МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Филиал федерального государственного бюджетного образовательного учреждения высшего образования

«Национальный исследовательский университет «МЭИ» в г. Смоленске

Кафедра вычислительной техники

УДК: 004.42:004.81

О-66

**Орлов Артем Игоревич**

СПОСОБ И ПРОГРАММНЫЕ СРЕДСТВА СТРУКТУРНО-ПАРАМЕТРИЧЕСКОЙ НАСТРОЙКИ НЕЧЕТКИХ КОГНИТИВНЫХ МОДЕЛЕЙ НА ОСНОВЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Выпускная квалификационная работа на соискание квалификации

**Магистр**

Направление подготовки

**09.04.01 – Информатика и вычислительная техника**

Профиль подготовки

**Информационное и программное обеспечение автоматизированных систем**

Студент группы ВМ-21 (маг) А. И. Орлов

Научный руководитель

д.т.н., профессор В. В. Борисов

*Допускается к защите*

Заведующий кафедрой

д.т.н., профессор А. С. Федулов

«\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2023 г.

Смоленск – 2023

Аннотация

Выпускная квалификационная работа на соискание квалификации «Магистр» по направлению 09.04.01 – Информатика и вычислительная техника. Работу выполнил студент группы ВМ-21 (маг) Орлов Артем Игоревич. Тема магистерской диссертации: «Способ и программные средства структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов». Руководитель: д.т.н., профессор Борисов Вадим Владимирович.

Диссертационная работа изложена на 104 страницах, из них 81 страницы основного текста, состоит из введения, четырех глав, заключения, списка литературы из 36 наименований и двух приложений.

Целью исследования является повышение качества и оперативности структурно-параметрической настройки нечетких когнитивных моделей на основе использования генетических алгоритмов.

В работе выполнен анализ основных типов нечетких когнитивных моделей, их динамических моделей, а также способов структурно-параметрической настройки. Создан способ структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов. Разработаны алгоритмы и программные средства, реализующие предложенный способ при помощи веб-технологий. Выполнена оценка качества и оперативности структурно-параметрической настройки с использованием разработанного способа и программных средств.

Ключевые слова: нечеткие когнитивные модели, настройка когнитивных моделей, обучение когнитивных моделей, генетические алгоритмы

ABSTRACT

Master’s thesis by direction 09.04.01 – Information science and Computer technology. The work was done by student of group VM-21 (mag) Orlov Artem Igorevich. The topic of work is «Method and software for structural and parametric adjustment of fuzzy cognitive models based on genetic algorithms». Supervisor: doctor of technical sciences, Professor Borisov Vadim Vladimirovich.

The work is presented on 104 pages, of which 81 pages of the main text. Paper consists of an introduction, four chapters, a conclusion, a bibliography of 36 titles and two appendices.

The purpose of research is to improve the quality and efficiency of the structural and parametric adjustment of fuzzy cognitive models based on the use of genetic algorithms.

The paper analyzes the main types of fuzzy cognitive models, their dynamic models, as well as methods of structural and parametric adjustment. A method for structural-parametric adjustment of fuzzy cognitive models based on genetic algorithms has been created. Algorithms and software tools have been developed that implement the proposed method using web technologies. An assessment of the quality and efficiency of structural-parametric adjustment was made using the developed method and software.

Keywords: fuzzy cognitive models, cognitive model adjustment, cognitive model training, genetic algorithms

Оглавление

[ВВЕДЕНИЕ 7](#_Toc138183265)

[1 Анализ задач и способов структурно-параметрической настройки нечетких когнитивных моделей 13](#_Toc138183266)

[1.1 Анализ нечетких когнитивных моделей 13](#_Toc138183267)

[1.1.1 Нечеткие когнитивные модели Б. Коско 14](#_Toc138183268)

[1.1.2 Нечеткие когнитивные модели В. Силова 17](#_Toc138183269)

[1.1.3 Нечеткие продукционные когнитивные модели и их обобщенная разновидность 19](#_Toc138183270)

[1.1.4 Нечеткие реляционные когнитивные модели 20](#_Toc138183271)

[1.1.5 «Совместимые» нечеткие когнитивные модели 22](#_Toc138183272)

[1.2 Анализ существующих способов структурно-параметрической настройки нечетких когнитивных моделей 24](#_Toc138183273)

[1.2.1 Структурно-параметрическая настройка нечетких когнитивных моделей Коско 25](#_Toc138183274)

[1.2.2 Структурно-параметрическая настройка нечетких когнитивных моделей Силова 29](#_Toc138183275)

[1.2.3 Параметрическая настройка «совместимых» нечетких когнитивных моделей 31](#_Toc138183276)

[1.3 Постановка задачи исследования 32](#_Toc138183277)

[1.4 Выводы по главе 34](#_Toc138183278)

[2 Разработка способа структурно-параметрической настройки нечетких Когнитивных моделей на основе генетических алгоритмов 35](#_Toc138183279)

[2.1 Общее описание способа структурно-параметрической настройки нечетких когнитивных моделей 35](#_Toc138183280)

[2.2 Алгоритмы генетических операторов структурно-параметрической настройки нечетких когнитивных моделей 39](#_Toc138183281)

[2.2.1 Оператор генерации новой особи 40](#_Toc138183282)

[2.2.2 Оператор скрещивания 41](#_Toc138183283)

[2.2.3 Оператор мутации 43](#_Toc138183284)

[2.2.4 Оператор селекции 44](#_Toc138183285)

[2.3 Генетический алгоритм структурно-параметрической настройки нечетких когнитивных моделей 47](#_Toc138183286)

[2.4 Выводы по главе 50](#_Toc138183287)

[3 разработка программных средств структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов 51](#_Toc138183288)

[3.1 Требования к функционалу программных средств 51](#_Toc138183289)

[3.2 Проектирование основных вариантов использования программных средств 52](#_Toc138183290)

[3.3 Выбор технологий для разработки программных средств 56](#_Toc138183291)

[3.4 Проектирование базы данных программных средств 58](#_Toc138183292)

[3.5 Разработка API серверной части программных средств 62](#_Toc138183293)

[3.6 Разработка интерфейса программных средств 63](#_Toc138183294)

[3.7 Разработка модульной структуры программных средств 65](#_Toc138183295)

[3.8 Развертывание программных средств 68](#_Toc138183296)

[3.9 Выводы по главе 68](#_Toc138183297)

[4 Оценка качества и оперативности разработанного способа структурно-параметрической настройки нечеткиХ когнитивных моделей на основе генетических алгоритмов 69](#_Toc138183298)

[4.1 Оценка качества разработанного способа структурно-параметрической настройки нечетких когнитивных моделей 69](#_Toc138183299)

[4.2 Оценка оперативности разработанного способа структурно-параметрической настройки нечетких когнитивных моделей 70](#_Toc138183300)

[4.3 Решение задачи «Управление ИТ-проектом» с помощью предложенного способа 73](#_Toc138183301)

[4.4 Выводы по главе 76](#_Toc138183302)

[ЗАКЛЮЧЕНИЕ 77](#_Toc138183303)

[Список литературы 78](#_Toc138183304)

[Приложение А 82](#_Toc138183305)

[приложение Б 90](#_Toc138183306)

ВВЕДЕНИЕ

**Актуальность темы исследования.** Одним из активно развивающихся направлений исследований в области поддержки и принятия решений является изучение возможности управления сложными и слабоструктурированными системами и ситуациями [1]. Такие системы плохо поддаются традиционным методам исследования и моделирования из-за отсутствия точной количественной информации о происходящих в них процессах, а доступная исследователю качественная информация часто является неполной и противоречивой. Другая трудность связана с тем, что управленческие решения зачастую приходится принимать быстро и в постоянно изменяющихся условиях.

Несмотря на вышеперечисленные сложности, существуют подходы, позволяющие исследовать такие системы, среди которых особо выделяется нечеткий когнитивный подход. Он позволяет формализовать разрозненные, неполные и противоречивые знания экспертов о слабоструктурированной системе с помощью единой когнитивной модели. Такая модель задается в виде ориентированного взвешенного графа. Вершины графа соответствуют концептам – значимым для цели моделирования и управления факторам системы, а дуги отражают причинно-следственные связи между концептами.

Когнитивный подход нашел широкое применение для моделирования и управления различными организационно-технологическими и социально-экономическими системами. К первым, например, можно отнести моделирование предаварийной ситуации [2] или управление запасами топлива на станции [3], а ко вторым анализ развития социально-экономических ситуаций [4] или моделирование инвестиционной деятельности муниципального образования [5]. Повышение качества и оперативности такого моделирования и управления является перспективным направлением для дальнейших исследований.

Для нечетких когнитивных моделей разработаны различные модели системной динамики, позволяющие ответить на вопрос о том, как моделируемая система будет изменяться с течением времени [6]. Однако кроме прогнозирования такое моделирование по времени может помочь в принятии управленческих решений. Сформулировав цель управления, эксперт может обозначить такое состояния когнитивной модели, достижение которого, по его мнению, будет обозначать достижение поставленной цели. После этого можно попытаться настроить структуру и начальные параметры когнитивной модели таким образом, чтобы она в конечном итоге попала в желаемое конечное состояние. Настроенную когнитивную модель возможно использовать для получения списка управленческих задач, выполнение которых в реальном мире приведет к выполнению цели управления.

Однако задача структурно-параметрической настройки нечеткой когнитивной модели является сложной. Классические алгоритмы оптимизации не обладают достаточной оперативностью для ее решения при большом количестве концептов и связей в модели. Одним из многообещающих подходов является использование генетических алгоритмов, методов оптимизации, вдохновленных естественным процессом эволюции. Этот подход может помочь повысить качество и оперативность структурно-параметрической настройки нечеткой когнитивной модели.

Вопросы настройки нечетких когнитивных моделей при помощи генетического алгоритма рассмотрены в работах [7-11]. В данных работах задача настройки нечетких когнитивных моделей ограничивается лишь изменением отношений взаимовлияния между концептами или изменением вектора начального состояния концептов и, кроме того, ориентирована на отдельные разновидности нечетких когнитивных моделей. Однако для задач анализа сложных и слабоструктурированных систем и ситуаций востребованными являются не только эти разновидности параметрической настройки указанных моделей, но и их структурно-параметрическая настройка, заключающаяся в возможности одновременного изменения значений концептов, весовых коэффициентов отношений взаимовлияния между концептами, состава концептов, а также перечня отношений взаимовлияния между концептами.

Кроме разработки способа структурно-параметрической оптимизации, необходимо предоставить удобный инструмент для его применения менеджерами и экспертами в различных областях. Большинство существующих приложений для разработки когнитивных моделей являются десктопными, что осложняет совместную работу над моделью и возможность делиться результатами. Поэтому разработка веб-приложения для осуществления структурно-параметрической настройки с возможностью совместной работы над когнитивной моделью является важной задачей.

Таким образом, задача разработки способа и программных средств структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов является актуальной, т.к. ее выполнение позволит повысить качество и оперативность управления сложными и слабоструктурированными системами при помощи когнитивного подхода.

**Объектом исследования** являются процессы построения, настройки и использования нечетких когнитивных моделей для анализа сложных систем и процессов. **Предметом исследования** являются способы и программные средства структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов.

**Целью исследования** является повышение качества и оперативности структурно-параметрической настройки нечетких когнитивных моделей на основе использования генетических алгоритмов.

**Научная задача** квалификационной работы заключается в исследовании и разработке способа и программных средств структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов.

Для этого необходимо решить следующие **задачи**:

1. Проанализировать нечеткие когнитивные модели, а также методы и подходы к их структурно-параметрической настройке.
2. Создать способ структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов.
3. Разработать алгоритмы и программные средства структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов.
4. Оценить качество и оперативность структурно-параметрической настройки нечетких когнитивных моделей на основе предлагаемого способа и программных средств.

В ходе работы над квалификационной работой использованы следующие **методы исследований**: нечеткого когнитивного моделирования, объектно-ориентированного и функционального проектирования и программирования, модульного тестирования.

**Обоснованность** научных результатов и выводов, представленных в работе, определяется корректным применением использованных методов исследования и подтверждается результатами компьютерного моделирования. **Достоверность** научных положений подтверждена соответствием теоретических положений и результатов экспериментов на основе компьютерного моделирования и итогами применения разработанного программного средства, а также апробацией основных результатов.

**Научная новизна** работы заключается в следующем:

Разработан новый способ структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов, позволяющий повысить качество и оперативность такой настройки для решения задачи управления сложными и слабоструктурированными системами.

**Практическую значимость** работы составляют следующие результаты:

1. Разработан генетический алгоритм, обеспечивающий программную реализацию предложенного способа структурно-параметрической настройки нечетких когнитивных моделей.
2. Созданы алгоритмы и программные средства структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов.

**На защиту выносятся** следующие положения:

1. Способ структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов.
2. Алгоритмы и программные средства, реализующие структурно-параметрическую настройку нечетких когнитивных моделей с помощью предложенного способа.

**Апробация работы.** По результатам исследования опубликована статья «Способ структурно-параметрической настройки нечетких когнитивных карт» в журнале «Научный аспект» [12].

Выпускная квалификационная работа состоит из введения, четырех глав, заключения, перечня сокращений и условных обозначений и списка литературы, включающего X наименований. Работа содержит X страниц машинописного текста, X рисунков, X таблиц, 2 приложения.

Во введении определена актуальность темы исследования, определены цель и задачи исследования, представлено краткое содержание по главам.

В первой главе были проанализированы основные типы нечетких когнитивных моделей, а также их динамические модели. Затем были показаны существующие способы структурно-параметрической настройки нечетких когнитивных моделей. Далее, исходя из выявленных недостатков этих способов, была поставлена задача разработки нового способа с использованием генетических алгоритмов, который позволил бы повысить качество и оперативность такой настройки.

Во второй главе был описан разработанный способ структурно-параметрической настройки нечетких когнитивных моделей с использованием генетического алгоритма. В этом способе генетический алгоритм подбирает состояние управляющих концептов и веса управляющих связей таким образом, чтобы исследуемая система в результате моделирования перешла в некоторое целевое состояние, задаваемое исследователем.

В третьей главе были описаны разработанные программные средства. Они построены на базе веб-технологий и позволяют создавать проекты нечетких когнитивных моделей Коско и Силова, запускать их структурно-параметрическую настройку с использованием предложенного способа, а также просматривать промежуточные и окончательные результаты этой настройки.

В четвертой главе было оценено качество и оперативность настройки нечетких когнитивных моделей с использованием предложенного способа. Анализ и эксперимент показали, что такая настройка обладает высоким качеством и оперативностью. Также в данной главе с помощью разработанного способа была настроена нечеткая когнитивная модель «Управление ИТ-проектом».

В заключении охарактеризованы результаты выпускной квалификационной работы.

В приложениях представлено задание на выпускную квалификационную работу и исходный код разработанных программных средств.

1. Анализ задач и способов структурно-параметрической настройки нечетких когнитивных моделей
   1. Анализ нечетких когнитивных моделей

Традиционная когнитивная модель представляет собой инструмент для формализации, анализа и моделирования сложных и слабоструктурированных систем и ситуаций. Такая модель задается в виде ориентированного графа, в котором вершины соответствуют концептам – значимым для цели моделирования факторам предметной области, а дуги отражают причинно-следственные связи между концептами. Дуги могут быть помечены знаками «+» (положительное влияние) и знаком «–» (отрицательное влияние), поэтому такие когнитивные модели также называют знаковыми.

Впервые знаковые когнитивные модели описал Р. Аксельрод для изучения процессов принятия решений политическими элитами [13]. Впоследствии благодаря наглядности представления анализируемых систем и ситуаций, а также относительной легкости интерпретации с помощью причинно-следственных связей, такие модели нашли широкое применение в различных сферах, таких как психология, социология, экономика, менеджмент, медицина, урбанистика и др. для анализа структуры и динамики исследуемых систем.

Традиционные когнитивные модели предоставляют широкие возможности для анализа сложных систем и ситуаций. Они позволяют исследователям определить ключевые понятия, их взаимосвязи, а также иерархический и неиерархический характер отношений. Кроме того, такие модели позволяют изучать поведение системы, исследуя причинно-следственные связи между концептами. Это позволяет идентифицировать циклы обратной связи и определить последствия различных вмешательств или изменений, что в свою очередь позволяет анализировать устойчивость системы.

Однако из-за бинарности отношений знаковые когнитивные модели позволяют исследовать характер протекающих процессов в системе лишь качественно и не дают точной количественной оценки силы влияния между концептами. Это ограничивает возможности анализа и моделирования систем, в которых присутствует неточность или неопределенность отношений. Для решения этой проблемы исследователями были предложены различные варианты нечетких когнитивных моделей.

Кроме расширенных возможностей анализа такие модели позволяют осуществлять количественное моделирование по времени, в котором время дискретно и представлено в виде безразмерной шкалы моментов , , , …, . Между модельным и физическим временем предполагается некоторое соответствие, которое задается исследователем. Для каждого момента времени последовательно осуществляется расчет состояния концептов. Формула такого расчета, называемая динамической моделью, зависит от разновидности когнитивной модели, типа моделируемой системы, перечня решаемых задач и используемых средств моделирования. Возможность динамического моделирования обеспечивает широкую применимость нечетких когнитивных моделей в сфере управления и системах принятия решений.

Среди нечетких когнитивных моделей известны следующие разновидности: нечеткие когнитивные модели Б. Коско, нечеткие когнитивные модели В. Силова, нечеткие продукционные когнитивные модели и их обобщенная разновидность, нечеткие реляционные когнитивные модели, а также «совместимые» нечеткие когнитивные модели. Рассмотрим каждую из этих разновидностей, уделив внимание применяемым в них динамическим моделям.

* + 1. Нечеткие когнитивные модели Б. Коско

Нечеткие когнитивные модели Б. Коско являются первым предложенным вариантом нечетких когнитивных моделей [14]. В этих моделях причинно-следственные связи между концептами могут принимать значения в диапазоне от -1 до 1, отражая тем самым степень влияния одного концепта на другой. Концепты при этом могут изменяться в диапазоне от 0 до 1, позволяя тем самым осуществлять моделирование по времени. Разновидностью нечетких когнитивных моделей Коско, предложенной в работах [16, 17] являются модели, в которых диапазон изменения значений концептов расширен до [-1; 1]. Пример нечеткой когнитивной модели Коско представлен на рисунке 4.1.

