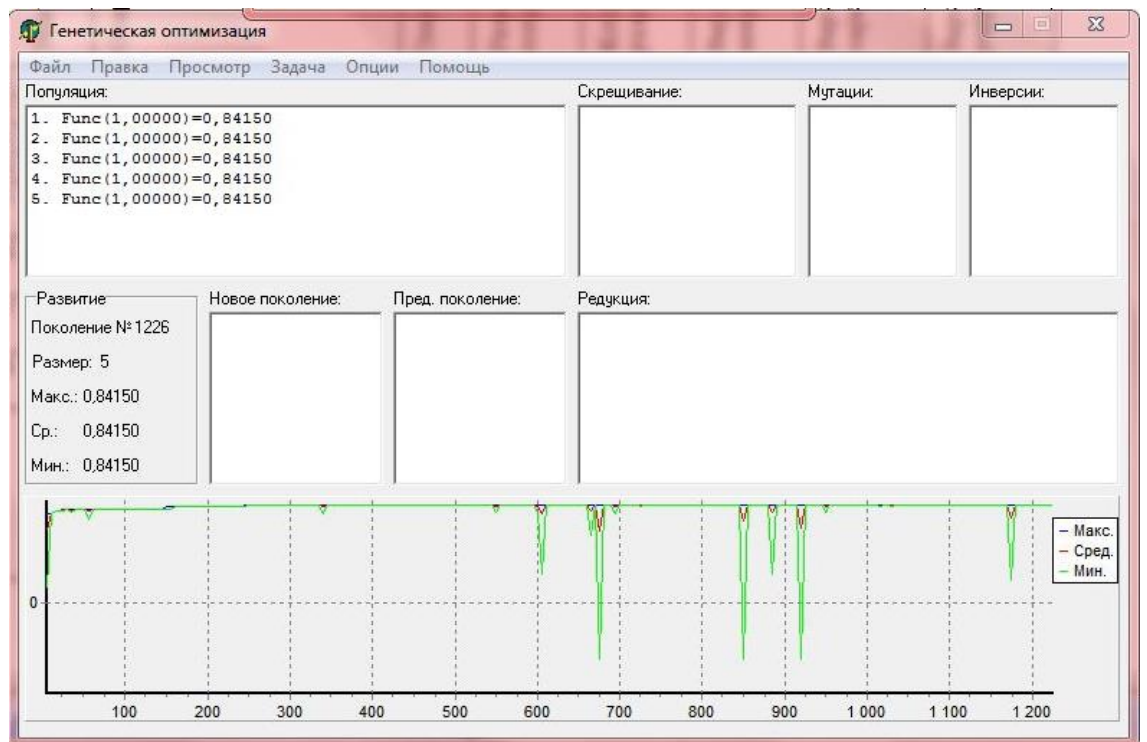


Рисунок 56 - Исследование двухточечного кроссовера при максимуме равном среднему значению (эксперимент 1)

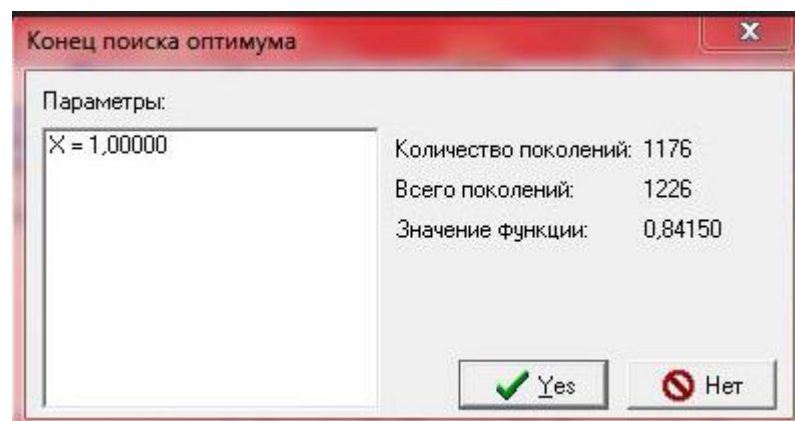


Рисунок 57 – Результат

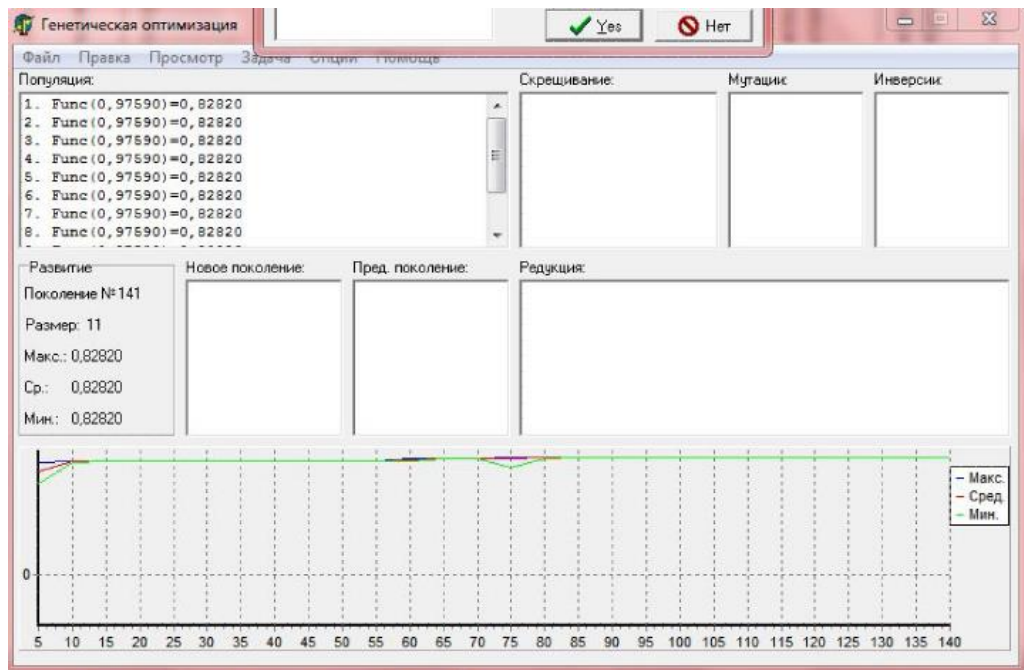


Рисунок 58 – Исследование двухточечного кроссовера при стабилизации максимума (эксперимент 2)

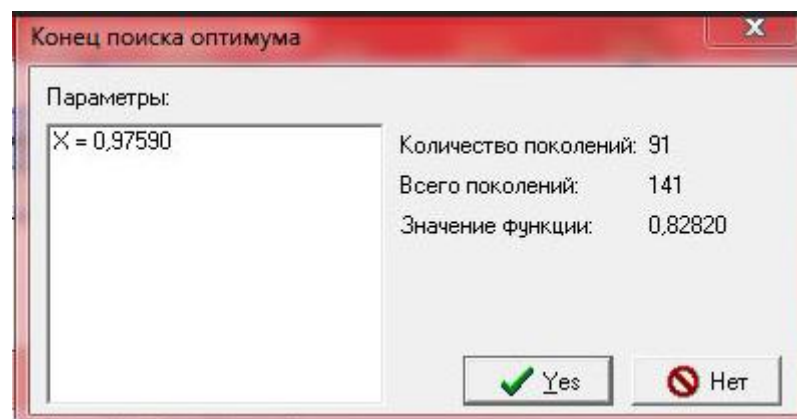


Рисунок 59 – Результат



Рисунок 60 – Исследование двухточечного кроссовера при стабилизации максимума (эксперимент 5)

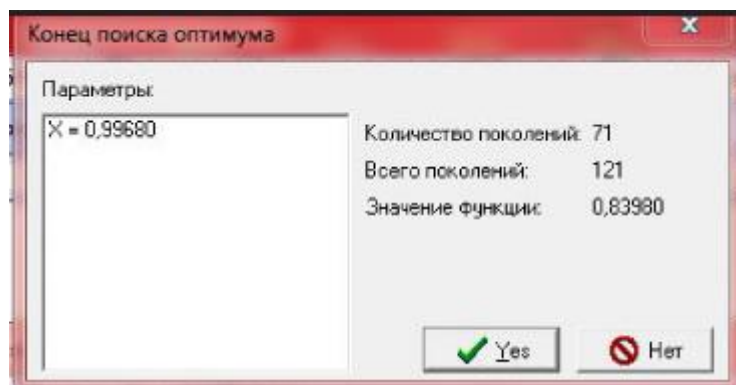


Рисунок 61 - Результат

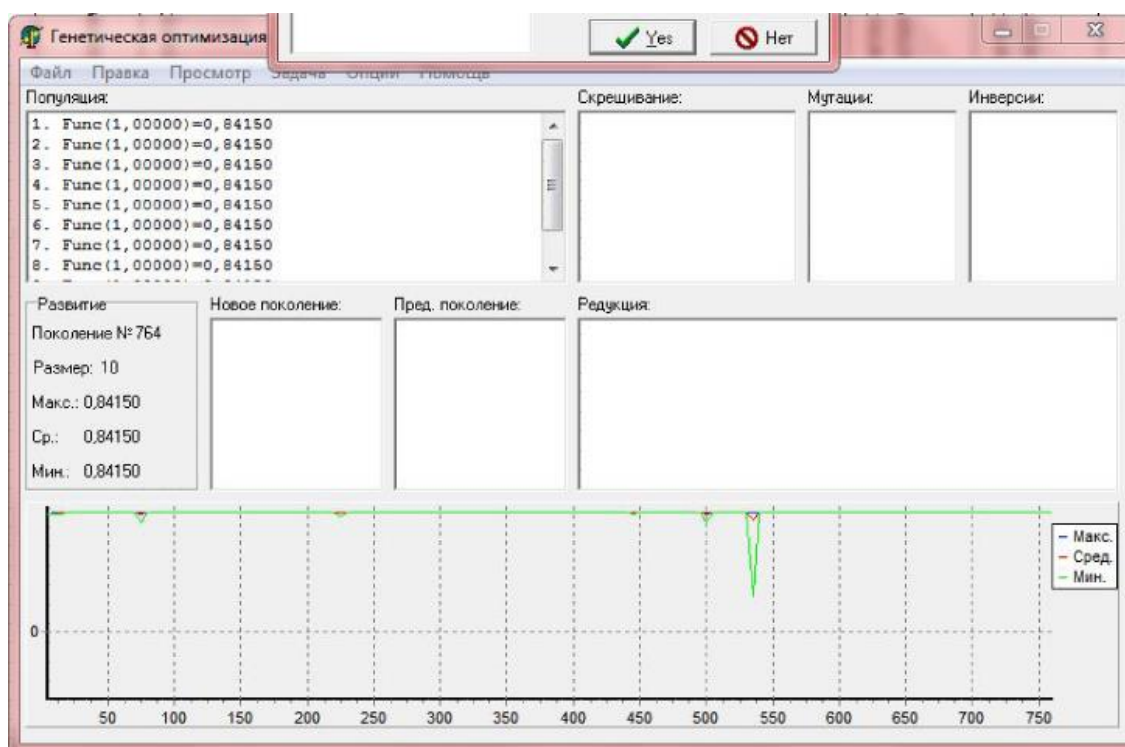


Рисунок 62 – Исследование двухточечного кроссовера при стабилизации среднего значения (эксперимент 2)

