МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики —
филиал федерального государственного автономного образовательного
учреждения высшего образования
«Национальный исследовательский ядерный университет «МИФИ»
(ИАТЭ НИЯУ МИФИ)

Отделение интеллектуальных кибернетических систем

На правах рукописи

Белявцев Иван Павлович

НАУЧНО-КВАЛИФИКАЦИОННАЯ РАБОТА

ИНТЕЛЛЕКТУАЛЬНЫЕ МЕТОДЫ И ПРОГРАММНЫЕ СРЕДСТВА ОПТИМИЗАЦИИ ЭКСПЛУАТАЦИОННЫХ ПРОЦЕССОВ ЯДЕРНЫХ РЕАКТОРОВ

Направление 09.06.01 — «Информатика и вычислительная техника»
Специальность 05.13.05 — «Элементы и устройства вычислительной техники
и систем управления»

Научный руководитель:
И.о. руководителя отделения ИКС
д.ф.-м.н., профессор С.О. Старков

Обнинск, 2018

Содержание

Введение **4**

1 Эксплутационные процессы ядерных реакторов **10**
1.1 Ядерные энергетические установки 10
1.1.1 Устройство энергетического реактора 10
1.1.2 Процесс перегрузки ядерного топлива 13
1.2 Существующие программные средства планирования перегрузки 14
1.2.1 Программное средство TRIGEX.051 15
1.2.2 Программное средство САПФИР_95 15
1.2.3 Программное средство RC_ВВЭР 15
1.2.4 Программное средство REPRORYV 16
1.3 Выводы по главе 17

2 Автоматная модель технологических процессов **18**
2.1 Задача оптимизации технологического процесса 18
2.2 Автоматная модель технологического процесса 20
2.3 Применения автоматной модели для описания процесса перегрузки реактора . 22
2.4 Выводы по главе 30

3 Поиск оптимальной последовательности автоматной модели **31**
3.1 Общий подход к поиску оптимальной последовательности 31
3.2 Классические алгоритмы поиска пути в графе 32
3.2.1 Алгоритм Дейкстры 33
3.2.2 Пример работы алгоритма Дейкстры 35
3.2.3 Алгоритм A* 38
3.2.4 Пример работы алгоритма A* 39
3.2.5 Волновой алгоритм 41
3.2.6 Пример работы волнового алгоритма 42
3.3 Сравнительный анализ алгоритмов 44
3.4 Нейросетевой подход 45
3.5 Выводы по главе 46

4 Аппроксимация эмерджентных свойств автоматной модели **47**

4.1 Проблема потери интегративных свойств в автоматной модели 47
4.2 Нейросетевая аппроксимация запаса реактивности 47
4.2.1 Постановка задачи 47
4.2.2 Применение искусственных нейронных сетей для аппроксимации запаса ективности 48
4.3 Аппроксимация модельных данных 49
4.3.1 Данные 49
4.3.2 Архитектура сети 50
4.3.3 Процесс обучения 50
4.3.4 Верификация 50
4.4 Аппроксимация измеренных данных 52
4.4.1 Данные 52
4.4.2 Архитектура сети 52
4.4.3 Процесс обучения 52
4.4.4 Валидация 52
4.4.5 Выводы 52
4.5 Программный комплекс предсказания 54
4.6 Выводы по главе 56

Заключение 58

Список рисунков 60

Список таблиц 61

Литература 62

А Исходный код нейросетевой аппроксимации 66

Введение

Актуальность темы исследования.

В настоящее время в мире эксплуатируется 451 ядерная энергетическая устновка [1]. Согласно прогнозам экспертов, к 2040 году ожидается 60%-ный рост мощности суммарной мощности ядерных электростанций. Таким образом в 20-летней перспективе суммарные мировые запасы отработанного ядерного топлива составят порядка 700 тыс. тонн [2]. Большое количество отработанного ядерного топлива представляет собой не только серьезную проблему по организации дорогостоющих процессов переработки и захоронения, но и также ставит задачи по обеспечению экологической и антитеррористической защите.

Существующий открытый ядреный топливный цикл обладает низкой эффективностью использования урана – менее 1%. В настоящее время РосАтом активно развивает технологии закрытого ядерного топливного цикла, которые позволят переиспользовать более 95% ядерного топлива, что поможет значительно снзить объем захоронений ядерных отходов. [3]

Замкнтый ядерный цикл требует постройки большого количества реакторов на быстрых нейтронах типа СВБР-100, БН-1200, БРЕСТ. Кроме того требуется развернуть в больших масштабах производство топлива для реакторов на быстрых нейтронах. Эти задачи запланированы РосАтомом как среднесрочные и долгосрочные. А в краткосрочной перспективе, существует задача оптимизации технологий существующих ядерных энергетических установок. [3]

Одним из направлений оптимизации существующих ядерных энергетических установок является достижение большей глубины выгорания топлива. При этом наименее затратным способом является оптимизация порядка перегруок ядерного топлива.

Существующие методы и программные средства моедлилрования и расчета реакторов пологаются на устоявшиеся последовательности перегрузки ядерного топлива или требуют ручного использования. Решаемая научная и практическая задача состоит в разработке и реализации метода оптимизации технологических процессов ЯЭУ, а также программных средств, реализующих данный метод.

Объект исследования — эсплутационные процесс перегрузки ядерного топлива в исследовательских и энергетических ядерных реакторах.

Предмет исследования — интеллектуальные методы и программные средства оптимизации эксплутационных процессов ядерных реакторов.

Целью исследования является решение задач автоматической оптимизации порядка перезагрузки ядерного топлива ядерных реакторов.

Для достижения поставенной цели тербуется решение следующих **задач**:

1. Разработать описательную модель эксплутационных процессов ядерных реакторов;
2. Исследовать и разработать алгоритмы поиска оптимальной последовательности экплутационного процесса;
3. Разработать алгоритм аппроксимации эмерджентных свойств для описательной модели.

Методы исследования. Работа базируется на методах теории автоматов, методах теории графов, численных методах и искуственных нейронных сетях.

Научная новизна результатов исследования состоит в следующем

1. Впервые предложен подход к моделированию эксплутационных процессов ядерных реакторов на основе иерархических автоматов;
2. Представлены алгоритмы поиска оптимальной последовательности экспутационного процесса;
3. Предложен и реализован метод построения аппроксимации эмерджентных свойств с использованием искуственных нейронных сетей;
4. Реализован программный комплекс аппроксимации запаса реактивности реактора ВВР-ц.

Практическая значимость.

Научные и практические результаты диссертации использованы при разработке программного комплекса аппроксимации запаса реактивности реактора ВВР-ц. Предполагаемым потребителем результатов диссертации является филиал АО «НИФХИ им. Л.Я.Карпова» в г. Обнинск, эксплуатирующий реактор ВВР-ц.

Основные положения, выносимые на защиту.

1. Подход к построению автоматной модели эксплутационных процессов ядерных реакторов;
2. Алгоритмы поиска оптимальной последовательности процесса в автоматной модели;
3. Метод аппроксимации эмерджентных свойств для автоматной модели на основе искусственных нейронных сетей;
4. Программный комплекс аппроксимации запаса реактивности реактора ВВР-ц.

Апробация работы.

Материалы диссертации докладывались на следующих всероссийских и международных конференциях:

1. Всероссийская научно-техническая конференция «Наукоемкие технологии в приборо- и машиностроении и развитие инновационной деятельности в ВУЗе», г. Калуга, 25-27 ноября 2014 г.;
2. Научная сессия НИЯУ МИФИ, г. Москва, 16-20 февраля 2015 г.;
3. XIV Международная конференция «Безопасность АЭС и подготовка кадров», г. Обнинск, 25-27 ноября 2015 г.;
4. VI-я Международная молодежная научная школа-конференция «Современные проблемы физики и технологий», г. Москва, 17-21 апреля 2017 г.;
5. VII-я Международная молодежная научная школа-конференция «Современные проблемы физики и технологий», г. Москва, 16-21 апреля 2018 г.

Публикации.

Основные результаты исследованиий изложены в 9 научных трудах, 2 из которых опубликованы в изданиях, рекомендованных ВАК, или приравненных к ним изданиям.

Содержание работы

Во **введении** обоснована актаульность темы научной работы, сформулированы цели и задачи, перечислены полученные в работе новые результаты, их практическая значимость, представлены положения, выносимые на защиту, описана структура диссертации.

В **первой главе** приведен обзор устройства ядерного реактора на примере реактора БН-350. Рассмотрны основные конструкционные элементы, а также произведен разбор процесса перегрузки ядерного топлива.

Выполнен сравнительный анализ существущих программных средств, применяемых для планирования перегрузки: TRIGEX, САПФИР_95, RC_ВВЭР, REPRORYV. Выявлено, что существующие программные комплексы способны проводить расчет либо только для стационарных схем перегрузки, либо по заданной вручную схеме.

Сделан вывод о необходимости алгоритмического решения задачи оптимизации технологических процессов ядерных реакторов на примере процесса перегузки ядерного топлива.

Во **второй главе** сформулирована постановка задачи оптимизации технологического процесса. Предложен общий подход к решению задачи оптимизации технологического процесса. Определены следующие входные данные: описание конструкции реактора, данные для физического моделирования и функция оптимизации. В качестве выходных данных определн оптимальный порядок технологического процесса.

Рассмотрены основные положения теории автоматов. Введено понятие автоматной модели (ТА-модели). Рассмотрен подход к построению композиции ТА-моделей.

Показан пример построения ТА-модели фиктивного реактора.

В **третьей главе** рассмотрен подход к поиску оптимальной последовательности в автоматной модели. Показана возможность представления детерминированного конечного автомата в виде направленного графа (диаграммы переходов). Таким образом, задача поиска оптимальной последовательности автоматной модели сведена к задачи посика кратчайшего пути в графе.

Рассмотрены классические алгоритмы поиска пути в графе: алгоритм Дейкстры, алгоритм A* и волновой алгоритм. Для каждого из алгоритмов приведен пример. Из класических алгоритмов наиболее подходящим является алгоритм A*, применение которого рекомендуется для небольших и средних ТА-моделей.

Для больших моделей предложен нейросетевой алгоритм для поиска последовательности.

В **четвертой главе** рассмотрена возможность аппроксимации запаса реактивности реактора с помощью полносвязной искусственной нейронной сети; обучены две искусственные нейронные сети на разных наборах данных (на модельных, полученных с помощью прецизионной модели реактора, и на измеренных данных реальных кампаний).

Показано, что обе аппроксимации обладают достаточной точностью для проведения предварительных расчетов запаса реактивности. По итогам вычислительных экспериментов максимальная относительная ошибка аппроксимации составила 3,13 и 3,56% соответственно.

На основе обученных искусственных нейронных сетей создан программный комплекс оценки запаса реактивности реактора ВВР-ц. Комплекс позволяет в удобной и наглядной форме получить предсказанное нейронной сетью значение запаса реактивности, а также пополнить обучающую выборку новыми данными для обучения.

Программный комплекс для оценки запаса реактивности готов для тестирования персоналом реактора ВВР-ц. Параллельно с тестированием в данный программный комплекс можно внести ряд изменений, повышающих удобство и безопасность эксплуатации: шифрование данных в обучающей выборке; авторизацию, аутентификацию и аккаунтинг пользователей; возможность ручного редактирования обучающей выборки.

Допустимо использование данного программного комплекса в виде компонента системы автоматического планирования перезагрузки реактора. С незначительными изменениями комплекс можно применять для реакторных установок других типов.

В **Приложении** представленны исходные тексты программных продуктов, разработанных в ходе научной работы.

Публикации по теме работы

Статьи, опубликованные в перечне ведущих рецензируемых научных журналов, рекомендованных ВАК, и приравненных к ним изданиям:

1. Approximation of the criticality margin of WWR-c reactor using artificial neuron networks. / I. Belyavtsev, D. Legchikov, S. Starkov [и др.] // Journal of Physics: Conference Series. 2018. Т. 945, № 1. С. 012–031. (Scopus) [4]

2. Белявцев И.П., Старков С.О. Программный комплекс оценки запаса реактивности реактора ВВР-ц. // Известия высших учебных заведений. Ядерная энергетика. 2018. № 2. С. 58–66. [5]

Статьи в сборниках научных трудов и сборниках трудов конференции:

3. Белявцев И.П., Старков С.О. Моделирование и оптимизация эксплутационных процессов на атомных электростанциях с использованием методов искусственного интеллекта. // Наукоемкие технологии в приборо- и машиностроении и развитие инновационной деятельности в ВУЗе: материалы Всероссийской научно-технической конференции, 25-27 ноября 2014 г. Т. 4. М.: Издательство МГТУ им. Н.Э. Баумана, 2014. С. 4–9. [6]

4. Построение нейросетевой модели реактора ВВР-ц для прогнозирования запаса критичности. / И.П. Белявцев, С.О. Старков, Д.К. Легчиков [и др.] // Наукоемкие технологии в приборо- и машиностроении и развитие инновационной деятельности в ВУЗе: материалы Всероссийской научно-технической конференции, 25-27 ноября 2014 г. Т. 4. М.: Издательство МГТУ им. Н.Э. Баумана, 2014. С. 10–15. [7]

5. Белявцев И.П., Старков С.О. Моделирование эксплутационных процессов ядерных реакторов с использованием методов искусственного интеллекта. // Научная сессия НИЯУ МИФИ-2015. Аннотации докладов. Т. 2. М.: НИЯУ МИФИ, 2015. С. 271. [8]

6. Прогнозирование запаса критичности реактора ВВР-ц методом нейросетевого моделирования. / И.П. Белявцев, С.О. Старков, Д.К. Легчиков [и др.] // Научная сессия НИЯУ МИФИ-2015. Аннотации докладов. Т. 2. М.: НИЯУ МИФИ, 2015. С. 272. [9]

7. Белявцев И.П., Старков С.О., Колесов В.В. Прогнозирование запаса критичности реактора ВВР-ц методом нейросетевого моделирования. // XIV Международная конференция «Безопасность АЭС и подготовка кадров». Тезисы докладов. Обнинск: ИАТЭ НИЯУ МИФИ, 2015. С. 138–139. [10]

8. Аппроксимация запаса критичности реактора ВВР-ц с использованием исскусственной нейронной сети. / И.П. Белявцев, Д.К. Легчиков, С.О. Старков [и др.] // Современные проблемы физики и технологий. VI-я Международная молодежная научная школа-конференция, 17-21 апреля 2017 г.: Тезисы докладов. Часть 1. М.: НИЯУ МИФИ, 2017. С. 80–81. [11]

9. Белявцев И.П., Старков С.О. Поиск оптимальной последовательности событий автоматной модели с использованием искусственных нейронных сетей. // Современные

проблемы физики и технологий. VII-я Международная молодежная научная школа-конференция, 16-21 апреля 2017 г.: Тезисы докладов. Часть 2. М.: НИЯУ МИФИ, 2018. С. 346–347. [12]

Глава 1

Эксплутационные процессы ядерных реакторов

1.1 Ядерные энергетические установки

1.1.1 Устройство энергетического реактора

Рассмотрим устройство энергетического реактора на примере ядерной энергетической установки БН-350. Реактор БН-350 — первый реактор на быстрых нейтронах энергетического назначения (рис. 1.1). В его конструкцию были заложены многие технические решения, выработанные на реакторе БОР-60, и впоследствии реализованные в конструкциях реакторов БН-600 и БН-800. Внутри герметичного корпуса реактора, выполненного в виде бака переменного диаметра, расположены тепловыделяющие сборки (ТВС) активной зоны и боковой зоны воспроизводства, хранилище отработавших ТВС, отражатель нейтронов, механизмы системы перегрузки, стержни системы управления и защиты (СУЗ). Натрий под давлением 1 МПа подводится от главных циркуляционных насосов (ГЦН) первого контура по шести напорным трубопроводам диаметром 500 мм в нижнюю часть корпуса реактора, образующую напорную камеру. Сверху на камере закреплен напорный коллектор с установленными в нем ТВС. В коллекторе поток теплоносителя распределяется по ТВС в соответствии с уровнем энерговыделения в них. Он представляет собой прочную силовую конструкцию, воспринимающую весовую нагрузку от ТВС и других выемных частей реактора, а также внутренние давление теплоносителя. Крепление коллектора к напорной камере допускает его извлечение и замену в случае необходимости.

Пройдя ТВС, нагретый натрий попадает в выходную смесительную камеру, откуда по шести сливным трубопроводам диаметром 600 мм отводится к промежуточным теплообменникам.

Активная зона набирается из шестигранных ТВС размером «под ключ» 96 мм. Каждая сборка содержит пучок гладкостержневых тепловыделяющих элементов (ТВЭЛов) диаметром 6,9 мм, расположенных с шагом 8 мм. Топливо — обогащенная двуокись урана. Ди-

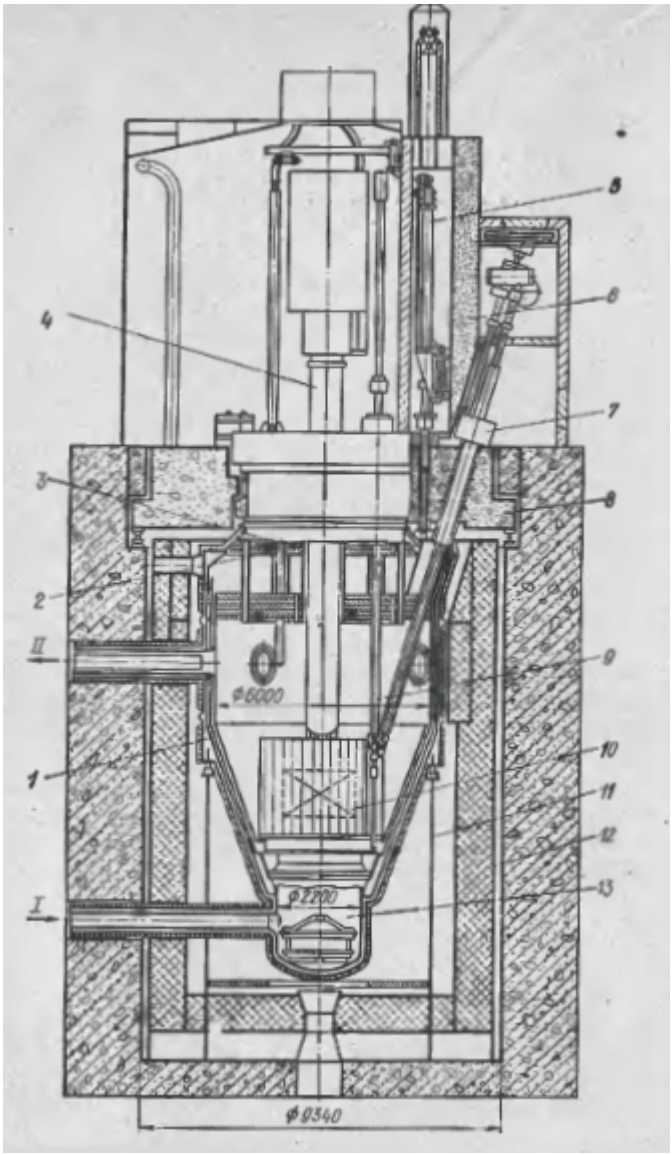

Рисунок 1.1: Общий вид реактора БН-350: 1 – корпус реактора; 2 – большая поворотная пробка; 3 – малая поворотная пробка; 4 – центральная поворотная колонка с механизмом СУЗ; 5 – механизм передачи сборок; 6 – передаточный бокс; 7 – элеватор загрузки-выгрузки; 8 – верхняя неподвижная защита; 9 – механизм перегрузки; 10 – активная зона; 11 – опора реактора; 12 – боковая защита; 13 – напорная камера; I – вход натрия; II – выход натрия

станцирование ТВЭЛов в пучке осуществляется проволокой, спирально навитой на оболочку ТВЭЛа. Верхняя торцевая зона воспроизводства набирается из элементов увеличенного диаметра (12 мм) с двуокисью обедненного урана, установленных внутри сборки над пучком ТВЭЛов. Гексагональная чехловая труба сборки несет давление теплоносителя, которое в нижней части значительно превышает давление в межкассетном пространстве реактора. К чехлу сборки с обоих торцов приварены концевые детали цилиндрической формы: нижний хвостовик с боковыми отверстиями, через которые в ТВС поступает теплоноситель, и фигурная головка с выходными окнами, предназначенная для захвата сборки механизмом перезагрузки. Для уменьшения паразитных протечек натрия из напорного коллектора вдоль хвостовика сборки на нем предусмотрены лабиринтные уплотнения. Зазор между ТВС принят минимальным (2 мм) исходя из условий нормального извлечения их при перезагрузке с учетом возможных при работе формоизменений. Для пространственной фиксации сборок они дистанцируются с помощью выступов («платиков») в верхней части чехловых труб.