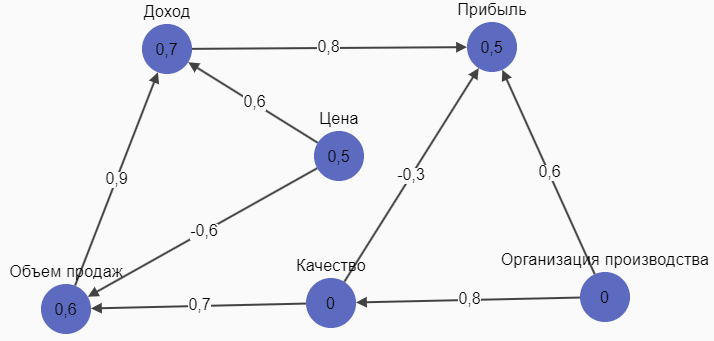


Рисунок 4.1 – Нечеткая когнитивная модель «Организация производства»

Проблема учета отрицательных весов влияния одного концепта на другой решается за счет применения правила замены каждой связи вида на связь вида [14]. Таким образом, если в концепт входит хотя бы одна отрицательная связь, то он разделяется на два, в один из которых входят все положительные связи, а в другой все отрицательные связи, взятые по модулю.

Влияние нескольких входных концептов на выходной концепт может быть аккумулировано следующим образом [14]:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.1) |

где и – значение входного и выходного концептов; – число концептов, у которых есть связь с концептом ; – вес связи между концептами и ; – функция для нормализации результата вычислений в диапазоне от 0 до 1.

Данное влияние схоже с воздействием входных нейронов на выходной в однослойном перцептроне, поэтому в качестве функции нормализации в формуле (1.1) часто используется сигмоидальная функция:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | , . | (1.2) |

Более простым вариантом функции нормализации является кусочно-линейная функция, отсекающая значения, выходящие за пределы диапазона [0; 1]:

|  |  |  |
| --- | --- | --- |
|  |  | . (1.3) |

Основные типы моделей динамики были предложены Робертсом при описании импульсных процессов в ориентированных графах [17] и в применении к нечетким когнитивным моделям Коско могут быть описаны с помощью следующих формул:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.4) |

|  |  |  |
| --- | --- | --- |
|  |  | , (1.5) |

|  |  |  |
| --- | --- | --- |
|  |  | , (1.6) |

где , – дискретные моменты времени; – номер выходного концепта; – общее число концептов модели; , – состояние и приращение значения -го выходного концепта ; , – состояние и приращение значения -го входного концепта; – вес влияния концепта на концепт ; – число входных концептов для -го выходного концепта; – описанная выше функция нормализации.

В работе [18] по аналогии с тремя динамическими моделями Робертсона предложена четвертая модель:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.7) |

Модели (1.4) – (1.7) не позволяют учитывать наличие петель в графе нечеткой когнитивной модели, поэтому в статьях [19, 20] было предложена новая динамическая модель, которая решает данную проблему, а также позволяет «сглаживать» изменение значений концептов в процессе моделирования:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.8) |

Несмотря на то, что когнитивные модели Коско называются нечеткими, в работе [18] отмечается, что их нельзя считать таковыми в полной мере, т.к. нечеткие множества требуют объявления базового множества и степени принадлежности к нему, а в рассматриваемых когнитивных моделях определена лишь степень принадлежности. Кроме того, в этих моделях не применяется математический аппарат теории нечетких множеств Заде.

* + 1. Нечеткие когнитивные модели В. Силова

В нечетких когнитивных моделях В. Силова когнитивная модель представляет собой причинно-следственную сеть, в которой веса связей лежат в диапазоне от 0 до 1 [21], что делает ее похожей на нечеткую когнитивную модель Коско. Анализ когнитивной модели в работе Силова осуществляется при помощи нечеткой когнитивной матрицы смежности. Проблема учета отрицательных весов влияния одного концепта на другой, как и в модели Коско, решается за счет разделения одного концепта на два. В модели Силова это разделение осуществляется за счет построения матрицы положительных связей размера .

Для определения взаимовлияния концептов и решения задач когнитивного моделирования Силов вводит так называемую каузальную алгебру, которая является разновидностью регулярной алгебры. Эта алгебра задается четверкой , где – множество нечетких матриц; – операция ; – макстриангулярная композиция, \* – унарный оператор замыкания.

Статический системный анализ когнитивной модели осуществляется при помощи расчета системных показателей, таких как взаимный консонанс, диссонанс, положительное и отрицательное влияние концептов друг на друга и на систему в целом. Для определения путей, которые привели к той или иной зависимости между концептами, используется так называемый алгоритм интеллектуального когнитивного агента, который позволяет найти максимально положительный и отрицательный пути на графе или максимальный консонанс (диссонанс) между двумя концептами.

Динамическая модель в работе Силова выражается в виде матричного уравнения:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.9) |

где – вектор, характеризующий состояние концептов в момент времени ; – вектор, характеризующий состояние концептов в момент времени ; – операция T-нормы; – операция S-нормы.

По аналогии с когнитивными моделями Коско, используя в качестве T-нормы операцию умножения, а S-нормы – операцию максимума, для пары концептов может быть получена следующая модель динамики:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.10) |

Такая модель из-за использования функции учитывает только наибольшее влияние, что снижает чувствительность результата к входным воздействиям и плохо согласуется с физическим смыслом некоторых моделируемых систем [18].

Когнитивные модели Силова используют математический аппарат нечеткой логики, поэтому они могут быть названы действительно нечеткими. Однако в этих моделях, как и в моделях Коско, для связей применяется лишь степень принадлежности, а базовое множество не определено.

* + 1. Нечеткие продукционные когнитивные модели и их обобщенная разновидность

В работе [22] были предложены нечеткие продукционные когнитивные модели, или по-другому нечеткие когнитивные модели, основанные на правилах. В таких моделях каждый концепт представлен набором функций принадлежности, которые определяют возможные значения концепта или возможные значения его изменения, а связи между концептами выражены в виде нечетких продукционных правил вида «Если …, то». Передача влияния между концептами осуществляется на основе алгоритма нечеткого логического вывода Мамдани.

Для объединения влияния нескольких входных концептов на выходной концепт в [23] вводится специальная операция нечеткого аккумулирования с переносом. В ней разные функции принадлежности входных концептов объединяются с помощью совмещения максимумов. Далее с помощью новой функции принадлежности получают одноточечные значения принадлежностей, которые затем суммируются, а превышение над единицей приводит к переполнению и переносу в сторону значения, представляющего большую вариацию.

Развитием идеи нечетких продукционных когнитивных моделей стали обобщенные нечеткие продукционные когнитивные модели [24]. Каждый концепт такой модели () описывается соответствующей лингвистической переменной , где – терм-множество лингвистической переменной [25] типовых состояний концепта; – число типовых состояний данного концепта; – базовое множество . Для описания термов () используются нечеткие переменные , где – нечеткое множество в базовом множестве :

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | , . | (1.11) |

Веса влияния () между типовыми состояниями каждой пары концептов задаются одним из значений терм-множества лингвистической переменной , где – терм-множество лингвистической переменной ; – число значений ; – базовое множество . Для описания термов используются нечеткие переменные , которые описываются нечеткими множествами в базовом множестве (, , , ) следующего вида:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | , . | (1.12) |

Для обобщенных нечетких продукционных когнитивных моделей может быть предложена следующая модель динамики:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.13) |

где , – нечеткие множества, представляющие значение уровней входного концепта и приращение этого концепта; , – нечеткие множества, представляющие значения уровня и приращения выходного концепта ; – операция нечеткого алгебраического сложения; – нечеткий оператор отображения нескольких входов на один выход; – нечеткий оператор, характеризующий силу влияния концептов по их абсолютным значениям. В работе [26] в качестве нечетких операторов и предлагается использовать нечеткие продукционные модели типа Мамдани.

* + 1. Нечеткие реляционные когнитивные модели

В работе [27] были предложены нечеткие реляционные когнитивные модели, основанные на теории нечетких бинарных отношений Заде [28]. В этих моделях концепты представлены нечеткими множествами, а связи между концептами задаются в виде бинарных нечетких отношений. Такая когнитивная модель может быть удобно описана с помощью матрицы нечетких отношений , элементами которой являются нечеткие отношения , определяющие силу влияния концепта на :

|  |  |  |
| --- | --- | --- |
|  |  | , (1.14) |

где – общее число концептов когнитивной модели.

Передача влияния между двумя концептами когнитивной модели осуществляется на основе нечеткой композиции:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.15) |

где – значение входного концепта в виде нечеткого подмножества множества ; – значение выходного концепта в виде нечеткого подмножества множества ; – нечеткое отношение между входным и выходными концептами; – операция нечеткой композиции.

В реляционных когнитивных моделях существуют два основных способа интерпретации и описания отношений между концептами, представления концептов и передачи влияния между ними. При использовании первого способа нечеткое отношение задается в виде функции принадлежности, зависящей от двух четких переменных базовых множеств входа и выхода. Такая функция может быть задана аналитически, представляя тем самым непрерывную поверхность над плоскостью значений принадлежностей нечетких множеств входного и выходного концептов или с помощью матрицы принадлежностей к нечеткому множеству отношения, построенной по декартовому произведению значений принадлежности нечетких множеств двух концептов.

Во втором способе нечеткие отношения строятся на основе лингвистической нечеткой продукционной модели, включающей в себя нечеткие правила следующего вида:

|  |  |  |
| --- | --- | --- |
|  | : ЕСЛИ есть , ТО есть , | . (1.16) |

где – входная переменная; , – лингвистические термы; – выходная переменная; , , , в общем случае – нечеткие множества, определенные на базовом множестве четких переменных , , и заданные соответствующими функциями принадлежности.

Нечеткое отношение формируется для всей базы правил по следующей формуле:

|  |  |  |
| --- | --- | --- |
|  |  | . (1.17) |

Для нечетких реляционных когнитивных моделей динамическая модель принимает следующий вид:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.18) |

где , , , – нечеткие значения концептов в соответствующие моменты времени; – число входных концептов; – нечеткое отношение между концептами; – операция нечеткого агрегирования совокупных влияний и предыдущего значения выходного концепта; – операция агрегирования отдельных влияний; – операция приращения нечетких значений концептов; – операция нечеткой композиции.

* + 1. «Совместимые» нечеткие когнитивные модели

Последней рассматриваемой разновидностью нечетких когнитивных моделей являются «совместимые» нечеткие когнитивные модели, предложенные в работе [29]. Эти модели возникли из-за необходимости учитывать совместимость концептов при определении влияния между ними, а также при аккумулировании влияния нескольких входных концептов на выходной концепт.

Формально «совместимая» нечеткая когнитивная модель задается следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.19) |

где – множество концептов; – множество весов влияния концептов друг на друга; – множество степеней совместимости пар концептов, [26].

Трактовка совместимости концептов зависит от особенностей решаемой задачи и целей исследования и либо задается экспертами, либо получается в результате экспериментов, либо вычисляется с помощью различных косвенных способов.

Для расчета влияния входного концепта на выходной концепт используется семейство параметризированных функций , удовлетворяющих аксиомам нормировки, неубывания, непрерывности и бисимметричности [30]. При этом в зависимости от решаемой задачи могут быть использованы следующие выражения:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.20) |

|  |  |  |
| --- | --- | --- |
|  |  | , (1.21) |

|  |  |  |
| --- | --- | --- |
|  |  | , (1.22) |

где – значение выходного концепта ; , – значение и приращение входного концепта ; , . Для предотвращения выхода значения концепта за границы разрешенного диапазона можно использовать функцию нормализации, например, сигмоидальную.

Для типизации выбора операций в зависимости от критериальных уровней совместимости концептов можно воспользоваться таблицей 1.1 [31].

Таблица 1.1 – Сопоставление операций оценки влияния концептов с уровнями их совместимости

|  |  |  |  |
| --- | --- | --- | --- |
| № | Операция «взвешивания» нечеткого влияния | Критериальные уровни совместимости | |
| Обозначение | Описание |
| 1 |  | NC | Низкий уровень |
| 2 |  | LC | Уровень ниже среднего |
| 3 |  | MC | Средний уровень |
| 4 |  | HC | Уровень выше среднего |
| 5 |  | FC | Высокий уровень |

Аккумулирование влияния нескольких входных концептов , на выходной концепт может быть выполнено с использований двух основных стратегий: либо от наименьшей к наибольшей степени согласованности концептов, либо от наибольшей к наименьшей степени согласованности концептов. Для построения динамической модели, используя формулу (1.20), можно предложить следующий двуместный предикат :

|  |  |  |
| --- | --- | --- |
|  |  | , (1.23) |

где при использовании стратегии от наименьшей к наибольшей степени согласованности и при использовании стратегии от наибольшей к наименьшей степени приспособленности.

Итоговая динамическая модель выглядит следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  | . (1.24) |

* 1. Анализ существующих способов структурно-параметрической настройки нечетких когнитивных моделей

Моделирование нечеткой когнитивной модели по времени с использованием различных динамических моделей для получения ее конечного состояния в момент времени является прямой задачей. Такое моделирование может ответить на вопрос, является ли исследуемая система устойчивой, а также определить ее реакцию на изменение начального состояния.

Однако часто требуется определить такое начальное состояние системы, которое в определенный момент времени приводит ее в желаемое целевое состояние. Такая задача называется обратной и часто используется в системах управления и принятия решений. Часто решение такой задачи называют настройкой или, по аналогии с нейронными сетями, обучением нечеткой когнитивной модели. Настройка, при которой искомое начальное состояние выражается в виде значений концептов и силы связи между ними, называется параметрической. Если же в результате настройки некоторые концепты могут быть добавлены в модель или удалены из модели, то такая настройка называется структурной. Рассмотрим известные способы структурной и параметрической настройки.

* + 1. Структурно-параметрическая настройка нечетких когнитивных моделей Коско

В структурном и вычислительном отношении нечеткая когнитивная модель Коско похожа на нейронную сеть, в частности на перцептрон. Основным отличием между ними является тот факт, что веса в нейронной сети настраиваются для решения какой-то практической задачи, а веса в нечеткой когнитивной модели задаются экспертами, исходя из их понимания моделируемой системы. В этот отношении когнитивная модель является белым ящиком, т.к. понятно, за что отвечает каждый концепт и каждая связь. Нейронная сеть, в свою очередь, является черным ящиком из-за того, что определить, за что отвечает отдельный нейрон в промежуточном слое и отдельная связь, возможно лишь в самых тривиальных случаях. Вычисление с помощью нейронной сети похоже на динамическое моделирование нечеткой когнитивной модели, однако для нейронной сети, как правило, рассчитывается только одна итерация, а нечеткие когнитивные модели могут моделироваться в течение нескольких отсчетов модельного времени, аккумулируя тем самым предыдущие результаты. Также нечеткие когнитивные модели могут не моделироваться совсем и использоваться исключительно для статического анализа.

Перечисленные различия могут оказаться несущественными, если для когнитивной модели необходимо подобрать только значения весов связей. В таком случае когнитивную модель можно настроить, используя метод обучения с учителем. Такой подход представлен в работе [32]. В ней авторы обучают нечеткую когнитивную модель Коско, предназначенную для стратегического управления бизнесом. В качестве обучающей выборки используются значения концептов, полученные в результате наблюдения или эксперимента. В роли динамической модели выступает формула (1.5), поскольку она является эквивалентом математической модели нейрона без инициализации и сдвига. Начальные веса задаются маленькими случайными значениями, а для изменения значений весов используется метод градиентного спуска.

Такое параметрическое обучение предоставляет определенные возможности по изменению структуры модели, поскольку определенные концепты могут быть исключены из нее, если все их связи в результате настройки установятся в 0. Добавление концептов в модель также возможно. Для этого необходимо соединить концепт-кандидат со всеми остальными концептами и осуществить обучение. Если в результате хотя бы одна связь получила значение отличное от 0, то гипотезу от том, что этот концепт входит в систему можно считать подтвержденной.

В работе [33] была исследована возможность обучать нечеткую когнитивную модель при помощи дифференциального алгоритма обучения Хэбба. Данный алгоритм осуществляет обучение без учителя и не требует обучающей выборки. Основная идея состоит в обновлении тех весов когнитивной модели, которые непосредственно связаны с изменениями значений концептов. Если значения концептов изменяются в одном направлении, то алгоритм увеличивает вес связи, в противном случае, если значения концептов изменяются в противоположном направлении, алгоритм уменьшает вес связи.

Веса связей когнитивной модели в дифференциальном методе обучения Хэбба изменяются по следующей формуле:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.25) |

где – изменение значения входного концепта ; – изменение значения выходного концепта ; и – сила связи между концептами и в момент времени и ; – коэффициент используемый для того, чтобы постепенно забывать старые веса и заменять их новыми.

Коэффициент рассчитывается по следующей формуле:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.26) |

Полный алгоритм с применение формулы (1.25) выглядит следующим образом:

1. Начальные веса задаются небольшими случайными значениями.
2. Осуществляется итерация динамического моделирования с использованием модели динамики (1.5).
3. На основании полученных значений концептов по формуле (1.25) осуществляется изменения весовых коэффициентов связей.
4. Цикл с шага 2, пока значения концептов не стабилизируются с заданной точностью.

Обучение нечеткой когнитивной модели Коско с использованием алгоритма Хэбба также можно условно считать структурно-параметрическим. Добавление и удаление концептов может быть осуществлено с использованием той же методики, что и при использовании стратегии обучения с учителем.

Основная проблема при применении алгоритма Хэбба заключается в том, что при изменении веса связи между двумя концептами и учитываются только эти два концепта, а остальные игнорируются. Для аккумулирования влияния остальных концептов в работе [34] была предложена модификация этого алгоритма, получившая название балансового дифференциального алгоритма обучения Хэбба. В данной модификации формула изменения связей выглядит следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.27) |

Кроме измененной формулы модифицированный алгоритм аналогичен оригинальному.

Еще одним способом структурной-параметрической настройки нечетких когнитивных моделей Коско является использование генетических алгоритмов. Такой подход в целом является более мощным, чем рассмотренные ранее, поскольку позволяет осуществлять точечную настройку. Так, например, генетический алгоритм позволяет настраивать веса отдельных связей модели, зафиксировав остальные. Кроме того, он дает возможность настраивать начальный вектор значений концептов при фиксированных весах связей.

Обучению нечетких когнитивных моделей Коско при помощи генетических алгоритмов посвящено большое количество публикаций. В работе [7] для получения итоговой матрицы весов связей между концептами используется обучающая выборка, что делает такую настройку схожей с настройкой методом градиентного спуска. В статье [8] также настраивается матрица весов между концептами, но уже при помощи одного вектора начального состояния, что делает такой способ похожим на обучение посредством алгоритма Хэбба. Отмеченная выше возможность настраивать вектор начального состояния концептов при фиксированных связях также изучалась и была представлена в работе [9].