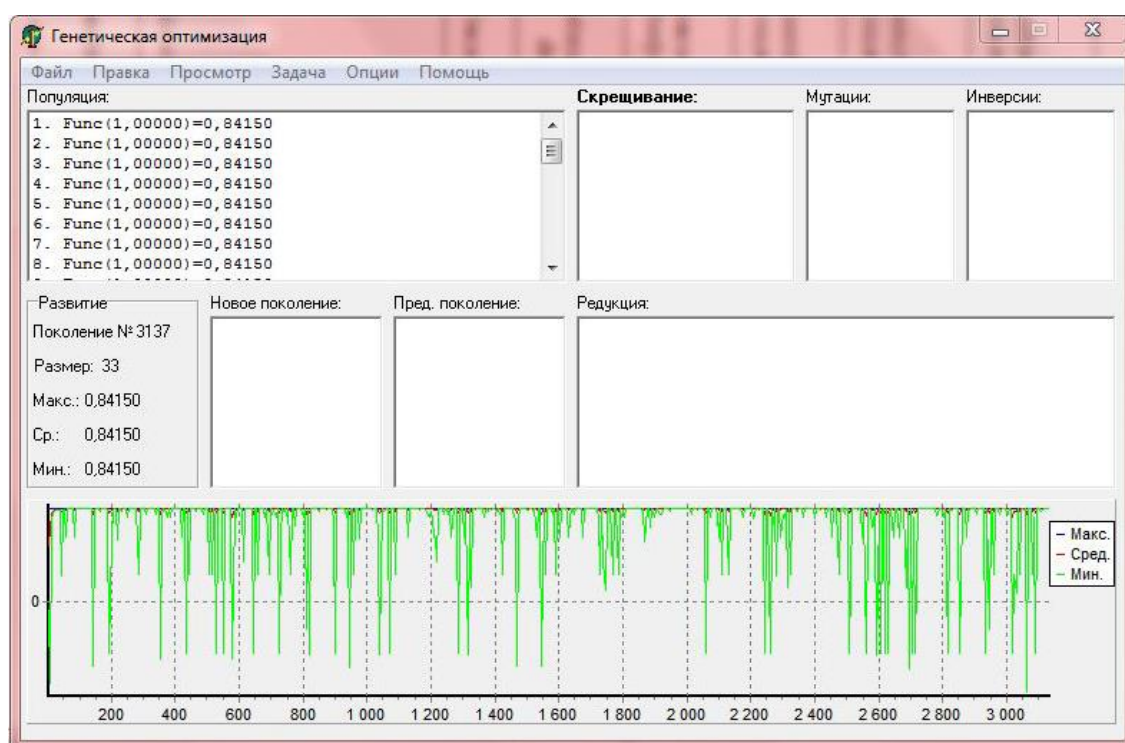


Рисунок 63 - Исследование двухточечного кроссовера при стабилизации среднего значения (эксперимент 8)

Решение задачи коммивояжера (таблица 3) представлено на рисунках 64-70.

Таблица 13 – Данные для задачи коммивояжера

	пункт 1	пункт 2	пункт 3	пункт 4	пункт 5	пункт 6	пункт 7
пункт 1	3	2	5	4	7	1	6
пункт 2	11	3	3	1	1	2	3
пункт 3	2	2	12	11	2	3	2
пункт 4	1	2	1	2	1	2	3
пункт 5	2	11	2	1	2	3	2
пункт 6	1	3	4	2	3	4	3
пункт 7	4	3	2	5	2	4	2

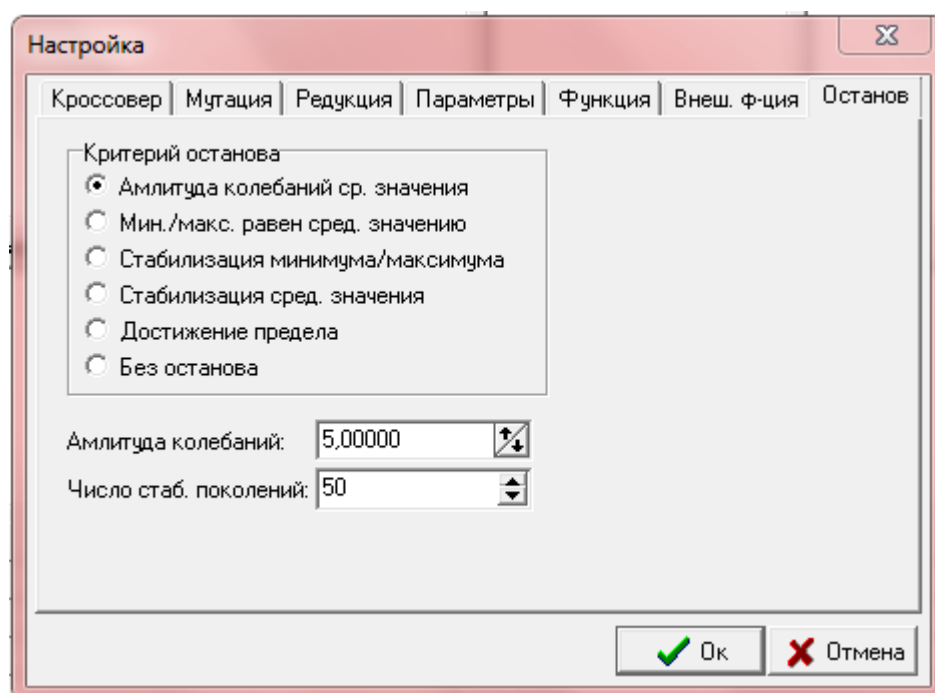


Рисунок 64 – Параметры для решения задачи

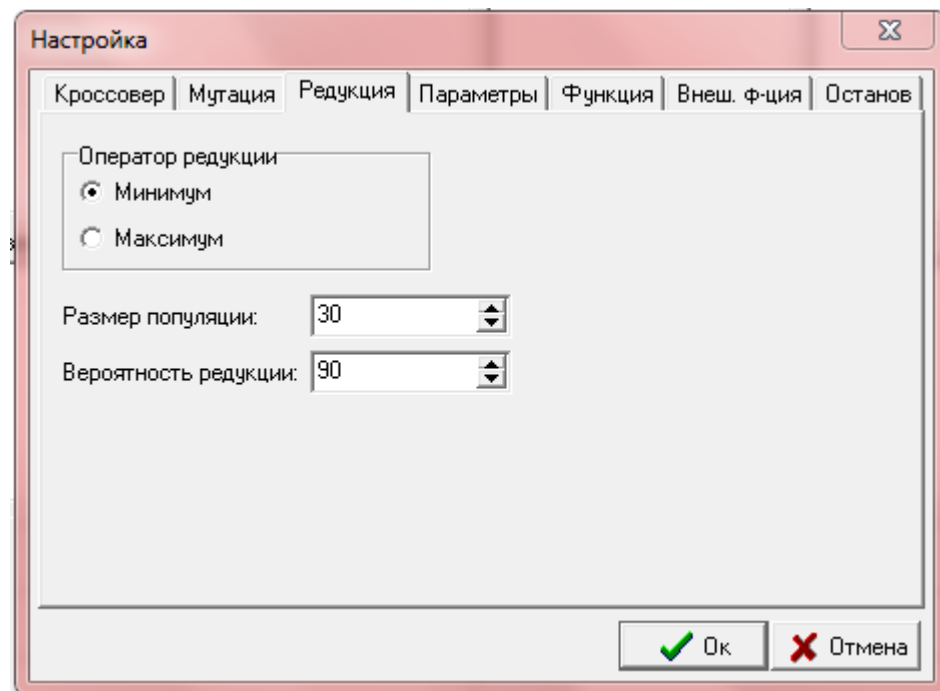


Рисунок 65 – Параметры для решения задачи

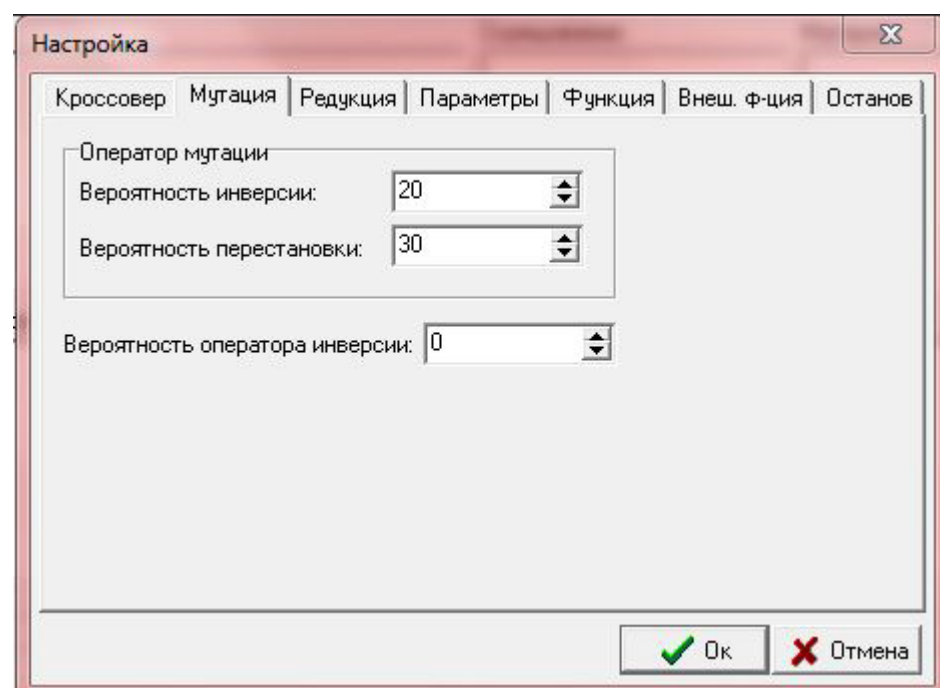


Рисунок 66 – Параметры для решения задачи

Настройка

Кроссовер | Мутация | Редукция | Параметры | Функция | Внеш. ф-ция | Останов

Оператор отбора

- ☒ Рулетка
- ☐ Модифицированная рулетка
- ☐ Турнирный отбор

Оператор кроссовера

- ☐ Одноточечный
- ☐ Двухточечный
- ☒ Изохренный

Вероятность кроссовера: 100

Использование элитизма: 0

Коэффициент размножения: 0,90

☒ Один потомок

Ок Отмена

Рисунок 67 - Параметры для решения задачи

Ввод матрицы

Размерность: 7

3	2	5	4	7	1	6
11	3	3	1	1	2	3
2	2	12	11	2	3	2
1	2	1	2	1	2	3
2	11	2	1	2	3	2
1	3	4	2	3	4	3
4	3	2	5	2	4	2

