ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

На правах рукописи

СИДОРЕНКО Евгений Васильевич

УПРАВЛЕНИЕ ПРОЦЕССАМИ МОНИТОРИНГА ПРОИЗВОДИТЕЛЬНОСТИ КОЛЛЕКТИВОВ ВСТРАИВАЕМЫХ ОБЪЕКТОВ БОЛЬШИХ ПРОГРАММНЫХ СИСТЕМ В ОБЛАЧНЫХ АРХИТЕКТУРАХ

Специальность: 2.3.5. Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

Диссертация
на соискание ученой степени
кандидата технических наук

Научный руководитель:
д.т.н., доцент Рындин Никита Александрович

Воронеж – 2025

Содержание

ВВЕДЕНИЕ

Актуальность темы. С быстрым развитием датчиков и информационных технологий, обеспечивающих промышленное производство, сетевые данные, которые представляют взаимодействия между взаимосвязанными объектами, представлены в коллективах встраиваемых объектов больших программных систем.

Статистический мониторинг процессов служит эффективным инструментом поддержки точного и своевременного принятия решений в. Применение подходов к мониторингу статистических процессов для мониторинга сетей значительно облегчает раннее обнаружение потенциальных сбоев в сложных реляционных системах и, следовательно, в последние годы все чаще изучается. Выбор эффективного метода сетевого мониторинга основывается на оценке эффективности методов-кандидатов. Однако исследований по систематической оценке и сравнению методов сетевого мониторинга очень мало. В частности, возможность частого сбора данных с помощью современных измерительных устройств, как правило, приводит к возникновению автокорреляций между сетями. Тем не менее, методов оценки производительности для автокоррелированных сетей крайне не хватает. Большой вклад в развитие методов статистического мониторинга программных систем внесли Афанасьев В.Н., Домрачев В.Г., Городецкий А.Я., Скуратов А.К., Farahani E.M., He Z., Kubacki M. Разумной идеей кажется разработать модель статистического мониторинга процессов с большими данными в компьютерных сетях, учитывающую свойства автокорреляции процессов образования и распада части локальных структур.

Сети обычно используются для описания взаимосвязей между различными объектами в сложных системах. Контрольные диаграммы как

правило, используются для мониторинга того, остается ли процесс со случайными шумами статистически контролируемым с течением времени. Ожидается, что хорошая контрольная диаграмма быстро обнаружит изменения в процессе в случае выхода из-под контроля и не вызовет ложной тревоги, когда процесс находится под контролем. Однако эти две цели не могут быть достигнуты одновременно. Разрешение этого противоречия является значимой задачей.

Мониторинг производительности и обнаружение аномалий являются основными целями при проектировании и обслуживании электронных устройств и систем. В последние годы они усложняются из-за усложнения аппаратного и программного обеспечения. Следовательно, важным моментом является сбор репрезентативных выборок сигналов и выявление характерных особенностей, позволяющих оценить рабочие характеристики устройств. Это приводит к необходимости эффективного анализа временных рядов. Анализ рабочих характеристик различных физических устройств или систем предполагает измерение различных параметров (метрик) во времени. Собранные выборки данных представляют собой временные ряды, которые необходимо исследовать для получения характерных признаков, коррелирующих с исследуемыми свойствами. Отсюда понятна необходимость создания архитектуры программного обеспечения оптимизации облачных сервисов «программное обеспечение как услуга» для решения задач мониторинга.

Таким образом, актуальность темы диссертационного исследования продиктована необходимостью дальнейшего развития средств управления процессами мониторинга производительности коллективов встраиваемых объектов больших программных систем в облачных архитектурах на основе учета автокорреляции образования и распада части локальных структур.

Тематика диссертационной работы соответствует научному

направлению ФГБОУ ВО «Воронежский государственный технический университет» «Вычислительные комплексы и проблемно-ориентированные системы управления».

Целью работы является разработка моделей и алгоритмов управления процессами мониторинга производительности коллективов встраиваемых объектов больших программных систем в облачных архитектурах на основе учета автокорреляции образования и распада части локальных структур.

Задачи исследования. Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ проблем создания моделей и алгоритмов управления процессами мониторинга производительности коллективов встраиваемых объектов больших программных систем в облачных архитектурах на основе учета автокорреляции образования и распада части локальных структур.

2. Разработать модель статистического мониторинга процессов с большими данными в компьютерных сетях, учитывающую свойства автокорреляции процессов образования и распада части локальных структур на основе контрольных диаграмм.

3. Модифицировать алгоритмы мониторинга производительности коллективов встраиваемых объектов больших программных систем для обеспечения возможности иерархического анализа данных и улучшения интерпретируемости результатов.

4. Создать оптимизационную модель и алгоритмы планирования пропускной способности облачных сервисов SaaS для нахождения компромисса между стоимостью ресурсов и штрафом за задержку выполнения во встраиваемых облачных приложениях "программное обеспечение как услуга" с точки зрения поставщика облачных услуг.

5. Разработать динамический алгоритм определения минимальной

верхней границы нулевого штрафа, обеспечивающий получение верхней границы объема вычислительных ресурсов, еще дающей нулевой штраф.

6. Разработать структуру программного обеспечения оптимизации облачных сервисов SaaS для решения задач мониторинга, обеспечивающую принятие решений о допустимости конфигурации сервиса с точки зрения штрафов, и также оценку ожидаемого качества обслуживания.

Объект исследования: процессы мониторинга производительности коллективов встраиваемых объектов больших программных систем.

Предмет исследования: средства управления процессами мониторинга производительности коллективов встраиваемых объектов больших программных систем в облачных архитектурах на основе учета автокорреляции образования и распада части локальных структур.

Методы исследования. При решении поставленных в диссертации задач использовались методы теории графов, теории вероятностей, теории принятия решений, а также методы объектно-ориентированного программирования.

Тематика работы соответствует следующим пунктам паспорта специальности 2.3.5 «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей»: п.3 «Модели, методы, архитектуры, алгоритмы, языки и программные инструменты организации взаимодействия программ и программных систем»; п.9 «Модели, методы, алгоритмы, облачные технологии и программная инфраструктура организации глобально распределенной обработки данных».

Научная новизна работы. В диссертации получены следующие результаты, характеризующиеся научной новизной:

1. Модель статистического мониторинга процессов с большими данными в компьютерных сетях, отличающаяся учетом свойств

автокорреляции процессов образования и распада части локальных структур на основе контрольных диаграмм и обеспечивающая моделирование сдвигов относительно уровней плотности, взаимности, изменчивости степени и свойств транзитивности сети.

2. Алгоритмы мониторинга производительности коллективов встраиваемых объектов больших программных систем, отличающиеся предварительной кластеризацией и идентификацией поведенческих последовательностей, удовлетворяющих условиям сходства, обеспечивающие возможность иерархического анализа данных и улучшающие интерпретируемость результатов.

3. Оптимизационная модель и алгоритмы планирования пропускной способности облачных сервисов SaaS, отличающиеся форсированным решением NP-сложной задачи минимизации затрат на задержку выполнения запросов и на ресурсы развернутых экземпляров и обеспечивающие нахождение компромисса между стоимостью ресурсов и штрафом за задержку выполнения во встраиваемых облачных приложениях "программное обеспечение как услуга" с точки зрения поставщика облачных услуг.

4. Динамический алгоритм определения минимальной верхней границы нулевого штрафа, отличающийся использованием ожидающей очереди неисполненных запросов и обеспечивающий получение верхней границы объема вычислительных ресурсов, еще дающей нулевой штраф.