* + 1. Структурно-параметрическая настройка нечетких когнитивных моделей Силова

Метод параметрической настройки нечетких когнитивных моделей Силова представлен в работе [21]. В ней осуществляется поиск векторов начального состояния концептов. Для описания метода преобразуем модель динамики (1.9) к следующему выражению:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.28) |

где – матрица, называемая когнитивной системой.

Условно матрицу можно разбить на матрицы – «состояния», – «управления», – выхода, где номеру нулевой строки матрицы будет соответствовать номер столбца в матрице «управления» , а номеру нулевого столбца матрицы будет соответствовать номер строки в матрице «выхода» . Вектор можно разделить на векторы , , , где – вектор управляемых концептов, – вектор концептов-состояний, – вектор выходных концептов. В результате из формулы (1.28) можно получить:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.29) |

При разделении (1.29) получить следующие выражения:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.30) |

|  |  |  |
| --- | --- | --- |
|  |  | , (1.31) |

Под действием вектора значения концептов изменяются, поэтому, усиливая или ослабляя «управляющие» концепты, можно добиться усиления или ослабления концептов «состояния», в том числе и желаемого, целевого. Поэтому настройка нечеткой когнитивной модели заключается в поиске множества начальных векторов управляющих концептов при заданном нечетком векторе – цели по состояниям.

Для решения задачи необходимо найти множество нечетких векторов , которое обеспечит решение

|  |  |  |
| --- | --- | --- |
|  |  | , (1.32) |

где искомый целевой вектор; ; – передаточная матрица .

Такое реляционное уравнение имеет одну верхнюю границу решения и множество нижних решений. В соответствии с условиями существования обратных решений, множество решений уравнения определяется из условия:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | , | , (1.33) |

Использование генетических алгоритмов для настройки нечеткой когнитивной модели Силова было исследовано в работе [10]. В ней в отличии от метода описанного выше настраиваются веса связей модели, а не значения концептов. Авторы указывают, что задача сводится к определению приращений значений концептов, возникающих в модели при внесении в нее изначального вектора приращений значений концептов. Исходными данными алгоритма служат 3 вектора начальных значений концептов , , в моменты времени , , соответственно. В качестве хромосомы используется одномерный массив значений, в который разложен двумерный массив весов модели. Исходный вектор приращений задается формулой:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.34) |

а результирующий – формулой:

|  |  |  |
| --- | --- | --- |
|  |  | . (1.35) |

Обучение состоит в минимизации ошибки, вычисляемой по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | , (1.36) |

где – приращение -го фактора, полученное в результате прогноза на исходном векторе .

В работе [11] также представлен способ настройки весов связей нечеткой когнитивной модели Силова с помощью генетического алгоритма. В ней когнитивное моделирование применяется для построения системы принятия решений. Из всего множества концептов нечеткой когнитивной модели выбираются те концепты, которые выражают цели управления – список концептов-целей, а также концепты, влияние на которые отражает все возможные действия, которые можно применить для достижения целей – список концептов-действий. В когнитивную модель также добавляется дополнительный концепт – решение. Данный концепт соединяется связями со всеми концептами из списка концептов-действий. Веса этих связей определяют, с какой силой воздействует концепт-решение на все концепты-действия. Далее путем настройки силы этих связей с помощью генетического алгоритма, происходит поиск самого эффективного и достоверного решения. Основными особенностями настройки нечеткой когнитивной модели в данной работе является то, что настройка осуществляется не при помощи модели динамики, а с помощью системных показателей, а также то, что настройке подвергается лишь часть связей.

* + 1. Параметрическая настройка «совместимых» нечетких когнитивных моделей

Для определения значений или диапазонов значений различных концептов «совместимой» нечеткой когнитивной модели по заданному значению (диапазону значений) целевого концепта используются разработанные в [30] правила определения значений (диапазонов значений) аргументов и параметризированных операций типа по значению представленные в таблице 1.2.

Таблица 1.2 – Правила определения значений (диапазонов значений) аргументов и параметризированной операции по результирующему значению

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Условия | | | | |
|  |  | или |  |  |
| Вариант решения 1 |  |  |  |  | 0 |
| Вариант решения 2 |  |  |  | 0 | 0 |
| – нижняя граница оценок искомых значений и позволяющая ограничивать варианты решений условиями и , устанавливая в соответствии | | | | | |

Эти правила применимы как для определения значений (диапазонов значений) входного концепта по значению непосредственно связанного с ним выходного концепта, так и для случая определения значений входных концептов по аккумулированному значению выходного концепта. Во втором случае требуется учитывать очередность операций, в зависимости от выбранной стратегии аккумулирования влияний.

* 1. Постановка задачи исследования

Из анализа существующих способов структурно-параметрической настройки видно, что все они могут быть разделены на два основных типа. В первом исходными данными является вектор (вектора) начального состояния концептов модели, а настройке повергается матрица связей между концептами. Такая настройка похожа на обучение нейронной сети, поэтому для нее применяются известные способы обучения, такие как обучение с учителем при помощи градиентного спуска или обучение без учителя с помощью алгоритма Хэбба. Кроме того, для такой настройки могут применяться генетические алгоритмы. В частности, генетические алгоритмы позволяют настраивать только часть связей модели.

Во втором типе настройке подвергается вектор начального состояния концептов при фиксированных весах связей. Как правило, при таком обучении концепты делятся на управляющие и целевые. Первые могут изменить свои значения в результате настройки, а вторые отражают цель, стоящую перед исследователем. Для осуществления настройки используются теоретические методы решения обратной задачи системной динами, хотя использование генетических алгоритмов также возможно.

Из этого следует, что существующие способы в основном направлены на осуществление параметрической настройки только по одной из двух координат. Структурная настройки при применении некоторых из этих способов возможна, но ее возможности сильно ограничены. Однако в реальных задачах может потребоваться обучить нечеткую когнитивную модель одновременно по двум координатам. При таком обучении по аналогии с управляющими концептами может быть введено понятие управляющих связей. Кроме того, в существующих способах на значения концептов и на веса связей между ними не накладывается никаких ограничений, но при этом в реальных системах такие ограничения могут присутствовать, и для их получения может быть использован экспертный подход.

Еще одним недостатком известных способов является тот факт, что все они направлены на обучение только одного типа нечетких когнитивных моделей. Вместе с тем анализ разновидностей моделей показал, что, несмотря на различия в значениях концептов и связей, для каждой из них может быть построена схожая динамическая модель.

Исходя из вышесказанного, задача исследования состоит в разработке обобщенного способа структурно-параметрической настройки нечетких когнитивных моделей, позволяющего обучать любую из их разновидностей по двух координатам с учетом ограничений, наложенных экспертами. В данном способе должны быть применены генетические алгоритмы из-за гибкости и оперативности этого подхода. Таким образом, предлагаемый обобщенный способ должен повышать качество структурно-параметрической настройки за счет использования двух координат и ограничений, а также оперативность настройки за счет использования генетических алгоритмов.

Для демонстрации созданного способа структурно-параметрической настройки требуется разработать программные средства, позволяющие обучать нечеткие когнитивные модели Коско и Силова с возможностью использования любой модели динамики, доступной для этих разновидностей когнитивных моделей.

* 1. Выводы по главе

В данной главе были проанализированы основные разновидности нечетких когнитивных моделей, такие как нечеткие когнитивные модели Коско, нечеткие когнитивные модели Силова, нечеткие продукционные когнитивные модели и их обобщенная разновидность, нечеткие реляционные когнитивные модели, а также «совместимые» нечеткие когнитивные модели. Для каждой из разновидностей была описана динамическая модель, позволяющая осуществлять моделирование по времени.

Далее были описаны известные способы структурно-параметрической настройки нечетких когнитивных моделей Коско, Силова, а также «совместимых» нечетких когнитивных моделей.

В завершении были выявлены недостатки существующих способов настройки и поставлена задача разработки обобщенного способа структурно-параметрической настройки нечетких когнитивных моделей с использование генетических алгоритмов, позволяющего повысить качество и оперативность такой настройки.

1. Разработка способа структурно-параметрической настройки нечетких Когнитивных моделей на основе генетических алгоритмов
   1. Общее описание способа структурно-параметрической настройки нечетких когнитивных моделей

Как было обозначено в первой главе, разрабатываемый способ структурно-параметрической настройки должен быть применим к различным типам нечетких когнитивных моделей, таким как нечеткие когнитивные модели Коско, нечеткие когнитивные модели Силова, обобщенные нечеткие продукционные когнитивные модели, нечеткие реляционные когнитивные модели, а также «совместимые» нечеткие когнитивные модели.

При применении разработанного способа нечеткая когнитивная модель любой разновидности строится с использованием экспертного подхода по известным принципам. Затем из всех концептов модели выделяют управляющие и целевые. Первые отражают факторы, на которые можно повлиять и скорректировать таким образом поведение и конечное состояние моделируемой слабоструктурированной системы. Вторые представляют собой важные для исследования факторы, которые невозможно изменить непосредственно. Конечное состояние целевых концептов зависит от начального состояния управляющих.

На значения управляющих концептов могут быть наложены некоторые ограничения, которые выражают тот факт, что на реальные системы можно оказывать влияние лишь в некотором разрешенном диапазоне. Для нечетких когнитивных моделей Коско и Силова, а также «совместимых» нечетких когнитивных моделей ограничение концепта задается в виде отрезка или интервала в области определения значений концептов от 0 до 1. Ограничение для обобщенных нечетких продукционных моделей и нечетких реляционных когнитивных моделей может быть представлено в виде отрезка, интервала или множества значений из базового множества концепта. Примеры ограничений концептов для различных типов нечетких когнитивных моделей представлены в таблице 2.1.

Таблица 2.1 – Примеры ограничений концептов для различных типов нечетких когнитивных моделей

|  |  |
| --- | --- |
| **Нечеткая когнитивная модель** | **Примеры ограничений** |
| Нечеткая когнитивная модель Коско |  |
| Нечеткая когнитивная модель Силова |  |
| Обобщенная нечеткая продукционная когнитивная модель |  |
| Нечеткая реляционная когнитивная модель |  |
| «Совместимая» нечеткая когнитивная модель |  |

Для целевых концептов задаются значения, отражающие желаемое состояние системы. Вид этих значений для различных типов нечетких когнитивных моделей аналогичен виду ограничений, накладываемых на управляющие концепты, и выражается в виде отрезков, интервалов или множеств, заданных на области определения значений соответствующих целевых концептов.

Среди связей нечеткой когнитивной модели также выделяют управляющие. Значение силы этих связей не известно исследователю заранее, а может изменяться в процессе настройки в рамках определенных ограничений. Изменение силы управляющей связи приводит к изменению конечного состояния целевых концептов в результате моделирования. Для нечетких когнитивных моделей Коско и Силова ограничение связи задается в виде отрезка или интервала в области определения силы связей от -1 до 1. Ограничение для обобщенных нечетких продукционных моделей может быть представлено в виде отрезка, интервала или множества значений из базового множества связи. Для нечетких реляционных когнитивных моделей ограничение выражается в виде матрицы отрезков или интервалов, определяя тем самым границы, в рамках которых может изменяться нечеткое отношение. Для «совместимых» нечетких когнитивных моделей ограничение описывается в виде двуместного кортежа отрезков или интервалов, в котором первый элемент определен в границах от -1 до 1 и выражает ограничение, накладываемое на силу связи, а второй – в границах от 0 до 1 и задает ограничение, накладываемое на степень совместимости концептов. Примеры ограничений связей для различных типов нечетких когнитивных моделей представлены в таблице 2.2

Таблица 2.2 – Примеры ограничений связей для различных типов нечетких когнитивных моделей

|  |  |
| --- | --- |
| **Нечеткая когнитивная модель** | **Примеры ограничений** |
| Нечеткая когнитивная модель Коско |  |
| Нечеткая когнитивная модель Силова |  |
| Обобщенная нечеткая продукционная когнитивная модель |  |
| Нечеткая реляционная когнитивная модель |  |
| «Совместимая» нечеткая когнитивная модель |  |

Если разновидность нечеткой когнитивной модели позволяет использовать несколько динамических моделей, то для любого концепта может быть выбрана любая из них. Так, например, для концепта нечеткой когнитивной модели Коско может быть выбраны любая из моделей: (1.4), (1.5), (1.6), (1.7). Установка динамической модели для отдельных концептов не является обязательной, поскольку ко всем концептам модели может применяться единая модель динамики, устанавливаемая при запуске структурно-параметрической настройки.

После выбора управляющих концептов и их ограничений, целевых концептов и их значений, динамических моделей для каждого концепта, а также управляющих связей и их ограничений запускается генетический алгоритм структурно-параметрической настройки, в котором особь представляет собой часть нечеткой когнитивной модели, у которой значения управляющих концептов и связей установлены в некоторые значения, удовлетворяющие установленным ограничениям.

Для каждой особи выполняется динамическое моделирование, определяющее ее приспособленность. При этом ошибка, непосредственно влияющая на приспособленность, для отдельного целевого концепта на некотором шаге моделирования определяется следующим образом. Если полученное значение этого концепта попадает в заданный целевой диапазон, то ошибка считается равной 0.

В противном случае ошибка вычисляется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | , (2.1) |

где– полученное значение целевого концепта в момент времени , а – минимальное значение целевого диапазона, если находится слева от диапазона, и максимальное значение, если находится справа.

Поскольку не все из рассматриваемых систем является устойчивыми, моделирование с разными значениями максимального модельного времени может привести систему в разное состояние. Для учета этого фактора будем считать, что исследователю известен некоторый диапазон модельного времени, в котором он хочет найти решение. Тогда в качестве ошибки будем рассматривать минимальную ошибку, полученную в течение этого диапазона.

Результатом применения генетического алгоритма должна стать нечеткая когнитивная модель, состояние которой после моделирования является наиболее близким к искомому целевому состоянию. Если в результате применения алгоритма удовлетворительное решение не было найдено, то исследователю необходимо попробовать либо изменить его настройки, либо установить менее жесткие ограничения на управляющие концепты и связи, т.к. возможно, что в рамках установленных ограничений решения не существует.

Стоит отметить, что в рассмотренном способе путем изменения состояния управляющих концептов и связей осуществляется не только параметрическая настройка, но и структурная. Если существует концепт-кандидат, который может быть потенциально включен в модель для достижения поставленной цели, то необходимо сделать все его связи с другими концептами управляющими, наложив такие ограничения, чтобы нулевое значение попадало в разрешенный диапазон. Таким образом, в результате настройки все связи с концептом-кандидатом могут установиться в 0, что означает, что цель достигается при отсутствии влияния этого концепта на остальные. Интерпретация данного факта остается за исследователем. Если влияние концепта-кандидата в реальной системе и так отсутствует, но его возможно добавить, то такой концепт следует убрать из рассмотрения и исключить из модели. Если же влияние в реальной системе существует, но его можно снизить до 0, то такое снижение должно стать задачей управления, а концепт необходимо оставить в модели.

Если исследователь может влиять не на связи концепта-кандидата, а на его значение, то вместо связей управляющим должен стать сам концепт. При решении об его удалении следует руководствоваться теми же принципами, что и для управляющих связей.

Аналогично концепту-кандидату в модель может быть добавлена связь-кандидат. Такая связь должна быть управляющей для принятия решения об ее включении в модель. Ситуация, при которой управляющими становятся концепты, которые соединяет связь-кандидат, также возможна, но ее интерпретация более сложна и зависит от моделируемой системы и целей исследователя.

* 1. Алгоритмы генетических операторов структурно-параметрической настройки нечетких когнитивных моделей

Как было отмечено выше, в рассматриваемом способе структурно-параметрической настройки особь выражается в виде состояния управляющих концептов и весов управляющих связей. Вид такой особи представлен на рисунке 2.1, где – хромосома состояний управляющих концептов; – число управляющих концептов; – хромосома весов управляющих связей; – число управляющих связей.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | |  | | | |
|  |  |  |  |  |  |  |  |

Рисунок 2.1 – Вид особи

Для реализации генетического алгоритма требуются операторы генерации новой особи, скрещивания, мутации и селекции. Рассмотрим каждый из этих операторов.

* + 1. Оператор генерации новой особи

Пусть ограничения, накладываемые на управляющие концепты, представлены в виде списка , где каждый элемент является отрезком, интервалом или пустым значением null, а – границы областей определения значений управляющих концептов, тогда схема алгоритма генерации хромосомы представлена на рисунке 2.2.



Рисунок 2.2 – Схема алгоритма генерации хромосомы

Данный алгоритм генерирует значение управляющего концепта в диапазоне возможно наложенного ограничения . Алгоритм генерации хромосомы выглядит аналогично, за тем исключением, что для нечеткой реляционной когнитивной модели алгоритм генерирует матрицу реляционного отношения для каждой связи, а для «совместимой» нечеткой когнитивной модели – двуместный кортеж.

* + 1. Оператор скрещивания

Хромосомы содержат действительные числа, поэтому в качестве метода скрещивания целесообразно выбрать скрещивание смешением [35]. Каждый потомок при таком скрещивании случайным образом выбирается из следующего отрезка, созданного родителями и :

|  |  |  |
| --- | --- | --- |
|  |  | , (2.2) |

где – константа от 0 до 1.

Значение родителя должно быть больше значения родителя . Если это не так, то родители меняются местами. Константа определяет ширину отрезка. При значении отрезок получается в два раза шире, чем между родителями, поэтому такое значение является наиболее целесообразным. Однако полученный таким методом отрезок может выходить за пределы диапазона, установленного ограничением или областью определения значений концепта или силы связи, поэтому его необходимо соответствующим образом скорректировать.

Схема алгоритма скрещивания двух хромосом управляющих концептов и представлена на рисунке 2.3. Алгоритм скрещивания двух хромосом управляющих связей строится аналогично. Итоговый алгоритм скрещивания с вероятностью, равной генерирует двух потомков, используя эти два алгоритма, а с вероятностью копирует родителей в следующее поколение без скрещивания.



Рисунок 2.3 – Схема алгоритма скрещивания хромосом и

* + 1. Оператор мутации

Для осуществления мутации хромосомы необходимо заменить один из ее генов на новый. Получение нового случайного значения гена осуществляется аналогично получению отдельного значения при генерации хромосомы. Выполнять мутацию необходимо с некоторой вероятностью. Пусть вероятность одновременной мутации обеих хромосом и особи равняется , а вероятность мутации только одной из хромосом равняется , тогда в одной половине случаев вероятности мутаций хромосом и равны и соответственно, а в другой половине случаев эти вероятности равны и . Схема алгоритма мутации особи представлена на рисунке 2.4.