Экспорт Импорт

Ок Отмена

Рисунок 68 - Параметры для решения задачи

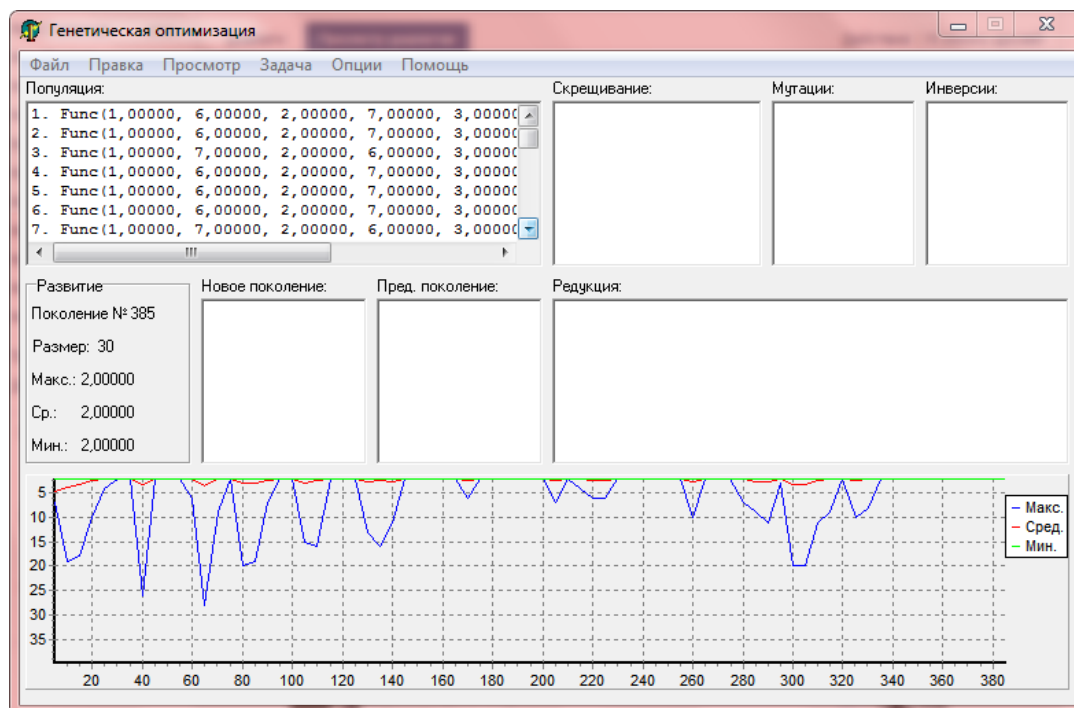


Рисунок 69 – Процесс решения задачи

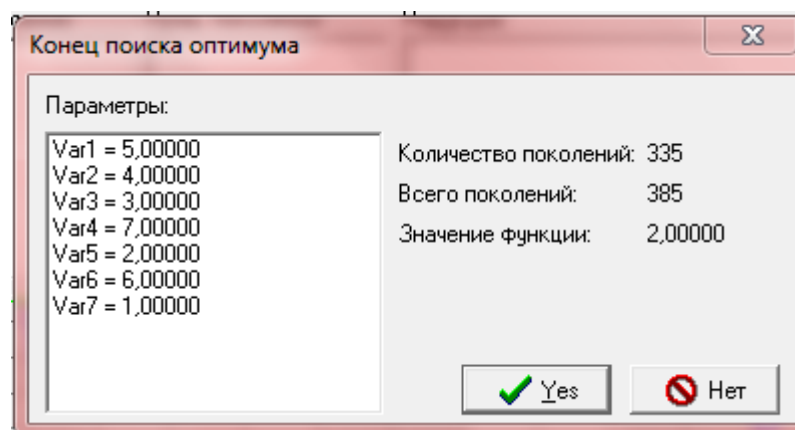


Рисунок 70 – Результат решения задачи

По результатам работы программы получены следующие значения: максимум функции равен 0,84150, при X равном 1.

Сравнительный анализ результатов поиска экстремума (максимума) для заданной функции показал, что использование одновременно таких параметров как “турнирный отбор” и критериев останова - “Максимум равен среднему значению” или “Стабилизация среднего значения” не обеспечивает сходимости процесса как при использовании одноточечного, так и двухточечного кроссовера.

Дополнительное задание (выполняется по согласованию с преподавателем):

1) выполнить эмуляцию решения задания в соответствии с вариантом на Python с применением генетического алгоритма.

2) выполнить эмуляцию решения задачи коммивояжера на Python с применением генетического алгоритма.

В отчет к дополнительному заданию включить исходный код, экранные формы, необходимые пояснения.

Пример №2

Максимизировать следующую простую целевую функцию:

$$f(x)=f(x)=x^2+10\sin(x)$$

Цель: Найти максимальное значение этой функции в интервале [0,10] методом генетического алгоритма.

```
import random
import numpy as np

# Фитнес-функция (целевая функция)
def fitness_function(x):
    return x**2 + 10 * np.sin(x)

# Функция инициализации начальной популяции
def initialize_population(population_size, chromosome_length):
    population = []
    for _ in range(population_size):
        # Генерация случайных двоичных хромосом заданной длины
        chromosome = ''.join(random.choice('01') for _ in range(chromosome_length))
        population.append(chromosome)
    return population

# Декодирование хромосомы обратно в число
def decode_chromosome(chromosome, min_value=0, max_value=10):
```



```

        decimal_value = int(chromosome, 2)

        scaled_value = min_value + (decimal_value / ((2**len(chromosome)) - 1)) * (max_value -
min_value)

        return scaled_value

# Отбор особей для скрещивания

def selection(population, fitness_values, num_parents):

    parents = sorted(zip(population, fitness_values), key=lambda x: x[1],
reverse=True)[:num_parents]

    return [parent[0] for parent in parents]

# Одноточечный кроссовер

def crossover(parent1, parent2):

    point = random.randint(1, len(parent1)-1)

    child1 = parent1[:point] + parent2[point:]

    child2 = parent2[:point] + parent1[point:]

    return child1, child2

# Мутация с определенной вероятностью

def mutation(child, mutation_rate):

    mutated_child = ""

    for bit in child:

        if random.random() < mutation_rate:

            mutated_child += '1' if bit == '0' else '0'

        else:

            mutated_child += bit

    return mutated_child

# Основной цикл генетического алгоритма

def genetic_algorithm(population_size=100, generations=100, chromosome_length=8,
mutation_rate=0.01):

    population = initialize_population(population_size, chromosome_length)

    for generation in range(generations):

        decoded_population = [decode_chromosome(chromosome) for chromosome in population]

```

```

fitness_values = [fitness_function(x) for x in decoded_population]

best_fitness = max(fitness_values)
print(f'Generation {generation}: Best Fitness={best_fitness}')

selected_parents = selection(population, fitness_values, population_size // 2)

new_population = []
while len(new_population) < population_size:
    parent1, parent2 = random.sample(selected_parents, 2)
    child1, child2 = crossover(parent1, parent2)

    child1 = mutation(child1, mutation_rate)
    child2 = mutation(child2, mutation_rate)

    new_population.extend([child1, child2])

population = new_population[:population_size]

best_chromosome = max(population, key=lambda chromo:
fitness_function(decode_chromosome(chromo)))
best_x = decode_chromosome(best_chromosome)
best_fitness = fitness_function(best_x)

return best_x, best_fitness

if __name__ == "__main__":
    best_solution, best_fitness = genetic_algorithm()
    print(f"\nBest solution found at x={best_solution}, with fitness value={best_fitness:.2f}")

```

Генерируем начальное поколение случайных решений (хромосом).

Для каждой хромосомы вычисляем её фитнес-значение, используя нашу целевую функцию.

Лучшие особи выбираются для участия в процессе воспроизводства.

Новые решения формируются путем смешивания частей родительских хромосом.

Небольшое количество случайно выбранных битов изменяется для повышения разнообразия решений.

Повторяем шаги 2-5 нужное количество поколений.

Пример №3

Пример реализации генетического алгоритма для задачи коммивояжера с четырьмя пунктами

```
import random
import numpy as np

# Функция расчета стоимости маршрута
def calculate_total_distance(route, distance_matrix):
    total_dist = sum(distance_matrix[route[i]][route[i+1]] for i in range(len(route)-1)) +
distance_matrix[route[-1]][route[0]]
    return total_dist

# Инициализация начальной популяции
def initialize_population(population_size, num_cities):
    population = []
    cities = list(range(num_cities))
    for _ in range(population_size):
        random.shuffle(cities)
        population.append(list(cities))
    return population

# Отбор особей (элитизм + турнирный отбор)
def selection(population, fitness_scores, elite_size=2):
    sorted_indices = np.argsort(fitness_scores)
    elites = [population[idx] for idx in sorted_indices[:elite_size]]
```

```

        tournament_participants = random.sample(sorted_indices[elite_size:], k=(len(population) -
elite_size))

        winners = [population[idx] for idx in tournament_participants]

    return elites + winners

# Скрещивание (частичный кроссовер ОХ)
def crossover(parent1, parent2):
    start, end = sorted(random.sample(range(len(parent1)), 2))
    child = [-1]*len(parent1)
    child[start:end+1] = parent1[start:end+1]

    remaining_genes = [gene for gene in parent2 if gene not in child]
    index = 0
    for i in range(len(child)):
        if child[i] == -1:
            child[i] = remaining_genes[index]
            index += 1

    return child

# Мутация (swap mutation)
def mutate(individual, mutation_rate):
    if random.random() < mutation_rate:
        idx1, idx2 = random.sample(range(len(individual)), 2)
        individual[idx1], individual[idx2] = individual[idx2], individual[idx1]
    return individual

# Основной процесс эволюции
def genetic_algorithm(distance_matrix, population_size=100, generations=100, elite_size=2,
mutation_rate=0.1):
    num_cities = len(distance_matrix)
    population = initialize_population(population_size, num_cities)

```

```

for generation in range(generations):
    fitness_scores = [calculate_total_distance(route, distance_matrix) for route in population]

    selected_parents = selection(population, fitness_scores, elite_size)

    offspring = []
    while len(offspring) < population_size:
        parent1, parent2 = random.sample(selected_parents, 2)
        child = crossover(parent1, parent2)
        mutated_child = mutate(child, mutation_rate)
        offspring.append(mutated_child)

    population = offspring

    best_fitness = min([calculate_total_distance(route, distance_matrix) for route in population])
    best_route = [route for route in population if calculate_total_distance(route, distance_matrix)
== best_fitness][0]

    return best_route, best_fitness

# Основная матрица расстояний (пример)
distance_matrix = [
    [0, 10, 15, 20],
    [10, 0, 35, 25],
    [15, 35, 0, 30],
    [20, 25, 30, 0]
]

# Запуск генетического алгоритма
best_route, best_fitness = genetic_algorithm(distance_matrix)

print("Лучший маршрут:", best_route)
print("Общая длина маршрута:", best_fitness)

```

- 1) Создается начальная популяция случайных маршрутов.
- 2) Вычисляется общая длина каждого маршрута.
- 3) Лучшие особи сохраняются ("элиты"), остальные подвергаются отбору по принципу турнира.
- 4) Производится скрещивание лучших особей, используя частичный кроссовер (OX-crossover).
- 4) Случайно мутируют некоторые гены потомков.
- 6) Итерации повторяются до достижения определенного числа поколений.