Часть гнезд в центральной области активной зоны занята органами СУЗ (рис. 1.2) Бо-

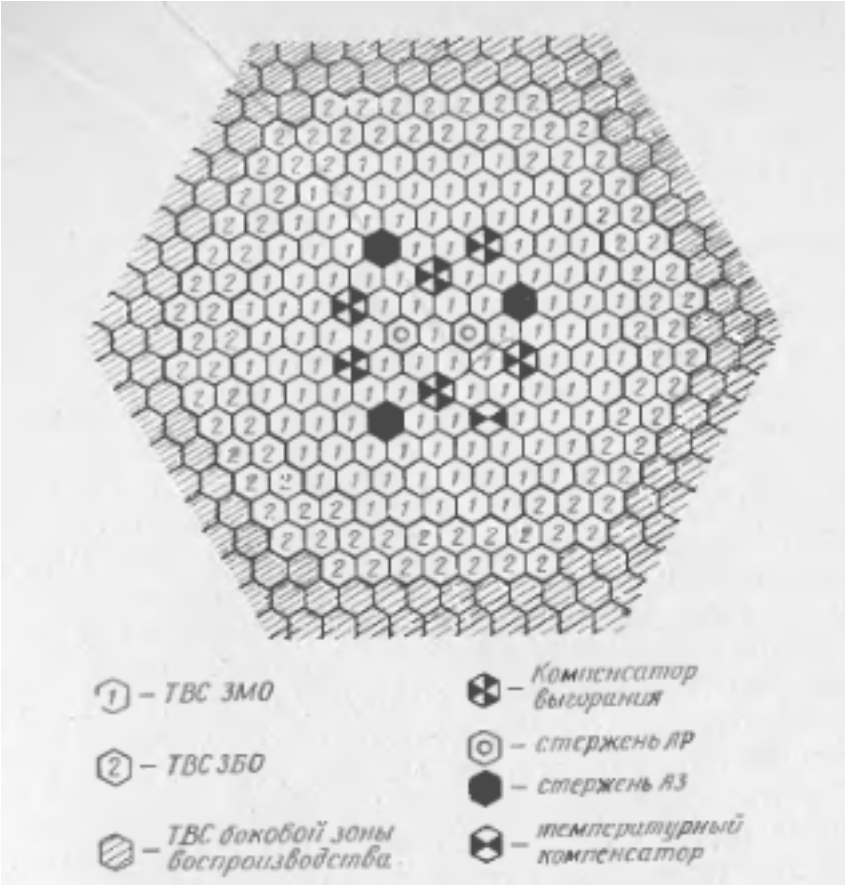

Рисунок 1.2: Сетка СУЗ в реакторе БН-350: ЗМО – зона малого обогащения, ЗБО – зона большого обогащения

ковая зона воспроизводства общей толщиной 600 мм набрана из нескольких рядов ТВС, каждая из которых состоит из элементов диаметром 14,2 мм с двуокисью обедненного урана. За внешним рядом сборок боковой зоны воспроизводства размещаются гнезда внутрире-

акторного хранилища, где отработавшие ТВС выдерживаются в течение интервала между перезагрузками реактора (2 месяца). Необходимость выдержки сборок активной зоны перед выгрузкой из реактора обусловлена высоким уровнем остаточных тепловыделений в них. В хранилище ТВС расхолаживаются теплоносителем, поступающим из напорного коллектора реактора. После выдержки сборок значительно упрощается обращение с ними в тракте перегрузки. Сборки боковой зоны воспроизводства имеют относительно низкое остаточное тепловыделение, поэтому предварительного охлаждения их перед выгрузкой не требуется. Непосредственно за хранилищем устанавливается отражатель нейтронов из нескольких рядов стальных болванок с внешней конфигурацией ТВС (общая толщина 200 мм).

Натрий, заполняющий корпус реактора, образует свободный уровень, над которым находится аргоновая подушка. Газовый объем служит для компенсации температурных расширений теплоносителя. Вместе с тем он изолирует верхнюю часть реактора от горячего теплоносителя, выходящего из активной зоны, и защищает натрий от контакта с воздухом при случайной разгерметизации реактора. Высота уровня над головками ТВС выбрана таким образом, чтобы транспортировка выгружаемых сборок проходила под слоем натрия, чем обеспечивается их надежное и эффективное охлаждение. [13]

1.1.2 Процесс перегрузки ядерного топлива

Рассмотрим процесс перегрузки реактора БН-350.

Характерной особенностью конструкции реактора является то, что он не имеет съемной герметизирующей крышки, а перегрузка ТВС осуществляется без общего вскрытия корпуса герметичными дистанционно управляемыми механизмами под защитой инертного газа по закрытому тракту от реактора до внешнего хранилища. Такое решение продиктовано специфическим требованием натриевой технологии — недопустимостью контакта натрия с воздухом. Сверху корпус реактора закрыт двумя многослойными плитами («пробками») верхней радиационной защиты. Защитные пробки являются одновременно частью системы перегрузки реактора. С их помощью осуществляются наведение внутриреакторного механизма перегрузки (ВМП) на ТВС, подлежащие перегрузке и перенос сборок внутри реактора. Эти операции выполняются совместным вращением обоих пробок — большой, перекрывающей горловину реактора, и расположенной в ней эксцентрично малой пробки, в которую вмонтирован ВМП. Обе пробки установлены на шаровых опорах, имеют по периферии зубчатый венец и электромеханические приводы, обеспечивающие их вращение по командам автоматизированной системы управления. Поворотные пробки имеют значительный диаметр (большая пробка – 4300 мм), поэтому создание надежного механического уплотнения по всему периметру, исключающего выход из реактора радиоактивного защитного газа, является весьма сложной технической задачей. Благодаря небольшой разнице давлений между газовой полостью реактора и окружающей средой оказалось возможным выполнить уплотнение пробок в виде гидрозатвора: каждая пробка имеет цилиндрическую юбку, которая опущена в кольцевую

ванну, заполненную тяжелой уплотняющей жидкостью, герметично связанную с корпусом реактора. Уплотняющей средой в гидрозатворе служит сплав олова и висмута (57% Sn и 43% Bi), имеющий плотность $8,3 \cdot 10^3$ кг/м^3 и температуру плавления 138°C. При работе реактора сплав находится в твердом состоянии и вращение пробок невозможно. Перед перегрузкой его расплавляют с помощью электронагревателей, вмонтированных в корпус гидрозатвора.

Перегрузка ТВС осуществляется на остановленном реакторе комплексом взаимодействующих механизмов в режиме автоматического управления. Операции с ТВС внутри реактора (извлечение, перенос, установка) выполняются ВМП. Он представляет собой прямую телескопическую штангу с захватным устройством и несколькими электромеханическими приводами. После сцепления захвата с головкой перегружаемой сборки последняя втягивается внутрь направляющей трубы ВМП и в таком положении переносится вращением пробок в заданное место. Перенос ТВС в направляющей трубе предохраняет сборку от раскачивания и случайных механических повреждений. Выгрузка отработавших ТВС из реактора и загрузка свежих в реактор осуществляются через специальный перегрузочный патрубок небольшого диаметра в верхней части корпуса реактора с помощью двух механизмов: элеватора и механизма передачи сборок (МПС). Элеватор представляет собой наклонный подъемник с цепным приводом и подвижной кареткой, в которую ВМП устанавливает перегружаемую сборку. Движением каретки по наклонной направляющей сборка перемещается вверх под перегрузочный патрубок, в котором установлен МПС. В точке встречи с механизмом передачи головка сборки выходит из-под уровня натрия в газовую полость реактора. Это исключает погружение в натрий захвата МПС и повышает его надежность. МПС втягивает сборку в герметичный передаточный бокс, установленный над перегрузочным патрубком, и транспортирует ее во внешние хранилище отработавших сборок. Двигаясь в обратном направлении, МПС захватывает свежую ТВС, переносит ее к патрубку и устанавливает в каретку элеватора, которая опускает сборку в нижнее положение. Отсюда она переносится с помощью ВМП в свободное гнездо активной зоны или боковой зоны воспроизводства. С учетом совмещения во времени операций, выполняемых различными механизмами перегрузочного комплекса, полный цикл перегрузки одного гнезда активной зоны (от извлечения отработавшей сборки до установки свежей) занимает около 50 минут. [13]

1.2 Существующие программные средства планирования перегрузки

Рассмотрим существующие программные комплексы, используемые при планировании кампаний реакторов и перегрузок ядерного топлива.

1.2.1 Программное средство TRIGEX.051

Программное средство TRIGEX разработанно в АО ГНЦ РФ «Физико-энергетический институт им. А.И. Лейпунского» г. Обнинск. Программное средство TRIGEX предназначено для расчета следующих характеристик:

- коэффициент размножения нейтронов в активной зоне реактора;
- эффективность средст управления и защиты (СУЗ);
- изменение реактивности реактивности в течение кампании;
- температурные и пусттные эффекты реактивности;
- долю запаздывающих нейтронов;
- пространственное энерговыделение.

Данные характеристики расчитываются для быстрых нейтронных реакторов с натриевым теплоносителем. Поддерживается оксимд урановое ядерное топливо, а также эксперементальные сборки с МОКС-топливом. Материал поглотителей органов СУЗ — карбид бора.

Программное средство TRIGEX.051 атестовано для работы с реактором БН-800. Для установившегося режима перегрузок реактора учитывается предполагаемое системное расходжение расчетных и фактических величин [14].

1.2.2 Программное средство САПФИР_95

Програмное средство САПФИР_95 (Система Алгоритмов и Программ для Физических Исследований Реакторов) разработано в ФГУП «Научно-исследовательский технологический институт им. А.П. Александрова» г. Сосновый Бор.

Средство САПФИР_95 предназначено для расчета пространственного и энергетичегого распределения нейтронов в ячейках водо-водяных и водо-графитовых реакторов. Для расчета энергетического распределения нейтронов используется метод группового приближения в быстрой области, обобщенный подгрупповой подход в резонансных областях, а также разбиение на 40 микрогрупп в тепловой области. Для расчета пространтвенного распредлеения нейтронов используется метод вероятностей первых столконовений.

САПФИР_95 является базовой расчетоной частью для программного средства RC_ВВЭР [15].

1.2.3 Программное средство RC_ВВЭР

Програмnot средство RC_ВВЭР предназначено для трехмерного расчета полей нейтронов и энерговыделения. В ходе моделирования производится расчет выгорания ядерного топлива при предварительно задаанных параметрах реактора. Моделируются следующие про-

цессы перегрузки реактора: перестановка и поворот ТВС, установка новых тепловыделющих сборок. В качестве основы для расчетов используетсяя программное средство САПФИР_95.

Комплекс программ САПФИР_95 и RC_ВВЭР используется для обоснования безопасности реакторов ВВЭР. В качестве расчетного сопровождения програмный комплек также используется в хранилоищах отработанного топлива на Ленинградской АЭС [16].

1.2.4 Программное средство REPRORYV

Программное средство REPRORYV (RecycleforPRORYV) моделирует процесс загрузки, переработки и перегрузки ТВС в активных зонах быстрых реакторов типа БН-600, БН-800 и БН-1200 и других перспективных быстрых реакторов.

Целью создания комплекса является анализ возможности осуществления режима самообеспечения реактора топливом на протяжении всего его срока службы.

Для расчёта по программе REPRORYV необходимо вначале сформировать входной файл для программы JARFR. Затем по JARFR будут рассчитываться функционалы на каждом из шагов. Для анализа замкнутого ядерного топливного цикла в отдельном входном файле REPRORYV необходимо задать схему перегрузок, количество лет на выдержку и переработку топлива, выделить изотопы в различные группы и задать условия переработки для каждой из групп.

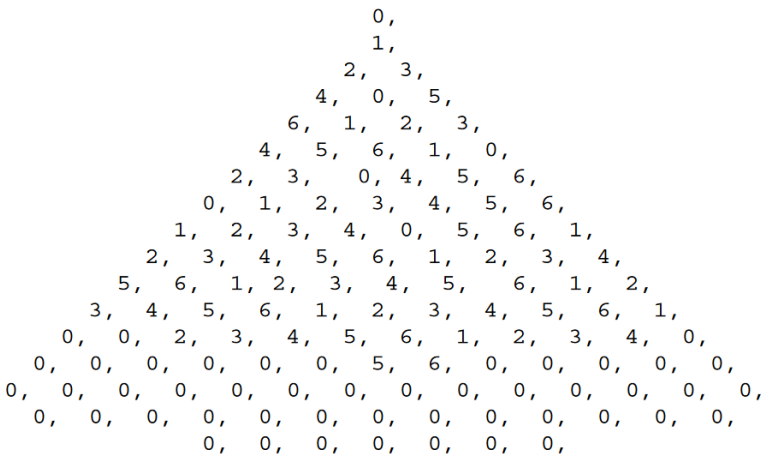

Рисунок 1.3: Картограмма порядка выгрузки ТВС для программного средства RERPRORYV

Схема перегрузок подразумевает задание групп ТВС, которые выгружаются по окончании очередной микрокампании (рис. 1.3). При этом выгружаются не все ТВС из активной зоны, а только их часть, через год – ещё одна часть, через ещё год – ещё одна и т .д . Пользователь может задать, какую часть активной зоны необходимо будет выгружать каждый год. Перестановок ТВС внутри активной зоны нет [17].

1.3 Выводы по главе

В данной главе приведен обзор устройства ядерного реактора на примере реактора БН-350. Рассмотрны основные конструкционные элементы, а также произведен разбор процесса перегрузки ядерного топлива.

Выполнен сравнительный анализ существущих программных средств, применяемых для планирования перегрузки: TRIGEX, САПФИР_95, RC_ВВЭР, REPRORYV. Выявлено, что существующие программные комплексы способны проводить расчет либо только для стационарных схем перегрузки, либо по заданной вручную схеме.

Таким образом, необходимо разработать метод для оптимизации технологических процессов ядерных реакторов на примере процесса перегузки ядерного топлива.

Глава 2

Автоматная модель технологических процессов

2.1 Задача оптимизации технологического процесса

Пусть имеется следующая информация о реакторе и о его процессе планово-предупредительного ремонта:

1. Абстрактное описание конструкции реактора и порядка процесса планового-предупредительного ремонта. Данное описание должно быть представлено в виде абстрактного цифрового автомата или композиции таких автоматов. То есть должны быть заданы множество состояний реактора, множество операций, производимых над реактором, функция перехода, указывающая допустимые переходы из одного состояния в другое при приведении операций, а также начальное и конечные состояния реактора при ремонте. При этом для каждого состояния реактора и каждой операции заданы известные физические характеристики. Назовем такое описание ТА-моделью.

2. Данные для физического моделирования. Данные для физического моделирования представляют собой набор физических характеристик, заданных в ТА-модели, и соответствующие им расчетные характеристики реактора, такие как мощность, температура активной зоны, запас реактивности и т.д. Расчётные характеристики могут быть рассчитаны с помощью эксплуатационных программ либо измерены в ходе предыдущих кампаний. Также для каждого значения должна быть указана физическая допустимость представленных параметров.

3. Функция оптимизации. Данная функция является весовой функцией характеристик операций, производимых над реактором. Она требуется для изменения значимости различных факторов при проведении оптимизации. Поиск оптимального порядка будет производиться по минимальным значениям данной функции. Обработав исходные данные, метод должен представить на выходе оптимальный согласно функции оптимиза-

ции порядок проведения перезагрузки реактора. При этом искомый порядок не должен содержать физически недопустимых состояний реактора.

Рассмотрим порядок работы метода моделирования и оптимизации (см. рис. 2.1). Искус-

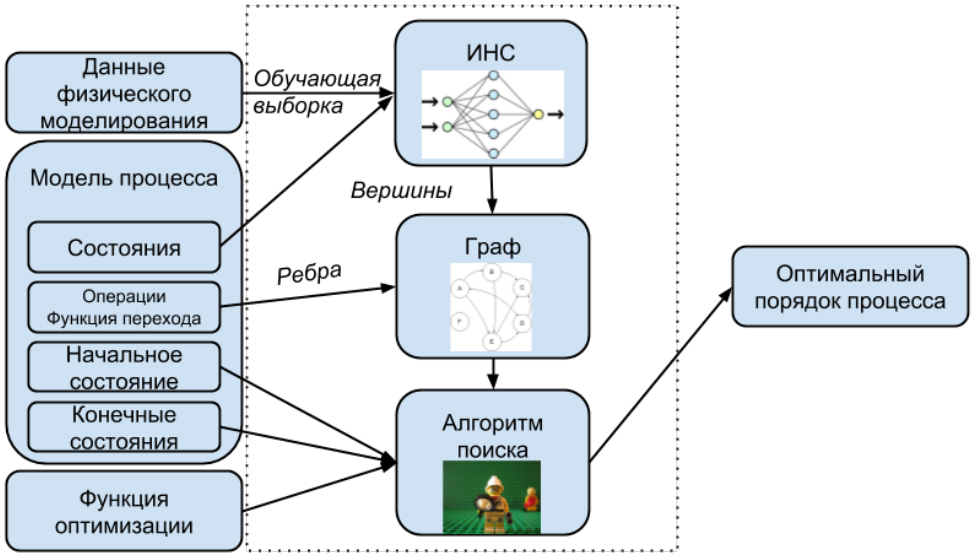

Рисунок 2.1: Обзорная диаграмма процесса моделирования и оптимизации

ственная нейронная сеть обучается на данных физического моделирования. После успешного завершения обучения нейронная сеть становится способна аппроксимировать расчётные характеристики реактора на основе физического состояния реактора (состояния ТА-модели реактора), а также принимать решение о физической допустимости состояния. Множество состояний ТА-модели реактора поэлементно подаются в качестве входного вектора искусственной нейронной сети. Нейронная сеть делит множество состояний на допустимые и недопустимые, дополняя при этом каждое состояние значениями расчетных характеристик. Допустимые состояния составляют узлы графа переходов ТА-модели, остальные отбрасываются. Между узлами графа переходов устанавливаются направленные ребра, соответствующие множеству операций и функции перехода ТА-модели. По завершению процесса построения имеется граф переходов между допустимыми состояниями реактора. С помощью алгоритма информированного поиска A* производится поиск кратчайшего пути от начального состояния графа до ближайшего конечного состояния. В качестве информирующей характеристики алгоритма выступает заданная функция активации. По завершения работы алгоритма поиска имеется множество путей, ведущих кратчайшим путем к каждому из конечных состояний. Путь с минимальным значением функции оптимизации является искомым. [6, 8]

2.2 Автоматная модель технологического процесса

риведем общие понятия и положения теории автоматов, необходимые для построения формальной информационной модели реактора.