Рисунок 2.4 – Схема алгоритма мутации особи

* + 1. Оператор селекции

Для выполнения селекции необходимо рассчитать приспособленность всех особей текущего поколения. Чтобы осуществить это для каждой особи выполняется моделирование по времени, способ которого зависит от разновидности нечеткой когнитивной модели. Итоговая ошибка особи вычисляется по следующей формуле:

|  |  |  |
| --- | --- | --- |
|  |  | , (2.3) |

где – значение целевого концепта в момент времени ; – искомое целевое значение концепта ; – минимальное время, в котором осуществляется поиск решения; – максимальное время, в котором осуществляется поиск решения; – число целевых концептов.

Целевое значение концепта задается в виде диапазона, поэтому значение может быть получено по следующей формуле, если диапазон является отрезком:

|  |  |  |
| --- | --- | --- |
|  |  | , (2.4) |

где – целевой отрезок.

Таким образом, чем меньше ошибка особи, тем выше ее приспособленность, а задача структурно-параметрической настройки сводится к минимизации этой ошибки. Для демонстрации на рисунке 2.5 представлена схема алгоритма моделирования по времени нечеткой когнитивной Коско с использованием динамической модели 1.6 и расчетом ошибки по формулам 2.3 и 2.4. В данном алгоритме – общее число концептов; – общее числе связей; – список концептов; – список связей; – список целевых концептов.



Рисунок 2.5 – Схема алгоритма моделирования нечеткой модели Коско с расчетом ошибки

В качестве метода селекции был выбран турнирный отбор [35]. В каждом раунде такого отбора из популяции выбираются две особи, и та, у которой ошибка меньше, выигрывает и отбирается в следующее поколение. Схема итогового алгоритма селекции представлена на рисунке 2.6. В данном алгоритме – размер поколение, – текущее поколение, на котором выполняется скрещивание, а результат – список родителей, выбранных для скрещивания.



Рисунок 2.6 – Схема алгоритма селекции

* 1. Генетический алгоритм структурно-параметрической настройки нечетких когнитивных моделей

После построения основных генетических операторов для получения итогового генетического алгоритма необходимо определиться с условиями его остановки. Очевидно, что алгоритм должен останавливаться, если значение ошибки наилучшей особи поколения меньше, чем некоторое заданное. В случае, если алгоритм по тем или иным причинам не может найти приемлемое решение, он должен останавливаться при достижении определенного числа поколений. Также о том, что решение не может быть найдено может свидетельствовать тот факт, что в течение нескольких поколений не наблюдается уменьшение средней ошибки популяции.

Еще одним важным фактором, влияющим на сходимость алгоритма, является размер популяции. Общий подход для его определения вытекает из утверждения, что особи популяции должны содержать максимальное количество различных значений, которые покрывают большую часть области поиска. Однако точно определить значение размера популяции для конкретной когнитивной модели не представляется возможным. Зачастую, это значение определяется методом проб и ошибок, поэтому оно должно быть включено в список входных данных алгоритма.

Таким образом, для итогового алгоритма можно обозначить следующий список входных данных:

* *KA* – список всех концептов;
* – список управляющих концептов;
* – список ограничений, накладываемых на управляющие концепты;
* – список областей определения значений концептов;
* – список всех связей;
* – список управляющих связей;
* – список ограничений, накладываемых на управляющие связи;
* – список областей определения весов связей;
* – список значений целевых концептов;
* – динамическая модель (для тех типов когнитивных моделей, которые позволяют применять различные типы динамических моделей);
* – минимальное модельное время, в котором осуществляется поиск решения;
* – максимальное модельное время, в котором осуществляется поиск решения;
* – ошибка, при которой решение считается удовлетворительным;
* – максимальное число поколений;
* – максимальное число поколений без улучшений;
* – размер поколения.

Схема итогового алгоритма, использующая данный список входных данных, а также описанные ранее генетические операторы представлена на рисунке 2.7. Алгоритм решает поставленную задачу, однако он имеет некоторый потенциал для ускорения сходимости. Так, например, для этого может быть применен принцип элитизма [35], который заключается в том, что определенное количество лучших особей копируются в следующее поколение, до того, как все места будут заняты потомками, полученными в результате селекции, скрещивания и мутации. Элитизм может оказать положительный эффект на качество алгоритма, поскольку не нужно тратить время на повторное открытие хороших решений, потерянных в результате эволюции.

Другим способом улучшение сходимости может являться использование ниш. Например, исходная популяция может быть разделена на три ниши, где особи первой стремятся к левой границе целевых диапазонов, особи второй – к правой границе, а особи третьей – к середине. Также ниши могут быть образованы с помощью разделения диапазона модельного времени, в котором осуществляется решение, на несколько. При этом особи каждой ниши будут стремиться к решению в своем диапазоне. Использование ниш при условии возможности скрещивания между ними может повысить генетическое разнообразие популяции и позволить быстрее найти решение.



Рисунок 2.7 – Схема генетического алгоритма

* 1. Выводы по главе

В данной главе был разработан способ структурно-параметрической настройки нечетких когнитивных моделей с использованием генетического алгоритма. В данном способе концепты когнитивной модели разделяются на управляющие и целевые, а также вводится понятие управляющих связей. Поиск значений целевых концептов осуществляется с помощью изменения состояния управляющих концептов и весов управляющих связей в рамках существующих ограничений.

1. разработка программных средств структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов
   1. Требования к функционалу программных средств

Разрабатываемые программные средства должны предоставлять пользователю интерфейс для осуществления структурно-параметрической настройки нечеткой когнитивной модели Коско или Силова. Отличие между ними в рамках рассматриваемого способа заключается только в использовании разных динамических моделей, поэтому функционал, связанный с их редактированием, может быть общим. Также программа должна позволять работать над когнитивной моделью нескольким экспертам. Таким образом, список требуемых функциональных возможностей выглядит следующим образом:

* регистрация и авторизация пользователей;
* создание проектов нечетких когнитивных моделей;
* добавление пользователей в проект;
* добавление, изменение и удаление концептов;
* выбор управляющих и целевых концептов;
* выбор динамических моделей для каждого концепта;
* выбор ограничений для управляющих концептов и целевых диапазонов для целевых концептов;
* добавление, изменение и удаление связей;
* выбор ограничений для управляющих связей;
* выбор параметров генетического алгоритма структурно-параметрической настройки;
* запуск алгоритма структурно-параметрической настройки;
* сохранение промежуточных и окончательных результатов структурно-параметрической настройки с возможностью просмотра.
  1. Проектирование основных вариантов использования программных средств

Формализуем основные варианты использования разрабатываемых программных средств с помощью диаграммы вариантов использования, представленной на рисунке 3.1.



Рисунок 3.1 – Диаграмма вариантов использования

Опишем варианты использования, непосредственно связанные со структурно-параметрической настройкой. Краткое описание варианта использования «Добавить концепт» представлено в таблице 3.1.

Таблица 3.1 – Краткое описание варианта использования «Добавить концепт»

|  |  |
| --- | --- |
| Название | Добавить концепт |
| Действующие лица | Участник проекта |
| Цель | Создать новый концепт |
| Предусловие | Открыта страница проекта нечеткой когнитивной модели |

Таблица 3.1 – Краткое описание варианта использования «Добавить концепт» (продолжение)

|  |  |
| --- | --- |
| Успешный сценарий | 1. Участник проекта нажимает на кнопку «Добавить концепт» для перехода в режим добавления концептов. 2. Участник проекта кликает в то место на рабочей области, куда он хочет добавить концепт. 3. Открывается окно добавления концепта. 4. Участник проекта вводит название концепта, его описание, значение, а также при необходимости меняет координаты. 5. Участник проекта нажимает на кнопку «Добавить». 6. Концепт добавляется на рабочую область. |
| Результат | В проект нечеткой когнитивной модели добавлен новый концепт |

Краткое описание варианта использования «Удалить концепт» представлено в таблице 3.2.

Таблица 3.2 – Краткое описание варианта использования «Удалить концепт»

|  |  |
| --- | --- |
| Название | Удалить концепт |
| Действующие лица | Участник проекта |
| Цель | Удалить существующий концепт |
| Предусловие | Открыта страница проекта нечеткой когнитивной модели |
| Успешный сценарий | 1. Участник проекта нажимает на кнопку «Изменить» для перехода в режим редактирования концептов и связей. 2. Участник проекта кликает по концепту, который он хочет удалить. 3. Справа открывается меню по изменению или удалению концепта. 4. Участник проекта нажимает на кнопку «Удалить» или на клавишу «Delete». 5. Концепт удаляется из рабочей области. |
| Результат | Из проекта нечеткой когнитивной модели удален существующий концепт |

Краткое описание варианта использования «Изменить концепт» представлено в таблице 3.3.

Таблица 3.3 – Краткое описание варианта использования «Изменить концепт»

|  |  |
| --- | --- |
| Название | Изменить концепт |
| Действующие лица | Участник проекта |
| Цель | Изменить параметры концепта |
| Предусловие | Открыта страница проекта нечеткой когнитивной модели |

Таблица 3.3 – Краткое описание варианта использования «Изменить концепт» (продолжение)

|  |  |
| --- | --- |
| Успешный сценарий | 1. Участник проекта нажимает на кнопку «Изменить» для перехода в режим редактирования концептов и связей. 2. Участник проекта кликает по концепту, который он хочет изменить. 3. Справа открывается меню по изменению или удалению концепта. 4. На вкладке «Концепт» участник проекта изменяет название, описание, значение, а также координаты концепта. 5. Участник проекта нажимает на кнопку «Изменить». 6. Концепт изменяет свои параметры на рабочей области. 7. На вкладке «Настройка» участник проекта при необходимости помечает концепт как управляющий или целевой, возможно добавляя при этом ограничение или целевой диапазон. Также на этой вкладке участник проекта при необходимости выбирает динамическую модель концепта. 8. При изменении каждого из пунктов на вкладке «Настройка» участник нажимает на кнопку «Изменить». 9. При изменении типа концепта, он меняет свой цвет на рабочей области, т.к. цвет концепта свидетельствует о том является ли концепт обычным, управляющим или целевым. |
| Результат | В проекте нечеткой когнитивной модели изменены параметры концепта |

Краткое описание варианта использования «Добавить связь» представлено в таблице 3.4.

Таблица 3.4 – Краткое описание варианта использования «Добавить связь»

|  |  |
| --- | --- |
| Название | Добавить связь |
| Действующие лица | Участник проекта |
| Цель | Создать новую связь |
| Предусловие | Открыта страница проекта нечеткой когнитивной модели |
| Успешный сценарий | 1. Участник проекта нажимает на кнопку «Добавить связь» для перехода в режим добавления связей. 2. Участник проекта кликает по выходному-концепту. 3. Участник проекта кликает по входному-концепту. 4. Открывается окно добавления связи. 5. Участник проекта вводит описание и силу связи. 6. Участник проекта нажимает на кнопку «Добавить». 7. Связь добавляется на рабочую область. |
| Результат | В проект нечеткой когнитивной модели добавлена новая связь |

Краткое описание варианта использования «Удалить связь» представлено в таблице 3.5.

Таблица 3.5 – Краткое описание варианта использования «Удалить связь»

|  |  |
| --- | --- |
| Название | Удалить связь |
| Действующие лица | Участник проекта |
| Цель | Удалить существующую связь |
| Предусловие | Открыта страница проекта нечеткой когнитивной модели |
| Успешный сценарий | 1. Участник проекта нажимает на кнопку «Изменить» для перехода в режим редактирования концептов и связей. 2. Участник проекта кликает по связи, которую он хочет удалить. 3. Справа открывается меню по изменению или удалению связи. 4. Участник проекта нажимает на кнопку «Удалить» или на клавишу «Delete». 5. Связь удаляется из рабочей области. |
| Результат | Из проекта нечеткой когнитивной модели удалена существующая связь |

Краткое описание варианта использования «Изменить связь» представлено в таблице 3.6.

Таблица 3.6 – Краткое описание варианта использования «Изменить связь»

|  |  |
| --- | --- |
| Название | Изменить связь |
| Действующие лица | Участник проекта |
| Цель | Изменить параметры связи |
| Предусловие | Открыта страница проекта нечеткой когнитивной модели |
| Успешный сценарий | 1. Участник проекта нажимает на кнопку «Изменить» для перехода в режим редактирования концептов и связей. 2. Участник проекта кликает по связи, которую он хочет изменить. 3. Справа открывается меню по изменению или удалению связи. 4. На вкладке «Связь» участник проекта изменяет описание и силу связи. 5. Участник проекта нажимает на кнопку «Изменить». 6. Связь изменяет свои параметры на рабочей области. 7. На вкладке «Настройка» участник проекта при необходимости помечает связь как управляющею, возможно добавляя при этом ограничение. 8. При изменении каждого из пунктов на вкладке «Настройка» участник нажимает на кнопку «Изменить». 9. При изменении типа связи, она меняет свой цвет на рабочей области, т.к. цвет связи свидетельствует о том является ли связь обычной или управляющей. |
| Результат | В проекте нечеткой когнитивной модели изменены параметры связи |

Краткое описание варианта использования «Осуществить структурно-параметрическую настройку» представлено в таблице 3.7.

Таблица 3.7 – Краткое описание варианта использования «Осуществить структурно-параметрическую настройку»

|  |  |
| --- | --- |
| Название | Осуществить структурно-параметрическую настройку |
| Действующие лица | Участник проекта |
| Цель | Запустить структурно-параметрическую настройку и просмотреть ее результат |
| Предусловие | Открыта страница структурно-параметрической настройки проекта нечеткой когнитивной модели |
| Успешный сценарий | 1. Участник проекта нажимает на кнопку «Настроить». 2. Открывается меню структурно-параметрической настройки. 3. Участник проекта вводит название настройки, описание настройки, минимальное модельное время, максимальное модельное время, динамическую модель, размер поколения, интервал сохранения поколений, максимальное число поколений, максимальное число поколений без улучшений, а также ошибку. 4. Участник проекта нажимает на кнопку «Настроить». 5. В таблицу структурно-параметрической настройки добавляется новая запись. 6. Участник проект ждет завершения настройки. 7. Участник проекта переходит по ссылке «Результат» в строке добавленной настройки. 8. Открывается нечеткая когнитивная модель с настроенными значениями управляющих концептов и весами управляющих связей. |
| Результат | Осуществлена структурно-параметрическая настройка нечеткой когнитивной модели |

* 1. Выбор технологий для разработки программных средств

Веб-приложение является наиболее подходящим способом реализации программных средств, поскольку такая архитектура позволяет без лишних сложностей реализовать многопользовательский режим. Для построения веб-приложения необходимо выбрать технологии для реализации серверной и клиентской части.

Выполнение генетического алгоритма требует больших вычислительных мощностей, поэтому для его реализации требуется производительный язык программирования. Скриптовые языки, такие как Python или JavaScript плохо подходят для этой задачи из-за их неэффективности в вычислительном смысле. Более быстрые компилируемые языки с автоматическим управлением памятью, запускаемые с помощью среды исполнения, такие как Java или C# подходят лучше, однако они также могут оказаться недостаточно эффективными. Лучшим вариантом является использование языков, компилируемых напрямую в машинный код, т.к. именно они предоставляют самую высокую производительность. Среди такие языков можно выделить C, C++, Go и Rust. Однако платой за высокую производительность обычно является более высокая сложность и подверженность ошибкам, связанным с ручным управлением памятью. Для разработки серверной части был выбран язык программирования Rust, поскольку он был разработан с целью решить эти проблемы. Rust предоставляет особый механизм управления памятью, основанный на концепции владения переменной, позволяющий избежать большинства ошибок. Кроме того, Rust дает возможность использования большого количества абстракций без падения производительности [36].

В качестве фреймворка для разработки серверной части был выбран Rocket, поскольку он является самым развитым в экосистеме Rust. Он позволяет значительно ускорить разработку, благодаря наличию большого числа встроенных возможностей, с которыми сталкиваются разработчики. Хранение данных обеспечивается с помощью СУБД PostgreSQL, а для взаимодействия кода с базой данных используется ORM Diesel.

Для разработки клиентской части были выбраны фреймворки Vue.js и Nuxt. Vue.js – это JavaScript фреймворк, позволяющий быстро и удобно создавать производительные, расширяемые и легко поддерживаемые приложения. Nuxt является фреймворком-надстройкой над Vue.js. Он позволяется единообразно структурировать проект на Vue.js, берет на себя некоторые рутинные операции, такие как создание маршрутов, а также обеспечивает рендеринг на стороне сервера.

Для быстрой разработки пользовательского интерфейса были выбраны библиотеки Vuetify и Cytoscape.js. Первая содержит большое количество стилизованных компонентов, поэтому ее использование обеспечивает единообразие интерфейса на всех страницах приложения. Вторая предназначена для построения графов и избавляет от необходимости написания большого количества кода для обеспечения взаимодействия пользователя с графом когнитивной модели.

* 1. Проектирование базы данных программных средств

Разделим схему базы данных на 2 части для удобной визуализации. Часть схемы базы данных, связанная с построением нечеткой когнитивной модели, представлена на рисунке 3.2.

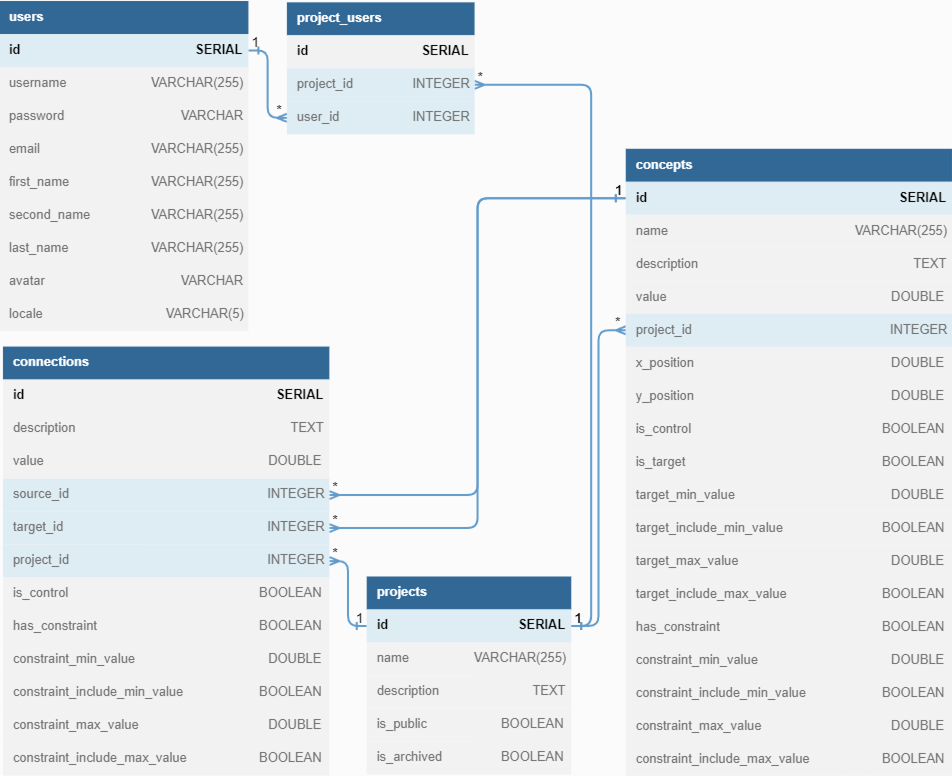


Рисунок 3.2 – Часть схемы базы данных, связанная с построением нечеткой когнитивной модели

Описание отношений и их атрибутов представлено в таблице 3.8.