5. Структура программного обеспечения оптимизации облачных сервисов SaaS для решения задач мониторинга, отличающаяся интеграцией системы мониторинга и сервисов SaaS и обеспечивающая принятие решений о допустимости конфигурации сервиса с точки зрения штрафов, и также оценку ожидаемого качества обслуживания.

Теоретическая и практическая значимость исследования заключается в разработке специальных средств математического и

программного обеспечения управления процессами мониторинга производительности коллективов встраиваемых объектов больших программных систем в облачных архитектурах на основе учета автокорреляции образования и распада части локальных структур.

Теоретические результаты работы могут быть использованы в проектных и научно-исследовательских организациях, занимающихся разработкой методов управления процессами мониторинга распределенного программного обеспечения.

Положения, выносимые на защиту

1. Модель статистического мониторинга процессов с большими данными в компьютерных сетях учитывает свойства автокорреляции процессов образования и распада части локальных структур на основе контрольных диаграмм.

2. Модифицированные алгоритмы мониторинга производительности коллективов встраиваемых объектов больших программных систем обеспечивают возможность иерархического анализа данных и улучшения интерпретируемости результатов.

3. Оптимизационная модель и алгоритмы планирования пропускной способности облачных сервисов SaaS решают задачу поиска компромисса между стоимостью ресурсов и штрафом за задержку выполнения во встраиваемых облачных приложениях "программное обеспечение как услуга" с точки зрения поставщика облачных услуг.

4. Динамический алгоритм определения минимальной верхней границы нулевого штрафа обеспечивает получение верхней границы объема вычислительных ресурсов, еще дающей нулевой штраф.

5. Структура программного обеспечения оптимизации облачных сервисов SaaS для решения задач мониторинга обеспечивает принятие решений о допустимости конфигурации сервиса с точки зрения штрафов, и также оценку ожидаемого качества обслуживания.

Результаты внедрения. Основные результаты внедрены в ООО «Центр информационных технологий» (г. Воронеж) при проектировании распределенной информационно-вычислительной системы управления мониторингом программных систем, в учебный процесс Воронежского государственного технического университета в рамках дисциплин: «Вычислительные машины, системы и сети», «Информационные сети и телекоммуникационные технологии», а также в рамках курсового и дипломного проектирования.

Апробация работы. Основные положения диссертационной работы докладывались и обсуждались на следующих конференциях: XXIX International Open Science Conference «Modern informatization problems in the technological and telecommunication systems analysis and synthesis» (Yelm, WA, USA, 2024); VII Международной НПК «Наука и технологии: перспективы развития и применения» (Петрозаводск, 2024); XXX International Open Science Conference «Modern informatization problems in simulation and social technologies» (Yelm, WA, USA, 2025), а также на научных семинарах кафедр САПРИС и искусственного интеллекта и цифровых технологий ВГТУ (2019-2025 гг.).

Достоверность результатов обусловлена корректным использованием теоретических методов исследования и подтверждена результатами сравнительного анализа данных вычислительных и натурных экспериментов.

Публикации. По результатам диссертационного исследования опубликовано 13 научных работ (4 – без соавторов), в том числе 7 – в изданиях, рекомендованных ВАК РФ (из них 2 – в изданиях, индексируемых в Scopus и WoS и одно свидетельство о регистрации программы для ЭВМ). В работах, опубликованных в соавторстве и приведенных в конце автореферата, лично автором получены следующие результаты: [4] - модель статистического мониторинга процессов с

большими данными в компьютерных сетях, учитывающую свойства автокорреляции процессов образования и распада части локальных структур; [2, 6] - алгоритмы мониторинга производительности коллективов встраиваемых объектов больших программных систем для обеспечения возможности иерархического анализа данных и улучшения интерпретируемости результатов; [1, 5, 12] - оптимизационная модель и алгоритмы планирования пропускной способности облачных сервисов SaaS для нахождения компромисса между стоимостью ресурсов и штрафом за задержку выполнения; [13] - динамический алгоритм определения минимальной верхней границы нулевого штрафа, обеспечивающий получение верхней границы объема вычислительных ресурсов, еще дающей нулевой штраф; [6, 7, 8] - структура программного обеспечения оптимизации облачных сервисов SaaS для решения задач мониторинга, обеспечивающего принятие решений о допустимости конфигурации сервиса с точки зрения штрафов.

Структура и объем работы. Диссертационная работа состоит из введения, четырех глав, заключения, списка литературы из 180 наименований. Работа изложена на 188 страницах основного текста.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

В первой главе исследуются проблемы управления процессами мониторинга производительности коллективов встраиваемых объектов больших программных систем в облачных архитектурах. Отмечено, что повысить эффективность управления процессами мониторинга можно путем разработки статистического мониторинга стохастических процессов на основе оценки производительности для автокоррелированных сетей, постановки оптимизационной задачи использования ограниченных ресурсов для достижения высокой производительности, а также

ограниченного энергопотребления для встраиваемых систем, алгоритмизации управления эластичностью для планирования мониторинговых мощностей в облачных архитектурах "программное обеспечение как услуга".

Исследована проблема статистического мониторинга процессов с большими данными в компьютерных сетях, учитывающая свойства автокорреляции процессов образования и распада части локальных структур на основе контрольных диаграмм. Предложена оптимизационная модель и алгоритмы планирования пропускной способности облачных сервисов SaaS для нахождения компромисса между стоимостью ресурсов и штрафом за задержку выполнения. Осуществлена алгоритмизация определения минимальной верхней границы нулевого штрафа, обеспечивающий получение верхней границы объема вычислительных ресурсов, еще дающей нулевой штраф.

Проанализированы требования к структуре программного обеспечения оптимизации облачных сервисов SaaS для решения задач мониторинга, обеспечивающей принятие решений о допустимости конфигурации сервиса с точки зрения штрафов, и также оценку ожидаемого качества обслуживания. Сформулирована цель и задачи исследования.

Вторая глава посвящена оценке производительности сетевого мониторинга на основе разделяемых временных экспоненциальных моделей случайных графов.

Основное внимание уделено параметрам создания и распада. Параметры создания и распада указывают на тенденцию соответствующих структур к новому созданию и распаду с течением времени. Следовательно, увеличение параметров создания при неизменных параметрах распада приводит к увеличению количества локальных

структур, а уменьшение параметров формирования приводит к уменьшению количества локальных структур.

Путем корректировки значений параметров моделируются сдвиги относительно уровней плотности, взаимности, изменчивости степени и свойств транзитивности сети.

Представлена модель статистического мониторинга процессов с большими данными в компьютерных сетях, отличающаяся учетом свойств автокорреляции процессов образования и распада части локальных структур на основе контрольных диаграмм и обеспечивающая моделирование сдвигов относительно уровней плотности, взаимности, изменчивости степени и свойств транзитивности сети.

Третья глава посвящена алгоритмизации мониторинга производительности коллективов встраиваемых объектов больших программных систем на основе временных рядов для анализа энергопотребления.

Предлагаемый метод анализа состоит из 3 этапов: сбор данных, создание объектно-ориентированной модели, поведенческая интерпретация этой модели. Исходные данные (выборки) преобразуются и агрегируются в иерархические объекты с учетом некоторых заданных свойств/показателей. В результате получается высокоуровневая графовая модель, которая облегчает извлечение нетривиальных сведений о поведении системы, в частности о состояниях активности устройств и их последовательности, связанных с потреблением энергии. Это полезно для оценки системы, а также для ее коррекции или уточнения.

Разработано 12 алгоритмов управления мониторингом производительности коллективов встраиваемых объектов больших программных систем на основе временных рядов для анализа энергопотребления.

