Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

ИНТЕЛЛЕКТУАЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Методические указания к лабораторным работам и организации
самостоятельной работы для студентов направления
«Программная инженерия»
(уровень магистратуры)

2018

Голубева Александра Александровна

Интеллектуальные вычислительные системы: Методические указания к лабораторным работам и организации самостоятельной работы для студентов направления «Программная инженерия» (уровень магистратуры) / А.А. Голубева. – Томск, 2018. – 55 с.

Оглавление

1 Введение

Целью лабораторных и самостоятельных работ в рамках изучения дисциплины «Интеллектуальные вычислительные системы» является формирование у студентов, обучающихся по направлению «Программная инженерия», навыков, позволяющих формулировать и решать задачи построения и обучений систем нечёткого вывода, экспертных систем, а также знакомство обучающихся с методами и подходами нечеткого моделирования.

Лабораторные и самостоятельные работы являются важной составляющей в изучении дисциплины и состоят из следующих видов деятельности: проработка лекционного материала для подготовки к лабораторным и самостоятельным работам, выполнение индивидуальных вариантов заданий в рамках лабораторных практикумов. Лабораторные и самостоятельные работы направлены на изучение основных понятий и принципов систем искусственного интеллекта.

2 Методические указания к проведению лабораторных работ

2.1 Лабораторная работа «Формирование базы правил нечеткой системы моделирования нелинейной системы»

Цель работы

Знакомство с методологией нечеткого моделирования. Описание заданной нелинейной функции множеством ЕСЛИ-ТО правил.

Порядок выполнения работы

1. Знакомство с методологией нечеткого моделирования

Описание предметной области может быть проведено посредством лингвистических переменных и правил естественного языка, содержащих качественную оценку ситуации. Основой для описания ситуации является нечеткое высказывание следующего вида:

$$x_i \text{ есть } X_i \text{ или } xi = X_i,$$

где x_i – некоторая величина, X_i – элемент терм-множества лингвистической переменной из исследуемой предметной области.

Нечеткая система выполняет отображение из входного пространства $A \subset \Re^m$ в выходное пространство $B \subset \Re^m$. Такая система является системой типа «много_входов – много_выходов» (MIMO – multiple_input-multiple_output). Если система имеет *m* входов и r выходов и входное и выходное пространства являются многомерными, то входное пространство определяется как $A = A_1 \times ... \times A_m$, а выходное пространство – как $B = B_1 \times ... \times B_r$, где $A_i, B_j \subset \Re$. Обозначим $a = [\, a_1\ a_2 \ldots a_m]^T$ и $b = [\, b_1\ b_2 \ldots b_r]^T$ как входной и выходной векторы, соответственно. Отображение вход/выход может быть представлено как множество нечетких правил типа «ЕСЛИ-ТО». Каждое правило состоит из двух частей: условной и заключительной. Антецедент или условная часть (ЕСЛИ-часть) содержит утверждение относительно значений входных переменных, в консеквенте или заключительной части (ТОчасти) указываются значения, которые принимают выходные переменные. Таким образом, нечеткая система типа «много_входов – много_выходов» может быть задана нечеткими правилами следующего вида:

Правило 1: **ЕСЛИ** $a_1 = A_{11}$ **И** $a_2 = A_{21} \ldots a_m = A_{m1}$
ТО $b_1 = B_{11}$ **И** $b_2 = B_{21}$ **И**…, **И** $b_r = B_{r1}$;

Правило 2: **ЕСЛИ** $a_1 = A_{12}$ **И** $a_2 = A_{22}$ … $a_m = A_{m2}$
ТО $b_1 = B_{12}$ **И** $b_2 = B_{22}$ **И**… **И** $b_r = B_{r2}$;

……

…

Правило n: **ЕСЛИ** $a_1 = A_{1n}$ **И** $a_2 = A_{2n}$ … $a_m = A_{mn}$
ТО $b_1 = B_{1n}$ **И** $b_2 = B_{2n}$ **И** … **И** $b_r = B_{rn}$;

где $a_1, a_2, \ldots, a_m$ – входные переменные, $b_1, b_2, \ldots, b_r$ – выходные переменные, A_{it} и B_{js} – нечеткие области определения входных и выходных переменных, которые определены на универсальных *множествах* $X_1, X_2, \ldots, X_m, Y_1, Y_2, \ldots, Y_r$, соответственно. Каждая нечеткая область A_{it} связана с функцией принадлежности $\mu_{Ait}(a_i)$.

Вход А нечеткой системы активизирует каждое из правил, хранимых в нечеткой ассоциативной памяти. Чем больше вход А соответствует антецеденту *i*-го правила, тем больше выход соответствует консеквенту этого правила.

Весьма популярными в практическом применении в настоящее время являются нечеткие системы типа «много_входов – один_выход».

Система такого типа выполняет отображение из входного пространства

$A \subset \Re^m$ в выходное пространство $B \subset \Re$. Известно три основных типа нечетких систем «много_входов –один_выход»:

1) Системы типа Мамдани имеют правило *i*: **ЕСЛИ** $a_1 = A_{1i}$ **И** $a_2 = A_{2i}$ … **И** $a_m = A_{mi}$ **ТО** $b = B_i$;

2) Другой тип – системы типа Сугено с правилами следующего вида: правило *i*: **ЕСЛИ** $a_1 = A_{1i}$ **И** $a_2 = A_{2i}$ … $a_m = A_{mi}$ **ТО** $b = f_i(a_1, \ldots a_m)$; где f_i – функция, определенная на переменных $a_1 \ldots a_m$.

3) Системы типа синглтон задаются правилами вида: правило i: **ЕСЛИ** $a_1 = A_{1i}$ **И** $a_2 = A_{2i}$ **И** … **И** $a_m = A_{mi}$ **ТО** $b = r_i$, где r_i – действительное число.

Для описания отображения входного вектора a в значение b используются методы нечеткой логики, например, аппроксимация Мамдани или метод, основанный на формальном логическом доказательстве. В процессе вывода участвуют операции конъюнкции и дизъюнкции. Задание этих операций на основе триангулярных норм позволяет более гибко настраивать нечеткую систему на исследуемую предметную область.

В общем случае процесс создания нечетких систем состоит из следующих шагов:

1) определение входных и выходных переменных системы;
2) задание функций принадлежности каждой переменной;

3) определение нечетких правил;

4) настройка параметров функций принадлежности и нечетких правил.

При разработке базы правил необходимо руководствоваться следующими принципами:

1) в базе правил существует правило для всяких сочетаний A_{1i}, A_{2i}, …, A_{mi}, B_i;

2) нет двух и более правил с одинаковым антецедентом и различным консеквентом.

Для осуществления вывода в такой системе можно воспользоваться композиционным правилом. Однако предварительно нужно выполнить операции конъюнкции (И) и далее операцию объединения (агрегации) n правил.

Пусть на вход системы поступают четкие значений x_1, x_2. Требуется определить четкий выход y. Для этого необходимо выполнить следующие операции:

1) фаззификация – для каждого правила вычисляется значения

$\mu A\ i1\ (x1)$ и $\mu\ A\ i2\ (x2)$;

2) конъюнкция – объединение посылок в антецеденте каждого правила, используя *t*-нормальную функцию, получим

$$T\ (\mu_{A1}\ (x_1),\ \mu_{A2}(x_2));$$

3) импликация – $I\ (T\ (\mu_{Ali}\ (x_1),\ \mu_{A2i}\ (x_2)),\ \mu_{Bi}\ (y))$;

4) агрегация – получение нечеткого выходного значения из множества объединенных правил, то есть определение итоговой функции принадлежности $\mu_B\ (y)$;

5) дефаззификация - преобразование итоговой функции принадлежности $\mu_B\ (y)$ в четкое значение *y*.

Структура нечеткой системы моделирования представлена на рис.1.

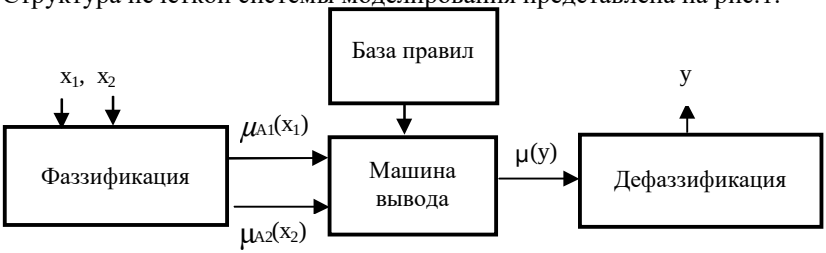

Рис. 1 – Структура нечеткой системы моделирования

2. Описание заданной нелинейной функции множеством ЕСЛИ-ТО правил

Пусть дана *функция F(x) = sin(x)* и следующие области изменения 0<x<2*π, -1<y<1 (рис. 2).

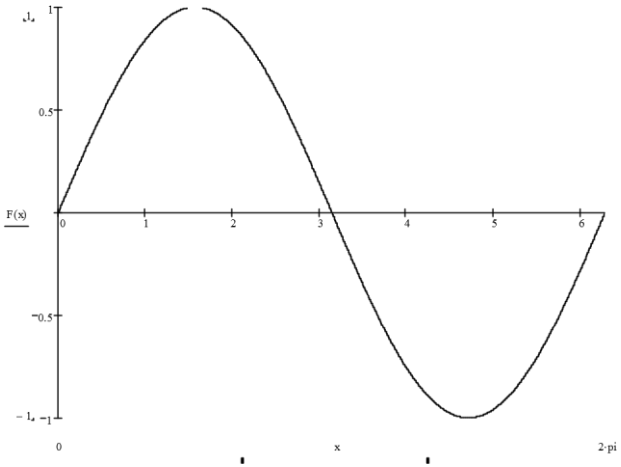

Рис. 2 – Пример нелинейной функции

Лингвистические переменные, описывающие *x* и F, определены на следующем множестве термов: {очень малая, малая, средняя, большая, очень большая}. Функции принадлежности для указанных термов приведены на рис. 3 и 4. Тогда нечеткая система типа Мамдани для моделирования указанной нелинейной функции будет задана следующей базой правил:

ЕСЛИ *x = очень малая* **ТО** *F = средняя,*
ЕСЛИ *x = малая* **ТО** F = *очень большая.*
ЕСЛИ *x = средняя* **ТО** *F = средняя,*
ЕСЛИ *x = большая* **ТО** *F = очень малая,*
ЕСЛИ *x = очень большая* **ТО** *F = средняя*