Таблица 3.8 – Описание отношений и их атрибутов для блока, связанного с построением нечеткой когнитивной модели

|  |  |
| --- | --- |
| **Атрибут** | **Описание** |
| **users (Пользователи)** | |
| id | Идентификатор пользователя |
| username | Логин пользователя |

Таблица 3.8 – Описание отношений и их атрибутов для блока, связанного с построением нечеткой когнитивной модели (продолжение)

|  |  |
| --- | --- |
| password | Пароль пользователя |
| email | Email пользователя |
| first\_name | Имя пользователя |
| second\_name | Отчество пользователя |
| last\_name | Фамилия пользователя |
| avatar | Путь к аватару пользователя |
| locale | Предпочтительный язык пользователя |
| **projects (Проекты)** | |
| id | Идентификатор проекта |
| name | Название проекта |
| description | Описание проекта |
| is\_public | Является ли проект открытым |
| is\_archived | Является ли проект архивированным |
| **project\_users (Пользователи проекта)** | |
| id | Идентификатор пользователя проекта |
| project\_id | Идентификатор проекта |
| user\_id | Идентификатор пользователя |
| **concepts (Концепты)** | |
| id | Идентификатор концепта |
| name | Название концепта |
| description | Описание концепта |
| value | Значение концепта |
| project\_id | Идентификатор проекта |
| x\_position | Позиция концепта по оси x |
| y\_position | Позиция концепта по оси y |
| is\_control | Является ли концепт управляющим |
| is\_target | Является ли концепт целевым |
| target\_min\_value | Минимальное значение целевого диапазона |
| target\_include\_min\_value | Входит ли минимальное значение в целевой диапазон |
| target\_max\_value | Максимальное значение целевого диапазона |
| target\_include\_max\_value | Входит ли максимальное значение в целевой диапазон |
| has\_constraint | Наложено ли на концепт ограничение |
| constraint\_min\_value | Минимальное значение ограничения |
| constraint\_include\_min\_value | Входит ли минимальное значение в ограничение |
| constraint\_max\_value | Максимальное значение ограничения |
| constraint\_include\_max\_value | Входит ли максимальное значение в ограничение |
| **concepts (Связи)** |  |
| id | Идентификатор связи |
| description | Описание связи |
| value | Сила связи |
| source\_id | Идентификатор выходного концепта |
| target\_id | Идентификатор входного концепта |
| project\_id | Идентификатор проекта |
| is\_control | Является ли связь управляющей |
| has\_constraint | Наложено ли на связь ограничение |
| constraint\_min\_value | Минимальное значение ограничения |
| constraint\_include\_min\_value | Входит ли минимальное значение в ограничение |

Таблица 3.8 – Описание отношений и их атрибутов для блока, связанного с построением нечеткой когнитивной модели (продолжение)

|  |  |
| --- | --- |
| constraint\_max\_value | Максимальное значение ограничения |
| constraint\_include\_max\_value | Входит ли максимальное значение в ограничение |

Часть схемы базы данных, связанная со структурно-параметрической настройкой представлена на рисунке 3.3.



Рисунок 3.3 – Часть схемы базы данных, связанная со структурно-параметрической настройкой

Описание отношений и их атрибутов представлено в таблице 3.9.

Таблица 3.9 – Описание отношений и их атрибутов для блока, связанного со структурно-параметрической настройкой

|  |  |
| --- | --- |
| **Атрибут** | **Описание** |
| **model\_copies (Копии модели предназначенные для сохранения модели перед запуском структурно-параметрической настройки)** | |
| id | Идентификатор копии модели |
| project\_id | Идентификатор проекта |
| model | Копия модели |
| **adjustment\_runs (Запуски структурно-параметрической настройки)** | |
| id | Идентификатор запуска настройки |
| project\_id | Идентификатор проекта |
| model\_copy\_id | Идентификатор копии модели |
| name | Название запуска настройки |
| description | Описание запуска настройки |
| min\_model\_time | Минимальное модельное время, в котором осуществляется поиск решения |
| max\_model\_time | Максимальное модельное время, в котором осуществляется поиск решения |
| dynamic\_model\_type | Динамическая модель запуска настройки |
| generation\_save\_interval | Интервал сохранения поколений запуска настройки |
| max\_generations | Максимальное число поколений запуска настройки |
| max\_without\_improvements | Максимальное число поколений без улучшений запуска настройки |
| error | Ошибка запуска настройки |
| result\_individual\_id | Результирующая особь запуска настройки |
| **adjustment\_generations (Поколения структурно-параметрической настройки)** | |
| id | Идентификатор поколения |
| adjustment\_run\_id | Идентификатор запуска настройки |
| number | Номер поколения |
| error | Средняя ошибка поколения |
| **adjustment\_individuals (Особи структурно-параметрической настройки)** | |
| id | Идентификатор особи |
| adjustment\_generation\_id | Идентификатор поколения |
| number | Номер особи |
| time | Модельное время, в котором получено решение |
| error | Ошибка особи |
| **adjustment\_concept\_value (Значение концепта особи)** | |
| id | Идентификатор значения концепта особи |
| adjustment\_individual\_id | Идентификатор особи |
| concept\_id | Идентификатор концепта |
| value | Значение концепта |
| **adjustment\_connection\_value (Сила связи особи)** | |
| id | Идентификатор силы связи особи |
| adjustment\_individual\_id | Идентификатор особи |
| connection\_id | Идентификатор связи |
| value | Сила связи |

* 1. Разработка API серверной части программных средств

Для серверной части программных средств было разработано REST API. Описание основных маршрутов представлено в таблице 3.10.

Таблица 3.10 – Описание основных маршрутов REST API

|  |  |  |
| --- | --- | --- |
| **Маршрут** | **Метод** | **Описание** |
| /user | POST | Регистрация нового пользователя |
| /users | GET | Получение списка пользователей |
| /me | GET | Получение текущего пользователя |
| /me | PUT | Изменение текущего пользователя |
| /sign\_in | POST | Вход в аккаунт |
| /sign\_out | PATCH | Выход из аккаунта |
| /storage/user\_avatars/{path} | GET | Получение аватара пользователя |
| /project | POST | Создание нового проекта |
| /projects/{project\_id} | GET | Получение проекта |
| /projects/{project\_id} | PUT | Изменение проекта |
| /projects/{project\_id}/users | GET | Получение пользователей проекта |
| /projects/{project\_id}/users/  {user\_id}/invite |  | Добавление пользователя в проект |
| /projects/{project\_id}/model | GET | Получение модели проекта (списка концептов и связей) |
| /models/{model\_copy\_id} | GET | Получение копии модели, сохраненной перед запуском структурно-параметрической настройки |
| /projects/{project\_id}/concept | POST | Добавление концепта |
| /concepts/{concept\_id} | PUT | Изменение концепта |
| /concepts/{concept\_id} | DELETE | Удаление концепта |
| /concepts/{concept\_id}/move | PATCH | Перемещение концепта |
| /projects/{project\_id}/connection | POST | Добавление связи |
| /connections/{connection\_id} | DELETE | Удаление связи |
| /connections/{connection\_id} | PATCH | Изменение связи |
| /concepts/{concept\_id}/  chage\_is\_control | PATCH | Изменение является ли концепт управляющим |
| /concepts/{concept\_id}/  change\_target\_concept | PATCH | Изменение целевого концепта |
| /connection/{connection\_id}/  change\_is\_control | PATCH | Изменение является ли связь управляющей |
| /concepts/{concept\_id}/  change\_concept\_constraint | PATCH | Изменение ограничения, наложенного на управляющий концепт |
| /connections/{connection\_id}/  change\_connection\_constraint | PATCH | Изменение ограничения, наложенного на управляющую связь |
| /concepts/{concept\_id}/  change\_dynamic\_model\_type | PATCH | Изменение динамической модели концепта |
| /projects/{project\_id}/adjust | POST | Запуск структурно-параметрической настройки |
| /adjustment\_runs/  {adjustment\_run\_id} | GET | Получение запуска структурно-параметрической настройки |

Таблица 3.10 – Описание основных маршрутов REST API (продолжение)

|  |  |  |
| --- | --- | --- |
| **Маршрут** | **Метод** | **Описание** |
| /projects/{project\_id}/  adjustment\_runs | GET | Получение запусков структурно-параметрической настройки проекта |
| /adjustment\_generations/  {adjustment\_generation\_id} | GET | Получение поколения структурно-параметрической настройки |
| /adjustment\_runs/  {adjustment\_run\_id}/  adjustment\_generations | GET | Получение поколений структурно-параметрической настройки |
| /adjustment\_individuals/  {adjustment\_individual\_id} | GET | Получение особи поколения структурно-параметрической настройки |
| /adjustment\_generations/  {adjustment\_generation\_id}/  adjustment\_individuals | GET | Получение особей поколения структурно-параметрической настройки |

Для получения изменений нечеткой когнитивной модели, а также для получения данных о ходе структурно-параметрической настройки было разработано WebSocket API, описание которого представлено в таблице 3.11.

Таблица 3.11 – Описание маршрутов WebSocket API

|  |  |
| --- | --- |
| **Маршрут** | **Описание** |
| /model/{project\_id} | Получение изменений нечеткой когнитивной модели |
| /adjustment\_run/{project\_id} | Получения данных о ходе структурно-параметрической настройки |

* 1. Разработка интерфейса программных средств

Для клиентской части программных средств было разработано несколько страниц, предназначенных для редактирования проектов нечетких когнитивных моделей, запуска алгоритма структурно-параметрической настройки, а также просмотра результатов структурно-параметрической настройки. Описание разработанных страниц представлено в таблице 3.12.

Таблица 3.12 – Описание разработанных страниц

|  |  |
| --- | --- |
| **Страница** | **Описание** |
| auth/sign\_in | Страница входа в аккаунт |
| auth/sign\_up | Страница регистрации |
| projects | Страница со списком проектов нечетких когнитивных моделей |
| project/{id} | Страница редактирования проекта нечеткой когнитивной модели с идентификатором id |

Таблица 3.12 – Описание разработанных страниц (продолжение)

|  |  |
| --- | --- |
| **Страница** | **Описание** |
| adjustment/{project\_id} | Страница со списком запусков структурно-параметрической настройки проекта с идентификатором project\_id и возможностью запустить новую структурно-параметрическую настройку |
| adjustment/{project\_id}/  {adjustment\_run\_id} | Страница со списком поколений запуска структурно-параметрической настройки с идентификатором adjustment\_run\_id |
| adjustment/{project\_id}/  {adjustment\_run\_id}/  {adjustment\_generation\_id} | Страница со списком особей поколения структурно-паретической настройки с идентификатором adjustment\_generation\_id |
| adjustment/{project\_id}/  {adjustment\_run\_id}/  {adjustment\_generation\_id}/  {individual\_id} | Страница с особью структурно-параметрической настройки с идентификатором individual\_id и возможностью динамического моделирования по шагам |

Диаграмма состояний диалога с основными действиями для запуска структурно-параметрической настройки представлен на рисунке 3.4.



Рисунок 3.4 – Диаграмма состояний диалога с основными действиями для запуска структурно-параметрической настройки

* 1. Разработка модульной структуры программных средств

Модульная структура основных модулей серверной части программных средств представлена на рисунке 3.5.



Рисунок 3.5 – Модульная структура основных модулей серверной части программных средств

Описание основных модулей серверной части представлено в таблице 3.13.

Таблица 3.13 – Описание основных модулей серверной части

|  |  |  |
| --- | --- | --- |
| **Модуль** | **Rust файл** | **Описание** |
| Модуль изменения пользователей | api/src/services/  users-services.rs | Модуль содержит сервисы для изменения пользователей, регистрации и авторизации |
| Модуль изменения проектов | api/src/services/  project\_services.rs | Модуль содержит сервисы для создания, изменения и удаления проектов |
| Модуль изменения пользователей проекта | api/src/services/  project\_user\_services.rs | Модуль содержит сервисы для управления пользователями проекта |
| Модуль изменения модели | api/src/services/  model\_services.rs | Модуль содержит сервисы для изменения модели (добавление, изменение, удаление концептов и связей и др.) |
| Модуль структурно-параметрической настройки | api/src/services/  adjustment\_services.rs | Модуль содержит сервисы для выполнения структурно-параметрической настройки и получения результатов |
| Модуль базы данных | api/src/db.rs | Модуль содержит вспомогательные функции для взаимодействия с базой данных |

Таблица 3.13 – Описание основных модулей серверной части (продолжение)

|  |  |  |
| --- | --- | --- |
| **Модуль** | **Rust файл** | **Описание** |
| Модуль для работы с WebSocket | api/src/web\_socket.rs | Модуль содержит вспомогательные функции для подключения через WebSocket |
| Модуль сервисов | api/src/services.rs | Модуль экспортирует сервисы |
| Модуль маршрутов, связанных с пользователями | api/src/routes/  user\_routes.rs | Модуль содержит маршруты для изменения пользователей, регистрации и авторизации |
| Модуль маршрутов, связанных с проектами | api/src/routes/  project\_routes.rs | Модуль содержит маршруты для создания, изменения и удаления проектов, а также для управления пользователями проекта |
| Модуль маршрутов, связанных с моделью | api/src/routes/  model\_routes.rs | Модуль содержит маршруты для изменения модели |
| Модуль маршрутов, связанных со структурно-параметрической настройкой | api/src/routes/  adjustment\_routes.rs | Модуль содержит маршруты для выполнения структурно-параметрической настройки и получения результатов |
| Модуль маршрутов | api/src/routes.rs | Модуль экспортирует маршруты |

Модульная структура основных модулей клиентской части программных средств представлена на рисунке 3.6.



Рисунок 3.6 – Модульная структура основных модулей клиентской части программных средств

Описание основных модулей клиентской части представлено в таблице 3.14.

Таблица 3.14 – Описание основных модулей клиентской части

|  |  |  |
| --- | --- | --- |
| **Модуль** | **Vue/TypeScript файл** | **Описание** |
| Модуль для взаимодействия с API | web/composables/  api/index.ts | Модуль содержит функции для взаимодействия с API |
| Модуль регистрации | web/pages/auth/  sign\_up.vue | Модуль содержит страницу регистрации |
| Модуль авторизации | web/pages/auth/  sign\_in.vue | Модуль содержит страницу авторизации |
| Модуль формы добавления проектов | web/components/  project/AddForm.vue | Модуль содержит форму для добавления проекта |
| Модуль проектов | web/pages/projects/  index.vue | Модуль содержит страницу со списком проектов |
| Модуль редактора | web/components/  model/Editor.vue | Модуль содержит редактор нечеткой когнитивной модели |
| Модуль действий с моделью | web/composables/  model-actions.ts | Модуль содержит функции для выполнения действий с моделью (добавление, изменение, удаление концептов и связей и др.) |
| Модуль модели | web/pages/projects/  [id].vue | Модуль содержит страницу для просмотра нечеткой когнитивной модели |
| Модуль формы запуска структурно-параметрической настройки | web/components/  adjustment/  AdjustForm.vue | Модуль содержит форму для запуска структурно-параметрической настройки |
| Модуль запусков структурно-параметрической настройки | web/pages/adjsutment/  [project\_id]/index.vue | Модуль содержит страницу со списком запусков структурно-параметрической настройки |
| Модуль поколений структурно-параметрической настройки | web/pages/adjustment/  [project\_id]/  [adjustment\_run\_id]/  index.vue | Модуль содержит страницу со списком поколений запуска структурно-параметрической настройки |
| Модуль особей структурно-параметрической настройки | web/pages/adjustment/  [project\_id]/  [adjustment\_run\_id]/  [generation\_id]/  index.vue | Модуль содержит список особей поколения структурно-параметрической настройки |
| Модуль просмотра особи | web/components/  adjustment/  Editor.vue | Модуль содержит редактор нечеткой когнитивной модели для просмотра особи структурно-параметрической настройки |
| Модуль особи структурно-параметрической настройки | web/pages/adjustment/  [project\_id]/  [adjustment\_run\_id]/  [generation\_id]/  [individual\_id].vue | Модуль содержит страницу для просмотра особи структурно-параметрической настройки |

* 1. Развертывание программных средств

Развертывание программных средств осуществляется на Linux машине с использованием сервера Nginx. Диаграмма развертывания представлена на рисунке 3.7.

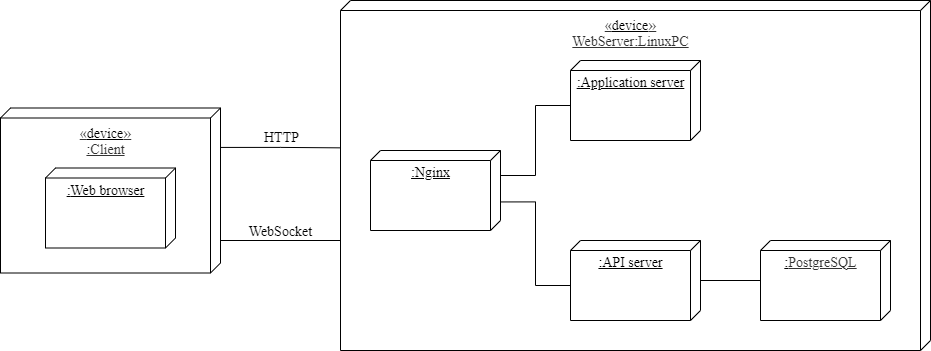


Рисунок 3.7 – Диаграмма развертывания

* 1. Выводы по главе

В данной главе были описаны разработанные программные средства структурно-параметрической настройки нечетких когнитивных моделей Коско и Силова в рамках требований к их функционалу. В частности, были представлены варианты использования программных средств и выбранные веб-технологии для их разработки. Затем была описана схема базы данных, API серверной и интерфейс клиентской частей, а также их модульная структура. В конце была предложена диаграмма развертывания программных средств.