На практике производный граф состояний может быть довольно сложным, в частности, содержать длинные последовательности состояний с одним входным и выходным ребрами. Но во многих случаях нецелесообразно сосредотачиваться на них – упрощение графика состояний путем объединения последовательности загрузки в одно состояние может облегчить дальнейший анализ. Следовательно, может быть создан сокращенный граф, который объединяет такие последовательности состояний в уникальное обобщенное состояние (в зависимости от целей анализа).

Выбор параметров для алгоритмов выполняется тремя способами:

1) на основе предыдущего опыта анализа (например, взятие параметров из ранее проанализированной версии проекта для проверки свойств следующей версии),

2) на основе практического опыта эксперта, который знает ожидаемые классы циклов (их периодичность, уровни мощности и т.д.),

3) итеративный поиск в экспериментах с репрезентативным фрагментом изучаемой TS или полным фрагментом.

В итоге предложены алгоритмы мониторинга производительности коллективов встраиваемых объектов больших программных систем, отличающиеся предварительной кластеризацией и идентификацией поведенческих последовательностей, удовлетворяющих условиям сходства, обеспечивающие возможность иерархического анализа данных и улучшающие интерпретируемость результатов.

В главе 4 представлены новые результаты в области управления эластичностью для планирования мощностей в облачных средах типа "Программное обеспечение как услуга", обслуживающих системы мониторинга.

Поставщики облачных услуг используют планирование производительности для поддержания производительности

вычислительных ресурсов (экземпляров), необходимой для удовлетворения динамического характера спроса (запросов). Однако существует компромисс между развертыванием слишком большого количества дорогостоящих экземпляров и развертыванием слишком малого количества экземпляров и выплатой штрафных санкций за невозможность своевременной обработки запросов. Определение оптимального количества экземпляров, необходимого для данного горизонта планирования, является сложной задачей из-за комбинаторного характера задачи оптимизации.

Представлены оптимизационная модель и алгоритмы планирования пропускной способности облачных сервисов SaaS, отличающиеся форсированным решением NP-сложной задачи минимизации затрат на задержку выполнения запросов и на ресурсы развернутых экземпляров и обеспечивающие нахождение компромисса между стоимостью ресурсов и штрафом за задержку выполнения во встраиваемых облачных приложениях "программное обеспечение как услуга" с точки зрения поставщика облачных услуг.

Также разрабатываются алгоритмы решения проблемы неопределенности спроса в задачах мониторинга.

По результатам имитационного моделирования среднее снижение затрат при использовании алгоритма по сравнению с существующим составляет 4,73% и позволяет использовать меньше задействованных ресурсов.

Представлен динамический алгоритм определения минимальной верхней границы нулевого штрафа, отличающийся использованием ожидающей очереди неисполненных запросов и обеспечивающий получение верхней границы объема вычислительных ресурсов, еще дающей нулевой штраф.

Далее разработана структура программного прототипа оптимизации

облачных сервисов SaaS для решения задач мониторинга (рис. 4). Соответствующая архитектура программного обеспечения оптимизации облачных сервисов SaaS для решения задач мониторинга отличается интеграцией системы мониторинга и сервисов SaaS и обеспечивает принятие решений о допустимости конфигурации сервиса с точки зрения штрафов, и также оценку ожидаемого качества обслуживания**.** Элементы программной реализации прошли государственную регистрацию в Роспатенте.

Создана структура программного обеспечения оптимизации облачных сервисов SaaS для решения задач мониторинга, отличающаяся интеграцией системы мониторинга и сервисов SaaS и обеспечивающая принятие решений о допустимости конфигурации сервиса с точки зрения штрафов, и также оценку ожидаемого качества обслуживания.

1. АНАЛИЗ ПРОБЛЕМ УПРАВЛЕНИЯ ПРОЦЕССАМИ МОНИТОРИНГА ПРОИЗВОДИТЕЛЬНОСТИ КОЛЛЕКТИВОВ ВСТРАИВАЕМЫХ ОБЪЕКТОВ БОЛЬШИХ ПРОГРАММНЫХ СИСТЕМ В ОБЛАЧНЫХ АРХИТЕКТУРАХ

1.1. Проблемы статистического мониторинга стохастических процессов на основе оценки производительности для автокоррелированных сетей

С быстрым развитием датчиков и информационных технологий, обеспечивающих промышленное производство, сетевые данные, которые представляют взаимодействия между взаимосвязанными объектами, широко появились в обрабатывающей промышленности и сфере услуг.

Статистический мониторинг процессов служит эффективным инструментом поддержки точного и своевременного принятия решений в. Применение подходов к мониторингу статистических процессов для мониторинга сетей значительно облегчает раннее обнаружение потенциальных сбоев в сложных реляционных системах и, следовательно, в последние годы все чаще изучается. Выбор эффективного метода сетевого мониторинга основывается на оценке эффективности методов-кандидатов. Однако исследований по систематической оценке и сравнению методов сетевого мониторинга очень мало. В частности, возможность частого сбора данных с помощью современных измерительных устройств, как правило, приводит к возникновению автокорреляций между сетями. Тем не менее, методов оценки производительности для автокоррелированных сетей крайне не хватает. Предлагается обзор методов оценки производительности сетевого мониторинга, основанный на моделях разделяемых временных экспоненциальных случайных графов, который применим как к

независимым, так и к автокоррелированным сетям. Далее, в качестве применения предложенного метода изучается влияние пренебрежения автокорреляциями на мощность обнаружения сетевых управляющих диаграмм. Результаты моделирования показывают неблагоприятное влияние автокорреляций на показатели контрольных диаграмм Shewhart, EWMA, CUSUM для плотности сети, и в сценариях с высокой степенью автокорреляции предлагается использовать остаточную контрольную диаграмму.

Принятие цифровых решений на основе анализа информации об операционных процессах является важнейшим компонентом современного промышленного производства по сравнению с принятием решений на основе знаний предметной области и опыта в традиционном промышленном производстве [2.4, 2.15, 2.24]. Как одно из применений аналитики больших данных, статистический мониторинг процессов (SPM) играет решающую роль в оперативном выявлении моментов изменений по данным реального времени для обеспечения безотказного и экономически эффективного выполнения процесса [2.12, 2.29, 2.47]. Он служит эффективным инструментом для поддержки точного и своевременного принятия решений в промышленном производстве, который может расширить возможности интеллектуального анализа производственных предприятий при управлении технологическими процессами.

С быстрым развитием датчиков и информационных технологий, обеспечивающих промышленное производство, в производственных процессах обрабатывающей промышленности и сферы услуг генерируется большое количество сложных данных. Среди этих “больших данных” важную роль играют сетевые данные, которые представляют взаимодействия между связанными объектами. Сети распространены повсеместно, такие как датчики, передающие данные, компьютеры, обменивающиеся информацией, и сотрудники, отправляющие сообщения

по электронной почте о корпоративных операциях. Машины на заводах также подключены как сеть для обмена информацией и совместной работы [2.4]. Сети постоянно претерпевают изменения во времени, и поэтому их называют динамическими, или изменяющимися во времени сетями [2.45, 2.48]. Благодаря сетевому мониторингу обнаруживаются аномальные изменения в динамических сетях, что значительно облегчает раннее выявление потенциальных сбоев в сложных реляционных системах. Сетевой мониторинг имеет широкое применение в компьютерных, биологических и социальных сетях, таких как обнаружение мошенничества, обнаружение вторжений, патологическая диагностика и корпоративное наблюдение. Применение SPM-подходов для мониторинга динамических сетей в последние годы привлекает быстро растущий интерес исследователей.