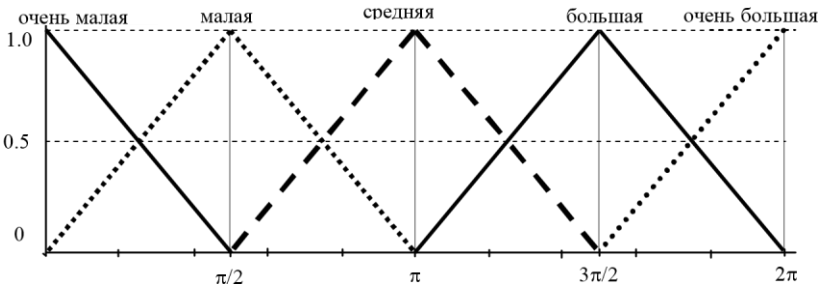

Рис. 3 – Пример функций принадлежности для переменной x

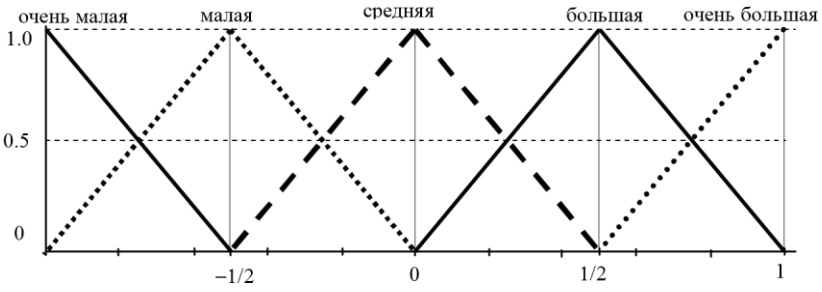

Рис. 4 – Пример функций принадлежности для переменной y

Нечеткую систему типа синглтон можно задать следующей базой правил:

ЕСЛИ *x = очень малая* **ТО** *F = 0*
ЕСЛИ *x = малая* **ТО** *F = 1*
ЕСЛИ *x = средняя* **ТО** *F = 0*
ЕСЛИ *x = большая* **ТО** F = -1
ЕСЛИ *x = очень большая* **ТО** *F = 0.*

Форма проведения

Выполнение индивидуального задания:

1. Используя пакет Mathcad, построить график выбранной нелинейной функции.

2. Выбранную нелинейную функцию описать базами правил типов Мамдани и синглтон для лингвистических переменных, определенных на множестве из пяти, семи, девяти и одиннадцати термов. Примечание.

- Если z выходит за заданные границы, то z принимает значение равное значению минимальной или максимальной границы, соответственно.
- π = 3,14159265358979323846264338327 95

Варианты заданий

№	Функция	Область изменения		
		x	y	z
1	$z = {}^{x2} + \sin(y - \pi/2)$	-2≤ x≤ 2	- π≤ y≤ π	-1≤ z≤ 4
2	$z = \cos(x) + \sin(y - \pi/2)$	- π ≤x≤ π	- π ≤y≤ π	- 2 ≤z≤ 2
3	$z = x * \sin(y)$	-π≤x≤ π	-π≤y≤ π	-3≤z≤ 3
4	$z = x^2 * y^2$	-2≤x≤ 2	-2≤y≤ 2	-2≤z≤ 2
5	$z = x * y$	-4≤x≤ 4	-4≤y≤ 4	-4≤z≤4
6	$z = (x - y) * y + 1$	-3≤x≤ 3	-3≤y≤ 3	-4≤z≤ 3
7	$z = y * \sin(x + y)$	-π/2≤x≤π/2	π/2≤ y≤ π/2	-0.5≤ z≤ 1.5
8	$z = \sin(2*x/\pi) * \sin(2*y/\pi)$	-5≤ x≤ 5	-5≤ y≤ 5	-1≤ z≤ 1
9	$z = y * \sin(x)$	-π/2≤x≤π/2	π/2≤ y≤ π/2	-1≤ z≤ 1.5
10	$z = y * \cos(x)$	-π/2≤x≤π/2	π/2≤ y≤ π/2	-1≤ z≤ 1.5
11	$z = x * \cos(x) + y * \sin(y)$	- 2 ≤ x≤ 2	- π/2≤ y≤ π/2	-0.5≤ z≤ 2

12	$z = x^2 - y^2$	$-3 \le x \le 3$	$-3 \le y \le 3$	$-4 \le z \le 3$

Форма отчетности

Отчет должен содержать следующие обязательные пункты:

1) цель работы;

2) графическое и формульное представление моделируемой функции;

3) базы правил двух типов нечетких систем (Мамдани и синглтон) для пяти, семи, девяти и одиннадцати термов с графиками соответствующих функций принадлежности;

4) выводы о проделанной работе.

2.2 Лабораторная работа «Нечеткая система типа синглтон. Создание базы правил»

Цель работы

Разработка алгоритма нечеткого вывода для моделей типа синглтон. Формирование базы правил

Порядок выполнения работы

1. Знакомство с нечеткими системами типа синглтон

В работе рассматривается нечеткая система типа синглтон, в которой n входных переменных, m нечетких правил, каждое из которых имеет следующий вид:

правило j: **ЕСЛИ** $x_1=A_{1j}$ **И** $x_2=A_{2j}$ **И** … **И** $x_n=A_{nj}$ **ТО** r_j,

где r_j - действительное число, $r_j \in \Re$. Нечеткая система осуществляет отображение $F: \Re^n \to \Re$, заменяя оператор нечеткой конъюнкции произведением, а оператор агрегации нечетких правил - сложением. Тогда выходное значение F(**x**) вычисляется следующим образом:

$$F(\mathbf{x})=\frac{\sum_{j=1}^{m} r_j \cdot \prod_{i=1}^{n} \mu_{Aij}(x_i)}{\sum_{j=1}^{m} \prod_{i=1}^{n} \mu_{Aij}(x_i)},$$

где $x = [x_1,...,x_n]^T \in \Re^n$, $\mu_{Aij}(x_j)$ - функция принадлежности нечеткого понятия A_{ij}.

2. Знакомство со способами задания функции принадлежности

Существует множество типов и форм функций принадлежности термов лингвистических переменных. Рассмотрим четыре из них наиболее часто используемые в практических и теоретических работах: треугольные, трапециевидные, параболические, гауссовы.

Треугольная функция принадлежности $\mu_{\tilde{a}}(x)$ аналитически описывается следующим образом:

$$\mu_{\tilde{a}}(x) = \begin{cases} \dfrac{x-a_1}{a_0-a_1}, a_1 \le x \le a_0, \\ \dfrac{x-a_2}{a_0-a_2}, a_0 \le x \le a_2, \\ 0 \end{cases}$$

(0, в других случаях)

Трапециевидная функция принадлежности $\mu_{\tilde{u}}(x)$ аналитически описывается следующим образом:

$$\mu_{\tilde{u}}(x) = \begin{cases} \dfrac{x-u_1}{u_{01}-u_1}, u_1 \le x \le u_{01}, \\ 1, u_{01} \le x \le u_{02}, \\ \dfrac{x-u_2}{u_{02}-u_2}, u_{02} \le x \le u_2, \\ 0 \end{cases}$$

(0, в других случаях).

Параболическая функция принадлежности аналитически задается следующим образом:

$$\mu_o(x) = 1 - \left(\frac{x-a}{b}\right)^2, (a-b) \le x \le (a+b)$$

Гауссова функция принадлежности $\mu_{\tilde{u}}(x)$ описывается следующим образом:

$$\mu(x) = \exp\left(-\left(\frac{x - m_0}{\sigma_0}\right)^2\right)$$

3. Создание базы правил нечеткой модели типа синглтон

В любой среде разработки создать приложение для работы с нечеткими системами со следующим функциями:

a) задание и изменение таких параметров нечеткой системы, как количество термов, на которые разбиваются входы, и параметры функций принадлежности;

b) формирование базы правил в зависимости от количества термов и параметров функций принадлежности.

Приложение должно включать в себя форму (формы) со следующими компонентами:

1) графический компонент для отображения графиков функций принадлежностей (рис. 6);

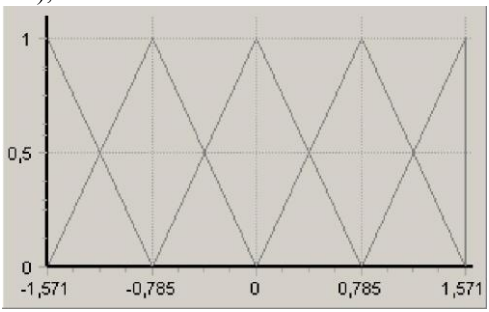

Рис. 6 – Компонент для отображения графиков функций принадлежностей

2) компонент для отображения параметров функций принадлежности (рис. 7);

	ФП	1	2	3
▶	1	-1,570796	-1,570796	-0,785398
	2	-1,570796	-0,785398	0
	3	-0,785398	0	0,785398
	4	0	0,785398	1,570796
	5	0,785398	1,570796	1,570796
*				

Рис. 7 – Компонент для отображения параметров функций принадлежности

3) компонент для отображения базы правил (рис. 8).

№	Пер-я 1	Пер-я 2	Значение
1	1	1	1,570796
2	1	2	0,785398
3	1	3	0
4	1	4	-0,785398
5	1	5	-1,570796
6	2	1	1,11072
7	2	2	0,55536
8	2	3	0
9	2	4	-0,55536
10	2	5	-1,11072
11	3	1	0
12	3	2	0

Рис. 8 – Компонент для отображения базы правил

Форма отчетности

Отчет должен содержать следующие обязательные пункты:

1) цель работы;

2) моделируемая нелинейная функция;

3) описание созданных классов, методов, глобальных переменных.

4) примеры работы программы при различных значениях количества термов и параметров функций принадлежностей;

5) выводы по лабораторной работе.

2.3 Лабораторная работа «Нечеткая система типа синглтон. Создание машины нечеткого вывода»

Цель работы

Разработка алгоритмов нечеткого вывода для моделей типа синглтон. Создание подсистемы нечеткого вывода. Вычисление ошибки вывода нечетких систем.

Порядок выполнения работы

1. Формирование таблицы наблюдений

Области определения входных переменных разделяются на *k* интервалов (*k* задается через компонент на форме), в результате чего имеем *k+1* значение входа . Путем полного перебора сочетаний значений входов и вычисления значений выхода в получившихся точках, получаем таблицу наблюдений из $N = (k+1)^2$ строк (в случае системы 2ISO). Например, для функции $z = y * sin(x)$ для $k = 3$ таблица наблюдений будет иметь вид, представленный на рис. 9.