1. Оценка качества и оперативности разработанного способа структурно-параметрической настройки нечеткиХ когнитивных моделей на основе генетических алгоритмов
   1. Оценка качества разработанного способа структурно-параметрической настройки нечетких когнитивных моделей

Предложенный способ структурно-параметрической настройки нечетких когнитивных моделей в отличие от альтернативных позволяет осуществлять точечную настройку. Он дает возможность выбирать концепты и связи, которые необходимо настроить, отражая тем самым тот факт, что в реальных системах не все факторы поддаются влиянию. Кроме того, способ позволяет накладывать ограничения на настраиваемые факторы по той причине, что в реальности они не всегда могут изменяться на всей области значений. Такой подход позволяет осуществлять настройку слабоструктурированной системы исходя из ее понимания исследователем и целей управления, что позволяет повысить качество настройки, а следовательно, и качество управленческих решений.

Стоит заметить, что некоторые альтернативные способы параметрической настройки можно рассматривать как частные случаи разработанного способа. Так, например, способ может дать те же результаты, что и настройка весов нечеткой когнитивной модели при помощи алгоритма Хэбба. Для этого необходимо сделать все связи модели управляющими без ограничений, а все концепты – целевыми. Получение тех же результатов, что и при помощи теоретических методов решения обратной задачи также возможно. Для этого необходимо осуществить настройку с теми же управляющими и целевыми концептами, что и в этих методах. Исходя из этого, можно сделать вывод о том, что разработанный способ во многих ситуациях может заменить существующие, предоставляя при этом большие возможности, что также свидетельствует о его качестве.

Также предложенный способ предоставляет механизм для осуществления структурной настройки, при котором в нечеткую когнитивную модель добавляется концепт-кандидат или связь кандидат. Это также повышает качество настройки, поскольку существующие способы не дают возможности изменять структуру модели.

* 1. Оценка оперативности разработанного способа структурно-параметрической настройки нечетких когнитивных моделей

Для оценки оперативности разработанного способа рассмотрим функциональный блок нечеткой когнитивной модели, представленный на рисунке 4.1.

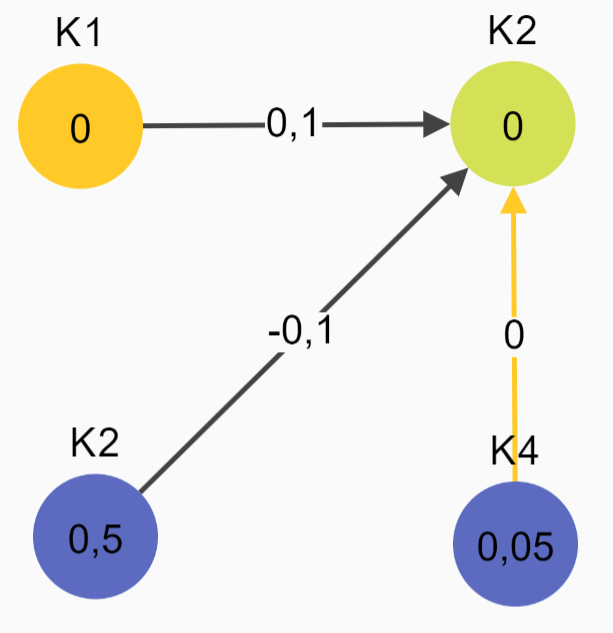


Рисунок 4.1 – Функциональный блок нечеткой когнитивной модели

Концепт является управляющим без ограничений, а концепт является целевым с целевым отрезком . Также управляющей является связь между концептами и . Пусть , тогда целевой концепт при использовании динамической модели (1.6) попадает в целевое значение при и .

С помощью нескольких функциональных блоков может быть построена более крупная нечеткая когнитивная модель. Например, когнитивная модель, состоящая из двух функциональных блоков, представлена на рисунке 4.2. В такой модели целевым является концепт, объединяющий функциональные блоки, т.е. , а его динамической моделью является выражение (1.5). Концепты и должны получить значение в результате моделирования, поэтому, чтобы такое же значение получил целевой концепт, силы связей между концептами и , а также и должны быть равны . В общем случае сила связи между целевым концептом функционального блока и целевым концептом модели вычисляется по формуле , где – число функциональных блоков. Для учета задержки время моделирования должно быть увеличено на с до .

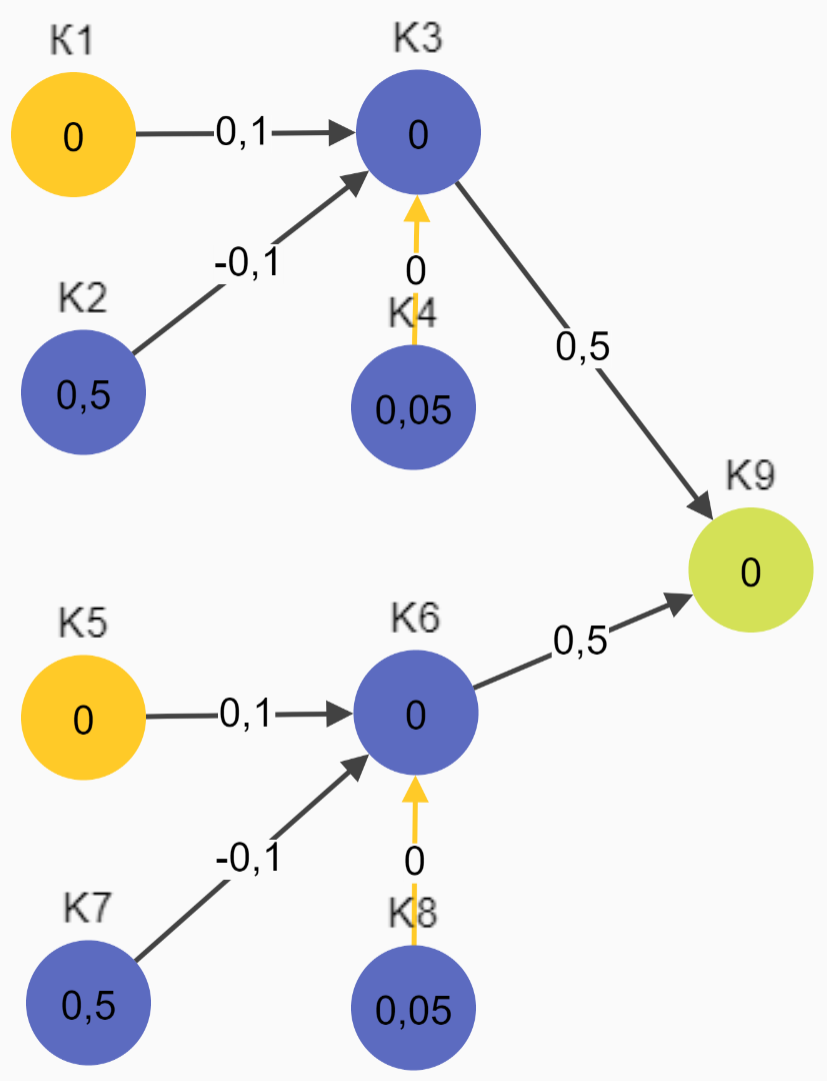


Рисунок 4.2 – Когнитивная модель, состоящая из двух функциональных блоков

Будем осуществлять тестирование производительности разработанного способа с помощью когнитивных моделей, построенных из , , , и функциональных блоков. Максимальное число поколение зададим в , а число особей в поколении будем последовательно устанавливать в , , и . Ошибку зададим в и осуществим по попыток каждого из запусков. В таблице 4.1 представлено среднее время в секундах получения результата для каждого из запусков, а в таблице 4.2 представлены средние номера поколений, в которых был получен результат. Тестирование выполнялось на ядре процессора AMD Ryzen 7 5800H.

Таблица 4.1 – Среднее время в секундах получения результата

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 25 | 50 | 100 | 250 |
| 1 | 00:00 | 00:00 | 00:00 | 00:01 |
| 2 | 00:02 | 00:05 | 00:05 | 00:06 |
| 5 | 00:15 | 00:20 | 00:40 | 00:59 |
| 10 | 01:55 | 02:19 | 03:47 | 05:58 |
| 25 | 14:12 | 24:39 | 40:31 | 59:16 |

Таблица 4.2 – Средние номера поколений, в которых был получен результат

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 25 | 50 | 100 | 250 |
| 1 | 21 | 45 | 51 | 48 |
| 2 | 489 | 329 | 229 | 157 |
| 5 | 2798 | 1256 | 949 | 526 |
| 10 | 4018 | 2510 | 2025 | 1238 |
| 25 | 8769 | 6735 | 5434 | 3436 |

Графики зависимости среднего времени от числа функциональных блоков при разном размере поколения представлены на рисунке 4.3. Из них видно, что лучшим размером поколения является для любого числа функциональных блоков.

Рисунок 4.3 – Графики зависимости среднего времени от числа функциональных блоков при разном размере поколения

Графики зависимости среднего номера поколения от размера поколения при разном числе функциональных блоков представлены на рисунке 4.4. Из них видно, что номер поколения, в котором найдено решение, снижается при увеличении размера поколения.

Рисунок 4.4 – Графики зависимости среднего номера поколения от размера поколения при разном числе функциональных блоков

Таким образом, тестирование показало, что программные средства практически мгновенно находят решение при оптимизации 2 переменных, а при оптимизации 50 переменных находят решение за 15 минут при лучшем размере поколения. Это свидетельствует о том, что программные средства, а значит и разработанный способ обладают высокой оперативностью.

* 1. Решение задачи «Управление ИТ-проектом» с помощью предложенного способа

Рассмотрим нечеткую когнитивную модель, представленную на рисунке 4.5. Это модель была построена с помощью экспертного подхода и предназначена для моделирования управления ИТ-проектом. В модели концепты «Риски», «Выполнение задач» и «Стоимость проекта» используют динамическую модель (1.5), а остальные концепты используют модель (1.6). Концепты «Выполнение проекта» и «Стоимость проекта» являются целевыми. Проект необходимо завершить, поэтому целевой диапазон концепта «Выполнение проекта» равен . «Стоимость проекта» не должна превысить значение , поэтому ее целевой диапазон равен . Командное взаимодействие можно несколько улучшить путем менеджерских практик, поэтому концепт «Командное взаимодействие» является управляющим с ограничением . Планирование также может быть усилено, но уже более значительно, поэтому ограничение этого управляющего концепта равно . Для завершения проекта можно нанять новых сотрудников или при необходимости уволить старых, поэтому «Число сотрудников» является управляющим концептом с ограничением . Руководство может регулировать периодическое повышение зарплат, поэтому концепт «Повышение зарплат» также является управляющим с ограничением . Связь между концептами «Командное взаимодействие» и «Выполнение проекта» является управляющей с ограничением , поскольку менеджмент может внедрить более эффективные с точки зрения продуктивности формы командного взаимодействия.

Пусть необходимо завершить проект за 20-25 месяцев, тогда осуществим настройку с параметрами и . Результат настройки представлен на рисунке 4.6. Он свидетельствует о том, что проект может быть выполнен за 25 месяцев при условии значительного усиления планирования и незначительного повышения командного взаимодействия, числа сотрудников, зарплат, а также влияния командного взаимодействия на выполнение проекта.

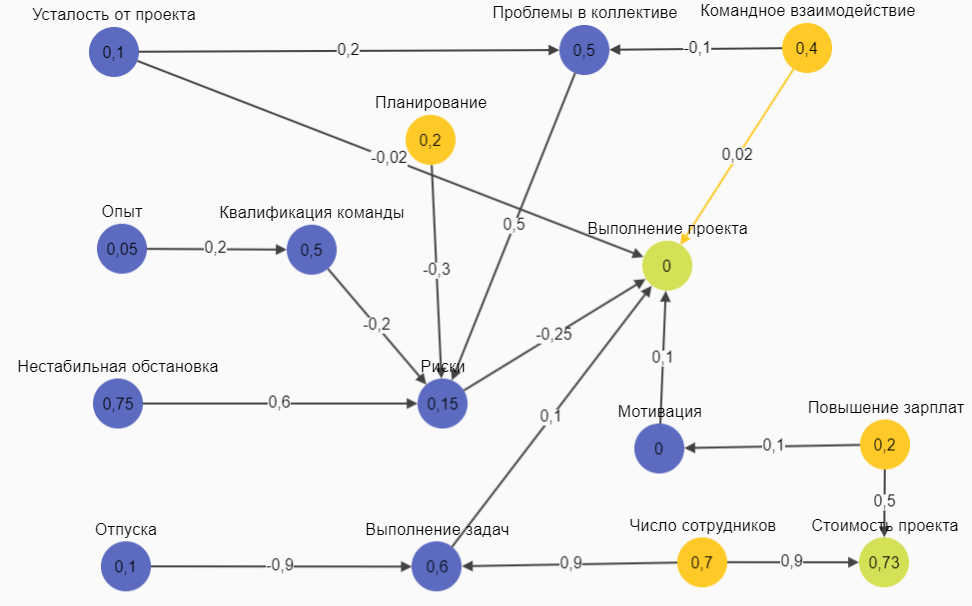


Рисунок 4.5 – Когнитивная модель «Управление ИТ-проектом»

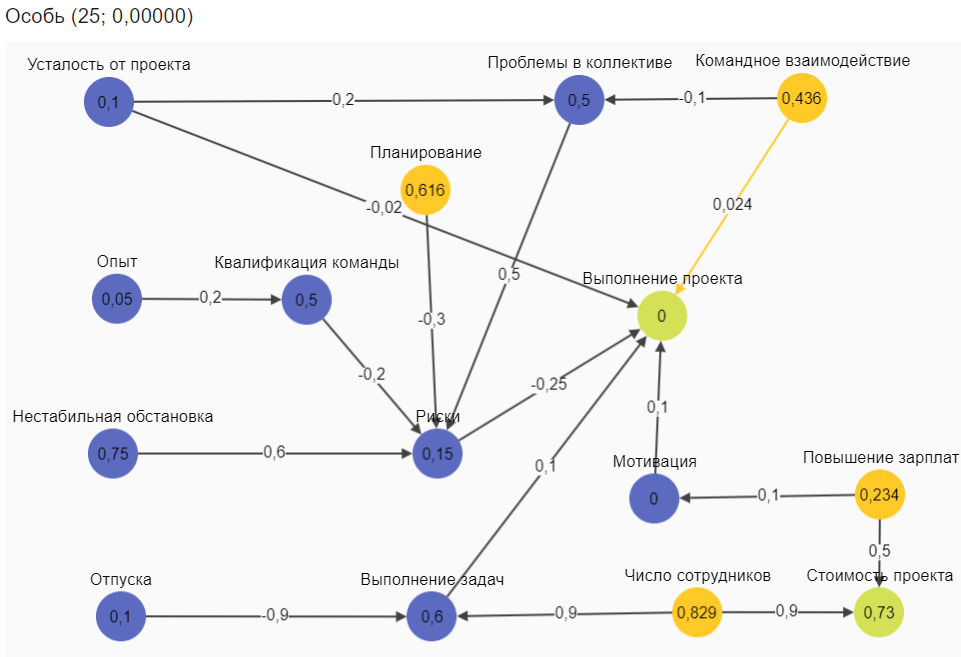


Рисунок 4.6 – Результат настройки нечеткой когнитивной модели «Управление ИТ-проектом»

Конечное состояние нечеткой когнитивной модели после 25 итераций представлено на рисунке 4.7.

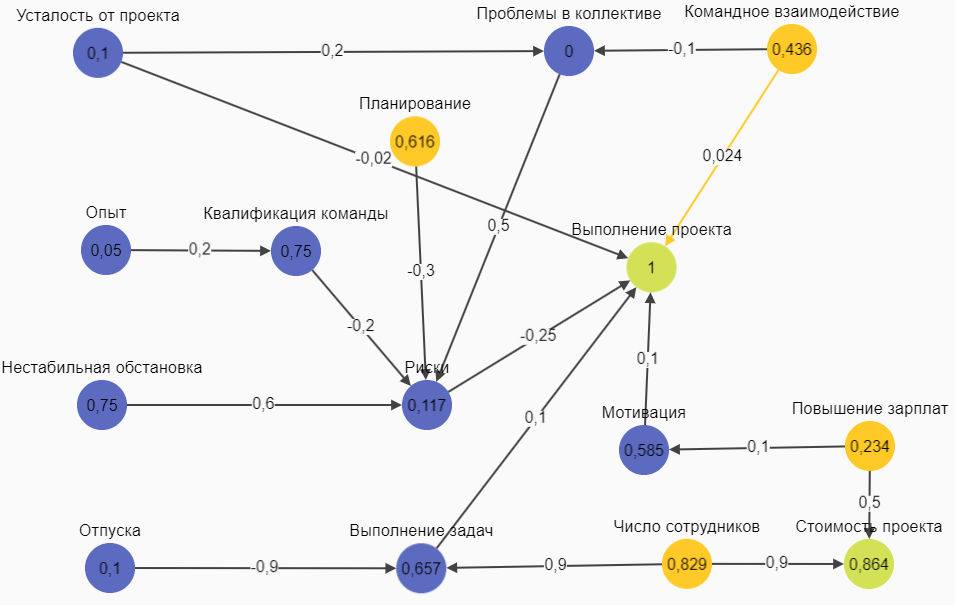


Рисунок 4.6 – Конечное состояние нечеткой когнитивной модели «Управление ИТ-проектом» после 25 итераций

* 1. Выводы по главе

В данной главе было оценено качество и оперативность предложенного способа структурно-параметрической настройки нечетких когнитивных моделей. Оценка качества осуществлялась из общих соображений в сравнении с другими способами. Она показала, что настройка с использованием разработанного способа обладает более высоким качеством, чем с использованием альтернативных способов. Оперативность оценивалась с помощью эксперимента, который показал, что способ обладает высокой оперативностью. Также в этой главе с помощью предложенного способа была решена задача «Управление ИТ-проектом».

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы были достигнута цель и решена научная задача, заключающаяся в исследовании и разработке способа и программных средств структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов.