Population size (population_size) — размер популяции.

Generations (generations) — количество итераций эволюции.

Elite size (elite_size) — число элитных особей, сохраняемых в каждом поколении.

Mutation rate (mutation_rate) — вероятность мутации отдельной хромосомы.

Лабораторная работа №5

Разработка программной оболочки для работы с Базой Знаний на языке Prolog

Цель работы: изучение основ языка программирования Prolog и получение навыков разработки баз знаний на языке программирования Prolog.

Отчет должен содержать краткие теоретические сведения, описание базы знаний, исходный код на языке Prolog, экранные формы работы, выводы.

Задание: разработать программную оболочку в среде разработки Visual Prolog, для работы с Базой знаний. Тематика БЗ соответствует лабораторной работе №1.

Пролог (англ. Prolog) — язык и система логического программирования, основанные на языке предикатов математической логики дизъюнктов Хорна, представляющей собой подмножество логики предикатов первого порядка. Язык сосредоточен вокруг небольшого набора основных механизмов, включая сопоставление с образцом, древовидного представления структур данных и автоматического перебора с возвратами. Хорошо подходит для решения задач, где рассматриваются объекты (в частности, структурированные объекты) и отношения между ними. Пролог, благодаря своим особенностям, используется в области искусственного интеллекта, компьютерной лингвистики и нечислового программирования в целом. В некоторых случаях реализация символьных вычислений на других стандартных языках вызывает необходимость создавать большое количество кода, сложного в понимании, в то время как реализация тех же алгоритмов на языке Пролог дает простую программу, легко помещающуюся на одной странице. Prolog является декларативным языком программирования: логика программы выражается в терминах отношений, представленных в виде фактов и правил. Для того, чтобы инициировать вычисления, выполняется специальный *запрос* к базе знаний, на которые система логического программирования генерирует ответы «истина» и «ложь». Для обобщённых запросов с переменными в качестве аргументов созданная система Пролог выводит конкретные данные в подтверждение истинности обобщённых сведений и правил вывода.

База знаний — база данных, содержащая правила вывода и информацию о человеческом опыте и знаниях в некоторой предметной области. В самообучающихся системах база знаний также содержит информацию, являющуюся результатом решения предыдущих задач.

Современные базы знаний работают совместно с системами поиска и извлечения информации. Для этого требуется некоторая модель классификации понятий и определённый формат представления знаний. Иерархический способ представления в базе знаний набора понятий и их отношений называется онтологией.

Онтологию некоторой области знаний вместе со сведениями о свойствах конкретных объектов часто называют «базой знаний». Вместе с тем, полноценные базы знаний (в отличие от обычной базы данных) содержат в себе не только фактическую информацию, но и правила вывода, позволяющие делать автоматические умозаключения об уже имеющихся или вновь вводимых фактах и тем самым производить семантическую (осмысленную) обработку информации.

Область наук об искусственном интеллекте, изучающая базы знаний и методы работы со знаниями, называется инженерией знаний.

База знаний — важный компонент интеллектуальной системы. Наиболее известный класс таких программ — это экспертные системы. Они предназначены для поиска способов решения проблем из некоторой предметной области, основываясь на записях БЗ и на пользовательском описании ситуации.

Простые базы знаний могут использоваться для создания экспертных систем хранения данных в организации: документации, руководств, статей технического обеспечения. Главная цель создания таких баз — помочь менее опытным людям найти уже существующее описание способа решения какой-либо проблемы.

Ниже перечислены некоторые из особенностей, которые могут (но не обязаны) быть у системы, оперирующей базами знаний.

- Автоматическое доказательство (вывод). Способность системы выводить новые знания из старых, находить закономерности в БЗ. Часто принимается, что база знаний отличается от базы данных именно наличием механизма вывода.
- Доказательство заключения. Способность системы после выдачи ответа «объяснить» ход её рассуждений, причем «по первому требованию».
- Интроспекция. Нахождение противоречий, нестыковок в БЗ, контроль правильной организации БЗ.
- Машинное обучение. Превращение БЗ в гибкую систему, адаптация к проблемной области. Аналогична человеческой способности «набирать опыт».

В языке Пролог базы знаний описываются в форме конкретных фактов и правил логического вывода над базами данных и процедурами обработки информации, представляющих сведения и знания о людях, предметах, фактах, событиях и процессах в логической форме. В ответах на простейшие запросы к базам знаний система логического программирования Пролог выдает значения «истина» и «ложь» в зависимости от наличия соответствующих фактов.

Обобщенные сведения в языке Пролог задаются с помощью правил логического вывода, выполняющих роль определения понятий, а также логических процедур, состоящих из наборов правил логического вывода. Достоверность обобщенных сведений зависит от наличия необходимых фактов и достоверности данных в базах знаний.

Язык сосредоточен вокруг небольшого набора основных механизмов, включая сопоставление с образцом, древовидного представления структур данных и автоматического перебора с возвратами. Хорошо подходит для решения задач, где рассматриваются объекты (в частности, структурированные объекты) и отношения между ними. Пролог, благодаря своим особенностям, используется в области искусственного интеллекта, компьютерной лингвистики и нечислового программирования в целом. В некоторых случаях реализация символьных вычислений на других стандартных языках вызывает необходимость создавать большое количество кода, сложного в понимании, в то время как реализация тех же алгоритмов на языке Пролог дает простую программу, легко помещающуюся на одной странице.

Prolog является декларативным языком программирования: логика программы выражается в терминах отношений, представленных в виде фактов и правил. Для того, чтобы инициировать вычисления, выполняется специальный запрос к базе знаний, на которые система логического программирования генерирует ответы «истина» и «ложь». Для обобщённых запросов с переменными в качестве аргументов созданная система Пролог выводит конкретные данные в подтверждение истинности обобщённых сведений и правил вывода.

Начало истории языка относится к 1970-м годам. Будучи декларативным языком программирования, Пролог воспринимает в качестве программы некоторое описание задачи или баз знаний и сам производит логический вывод, а также поиск решения задач, пользуясь механизмом поиска с возвратом и унификацией.

Интерес к Прологу поднимался и затихал несколько раз, энтузиазм сменялся жёстким неприятием. Наиболее высоко был поднят интерес к языку

Пролог, как к языку будущего, во время разработок японской национальной программы компьютеры пятого поколения в 1980-х годах, когда разработчики надеялись, что с помощью Пролога можно будет сформулировать новые принципы, которые приведут к созданию компьютеров более высокого уровня интеллекта.

Язык Пролог в 1980-х годах был включен в ряд советских вузовских и школьных учебников информатики для изучения элементов математической логики, принципов логического программирования и проектирования баз знаний и моделей экспертных систем. С этой целью на IBM PC и ряде советских школьных компьютеров были реализованы учебные русскоязычные интерпретаторы Пролога.

В языке Пролог факты описываются в форме логических предикатов с конкретными значениями. Правила вывода описываются логическими предикатами с определением правил логического вывода в виде списка предикатов над базами знаний и процедурами обработки информации.

В настоящее время Пролог, несмотря на неоднократные пессимистические прогнозы, продолжает развиваться в разных странах и вбирает в себя новые технологии и концепции, а также парадигмы императивного программирования. В частности, одно из направлений развития языка (в том числе и в России) реализует концепцию интеллектуальных агентов.

Пролог реализован практически для всех известных операционных систем (ОС) и платформ (в том числе для Java и .NET). В число операционных систем входят: ОС для мейнфреймов, всё семейство Unix, Windows, ОС для мобильных платформ.

Многие современные реализации языка имеют внутреннее расширение за счёт ООП-архитектуры. Кроме несвободных решений также существуют свободные реализации Пролога. В 1996 году был принят стандарт ISO, получивший название ISO/IEC JTC1/SC22/WG17.

Базовым принципом языка является равнозначность представления программы и данных (декларативность), отчего утверждения языка одновременно являются и записями, подобными записям в базе данных, и правилами, несущими в себе способы их обработки. Сочетание этих качеств приводит к тому, что по мере работы системы Пролога знания (и данные, и правила) накапливаются. Поэтому Пролог-системы считают естественной средой для накопления базы знаний и обучения студентов и школьников принципам логического программирования.

Основными понятиями в языке Пролог являются факты, правила логического вывода и запросы, позволяющие описывать базы знаний, процедуры логического вывода и принятия решений.

Программа на Прологе описывает отношения, определяемые с помощью предложений. Как и в любом другом языке, ориентированном на символьные вычисления, предложения выстраиваются из термов, которые в свою очередь подразделяются на атомы, числа, переменные и структуры. Атом записывается со строчной буквы или заключается в кавычки, когда требуется запись с прописной буквы.

Atom

'Atom'

Переменные, записываемые с прописной буквы, отличаются от переменных в процедурных языках программирования, они не связаны с конкретной ячейкой памяти, а скорее ближе к математической переменной.

X is 2 + 2.

Структуры представляют собой совокупности термов, заключенные в круглые скобки, в том числе и другие структуры. Структура обозначается именем (функтором), которое располагается перед круглыми скобками.

book('Название', '2009', 'Спб', authors('Первый автор', 'Второй автор')).

Ещё одной конструкцией являются списки, элементы которых заключаются в квадратные скобки. В основе списков в Пролог лежат связные списки.

List = [a, b, [c, d], e].

Правила в Прологе записываются в форме правил логического вывода с логическими заключениями и списком логических условий. В чистом Прологе предложения ограничиваются дизъюнктами Хорна:

Вывод :- Условие.

и читаются так: «Заголовок ИСТИНА, если тело ИСТИНА». Тело правила содержит ссылки на предикаты, которые называются целями правила.