Подходы SPM заключаются в построении контрольных диаграмм для статистических данных, обобщающих выборочные объекты. Для сетевых данных объекты могут характеризоваться сводными метриками или параметрами модели. В подходах SPM, основанных на метриках, контрольные диаграммы используются для мониторинга структурных показателей сети, таких как центральность, расстояние и показатели транзитивности [2.28, 2.34, 2.35, 2.36, 2.43]. Для подходов SPM, основанных на моделях, идея аналогична мониторингу профиля [2.26]. Сначала строятся статистические модели для объяснения сетевых данных как реализаций случайных сетей, зависящих от определенных ковариат, а затем используются методы SPM для мониторинга параметров модели [2.7, 2.8, 2.9, 2.45, 2.48, 2.50]. Большинство предыдущих методов мониторинга динамических сетей не учитывали потенциальные автокорреляции. Однако возможность частого сбора данных современными измерительными приборами, как правило, приводит к возникновению автокорреляций между сетями. В таком контексте

рискованно напрямую применять эти методы мониторинга, поскольку их эффективность неизвестна при наличии автокорреляций. Оценка эффективности методов-кандидатов на основе моделирования может послужить руководством для выбора эффективного подхода SPM. Моделирование - это эффективный инструмент для имитации операционной среды, который практичен для проверки жизнеспособности операционных стратегий в промышленном производстве [2.3]. С помощью хорошо продуманного систематического моделирования можно имитировать различные аномальные сценарии, а методы мониторинга - кандидаты можно сравнивать в различных потенциально опасных сценариях, не требуя их реального применения. Кроме того, когда предлагается новый метод мониторинга автокоррелированных сетей, мощность обнаружения может быть хорошо оценена с помощью системной оценки производительности до его применения для мониторинга реальных данных. Кроме того, оптимальные пределы управления для обнаружения определенных специфических сдвигов могут быть получены на основе оцененных характеристик путем манипулирования сценарием моделирования. Подводя итог, поскольку “истинные” сетевые процессы на практике известны очень редко и обычно очень сложны, крайне полезно проверить эффективность методов мониторинга в заранее определенных “идеальных” условиях с помощью имитационных исследований контролируемым образом [2.39, 2.46].

Было проведено несколько исследований методов оценки производительности на основе имитационного моделирования, которые разработаны для различных типов сетей, которые можно классифицировать по трем аспектам. По зависимости от времени сети подразделяются на независимые и автокоррелированные сети. По направленности границ сети классифицируются на ненаправленные и направленные сети. В зависимости от того, представляет ли граница

существование взаимодействия или количество взаимодействий, сети классифицируются на бинарные и взвешенные сети. В [2.49] смоделированы независимые неориентированные бинарные социальные сети с изменениями вероятности взаимодействия между различными пропорциями узлов на основе модели Эрдеша-Райна и оценили эффективность метода сканирования, предложенного в [2.36]. В [2.45, 2.48] предложено использование стохастических блочных моделей с поправкой на степень для моделирования индивидуальных, локальных и глобальных изменений склонностей к взаимодействию и изменения степени изменчивости, а также структурных изменений сообщества для независимых ненаправленных взвешенных сетей. В [2.14] проведено сравнение эффективности контрольных карт EWMA и CUSUM для средней степени и стандартного отклонения показателей степени при обнаружении вспышек в независимых неориентированных взвешенных сетях на основе моделей Пуассона. В [2.20] проведена статистическая оценка набора популярных методов обнаружения спектральных аномалий с помощью независимых неориентированных бинарных сетей, смоделированных на основе модели Эрдеша-Райна, рекурсивной матричной модели и модели Чунг Лу. В приведенной выше литературе были проведены исследования по оценке производительности для мониторинга независимых неориентированных бинарных и взвешенных сетей. Методы оценки для автокоррелированных сетей и направленных сетей, однако, изучены очень мало. В этой статье мы предлагаем систематическую структуру для оценки производительности контрольных диаграмм для автокоррелированных неориентированных бинарных сетей, основанную на модели разделяемого временного экспоненциального случайного графа (STERGM), предложенной в [2.22].

Обсуждения неблагоприятных последствий автокорреляций и стратегий улучшения были изучены для одномерных, многомерных и

профильных контрольных карт во многих работах [2.1, 2.6, 2.13, 2.19, 2.27, 2.32, 2.41, 2.42, 2.44]. Разумно предположить, что автокорреляции также могут оказывать неблагоприятное воздействие на мониторинг сети. Однако количественному изучению таких эффектов было посвящено немного усилий. В дальнейших исследованиях будем применять предложенный метод оценки производительности для изучения эффектов автокорреляции на сетевых контрольных диаграммах.

Дальнейшие исследования будут строиться следующим образом. Во-первых, будет предложена систематическая структура для оценки схем управления сетью на основе STERGM с гибкостью моделирования как независимых, так и автокорреляционных сетей. Во-вторых, этот метод будет использован для изучения влияния пренебрежения автокорреляциями на производительность некоторых часто используемых сетевых управляющих диаграмм. Эксперименты по моделированию показывают, что

- автокорреляции оказывают неблагоприятное влияние на характеристики графиков контроля плотности сети Shewhart, EWMA и CUSUM;

- контрольная диаграмма CUSUM, как правило, лучше обнаруживает малые и средние сдвиги в сценариях низкой и средней автокорреляции;

- меньший весовой параметр для диаграммы EWMA способствует лучшей производительности, когда автокорреляция сети невысока, а в противном случае предпочтительно большее значение;

- при наличии высоких автокорреляций предлагается таблица остаточного контроля.

1.2. Особенности сетевых схем управления

Сети обычно используются для описания взаимосвязей между различными объектами в сложных системах. Сеть состоит из набора узлов

и ребер, соединяющих пары узлов. Например, в локальной компьютерной сети каждый компьютер является узлом, и разные компьютеры образуют границы, если они передают данные друг от друга; в социальной сети каждый человек является узлом, а те, кто общается друг с другом, образуют границы; в производственной сети производственное оборудование является узлами и два оборудования образуют направленный край, если одно отправляет детали другому. На рис. 1.1 показан пример направленной компьютерной сети и неориентированной сети. В компьютерной сети пять компьютеров, проиндексированных от A до E, являются узлами. Данные передаются из A в D, A в B, E в D, A в C, C в A, A в E и E в A. Таким образом, в этой сети имеется 7 направленных ребер. В неориентированной сети L, M, P, D соседями для J; D и P также соседи. Здесь топологическая связь всегда биективна и не имеет направления. Пять узлов образуют ненаправленную сеть с 5 ребрами.