При выполнении выпускной квалификационной работы получены следующие основные результаты:

1. Выполнен анализ разновидностей нечетких когнитивных моделей, таких как нечеткие когнитивные модели Коско, нечеткие когнитивные модели Силова, нечеткие продукционные когнитивные модели и их обобщенная разновидность, нечеткие реляционные когнитивные модели, а также «совместимые» нечеткие когнитивные модели. Проведен анализ методов и подходов к их структурно-параметрической настройке, который показал, что существующие методы направлены на отдельные разновидности нечетких когнитивных моделей и позволяют осуществлять только параметрическую настройку.
2. Создан способ структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов, который может быть применим к любой из разновидностей этих моделей.
3. Разработаны алгоритмы и программные средства структурно-параметрической настройки нечетких когнитивных моделей Коско и Силова на основе генетических алгоритмов.
4. Выполнена оценка качества и оперативности структурно-параметрической настройки нечетких когнитивных моделей на основе предлагаемого способа и программных средств, которая показала высокую степень качества и оперативности.

Дальнейшее развитие исследования может заключаться в рассмотрении различных методов ускорения сходимости генетического алгоритма, а также в изучении возможности использования непрерывной функции принадлежности или нечетких продукций для нечетких реляционных когнитивных моделей.

Список литературы

1. Авдеева З. К., Коврига С. В., Макаренко Д. И. Когнитивное моделирование для решения задач управления слабоструктурированными системами (ситуациями) // УБС. 2006. №16.
2. Прокофьев О. В. Савочкин А. Е. Моделирование системы обнаружения предаварийной ситуации на основе нечеткой когнитивной карты // НиКСС. 2018. №2 (22).
3. Дулесов А. С., Пантелеев В. И., Баркова Д. В. Когнитивное моделирование как инструмент управления запасами топлива на станции // Журнал СФУ. Техника и технологии. 2013. №1.
4. Максимов В. И. Структурно-целевой анализ развитии социально экономических ситуаций // Проблемы управления. 2005. №3.
5. Молочкова В. М. Когнитивная карта инвестиционной деятельности муниципального образования // Известия ЮФУ. Технические науки. 2010. №4.
6. Робертс Ф. С. Дискретные модели с приложениями к социальным, биологическим и экологическим задачам. М.: Наука, 1986
7. Koulouriotis D.E., Diakoulakis I.E., Emiris D.M., Antonidakis E.N., Kaliakatsos I.A. Efficiently modeling and controlling complex dynamic systems using evolutionary fuzzy cognitive maps (Invited Paper) // Internat. J. Comput. Cognition 1 (2) (2003) 41–65
8. Stach W., Kurgan L., Pedrycz W., Reformat M. Genetic learning of fuzzy cognitive maps // Fuzzy Sets and Systems 153 (2005) P. 371 – 401
9. M.S. Khan, A. Chong, Fuzzy cognitive map analysis with genetic algorithm // Proc. 1st Indian Internat. Conf. on Artificial Intelligence (IICAI-03), 2003
10. Аверкин А. Н., Паринов А. А. Генетический алгоритм обучения нечетких когнитивных карт // Научные труды Вольного экономического общества России. 2010. №.
11. Горбачев Ю. В. Исследование и разработка способов и алгоритмов когнитивного моделирования для систем поддержки принятия решений // Магистерская диссертация. 1997.
12. Орлов А.И. Способ структурно-параметрической настройки нечетких когнитивных карт // Научный аспект. 2023. № 6.
13. Axelrod R. Structure of Decision: the cognitive maps of political elites. Prinston Univ, Press, N. Y.: 1976.
14. Kosko B. Fuzzy cognitive maps // Int. Journal of Man-Machine Studies. 1986. V. 24. P. 65–75.
15. Zhang W. R., Chen S. S. A logical architecture for cognitive maps // In Proc. of the 2nd IEEE International Conference on Neural Networks (ICNN-88). 1988. V. 1. P. 231–238.
16. Zhang W. R., Chen S., King R. S. A cognitive map based approach to the coordination of distributed cooperative agents // IEEE Transactions on Systems, Man and Cybernetics1992. . V. 22. P. 103–113
17. Робертс Ф. С. Дискретные модели с приложениями к социальным, биологическим и экологическим задачам. М.: Наука, 1986.
18. Борисов В. В., Круглов В. В., Федулов А. С.. Нечёткие модели и сети. – М.: Горячая линия – Телеком, 2012. – 284 с.
19. Stylios C. D., Groumpos P. P. Application of fuzzy cognitive maps in large manufacturing systems // In Proc. of the IFAC LSS’98, Rio, Patras, Greece. V. 1. 1998. P. 531–536.
20. Stylios C. D., Groumpos P. P. Fuzzy cognitive maps in modeling supervisory control systems // Journal of Intelligent & Fuzzy Systems. 2000. V. 8. №. 2. P. 83–98.
21. Силов В. Б. Принятие стратегических решений в нечеткой обстановке. М.: ИНПРО–РЕС, 1995.
22. Carvalho J. P., Tomé J. A. Rule-based fuzzy cognitive maps and fuzzy cognitive maps – a comparative study // In Proc. of the 18th International Conference of the North American Fuzzy Information Processing Society. NAFIPS’99, New York, 1999. P. 115 – 119.
23. Carvalho, J. P. and Tomé J. A. B. Fuzzy Mechanisms for causal Relations, Proceedings of the IFSA99, 8th International Fuzzy systems Association World Congress, Taiwan 1999.
24. Борисов В. В., Федулов А. С., Мнев В. И. Моделирование сложных нечетко-целевых систем на основе обобщенных нечетких когнитивных карт // Информ. бюл. Акад. воен. наук. Смоленск: Изд-во ВУ ВПВО ВС РФ, 2001. С. 22–25.
25. Заде Л. А. Понятие лингвистической переменной и его применение к принятию приближенных решений. М.: Мир, 1976.
26. Борисов В. В., Федулов А. С. Нечеткие оценочные модели сложных систем с учетом согласования неравнозначных целей // Нейрокомпьютеры: разработка, применение. 2003. № 5. С. 3–12
27. Федулов А. С. Нечеткие реляционные когнитивные карты // Изв. РАН. Теория и системы управления. 2005. № 1. С. 120–133.
28. Zadeh L. A. Fuzzy sets //Information and Control. 1965. Vol. 8, № 3. Pp. 338-353
29. Борисов В. В., Федулов А. С. «Совместимые» нечеткие когнитивные модели // Сборник трудов Пятнадцатой национальной конференции по искусственному интеллекту с международным участием КИИ-2016. Смоленск, 3–7 октября 2016 г. В 3-х томах. Т.2. С. 244–249.
30. Борисов В.В., Федулов Я.А. Нечеткая модель оценки сложных организационно-технических систем // Естественные и технические науки. 2014. №5. С. 134–145
31. Борисов В.В., Федулов А.С., Федулов Я.А. «Совместимые» нечеткие когнитивные модели: прямые и обратные задачи // Нечеткие системы и мягкие вычисления. 2016. №2 (11). С. 103–114
32. Carlsson C., Fuller R. Adaptive fuzzy cognitive maps for hyperknowledge representation in strategy formation process // In Proc. of the International Panel Conference on Soft and Intelligent Computing, Technical University of Budapest, 1996. P. 43–50.
33. Kosko B. Differential hebbian learning // AIP Conference Proceedings. V. 151. 1986. P. 265–270.
34. Huerga A. A balanced differential learning algorithm in fuzzy cognitive maps // In Proc. of the Sixteenth International Workshop on Qualitative Reasoning, QR’2002, Spain, 2002. P. 210–214.
35. Вирсански Э. Генетические алгоритмы на Python. Применение генетических алгоритмов к решению задач глубокого обучения и искусственного интеллекта.
36. The Rust Programming Language [Электронный ресурс] URL: https://doc.rust-lang.org/book/ (дата обращения 17.04.2023)

Приложение А

Задание на выпускную квалификационную работу

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Филиал федерального государственного бюджетного образовательного учреждения высшего образования

«Национальный исследовательский университет «МЭИ» в г. Смоленске

Кафедра вычислительной техники

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

НА СОИСКАНИЕ КВАЛИФИКАЦИИ

**Магистр**

Тема: СПОСОБ И ПРОГРАММНЫЕ СРЕДСТВА СТРУКТУРНО-ПАРАМЕТРИЧЕСКОЙ НАСТРОЙКИ НЕЧЕТКИХ КОГНИТИВНЫХ МОДЕЛЕЙ НА ОСНОВЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Направление подготовки

**09.04.01 – Информатика и вычислительная техника**

Профиль подготовки

**Информационное и программное обеспечение автоматизированных систем**

Студент группы ВМ-21 (маг) А. И. Орлов

Руководитель

д.т.н., профессор В. В. Борисов

УТВЕРЖДАЮ

Заведующий кафедрой

д.т.н., профессор А.С. Федулов

«\_\_\_\_» \_\_\_\_\_\_\_\_\_2023 г.

Место выполнения выпускной квалификационной работы: филиал НИУ «МЭИ» в г. Смоленске, кафедра вычислительной техники

Смоленск – 2023

1. Обоснование выбора темы выпускной квалификационной работы

Нечеткие когнитивные модели являются отличным иструментом для анализа сложных и слабоструктурированных систем и ситуаций. Они находят широкое применение в различных областях, таких как прогнозирование, сценарное моделирование, принятие решений, диагностика, классификация и управление. Однако, несмотря на их эффективность, процесс настройки таких моделей может быть достаточно трудоемким и требовать значительных затрат времени и ресурсов.

Генетические алгоритмы являются хорошим способом оптимизации, применяемым в различных областях. Использование генетических алгоритмов для структурно-параметрической настройки нечетких когнитивных моделей может позволить значительно ускорить процесс настройки и повысить качество получаемых результатов. Поэтому разработка способа применения генетических алгоритмов для настройки нечетких когнитивных моделей является важной и актуальной задачей.

1. Основание для разработки

Государственный общеобразовательный стандарт высшего образования. Направление 09.04.01 – «Информатика и вычислительная техника».

1. Назначение и цель разработки

Целью данной работы является, во-первых, создание способа структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов, который позволит настроить когнитивную модель таким образом, чтобы в результате ее моделирования по времени, целевые концепты максимально приблизились к желаемому состоянию. Во-вторых, разработка программных средств, реализующих разработанный способ сруктурно-параметрической настройки. В-третьих, выполнение оценки качества и оперативности настройки нечетких когнитивных моделей с использованием предлагаемого способа и программных средств.

1. Технические требования

4.1 Требования к функциональным характеристикам

Разрабатываемая система должна реализовывать основные функции, обеспечивая возможность осуществления структурно-параметрической настройки нечетких когнитивных моделей.

Пользовательский интерфейс должен обеспечивать наглядность предоставляемой информации, быть эргономичным и интеллектуально понятным.

* 1. Требования к структуре и функционированию

Разрабатываемая программа должна предоставлять пользователю простой и интеллектуально понятный интерфейс и иметь следующие основные функциональные возможности:

* регистрация и авторизация пользователей;
* создание проектов нечетких когнитивных моделей и добавление пользователей в проект;
* редактирование нечеткой когнитивной модели (добавление/удаление концептов и связей);
* редактирование параметров генетического алгоритма структурно-параметрической настройки нечеткой когнитивной модели;
* запуск генетического алгоритма структурно-параметрической настройки нечеткой когнитивной модели;
* сохранение промежуточных и окончательных результатов структурно-параметрической настройки нечеткой когнитивной модели.

4.3 Требования к надёжности

Все внутренние ошибки системы должны обрабатываться корректным образом, информация об ошибке должна выводиться на экран.

Необходимо предусмотреть контроль правильности вводимой информации.

4.4 Требования к защите информации от несанкционированного доступа

Аккаунты всех пользователей системы должны быть защищены паролем.

4.5 Требования по сохранению информации при авариях

Защита от сбоев в сети питания осуществляется путём использования устройства бесперебойного питания и программ, корректно завершающих работу приложений при отключении питания.

4.6 Требования к видам обеспечения

* + 1. Информационное обеспечение

Диалог осуществляется через пользовательский интерфейс системы в браузере.

* + 1. Лингвистическое обеспечение

Серверная логика системы разрабатывается с использованием языка программирования Rust и фреймворка Rocket. Клиентская часть формируется с помощью языка TypeScript и фреймворка Nuxt.js.

* + 1. Программное обеспечение

Для серверной части: операционная система Ubuntu Server, СУБД PostgreSQL, Rust, Node.js.

Для клиентской части: наличие браузера Mozilla Firefox или браузера на основе Chromium.

* + 1. Техническое обеспечение

Для работы серверной части программного продукта рекомендуется компьютер, оснащенный процессором с тактовой частотой не ниже 3 ГГц, двухъядерный или более; 8 ГБ ОЗУ; минимум 50 ГБ свободного места на жестком диске.

* + 1. Организационное обеспечение

Программа не требует дополнительного организационного обеспечения.

* + 1. Требования к тестированию

Тестирование должно показать полную работоспособность программного продукта, правильное выполнение поставленных задач. Тестирование разработанной системы следует проводить в соответствии с ГОСТ Р ИСО МЭК 12207-99.

1. Требования к программной документации

Программная документация будет оформлена в виде пояснительной записки. Пояснительная записка должна быть выполнена согласно ГОСТ 2.105-95.

6 План работы над выпускной квалификационной работой

| Планируемая работа | Трудоёмкость  % | Срок выполнения | Отметка о выполнении |
| --- | --- | --- | --- |
| Анализа задачи структурно-параметрической настройки нечетких когнитивных моделей | 5 | 20.02.2023 |  |
| Анализа существующих способов и программных средств структурно-параметрической настройки нечетких когнитивных моделей | 10 | 13.03.2023 |  |
| Разработка способа структурно-параметрической настройки нечетких когнитивных моделей на основе генетических алгоритмов | 20 | 03.04.2023 |  |
| Разработка генетического алгоритма структурно-параметрической настройки нечетких когнитивных моделей | 10 | 17.04.2023 |  |
| Создание программных средств, реализующих предложенный способ структурно-параметрической настройки нечетких когнитивных моделей | 30 | 08.05.2023 |  |
| Оценка качества и оперативности структурно-параметрической настройки нечетких когнитивных моделей с использованием предлагаемого способа и программных средств | 10 | 15.05.2023 |  |
| Оформление пояснительной записки | 15 | 12.06.2023 |  |
| Сдача проекта на проверку | – | 12.06.2023 |  |

1. Перечень графического материала

* Техническое задание
* Диаграммы UML
* Модульная структура программных средств
* Результаты тестирования

8 Рекомендуемая литература

1. Борисов В. В., Круглов В. В., Федулов А. С. Нечеткие модели и сети. – 2-е изд., стереотип. – М.:Горячая линия – Телеком, 2012. – 284 с.; ил.
2. Силов В.Б. Принятие стратегических решений в нечеткой обстановке – М.: ИНПРО-РЕС, 1995.
3. Bart Kosko. Fuzzy Thinking: The New Science of Fuzzy Logic / Bart Kosko. – New York : Hyperion, 1993. – 320 p.

**Сведения об авторе выпускной квалификационной работы:**

Домашний адрес: г. Смоленск, ул. Академика Петрова, д. 5, кв. 48.

Телефон: 8-905-698-31-03.

Электронная почта: squakrazv@yandex.ru.

Приложение Б

Текст программы

use async\_trait::async\_trait;

use ordered\_float::OrderedFloat;

use rand::rngs::ThreadRng;

use rand::Rng;

use serde::{Deserialize, Serialize};

use std::collections::HashMap;

use std::sync::Arc;

#[async\_trait]

pub trait SaveResult<T, E> {

    async fn save\_result(&mut self, result\_individual: &Individual) -> Result<T, E>;

    async fn save\_generation(&mut self, generation: &mut Generation, number: i32) -> Result<T, E>;

}

#[derive(Clone, Deserialize)]

#[serde(rename\_all = "snake\_case")]

pub enum DynamicModel {

    KoskoDeltaDelta,

    KoskoDeltaValue,

    KoskoValueDelta,

    KoskoValueValue,

    Silov,

}

#[derive(Deserialize)]

#[serde(rename\_all = "camelCase")]

pub struct StopCondition {

    pub max\_generations: i32,

    pub max\_without\_improvements: i32,

    pub error: f64,

}

#[derive(Deserialize)]

#[serde(rename\_all = "camelCase")]

pub struct AdjustmentInput {

    pub name: String,

    pub description: String,

    pub min\_model\_time: i32,

    pub max\_model\_time: i32,

    pub dynamic\_model: DynamicModel,

    pub generation\_size: i32,

    pub generation\_save\_interval: i32,

    pub stop\_condition: StopCondition,

}

#[derive(Deserialize)]

#[serde(rename\_all = "camelCase")]

pub struct AdjustmentModel {

    pub adjustment\_input: AdjustmentInput,

    pub concepts\_map: HashMap<i32, Arc<Concept>>,

    pub control\_concepts: Vec<Arc<Concept>>,

    pub target\_concepts: Vec<Arc<Concept>>,

    pub regular\_concepts: Vec<Arc<Concept>>,

    pub connections\_map: HashMap<i32, Arc<Connection>>,

    pub control\_connections: Vec<Arc<Connection>>,

    without\_improvements: i32,

    current\_generation: Option<Generation>,

    generation\_number: i32,

    is\_generation\_saved: bool,

}

#[derive(Deserialize)]

#[serde(rename\_all = "camelCase")]

pub struct Concept {

    pub id: i32,

    pub value: f64,

    pub is\_control: bool,

    pub is\_target: bool,

    pub target\_value: Option<TargetValue>,

    pub constraint: Option<Constraint>,

    pub dynamic\_model: Option<DynamicModel>,

}

#[derive(Deserialize)]

#[serde(rename\_all = "camelCase")]

pub struct Connection {

    pub id: i32,

    pub value: f64,

    pub source\_id: i32,

    pub target\_id: i32,

    pub is\_control: bool,

    pub constraint: Option<Constraint>,

}

#[derive(Deserialize)]

#[serde(rename\_all = "camelCase")]

pub struct TargetValue {

    pub min\_value: f64,

    pub include\_min\_value: bool,

    pub max\_value: f64,

    pub include\_max\_value: bool,

}

#[derive(Deserialize)]

#[serde(rename\_all = "camelCase")]

pub struct Constraint {

    pub min\_value: f64,

    pub include\_min\_value: bool,

    pub max\_value: f64,

    pub include\_max\_value: bool,

}

#[derive(Clone, Serialize, Deserialize)]

pub struct Fitness {

    pub time: i32,

    pub error: f64,

}

#[derive(Clone, Serialize, Deserialize)]