Встроенные предикаты

,/2

Значение: оператор с двумя аргументами. Определяет конъюнкцию целей.

;/2

Оператор определяет дизъюнкцию.

Факты в языке Пролог описываются логическими предикатами с конкретными значениями. Факты в базах знаний на языке Пролог представляют конкретные сведения (знания). Обобщённые сведения и знания в языке Пролог задаются правилами логического вывода (определениями) и наборами таких правил вывода (определений) над конкретными фактами и обобщёнными сведениями. Предложения с пустым телом называются фактами. Пример факта:

Кот(Иван).

Этот факт эквивалентен правилу:

Кот(Иван) :- ИСТИНА.

Visual Prolog — объектно-ориентированное расширение языка программирования PDC Prolog, развивавшегося из Turbo Prolog (Borland), семейства Prolog, а также система визуального программирования датской фирмы Prolog Development Center.

Prolog Development Center затратил более трех лет на разработку системы Visual Prolog с поэтапным бета-тестированием, поставки коммерческой версии которой начались с февраля 1996.

Visual Prolog автоматизирует построение сложных процедур и освобождает программиста от выполнения тривиальных операций. С помощью Visual Prolog проектирование пользовательского интерфейса и связанных с ним окон, диалогов, меню, строки уведомлений о состояниях и т. д. производится в графической среде. С созданными объектами могут работать различные Кодовые Эксперты (Code Experts), которые используются для генерации базового и расширенного кодов на языке Prolog, необходимых для обеспечения их функционирования.

Мощность языка Prolog в сочетании с системой пользовательских интерфейсов упрощает разработку систем, основанных на знаниях, систем поддержки принятия решений, планирующих программ, развитых систем управления базами данных и т. д.

Язык программирования, реализованный в Visual (а ранее в Turbo) Prolog, отличается от классического пролога тем, что он основан на строгой статической типизации. В него также добавлены средства объектно-ориентированного программирования, анонимные предикаты (лямбда-предикаты), факты-переменные и разрушающее присваивание для них, аргументы-домены (Generic Interfaces and Classes) и параметрический

полиморфизм, мониторы (Monitors with guards), императивные конструкции (foreach, if...then...else), коллекторы списков ([...||...]) и пр.

Среда разработки приложений системы Visual Prolog включает текстовый редактор, различные редакторы ресурсов, средства разработки справочных систем в гипертекстовом представлении, систему отслеживания изменений, которая обеспечивает перекомпиляцию и регенерацию только измененных ресурсов и модулей, ряд экспертов Кода, оптимизирующий компилятор, набор средств просмотра различных типов информации о проекте и отладчик. Полная интеграция всех средств обеспечивает повышение скорости разработки приложений. Полученные приложения являются исполняемыми .EXE программами. В коммерческой версии Visual Prolog 7.x возможно создание .DLL-файлов, в персональной версии такая возможность существовала вплоть до версии 5.x.

Система программирования пользовательских интерфейсов (GUI — Graphic User Interface) системы Visual Prolog является высокоуровневой абстракцией к функциям операционной системы.

В систему включен также интерфейс с базами данных типа SQL. Почти все типы баз данных доступны с использованием Windows ODBC интерфейса. Поддерживаются также обращения к базам данных Oracle.

В инсталляционный пакет входит 50 классов (Prolog Foundation Classes). Среди них есть GDI+, криптографический, компрессия данных, COM, интерпретатор Классического Пролога PIE (Prolog Inference Engine) и пр.

Пример

Разработать программу в среде разработки Visual Prolog, для работы с Базой знаний. Тематика – спортивное питание.

База знаний содержит следующие факты:

- Протеин(название,цена, критерий соответствия продукции для тяжелой атлетики)
- Аминокислоты(название,цена,фирма производитель)
- Гейнер(название,цена,фирма, производитель)
- Витамин(название,цена)
- Креатин(название,цена,критерий соответствия продукции для тяжелой атлетики)
- Л-Карнитин(название,цена,скорость жиросжигания)

- БАД(название,цена, критерий соответствия продукции для тяжелой атлетики)
- Трибулус(название,цена, критерий соответствия продукции для тяжелой атлетики)

где параметрами являются следующие строковые данные:

- цена: низкая, средняя, высокая
- критерий соответствия продукции для тяжелой атлетики: соответствует, не соответствует
- скорость жиросжигания: высокая, низкая
- фирма производитель: Twinlab, Muscletech

Список фактов представлен на рисунке 71.

Предикаты второго уровня построены по типу выбрать комплексы спортивного питания с низкой ценой, имеющие конкретного производителя, подходящие для занятий тяжелой атлетикой.

Предикаты третьего уровня построены по типу выбрать комплексы спортивного питания, имеющие конкретного производителя и подходящие для тяжелой атлетики, имеющие низкую цену и высокую скорость жиросжигания, и т.п.

Результат работы программы представлен на рисунках 72-78.



```

fa — БЛОКНОТ
Файл  Правка  Формат  Вид  Справка

clauses
prot("Протеин 1", varysmallprice, yes).
prot("Протеин 2", highprice, yes).
prot("Протеин 3", middleprice, yes).
prot("Протеин 4", middleprice, not).

bca("BCA1", varysmallprice, muscletech).
bca("BCA2", varysmallprice, twinlab).
bca("BCA3", highprice, twinlab).

gein("Гейнер 1", middleprice, twinlab).
gein("Гейнер 2", varysmallprice, twinlab).
gein("Гейнер 3", varysmallprice, muscletech).
gein("Гейнер 4", highprice, muscletech).
gein("Гейнер 5", highprice, twinlab).

vita("Супрадин 1", highprice).
vita("Тријех ", varysmallprice).

creatine("Креатин моногидрат 1", highprice, yes).
creatine("Креатин моногидрат 2", varysmallprice, not).
creatine("Креатин эфир3", varysmallprice, yes).

lcarnitine("Л-карнитин 1", middleprice, lowfat).
lcarnitine("Л-карнитин 2", middleprice, highfat).
lcarnitine("Л-карнитин 3", varysmallprice, highfat).
lcarnitine("Л-карнитин 4", highprice, lowfat).

bad("Био-1", varysmallprice, yes).
bad("НЛ", highprice, yes).
bad("Био-2", highprice, not).

triba("Трибулус 1", varysmallprice, not).
triba("Трибулус 2", highprice, yes).

```

Рисунок 71 – Список фактов БЗ

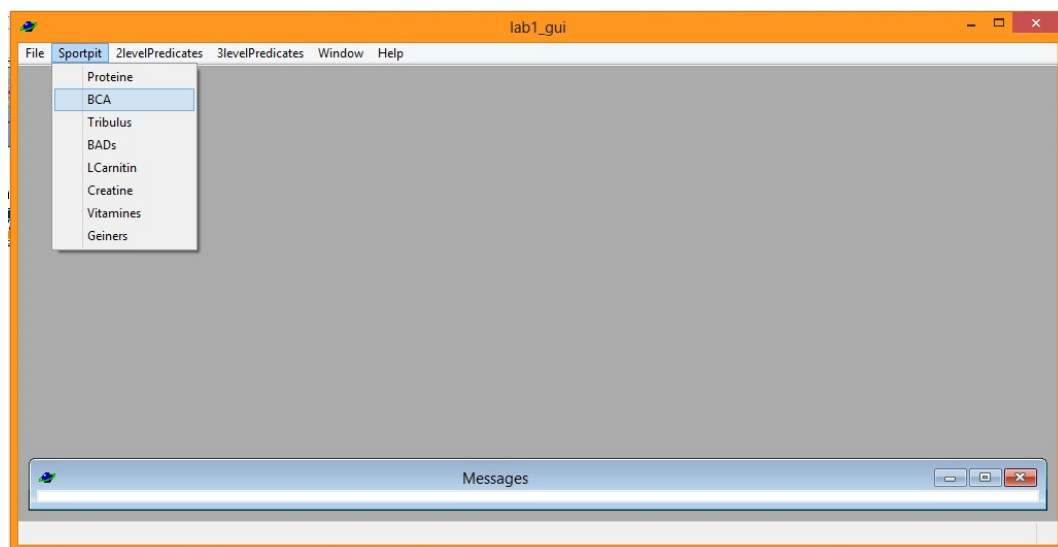


Рисунок 72 – Меню

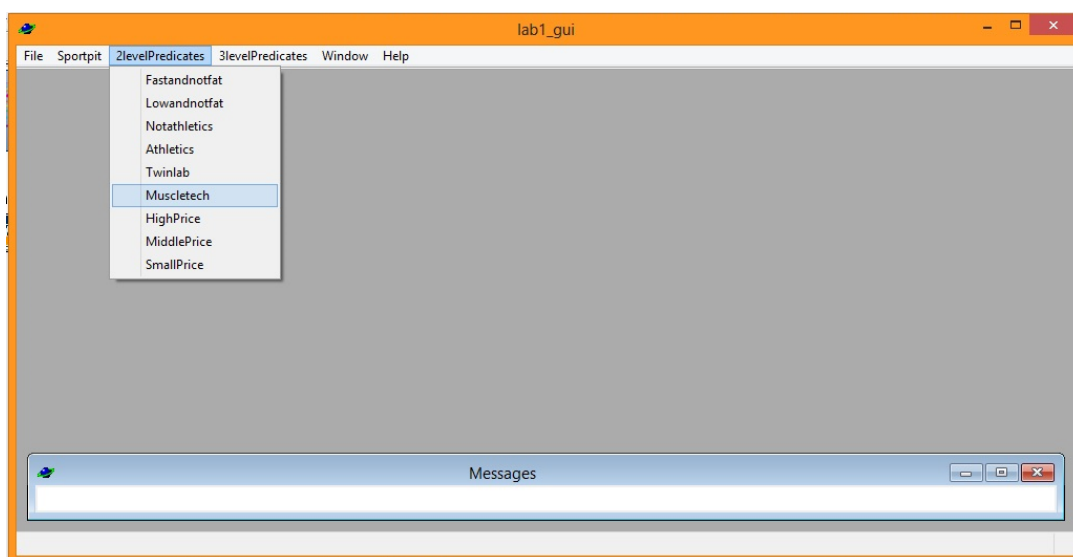


Рисунок 73 – Меню (предикаты второго уровня)