Определение 2.1. *Алфавитом* Σ *называют некоторое непустое конечное множество символов.*

Определение 2.2. *Словом называют некоторую конечную (возможно, пустую) последовательность символов алфавита:* $\omega = \sigma_1\sigma_2\ldots\sigma_l$. *Количество символов в слове (число* l*) называют длиной слова. Пустое слово принято обозначать* ε.

Определение 2.3. *Детерминированным конечным автоматом (ДКА) называется пятерка (кортеж)* $A = \langle Q, \Sigma, \delta, q_0, F\rangle$, *где:*

- Q — *конечное множество состояний;*
- Σ — *алфавит;*
- $\delta : Q \times \Sigma \to Q$ — *функция переходов;*
- $q_0 \in Q$ — *начальное состояние;*
- $F \subseteq Q$ — *множество терминальных (конечных) состояний.*

Обработка слова $\omega = \sigma_1\sigma_2\ldots\sigma_l$ ДКА A происходит следующим образом. Сначала автомат A находится в стартовом состоянии q_0. Затем на каждом шаге обработки автомат считывает очередной символ $sigma_i$ слова ω и переходит из своего текущего состояния q в состояние $\delta(q;\omega_i)$. К моменту, когда все символы входного слова ω обработаны, автомат находится в некотором состоянии p. Говорят, что автомат А принимает слово ω, если $p \in F$, и не принимает в обратном случае. [18]

Определение ДКА как пятерки объектов с детальным описанием функций переходов слишком сухое и неудобочитаемое. Существует два более удобных способа описания автоматов.

1. *Диаграмма переходов*, которая представляет собой граф.
2. *Таблица переходов*, дающая табличное представление функции δ. Из нее очевидны состояния и входной алфавит.

Диаграмма переходов (см. рис. 2.2) для ДКА вида $A = \langle Q, \Sigma, \delta, q_0, F\rangle$ есть граф, определяемый следующим образом:

а) всякому состоянию из Q соответствует некоторая вершина;

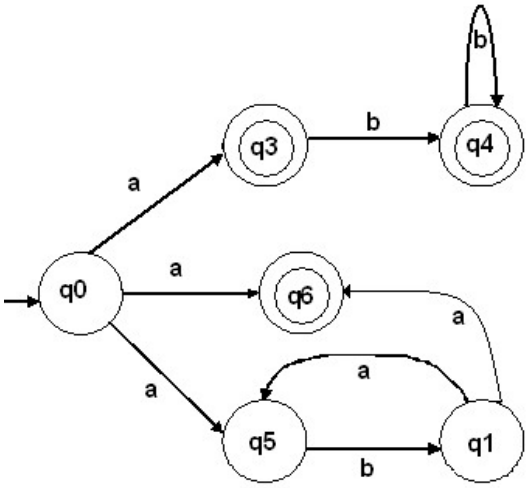

Рисунок 2.2: Пример диаграммы перехода конечного автомата

б) пусть $\delta(q, a) = p$ для некоторого состояния q из Q и входного символа a из Σ. Тогда диаграмма переходов должна содержать дугу из вершины q в вершину p, отмеченную a. Если существует несколько входных символов, переводящих автомат A из состояния q в состояние p, то диаграмма переходов может содержать одну дугу, отмеченную списком этих символов;

в) диаграмма содержит стрелку в начальное состояние, отмеченную как *Начало*. Эта стрелка не выходит не из какого состояния;

г) вершины, соответствующие допускающим состояниям (состояниям из F), отмечаются двойным кружком. Состояния, не принадлежащие F, изображаются простым (одинарным) кружком.

Таблица переходов представляет собой обычное табличное представление функции, подобной δ, которая двум значениям ставит в соответствие одно значение. Строки таблицы соответствуют состояниям, а столбцы — входным символам. На пересечении строки, соответствующей состоянию q, и столбца, соответствующего символу a, находится состояние $\delta(q, a)$. [19]

Определение 2.4. *Автомат называется полностью определенным или полным, если* $D_\delta = Q \times \Sigma$. *Иными словами, для любого начального состояния и любой входной последовательности (в пределах алфавита* Σ*) однозначно определена выходная последовательность.*

Определение 2.5. *Автомат называется частично определенным или частичным, если функция* δ *определена не для всех пар* $(q, a) \in Q \times \Sigma$.

В табличном задании таких автоматов на месте неопределенных состояний и выходных сигналов ставится прочерк. На графе автомата неопределенному состоянию можно поставить в соответствие дополнительную вершину.

Неопределенное состояние автомата называется граничным. Дальнейшее поведение автомата, попавшего в граничное состояние, не определяется. Выходная последовательность, содержащая неопределенные сигналы, называется частичной. Заметим, что модель частичного автомата является более абстрактной, чем модель полного автомата: она задает целое множество дискретных устройств. [20]

Под *композицией элементарных автоматов* в общем случае понимается следующее. Пусть заданы элементарные автоматы $A_1, A_2, \ldots, A_k$. Произведем объединение элементарных автоматов в систему совместно работающих автоматов. Введем в рассмотрение некоторое конечное множество узлов, называемых внешними выходными узлами. Эти узлы отличаются от узлов рассматриваемых элементарных автоматов, которые носят название внутренних. Композиция автомата состоит в том, что в полученной системе элементарных автоматов $A_1, A_2, \ldots, A_k$ и внешних узлов производится отождествление некоторых узлов (как внешних так и внутренних).

С точки зрения совместной работы системы автоматов смысл операции отождествления узлов состоит в том, что элементарный сигнал, попадающий на один из узлов, входящих в множество отождествляемых между собой узлов, попадает тем самым на все узлы этого множества. После проведенных отождествлений узлов система автоматов превращается в так называемую схему (сеть) автоматов. Будем считать, что автоматы, входящие в систему автоматов, работают совместно, если в каждый момент автоматного времени на все внешние входные узлы подается набор входных сигналов (структурный входной сигнал схемы) и со всех внешних выходных узлов снимается набор выходных сигналов (структурный выходной сигнал). Следует заметить, что при построении схемы автоматов должны выполняться условия корректности. [21]

2.3 Применения автоматной модели для описания процесса перегрузки реактора

Исходя из положений, представленных в предидущих разделах, представим реактор как композицию конечных детерминированных автоматов.

Пусть реактор состоит из k структурных элементов, каждый из которых можно представить как конечный детерминированный автомат. Таким образом $i-$й структурный элемент реактора описывается кортежем $A_i = \langle Q^i, \Sigma^i, \delta^i, q_0^i, F^i \rangle$.

Тогда реактор можно представить следующим кортежем $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, где

- $Q = \cup_{i=0}^{n} Q^i$;

- $\Sigma = \cup_{i=0}^{n} \Sigma^i$;

- $\delta = f(q, a)$;

- $q_0 = \cup_{i=0}^{n} q_0^i$;

- $F = \cap_{i=0}^{n} F^i$.

Рассмотрим данную модель на примере фиктивного реактора. Пусть реактор имеет следующую конфигурацию активной зоны (см. рис. 2.3). Ячейки А1, Б1, В1, Г1, Д1, Е1 предназначены для организации внутриреакторного хранилища отработавших ТВС, ячейки А2, Б2, В2, Г2, Д2, Е2 содержат сборки для организации боковой зоны воспроизводства, ячейки А3, Б3, В3, Г3, Д3, Е3 содержат сборки активной зоны реактора и ячейка А4 содержит механизм автоматического регулирования реактора.

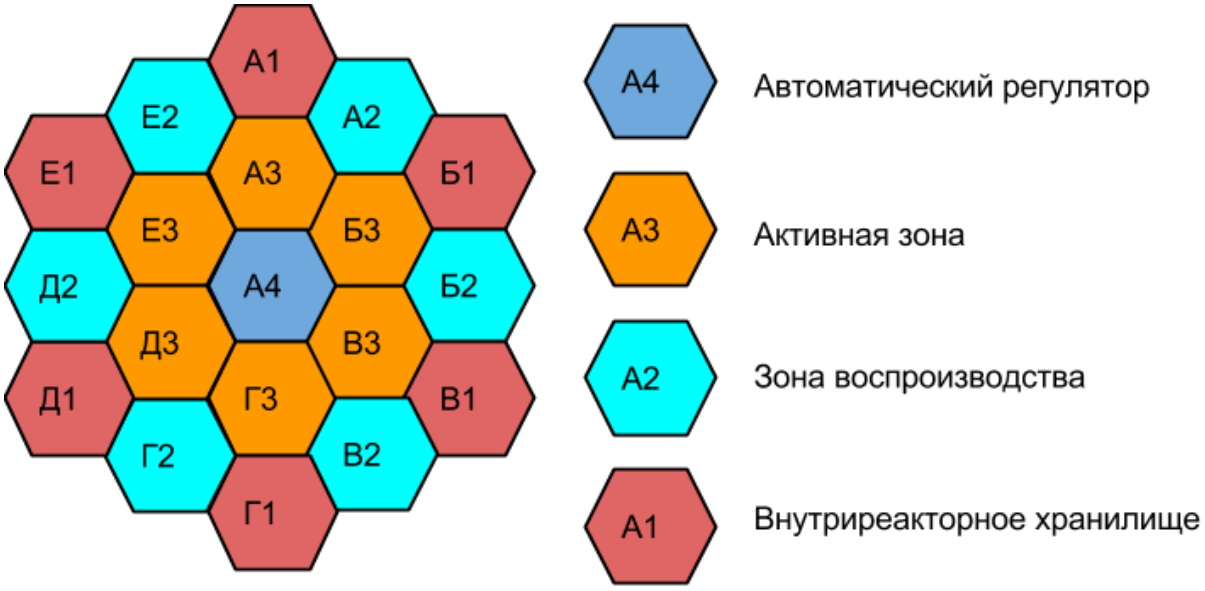

Рисунок 2.3: Конфигурация активной зоны фиктивного реактора: А1 – внутриреакторное хранилище, А2 – зона воспроизводства, А3 — активная зона, А4 — автоматический регулятор

Фиктивный реактор имеет механизм перегрузки, аналогичный описанному в **??**. Большая поворотная пробка имеет 6 положений, которые соответствуют букве в названии ячейки, малая поворотная пробка - 4 положения, которые соответствуют цифре в названии ячейки.

Перед началом планово-профилактического ремонта поворотные пробки находятся в положении А1, активная зона и зоны воспроизводства полностью заполнены, внутриреакторное хранилище пусто, частично или полностью заполнено. Некоторые ТВС активной зоны и зоны воспроизводства, а также все ТВС из внутриреакторного хранилищ помечены как требующие перезагрузки. После планово-профилактического ремонта поворотные пробки находятся в положении А1, активная зона и зоны воспроизводства полностью заполнены, внутриреакторное хранилище пусто, частично или полностью заполнено. Все ТВС активной зоны и

зоны воспроизводства, а также все ТВС из внутриреакторного хранилищ помечены как не требующие перезагрузки.

Разделим реактор на следующие структурные компоненты: тепловыделяющие сборки с обогащенным ураном, тепловыделяющие сборки с обедненным ураном, гнезда внутриреакторного хранилища, гнезда зоны воспроизводства, гнезда активной зоны, большая поворотная пробка, малая поворотная пробка, перегрузочный механизм. Рассмотрим отдельные структурные компоненты реактора как автоматы.

Тепловыделяющие сборки с обогащенным ураном:

- $Q = \{$В активную зону, Во внутриреакторное хранилище, Во внешнее хранилище, Обслуживание завершено$\}$;
- $\Sigma = \{$Обслужить$\}$;
- $\delta = f(q, a)$, где f соответствует таблице 2.1;
- $q_0 \in \{$В активную зону, Во внутриреакторное хранилище, Во внешнее хранилище, Обслуживание завершено$\}$;
- $F = \{$Обслуживание завершено$\}$.

Таблица 2.1: Таблица переходов автомата «Тепловыделяющая сборка с обогащенным ураном»

$q_i \diagdown \sigma_j$	Обслужить
В активную зону	Обслуживание завершено
Во внутриреакторное хранилище	Обслуживание завершено
Во внешнее хранилище	Обслуживание завершено
Обслуживание завершено	—

Тепловыделяющие сборки с обедненным ураном:

- $Q = \{$В зону воспроизводства, Во внешнее хранилище, Обслуживание завершено$\}$;
- $\Sigma = \{$Обслужить$\}$;
- $\delta = f(q, a)$, где f соответствует таблице 2.2;
- $q_0 \in \{$В зону воспроизводства, Во внешнее хранилище, Обслуживание завершено$\}$;
- $F = \{$Обслуживание завершено$\}$.

Гнезда внутриреакторного хранилища:

Таблица 2.2: Таблица переходов автомата «Тепловыделяющая сборка с обедненным ураном»

$q_i \diagdown \sigma_j$	Обслужить
В зону воспроизводства	Обслуживание завершено
Во внешнее хранилище	Обслуживание завершено
Обслуживание завершено	—

- $Q = A_{ef} \cup \varnothing$, где A_{ef} – множество автоматов «Тепловыделяющая сборка с обогащенным ураном»;
- $\Sigma = A_{ef} \cup \varnothing$, где A_{ef} – множество автоматов «Тепловыделяющая сборка с обогащенным ураном»;
- $\delta = f(q, a)$, где f соответствует таблице 2.3;
- $q_0 \in Q$;
- $F \in Q$.

Таблица 2.3: Таблица переходов автомата «Гнездо внутриреакторного хранилища»

$q_i \diagdown \sigma_j$	a_{ef}	$\varnothing$
a_{ef}	—	$\varnothing$
$\varnothing$	a_{ef}	—

Гнезда зоны воспроизводства:

- $Q = A_{uf} \cup \varnothing$, где A_{uf} – множество автоматов «Тепловыделяющая сборка с обедненным ураном»;
- $\Sigma = A_{uf} \cup \varnothing$, где A_{uf} – множество автоматов «Тепловыделяющая сборка с обедненным ураном»;
- $\delta = f(q, a)$, где f соответствует таблице 2.4;
- $q_0 \in A_{uf}$;
- $F \in A_{uf}$.

Гнезда активной зоны:

Таблица 2.4: Таблица переходов автомата «Гнездо зоны воспроизводства»

$q_i \diagdown \sigma_j$	a_{uf}	$\varnothing$
a_{uf}	—	$\varnothing$
$\varnothing$	a_{uf}	—

- $Q = A_{ef} \cup \varnothing$, где A_{ef} – множество автоматов «Тепловыделяющая сборка с обогащенным ураном»;
- $\Sigma = A_{ef} \cup \varnothing$, где A_{ef} – множество автоматов «Тепловыделяющая сборка с обогащенным ураном»;
- $\delta = f(q, a)$, где f соответствует таблице 2.5;
- $q_0 \in A_{ef}$;
- $F \in A_{ef}$.

Таблица 2.5: Таблица переходов автомата «Гнездо активной зоны»

$q_i \diagdown \sigma_j$	a_{ef}	$\varnothing$
a_{ef}	—	$\varnothing$
$\varnothing$	a_{ef}	—

Большая поворотная пробка:

- $Q = \{\text{А}, \text{Б}, \text{В}, \Gamma, \text{Д}, \text{Е}\}$;
- $\Sigma = \{\text{влево}, \text{вправо}\}$;
- $\delta = f(q, a)$, где f соответствует таблице 2.6;
- $q_0 = \text{А}$;
- $F = \text{А}$.

Малая поворотная пробка:

- $Q = \{1, 2, 3, 4\}$;
- $\Sigma = \{\langle \text{влево}, q_{bp} \rangle, \langle \text{вправо}, q_{bp} \rangle\}$, где q_{bp} – текущее состояние автомата «Большая поворотная пробка»;
- $\delta = f(q, a)$, где f соответствует таблице 2.7;

Таблица 2.6: Таблица переходов автомата «Большая поворотная пробка»

$q_i \diagdown \sigma_j$	влево	вправо
А	Е	Б
Б	А	В
В	Б	Г
Г	В	Д
Д	Г	Е
Е	Д	А

- $q_0 = 1$;
- $F = 1$.

Таблица 2.7: Таблица переходов автомата «Малая поворотная пробка»

$q_i \diagdown \sigma_j$	⟨влево, А⟩	⟨вправо, А⟩	⟨влево, $\overline{\text{А}}$⟩	⟨вправо, $\overline{\text{А}}$⟩
1	4	2	3	2
2	1	3	1	3
3	2	4	2	1
4	3	1	-	-

Перегрузочный механизм: Рассмотрим перегрузочный механизм как совокупность следующих автоматов: большая поворотная пробка, малая поворотная пробка, определитель типа гнезда, селектор гнезда, перегрузочный механизм.

Определитель типа гнезда:

- $Q = \{$Внутриреакторное хранилище, Зона воспроизводства, Активная зона, Автоматический регулятор$\}$;
- $\Sigma = \{1, 2, 3, 4\}$;
- $\delta = f(q, a)$, где f соответствует таблице 2.8;
- $q_0 =$ Внутриреакторное хранилище;
- $F =$ Внутриреакторное хранилище.

Селектор гнезда:

- $Q = Q_{\text{гнезда}}$;

Таблица 2.8: Таблица переходов автомата «Определитель типа гнезда»: ВХ – Внутриреакторное хранилище; ЗВ – Зона воспроизводства; АЗ – Активная зона; АР – Автоматический регулятор.