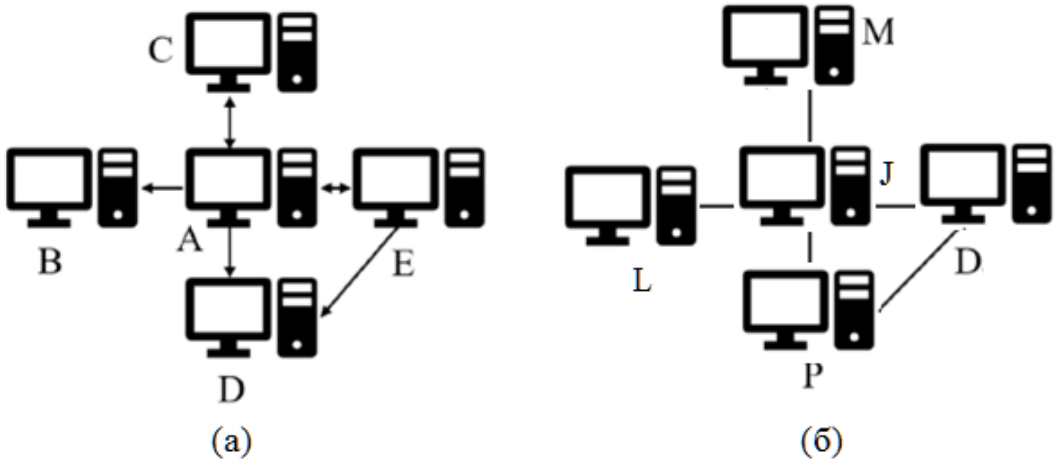
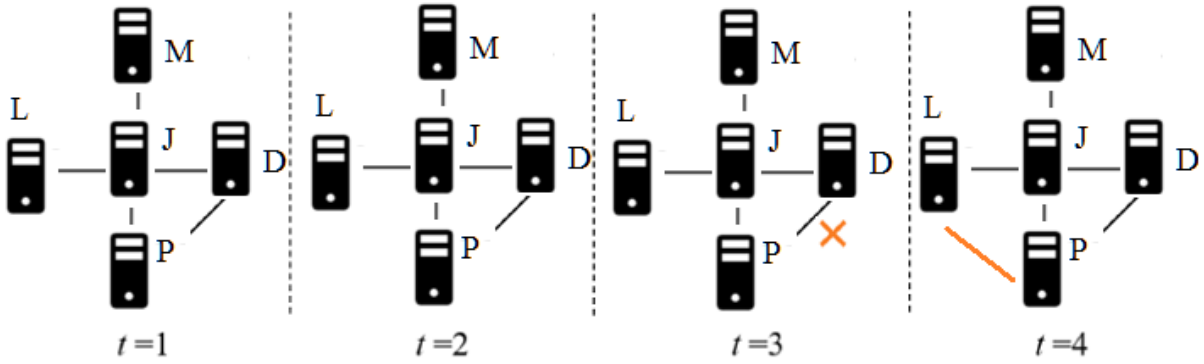


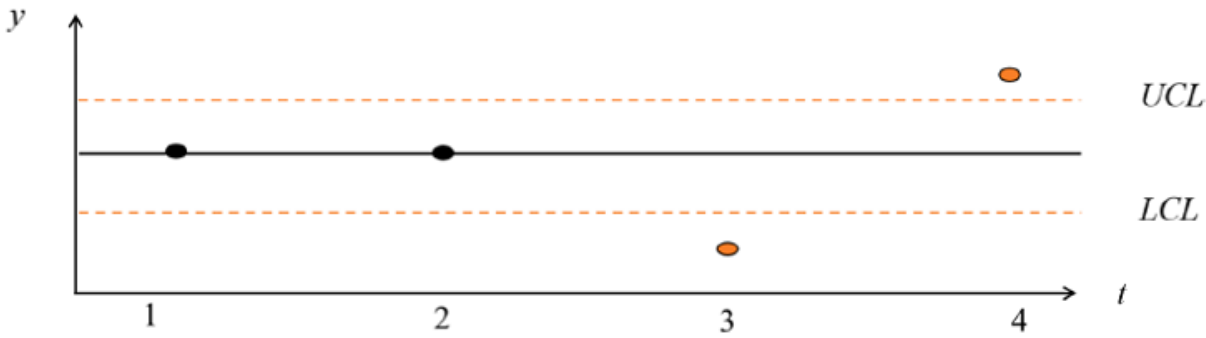
Рис. 1.1. Примеры направленной сети (а) и ненаправленной сети (б)

Контрольные диаграммы обычно используются для мониторинга того, остается ли процесс со случайными шумами статистически контролируемым с течением времени. Строится статистика графиков, такая как среднее значение выборки, статистика EWMA и статистика CUSUM. Контрольные пределы получаются на основе распределения

статистики графиков для обнаружения аномалий. Точки, выходящие за пределы допустимых значений, идентифицируются как выбросы, указывающие на возможные причины изменений процесса. Сетевая контрольная диаграмма - это применение методов построения контрольных диаграмм к сетевым данным с целью обнаружения аномальных изменений сетей. Например, рис. 1.2(а) иллюстрирует эволюцию сети, сформированной L, M, P, J и D. С момента времени t=1 по 2 сеть остается неизменной. В момент времени t=3 связь между D и P разрывается. В момент времени t=4 соединение между L и P формируется заново. Если мы беспокоимся об общей связности в сети, возможным способом является прямой мониторинг общего количества ребер, что показано на рис. 1.2(б).



(а) Изменения связей во времени



(б) Контрольная диаграмма количества связей в сети

Рис. 1.2. Пример использования контрольной диаграммы для мониторинга изменений сети

Рассмотрим крайний случай, когда допустимо только 5, а контрольные пределы установлены как 4.5 и 5.5. Тогда сети в моменты времени 3 и 4 являются выбросами, поскольку они имеют 4 и 6 ребер соответственно. Построим график количества ребер с течением времени, и тогда можно идентифицировать сеть в моменты времени 3 и 4 как выбросы.

Ожидается, что хорошая контрольная диаграмма быстро обнаружит изменения в процессе в случае выхода из-под контроля и не вызовет ложной тревоги, когда процесс находится под контролем. Однако эти две цели не могут быть достигнуты одновременно. Другими словами, увеличение производительности при контроле приведет к снижению производительности при выходе из-под контроля. Типичный способ сравнения методов состоит в том, чтобы установить одинаковые характеристики при контроле и сравнить характеристики при выходе из-под контроля при обнаружении сдвигов разного масштаба. Эффективность контрольной диаграммы обычно оценивается на основе распределения длины «пробега» (RL), которое может быть получено путем аналитического вывода и/или с помощью имитационных исследований. Для сетей со сложными структурами данных вывод теоретической формы распределения RL обычно очень сложен. Следовательно, подход к оценке производительности, основанный на моделировании, необходим для получения эмпирического распределения RL, на основе которого рассчитываются такие показатели, как средняя продолжительность «пробега» (ARL) и стандартное отклонение RL.

1.3. Проблема мониторинга производительности коллективов встраиваемых объектов больших программных систем

1.3.1. Мониторинг производительности и обнаружение аномалий

Мониторинг производительности и обнаружение аномалий являются основными задачами при проектировании и обслуживании электронных устройств и систем. В последние годы они усложняются из-за усложнения аппаратного и программного обеспечения. Следовательно, важным моментом является сбор репрезентативных выборок сигналов и выявление характерных особенностей, позволяющих оценить рабочие характеристики устройств. Это приводит к необходимости эффективного анализа временных рядов. Эта проблема рассматривается применительно к встраиваемым системам и устройствам Интернета вещей. Представлена новая схема декомпозиции временных рядов путем введения объектов более высокого уровня, ориентированных на искомые свойства системы. Они создают компактную модель состояния, которая облегчает получение знаний о поведении системы для проверки правильности ее работы. Собранные образцы объединяются в объекты в соответствии с предопределенными показателями сходства, эти объекты можно отслеживать, сопоставлять и объединять с соответствующими событиями операционного журнала. Для этой цели был составлен набор оригинальных алгоритмов, которые были включены в разработанный программный инструмент. Представленный подход был оценен на репрезентативном наборе данных, и был использован для изучения эффективности их энергопотребления.