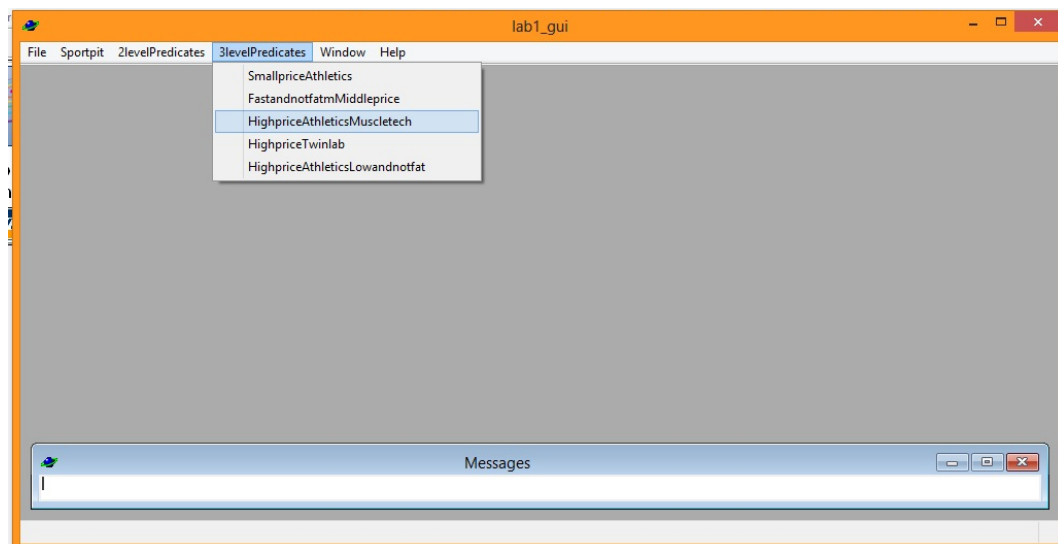


Рисунок 74 – Меню (предикаты третьего уровня)

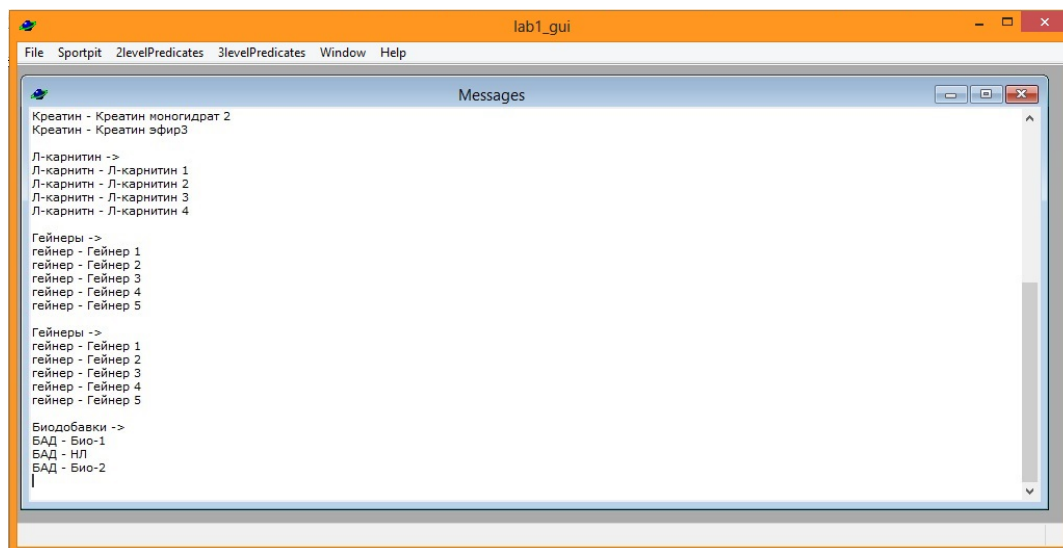


Рисунок 75 – Работа программы

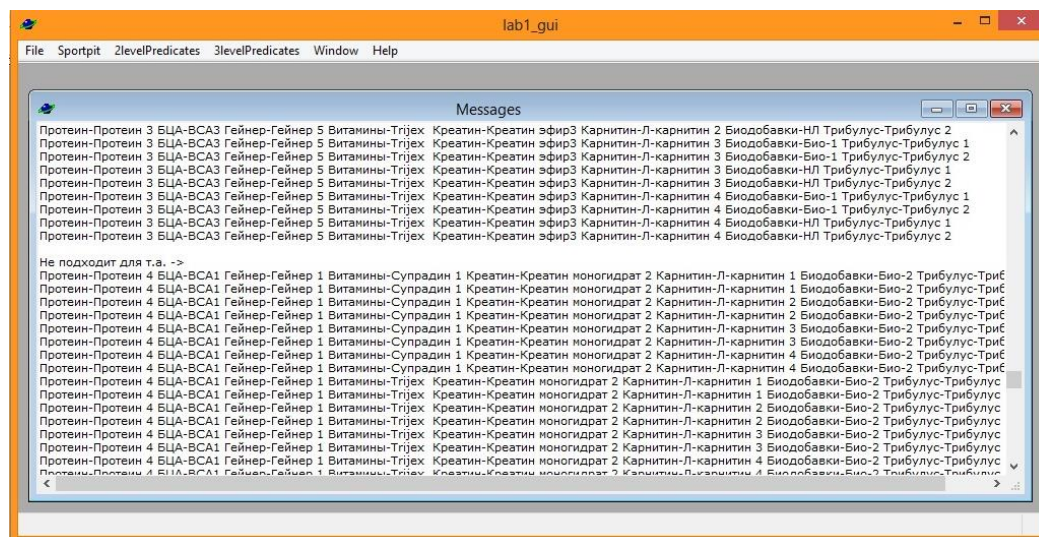


Рисунок 76 – Работа программы (комплексы спортивного питания, не подходящие для занятий тяжелой атлетикой)

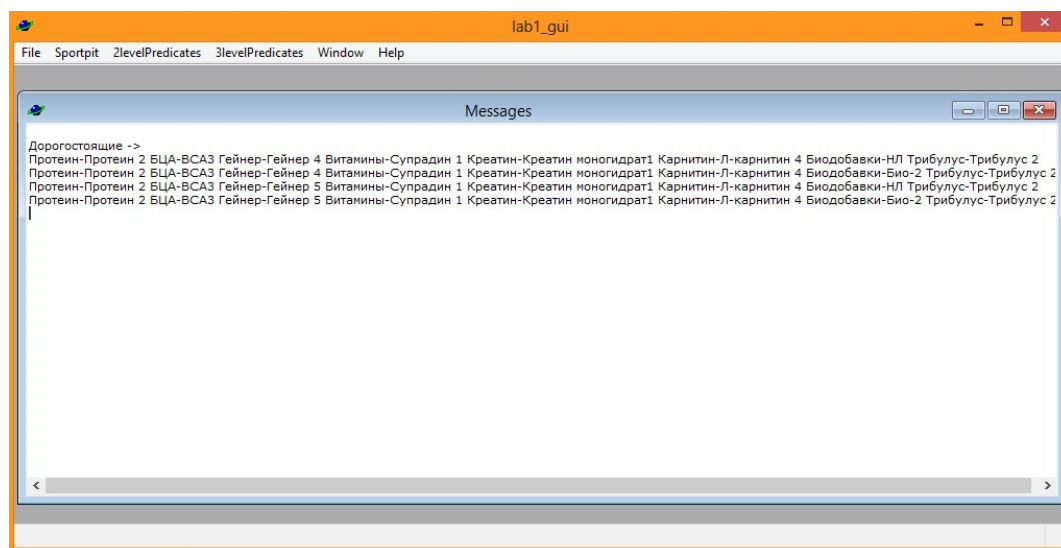


Рисунок 77 – Работа программы (дорогостоящие комплексы спортивного питания)

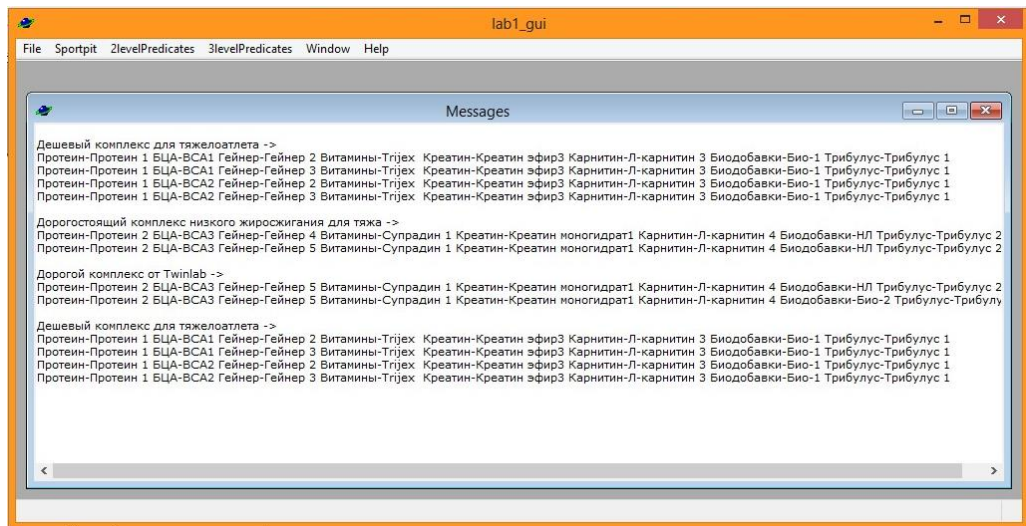


Рисунок 78 – Работа программы (несколько вариантов поиска специализированных комплексов спортивного питания)

Ниже представлен исходный код на языке Visual Prolog.