$q_i \diagdown \sigma_j$	1	2	3	4
ВХ	ВХ	ЗВ	АЗ	АР
ЗВ	ВХ	ЗВ	АЗ	АР
АЗ	ВХ	ЗВ	АЗ	АР
АР	ВХ	ЗВ	АЗ	АР

- $\Sigma = \langle Q_{\text{A1}}, \ldots, Q_{\text{E3}}, Q_{bp}, Q_{lp} \rangle$;
- $\delta = f(q, a)$, где f соответствует таблице 2.9;
- $q_0 = Q_{\text{A1}}$;
- $F = Q_{\text{A1}}$.

Таблица 2.9: Таблица переходов автомата «Селектор гнезда».

$q_i \diagdown \sigma_j$	$\langle q_{\text{A1}}, \ldots, q_{\text{E3}}, \text{A}, 1 \rangle$	$\cdots$	$\langle q_{\text{A1}}, \ldots, q_{\text{E3}}, \text{E}, 3 \rangle$
$\forall q \in Q$	q_{A1}	$\cdots$	q_{E3}

Перегрузочный механизм:

- $Q = A_{\text{ТВС}} \cup \varnothing$;
- $\Sigma = \langle Q_{\text{Селектора}}, \ldots, Q_{\text{Определителя}}, \{\text{извлечь}, \text{установить}, \text{во внешнее хранилище}, \text{из внешнего хранилища}\} \rangle$;
- $\delta = f(q, a)$, где f соответствует таблице 2.10;
- $q_0 = \varnothing$;
- $F = \varnothing$.

Рассмотрев отдельные автоматы, моделирующие поведение структурных элементов реактора, рассмотрим композицию этих автоматов (ТА-модель). Согласно описанию фиктивного реактора, приведенного выше (см. рис. 2.3), ТА-модель реактора состоит из $6-12$ ТА-моделей тепловыделяющих сборок с обогащенным ураном, 6 ТА-моделей тепловыделяющих сборок с обедненным ураном, 6 ТА-моделей гнезд активной зоны, 6 ТА-моделей гнезд зоны воспроизводства, 6 ТА-моделей гнезд внутриреакторного хранилища, ТА-модели большой поворотной

Таблица 2.10: Таблица переходов автомата «Перегрузочный механизм»: ВХ – Внутриреакторное хранилище; ВнешХ – Внешнее хранилище; ЗВ – Зона воспроизводства; АЗ – Активная зона; АР – Автоматический регулятор.

$\sigma_j \diagdown q_i$	$\varnothing$	В АЗ	Во ВХ	Во внешХ	В ЗВ
$\langle \varnothing, \text{АЗ}, \text{извлечь} \rangle$	—	—	—	—	—
$\langle \varnothing, \text{АЗ}, \text{установить} \rangle$	—	$\varnothing$	—	—	—
$\langle \varnothing, \text{АЗ}, \text{во внешнее хранилище} \rangle$	—	—	—	—	—
$\langle \varnothing, \text{АЗ}, \text{из внешнего хранилища} \rangle$	$A_{\text{ТВС}}$	—	—	—	—
...	...	...	...	...	...
$\langle \text{В ЗВ}, \text{ВХ}, \text{из внешнего хранилища} \rangle$	—	—	—	—	—

пробки, ТА-модели малой поворотной пробки, ТА-модели селектора гнезд, ТА-модели определителя типа гнезда и ТА-модели перегрузочного механизма.

Выходы ТА-моделей ТВС с обогащенным ураном являются входными символами для ТА-моделей гнезд активной зоны и ТА-моделей гнезд внутриреакторного хранилища. Выходы ТА-моделей ТВС с обедненным ураном являются входными символами для ТА-модели гнезд зоны воспроизводства. Выходы ТА-моделей гнезд реактора являются входными символами ТА-модели селектора гнезд. Выход ТА-модели большой поворотной пробки является входным символом для ТА-моделей малой поворотной пробки, селектора гнезд и определителя типа гнезд. Выход ТА-модели малой поворотной пробки является входным символом для ТА-модели селектора гнезд и определителя типа гнезд. Выходы ТА-моделей селектора гнезд и определителя типа гнезд являются входными символами для ТА-модели перегрузочного механизма. Композиционная схема ТА-модели фиктивного реактора представлена на рисунке 2.4.

Исходя из описанного выше представим модель фиктивного реактора в общем виде следующим образом:

- $Q = \langle Q_{\text{ТВС 1}}, \ldots, Q_{\text{ТВС 18}}, Q_{\text{А1}}, \ldots, Q_{\text{Е3}}, Q_{bp}, Q_{lp}, Q_{\text{перегрузочного механизма}} \rangle$;
- $\Sigma = \{$извлечь, установить, во внешнее хранилище, из внешнего хранилища, БПП влево, БПП вправо, МПП влево, МПП вправо$\}$;
- $\delta = \delta_{\text{ТВС 1}} \cdot \cdots\cdots \delta_{\text{ТВС 18}} \cdot \delta_{\text{А1}} \cdot \cdots\cdots \delta_{\text{Е3}} \cdot \delta_{bp} \cdot \delta_{lp} \cdot \delta_{\text{перегрузочного механизма}}$;
- $q_0 = \langle q_{\text{ТВС 1}}, \ldots, q_{\text{ТВС 18}}, q_{\text{А1}}, \ldots, q_{\text{Е3}}, q_{bp}, q_{lp}, q_{\text{перегрузочного механизма}} \rangle$;
- $F = \langle F_{\text{ТВС 1}}, \ldots, F_{\text{ТВС 18}}, F_{\text{А1}}, \ldots, F_{\text{Е3}}, F_{bp}, F_{lp}, F_{\text{перегрузочного механизма}} \rangle$.

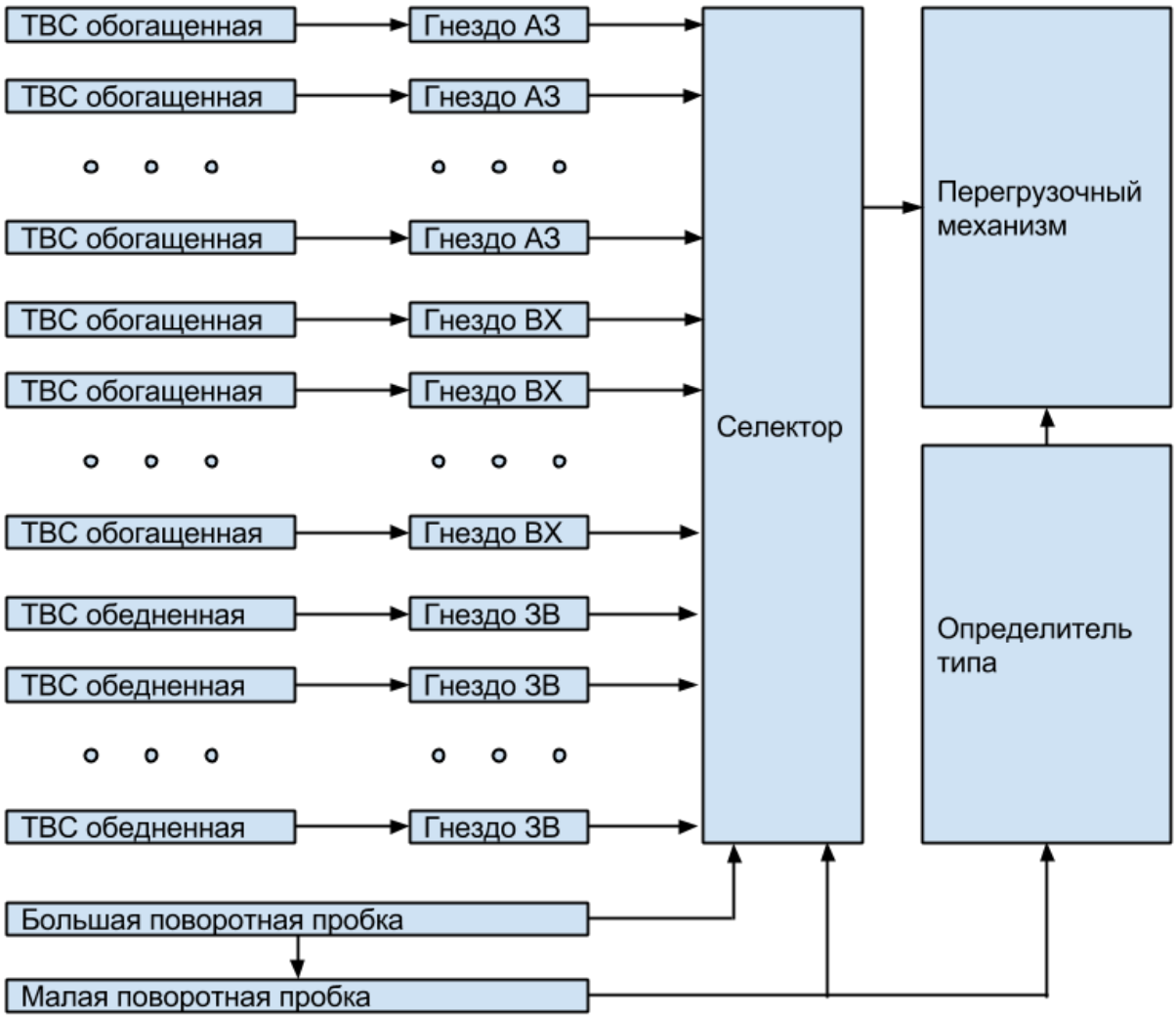

Рисунок 2.4: Композиционная схема автоматов модели фиктивного реактора: ТВС – тепловыделющая сборка, ВХ – Внутриреакторное хранилище; ЗВ – Зона воспроизводства; АЗ – Активная зона.

2.4 Выводы по главе

В данной главе сформулирована постановка задачи оптимизации технологического процесса. Предложен общий подход к решению задачи оптимизации технологического процесса. Определены следующие входные данные: описание конструкции реактора, данные для физического моделирования и функция оптимизации. В качестве выходных данных определн оптимальный порядок технологического процесса.

Рассмотрены основные положения теории автоматов. Введено понятие автоматной модели (ТА-модели). Рассмотрен подход к построению композиции ТА-моделей.

Показан пример построения ТА-модели фиктивного реактора.

Глава 3

Поиск оптимальной последовательности автоматной модели

3.1 Общий подход к поиску оптимальной последовательности

Описанная в предидущей главе модель реактора обладает следующими свойствами:

- состояние ТА-модели q_0 является моделью состояния реактора к началу планово-профилактического ремонта;
- состояния ТА-модели из множества F являются искомыми состояниями, в которых планово-профилактический ремонт будет завершен;
- входное слово для ТА-модели является последовательно выполняемыми действиями над реактором, которые приводят его в новое состояние.

Таким образом, задача построения порядка проведения планово-профилактического ремонта реактора сводится к поиску входных слов конечного автомата, которые приводя его в конечное состояние.

Описанный в разделе **??** метод описания детерминированных конечных автоматов в виде диаграммы переходов представляет собой ориентированный граф. Вершинами графа являются состояния автомата, а ребрами графа — символы входного алфавита автомата. Тогда становится возможным произвести поиск по графу из вершины q_0 в вершины из множества F, используя алгоритмы поиска пути в графе (будут рассмотрены далее).

Каждое действие, производимое над реактором имеет различные трудо- и энергозатраты, следовательно для построения оптимального порядка перезагрузки требуется учитывать эти затраты в модели. Установим в соответствие каждому входному символу КДА коэффициент затрат и введем его в ТА-модель. Для этого каждому ребру графа поставим в соответствие коэффициент затрат, т.е. сделаем граф взвешенным. Тогда оптимальным порядком

проведения планово-профилактического ремонта будет являться такая последовательность операций, которое приводит ТА-модель реактора из начального в конечное состояние с минимальной суммой энергозатрат.

3.2 Классические алгоритмы поиска пути в графе

Задача о кратчайшем пути — задача поиска самого короткого пути (цепи) между двумя точками (вершинами) на графе, в которой минимизируется сумма весов ребер, составляющих путь. Кратчайшая (простая) цепь часто называется геодезической. [22] Задача о кратчайшем пути является одной из важнейших классических задач теории графов. Сегодня известно множество алгоритмов для ее решения. У данной задачи существуют и другие названия: задача о минимальном пути или, в устаревшем варианте, задача о дилижансе.

Задача поиска кратчайшего пути на графе может быть определена для неориентированного, ориентированного или смешанного графа. Далее будет рассмотрена постановка задачи в самом простом виде для неориентированного графа. Для смешанного и ориентированного графа дополнительно должны учитываться направления ребер.

Граф представляет собой совокупность непустого множества вершин и ребер (наборов пар вершин). Две вершины на графе смежны, если они соединяются общим ребром. Путь в неориентированном графе представляет собой последовательность вершин $P = (v_1, v_2, \ldots, v_n) \in V \times V \times \cdots \times V$, таких что, v_i смежна с v_{i+1} для $1 \leqslant i < n$. Такой путь P называется путем длиной n из вершины v_1 в v_n (i указывает на номер вершины пути и не имеет никакого отношения к нумерации вершин на графе).

Пусть $e_{i,j}$ — ребро соединяющее две вершины: v_i и v_j. Дана весовая функция $f : E \to \mathbb{R}$, которая отображает ребра на их веса, значения которых выражаются действительными числами, и неориентированный граф G. Тогда кратчайшим путем из вершины v в вершину v' будет называться путь $P = (v_1, v_2, \ldots, v_n)$ (где $v_1 = v$ и $v_n = v'$), который имеет минимальное значение суммы $\sum_{i=1}^{n-1} f(e_{i,i+1})$. Если все ребра в графе имеют единичный вес, то задача сводится к определению наименьшего количества обходимых ребер. [23]

Существуют различные постановки задачи о кратчайшем пути:

- Задача о кратчайшем пути в заданный пункт назначения. Требуется найти кратчайший путь в заданную вершину назначения t, который начинается в каждой из вершин графа (кроме t). Поменяв направление каждого принадлежащего графу ребра, эту задачу можно свести к задаче о единой исходной вершине (в которой осуществляется поиск кратчайшего пути из заданной вершины во все остальные).

- Задача о кратчайшем пути между заданной парой вершин. Требуется найти кратчайший путь из заданной вершины u в заданную вершину v.

- Задача о кратчайшем пути между всеми парами вершин. Требуется найти кратчайший путь из каждой вершины u в каждую вершину v. Эту задачу тоже можно решить с по-

мощью алгоритма, предназначенного для решения задачи об одной исходной вершине, однако обычно она решается быстрее.

В различных постановках задачи, роль длины ребра могут играть не только сами длины, но и время, стоимость, расходы, объем затрачиваемых ресурсов (материальных, финансовых, топливно-энергетических и т. п.) или другие характеристики, связанные с прохождением каждого ребра. Таким образом, задача находит практическое применение в большом количестве областей (информатика, экономика, география и др.).

В связи с тем, что существует множество различных постановок данной задачи, есть наиболее популярные алгоритмы для решения задачи поиска кратчайшего пути на графе:

- Алгоритм Дейкстры находит кратчайший путь от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса.

- Алгоритм Беллмана – Форда находит кратчайшие пути от одной вершины графа до всех остальных во взвешенном графе. Вес ребер может быть отрицательным.

- Алгоритм поиска A* находит маршрут с наименьшей стоимостью от одной вершины (начальной) к другой (целевой, конечной), используя алгоритм поиска по первому наилучшему совпадению на графе.

- Алгоритм Флойда — Уоршелла находит кратчайшие пути между всеми вершинами взвешенного ориентированного графа.

- Алгоритм Джонсона находит кратчайшие пути между всеми парами вершин взвешенного ориентированного графа.

- Алгоритм Ли (волновой алгоритм) основан на методе поиска в ширину. Находит путь между вершинами s и t графа (s не совпадает с t), содержащий минимальное количество промежуточных вершин (ребер). Основное применение — трассировки электрических соединений на кристаллах микросхем и на печатных платах. Так же используется для поиска кратчайшего расстояния на карте в стратегических играх.

- Поиск кратчайшего пути на основе алгоритма Килдала.

Далее рассмотрим подробно алгоритм Дейкстры, алгоритм A* и волновой алгоритм.

3.2.1 Алгоритм Дейкстры

Алгоритм Дейкстры — алгоритм на графах, изобретённый нидерландским ученым Эдсгером Дейкстрой в 1959 году. Алгоритм Дейкстры решает задачу о кратчайших путях из одной вершины для взвешенного ориентированного графа $G = (V, E)$ с исходной вершиной s, в котором веса всех рёбер неотрицательны ($\omega(u, v) \geqslant 0$ для всех $(u, v) \in E$).

Формальное объяснение. В процессе работы алгоритма Дейкстры поддерживается множество $S \subseteq V$, состоящее из вершин v, для которых $\delta(s, v)$ уже найдено. Алгоритм выбирает вершину $u \in V$
S с наименьшим $d[u]$, добавляет u к множеству S и производит релаксацию всех рёбер, выходящих из u, после чего цикл повторяется. Вершины, не лежащие в S , хранятся в очереди Q с приоритетами, определяемыми значениями функции d. Предполагается, что граф задан с помощью списков смежных вершин.

Неформальное объяснение. Каждой вершине из V сопоставим метку — минимальное известное расстояние от этой вершины до a. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Инициализация. Метка самой вершины a полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от a до других вершин пока неизвестны. Все вершины графа помечаются как непосещенные.

Шаг алгоритма. Если все вершины посещены, алгоритм завершается. В противном случае из еще не посещенных вершин выбирается вершина u, имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, соединенные с вершиной u ребрами, назовем соседями этой вершины. Для каждого соседа рассмотрим новую длину пути, равную сумме текущей метки u и длины ребра, соединяющего u с этим соседом. Если полученная длина меньше метки соседа, заменим метку этой длиной. Рассмотрев всех соседей, пометим вершину u как посещенную и повторим шаг.

Время работы алгоритма Дейкстры. Сложность алгоритма Дейкстры зависит от способа нахождения вершины v, а также способа хранения множества непосещенных вершин и способа обновления меток. Обозначим через n количество вершин, а через m — количество ребер в графе G.

В простейшем случае, когда для поиска вершины с минимальным $d[v]$ просматривается все множество вершин, а для хранения величин d — массив, время работы алгоритма есть $O(n^2 + m)$. Основной цикл выполняется порядка n раз, в каждом из них на нахождение минимума тратится порядка n операций, плюс количество релаксаций (смен меток), которое не превосходит количества ребер в исходном графе.

Для разреженных графов (то есть таких, для которых m много меньше n^2) непосещенные вершины можно хранить в двоичной куче, а в качестве ключа использовать значения $d[i]$, тогда время извлечения вершины из U станет $\log n$, при том, что время модификации $d[i]$ возрастет до $\log n$. Так как цикл выполняется порядка n раз, а количество релаксаций не больше m, скорость работы такой реализации $O(n \log n + m \log n)$.

Если для хранения непосещенных вершин использовать фибоначчиеву кучу, для которой удаление происходит в среднем за $O(\log n)$, а уменьшение значения в среднем за $O(1)$, то время работы алгоритма составит $O(n \log n + m)$.