	X	Y	Z
1	-1,570796	-1,570796	1,570796
2	-1,570796	-0,523599	0,523599
3	-1,570796	0,523599	-0,523599
4	-1,570796	1,570796	-1,570796
5	-0,523599	-1,570796	0,785398
6	-0,523599	-0,523599	0,261799
7	-0,523599	0,523599	-0,261799
8	-0,523599	1,570796	-0,785398
9	0,523599	-1,570796	-0,785398
10	0,523599	-0,523599	-0,261799
11	0,523599	0,523599	0,261799
12	0,523599	1,570796	0,785398
13	1,570796	-1,570796	-1,570796
14	1,570796	-0,523599	-0,523599
15	1,570796	0,523599	0,523599
16	1,570796	1,570796	1,570796

Рис. 9 – Пример таблицы наблюдений

2. Нечеткий вывод

Выходное значение $F(x)$ вычисляется следующим образом:

$$F(\boldsymbol{x})=\frac{\sum_{j=1}^{m} r_j \cdot \prod_{i=1}^{n} \mu_{Aij}(x_i)}{\sum_{j=1}^{m} \prod_{i=1}^{n} \mu_{Aij}(x_i)}$$

где x_i – значение i-го входа, r_j – значение консеквента в j-м правиле, $\mu_{Aij}(x_j)$ - функция принадлежности терму A_{ij}, n – количество входов, m – количество правил.

3. Ошибки нечеткого вывода

Качество нечеткого вывода можно оценить при помощи значения ошибки вывода – разницы между значениями выходной переменной из таблицы наблюдений f(x) и значениями F(x), полученными нечеткой системой. Выделим три типа ошибки вывода:

Среднеквадратичная ошибка (СКО): $\sqrt{\frac{\sum_{i}^{N}(f(x_i)-F(x_i))^2}{N}}$;

Средняя абсолютная ошибка (САО) $\frac{\sum_{i}^{N}\left|f(x_i)-F(x_i)\right|}{N}$;

Максимальная ошибка (МО) $\max\left(\left|f(x_i)-F(x_i)\right|\right).$

Варианты заданий

Дополнить приложение, созданное в лабораторной работе «Нечеткая система типа синглтон. Создание машины нечеткого вывода» следующими функциями*:*

- формирование таблицы наблюдений в зависимости от количества интервалов, на которые разбиваются области определения входных переменных;

- вычисление значения выхода нечеткой системы в зависимости от вводимых значений входов;
- вычисление ошибок нечеткого вывода по сформированной таблице наблюдений.

1. Добавить следующие компоненты:
- компонент для отображения таблицы наблюдений;
- компоненты для отображения значений трех типов ошибки нечеткой системы;
- компонент для отображения результатов работы системы по всем строкам таблицы наблюдений (таблица с полями: *x1, x2, f(x1,x2),F(x1,x2),* ошибка);
- компоненты для вычисления значения выхода.

2. Вычислить значения среднеквадратичной, средней абсолютной и максимальной ошибок нечеткого вывода для различных значений параметра k для четырех баз правил (для пяти, семи, девяти и одиннадцати термов). Результаты представить в виде таблицы.

Форма отчетности

Отчет должен содержать следующие обязательные пункты:

1) цель работы;
2) моделируемая нелинейная функция;
3) описание созданных классов, методов, глобальных переменных;
4) примеры работы программы;
5) таблица значений ошибок;
6) выводы по лабораторной работе.

2.4 Лабораторная работа «Идентификация нечеткой системы с помощью генетического алгоритма. Генерация начальной популяции. Оператор селекции. Операторы скрещивания и мутации»

Цель работы

Знакомство с основными принципами работы генетического алгоритма и особенностями его использования для настройки параметров функций принадлежности. Создание начальной популяции и применение оператора селекции, скрещивания и мутации в задаче идентификации нечетких систем.

Порядок выполнения работы

Основанием для выполнения настоящей работы являются результаты, полученные при выполнении лабораторной работы «Нечеткая система типа синглтон. Создание базы правил» и «Нечеткая система типа синглтон. Создание машины нечеткого вывода».

1. Основные принципы работы генетического алгоритма

Генетический алгоритм основан на механизмах естественного отбора и наследования. В нем используется принцип выживания наиболее приспособленных особей. Преимущества алгоритма перед другими методами оптимизации заключаются в параллельной обработке множества альтернативных решений.

Генетический алгоритм работает с популяцией особей (хромосом), каждая из которых представляет собой упорядоченный набор параметров задачи, подлежащих оптимизации. Основной характеристикой каждой особи является ее мера приспособленности (фитнесфункция). Более приспособленные особи участвуют в воспроизводстве потомства путем скрещивания с другими особями из популяции, при этом гены потомков получают свои значения путем обмена генов родителей. Наименее приспособленные особи постепенно исчезают из популяции в процессе эволюции. Таким образом, из поколения в поколение по всей популяции распространяются лучшие характеристики. Иногда в популяции происходят мутации – случайные изменения в генах некоторых хромосом, за счет чего происходит внесение некоторой случайности в процесс работы генетического алгоритма, что снижает вероятность застревания в локальных минимумах.

Работа генетического алгоритма представляет собой итерационный процесс, который продолжается до выполнения одного из условий останова:

- выполнение заданного числа поколений;
- достижение заданной точности;
- прекращение улучшения популяции.

Общая схема генетического алгоритма представлена на рис. 10.

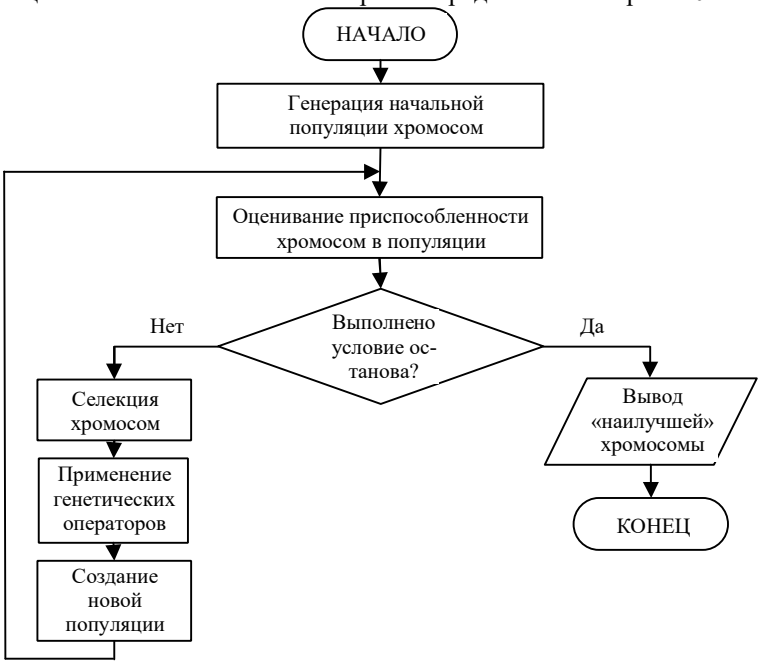

Рис. 10 – Блок-схема генетического алгоритма

2. Формирование начальной популяции

Основным понятием в генетическом алгоритме является понятие «хромосомы» Хромосома – упорядоченный набор генов – закодированных параметров задачи. В нашем случае ген кодирует один параметр функции принадлежности. Хромосома будет соответствовать отдельному потенциальному решению задачи. В нулевой ген помещается значение фитнес-функции, которое определяется ошибкой системы. Значения ненулевых генов соответствуют значениям параметров функций принадлежности. На рис. 11 представлена структура хромосомы для задачи

настройки треугольных функций принадлежности, где a_t^j, b_t^j, c_t^j – значения параметров i-го терма j-й переменной, t – количество термов переменной.

ошибка	a_1^1	b_1^1	c_1^1	…	a_t^j	b_t^j	c_t^j

Рис. 11 – Структура хромосомы

В зависимости от типа функций принадлежности должны соблюдаться специфические условия включения хромосомы в популяцию. Например, для треугольных функций принадлежности:

1) $a_j^i \le b_j^i \le c_j^i$;

2) $b_{i-1}^j \le b_j^i \le b_{i+1}^j$;

3) функции принадлежности должны покрывать весь универсум, на котором они определены (не должно быть разрывов между соседними функциями принадлежности).

В лабораторной работе необходимо реализовать следующий способ задания начальной популяции: одна хромосома соответствует равномерному распределению параметров по области определения входных переменных (параметры функций принадлежности из лабораторных работ №3 и №4), гены остальных хромосом задаются случайным образом (при создании каждой особи учитываются правила включения в популяцию).

3. Оператор селекции

Селекция хромосом заключается в выборе тех хромосом, которые в дальнейшем будут участвовать в создании потомков для следующей популяции.

Рассмотрим следующие типы оператора селекции (в работе необходимо реализовать одну стратегию селекции согласно выбранному варианту).

1) *«Случайный отбор».* Случайным образом выбираем первого родителя. Из числа оставшихся хромосом выбрать случайным образом второго родителя.

Алгоритм «Случайный отбор».

Вход. Хромосома [1 … 2*N*] - популяция хромосом, 2*N* - размер популяции.

Выход. Родитель1[1 … *N*], Родитель2[1 … *N*] - массивы родительских хромосом. Алгоритм.

k:=1;

пока (*k* <= *N*)

{

{i:= random*[2N]; j:=* random*[2N];*

*}*пока *(i = j);*

Родитель1*[k]:=Хромосома[i];*

Родитель2*[k]:=Хромосома[j];*

}

2) *«Турнирный отбор».* Для каждого турнира выбирается случайным образом *t* хромосом. Из этих *t* хромосом выбирается наилучшая хромосома (наилучшей оценкой), которая берется в качестве одного из родителей.

Алгоритм «Турнирный отбор».

Вход. Хромосома [1 … 2*N*] - популяция хромосом, 2*N* - размер популяции.

Выход. Родитель1[1 … *N*], Родитель2[1 ... *N*] - массивы родительских хромосом.

Алгоритм.

k: =1;

пока *(k <= N)*

{

{M:= random*[2N]; //*размер турнира *t:= 1;* пока *(t <= M) {i:=* random*[N];*

Участник[t]:= Хромосома*[i]; //*участник в турнире

}

i:= номер лучшей хромосомы в турнире*; M:=* random*[2N];* повтор проведения турнира*;*

j:= номер лучшей хромосомы в турнире*;*

}пока (i = j);

Родитель1*[k]:=*Хромосома*[i];*

Родитель2*[k]:=*Хромосома*[j];*

}

3) *«Рулеточный отбор».* Вероятность *i*-ой хромосомы принять участие в скрещивании p_i пропорциональна значению ее приспособленности f_i и равна $p_i = \frac{1/f}{\sum_k (1/f_k)}$, т.е. весь круг рулетки равен 1, а площадь сектора *i*-ой особи пропорциональна значению p_i.