% Copyright

```
implement taskWindow
inherits applicationWindow
open core, vpiDomains, hanoi

%%%%%%%%%%%%%%

domains
    price = varysmallprice;middleprice;highprice.
    beauty = yes;notyes.
    fat = lowfat;highfat.
    procreator = muscletech;twinlab.

class facts - familyDB
    prot : (string Name, price Price, beauty Beauty).
    gein : (string Name, price Price, procreator Procreator).
    bca : (string Name, price Price, procreator Procreator).
    vita : (string Name, price Price).
    creatine : (string Name, price Price, beauty Beauty).
    lcarnitine : (string Name, price Price, fat Fat).
    bad : (string Name, price Price, beauty Beauty).
    tribulus : (string Name, price Price, beauty Beauty).

% 2 level predicates
class predicates
    varysmallpricepack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName
, string PLcarnName, string BadName, string TribName) nondeterm anyflow.
    clauses
        varysmallpricepack(ProtName,GeinName,BcaName,VitaName,CreatName,PLcarnName,BadName,TribName) :-
            prot(ProtName,varysmallprice,_,_),
            gein(GeinName,varysmallprice,_),
            bca(BcaName,varysmallprice,_),
            vita (VitaName,varysmallprice),
            creatine (CreatName,varysmallprice,_),
            lcarnitine (PLcarnName,varysmallprice,_),
            bad (BadName,varysmallprice,_,_),
            tribulus (TribName,varysmallprice,_).
    class predicates
        middlepricepack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName, s
tring PLcarnName, string BadName, string TribName) nondeterm anyflow.
```

```

clauses
  middlepricepack(ProtName,GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName,
TribName) :-
  prot(ProtName,middleprice,_,_),
  gein(GeinName,middleprice,_,_),
  bca(BcaName,middleprice,_,_),
  vita (VitaName,middleprice),
  creatine (CreatName,middleprice,_,_),
  lcarnitine (PLcarnName,middleprice,_,_),
  bad (BadName,middleprice,_,_),
  tribulus (TribName,middleprice,_,_).
class predicates
  highpricepack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName, string PLcarnName, string BadName, string TribName) nondeterm anyflow.
clauses
  highpricepack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName) :-
  prot(ProtName,highprice,_,_),
  gein(GeinName,highprice,_,_),
  bca(BcaName,highprice,_,_),
  vita (VitaName,highprice),
  creatine (CreatName,highprice,_,_),
  lcarnitine (PLcarnName,highprice,_,_),
  bad (BadName,highprice,_,_),
  tribulus (TribName,highprice,_,_).
class predicates
  amdpack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName, string PLcarnName, string BadName, string TribName) nondeterm anyflow.
clauses
  amdpack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName,Key BadName bName,
TribName) :-
  prot(ProtName,_,_,_),
  gein(GeinName,_,_,amd),
  bca(BcaName,_,_,amd),
  vita (VitaName,_,_),
  creatine (CreatName,_,_,_),
  lcarnitine (PLcarnName,_,_,_),
  bad (BadName,_,_,_,_),
  tribulus (TribName,_,_,_,_).
class predicates
  nvidiapack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName, string PLcarnName, string BadName, string TribName) nondeterm anyflow.
clauses
  nvidiapack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName) :-
  prot(ProtName,_,_,_,_),
  gein(GeinName,_,_,nvidia),
  bca(BcaName,_,_,nvidia),
  vita (VitaName,_,_),
  creatine (CreatName,_,_,_),
  lcarnitine (PLcarnName,_,_,_,_),
  bad (BadName,_,_,_,_,_),
  tribulus (TribName,_,_,_,_,_).
class predicates
  nicepack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName, string PLcarnName, string BadName, string TribName) nondeterm anyflow.
clauses
  nicepack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName) :-
  prot(ProtName,_,_,yes,_,_),
  gein(GeinName,_,_,_,_),
  bca(BcaName,_,_,_,_),
  vita (VitaName,_,_,_),
  creatine (CreatName,_,_,yes),
  lcarnitine (PLcarnName,_,_,_,_),
  bad (BadName,_,_,yes,_,_),
  tribulus (TribName,_,_,_,_,_).
class predicates
  notnicepack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName, string PLcarnName, string BadName, string TribName) nondeterm anyflow.
clauses
  notnicepack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName) :-
  prot(ProtName,_,_,notyes,_,_),
  gein(GeinName,_,_,_,_),
  bca(BcaName,_,_,_,_),
  vita (VitaName,_,_,_),
  creatine (CreatName,_,_,notyes),

```

```

        lcarnitine (PLcarnName,_,_),
        bad (BadName,_,notyes,_),
        tribulus (TribName,_,_).
class predicates
    lowsoundpack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName, string PLcarnName, string BadName, string TribName) nondeterm anyflow.
    clauses
        lowsoundpack(ProtName,GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName) :-
            prot(ProtName,_,_,_),
            gein(GeinName,_,_),
            bca(BcaName,_,_),
            vita (VitaName,_,_),
            creatine (CreatName,_,_),
            lcarnitine (PLcarnName,_,_,lowsound),
            bad (BadName,_,_,_),
            tribulus (TribName,_,_).
class predicates
    highsoundpack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName, string PLcarnName, string BadName, string TribName) nondeterm anyflow.
    clauses
        highsoundpack(ProtName,GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName) :-
            prot(ProtName,_,_,_),
            gein(GeinName,_,_),
            bca(BcaName,_,_),
            vita (VitaName,_,_),
            creatine (CreatName,_,_),
            lcarnitine (PLcarnName,_,_,highsound),
            bad (BadName,_,_,_),
            tribulus (TribName,_,_).

% 3 level predicates
class predicates
    highprice_nice_lowsound_pack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName, string PLcarnName, string BadName, string TribName) nondeterm anyflow.
    clauses
        highprice_nice_lowsound_pack(ProtName,GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName) :-
            lowsoundpack(ProtName,GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName),
            nicepack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName),
            highpricepack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName).
class predicates
    highprice_nvidia_pack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName, string PLcarnName, string BadName, string TribName) nondeterm anyflow.
    clauses
        highprice_nvidia_pack(ProtName,GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName) :-
            nvidiapack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName),
            highpricepack(ProtName,GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName).
class predicates
    highprice_amd_nice_pack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName, string PLcarnName, string BadName, string TribName) nondeterm anyflow.
    clauses
        highprice_amd_nice_pack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName) :-
            amdpack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName),
            nicepack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName),
            highpricepack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName).
class predicates
    middleprice_amd_highsound_pack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName, string PLcarnName, string BadName, string TribName) nondeterm anyflow.
    clauses
        middleprice_amd_highsound_pack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName) :-
            amdpack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName),
            highsoundpack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName),
            middlepricepack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName).
class predicates
    varysmallprice_nice_pack : (string ProtName, string GeinName, string BcaName, string VitaName, string CreatName, string PLcarnName, string BadName, string TribName) nondeterm anyflow.

```

```

tName, string PLcarnName, string BadName, string TribName) nondeterm anyflow.
    clauses
        varysmallprice_nice_pack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName,
TribName) :-
            nicepack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName, TribName),
            varysmallpricepack(ProtName, GeinName, BcaName, VitaName, CreatName, PLcarnName, BadName,
TribName).

% all items
class predicates
    allprot : (string Name) nondeterm anyflow.
    clauses
        allprot(Name) :-
            prot(Name,_,_,_).
    class predicates
        allgein : (string Name) nondeterm anyflow.
        clauses
            allgein(Name) :-
                gein(Name,_,_).
    class predicates
        allbca : (string Name) nondeterm anyflow.
        clauses
            allbca(Name) :-
                bca(Name,_,_).
    class predicates
        allvita : (string Name) nondeterm anyflow.
        clauses
            allvita (Name) :-
                vita (Name,_).
    class predicates
        allcreatine : (string Name) nondeterm anyflow.
        clauses
            allcreatine (Name) :-
                creatine (Name,_,_).
    class predicates
        alllcarnitine : (string Name) nondeterm anyflow.
        clauses
            alllcarnitine (Name) :-
                lcarnitine (Name,_,_).
    class predicates
        allbad : (string Name) nondeterm anyflow.
        clauses
            allbad (Name) :-
                bad (Name,_,_,_).
    class predicates
        alltribulus : (string Name) nondeterm anyflow.
        clauses
            alltribulus (Name) :-
                tribulus (Name,_,_).

    class predicates
        reconsult : (string FileName).
    clauses
        reconsult(Filename) :-
            retractFactDB(familyDB),
            file::consult(Filename, familyDB).

%%%%%%%%%%%%%%

constants
    mdiProperty : boolean = true.

clauses
    new() :-
        applicationWindow::new(),
        generatedInitialize().

predicates
    onShow : window::showListener.
clauses
    onShow(_, _CreationData) :-
        _MessageForm = messageForm::display(This).

class predicates

```

```

onDestroy : window::destroyListener.
clauses
onDestroy(_).

class predicates
onHelpAbout : window::menuItemListener.
clauses
onHelpAbout(TaskWin, _MenuTag) :-
    _AboutDialog = aboutDialog::display(TaskWin).

predicates
onFileExit : window::menuItemListener.
clauses
onFileExit(_, _MenuTag) :-
    close().

predicates
onSizeChanged : window::sizeListener.
clauses
onSizeChanged(_) :-
    vpiToolbar::resize(getVPIWindow()).

%%%%%%%%%%%% all items menu
predicates
onEditAlltribulus : window::menuItemListener.
clauses
onEditAlltribulus(_Source, _MenuTag):-
    stdIO::writef("\nТрибулус ->\n"),
    alltribulus (X),
    stdIO::writef("Трибулус - % \n", X),
    fail.
onEditAlltribulus (_Source, _MenuTag).

predicates
onEditAllbad : window::menuItemListener.
clauses
onEditAllbad (_Source, _MenuTag):-
    stdIO::writef("\nБиодобавки ->\n"),
    allbad (X),
    stdIO::writef("БАД - % \n", X),
    fail.
onEditAllbad (_Source, _MenuTag).

predicates
onEditAllcarnitine : window::menuItemListener.
clauses
onEditAllcarnitine (_Source, _MenuTag):-
    stdIO::writef("\nЛ-карнитин ->\n"),
    allcarnitine (X),
    stdIO::writef("Л-карнитин - % \n", X),
    fail.
onEditAllcarnitine (_Source, _MenuTag).

predicates
onEditAllcreatine : window::menuItemListener.
clauses
onEditAllcreatine (_Source, _MenuTag):-
    stdIO::writef("\nКреатин ->\n"),
    allcreatine (X),
    stdIO::writef("Креатин - % \n", X),
    fail.
onEditAllcreatine (_Source, _MenuTag).

predicates
onEditAllvita : window::menuItemListener.
clauses
onEditAllvita (_Source, _MenuTag):-
    stdIO::writef("\nПоливитамины ->\n"),
    allvita (X),
    stdIO::writef("Витамины - % \n", X),
    fail.
onEditAllvita (_Source, _MenuTag).

predicates
onEditAllbca : window::menuItemListener.
clauses
onEditAllbca(_Source, _MenuTag):-
    stdIO::writef("\nГейнеры ->\n"),
    allbca(X),

```



```

stdIO::writef("гейнер - % \n", X),
fail.
onEditAllbca(_Source, _MenuTag).
predicates
onEditAllgein : window::menuItemListener.
clauses
onEditAllgein(_Source, _MenuTag):-
stdIO::writef("\nАминокислоты ->\n"),
allgein(X),
stdIO::writef("БЦА - % \n", X),
fail.
onEditAllgein(_Source, _MenuTag).
predicates
onEditAllprot : window::menuItemListener.
clauses
onEditAllprot(_Source, _MenuTag):-
stdIO::writef("\nПротеины ->\n"),
allgein(X),
stdIO::writef("Протеин - % \n", X),
fail.
onEditAllprot(_Source, _MenuTag).

%%%%%%%%%%%%%% 2 level predicates menu
predicates
onHighsoundpack : window::menuItemListener.
clauses
onHighsoundpack(_Source, _MenuTag):-
stdIO::writef("\nКомплекс быстрого жиросжигания ->\n"),
highsoundpack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
onHighsoundpack(_Source, _MenuTag).
predicates
onLowsoundpack : window::menuItemListener.
clauses
onLowsoundpack(_Source, _MenuTag):-
stdIO::writef("\nКомплекс низкого жиросжигания ->\n"),
lowsoundpack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
onLowsoundpack(_Source, _MenuTag).
predicates
onNotnicepack : window::menuItemListener.
clauses
onNotnicepack(_Source, _MenuTag):-
stdIO::writef("\nНе подходит для т.а. ->\n"),
notnicepack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
onNotnicepack(_Source, _MenuTag).
predicates
onNicepack : window::menuItemListener.
clauses
onNicepack(_Source, _MenuTag):-
stdIO::writef("\nПодходит для т.а. ->\n"),
nicepack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
onNicepack(_Source, _MenuTag).
predicates
onNvidiapack : window::menuItemListener.
clauses
onNvidiapack(_Source, _MenuTag):-
stdIO::writef("\nTwinlab ->\n"),
nvidiapack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
onNvidiapack(_Source, _MenuTag).
predicates