3.2.2 Пример работы алгоритма Дейкстры

Для примера возьмем такой ориентированный граф G (см. рис. 3.1):

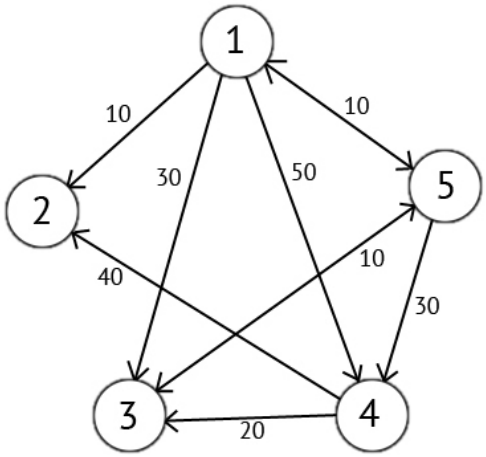

Рисунок 3.1: Пример алгоритма Дейкстры. Исходный граф

Возьмем в качестве источника вершину 1. Это значит что мы будем искать кратчайшие маршруты из вершины 1 в вершины 2, 3, 4 и 5. Данный алгоритм пошагово перебирает все вершины графа и назначает им метки, которые являются известным минимальным расстоянием от вершины источника до конкретной вершины. Рассмотрим этот алгоритм на примере.

Присвоим 1-й вершине метку равную 0, потому как эта вершина — источник. Остальным вершинам присвоим метки равные бесконечности (см. рис. 3.2).

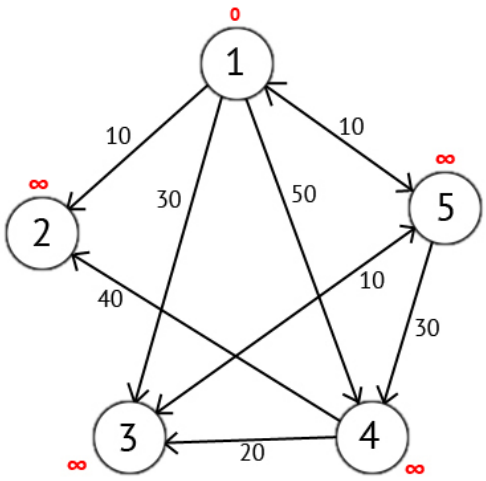

Рисунок 3.2: Пример алгоритма Дейкстры. Инициализация

Далее выберем такую вершину W, которая имеет минимальную метку (сейчас это вершина 1) и рассмотрим все вершины в которые из вершины W есть путь, не содержащий вершин посредников. Каждой из рассмотренных вершин назначим метку равную сумме метки W

и длинны пути из W в рассматриваемую вершину, но только в том случае, если полученная сумма будет меньше предыдущего значения метки. Если же сумма не будет меньше, то оставляем предыдущую метку без изменений (см. рис. 3.3).

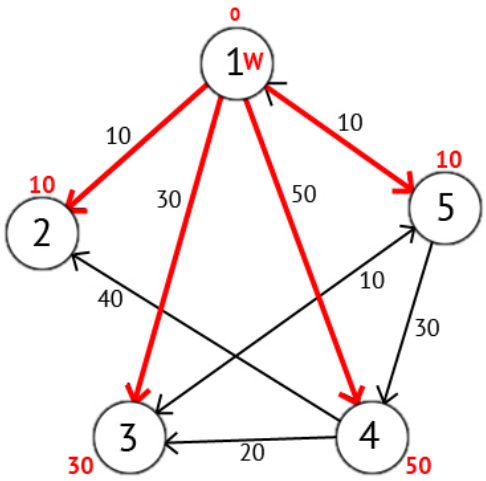

Рисунок 3.3: Пример алгоритма Дейкстры. Первый шаг алгоритма

После того как мы рассмотрели все вершины, в которые есть прямой путь из W, вершину W мы отмечаем как посещённую, и выбираем из ещё не посещенных такую, которая имеет минимальное значение метки, она и будет следующей вершиной W. В данном случае это вершина 2 или 5. Если есть несколько вершин с одинаковыми метками, то не имеет значения какую из них мы выберем как W.

Мы выберем вершину 2. Но из нее нет ни одного исходящего пути, поэтому мы сразу отмечаем эту вершину как посещенную и переходим к следующей вершине с минимальной меткой. На этот раз только вершина 5 имеет минимальную метку. Рассмотрим все вершины в которые есть прямые пути из 5, но которые ещё не помечены как посещенные. Снова находим сумму метки вершины W и веса ребра из W в текущую вершину, и если эта сумма будет меньше предыдущей метки, то заменяем значение метки на полученную сумму (см. рис. 3.4).

Исходя из картинки мы можем увидеть, что метки 3-ей и 4-ой вершин стали меньше, то есть был найден более короткий маршрут в эти вершины из вершины источника. Далее отмечаем 5-ю вершину как посещенную и выбираем следующую вершину, которая имеет минимальную метку. Повторяем все перечисленные выше действия до тех пор, пока есть непосещенные вершины.

Выполнив все действия получим такой результат (см. рис. 3.5)

Также есть вектор P, исходя из которого можно построить кратчайшие маршруты. По количеству элементов этот вектор равен количеству вершин в графе. Каждый элемент содержит последнюю промежуточную вершину на кратчайшем пути между вершиной-источником и конечной вершиной. В начале алгоритма все элементы вектора P равны вершине источнику (в нашем случае $P = \{1, 1, 1, 1, 1\}$). Далее на этапе пересчета значения метки для рассмат-

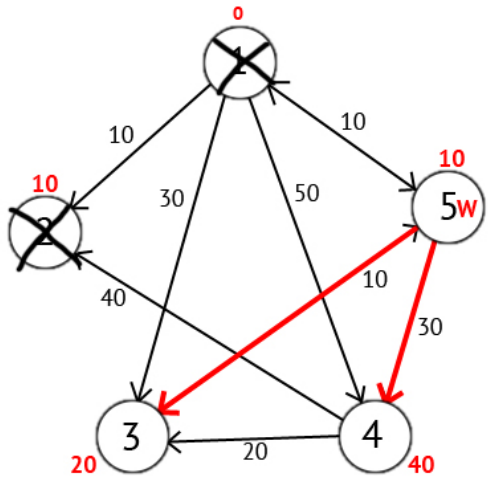

Рисунок 3.4: Пример алгоритма Дейкстры. Второй шаг алгоритма

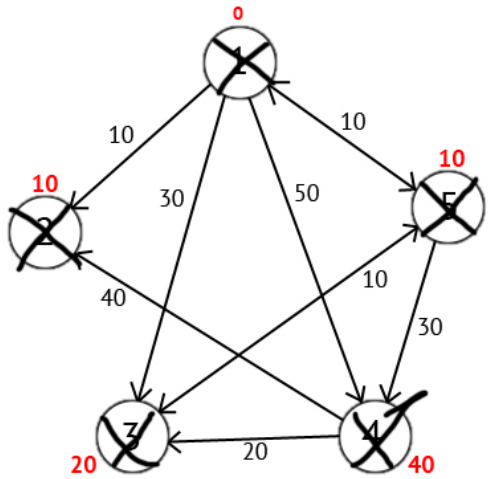

Рисунок 3.5: Пример алгоритма Дейкстры. Конец работы алгоритма

риваемой вершины, в случае если метка рассматриваемой вершины меняется на меньшую, в массив P мы записываем значение текущей вершины W. Например: у 3-ей вершины была метка со значением «30», при $W = 1$. Далее при $W = 5$, метка 3-ей вершины изменилась на «20», следовательно, мы запишем значение в вектор P — $P[3] = 5$. Также при $W = 5$ изменилось значение метки у 4-й вершины (было «50», стало «40»), значит нужно присвоить 4-му элементу вектора P значение W — $P[4] = 5$. В результате получим вектор $P = \{1, 1, 5, 5, 1\}$.

Зная что в каждом элементе вектора P записана последняя промежуточная вершина на пути между источником и конечной вершиной, мы можем получить и сам кратчайший маршрут. [24]

3.2.3 Алгоритм A*

Алгоритм поиска A* — алгоритм поиска по первому наилучшему совпадению на графе, который находит маршрут с наименьшей стоимостью от одной вершины (начальной) к другой (целевой, конечной).

Порядок обхода вершин определяется эвристической функцией «расстояние + стоимость» (обычно обозначаемой как $f(x)$). Эта функция — сумма двух других: функции стоимости достижения рассматриваемой вершины (x) из начальной (обычно обозначается как $g(x)$ и может быть как эвристической, так и нет) и эвристической оценкой расстояния от рассматриваемой вершины к конечной (обозначается как $h(x)$). Функция $h(x)$ должна быть допустимой эвристической оценкой, то есть не должна переоценивать расстояния к целевой вершине. Например, для задачи маршрутизации $h(x)$ может представлять собой расстояние до цели по прямой линии, так как это физически наименьшее возможное расстояние между двумя точками.

A* пошагово просматривает все пути, ведущие от начальной вершины в конечную, пока не найдёт минимальный. Как и все информированные алгоритмы поиска, он просматривает сначала те маршруты, которые «кажутся» ведущими к цели. От жадного алгоритма (который тоже является алгоритмом поиска по первому лучшему совпадению) его отличает то, что при выборе вершины он учитывает, помимо прочего, весь пройденный до неё путь (составляющая $g(x)$ — это стоимость пути от начальной вершины, а не от предыдущей, как в жадном алгоритме). В начале работы просматриваются узлы, смежные с начальным; выбирается тот из них, который имеет минимальное значение $f(x)$, после чего этот узел раскрывается. На каждом этапе алгоритм оперирует с множеством путей из начальной точки до всех ещё не раскрытых (листовых) вершин графа («множеством частных решений»), которое размещается в очереди с приоритетом. Приоритет пути определяется по значению $f(x) = g(x) + h(x)$. Алгоритм продолжает свою работу до тех пор, пока значение $f(x)$ целевой вершины не окажется меньшим, чем любое значение в очереди (либо пока всё дерево не будет просмотрено). Из множественных решений выбирается решение с наименьшей стоимостью.

Как и алгоритм поиска в ширину, A* является полным в том смысле, что он всегда находит решение, если таковое существует. Если эвристическая функция h допустима, то есть никогда не переоценивает действительную минимальную стоимость достижения цели, то A* сам является допустимым (или оптимальным), также при условии, что мы не отсекаем пройденные вершины. Если же мы это делаем, то для оптимальности алгоритма требуется, чтобы $h(x)$ была ещё и монотонной, или преемственной эвристикой. Свойство монотонности означает, что если существуют пути $A - B - C$ и $A - C$ (не обязательно через B), то оценка стоимости пути от A до C должна быть меньше либо равна сумме оценок путей $A-B$ и $B-C$. (Монотонность также известна как неравенство треугольника: одна сторона треугольника не может быть длиннее, чем сумма двух других сторон.) Математически, для всех путей x, y

(где y – потомок x) выполняется:

$$g(x) + h(x) \leq g(y) + h(y). \tag{3.1}$$

A* также оптимально эффективен для заданной эвристики h. Это значит, что любой другой алгоритм исследует не меньше узлов, чем A* (за исключением случаев, когда существует несколько частных решений с одинаковой эвристикой, точно соответствующей стоимости оптимального пути). В то время как A* оптимален для «случайно» заданных графов, нет гарантии, что он сделает свою работу лучше, чем более простые, но и более информированные относительно проблемной области алгоритмы. Например, в неком лабиринте может потребоваться сначала идти по направлению от выхода, и только потом повернуть назад. В этом случае обследование вначале тех вершин, которые расположены ближе к выходу (по прямой дистанции), будет потерей времени. [25]

3.2.4 Пример работы алгоритма A*

Представим, что у нас есть поле, разделенное на сетку с ячейками, на котором отмечены начальная (зеленая) и конечная (красная) ячейки. Каждая ячейка имеет состояние (открыта и закрыта), характеризующее её проходимость. В примере — открытые ячейки обведены точками, а закрытые пунктиром. Принцип алгоритма состоит в том, что он постепенно просматривает все окружающие текущую позицию ячейки и выбирает ячейку с наименьшим «весом». Стартовая ячейка по умолчанию является первой и единственной «открытой» ячейкой, с которой мы и начинаем поиск пути (см. рис. 3.6).

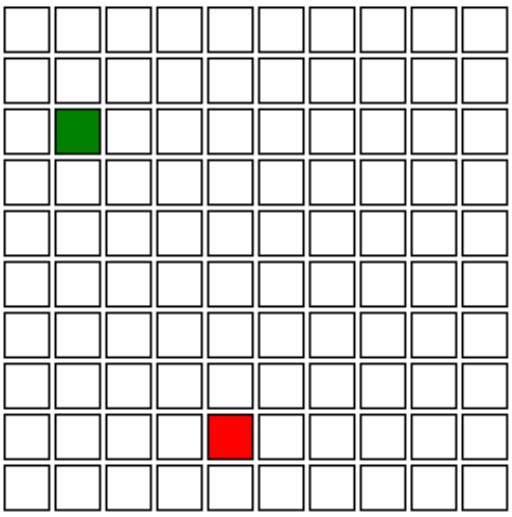

Рисунок 3.6: Пример алгоритма A*. Исходное поле

Теперь мы можем начать поиск пути. Ищем у текущей ячейки все «открытые» соседние ячейки и добавляем их в «открытый» список, а текущую ячейку в «закрытый» (см. рис. 3.7).

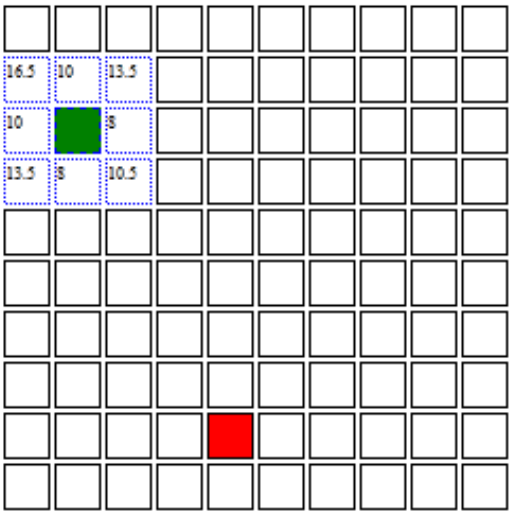

Рисунок 3.7: Пример алгоритма A*. Первый шаг алгоритма

Нужно переместиться в «открытую» соседнюю ячейку, выбрав ту, у которой стоимость (F) минимальна. В нашем примере стоимость (F) рассчитывается, как «прямое» расстояние до конечно точки, с учетом того, что ячейки по диагонали имеют вес в 1,5 раза больше, чем ортогональные (не по диагонали). Стоимость ячеек для наглядности указана внутри (см. рис. 3.8).

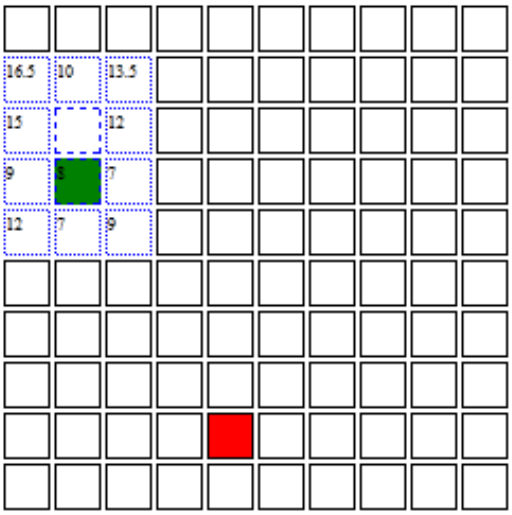

Рисунок 3.8: Пример алгоритма A*. Второй шаг алгоритма

Продолжаем повторение действий:

- Выставляем текущей ячейке статус «закрытой»;
- Ищем «открытые» соседние ячейки;
- Рассчитываем стоимость (F);
- Перемещаемся в ячейку с минимальной стоимостью.

В результате мы дойдем до конечной ячейки самым коротким путем (см. рис. 3.9). [26]

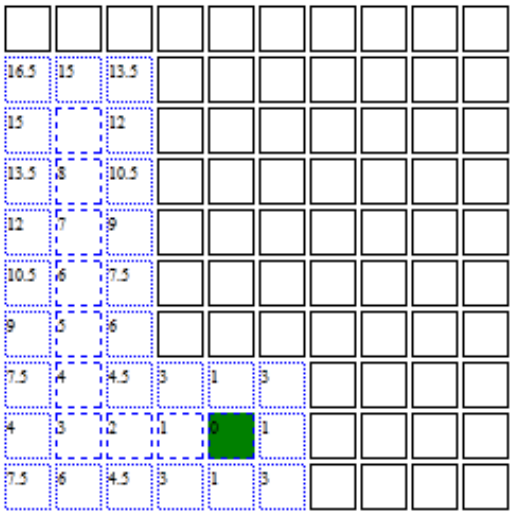

Рисунок 3.9: Пример алгоритма A*. Конец работы алгоритма

3.2.5 Волновой алгоритм

Алгоритм волновой трассировки (волновой алгоритм, алгоритм Ли) — алгоритм поиска кратчайшего пути на планарном графе. Принадлежит к алгоритмам, основанным на методах поиска в ширину.

В основном используется при компьютерной трассировке (разводке) печатных плат, соединительных проводников на поверхности микросхем. Другое применение волнового алгоритма — поиск кратчайшего расстояния на карте в компьютерных стратегических играх.

Волновой алгоритм в контексте поиска пути в лабиринте был предложен Э.Ф. Муром. Ли независимо открыл этот же алгоритм при формализации алгоритмов трассировки печатных плат в 1961 году.

Описание алгоритма. Алгоритм работает на «дискретном рабочем поле» (ДРП), представляющем собой ограниченную замкнутой линией фигуру, не обязательно прямоугольную, разбитую на прямоугольные ячейки, в частном случае — квадратные. Множество всех ячеек ДРП разбивается на подмножества: «проходимые» (свободные), т. е при поиске пути их можно проходить, «непроходимы» (препятствия), путь через эту ячейку запрещен, стартовая ячейка (источник) и финишная (приемник). Назначение стартовой и финишной ячеек условно, достаточно — указание пары ячеек, между которыми нужно найти кратчайший путь.

Алгоритм предназначен для поиска кратчайшего пути от стартовой ячейки к конечной ячейке, если это возможно, либо, при отсутствии пути выдать сообщение о непроходимости.

Работа алгоритма включает в себя три этапа: «инициализацию», «распространение волны» и «восстановление пути».

Во время инициализации строится образ множества ячеек обрабатываемого поля, каждой ячейке приписываются атрибуты проходимости/непроходимости, запоминаются стартовая и финишная ячейки.

Далее, от стартовой ячейки порождается шаг в соседнюю ячейку, при этом проверяется, проходима ли она, и не принадлежит ли ранее меченной в пути ячейке.