Анализ рабочих характеристик различных физических устройств или систем предполагает измерение различных параметров (метрик) во времени. Собранные выборки данных представляют собой временные ряды (TS), которые необходимо исследовать для получения характерных

признаков, коррелирующих с исследуемыми свойствами. Для этой цели в литературе были предложены различные методы, проиллюстрированные некоторыми практическими примерами, ориентированными на конкретные области применения, например, сигналы электрокардиограммы (ЭКГ) [3.1, 3.10], распознавание деятельности человека [3.27], потребление электроэнергии [3.4, 3.13, 3.20], эконометрика, биоинформатика, контроль качества. В целом, рассматриваются три класса задач обработки сигналов: обнаружение аномалий, получение операционных профилей (извлечение поведенческих признаков) и прогнозирование будущего поведения. Во многих приложениях целесообразно выполнять аддитивную или мультипликативную декомпозицию временных рядов. Например, сезонные компоненты и трендовые компоненты получены в работе [3.8]. Это особенно полезно в таких областях, как финансы, управление услугами, запасы и т.д. [3.31]. Вывод различных классов импульсов на ЭКГ, представленный в работе [3.24], является еще одним примером декомпозиции, зависящей от приложения. Обычно анализ временных рядов выполняется во временной, частотной и амплитудной областях. Более того, он может быть одномерным или многомерным (включая корреляции с другими временными рядами). Как правило, анализ временных рядов (TS) фокусируется на отслеживании аномалий и оценке свойств устройства/системы (например, памяти, использования процессора, активности прерываний, профилей передачи, энергопотребления), которые могут быть использованы для выполнения соответствующих улучшений. Исследования по обнаружению аномалий [3.22] включают:

• Точечная аномалия - отдельный экземпляр данных рассматривается как аномалия на основе свойств остальных данных (например, чрезмерная амплитуда, слишком большая длительность импульса).

• Одномерная контекстуальная аномалия - признак аномальных

данных квалифицируется как аномалия только в определенном контексте.

• Коллективная аномалия - набор данных временных рядов, идентифицированных как аномальные по отношению ко всему сигналу. Независимо от целей анализа, практики и исследователи сталкиваются с проблемой декомпозиции сигнала, извлечения характерных особенностей и создания некоторых абстрактных моделей поведения, адаптированных к специфике анализируемой системы и искомым свойствам. Анализируя сигналы, описывающие работу встроенной системы, мы выделили четыре класса соответствующих TS: простые регулярные временные ряды (SRT), составные регулярные временные ряды (CRT), нерегулярные (IRT) и неопределенные временные ряды (IDT). Временные ряды СТО обычно демонстрируют некоторые периодические свойства с повторяющимися четко определенными формами сигналов (например, ЭКГ). ЭЛТ демонстрируют некоторую регулярность, которая включает в себя дополнительные факторы, связанные с периодами времени, внешними обстоятельствами (условиями), и с этим можно столкнуться в различных встроенных системах или устройствах Интернета вещей (IoT), выполняющих предопределенные схемы обработки. В случае IRT трудно выявить уникальную закономерность, однако ее можно рассматривать как детерминированную в отношении известных внешних взаимодействий, эти особенности не встречаются в серии IDT. Наше исследование нацелено на временные ряды SRT, CRT и IRT с периодами низкой фоновой активности и конечным набором групп выборок, описываемых некоторыми согласованными свойствами (например, определяемыми метриками сходства). Изучение таких TS сосредоточено на двух исследовательских задачах:

- Создание агрегированной, гибкой и иерархической (многоуровневой) структуры собранных выборок сигналов, описывающих работу устройства.

- Сопоставление полученных характеристик уровня сигнала с журналами работы устройства для получения более глубокого представления о поведении устройства.

Основным вкладом является представление оригинальной объектно-ориентированной модели временных рядов с разнообразными функциями, которые отличают ее от классических плоских моделей, используемых при анализе большинства временных рядов. Свойства объектов и их иерархические отношения определяются с помощью соответствующих метрик. Представленная модель обеспечивает возможность иерархического анализа данных и улучшает интерпретируемость результатов. Предлагаемый многоуровневый анализ выявляет некоторые базовые объекты во временном ряду, а затем выполняет расширенный анализ на более высоком уровне агрегированных объектов. Созданная уникальная модель конечного автомата облегчает проведение обширных исследований, поддерживающих решения по разработке в зависимости от полученных результатов оценки. Это также может быть скорректировано с учетом специфики приложения для получения более глубоких знаний о поведении устройства. Необработанные данные временных рядов преобразуются в репрезентативные структуры данных (относящиеся к введенным объектам) с информативными агрегированными характеристиками, которые улучшают процессы анализа, включая сопоставление идентифицированных объектов с журналами системы/приложений (помогая экспертам в интерпретации результатов). Объединение модели временных рядов уровня сигнала с отчетными журналами потребовало разработки новых схем временной и контекстной корреляции. Эти проблемы были проигнорированы в классических схемах анализа TS. Результатом нашего подхода стала разработка эффективных алгоритмов, которые были включены во внедренный инструмент анализа временных рядов (TS_AN). Подход может быть адаптирован для

исследования более широкого класса встраиваемых систем, охватывающих не только энергопотребление, но и другие аспекты производительности или поведения.

1.3.2. Проблема оптимизации использования ограниченных ресурсов для достижения высокой производительности, а также ограниченного энергопотребления для встраиваемых систем

Фоновые аномалии во временных рядах [3.33] обычно связаны с амплитудой, формой или временными особенностями временных сигналов. Они могут быть определены как показатели аномалий, например, минимальные, максимальные (экстремальные) значения, некоторые статистические характеристики и т.д. Более сложные функции могут включать в себя технику сегментации на основе формы [3.9] или на основе определения последовательности символов [3.25]. Символы генерируются с использованием различных методов разделения, например, с использованием дисперсии, энтропии или иерархической кластеризации. Преобразованные временные ряды в последовательности символов создают соответствующие вероятностные конечные автоматы, полезные для моделирования поведения системы (например, используемые в вычислительной технике, лингвистике и распознавании речи). В некоторых приложениях (связанных с контролем качества, финансами, эконометрикой) важно выявлять выборочные кластеры или точки изменений. В работе [3.32] точки резких изменений извлекаются из данных с помощью введенного метода сокращения. Особенности извлечения временных рядов могут быть адаптированы к их специфике, например, к набору данных электрокардиограммы [3.10]. В ЭКГ есть несколько стандартных характеристик формы сигнала, связанных с некоторыми импульсами, например, их периодичность, временные

соотношения и амплитуды. Таким образом, для их анализа могут быть использованы специализированные методы. В работе [3.21] шесть первичных волн, связанных с атрибутами P, Q, R, S, T и U, сегментированы и обнаружены, алгоритмы машинного обучения используются для классификации нормального и аномального поведения сердца. Также необходимо учитывать тенденции и сезонные особенности [3.8]. В работе [3.23] предложены различные методы спектрального анализа, например, Фурье- и вейвлет–преобразования - они разлагают анализируемый временной ряд (TS) на набор наиболее важных компонентов (например, коэффициенты Фурье с преобладающей частотой и амплитудой). Временные ряды могут быть разделены на стохастические и детерминированные компоненты, в результате чего получается комбинированная гибридная модель. В прогнозах погоды декомпозиция TS может привести к получению регулярных, периодических, стохастических и трендовых компонентов [3.2], причем линейные компоненты обрабатываются моделью ARIMA на основе временных рядов MapReduce, а нелинейные компоненты - моделью M-K-N. Было разработано множество моделей временных рядов, предназначенных для решения задач прогнозирования и контроля. Выделение признаков во временных рядах на основе окон [3.6] включает методы кластеризации и построения информационных гранул. Это полезно для наборов данных с плотной выборкой и заполнением, а также для изучения тенденций в наборах медицинских данных. В работе [3.29] авторегрессионные модели используются для прогнозирования будущих значений TS, а соответствующие показатели разброса позволяют отмечать наблюдаемые значения как аномалии. Тем не менее, проблема декомпозиции временных рядов и получения характерных признаков требует постоянных исследований (в соответствии с исследуемыми вопросами). Анализ временных рядов, основанный на классификации, использует показатели

сходства [3.7] и выявляет сходство между рядами. Различают три категории сходства: по времени (на основе корреляции), по изменению сигнала (на основе автокорреляции) и по форме (на основе shape-based). Некоторые представления TS ориентированы на сходство по форме, они имеют дело с подпоследовательностями временных рядов, называемыми шейплетами. Классификация на основе шейплетов использует сходство между шейплетами и сериями в качестве отличительного признака. Шейплеты понятны и могут дать представление о предметной области [3.17]. Во многих публикациях, посвященных классификации TS, рассматриваются различные дискриминационные признаки сходства, основанные на корреляции во времени, тенденциях изменения сигнала или его форме. В работе [3.28] авторы представляют подход скользящего окна для преобразования временных рядов в вектор признаков, используемый в классификации TS для машинного обучения. Здесь важно выбрать и сократить количество рассматриваемых признаков. Классификация TS полезна для решения некоторых задач, например, для определения профилей энергопотребления пользователей для улучшения прогнозирования будущего потребления энергии [3.19], выявления сердечных аномалий/патологии, различных задач обучения [3.3]. Схема классификации TS, приведенная в работе [3.3], извлекает информативные признаки, такие как производные, кумулятивные интегралы, спектр мощности.