```

```

onAmdpack : window::menuItemListener.
clauses
onAmdpack(_Source, _MenuTag):-
stdIO::writef("\nMuscletech ->\n"),
amdpack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-
% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
onAmdpack(_Source, _MenuTag).
predicates
onHighpricepack : window::menuItemListener.
clauses
onHighpricepack(_Source, _MenuTag):-
stdIO::writef("\nДорогостоящие ->\n"),
highpricepack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-
% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
onHighpricepack(_Source, _MenuTag).
predicates
onMiddlepricepack : window::menuItemListener.
clauses
onMiddlepricepack(_Source, _MenuTag):-
stdIO::writef("\nСредняя стоимость ->\n"),
middlepricepack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-
% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
onMiddlepricepack(_Source, _MenuTag).
predicates
onVarysmallpricepack : window::menuItemListener.
clauses
onVarysmallpricepack(_Source, _MenuTag):-
stdIO::writef("\nДешевые ->\n"),
varysmallpricepack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-
% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
onVarysmallpricepack(_Source, _MenuTag).

%%%%%%%%%%%%%% 3 level predicates menu
predicates
onVarysmallpriceNicePack : window::menuItemListener.
clauses
onVarysmallpriceNicePack(_Source, _MenuTag):-
stdIO::writef("\nДешевый комплекс для тяжелоатлета ->\n"),
varysmallprice_nice_pack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-
% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
onVarysmallpriceNicePack(_Source, _MenuTag).
predicates
onMiddlepriceAmdHighsoundPack : window::menuItemListener.
clauses
onMiddlepriceAmdHighsoundPack(_Source, _MenuTag):-
stdIO::writef("\nСредней стоимости комплекс быстрого жиросжигания от Muscletech ->\n"),
middleprice_amd_highsound_pack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-
% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
onMiddlepriceAmdHighsoundPack(_Source, _MenuTag).
predicates
onHighpriceAmdNicePack : window::menuItemListener.
clauses
onHighpriceAmdNicePack(_Source, _MenuTag):-
stdIO::writef("\nДорогостоящий комплекс Muscletech для тяжелоатлета ->\n"),
highprice_amd_nice_pack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-
% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
onHighpriceAmdNicePack(_Source, _MenuTag).
predicates
onHighpriceNvidiaPack : window::menuItemListener.
clauses

```



```

onHighpriceNvidiaPack(_Source, _MenuTag):-
stdIO::writef("\nДорогой комплекс от Twinlab ->\n"),
highprice_nvidia_pack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-
% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
onHighpriceNvidiaPack(_Source, _MenuTag).
predicates
onHighpriceNiceLowsoundPack : window::menuItemListener.
clauses
onHighpriceNiceLowsoundPack(_Source, _MenuTag):-
stdIO::writef("\nДорогостоящий комплекс низкого жиросжигания для тяжа ->\n"),
highprice_nice_lowsound_pack(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
stdIO::writef("Протеин-% БЦА-% Гейнер-% Витамины-% Креатин-% Карнитин-% Биодобавки-% Трибулус-
% \n", Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8),
fail.
generatedInitialize : ().
clauses
generatedInitialize():-
setText("lab1_gui"),
setDecoration(titlebar([closeButton,maximizeButton,minimizeButton])),
setBorder(sizeBorder()),
setState([wsf_ClipSiblings]),
whenCreated({ :- projectToolBar::create(getVpiWindow()) }),
addSizeListener({ :- vpiToolBar::resize(getVpiWindow()) }),
setMdiProperty(mdiProperty),
menuSet(resMenu
onHighpriceNiceLowsoundPack(_Source, _MenuTag).

% This code is maintained automatically, do not update it manually. 13:54:31-26.12.2016
predicates (resourceIdentifiers::id_TaskMenu)),
addShowListener(onShow),
addSizeListener(onSizeChanged),
addDestroyListener(onDestroy),
addMenuItemListener(resourceIdentifiers::id_help_about, onHelpAbout),
addMenuItemListener(resourceIdentifiers::id_file_exit, onFileExit),
addMenuItemListener(resourceIdentifiers::id_edit_alltribulus, onEditAlltribulus),
addMenuItemListener(resourceIdentifiers::id_edit_allbad, onEditAllbad),
addMenuItemListener(resourceIdentifiers::id_edit_allcarnitine, onEditAllcarnitine),
addMenuItemListener(resourceIdentifiers::id_edit_allcreatine, onEditAllcreatine),
addMenuItemListener(resourceIdentifiers::id_edit_allvita, onEditAllvita),
addMenuItemListener(resourceIdentifiers::id_edit_allbca, onEditAllbca),
addMenuItemListener(resourceIdentifiers::id_2_highsoundpack, onHighsoundpack),
addMenuItemListener(resourceIdentifiers::id_2_lowsoundpack, onLowsoundpack),
addMenuItemListener(resourceIdentifiers::id_2_notnicepack, onNotnicepack),
addMenuItemListener(resourceIdentifiers::id_2_nicepack, onNicepack),
addMenuItemListener(resourceIdentifiers::id_2_nvidiapack, onNvidiapack),
addMenuItemListener(resourceIdentifiers::id_2_amdpack, onAmdpack),
addMenuItemListener(resourceIdentifiers::id_2_highpricepack, onHighpricepack),
addMenuItemListener(resourceIdentifiers::id_2_middlepricepack, onMiddlepricepack),
addMenuItemListener(resourceIdentifiers::id_2_varysmallpricepack, onVarysmallpricepack),
addMenuItemListener(resourceIdentifiers::id_3_varysmallprice_nice_pack, onVarysmallpriceNicePack),
addMenuItemListener(resourceIdentifiers::id_3_middleprice_amd_highsound_pack, onMiddlepriceAmdHighso
undPack),
addMenuItemListener(resourceIdentifiers::id_3_highprice_nvidia_pack, onHighpriceNvidiaPack),
addMenuItemListener(resourceIdentifiers::id_3_highprice_nice_lowsound_pack, onHighpriceNiceLowsoundPa
ck),
addMenuItemListener(resourceIdentifiers::id_edit_allgein, onEditAllgein),
addMenuItemListener(resourceIdentifiers::id_edit_allprot, onEditAllprot),
addMenuItemListener(resourceIdentifiers::id_3_highprice_amd_nice_pack, onHighpriceAmdNicePack).
% end of automatic code
end implement taskWindow

```

Список дополнительной литературы:

- 1) Ростовцев, Владимир Сергеевич. Принципы построения экспертных систем: учеб. пособие / Ростовцев, Владимир Сергеевич; ВятГУ, ФАВТ, каф. ЭВМ. - 2-е изд., перераб. и доп. - Киров: О-Краткое, 2008. - 156с. (50 экз.)
- 2) Крутиков А.К., Подковырин В.Д., Шалаев Д.А. Линейная нейронная как инструмент прогнозирования индивидуальных спортивных результатов // Синергия наук. 2018. № 25. – С. 657-663. – URL: <http://synergy-journal.ru/archive/article2622>
- 3) Крутиков А.К. Прогнозирование индивидуальных результатов в легкой атлетике спортсменов высших спортивных разрядов с использованием искусственных нейронных сетей // Universe: технические науки : электрон. научн. журн. 2019. № 10 (67). URL: <https://7universum.com/ru/tech/archive/item/7898> (дата обращения: 04.09.2025).
- 4) Хабаров С.П. Интеллектуальные информационные системы. PROLOG- язык разработки интеллектуальных и экспертных систем: учебное пособие / С.П.Хабаров.- СПб. СПбГЛТУ, 2013.- 138 с
- 5) Представление знаний в экспертных системах : учебное пособие / сост. В.А. Морозова, В.И. Паутов.— Екатеринбург : Изд-во Урал. ун-та, 2017.— 120 с. ISBN 978-5-7996-2037-0
- 6) Погружение в Марковские цепи URL: <https://habr.com/ru/companies/otus/articles/732424/>
- 7) Искусство моделирования. Краткое введение в цепи Маркова URL: <https://habr.com/ru/articles/455762/>
- 8) Панченко, Т. В. Генетические алгоритмы [Текст] : учебно-методическое пособие / под ред. Ю. Ю. Тарасевича. — Астрахань : Издательский дом «Астраханский университет», 2007. — 87 [3] с
- 9) Вирсански Э. Генетические алгоритмы на Python / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2020. – 286 с.: ил.
- 10) Howard Demuth, Mark Beale, Martin Hagan, Neural Network Toolbox™ 6 User's Guide
- 11) Крутиков А.К. Использование нейронной сети прямого распространения для прогнозирования спортивных результатов в индивидуальных видах спорта НАУЧНОЕ СООБЩЕСТВО СТУДЕНТОВ. МЕЖДИСЦИПЛИНАРНЫЕ ИССЛЕДОВАНИЯ Электронный сборник статей

по материалам XLVI студенческой международной научно-практической конференции. Том 11(46). 2018

12) Крутиков А.К., Мельцов В.Ю. Применение методов искусственного интеллекта при прогнозировании событий в индивидуальных видах спорта [Текст] // Разработка и применение наукоёмких технологий в интересах модернизации современного общества: монография. Выпуск 92. – Уфа: Аэтерна, 2024. – с. 23-82

13) Кельберт М. Я., Сухов Ю. М. Вероятность и статистика в примерах и задачах. Т.II: Марковские цепи как отправная точка теории случайных процессов и их приложения. М.: МЦНМО, 2009.

14) Бородачёв, С. М. Теория принятия решений: учебное пособие / С. М. Бородачёв. - Екатеринбург : Изд-во Урал, ун-та, 2014. - 124 с

15) Ростовцев В.С. Искусственные нейронные сети: учебник / В.С. Ростовцев. – Киров: Изд-во ВятГУ, 2014. – 208 с.

16) Ростовцев В.С. Теория принятия решений: методические указания к самостоятельным и лабораторным работам.- Киров: Изд-во ВятГУ, 2020.-49 с.