Соседние ячейки принято классифицировать двояко: в смысле окрестности Мура и окрестности фон Неймана, отличающийся тем, что в окрестности фон Неймана соседними ячейками считаются только 4 ячейки по вертикали и горизонтали, в окрестности Мура — все 8 ячеек, включая диагональные.

При выполнении условий проходимости и непринадлежности её к ранее помеченным в пути ячейкам, в атрибут ячейки записывается число, равное количеству шагов от стартовой ячейки, от стартовой ячейки на первом шаге это будет 1. Каждая ячейка, меченая числом шагов от стартовой ячейки становится стартовой и из неё порождаются очередные шаги в соседние ячейки. Очевидно, что при таком переборе будет найден путь от начальной ячейки к конечной, либо очередной шаг из любой порождённой в пути ячейки будет невозможен.

Восстановление кратчайшего пути происходит в обратном направлении: при выборе ячейки от финишной ячейки к стартовой на каждом шаге выбирается ячейка, имеющая атрибут расстояния от стартовой на единицу меньше текущей ячейки. Очевидно, что таким образом находится кратчайший путь между парой заданных ячеек.

Вычислительная сложность волнового алгоритма близка к $O(N^2)$. В реальных задачах, например при трассировке печатных плат, приложение пути (трассы) выполняется многократно, что влечет существенные временные затраты. [27]

3.2.6 Пример работы волнового алгоритма

ДРП — это прямоугольник, разбитый на квадратные ячейки одинакового размера. Ячейки ДРП подразделяются на свободные, препятствия, источники и приемники. На рис. 3.10 свободные ячейки имеют светло-зеленый цвет, а препятствия — светло-коричневый. Источник залит синим цветом, а приемник — черным. Путь может быть проложен только по свободным ячейкам.

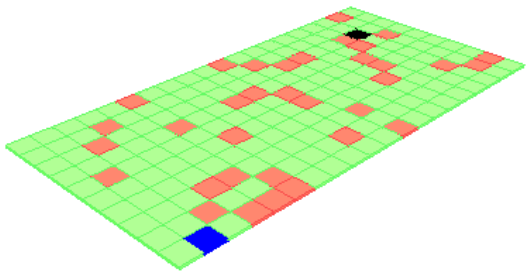

Рисунок 3.10: Пример волнового алгоритма. Исходное поле

Рассматривается алгоритм построения ортогонального пути. Алгоритм состоит из двух частей. В первой от источника к приемнику распространяется волна. Во второй выполняется обратный ход, в процессе которого из ячеек волны формируется путь. Волна, идущая от источника к приемнику, на каждом шаге первой части алгоритма пополняется свободными

ячейками ДРП, которые, во-первых, еще не принадлежат волне, и, во-вторых, являются 4-соседями ячеек, попавших в волну на предыдущем шаге. Процесс распространения волны иллюстрируют рис. 3.11 – 3.14.

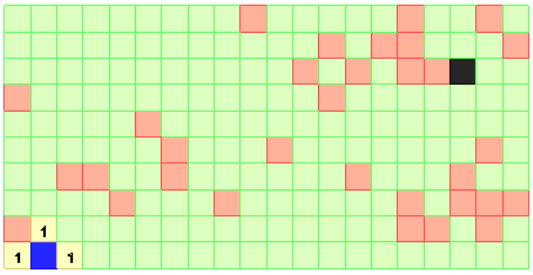

Рисунок 3.11: Пример волнового алгоритма. Первый шаг распространения волны

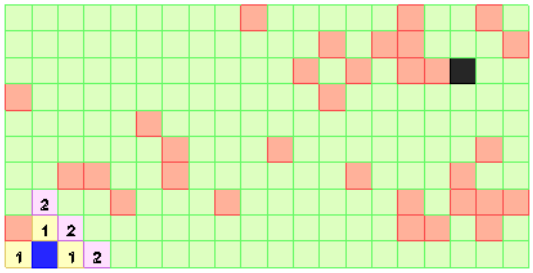

Рисунок 3.12: Пример волнового алгоритма. Второй шаг распространения волны

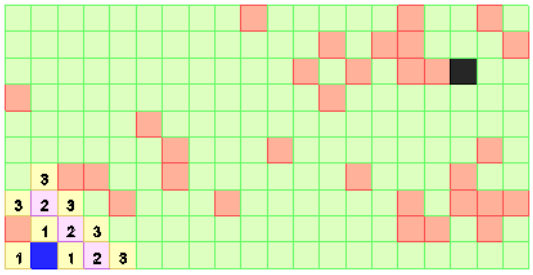

Рисунок 3.13: Пример волнового алгоритма. Третий шаг распространения волны

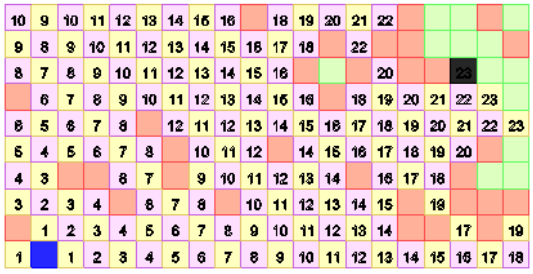

Рисунок 3.14: Пример волнового алгоритма. Последний шаг распространения волны

В примере волна достигла ячейку-приемник за 23 шага. При обратном ходе в путь включается по одной ячейке каждого шага распространения волны. При выборе из двух ячеек приоритет имеет ячейка, обеспечивающая горизонтальное продвижение, что приводит к пути, показанному на рис. 3.15.

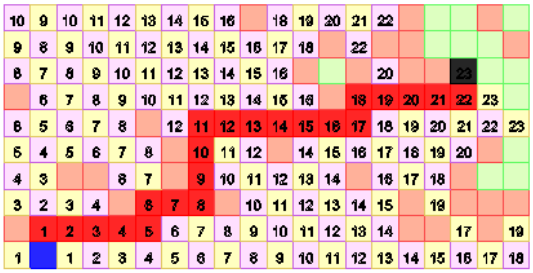

Рисунок 3.15: Пример волнового алгоритма. Обратный ход: формирование пути

3.3 Сравнительный анализ алгоритмов

Многообразие алгоритмов поиска пути обусловлено многообразием их применений, для каждого из которых эффективнее работает определенный алгоритм поиска пути.

- Алгоритм Дейкстры приоритетен в случаях поиска пути до всех точек области поиска, а также в случае отсутствия сколь либо эффективной эвристической функции оценки расстояния между элементами области поиска.

- Волновой алгоритм эффективен, если область поиска имеет неравномерную проходимость, что затрудняет эвристические вычисления для A*.

- Алгоритм A* эффективен при одиночном поиске пути между двумя точками, если возможно эффективно эвристически получать примерную дистанцию между элементами области поиска.

- Навигационная сетка с использованием алгоритма A* эффективна при создании AI, в чьи задачи входит не только поиск пути. Также этот метод эффективен при необходимости построить реалистичную сглаженную траекторию движения между двумя точками. Данный метод предполагает наличие детальной информации об области поиска или возможности получения такой информации.

- Эвристические алгоритмы поиска пути применимы и оптимальны, если необходим максимально простой алгоритм, при этом область поиска достаточно проста, а применения алгоритма допускают неточность полученного пути.

Таким образом, для построения оптимального порядка планово-профилактического ремонта реактора оптимальным будет использование алгоритма A* к графу ТА-модели реактора. Особым преимуществом алгоритма A* является наличие эвристической функции. Это позволяет проводить различные виды оптимизации ППР, изменяя только используемую эвристическую функцию.

3.4 Нейросетевой подход

Одной из задач, которую можно решать над автоматными моделями – это поиск кратчайших последовательностей входных символов (событий) из начального состояния в одно из конечных состояний. Так как одним из возможных представлений функции переходов является диаграмма состояний или граф переходов, то задача поиска оптимальной последовательности событий сводится к поиску кратчайшего расстояния от вершины начального состояния до вершины одного из конечных состояний.

Есть множество эффективных алгоритмов поиска пути в графе, однако, сложность алгоритмов зависит от количества вершин и ребер графа. Также задача усложняется необходимостью проверки каждой конечной вершины. В связи с этим ставится задача построения функции $f : Q \to X$, выбирающей следующий входной символ таким образом, чтобы за минимальное количество символов автомат достиг одного из конечных состояний.

Для построения данной функции создадим искусственную нейронную сеть прямого распространения, которая принимает состояние автомата (представленное в One-Hot кодировке), а на выходе выдает вектор размерности $|X|$, представляющий меру близости до конечного состояния.

Рассмотрим процедуру обучения данной нейронной сети.

Инициализация. Конечный автомат A переводится в состояние q_0. Искусственная нейронная сеть N инициализируется случайными малыми весами. Список D пуст. Стек S пуст.

Шаг 1. Искусственная нейронная сеть N по текущему состоянию q_i вычисляет меру близости. По мере близости выбирается следующий входной символ x_i. При $(q_i, x_i) \in D$, x_i заменяется на случайно выбранный другой элемент множества X. Кортеж (q_i, x_i) добавляется в стек S. Автомат A получает на вход символ x_i и переходит в следующее состояние q_{i+1}.

Шаг 2. Если $q_{i+1} \in F$ переходим к шагу 3, в противном случае – переходим к шагу 1.

Шаг 3. Из стека S по очереди извлекаются кортежи (q_i, x_i), дополнятся порядковым номером в стеке записываются в список D. Если $(q_i, x_i,\ n_i)$ уже был записан в списке D, то номер обновляется минимальным значением.

Шаг 4. Нейронная сеть N обучается методом обратного распространения ошибки. Для каждого q_i формируется вектор размерности $|X|$. Если для $(q_i, x_j,\ n_j)$, есть значения в D, то j-ый компонент вектора будет равен $1 - \frac{n_j}{\sum_{(q_i, x_k) \in D} n_k}$. В противном случае – j-ый компонент вектора равен 0.

Шаг 5. При необходимости продолжить обучение. Конечный автомат A переводится в состояние q_0. Переходим к шагу 1.

Данный алгоритм был реализован с помощью фреймворка машинного обучения TensorFlow. Результаты тестирования на различных случайно сгенерированных графах (количество вершин от 10 до 10000) подтверждают работоспособность данного подхода [12].

3.5 Выводы по главе

В данной главе рассмотрен подход к поиску оптимальной последовательности в автоматной модели. Показана возможность представления детерминированного конечного автомата в виде направленного графа (диаграммы переходов). Таким образом, задача поиска оптимальной последовательности автоматной модели сведена к задачи посика кратчайшего пути в графе.

Рассмотрены классические алгоритмы поиска пути в графе: алгоритм Дейкстры, алгоритм A* и волновой алгоритм. Для каждого из алгоритмов приведен пример. Из класических алгоритмов наиболее подходящим является алгоритм A*, применение которого рекомендуется для небольших и средних ТА-моделей.

Для больших моделей предложен нейросетевой алгоритм для поиска последовательности.

Глава 4

Аппроксимация эмерджентных свойств автоматной модели

4.1 Проблема потери интегративных свойств в автоматной модели

Как пример, запас реактивности реактора [28]

4.2 Нейросетевая аппроксимация запаса реактивности

4.2.1 Постановка задачи

В г. Обнинске Калужской области на базе филиала АО «НИФХИ им. Л.Я.Карпова» с 1964 г. находится в эксплуатации экспериментальная ядерная установка ВВР-ц – гетерогенный водо-водяной исследовательский реактор, специализированный для проведения широкого круга исследовательских работ в области радиационной химии, структурных и материаловедческих исследований, активационного анализа, нейтронного легирования полупроводников и т.д. [29]. С 1980 г. на базе реактора действует производство радионуклидов медицинского назначения, а также радиофармпрепаратов на их основе. В связи с успешностью развития данного направления и удобным географическим положением в 1986 г. было принято решение о реконструкции реактора [30].

В связи с необходимостью улучшения параметров реактора и повышения эффективности наработки радионуклидов (^{99}Mo, ^{131}I и др.) в 2011 г. была проведена работа по созданию прецизионной нейтронно-физической расчетной модели активной зоны, отражателя реактора и органов СУЗ. При моделировании в полном объеме учитывается геометрия всех твэлов (топливо, оболочка, водяной зазор с соответствующими температурами), изменение изотопного состава топлива в зависимости от выгорания, геометрия и состав органов СУЗ, отражателей,

экспериментальных каналов, конструкций. Полученная прецизионная модель была верифицирована для расчета запаса реактивности реактора [30].

Прецизионная модель основана на методе Монте-Карло, что позволяет добиться большой точности моделирования физических процессов активной зоны реактора. Однако данный подход требует большого количества машинного времени для проведения вычислительных экспериментов, например, расчет запаса реактивности для одной конфигурации активной зоны занимает около восьми часов.

Кампания реактора ВВР-ц составляет 100 часов в неделю с последующей остановкой для расхолаживания, перегрузки топлива, мишеней и других технологических операций [31]. При такой довольно короткой кампании использование прецизионной модели для проведения оперативных расчетов довольно затруднительно: при полной утилизации одной вычислительной системы можно сделать не более 12 расчетов в течение одной кампании. Таким образом, появляется задача по созданию программного комплекса для оценки запаса реактивности реактора ВВР-ц. Данный программный комплекс должен помочь научно-исследовательскому персоналу реактора в проведении предварительных расчетов запаса реактивности. К программному обеспечению предъявляется требование по кратному увеличению производительности расчета при сохранении достаточного уровня точности. Основная задача программного комплекса сводится к аппроксимации запаса реактивности реактора в зависимости от выгорания топлива и положения органов СУЗ.

4.2.2 Применение искусственных нейронных сетей для аппроксимации запаса рективности

В общем виде искусственными нейронными сетями (ИНС) называется подход к построению вычислительных алгоритмов и устройств, основанный на подобии биологических нейронов [32]. В рамках данной работы будем рассматривать искусственные нейронные сети как семейство алгоритмов для обработки информации.

Искусственный нейрон (формальный нейрон) – это элементарная вычислительная ячейка искусственной нейронной сети. Каждый искусственный нейрон получает вектор входных сигналов $\vec{x} = (x_0, x_1, \ldots, x_n)$, для которого вычисляется взвешенная сумма. Затем от этой взвешенной суммы вычисляется значение функции активации:

$$f(\sum_{i=0}^{n} \omega_i x_i + b) \tag{4.1}$$

где $\vec{\omega}$ – вектор весов, b – смещение.

Множество искусственных нейронов, получающих на вход единый вектор входных сигналов, называется полносвязным нейронным слоем. Последовательность нейронных слоев, в которой вектор выходных сигналов предыдущего слоя является вводным вектором последующего слоя, называется многослойным персептроном.

Все веса в многослойном персептроне инициализируются случайными малыми значениями. В такой конфигурации персептрон производит шум в ответ на любой входной вектор. Для настройки персептрона для выполнения заданной функции производится итеративный процесс обучения. Процесс обучения состоит в последовательном предъявлении на вход нейронной сети вектора из обучающего набора данных, получении результата на выходе нейронно сети, сравнении полученного выхода с ожидаемым выходом и корректировки весов для уменьшения полученной разницы. Одним из наиболее часто используемых алгоритмов для обучения является градиентный спуск [33].

Обоснуем возможность использования искусственной нейронной сети для построения аппроксимации. Для построения аппроксимации применима обобщенная аппроксимационная теорема. В соответствии с этой теоремой можно получить сколь угодно точное приближение любой непрерывной функции многих переменных, используя операции сложения и умножения на число, суперпозицию функций, линейные функции, а также одну произвольную непрерывную нелинейную функцию одной переменной [34]. Поскольку указанные операции полностью реализуются искусственной нейронной сетью с одним нелинейным формальным нейроном, то допустимо применять искусственную нейронную сеть для построения требуемой аппроксимации.

Для подтверждения возможности аппроксимации запаса реактивности с помощью ИНС было проведено два вычислительных эксперимента. В первом эксперименте искусственная нейронная сеть была обучена на данных, рассчитанных с помощью прецизионной модели. Во втором эксперименте искусственна нейронная сеть была обучена на реальных данных кампаний реактора ВВР-ц.

4.3 Аппроксимация модельных данных

4.3.1 Данные

Для проведения эксперимента по аппроксимации модельных данных искусственной нейронной сетью был построен набор данных. С помощью прецизионной модели были проведены вычислительные эксперименты для 34-х различных конфигураций реактора (выгорание ТВС, положения СУЗ) и для каждой конфигурации получено значение запаса реактивности. Из полученного набора данных была сформирована обучающая выборка (25 конфигураций) и тестовая выборка (9 конфигураций). Для конечной верификации использовались все 34 конфигурации активной зоны.

4.3.2 Архитектура сети

Для проведения эксперимента была создана трехслойная искусственная нейронная сеть. Входной слой состоит из 50-ти формальных нейронов с функцией активации ReLu:

$$ReLu(x) = \begin{cases} 0, & \text{при } x < 0; \\ x, & \text{при } x \geq 0. \end{cases} \tag{4.2}$$

Скрытый слой состоит из 10-ти формальных нейронов с функцией активации ReLu (4.2). Выходной слой состоит из одного формального нейрона с логистической функцией активации:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.3}$$

Данная искусственная нейронная сеть была реализована с помощью библиотеки машинного обучения TensorFlow [35–37]. На рис. 4.1 изображен граф вычислений, сформированный TensorFlow для обуения искуственной нейронной сети.

4.3.3 Процесс обучения

Для обучения искусственной нейронной сети было проведено 50000 эпох обучения. Проводилось обучение методом обратного распространения ошибки по среднеквадратичной ошибке на обучающей выборке. Через каждые 100 эпох проводилась оценка среднеквадратичной ошибки на тестовой выборке. В течение всего процесса обучения ошибка сходилась к нулю без расхождения.

На рис. 4.2 показан график зависимости среднеквадратичной ошибки от номера итерации вычислительного графа.

4.3.4 Верификация

После завершения процесса обучения была проведена верификация ИНС на модельных данных. На рис.4.3 приведены полные результаты сравнения прецизионных расчетов запаса реактивности с результатами работы ИНС. Обобщенные результаты верификации:

- средняя абсолютная ошибка аппроксимации – $0,0405$;
- максимальная абсолютная ошибка аппроксимации – $0,1029$;
- средняя относительная ошибка аппроксимации – $1,21\%$;
- максимальная относительная ошибка аппроксимации – $3,13\%$.

Среднее время расчета запаса реактивности с помощью искусственной нейронной сети составляет 100 мс [4].