Алгоритм «Рулеточных отбор».

Вход. Хромосома [1 … 2*N*] - популяция хромосом, 2*N* - размер популяции. Ф [1 … 2*N*] – приспособленность хромосом.

Выход. Родитель1[1 ... *N*], Родитель2[1 … *N*] - массивы родительских хромосом. Алгоритм.

```
сумма:=0;
i:=1;
пока (i <= 2N)
{сумма:=1/Ф[i]+сумма;
}
i:=1; сектор[1]:= 0; пока (i <= 2N) {p:=1/(Ф[i]*сумма);
сектор[i+1]:= сектор[i]+p;
} k:=1; пока (k <= N)
{
{x1:= random;
 x2:= random;
 t:= 1;
 пока (t <= N+1)
 {если сектор[t]<= x1 < сектор[t+1] то i:=t;
  если сектор[t]<= x2 < сектор[t+1] то j:=t;
 }
 }пока (i = j);
 Родитель1[k]:=Хромосома[i];
 Родитель2[k]:=Хромосома[j];
}
```

4) *«Стратегия элитаризма».* Суть метода сводится к скрещиванию наилучшей хромосомы последовательно со всеми остальными.

Алгоритм «Стратегия элитаризма».

Вход. Хромосома [1 … 2*N*] - популяция хромосом, 2*N* - размер популяции.

Выход. Родитель1[1 … *N*], Родитель2[1 ... *N*] - популяции родительских хромосом.

Алгоритм

```
k: =2;
пока (k <= N)
{Родитель1[k]:=Хромосома[1];
 Родитель2[k]:=Хромосома[k];
```

}

4. Оператор скрещивания

На этапе скрещивания (кроссовера) родительские пары участвуют в создании потомков. Гены потомков получают свои значения путем обмена генетического материала между хромосомами-родителями. В результате скрещивания каждая пара хромосом-родителей дает пару потомков. С учетом специфики задачи, после скрещивания двух хромосом-родителей, оба их потомка должны быть проверены на условия включения в популяцию. Если хотя бы один из них не соответствует этим условиям, то хромосомы-родители скрещиваются заново. Процесс продолжается до тех пор, пока оба потомка не окажутся пригодными для выживания.

Выделим следующие типы оператора скрещивания:

1) «Одноточечное скрещивание». Случайным образом определяется точка внутри хромосомы, в которой обе хромосомы делятся на две части и обмениваются ими (рис. 12).

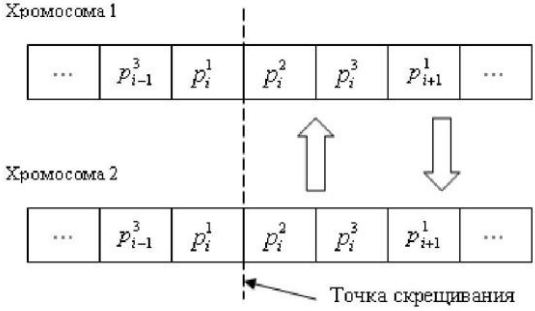

Рис. 12 – Одноточечное скрещивание

Алгоритм «Одноточечное скрещивание».

Вход. Родитель1[1 ... *N*], Родитель2[1 … *N*] - массивы родительских хромосом, 2*N* – размер популяции, *M* – размер хромосомы.

Выход. Потомок*[1..2×N]* - дочерняя популяция хромосом.

Алгоритм. *k:=1;*

пока *(k <= N)*

{

{t:= random*[M];* //точка скрещивания *i:=* 1; пока *(i <= M) {*если *(i <= t)* то

*{Потомок[2*k-1]:= Родитель1[k];*

*Потомок[k*2]:= Родитель2[k];*

}

иначе
*{Потомок[2*k-1]:= Родитель2[k];*
*Потомок[k*2]:= Родитель1[k];*
}
}
} пока *Потомок[2*k-1]* и *Потомок[k*2]* не удовлетворяют условиям включения в популяцию
}

2) «Двухточечное скрещивание». Случайным образом определяется две точки разрыва и хромосомы обмениваются генами между этими точками (рис. 13).

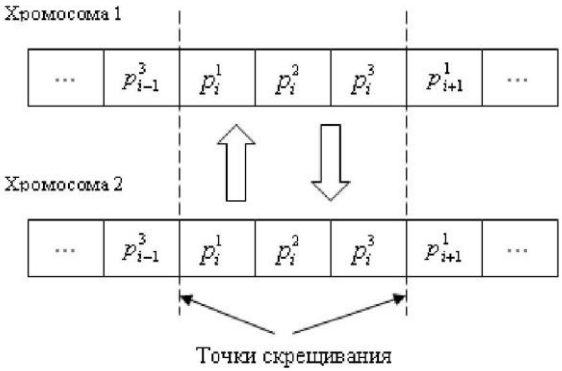

Рис. 13 – Двухточечное скрещивание

Алгоритм «Двухточечное скрещивание».

Вход. Родитель1[1..*N*], Родитель2[1..*N*] - массивы родительских хромосом, *2N* – размер популяции, *M* – размер хромосомы.

Выход. Потомок*[1..2×N]* - дочерняя популяция хромосом.

Алгоритм. *k:=1;*
пока *(k < N)*
{
{
{t1:= random*[N]; //точка скрещивания t2:=* random*[N]; //точка скрещивания*
} пока *(t1 >= t2); i:= 1; пока (i <= M)*
*{*если *(i <= t1) и (i >= t2)* то
*{Потомок[2*k-1]:= Родитель1[k];*
*Потомок[k*2]:= Родитель2[k];*

}
иначе
*{Потомок[2*k-1]:= Родитель2[k];*
*Потомок[k*2]:= Родитель1[k];*
}
}
*}*пока *Потомок[2*k-1] и Потомок[k*2]* не удовлетворяют условиям включения в популяцию
}

3) «Многоточечное скрещивание». Случайным образом выбирается *n* точек разрыва из отрезка. Таким образом, получается разбиение хромосом на *n+1* часть. Участки с четными номерами меняются. Нечетные участки остаются без изменений.

Алгоритм «Многоточечное скрещивание».

Вход. Родитель1[1..*N*], Родитель2[1..*N*] - популяции родительских хромосом, *2N* – размер популяции, *M* – размер хромосомы.

Выход. Потомок*[1..2×N]* - дочерняя популяция хромосом.

Алгоритм.

k:=1;
пока *(k <= N)*
{
{L:= random(M); //количество точек скрещивания
t:= 1; пока *(t <= L) {p:=* random*(M);*
Точка*[t]:=p;*
}
сортировка по возрастанию(*Точка[1..L]); i:= 1;*
t:=1; пока *(i <= M)*
{если *(i < Точка[t])* то *t:=t+1;*
если n — четное, то
*{Потомок[2*k-1]:=Родитель1[k];*
*Потомок[2*k]:=Родитель2[k];*
}
иначе
*{Потомок[2*k-1]:=Родитель2[k];*
*Потомок[2*k]:=Родитель1[k];*
}
}
*}пока Потомок[2*k-1] и Потомок[k*2]* не удовлетворяют условиям включения в популяцию
}

4) «Унифицированное скрещивание». Особенность заключается в том, что значение каждого гена в хромосоме потомка определяется случайным образом из соответствующих генов родителей. Для этого вводится некоторая величина $0<p_0<1$, и если случайное число больше p_0, то на n-ю позицию первого потомка попадает n-й ген первого родителя, а на n-ю позицию второго - *n-й* ген второго родителя. В противном случае к первому потомку попадает ген второго родителя, а ко второму - первого. Такая операция проводится для всех генов хромосомы.

Алгоритм «Унифицированное скрещивание».

Вход. Родитель1*[1..N]*, Родитель2*[1..N]* - популяции родительских хромосом, *2N* – размер популяции, *M* – размер хромосомы.

Выход. Потомок*[1..2×N]* — дочерняя популяция хромосом.

Алгоритм.

k:=1;

пока *(k <= N)*

{

{i:= 1; пока *(i <= M)* {*P:*= random; *x:*= random;

если *(x < p)* то

*{Потомок[2*k-1]:= Родитель1[k];*

*Потомок[k*2]:= Родитель2[k];*

}

иначе

*{Потомок[2*k-1]:= Родитель2[k];*

*Потомок[k*2]:= Родитель1[k];*

}

}

}пока *Потомок[2*k-1] и Потомок[k*2]* не удовлетворяют условиям включения в популяцию

}

5. Оператор мутации

Мутация – это преобразование хромосомы, случайно изменяющее один или несколько ее генов. Оператор мутации необходим для внесения случайности в процесс работы генетического алгоритма, что снижает вероятность застревания в локальных минимумах. С учетом специфики задачи, хромосомы подвергаются мутации с соблюдением условий включения в популяцию.

Рассмотрим следующие типы оператора мутации:

1) «Одноточечная мутация». Случайным образом выбирается один ген хромосомы, который меняет свое значение на другое.

Алгоритм «Одноточечная мутация».

Вход. Потомок*[1..2×N]* - дочерняя популяция хромосом, *2N* – размер популяции, M – размер хромосомы, P – вероятность мутации.

Выход. Потомок*[1..2×N]* - смутированная дочерняя популяция хромосом. Алгоритм.

k:=1;

пока *(k <= 2*N)*

{x:= random; если *(x < P) {t:=* random(M); *{x:=*random; ген *t* Потомка*[k]:= x;*

}пока *Потомок[k]* не удовлетворяет условиям включения в популяцию

}

}

2) «Многоточечная мутация». Случайным образом выбирается n генов хромосомы, которые меняют свое значение.

Алгоритм «Многоточечная мутация».

Вход. Потомок*[1..2×N]* - дочерняя популяция хромосом, *N* – размер популяции, *M* – размер хромосомы, *P* – вероятность мутации.

Выход. Потомок*[1..2×N]* - смутированная дочерняя популяция хромосом. Алгоритм.

k:=1;

пока *(k <= 2*N)*

{

p:= random*;* если *(x < P)*

{

d:= random*(M); //количество точек мутации*

i:= 1; пока *(i <= d) {t:=* random*(M);*

Точка[i]:= t;

} i:= 1; t:= 1; пока *(i <= M) {*если *(i = Точка[t])*

{

*{x:=*random*; ген i Потомка[k]:= x;*

}пока *Потомок[k] не удовлетворяет условиям включения в популяцию*

t:= t+1;

}

}

6. Создание новой популяции

В процессе выполнения одной итерации генетического алгоритма особи родительской популяции (старой популяции) «воспроизводят» потомство. Это приводит к появлению новых хромосом, которые сочетают в себе некоторые характеристики, наследуемые ими от родителей. Чтобы перейти к следующей итерации генетического алгоритма, необхо-

димо сформировать промежуточную популяцию, хромосомы которой будут участвовать в «воспроизведении» следующей популяции потомков.