Метод основан на выборочном наивном байесовском классификаторе, полученном на основе собранных данных. В [3.26] обсуждается кластеризация временных рядов для извлечения информации из данных о потреблении энергии в здании. Представленная методология позволяет строить энергетические профили, в частности, кластеры TS (паттерны), соотнесенные с несколькими часовыми диапазонами в течение дня и ночи. Ее можно использовать для предотвращения потерь энергии и

улучшения прогнозов энергопотребления. Представленный обзор литературы показывает, что большинство схем анализа временных рядов сосредоточены либо на классификации, обнаружении аномалий, либо на прогнозировании будущих проблем в поведении. Первые две задачи основаны на наличии некоторых эталонных моделей и знании предметной области, например, электрокардиограммы с четко заданными правильными формами импульсов, амплитудами и временными соотношениями. В отличие от этого, во многих встроенных устройствах и устройствах Интернета вещей поведенческие схемы относятся к типам временных рядов SRT, CRT и IRT, при этом различные действия запускаются спорадически (довольно часто на короткое время) и повторяются. Эти действия могут включать в себя работу с различными аппаратными компонентами. Даже в случае небольших устройств используются довольно сложные системы на кристалле (SoC). Они включают в себя несколько процессоров, различные периферийные устройства (например, радиомодемы, мини-дисплеи, датчики) и усовершенствованную схему управления питанием. Взаимодействие всех этих компонентов может быть довольно сложным (управляться различными программными модулями) и его трудно отследить или оценить, например, его влияние на энергопотребление (критично для устройств, работающих на автономных источниках питания). Следовательно, анализ поведения таких устройств требует выявления характерных особенностей, которые либо неизвестны, либо частично известны разработчикам/дизайнерам. Эти особенности скрыты в необработанных данных мониторинга и требуют их обнаружения. Установлено, что классические подходы, рассмотренные в литературе, не являются удовлетворительными в случае анализа временных рядов, предназначенных для описания различных характеристик производительности встроенных устройств и устройств Интернета вещей. Это связано с тем, что выборки данных могут

рассматриваться как крайне нерегулярные. Выполнение некоторой агрегации выборок создает возможность работать с более крупными компонентами (объектами), описываемыми некоторыми статистическими свойствами. Они облегчают их кластеризацию и вывод поведенческих моделей на более высоком абстрактном уровне, например, иерархических объектных моделей или моделей автоматов состояний, которые позволяют выявлять специфические закономерности (например, специфические последовательности объектов) или характерные особенности. Более того, они могут быть соотнесены с изменяющимися условиями в рамках операционного профиля. Это может быть дополнительно объединено с корреляциями в других пространственных представлениях системы (например, программный процесс, происходящие события). Такой подход к импульсно-ориентированной модели представлен в работе [3.16] для анализа аномалий сложных серверов. Однако эта модель была ориентирована на периодические характеристики импульсов и не соответствовала специфике рассматриваемых временных рядов SRT, CRT и IRT, а также ограниченным ресурсам, характерным для встраиваемых систем. Проблема оптимизации использования ограниченных ресурсов для достижения высокой производительности, а также ограниченного энергопотребления имеет решающее значение для встраиваемых систем. Последнее свойство особенно важно в беспроводных сенсорных сетях и мобильных устройствах [3.13]. В работе [3.30] предложены модели энергопотребления, основанные на динамике напряжения батареи. Они поддерживают модели энергопотребления на уровне компонентов, используемые в приложениях для подсчета энергии, ориентированных на смартфоны. В [3.4] выполняются измерения мощности на основе доступных схемотехнических решений смартфона. Измерительные резисторы подключены к шинам питания соответствующих компонентов и они измерили мощность, потребляемую компонентами устройства, для

набора синтетических программ, активирующих эти компоненты. В работе [3.1] проанализировано энергопотребление некоторых плат микроконтроллеров для выполнения базовых операций. Это включает в себя изучение влияния тактовой частоты, скорости передачи, частоты дискретизации, использования FPU и т.д. В работе [3.20] предлагается моделирование энергопотребления устройств Интернета вещей с учетом трех функциональных блоков: обнаружения и сбора данных, обработки данных и передачи данных. Это позволяет оценить энергопотребление в зависимости от различных технологических параметров. Однако для такого подхода требуется профиль использования блоков в устройстве, который трудно получить для реальных применений, тем более что современные устройства содержат гораздо больше компонентов, работающих в различных режимах энергопотребления и со сложными взаимодействиями. Многие статьи, посвященные энергопотреблению, ориентированы на уровень компонентов, в них делается попытка определить энергетическое воздействие различных компонентов (обычно некоторых базовых) для создания прикладной энергетической модели, основанной на восходящем подходе. К сожалению, как указывалось выше, объединить это с реальным применением сложно. Наш подход использует TS, представляющие текущие выборки, собранные во время работы реального устройства, а затем выполняет оценку путем декомпозиции этих данных в соответствии с внедренной гибкой и многоуровневой моделью. Полученная модель построена путем агрегирования исходных данных, поэтому она обещает быть более реалистичной, поскольку все взаимодействия на низком уровне естественным образом отражаются в данных, используемых для синтеза модели. Встраиваемые устройства производятся в больших количествах, систематически модернизируются в соответствии с новыми функциональными запросами или адаптируются к другим аппаратным платформам. Следовательно, важным вопросом

является проверка их работоспособности в реальных условиях эксплуатации. В результате их работа контролируется с помощью специального измерительного оборудования на уровне сигнала. К сожалению, из-за сложности выполняемой обработки данных отслеживание и интерпретация собранных сигналов является сложной задачей как с точки зрения обнаружения аномалий, так и с точки зрения повышения качества. Эта проблема возникла при разработке микродатчиков для микросенсорных устройств. Проанализированные TS, характеризующие работу разработанных встраиваемых устройств, показали необходимость создания нового взгляда на декомпозицию TS путем создания иерархической и объектно-ориентированной модели, включающей кластеризацию этих объектов и идентификацию поведенческих последовательностей, удовлетворяющих некоторым условиям сходства. Такая оригинальная модель упрощает работу системы отслеживания и помогает экспертам (разработчикам/дизайнерам) принимать решения по оценке/совершенствованию. Она преобразует необработанные образцы TS в компактную модель, отражающую особенности работы системы на более высоком агрегированном уровне, которую эксперту легче понять и выполнить интерпретацию. Кроме того, процесс анализа поддерживается предоставленными алгоритмами, совместимыми с моделью. Выявив подозрительную ситуацию на более высоком уровне, эксперт может детализировать ее причину, обратившись к соответствующим объектам более низкого уровня, используя встроенные механизмы разработанного программного обеспечения для анализа (TS_AN). Полезность подхода была подтверждена при анализе энергопотребления, что важно для многих встраиваемых систем и устройств Интернета вещей, работающих от батарей или собирающих энергию, которые должны работать в течение длительного времени [3.11]. Принимая во внимание тот факт, что эти устройства выполняют довольно