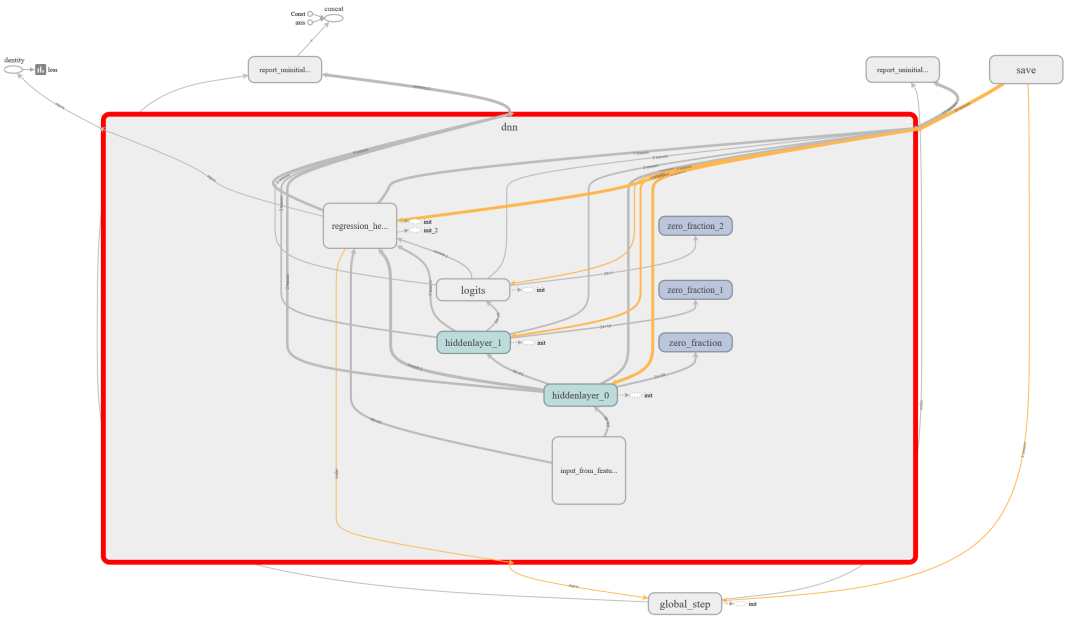

Рисунок 4.1: Граф вычислений TensorFlow для обучения искуственной нейронной сети

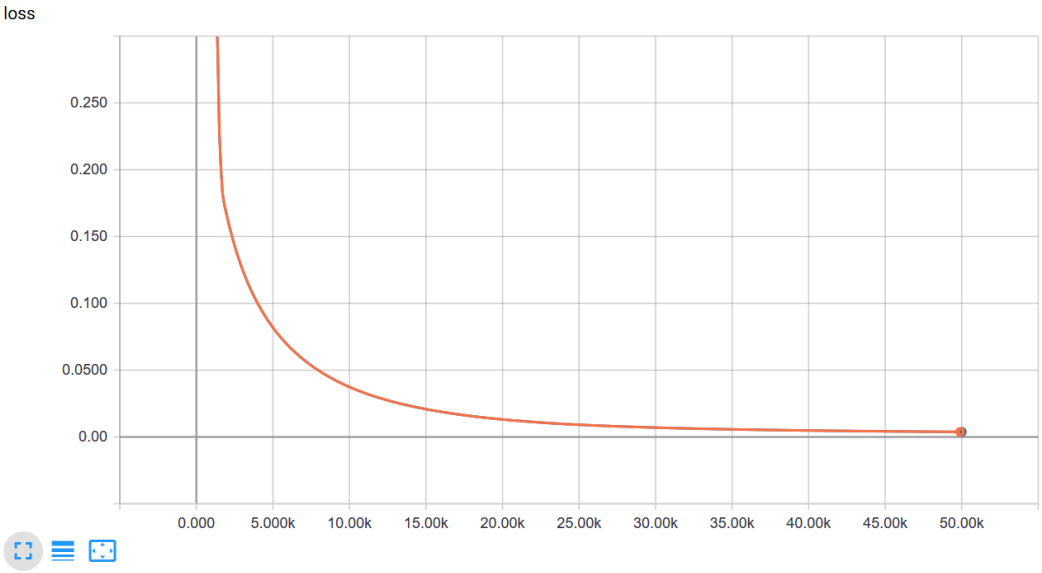

Рисунок 4.2: График зависимости среднеквадратичной ошибки от номера итерации вычислительного графа при обучении на модельных данных

4.4 Аппроксимация измеренных данных

4.4.1 Данные

Для проведения эксперимента по аппроксимации измеренных данных для обучения были взяты данные 24-х реальных кампаний реактора. Данные были разбиты на два набора: обучающая выборка (18 кампаний) и тестовая (6 кампаний). Для валидации обученной ИНС использовались данные всех 24-х кампаний.

4.4.2 Архитектура сети

Архитектура искусственной нейронной сети идентична архитектуре в разделе 4.3.2.

4.4.3 Процесс обучения

Для обучения ИНС было проведено 50000 эпох обучения. Обучение также проводилось методом обратного распространения ошибки по среднеквадратичной ошибке на обучающей выборке. Через каждые 100 эпох проводилась оценка среднеквадратичной ошибки на тестовой выборке. В течение всего процесса обучения ошибка сходилась к нулю без расхождения.

На рис. 4.4 показан график зависимости среднеквадратичной ошибки от номера итерации вычислительного графа.

4.4.4 Валидация

После завершения процесса обучения была проведена валидация ИНС на измеренных данных. На рис.4.5 приведены полные результаты сравнения измеренного запаса реактивности с результатами работы ИНС. Обобщенные результаты валидации:

- средняя абсолютная ошибка аппроксимации – $0,0412$;
- максимальная абсолютная ошибка аппроксимации – $0,1159$;
- средняя относительная ошибка аппроксимации – $1,26\%$;
- максимальная относительная ошибка аппроксимации – $3,56\%$.

Среднее время расчета запаса реактивности с помощью искусственной нейронной сети составляет 100 мс.

4.4.5 Выводы

В рамках описанных вычислительных экспериментов было показано, что полученные нейросети реализуют корректную аппроксимацию, обладают высокой точностью и скоростью работы [4].

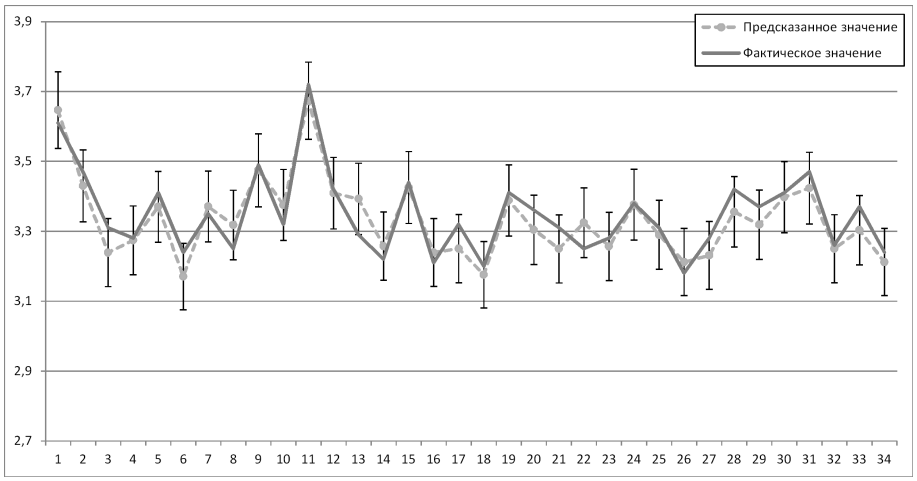

Рисунок 4.3: График верификации работы искусственной нейронной сети, обученной на модельных данных (по оси абсцисс отложены номера измерений в вариационной выборке, по оси ординат – значение запаса реактивности, % $\Delta K/K$)

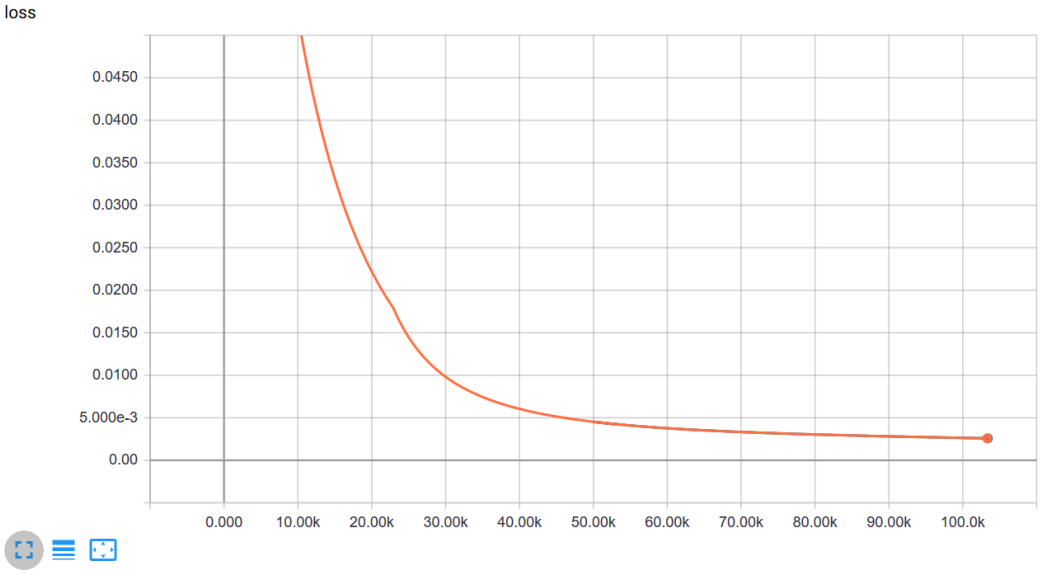

Рисунок 4.4: График зависимости среднеквадратичной ошибки от номера итерации вычислительного графа при обучении на измеренных данных

4.5 Программный комплекс предсказания

Было показано, что на основании конечного числа прецизионных расчетов либо изменений с помощью искусственной нейронной сети можно реализовать аппроксимацию запаса реактивности реактора ВВР-ц. Для конфигураций активной зоны реактора в рамках обучающей выборки возможно получить быструю и достаточно точную оценку запаса реактивности.

Следующим шагом необходимо обеспечить возможность использования искусственных нейронных сетей для предварительных расчетов запаса реактивности. Для решения этой задачи разработан программный комплекс оценки запаса реактивности реактора ВВР-ц.

Программный комплекс оценки запаса реактивности обеспечивает следующие требования:

- пополнять обучающую выборку для искусственной нейронной сети;
- работать в режимах обучения и использования обученной нейронной сети;
- иметь удобный и интуитивно понятный пользовательский интерфейс;
- обладать легкостью установки и эксплуатации.

Разработанный программный комплекс состоит из следующих структурных элементов:

- искусственная нейронная сеть;
- хранилище данных для обучения;
- REST API для обмена данными;
- пользовательский интерфейс.

Рассмотрим эти элементы подробно.

В качестве основы для построения искусственной нейронной сети использован фреймворк TensorFlow [35–37], который строит и исполняет граф вычислений в гетерогенных вычислительных системах, имеет богатую библиотеку примитивов для построения искусственных нейронных сетей и обеспечивает эффективное использование доступных вычислительных сред.

С помощью примитива DNNRegressor из библиотеки TensorFlow создана искусственная нейронная сеть. Входной слой сети – 50 формальных нейронов с функцией активации ReLu, скрытый слой – 10 формальных нейронов с функций активации ReLu, выходной слой – один формальный нейрон с логистической функцией активации. Таким образом, структура нейронной сети полностью повторяет архитектуру сети, использованную в вычислительных экспериментах.

Примитив DNNRegressor может функционировать в режимах обучения, оценки и предсказания.

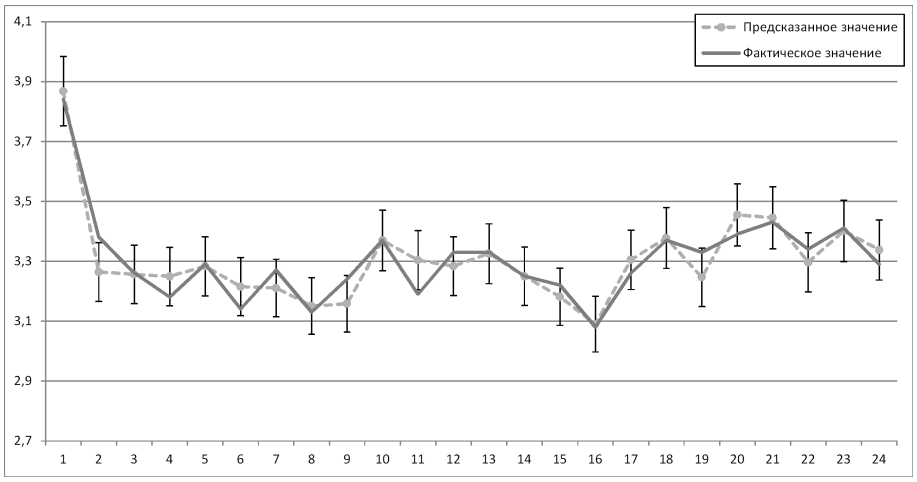

Рисунок 4.5: График валидации работы искусственной нейронной сети, обученной на измеренных данных (по оси абсцисс отложены номера измерений в вариационной выборке, по оси ординат – значение запаса реактивности, % $\Delta K/K$)

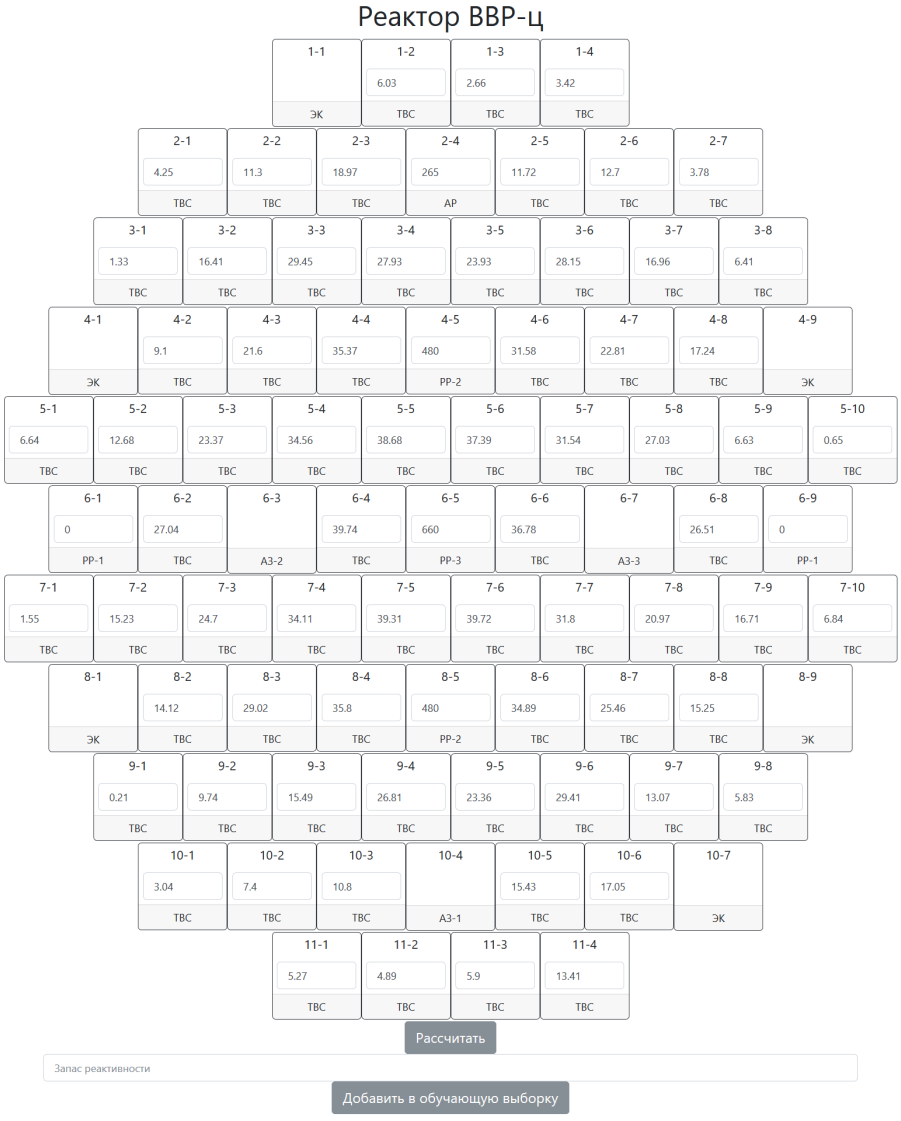

Рисунок 4.6: Интерфейс программного комплекса оценки запаса реактивности

Для обучения сети DNNRegressor применяет режимы обучения и оценки, а для использования ИНС – режим предсказания. Предварительно обученный примитив DNNRegressor сохраняет в файле tfdata, где содержатся описание графа предсказания (Inference) и все весовые коэффициенты формальных нейронов.

Для хранения обучающих данных используются файлы формата Comma-separated Values (CSV). Каждая запись представляет собой строку, в которую последовательно записаны проценты выгорания каждой ТВС, положения СУЗ, а также значение запаса реактивности при данной конфигурации активной зоны. Для проведения процедуры обучения данные из CSV-файлов загружаются в память в виде массивов NumPy.

Для обеспечения взаимодействия с пользователями программного комплекса с помощью библиотеки VueJs реализован веб-интерфейс управления [38–40], представляющий собой упрощенный вид картограммы реактора. В каждой из ТВС можно указать процент выгорания, а для каждого органа СУЗ – положение (см. рис. 4.6)). В зависимости от выбранного режима работы будет либо произведена оценка запаса критичности реактора в данной конфигурации, либо добавлено еще одно значение в обучающую выборку. Для обеспечения связи между частями программного комплекта с помощью библиотеки Flask [41–43] реализован программный интерфейс приложения передачи состояния представления (REST API) [44,45].

4.6 Выводы по главе

В данной главе рассмотрена возможность аппроксимации запаса реактивности реактора с помощью полносвязной искусственной нейронной сети в качестве предварительных расчетов; обучены две искусственные нейронные сети на разных наборах данных (на модельных, полученных с помощью прецизионной модели реактора, и на измеренных данных реальных кампаний).

Показано, что обе аппроксимации обладают достаточной точностью для проведения предварительных расчетов запаса реактивности. По итогам вычислительных экспериментов максимальная относительная ошибка аппроксимации составила 3,13 и 3,56% соответственно.

На основе обученных искусственных нейронных сетей создан программный комплекс оценки запаса реактивности реактора ВВР-ц. Комплекс позволяет в удобной и наглядной форме получить предсказанное нейронной сетью значение запаса реактивности, а также пополнить обучающую выборку новыми данными для обучения.

Программный комплекс для оценки запаса реактивности готов для тестирования персоналом реактора ВВР-ц. Параллельно с тестированием в данный программный комплекс можно внести ряд изменений, повышающих удобство и безопасность эксплуатации: шифрование данных в обучающей выборке; авторизацию, аутентификацию и аккаунтинг пользователей; возможность ручного редактирования обучающей выборки.

Допустимо использование данного программного комплекса в виде компонента системы автоматического планирования перезагрузки реактора. С незначительными изменениями комплекс можно применять для реакторных установок других типов.