Варианты заданий

1. Реализовать операторы селекции, скрещивания и мутации согласно выбранному варианту.

2. Проверить работоспособность генетического алгоритма с различными параметрами: количество термов лингвистических переменных (5,7,9); количество шагов генетического алгоритма (10, 100); размер популяции (10,20,30);

Форма отчетности

Отчет должен содержать следующие обязательные пункты:

1) цель работы;

2) аналитическое описание моделируемой функции;

3) описание созданных классов, методов, глобальных переменных.

4) пример сгенерированной начальной популяции с соответствующими значениями фитнес-функций.

2.5 Лабораторная работа «Идентификация параметров нечеткой системы с помощью алгоритма муравьиной колонии»

Цель работы

Знакомство с основными принципами работы генетического алгоритма и особенностями его использования для настройки параметров функций принадлежности. Создание начальной популяции и применение оператора селекции, скрещивания и мутации в задаче идентификации нечетких систем.

Порядок выполнения работы

Основанием для выполнения настоящей работы являются результаты, полученные при выполнении лабораторной работы «Нечеткая система типа синглтон. Создание базы правил» и «Нечеткая система типа синглтон. Создание машины нечеткого вывода».

1. Классический алгоритм муравьиной колонии

Впервые алгоритм муравьиной колонии был предложен М. Дориго и его коллегами для решения задач дискретной оптимизации, например, для решения задач коммивояжера.

АМК является метаэвристической процедурой, в основе которой лежат наблюдения за поведением муравьев, в частности способность муравьев при передвижении выделять фермент и использовать его в качестве маркера при поиске пищи. Таким образом, чем большее количество муравьев выбирает некоторый путь или чем чаще один и тот же муравей будет ходить по этому пути, тем выше концентрация фермента на данном пути, а значит можно считать, что количество фермента пропорционально качеству искомого решения.

2. Дискретный алгоритм муравьиной колонии для параметрической идентификации нечетких систем

АМК — процедура дискретной оптимизации, в то время как параметры ФП меняются непрерывно. Переход от непрерывной оптимизации к дискретной осуществляется посредством построения полного ориентированного графа поиска решения, количество вершин в котором определяется точностью нахождения значений параметров. Из каждой вершины выходят дуги с равномерно распределенными значениями нормированных параметров ФП входных переменных (рис. 14). Во все вершины графа равномерно распределяются муравьи. Цель муравья в задаче идентификации параметров нечеткой системы — посетить столько вершин, сколькими параметрами задается ФП. Например, треугольная — тремя

вершинами. Пометки дуг, по которым прошел муравей, будут являться найденным им решением. В отличие от классического алгоритма муравьи могут повторно проходить пройденные вершины, т.к. параметры ФП могут иметь одинаковые значения.

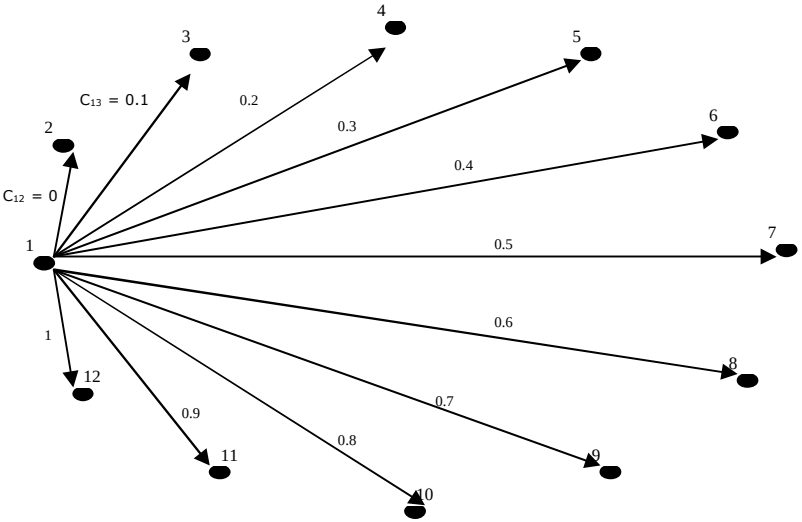

Рис. 14 – Фрагмент начального графа поиска решения для муравьев, начинающих движение из узла 1

Каждая лингвистическая переменная описана несколькими ФП. Муравьи в алгоритме делятся на колонии. Каждая колония муравьев отвечает за нахождение параметров своей ФП.

Количество фермента, наносимого на дуги пропорционально качеству решения, чем меньше ошибка вывода нечеткой системы, выполненного на выбранных параметрах, тем больше фермента наносится на дуги:

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Q}{E^{k}(t)} & \text{, если муравей проходит дугу (i.j)}\\ 0, \text{иначе}\end{cases}$$

где Q — константа, определяющая количество фермента у отдельного муравья,

$E^k(t)$ — значение ошибки на t-ой итерации для параметров, выбранных k-ым муравьем.

Вероятность выбора муравьем дуги (i, j) на t-ой итерации, вычисляется по формуле:

$$p_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha}}{\sum\limits_{\substack{l=1 \\ l \neq i}}^{N} \tau_{il}(t)^{\alpha}} \cdot$$

где α — параметр, учитывающий важность (приоритет, значимость) фермента на пути,

$p_{ij}(t)$ — интенсивность фермента на дуге между узлами *i* и *j* на *t*-ой итерации,

N — множество узлов смежных вершине i, причем узел j может быть выбран многократно; в отличии от классического использования алгоритма в нашем случае не формируется запрещенный список узлов.

Увеличение количества фермента определяется следующим образом:

$$\tau_{ij}(t+1) = \sum_{k=1}^{M} \left(\Delta \tau_{ij}^{k}(t) + \tau_{ij}(t) \right) \cdot p$$

а испарение фермента — по формуле:

$$\tau_{ij}(t+1) = \tau_{ij}(t) \cdot (1-p)$$

здесь $\rho \in [0;1]$ — коэффициент снижения интенсивности фермента, *M* — количество муравьев, прошедших по дуге *(i, j)*.

На рисунке 15 представлен пример поиска решения муравьем, выходящего из точки 1. Допустим, муравей сделал свой выбор три раза – ровно столько сколько параметров необходимо оптимизировать для треугольной ФП. Он нашел набор параметров {0; 0.7; 0.3}. Это Нормированные неупорядоченные параметры ФП. Далее необходимо сделать следующее:

- упорядочить найденные параметры – {0; 0.3; 0.7};
- масштабировать найденные параметры. Если лингвистическая переменная изменяется от 0 до 5, то масштабированные упорядоченные параметры ФП будут равны {0; 1.5; 3.5};
- передать полученные параметры в НС и проверить на валидность;
- проверить удовлетворяют ли условиям построения НС найденные параметры. Если ДА, то найти ошибку.

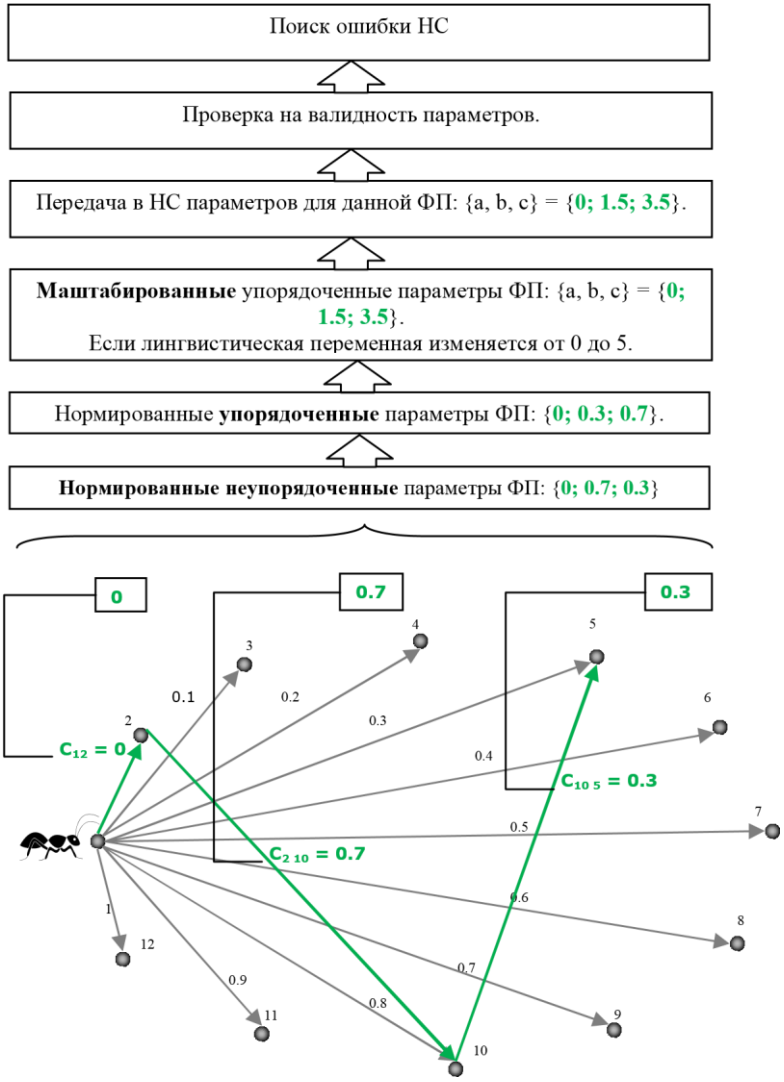

Рис. 15 – Пример поиска муравьем решения

Ниже приведен алгоритм оптимизации параметров функций принадлежности.

Шаг 1. Задать начальные параметры алгоритма и нечеткой системы.

Шаг 2. Задать популяции муравьев в колониях.

Шаг 3. Для всех муравьев текущей колонии определить три дуги, используя два раза одну и вторую формулу.

Шаг 4. Передать в нечеткую систему значения параметров функций принадлежности, определенных муравьями текущей колонии, и вычислить ошибки. Если параметры, переданные муравьем в нечеткую систему, лучше текущих, то сохранить новые значения параметров.

Шаг 5. Если имеется следующая колония, то сделать ее текущей и перейти на шаг 3.

Шаг 6. Вычислить количество фермента на каждой дуге по формуле 4.

Шаг 7. Вычислить количество испаренного фермента по формуле.