#[serde(rename\_all = "camelCase")]

pub struct Individual {

    pub id: Option<i32>,

    pub concepts: HashMap<i32, f64>,

    pub connections: HashMap<i32, f64>,

    pub fitness: Option<Fitness>,

}

#[derive(Serialize, Deserialize)]

#[serde(rename\_all = "camelCase")]

pub struct Generation {

    pub individuals: Vec<Individual>,

    pub error: f64,

}

const ALPHA: f64 = 0.5;

const ERROR\_DIFF: f64 = 0.001;

impl AdjustmentModel {

    pub fn new(

        adjustment\_input: AdjustmentInput,

        concepts\_map: HashMap<i32, Arc<Concept>>,

        control\_concepts: Vec<Arc<Concept>>,

        target\_concepts: Vec<Arc<Concept>>,

        regular\_concepts: Vec<Arc<Concept>>,

        connections\_map: HashMap<i32, Arc<Connection>>,

        control\_connections: Vec<Arc<Connection>>,

    ) -> Self {

        Self {

            adjustment\_input,

            concepts\_map,

            control\_concepts,

            target\_concepts,

            regular\_concepts,

            connections\_map,

            control\_connections,

            without\_improvements: 0,

            current\_generation: None,

            generation\_number: 0,

            is\_generation\_saved: false,

        }

    }

    pub fn start(&mut self) -> () {

        self.without\_improvements = 0;

        self.current\_generation = Some(self.create\_first\_generation());

        self.generation\_number = 0;

        self.is\_generation\_saved = false;

    }

    pub async fn next<S, T, E>(&mut self, save\_result: &mut S) -> Result<bool, E>

    where

        S: SaveResult<T, E>,

    {

        if self.generation\_number >= self.adjustment\_input.stop\_condition.max\_generations

            || self.without\_improvements

                >= self

                    .adjustment\_input

                    .stop\_condition

                    .max\_without\_improvements

        {

            return Ok(false);

        }

        if self.generation\_number % self.adjustment\_input.generation\_save\_interval == 0 {

            save\_result

                .save\_generation(

                    self.current\_generation.as\_mut().unwrap(),

                    self.generation\_number + 1,

                )

                .await?;

            self.is\_generation\_saved = true;

        }

        let best\_individual\_error = self.current\_generation.as\_ref().unwrap().individuals[0]

            .fitness

            .as\_ref()

            .unwrap()

            .error;

        if best\_individual\_error < self.adjustment\_input.stop\_condition.error {

            return Ok(false);

        }

        let next\_generation = self.create\_next\_generation();

        self.generation\_number += 1;

        self.is\_generation\_saved = false;

        if (next\_generation.error - self.current\_generation.as\_ref().unwrap().error).abs()

            < ERROR\_DIFF

        {

            self.without\_improvements += 1;

        } else {

            self.without\_improvements = 0;

        }

        self.current\_generation = Some(next\_generation);

        Ok(

            self.generation\_number < self.adjustment\_input.stop\_condition.max\_generations

                && self.without\_improvements

                    < self

                        .adjustment\_input

                        .stop\_condition

                        .max\_without\_improvements,

        )

    }

    pub async fn finish<S, T, E>(&mut self, save\_result: &mut S) -> Result<Individual, E>

    where

        S: SaveResult<T, E>,

    {

        if !self.is\_generation\_saved {

            save\_result

                .save\_generation(

                    self.current\_generation.as\_mut().unwrap(),

                    self.generation\_number + 1,

                )

                .await?;

        }

        let best\_individual = &self.current\_generation.as\_ref().unwrap().individuals[0];

        save\_result.save\_result(best\_individual).await?;

        return Ok(best\_individual.clone());

    }

    fn get\_individual\_fitness(

        &self,

        concepts: &HashMap<i32, f64>,

        connections: &HashMap<i32, f64>,

    ) -> Fitness {

        let concepts = self.get\_initial\_state(concepts);

        let time\_simulation = TimeSimulation::new(

            self.adjustment\_input.max\_model\_time,

            self.concepts\_map.clone(),

            self.connections\_map.clone(),

            self.target\_concepts.clone(),

            self.adjustment\_input.dynamic\_model.clone(),

            concepts,

            connections.clone(),

        );

        let mut fitness = Fitness {

            error: f64::MAX,

            time: self.adjustment\_input.min\_model\_time,

        };

        for data in time\_simulation {

            if data.time >= self.adjustment\_input.min\_model\_time && data.error < fitness.error {

                fitness.error = data.error;

                fitness.time = data.time;

            }

        }

        fitness

    }

    fn get\_generation\_error(individuals: &[Individual]) -> f64 {

        individuals

            .iter()

            .map(|individual| individual.fitness.as\_ref().unwrap().error)

            .sum::<f64>()

            / individuals.len() as f64

    }

    fn select\_parent\_candidates(&self, rng: &mut ThreadRng, best\_count: i32) -> Vec<&Individual> {

        let generation = self.current\_generation.as\_ref().unwrap();

        let mut parents = Vec::new();

        for \_ in 0..self.adjustment\_input.generation\_size - best\_count {

            let candidate1 =

                &generation.individuals[rng.gen\_range(0..generation.individuals.len())];

            let candidate2 =

                &generation.individuals[rng.gen\_range(0..generation.individuals.len())];

            if candidate1.fitness.as\_ref().unwrap().error

                <= candidate2.fitness.as\_ref().unwrap().error

            {

                parents.push(candidate1);

            } else {

                parents.push(candidate2);

            }

        }

        parents

    }

    fn cross\_individuals(

        &self,

        parent1: &Individual,

        parent2: &Individual,

        rng: &mut ThreadRng,

    ) -> Vec<Individual> {

        if rng.gen::<f64>() < 0.05 {

            return vec![parent1.clone(), parent2.clone()];

        }

        return vec![

            self.create\_child\_individual(parent1, parent2, rng),

            self.create\_child\_individual(parent1, parent2, rng),

        ];

    }

    fn mutate\_individual(&self, mut individual: Individual, rng: &mut ThreadRng) -> Individual {

        let (concept\_probability, connection\_probability) = if rng.gen::<f64>() < 0.5 {

            (0.9, 0.5)

        } else {

            (0.5, 0.9)

        };

        if individual.concepts.len() > 0 && rng.gen::<f64>() < concept\_probability {

            let concept = &self.control\_concepts[rng.gen\_range(0..self.control\_concepts.len())];

            \*individual.concepts.get\_mut(&concept.id).unwrap() = concept.generate\_value(rng);

        }

        if individual.connections.len() > 0 && rng.gen::<f64>() < connection\_probability {

            let connection =

                &self.control\_connections[rng.gen\_range(0..self.control\_connections.len())];

            \*individual.connections.get\_mut(&connection.id).unwrap() =

                connection.generate\_value(rng);

        }

        let fitness = self.get\_individual\_fitness(&individual.concepts, &individual.connections);

        individual.fitness = Some(fitness);

        individual

    }

    fn create\_first\_generation(&self) -> Generation {

        let mut rng = rand::thread\_rng();

        let mut individuals = Vec::new();

        for \_ in 0..self.adjustment\_input.generation\_size {

            individuals.push(Self::create\_random\_individual(self, &mut rng));

        }

        Self::sort\_by\_fitness(&mut individuals);

        let error = Self::get\_generation\_error(&individuals);

        Generation { individuals, error }

    }

    fn create\_next\_generation(&self) -> Generation {

        let mut rng = rand::thread\_rng();

        let mut rng\_clone = rng.clone();

        let best\_count = self.adjustment\_input.generation\_size / 10;

        let mut individuals = self

            .select\_parent\_candidates(&mut rng, best\_count)

            .chunks(2)

            .flat\_map(|chunk| match chunk {

                &[p1, p2] => self.cross\_individuals(p1, p2, &mut rng),

                &[p] => vec![p.clone()],

                \_ => unreachable!(),

            })

            .map(|individual| self.mutate\_individual(individual, &mut rng\_clone))

            .collect::<Vec<\_>>();

        for individual in

            &self.current\_generation.as\_ref().unwrap().individuals[0..best\_count as usize]

        {

            individuals.push(individual.clone());

        }

        Self::sort\_by\_fitness(&mut individuals);

        let error = Self::get\_generation\_error(&individuals);

        Generation { individuals, error }

    }

    fn create\_random\_individual(&self, rng: &mut ThreadRng) -> Individual {

        let mut concepts = HashMap::new();

        for concept in &self.control\_concepts {

            concepts.insert(concept.id, concept.generate\_value(rng));

        }

        let mut connections = HashMap::new();

        for connection in &self.control\_connections {

            connections.insert(connection.id, connection.generate\_value(rng));

        }

        let fitness = self.get\_individual\_fitness(&concepts, &connections);

        Individual {

            id: None,

            concepts,

            connections,

            fitness: Some(fitness),

        }

    }

    fn create\_child\_individual(

        &self,

        parent1: &Individual,

        parent2: &Individual,

        rng: &mut ThreadRng,

    ) -> Individual {

        let mut concepts = HashMap::new();

        for (id, p1) in &parent1.concepts {

            let mut p1 = \*p1;

            let mut p2 = parent2.concepts[&id];

            if p1 > p2 {

                (p1, p2) = (p2, p1);

            }

            let mut min = f64::max(p1 - ALPHA \* (p2 - p1), 0.0);

            let mut max = f64::min(p2 + ALPHA \* (p2 - p1), 1.0);

            let concept = &self.concepts\_map[id];

            match &concept.constraint {

                Some(constraint) => {

                    min = constraint.get\_min(min);

                    max = constraint.get\_max(max);

                }

                None => {}

            }

            concepts.insert(\*id, rng.gen\_range(min..=max));

        }

        let mut connections = HashMap::new();

        for (id, p1) in &parent1.connections {

            let mut p1 = \*p1;

            let mut p2 = parent2.connections[&id];

            if p1 > p2 {

                (p1, p2) = (p2, p1);

            }

            let mut min = f64::max(p1 - ALPHA \* (p2 - p1), -1.0);

            let mut max = f64::min(p2 + ALPHA \* (p2 - p1), 1.0);

            let connection = &self.connections\_map[id];

            match &connection.constraint {

                Some(constraint) => {

                    min = constraint.get\_min(min);

                    max = constraint.get\_max(max);

                }

                None => {}

            }

            connections.insert(\*id, rng.gen\_range(min..=max));

        }

        Individual {

            id: None,

            concepts,

            connections,

            fitness: None,

        }

    }

    fn get\_initial\_state(&self, concepts: &HashMap<i32, f64>) -> State {

        let mut state = concepts.clone();

        for concept in &self.regular\_concepts {

            state.insert(concept.id, concept.value);

        }

        for concept in &self.target\_concepts {

            state.insert(concept.id, concept.value);

        }

        state

    }

    fn sort\_by\_fitness(individuals: &mut Vec<Individual>) -> () {

        individuals

            .sort\_by\_key(|individual| OrderedFloat(individual.fitness.as\_ref().unwrap().error))

    }

}

pub struct TimeSimulation {

    max\_model\_time: i32,

    current\_time: i32,

    error: f64,

    concepts\_map: HashMap<i32, Arc<Concept>>,

    connections\_map: HashMap<i32, Arc<Connection>>,

    target\_concepts: Vec<Arc<Concept>>,

    dynamic\_model: DynamicModel,

    previous\_state: HashMap<i32, f64>,

    delta\_state: HashMap<i32, f64>,

    connections: HashMap<i32, f64>,

}

#[derive(Serialize)]

pub struct TimeSimulationData {

    pub time: i32,

    pub error: f64,

    pub state: HashMap<i32, f64>,

}

impl TimeSimulation {

    pub fn new(

        max\_model\_time: i32,

        concepts\_map: HashMap<i32, Arc<Concept>>,

        connections\_map: HashMap<i32, Arc<Connection>>,

        target\_concepts: Vec<Arc<Concept>>,

        dynamic\_model: DynamicModel,

        concepts: HashMap<i32, f64>,

        connections: HashMap<i32, f64>,

    ) -> Self {

        let previous\_state = concepts;

        let delta\_state = previous\_state.clone();

        Self {

            max\_model\_time,

            current\_time: 0,

            concepts\_map,

            error: Self::calculate\_error(&previous\_state, &target\_concepts),

            connections\_map,

            target\_concepts,

            dynamic\_model,

            previous\_state,

            delta\_state,

            connections,

        }

    }

    pub fn get\_max\_model\_time(&self) -> i32 {

        self.max\_model\_time

    }

    pub fn get\_current\_time(&self) -> i32 {

        self.current\_time

    }

    pub fn get\_error(&self) -> f64 {

        self.error

    }

    pub fn get\_state(&self) -> HashMap<i32, f64> {

        self.previous\_state.clone()

    }

    fn execute\_next\_value(

        &self,

        current\_state: &mut HashMap<i32, f64>,

        concept\_id: i32,

        dynamic\_model: &DynamicModel,

        to\_connections: &[(i32, f64)],

    ) -> () {

        let current\_value = current\_state.get\_mut(&concept\_id).unwrap();

        match dynamic\_model {

            DynamicModel::KoskoDeltaDelta => {

                \*current\_value += to\_connections

                    .iter()

                    .map(|(source\_id, value)| value \* self.delta\_state[&source\_id])

                    .sum::<f64>();

                \*current\_value = Self::normalize\_value(\*current\_value)

            }

            DynamicModel::KoskoDeltaValue => {

                \*current\_value += to\_connections

                    .iter()

                    .map(|(source\_id, value)| value \* self.previous\_state[&source\_id])

                    .sum::<f64>();

                \*current\_value = Self::normalize\_value(\*current\_value)

            }

            DynamicModel::KoskoValueDelta => {

                \*current\_value = Self::normalize\_value(

                    to\_connections

                        .iter()

                        .map(|(source\_id, value)| value \* self.delta\_state[&source\_id])

                        .sum::<f64>(),

                );

            }

            DynamicModel::KoskoValueValue => {

                \*current\_value = Self::normalize\_value(

                    to\_connections

                        .iter()

                        .map(|(source\_id, value)| value \* self.previous\_state[&source\_id])

                        .sum::<f64>(),

                );

            }

            DynamicModel::Silov => {

                \*current\_value = to\_connections

                    .iter()

                    .map(|(source\_id, value)| OrderedFloat(value \* self.previous\_state[&source\_id]))

                    .max()

                    .unwrap()

                    .0;

            }

        };

    }

    fn calculate\_delta\_state(&self, current\_state: &State) -> State {

        State::from\_iter(

            current\_state

                .iter()

                .map(|(k, v)| (\*k, v - self.previous\_state[k])),

        )

    }

    fn calculate\_error(state: &HashMap<i32, f64>, target\_concepts: &[Arc<Concept>]) -> f64 {

        let sum = target\_concepts

            .iter()

            .map(|concept| {

                let value = state[&concept.id];

                let target\_value = concept.target\_value.as\_ref().unwrap();

                if target\_value.include\_min\_value && value < target\_value.min\_value

                    || !target\_value.include\_min\_value && value <= target\_value.min\_value

                {

                    return (value - target\_value.min\_value).powf(2.0);

                }

                if target\_value.include\_max\_value && value > target\_value.max\_value

                    || !target\_value.include\_max\_value && value >= target\_value.max\_value

                {

                    return (value - target\_value.max\_value).powf(2.0);

                }

                0.0

            })

            .sum::<f64>();

        (sum / target\_concepts.len() as f64).sqrt()

    }

    fn normalize\_value(value: f64) -> f64 {

        if value > 1.0 {

            return 1.0;

        }

        if value <= 0.0 {

            return 0.0;

        }

        value

    }

}

impl Iterator for TimeSimulation {

    type Item = TimeSimulationData;

    fn next(&mut self) -> Option<Self::Item> {

        if self.current\_time > self.max\_model\_time {

            panic!("The current time must be less or equal to the max model time");

        }

        let mut current\_state = self.previous\_state.clone();

        for concept in self.concepts\_map.values() {

            let dynamic\_model = concept

                .dynamic\_model

                .as\_ref()

                .unwrap\_or(&self.dynamic\_model);

            let to\_connections = self

                .connections\_map

                .values()

                .filter(|connection| connection.target\_id == concept.id)

                .map(|connection| match self.connections.get(&connection.id) {

                    Some(value) => (connection.source\_id, \*value),

                    None => (connection.source\_id, connection.value),

                })

                .collect::<Vec<\_>>();

            if to\_connections.len() == 0 {

                continue;

            }

            self.execute\_next\_value(

                &mut current\_state,

                concept.id,

                dynamic\_model,

                &to\_connections,

            )

        }

        self.delta\_state = self.calculate\_delta\_state(&current\_state);

        self.previous\_state = current\_state;

        self.current\_time += 1;

        self.error = Self::calculate\_error(&self.previous\_state, &self.target\_concepts);

        if self.current\_time <= self.max\_model\_time {

            Some(TimeSimulationData {

                time: self.current\_time,

                error: self.error,

                state: self.previous\_state.clone(),

            })

        } else {

            None

        }

    }

}

impl Concept {

    fn generate\_value(&self, rng: &mut ThreadRng) -> f64 {

        match &self.constraint {

            Some(constraint) => constraint.generate\_value(rng),

            None => rng.gen\_range(0.0..=1.0),

        }

    }

}

impl Connection {

    fn generate\_value(&self, rng: &mut ThreadRng) -> f64 {

        match &self.constraint {

            Some(constraint) => constraint.generate\_value(rng),

            None => {

                if self.value >= 0.0 {

                    rng.gen\_range(0.0..=1.0)

                } else {

                    rng.gen\_range(-1.0..=0.0)

                }

            }

        }

    }

}

const SIGNIFICANT\_DIFF: f64 = 0.0000001;

impl Constraint {

    fn generate\_value(&self, rng: &mut ThreadRng) -> f64 {

        let mut number = rng.gen\_range(self.min\_value..=self.max\_value);

        let mut attempts = 0;

        while !self.include\_min\_value && number == self.min\_value

            || !self.include\_max\_value && number == self.max\_value

        {

            number = rng.gen\_range(self.min\_value..=self.max\_value);

            attempts += 1;

            if attempts >= 1000 {

                return (self.min\_value + self.max\_value) / 2.0;

            }

        }

        number

    }

    fn get\_min(&self, min: f64) -> f64 {

        f64::max(

            min,

            if self.include\_min\_value {

                self.min\_value

            } else {

                self.min\_value + SIGNIFICANT\_DIFF

            },

        )

    }

    fn get\_max(&self, max: f64) -> f64 {

        f64::min(

            max,

            if self.include\_max\_value {

                self.max\_value

            } else {

                self.max\_value - SIGNIFICANT\_DIFF

            },

        )

    }

}

type State = HashMap<i32, f64>;