Заключение

В рамках данной научной работы были достигнуты следующие результаты:

1. Рассмотрены конструкция ядерного реактора и процесс перегрузки ядерного топлива;
2. Исследованы возможности программных средств, применяемых для расчета процессов перегрузки;
3. Обоснована необходимость разработки метода и программных средств для оптимизации эксплутационных процессов;
4. Предложена автоматная модель описания эксплутационных процессов;
5. Показана применимость автоматной модели для описания процесса перегрузки ядерного топлива;
6. Показана применимость методов поиска в графе для поиска оптимальной последовательности ТА-модели;
7. Предложен нейросетевой подход к поиску оптимальной последовательности для больших автоманых моделей;
8. Предложено использование искусственной нейронной сети для аппроксимации эмерджентных свойств;
9. Проведены эксперименты по построению нейросетевой аппроксимации для запаса реактивности реактора ВВР-ц;
10. Разработан программный комплекс аппроксимации запаса реактивности реактора ВВР-ц.

На защиту выносятся следующие основные положения:

1. Разработан подход к построению автоматной модели эксплутационных процессов ядерных реакторов;
2. Предложены алгоритмы поиска оптимальной последовательности процесса в автоматной модели (в том числе и с использованием искусственных нейронных сетей);

3. Разработан и проверен метод аппроксимации эмерджентных свойств для автоматной модели на основе искусственных нейронных сетей;

4. Разработан программный комплекс аппроксимации запаса реактивности реактора ВВР-ц.

Список рисунков

1.1 Общий вид реактора БН-350 . . . 11
1.2 Сетка СУЗ в реакторе БН-350 . . . 12
1.3 Картограмма порядка выгрузки ТВС для программного средства RERPRORYV 16

2.1 Обзорная диаграмма процесса моделирования и оптимизации . . . 19
2.2 Пример диаграммы перехода конечного автомата . . . 21
2.3 Конфигурация активной зоны фиктивного реактора . . . 23
2.4 Композиционная схема автоматов модели фиктивного реактора . . . 30

3.1 Пример алгоритма Дейкстры. Исходный граф . . . 35
3.2 Пример алгоритма Дейкстры. Инициализация . . . 35
3.3 Пример алгоритма Дейкстры. Первый шаг алгоритма . . . 36
3.4 Пример алгоритма Дейкстры. Второй шаг алгоритма . . . 37
3.5 Пример алгоритма Дейкстры. Конец работы алгоритма . . . 37
3.6 Пример алгоритма A*. Исходное поле . . . 39
3.7 Пример алгоритма A*. Первый шаг алгоритма . . . 40
3.8 Пример алгоритма A*. Второй шаг алгоритма . . . 40
3.9 Пример алгоритма A*. Конец работы алгоритма . . . 41
3.10 Пример волнового алгоритма. Исходное поле . . . 42
3.11 Пример волнового алгоритма. Первый шаг распространения волны . . . 43
3.12 Пример волнового алгоритма. Второй шаг распространения волны . . . 43
3.13 Пример волнового алгоритма. Третий шаг распространения волны . . . 43
3.14 Пример волнового алгоритма. Последний шаг распространения волны . . . 43
3.15 Пример волнового алгоритма. Обратный ход: формирование пути . . . 44

4.1 Граф вычислений TensorFlow . . . 51
4.2 График среднеквадратичной ошибки ИНС на модельных данных . . . 51
4.3 График верификации ИНС на модельных данных . . . 53
4.4 График среднеквадратичной ошибки ИНС на измеренных данных . . . 53
4.5 График валидации ИНС на измеренных данных . . . 55
4.6 Интерфейс программного комплекса оценки запаса реактивности . . . 55

Список таблиц

2.1 Таблица переходов автомата «Тепловыделяющая сборка с обогащенным ураном» 24
2.2 Таблица переходов автомата «Тепловыделяющая сборка с обедненным ураном» 25
2.3 Таблица переходов автомата «Гнездо внутриреакторного хранилища» 25
2.4 Таблица переходов автомата «Гнездо зоны воспроизводства» 26
2.5 Таблица переходов автомата «Гнездо активной зоны» 26
2.6 Таблица переходов автомата «Большая поворотная пробка» 27
2.7 Таблица переходов автомата «Малая поворотная пробка» 27
2.8 Таблица переходов автомата «Определитель типа гнезда» 28
2.9 Таблица переходов автомата «Селектор гнезда». 28
2.10 Таблица переходов автомата «Перегрузочный механизм» 29

Литература

1. Power reactor Informational System. Nuclear Share of Electricity Generation in 2017. Электронный ресурс: https://www.iaea.org/PRIS/WorldStatistics/NuclearShareofElectricityGeneration.aspx (дата обращения 2.05.2018).

2. Захаров А., Овакимян М. Тенденции развития мировой энергетики // Мировое и национальное хозяйство. 2015. № 1(32).

3. Ильина Н.А. РОСАТОМ. Замкнутый ядерный топливный цикл. Электронный ресурс: http://science.spb.ru/files/tehplatformy/neutron/presentation/files/assets/common/downloads/publication.pdf (дата обращения 2.05.2018).

4. Approximation of the criticality margin of WWR-c reactor using artificial neuron networks. / I. Belyavtsev, D. Legchikov, S. Starkov [и др.] // Journal of Physics: Conference Series. 2018. Т. 945, № 1. С. 012–031.

5. Белявцев И.П., Старков С.О. Программный комплекс оценки запаса реактивности реактора ВВР-ц. // Известия высших учебных заведений. Ядерная энергетика. 2018. № 2. С. 58–66.

6. Белявцев И.П., Старков С.О. Моделирование и оптимизация эксплутационных процессов на атомных электростанциях с использованием методов искусственного интеллекта. // Наукоемкие технологии в приборо- и машиностроении и развитие инновационной деятельности в ВУЗе: материалы Всероссийской научно-технической конференции, 25-27 ноября 2014 г. Т. 4. М.: Издательство МГТУ им. Н.Э. Баумана, 2014. С. 4–9.

7. Построение нейросетевой модели реактора ВВР-ц для прогнозирования запаса критичности. / И.П. Белявцев, С.О. Старков, Д.К. Легчиков [и др.] // Наукоемкие технологии в приборо- и машиностроении и развитие инновационной деятельности в ВУЗе: материалы Всероссийской научно-технической конференции, 25-27 ноября 2014 г. Т. 4. М.: Издательство МГТУ им. Н.Э. Баумана, 2014. С. 10–15.

8. Белявцев И.П., Старков С.О. Моделирование эксплутационных процессов ядерных реакторов с использованием методов искусственного интеллекта. // Научная сессия НИЯУ МИФИ-2015. Аннотации докладов. Т. 2. М.: НИЯУ МИФИ, 2015. С. 271.

9. Прогнозирование запаса критичности реактора ВВР-ц методом нейросетевого моделирования. / И.П. Белявцев, С.О. Старков, Д.К. Легчиков [и др.] // Научная сессия НИЯУ МИФИ-2015. Аннотации докладов. Т. 2. М.: НИЯУ МИФИ, 2015. С. 272.

10. Белявцев И.П., Старков С.О., Колесов В.В. Прогнозирование запаса критичности реактора ВВР-ц методом нейросетевого моделирования. // XIV Международная конференция «Безопасность АЭС и подготовка кадров». Тезисы докладов. Обнинск: ИАТЭ НИЯУ МИФИ, 2015. С. 138–139.

11. Аппроксимация запаса критичности реактора ВВР-ц с использованием исскусственной нейронной сети. / И.П. Белявцев, Д.К. Легчиков, С.О. Старков [и др.] // Современные проблемы физики и технологий. VI-я Международная молодежная научная школа-конференция, 17-21 апреля 2017 г.: Тезисы докладов. Часть 1. М.: НИЯУ МИФИ, 2017. С. 80–81.

12. Белявцев И.П., Старков С.О. Поиск оптимальной последовательности событий автоматной модели с использованием искусственных нейронных сетей. // Современные проблемы физики и технологий. VII-я Международная молодежная научная школа-конференция, 16-21 апреля 2018 г.: Тезисы докладов. Часть 2. М.: НИЯУ МИФИ, 2018. С. 346–347.

13. Г.Б. Усынин, Кусмарцев. Реакторы на быстрых нейтронах: Учеб. пособие для вузов. М.: Энергоатомиздат, 1985. С. 288.

14. TRIGEX.051. Аттестационный паспорт программного средства. Регистрационный No 313 от 9 октября 2012 г. Паспорт ФБУ «НТЦ ЯРБ».

15. Программа расчета нейтронно-физических характеристик ячеек реакторов САПФИР_95.1 — Аттестационный паспорт №390 от 16.12.2015 г. Федеральная служба по экологическому, технологическому и атомному надзору.

16. Программа расчета нейтронно-физических характеристик реактора. САПФИР_95&RC_ВВЭР.2 — Аттестационный паспорт. №321 от 18.04.13 г. Федеральная служба по экологическому, технологическому и атомному надзору.

17. Расчет нейтронно-физических характеристик быстрого реактора в замкнутом топливном цикле с помощью кода REPRORYV // Вопросы атомной науки и техники. Серия: Ядерно-реакторные константы. 2017. № 1. С. 5–18.

18. И.Р. Акишев, М.Э. Дворкин. О построении минимальных детерминированных конечных автоматов, распознающих префиксный код заданной мощности // Прикладная дискретная математика. 2010.

19. Дж. Хопкрофт, Р. Мотвани, Дж. Ульман. Введение в теорию автоматов, языков и вычислений = Introduction to Automata Theory, Languages, and Computation. М.: Вильямс, 2002. С. 528. ISBN 0-201-44124-1.

20. Богаченко Н.Ф. Файзуллин Р.Т. Синтез дискретных автоматов: Учебное пособие. Омск: Издательство Наследие. Диалог-Сибирь, 2006. С. 150.

21. А.М. Лупал. Теория автоматов: Учебное пособие. СПб.: ГУАП, 2000. С. 119.

22. Ф. Харари. Теория графов / под ред. Г.П. Гаврилова. Издательство мир, 1973. С. 300 с. ISBN 5-354-00301-6.

23. В.А. Евстигнеев. Применение теории графов в программировании / под ред. А.П. Ершова. Москва: Наука. Главная редакция физико-математической литературы, 1985. С. 352 с.

24. А. Сплитфейс. Алгоритм Дейкстры. Поиск оптимальных маршрутов на графе. URL: http://habrahabr.ru/post/111361/.

25. С.Дж. Рассел, П. Норвиг. Искусственный интеллект: современный подход = Artificial Intelligence: A Modern Approach / Пер. с англ. и ред. К. А. Птицына. — 2-е изд. М.: Вильямс, 2006. С. 157—162. ISBN 5-8459-0887-6.

26. Д. Платонов. Поиск пути: алгоритм A* для новичков. URL: http://savepearlharbor.com/?p=159417.

27. Л.Б. Абрайтис. Автоматизация проектирования топологии цифровых интегральных микросхем. М.:Радио и связь, 1985. С. 200.

28. Родионов И.Б. Теория систем и системный анализ. Конспекты лекций. Электронный ресурс: http://victor-safronov.ru/systems-analysis/lectures/rodionov.html (дата обращения 2.05.2018).

29. Кочнов О.Ю., Лукин Н.Д., Аверин Л.В. Реактор ВВР-ц: опыт эксплуатации и перспективы развития. // Ядерная и радиационная безопасность. 2008. № 1. С. 18–25.

30. Создание прецизионной модели реактора ВВР-ц для последующей оптимизации его конструкции и наработки ^{99}Mo и других радионуклидов / В.В. Колесов, О.Ю. Кочнов, Ю.В. Волков [и др.] // Известия высших учебных заведений. Ядерная энергетика. 2011. № 4. С. 129–133.

31. Оценка увеличения производства ^{131}I при использовании теллуровых мишеней усовершенствованной конструкции на реакторе ВВР-ц. / О.Ю. Кочнов, В.В. Колесов, Р.И. Фомин [и др.] // Известия высших учебных заведений. Ядерная энергетика. 2014. № 4. С. 102–110.

32. С. Хайкин. Нейронные сети: полный курс, 2-е издание : Пер. с англ. М.: Издательский дом «Вильямс», 2006. С. 1104.

33. Т.В. Филатова. Применение нейронных сетей для аппроксимации данных // Вестник Томского государственного университета. 2004. № 284. С. 121–125.

34. А.Н. Горбань. Обобщенная аппроксимационная теорема и вычислительные возможности нейронных сетей // Сибирский журнал вычислительной математики. 1998. Т. 1, № 1. С. 12—24.

35. Abadi Martín, Agarwal Ashish, Barham Paul [и др.]. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

36. Zaccone G. Getting Started with TensorFlow. Packt Publishing, 2016. С. 180.

37. Lieder I., Resheff Y., Hope T. Learning TensorFlow. A Guide to Building Deep Learning Systems. O'Reilly Media, 2017. С. 242.

38. You E. VueJs. Электронный ресурс: https://vuejs.org/v2/guide/ (дата обращения 2.05.2018).

39. Street M. Vue.js 2.x by Example. Packt Publishing, 2017. С. 412.

40. Filipova O. Learning Vue.js 2. Learn how to build amazing and complex reactive web applications easily with Vue.js. Packt Publishing, 2016. С. 334.

41. Ronacher A. Flask. Электронный ресурс: http://flask.pocoo.org/ (дата обращения 2.05.2018).

42. Grinberg M. Flask Web Development. 2 изд. O'Reilly Media, 2018. С. 316.

43. Dwyer G. Flask by Example. Packt Publishing, 2016. С. 276.

44. Richardson L., Ruby S. RESTful Web Services. O'Reilly Media, 2008. С. 448.

45. Masse M. REST API Design Rulebook. O'Reilly Media, 2011. С. 116.

Приложение А

Исходный код нейросетевой аппроксимации

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import itertools

import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

tf.logging.set_verbosity(tf.logging.INFO)

COLUMNS = ['1-1', '1-2', '1-3', '1-4', '10-1', '10-2', '10-3', '10-4', '10-5',
'10-6', '10-7', '11-1', '11-2', '11-3', '11-4', '2-1', '2-2', '2-3', '2-4', '
2-5', '2-6', '2-7', '3-1', '3-2', '3-3', '3-4', '3-5', '3-6', '3-7', '3-8', '
4-1', '4-2', '4-3', '4-4', '4-5', '4-6', '4-7', '4-8', '4-9', '5-1', '5-10', '
5-2', '5-3', '5-4', '5-5', '5-6', '5-7', '5-8', '5-9', '6-1', '6-2', '6-3', '
6-4', '6-5', '6-6', '6-7', '6-8', '6-9', '7-1', '7-10', '7-2', '7-3', '7-4', '
7-5', '7-6', '7-7', '7-8', '7-9', '8-1', '8-2', '8-3', '8-4', '8-5', '8-6', '
8-7', '8-8', '8-9', '9-1', '9-2', '9-3', '9-4', '9-5', '9-6', '9-7', '9-8', 'AP
', 'PP1', 'PP2', 'PP3', 'ro']
FEATURES = ['1-1', '1-2', '1-3', '1-4', '10-1', '10-2', '10-3', '10-4', '10-5',
 '10-6', '10-7', '11-1', '11-2', '11-3', '11-4', '2-1', '2-2', '2-3', '2-4', '
2-5', '2-6', '2-7', '3-1', '3-2', '3-3', '3-4', '3-5', '3-6', '3-7', '3-8', '
4-1', '4-2', '4-3', '4-4', '4-5', '4-6', '4-7', '4-8', '4-9', '5-1', '5-10', '
5-2', '5-3', '5-4', '5-5', '5-6', '5-7', '5-8', '5-9', '6-1', '6-2', '6-3', '
6-4', '6-5', '6-6', '6-7', '6-8', '6-9', '7-1', '7-10', '7-2', '7-3', '7-4', '
7-5', '7-6', '7-7', '7-8', '7-9', '8-1', '8-2', '8-3', '8-4', '8-5', '8-6', '
8-7', '8-8', '8-9', '9-1', '9-2', '9-3', '9-4', '9-5', '9-6', '9-7', '9-8', 'AP
', 'PP1', 'PP2', 'PP3']
```

```
16    LABEL = 'ro'
17
18    #training_set = pd.read_csv("wwrc_data/real_data.csv", skipinitialspace=True,
19    #                            skiprows=1, names=COLUMNS)
20    #test_set = pd.read_csv("wwrc_data/real_data.csv", skipinitialspace=True,
21    #                        skiprows=1, names=COLUMNS)
22    prediction_set = pd.read_csv("wwrc_data/real_data.csv", skipinitialspace=True,
23                                 skiprows=1, names=COLUMNS)
24    training_set = prediction_set.head(int(prediction_set.size*0.75))
25    test_set = prediction_set.tail(int(prediction_set.size*0.25))
26
27    feature_cols = [tf.contrib.layers.real_valued_column(k) for k in FEATURES]
28
29    regressor = tf.contrib.learn.DNNRegressor(feature_columns=feature_cols,
30                                              hidden_units=[50, 10],
31                                              model_dir="real_model")
32
33    def input_fn(data_set):
34      feature_cols = {k: tf.constant(data_set[k].values)
35                      for k in FEATURES}
36      labels = tf.constant(data_set[LABEL].values)
37      return feature_cols, labels
38
39    def train_input_fn():
40      return input_fn(training_set)
41
42    def eval_input_fn():
43      return input_fn(test_set)
44
45
46    regressor.fit(input_fn=lambda: input_fn(training_set), steps=50000)
47
48    ev = regressor.evaluate(input_fn=lambda: input_fn(test_set), steps=1)
49
50    loss_score = ev["loss"]
51    print("Loss: {0:f}".format(loss_score))
52
53    y = regressor.predict(input_fn=lambda: input_fn(prediction_set))
54    # .predict() returns an iterator; convert to a list and print predictions
55    predictions = list(itertools.islice(y, int(prediction_set.size)))
56
57    print ("Predictions: {}".format(str(predictions)))
58
59    y_target = prediction_set[LABEL].values
60    y_pred = np.array(predictions)
61
62    print("Diffs:", y_target-y_pred)
```

```
63      print("Max relative error: ",
64              np.amax(np.abs(100*(y_target-y_pred)/y_target)), "%")
65
66      print("Mean relative error: ",
67              np.mean(np.abs(100*(y_target-y_pred)/y_target)), "%")
68
69      print("Max absolute error: ",
70              np.amax(np.abs(y_target-y_pred)))
71
72      print("Min absolute error: ",
73              np.amin(np.abs(y_target-y_pred)))
74
75      sigma = np.sqrt(np.sum(y_target-y_pred)*np.sum(y_target-y_pred) /
76                          (len(y_target)-1))
77      print("Sigma: ", 2*sigma)
78
79      x = np.arange(1, y_target.shape[0]+1, 1)
80      plt.figure(figsize=(12, 8))
81      plt.errorbar(x, y_pred, color='black',
82                   yerr=sigma*2, label=u'Output')
83      plt.plot(x, y_target, color='green', label=u'Source value', marker='o')
84      plt.xlabel(u'Number of company')
85      plt.ylabel(r'$\rho$')
86      plt.legend()
87      plt.grid(True)
88      plt.savefig('graph.png')
```