Шаг 8. Если условие окончание работы алгоритма выполнено, то закончить, иначе перейти к шагу 2.

Условием окончания работы алгоритма является достижение заданного числа итераций.

3. Непрерывный алгоритм муравьиной колонии для параметрической идентификации нечетких систем

В АМК для дискретной оптимизации при выборе очередной дуги, муравей руководствуется дискретным распределением вероятности (рис. 16). Где c_{ij} — это вес дуги *(i;j).*

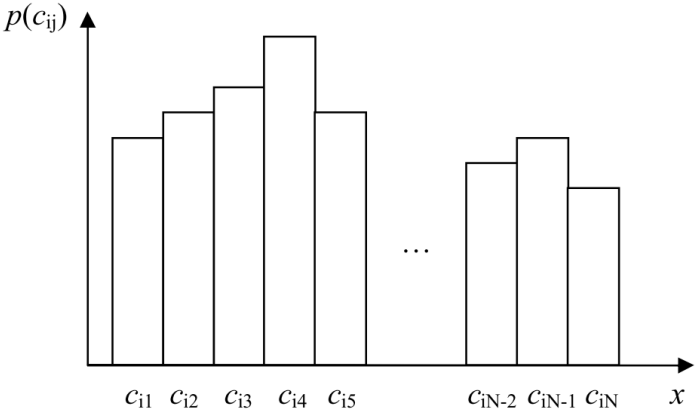

Рис. 16 – Дискретное распределение вероятностей выбора муравьем *j-й* дуги при текущей *i-й*

В случае НАМК выбор, который делает муравей не ограничен конечным множеством для этого дискретное распределение заменяется на непрерывное, то есть на функцию плотности вероятности (рис. 17).

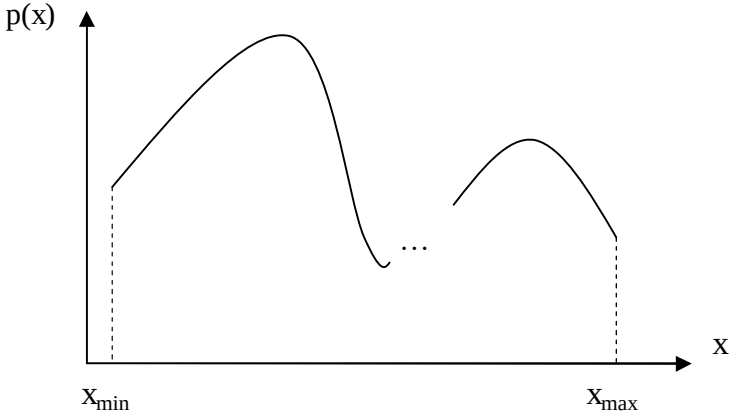

Рис. 17 – Функция плотности вероятности

Одна из самых популярных функций, которая используется как ФПВ — функция Гаусса. Ее преимущество в простом способе генерации случайных чисел. Но очевидный недостаток в том, что функция Гаусса имеет только один максимум. Поэтому в НАМК для описания используется ФПВ с Гауссовым ядром. Под Гауссовым ядром $G^i(x)$ понимается функция, основанная на взвешенной сумме нескольких одномерных Гауссовых функций $g_l^i(x)$:

$$G^i(x) = \sum_{l=1}^{k} \omega_l g_l^i(x) = \sum_{l=1}^{k} \omega_l \frac{1}{\sigma_l^i \sqrt{2\pi}} e^{-\frac{(x-\mu_l^i)^2}{2\sigma_l^{i2}}}$$

Каждому параметру соответствует свое Гауссово ядро, поэтому $i = 1, ..., n$. Каждая функция $G^i(x)$ описывается тремя векторами параметров: ω — вектор весов, связанных с индивидуальными Гауссовыми функциями, μ^i — вектор математических ожиданий, и σ^i — вектор среднеквадратичных отклонений. Количество элементов всех этих векторов равно числу функций Гаусса, составляющих Гауссово ядро, т.е. $|\omega| = |\mu^i| = |\sigma^i| = k$. Пример Гауссова ядра представлен на рисунке 18.

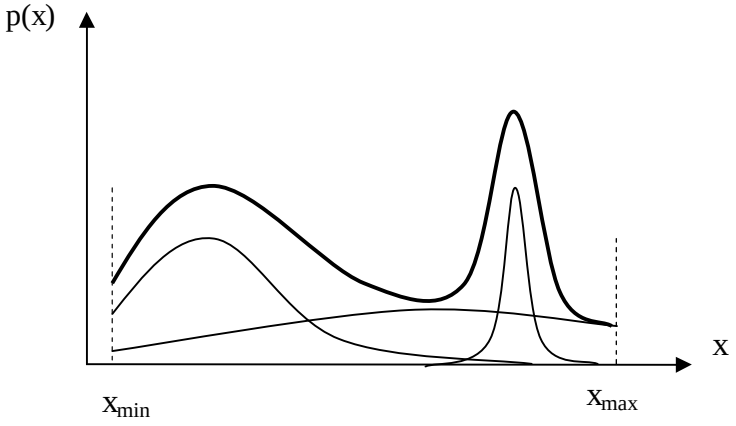

Рис. 18 – Пример Гауссова ядра состоящего из трех Гауссовых функций

В НАМК вводится понятие архива решений T (рис. 19). Архив решений представлен таблицей, в которой k строк. Каждая строка представляет собой найденное муравьем решение $s_l=\{s_l^1, s_l^2,\ldots, s_l^M\}$, ошибку НС $F(s_l)$ и вес решения ω_l. M – количество оптимизируемых параметров для одной ФП (для треугольной M = 3).

11	21		s_{M1}	$F(s_1)$	ω_1
12	22		s_{M2}	$F(s_2)$	ω_2
			.	..	
1k	2 k		$s_{M k}$	$F(s_k)$	ω_k

Рис. 19 – Архив решений

Решения упорядочены в архиве согласно их качеству, то есть $f(s_1) \le f(s_2) \le \ldots \le f(s_l) \le \ldots \le f(s_k)$. Вес ω_l решения s_l вычисляется согласно следующей формуле:

$$\omega = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}}$$

где q — задаваемый параметр алгоритма. Когда q мало, наилучшие решения строго предпочитаемы, а когда оно велико, вероятность (выбора решения) становится более однородной. Значение вектора ω вычисляется один раз, так как в процессе поиска решений значения q и k не меняются.

При поиске муравьем первого параметра s_1, вычисляется вероятность выбора l-й функции Гаусса, составляющей Гауссово ядро. Остальные параметры ищутся также для функций с номером l.

$$p = \frac{\omega_1}{\sum_{r=1}^{k} \omega_r}$$

Для того чтобы построить функцию $g_l{}^i(x)$, необходимо найти ее математическое ожидание $\mu_l{}^i$ и среднеквадратичное отклонение $\sigma_l{}^i$.

Параметр $\xi > 0$, является одинаковым для всех размерностей и имеет эффект подобный норме испарения фермента в ДАМК. Чем выше значение ξ, тем ниже скорость конвергенции алгоритма.

Далее генерируются случайное число с законом распределения $g_l^i(x)$, например, с помощью метода Бокса-Мюллера, это число и будет найденный муравьем параметр x_i. После того как муравей нашел n таких параметров, они передаются в НС, где вычисляется ошибка. Новое решение добавляется в архив решений T, решения сортируются и худшее из них удаляется. Этот процесс аналогичен процессу испарения фермента в ДАМК. Он гарантирует, что только лучшие решения хранятся в архиве, эффективно направляя муравьев в процессе поиска.

Следует отметить, что как и в ДАМК, муравьи в НАМК делятся на колонии, каждая из которых отвечает за нахождение параметров своей ФП. У каждой колонии свой независимый архив решений.

Ниже приведен алгоритм работы НАМК для оптимизации параметров ФП:

Шаг 1. Задать начальные параметры.

Шаг 2. Сгенерировать популяцию муравьев в колониях.

Шаг 3. Сгенерировать k случайных решений, для всех архивов решений с последующим оцениванием и ранжированием.

Шаг 4. Найти значения вектора ω.

Шаг 5. Для текущего муравья текущей колонии вычислить номер *l*, используемой функции Гаусса. Определить параметры μ_l^i и σ_l^i для $g_l^i(x)$ $i = 1, ..., n$. Сгенерировать n случайных величин $\{s_1, s_2, \ldots, s_M\}$ с законами распределения $g_l^i(x)$.

Шаг 6. Найти ошибку НС при параметрах $\{s_1, s_2, \ldots, s_M\}$, если ошибка меньше текущей, то сохранить новые параметры в НС.

Шаг 7. Добавить в архив решений новое решение, проранжировать архив, удалить из архива худшее решение.

Шаг 8. Если в текущей колонии имеется следующий муравей, то сделать его текущим и перейти к шагу 5.

Шаг 9. Если имеется следующая колония, то сделать текущим первого муравья в этой колонии и перейти на шаг 5.

Шаг 10. Если условие окончание работы алгоритма выполнено, то закончить, иначе сделать текущим первого муравья первой колонии и перейти к шагу 5.

Условием окончания работы алгоритма является достижение заданного числа итераций.

Варианты заданий

№	Алгоритм муравьиной колонии
нечетные	Дискретный алгоритм муравьиной колонии
четные	Непрерывный алгоритм муравьиной колонии

Задание

1) В зависимости от варианта реализовать требуемый алгоритм муравьиной колонии.

2) Взять произвольные параметры алгоритма и выполнить одну итерацию.

3) Представить проделанную работу в отчете.

Примечание.

• В качестве начальных параметров сгенерировать равномерные значения параметров ФП для всех лингвистических переменных. Под равномерным разбиением понимается, равномерное покрытие области изменений, с пересечением соседних термов на уровне 0.5 (как в лабора-

торной работе «Нечеткая система типа синглтон. Создание машины нечеткого вывода»).

- Для НАМК не изменять:
  - 1-ый и 2-ой параметры (a и b) для первой ФП, описывающей каждую лингвистическую переменную;
  - 2-ой и 3-ий параметры (b и c) для последней ФП, описывающей каждую лингвистическую переменную.
- В программе должна быть возможность изменять любые параметры алгоритма (для дискретного алгоритма муравьиной колонии: количество итераций, количество точек орграфа (12, 102 и т.д.), коэффициент количество муравьев к точкам, α, ρ, Q; для непрерывного алгоритма муравьиной колонии: количество итераций, количество муравьев, размер архива решения – k, параметры ξ и q)

Форма отчетности

Отчет должен содержать следующие обязательные пункты:

1) цель работы;
2) аналитическое описание моделируемой функции;
3) описание созданных классов, методов, глобальных переменных;
4) пример результата выполнения программы: начальные и результирующие параметры функций принадлежности с соответствующими значениями ошибки нечеткой системы;
5) выводы по лабораторной работе.

2.6 Лабораторная работа «Исследование влияния параметров алгоритмов и нечеткой системы на сходимость алгоритмов идентификации»

Цель работы

Исследование и анализ работы генетического алгоритма, алгоритма муравьиной колонии и нечеткой системы при изменении их параметров.

Порядок выполнения работы

Основанием для выполнения настоящей работы являются результаты, полученные при выполнении лабораторной работы «Идентификация нечеткой системы с помощью генетического алгоритма. Генерация начальной популяции. Оператор селекции. Операторы скрещивания и мутации» и «Идентификация параметров нечеткой системы с помощью алгоритма муравьиной колонии».

1. Влияние количества термов лингвистических переменных на сходимость алгоритма

Исследуйте работу генетического алгоритма и алгоритма муравьиной колонии при изменении количества термов лингвистических переменных. Для каждого разбиения проведите серию из пяти опытов: при каждом запуске алгоритмов необходимо сохранить промежуточные значения ошибки (для генетического алгоритма «лучшей» хромосомы) (рис. 20). В результате эксперимента – 4 графика динамики поиска решения: для пяти и семи термов, для двух алгоритмов идентификации.

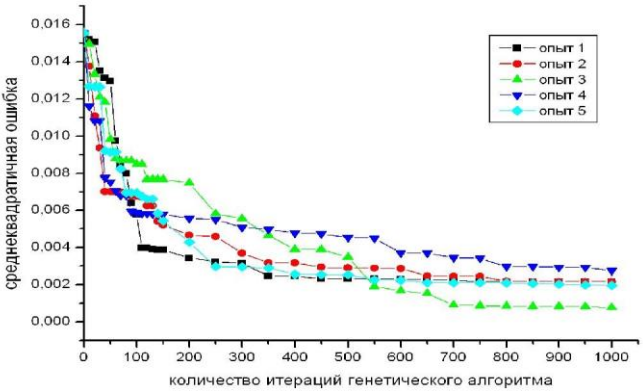

Рис. 20 – Динамика поиска решений генетическим алгоритмом при разделении входов на 5 термов

Проанализируйте полученные графики. Выясните, через сколько шагов алгоритмов не наблюдается существенного уменьшения ошибки и почему это происходит.

2. Влияние параметров алгоритмов на сходимость алгоритма

Для генетического алгоритма постройте график распределения ошибки нечеткой системы в зависимости от числа итераций генетического алгоритма и размера популяции (рис. 21). Для этого проведите серии из пяти опытов для каждого значения параметра. Проанализируйте полученный график.

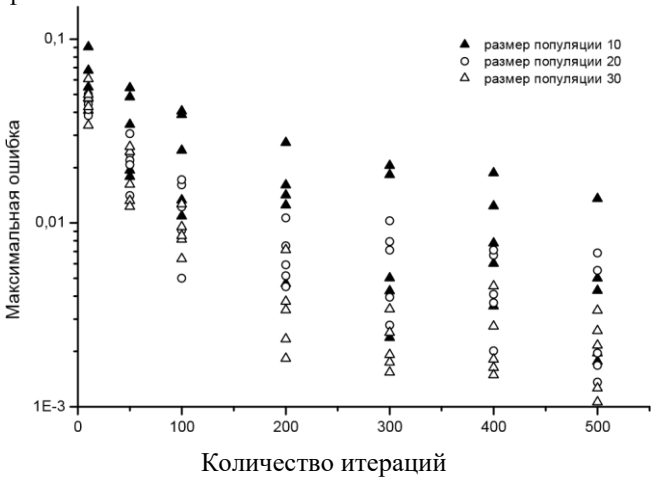

Рис. 21 – Распределение максимальной ошибки решения при различных размерах популяции

Для дискретного алгоритма муравьиной колонии постройте график распределения ошибки нечеткой системы в зависимости от зависимости от числа итераций и параметров α или ρ.

Для непрерывного алгоритма муравьиной колонии постройте график распределения ошибки нечеткой системы в зависимости от зависимости от числа итераций и размер архива решения – k. Для этого прове-

дите серии из пяти опытов для каждого значения параметра. Проанализируйте полученный график.

3. Сравнение генетического алгоритма и алгоритма муравьиной колонии

Зафиксируйте лучшие параметры обоих алгоритмов, полученные в результате предыдущих экспериментов. Для фиксированного количества итераций (1000) и термов выполнить 5 экспериментов для каждого алгоритма. Построить на одном графике среднюю динамику изменения ошибки для каждого алгоритма. Выявить и отобразить среднее время работы алгоритмов на заданном количестве итераций. Сделать выводы о полученных результатах идентификации.

Варианты заданий

1) Исследовать работу генетического алгоритма и алгоритма муравьиной колонии при изменении количества термов лингвистических переменных;

2) Исследовать работу генетического алгоритма и алгоритма муравьиной колонии при изменении специфических параметров алгоритмов;

3) Сравнить работу генетического алгоритма и алгоритма муравьиной колонии.

Результаты исследований представить в виде графиков. Провести анализ полученных графиков.

Примечание.

- В качестве начальных параметров сгенерировать равномерные значения параметров ФП для всех лингвистических переменных
- В качестве оператора мутации необходимо выбрать любой, если в варианте их два.
- В качестве оператора скрещивания необходимо выбрать любой, если в варианте их два.

Форма отчетности

Отчет должен содержать следующие обязательные пункты: 1) цель работы;

2) аналитическое описание моделируемой функции;

3) значения фиксируемых параметров алгоритмов идентификации;

4) пример результата выполнения программы: результирующие параметры функций принадлежности с соответствующими значениями ошибки нечеткой системы;

5) графики, полученные по результатам исследований, и выводы по влиянию каждого параметра на сходимость алгоритмов;

6) выводы по лабораторной работе.

2.7 Лабораторная работа «Программирование баз знаний»

Цель работы

Цель работы – создание базы данных в среде ПРОЛОГ по выбранной предметной области.

Порядок выполнения работы

Выполнение работы состоит из следующих этапов:

- выбор предметной области. Четкое (письменное) формулиро-
- вание цели создания системы;
- формальное описание предметной области. В качестве форма-
- лизма для описания должны быть выбраны таблицы;
- представление предметной области на языке ПРОЛОГ;
- формулировка запросов и оформление интерфейса.

Примерный круг предметных областей:

− расписание занятий;

− расписание движения транспорта (авто-, авиа-, железнодорожного);

− расписание приема врачами в поликлинике.

Для примера рассмотрим первую предметную область «Расписание занятий». В системе должны быть представлены отношения между четырьмя элементами: названием курса, временем, именем лектора и местом проведения занятий. Система должна выдавать сведения, например, увязывающие читаемый курс и фамилию лектора, день недели и фамилию лектора, занятость аудиторий по времени.

На языке ПРОЛОГ данный факт может быть представлен следующим образом:

```
represent("ОИИиЭС", time("среда",9,11),
lector("Ходашинский", "Илья", "Александрович"),
place("ФЭТ",426)).
```

Отношение, увязывающее читаемый курс и фамилию лектора, имеет следующий вид:

```
rel1(F,K):-represent(K,_,lector(F,_,_),_).
```

Отношение, увязывающее день недели и фамилию лектора, имеет следующий вид:

```
rel2(F,D):-represent(_, time(D,_,_),lector(F,_,_),_).
```

Отношение, увязывающее занятость аудитории во времени, имеет следующий вид:

rel3(A, K,D,H,K):- represent(_, время(D,H,K),_, place(K, A)).

Форма проведения

Выполнение индивидуального задания: в рамках выполнения лабораторной работы студент самостоятельно выбирает любую моделируемую им предметную область.

ТРЕБОВАНИЯ К БАЗЕ ЗНАНИЙ:

- произведение количества полей на количество записей должно
- быть не менее 300 для табличного представления;
- число запросов к базе данных должно быть не менее пяти.

2.8 Лабораторная работа «Сортировка. Представление графов и поиск пути на графе»

Цель работы

Цель работы – создание представления графов и поиск пути на графе в среде ПРОЛОГ.

Порядок выполнения работы

Выполнение работы состоит из следующих этапов:

- выбор предметной области. Четкое (письменное) формулиро-
- вание цели создания системы;
- формальное описание предметной области. В качестве формлима для описания должны быть выбраны продукционные правила;
- представление предметной области на языке ПРОЛОГ;
- формулировка запросов и оформление интерфейса.

Примерный круг предметных областей:

1) лекарственные растения (грибы, ягоды);
2) покупка компьютера (автомобиля, квартиры);
3) кулинария;
4) породы домашних (диких) животных;
5) починка телевизора (автомобиля, компьютера);
6) лечение собаки (кошки, человека).

ТРЕБОВАНИЕ К БАЗЕ ЗНАНИЙ: количество правил должно быть не менее 30.

Рассмотрим семь животных распространенных пород. Ниже приведены продукционные правила, задающие описание животных. Здесь биологический класс – это птицы или млекопитающие.

Правило 1.
ЕСЛИ животное имеет волосы,
ТО это животное млекопитающее.
Правило 2.
ЕСЛИ животное дает молоко,
ТО это животное млекопитающее.
Правило 3.
ЕСЛИ животное имеет перья,
ТО это животное птица.
Правило 4.
ЕСЛИ животное умеет летать

И несет яйца,
ТО это животное птица.
Правило 5.
ЕСЛИ животное – млекопитающее
И ест мясо,
ТО это хищник.
Правило 6.
ЕСЛИ животное – млекопитающее
И имеет острые зубы,
И имеет когти,
И имеет посаженные впереди глаза,
ТО это хищник.
Правило 7.
ЕСЛИ животное – млекопитающее
И имеет копыта,
ТО это копытное.
Правило 8.
ЕСЛИ животное – млекопитающее
И жует жвачку,
ТО это копытное.
Правило 9.
ЕСЛИ животное – хищник
И имеет рыжевато-коричневую окраску,
И имеет темные пятна,
ТО это леопард.
Правило 10.
ЕСЛИ животное – хищник
И имеет рыжевато-коричневую окраску,
И имеет черные полосы,
ТО это тигр.
Правило 11.
ЕСЛИ животное – копытное
И имеет длинные ноги,
И имеет длинную шею,
И имеет рыжевато-коричневую окраску,
И имеет темные пятна,
ТО это жираф.
Правило 12.
ЕСЛИ животное – копытное
И имеет белый цвет,
И имеет черные полосы,

ТО это зебра.
Правило 13.
ЕСЛИ животное – птица
И не умеет летать
И имеет длинные ноги,
И имеет длинную шею,
И имеет бело-черную окраску,
ТО это страус.
Правило 14.
ЕСЛИ животное – птица
И не умеет летать,
И умеет плавать,
И имеет бело-черную окраску,
ТО это пингвин.
Правило 15.
ЕСЛИ животное – птица
И умеет очень хорошо летать, ТО это альбатрос.

Работа системы распознавания сводится к генерации гипотезы о принадлежности животного к тому или иному классу и к попытке подтвердить эту гипотезу. В нашем случае генерируется первая гипотеза: «распознаваемое животное – это млекопитающее». Для подтверждения данной гипотезы необходимо, чтобы пользователь утвердительно ответил хотя бы на один из вопросов: «имеет ли животное волосы» или «дает ли животное молоко». Если положительный ответ получен уже на первый вопрос, то система генерирует следующую гипотезу «это млекопитающее – хищник». Если же положительный ответ не получен на первый вопрос, то система задает второй вопрос. Если на него получен положительный ответ, генерируется гипотеза «млекопитающее – хищник», если получен отрицательный ответ, генерируется гипотеза: «распознаваемое животное – птица». Процесс порождения гипотез и их проверки длится до тех пор, пока есть подходящие для этого правила. Описание таких правил приведено ниже:

```
rule(1,"животное","млекопитающее",[1]).
rule(2,"животное","млекопитающее",[2]).
rule(3,"животное","птица",[3]).
rule(4,"животное","птица",[4, 5]).
rule(5,"млекопитающее","хищник",[6]).
rule(6,"млекопитающее","хищник",[7, 8, 9]).
rule(7,"млекопитающее","копытное",[10]).
rule(8,"млекопитающее","копытное ",[11]).
rule(9,"хищник","леопард",[12, 13]).
```

```
rule(10,"хищник","тигр",[12, 14]).
rule(11,"копытное","жираф",[15, 16, 12, 13]).
rule(12,"копытное","зебра",[17, 14]).
rule(13,"птица","страус",[18, 15, 16, 19]).
rule(14,"птица","пингвин",[18, 20, 19]).
rule(15,"птица","альбатрос",[21]).
```

Первый аргумент в предикате rule – это номер правила, второй – род, третий – вид, четвертый – список вопросов, подтверждающий отношение род-вид.

Для того чтобы применить ту или иную продукцию, необходимо собрать факты, задав пользователю вопросы. Однако, прежде чем задвать вопрос, необходимо быть уверенным в том, что этот вопрос уже не был задан ранее при подтверждении других промежуточных гипотез. Информация о заданном вопросе и полученном на него ответе хранится в отношении fact(X, Y) динамической базы данных, где X – номер вопроса, Y – ответ на этот вопрос ("да", "нет"). Если вопрос был уже задан и на него получен положительный ответ, то вывод успешно продолжается, если же получен отрицательный ответ, то система сообщает о неуспехе. Множество задаваемых вопросов приведено ниже.

```
ask(X):- fact(X, "да"),!.
ask(X):- fact(X, "нет"),!,fail.
ask(1):- write("оно имеет волосы?"), !, complete(1).
ask(2):- write("оно дает молоко?"), !, complete(2).
ask(3):- write("оно имеет перья?"), !, complete(3).
ask(4):- write("оно умеет летать?"), !, complete(4).
ask(5):- write("оно несет яйца?"), !, complete(5).
ask(6):- write("оно ест мясо?"), !, complete(6).
ask(7):- write("оно имеет острые зубы?"), !, complete(7).
ask(8):- write("оно имеет когти?"), !, complete(8).
ask(9):- write("оно имеет посаженные впереди глаза?"), !,complete(9).
ask(10):- write("оно имеет копыта?"), !, complete(10).
ask(11):- write("оно жует жвачку?"), !, complete(11).
ask(12):- write("оно имеет рыжевато-коричневую окраску?"), !,
com     plete(12).
ask(13):- write("оно имеет темные пятна?"), !, complete(13).
ask(14):- write("оно имеет черные полосы?"), !, complete(14).
ask(15):- write("оно имеет длинные ноги?"), !, complete(15).
ask(16):- write("оно имеет длинную шею?"), !, complete(16).
ask(17):- write("оно имеет белый цвет?"), !, complete(17).
ask(18):- write("оно не умеет летать?"), !, complete(18).
ask(19):- write("оно имеет бело-черную окраску?"), !, complete(19).
```

```
ask(20):- write("оно умеет плавать?"), !, complete(20).
ask(21):- write("оно умеет очень хорошо летать?"), !,
complete(21).
```

Процедура recognition(X) занимает центральное место в программной реализации продукционной системы. Процедура состоит из трех предложений. В первом предложении генерируется гипотеза (rule(N, X, Y, Z)) и ищется ее подтверждение (discover(Z)); если гипотеза подтверждается, то выдается соответствующее сообщение, если выдвутая гипотеза не подтверждается, то генерируется следующая. Второе предложение процедуры описывает ситуацию, когда пользователь на все вопросы ответил отрицательно и системе не удалось выдвинуть ни одной гипотезы. И наконец, третье предложение задает успешное окончание работы, когда была подтверждена хотя бы одна гипотеза.

```
recognition(X):- rule(N, X, Y, Z), discover(Z), !,
write("_____", X, " -  ", Y, " по правилу ", N), nl,
recognition(Y).
recognition("животное"):- write("это животное мне неизвестно"),!.
recognition(_).
discover([]).
discover([X|Y]):- ask(X), discover(Y).
complete(X):- nl, read(Y), assert(fact(X, Y)), Y="да".
```

Рассмотрим пример работы системы.

```
?- retractall(_), recognition("животное").
"оно имеет волосы?"
да
_____  животное – млекопитающее по правилу 1
"оно ест мясо?"
  нет
  оно" имеет острые зубы?"
  да
"оно имеет когти?"
да
"оно имеет посаженные впереди глаза?"
да
_____  млекопитающее – хищник по правилу 6
"оно имеет рыжевато-коричневую окраску?"
да
"оно имеет темные пятна?"
нет
"оно имеет черные полосы?"
```

да

___ хищник – тигр по правилу 10.

Пример показывает полное распознавание животного.

Важный вопрос построения продукционной системы – это разработка структуры продукционного правила. Некоторые рекомендации приведены ниже:

- конструируйте правила, опираясь на структуру, присущую предметной области;
- используйте минимально достаточное количество условий при определении продукционного правила;
- избегайте противоречащих продукционных правил.

Каждое продукционное правило может быть независимым от других. Модульность правил позволяет легко модифицировать продукционную систему. Модификация заключается в добавлении, изменении, удалении правил и не затрагивает существующих процедур. Число правил в системе ограничено размерами памяти компьютера. Продукционные правила можно поместить и в динамическую базу данных, тогда возможно хранение правил и во внешней памяти компьютера.

Форма проведения

Выполнение индивидуального задания: в рамках выполнения лабораторной работы студент самостоятельно выбирает любую моделируемую им предметную область

3 Методические указания для организации самостоятельной работы

3.1 Общие положения

Целями самостоятельной работы являются систематизация, расширение и закрепление теоретических знаний.

Самостоятельная работа студента по дисциплине «Интеллектуальные вычислительные системы» включает следующие виды активности:

1. Изучение тем теоретической части дисциплины, вынесенных для самостоятельной проработки.
2. Подготовка к лабораторным работам.
3. Выполнение индивидуального (творческого) задания (ИЗ) по одной из предложенных тем.
4. Подготовка реферата.
5. Подготовка к тестовым опросам.

3.2 Изучение тем теоретической части дисциплины, вынесенных для самостоятельной проработки

Изучение тем теоретической части дисциплины, вынесенных для самостоятельной проработки:

- нечеткие множества.

В рамках данной темы необходимо обратить внимание на понятие нечёткого множества, лингвистической переменной, терма. Детально познакомиться с понятием функции принадлежности и операциями над нечеткими множествами.

- модели представления знаний.

В рамках данной темы необходимо обратить внимание на модели представления данных: логическую, продукционную, фреймовую, семантические сети.

3.3 Подготовка к лабораторным работам

В рамках выполнения подготовки к лабораторным работам рекомендуется детально познакомиться с теоретическим материалом по темам лабораторных работ.

3.4 Выполнение индивидуального (творческого) задания (ИЗ)

В рамках выполнения индивидуального (творческого) задания (ИЗ) необходимо подготовить 7 - минутный доклад, раскрывающий одну из следующих тем:

- нечеткие модели систем искусственного интеллекта;

- фреймовые модели систем искусственного интеллекта;
- архитектуры систем искусственного интеллекта.

Вариант индивидуального задания определяется преподавателем в индивидуальном порядке, основываясь на уровень знаний и студента.

3.5 Подготовка реферата

Подготовка реферата по одной из тем:
- фреймовые системы;
- формирование правил из нечетких данных;
- семантические сети;
- экспертные системы;
- онтологический подход к разработке интеллектуальных систем;
- системы ситуационного управления.

Рекомендуемый объем реферата не более 20 страниц.

Критерии оценивания работы:
- титульный лист оформить в соответствии с образовательным стандартом ТУСУРа;
- корректность (правильность) представленной информации;
- глубина проработанного материала, в т.ч. количество пунктов для сравнения (более 7);
- самостоятельность исследования (отсутствие плагиата);
- список литератыры;
- компактная и схематичная форма представления данных.

3.6 Подготовка к тестовым опросам

Тестовые опросы проводиться по темам, материал которых читался лектором на предыдущем занятии. В рамках подготовки к тестовым опросам рекомендуется пользоваться дополнительно литературой по курсу, рекомендуемой лектором.

4 Основная и дополнительная литература

Основная литература:

1. Ходашинский И. А. Методы искусственного интеллекта, базы знаний, экспертные системы : Учебное пособие / И. А. Ходашинский ; Министерство образования Российской Федерации, Томский государственный университет систем управления и радиоэлектроники, Кафедра автоматизации обработки информации. - Томск : ТУСУР, 2002. - 140 с. (наличие в библиотеке ТУСУР - 35 экз.)

Дополнительная литература:

1. Ясницкий Л. Н.. Введение в искусственный интеллект: Учебное пособие для вузов / Л. Н. Ясницкий. - М. : Academia, 2005. - 174 с. (наличие в библиотеке ТУСУР - 20 экз.)

2. Гаврилова, Т. А. Базы знаний интеллектуальных систем: Учебник для технических вузов / Т. А. Гаврилова, В. Ф. Хорошевский. - СПб.: Питер, 2001. - 384 с. (наличие в библиотеке ТУСУР - 22 экз.)