сложную обработку данных, используют различные аппаратные ресурсы и схемы связи, оптимизация энергопотребления является сложной задачей. Эти процессы еще более усложняются из-за возможности программирования различных уровней мощности, активации/деактивации заданной схемы [3.14]. При этом также необходимо учитывать требования к производительности (соотношение между энергопотреблением и производительностью). В этом процессе мониторинг текущего использования и анализ соответствующих временных рядов являются необходимыми исследованиями. Тем не менее, подход может быть использован и для анализа других характеристик. Основная идея заключается в разложении временных рядов на различные классы сегментов, определяемые заданными статистическими свойствами, и последующем создании модели конечного автомата более высокого уровня. Обработка данных основана на объектно-ориентированной программной парадигме, которая не встречается в литературе по TS. Такой подход облегчает выявление интересных особенностей и их интерпретацию в соответствии с функциональностью системы, операционными профилями и журналами.

1.4. Управление эластичностью для планирования мониторинговых мощностей в облачных архитектурах "программное обеспечение как услуга"

1.4.1. Управление эластичностью как многомерное планирование пропускной способности распределенной мониторинговой системы

Поскольку облачные вычисления становятся все более экономичными, компании переносят свои вычислительные потребности в мониторинге в облако. Тенденция роста рынка облачных вычислений обусловлена доступным доступом к надежному высокопроизводительному

программному и аппаратному обеспечению, отсутствием высоких затрат на техническое обслуживание и наличием высокозащищенных решений [4.34].

Учитывая эту тенденцию, перед поставщиками облачных услуг открываются огромные возможности и проблемы по мере роста конкуренции [4.10, 4.25, 4.43]. Одной из таких важнейших задач является оптимизация возможностей облачных провайдеров, чтобы обеспечить баланс между более высокими затратами на ресурсы и более высокими штрафными санкциями за несоответствие требованиям заказчика к уровню обслуживания. Улучшение этого баланса также может снизить общее энергопотребление, что имеет важные экологические преимущества. Исследование направлено на решение этой практической, чтобы помочь провайдерам определить, как использовать ресурсы для будущего планирования, используя модель облачных вычислений "программное обеспечение как услуга".

Интересуют три модели облачных сервисов: программное обеспечение как услуга (SaaS), платформа как услуга (PaaS) и инфраструктура как услуга (IaaS) [4.46, 4.56].

IaaS – облачный сервис, предоставляющий инфраструктуру (например, аппаратное обеспечение) клиентам, которые имеют полный контроль над операционными системами, хранилищем и развернутыми программными приложениями.

С помощью PaaS облачный провайдер предлагает инфраструктуру, языки программирования, библиотеки и инструменты клиентам, которые управляют развернутыми приложениями и некоторыми параметрами конфигурации.

Однако при использовании SaaS облачный провайдер управляет всем клиентским приложением, работающим в облачной инфраструктуре, и отвечает за обеспечение пропускной способности (в виде экземпляров),

достаточной для выполнения всех запросов по требованию (входящих запросов). Клиент имеет доступ к приложениям, но не контролирует возможности и пропускную способность отдельных приложений. Таким образом, в среде SaaS планирование пропускной способности чрезвычайно важно для облачного провайдера.

В практике облачных вычислений и литературе детальное многомерное планирование пропускной способности называется **управлением эластичностью** [4.28, 4.31]. Правильное управление эластичностью обеспечивает развертывание достаточного количества ресурсов (измеряемых в нескольких измерениях, таких как процессор, оперативная память (RAM), дисковое хранилище, пропускная способность сети и т.д.), чтобы приложение, развернутое в облаке, продолжало работать в соответствии с ожиданиями клиентов, независимо от изменений спроса [4.45]. Существует два распространенных типа управления эластичностью: вертикальное и горизонтальное [4.52]. Управление вертикальной эластичностью относится к увеличению или уменьшению размера одного (или нескольких) параметров ресурсов, используемых одним экземпляром приложения, например, путем увеличения объема оперативной памяти или дискового хранилища для развернутого экземпляра. Управление горизонтальной эластичностью - это метод развертывания или закрытия целых экземпляров фиксированного измерения в соответствии с поступающим спросом.

В исследовании внимание сосредоточено на автономном управлении горизонтальной эластичностью, которое позволяет поставщику облачных услуг определять план пропускной способности еще до начала горизонта планирования. Автономное управление горизонтальной эластичностью используется в облачных приложениях, которым требуется более длительное время развертывания для добавления дополнительных экземпляров при быстром изменении спроса (например, развертывание

экземпляра занимает несколько минут, в то время как запросы поступают в среднем раз в миллисекунду). Основной вопрос управления горизонтальной эластичностью заключается в следующем: сколько экземпляров следует развернуть на будущих горизонтах планирования, чтобы оптимизировать соотношение между стоимостью ресурсов развернутого экземпляра и штрафными издержками за задержку выполнения запроса? Облачному провайдеру SaaS необходимо принимать это решение для многих клиентских приложений на постоянной основе, каждый час, каждый день, 7 дней в неделю, 365 дней в году. Горизонт планирования может составлять 30 минут или несколько часов, в зависимости от предсказуемости шаблонов входящих запросов. Это позволяет по-разному прогнозировать утренний, дневной, вечерний и ночной спрос, а также учитывать различия в зависимости от дней недели. Наша исследовательская модель направлена на оптимизацию каждого из этих горизонтов планирования в отдельности, поскольку поставщик облачных услуг оптимизировал бы их на практике. Затем несколько оптимизаций по горизонтам планирования связываются и объединяются в план пропускной способности на каждый день. Для автономного управления горизонтальной эластичностью выполняются эти оптимизации и разрабатывается план пропускной способности на несколько дней или недель вперед. Любая экономия, достигнутая в рамках горизонта планирования, увеличивается за счет количества горизонтов планирования в год и количества клиентских приложений SaaS у облачного провайдера.

Из-за относительно длительного времени, необходимого, например, для развертывания, поставщик облачных услуг использует автономное управление горизонтальной эластичностью. Клиентом поставщика облачных услуг является ресурс мониторинга, а пользователями приложения являются владельцы объектов мониторинга и агенты ресурса мониторинга. Запрос состоит из запросов, которые включают в себя такие

услуги, как проверка подлинности логина пользователя, создание нового профиля пользователя, отправка подтверждения действий пользователя по электронной почте, регистрация заявки, информирование команды по рассмотрению претензий об обновлениях и анализ введенных пользователем данных. Очевидно, что некоторые из этих запросов требуют больше вычислительных ресурсов, чем другие.

Например, для рассмотрения претензии может потребоваться меньше места на диске, чем для загрузки фотографий нештатной ситуации, связанной с претензией. Оптимальное количество развернутых экземпляров на любом горизонте планирования зависит от частоты поступления/структуры различных запросов, вычислительных ресурсов, требуемых для каждого из запросов, вычислительной мощности каждого экземпляра, стоимости развертывания экземпляра и штрафных издержек за задержку, указанных в соглашении об уровне обслуживания (SLA). с клиентом. Существует множество реальных примеров SaaS-облачных вычислений, которые соответствуют исследовательской задаче.

Чтобы ответить на исследовательский вопрос и определить оптимальное развертывание экземпляра для каждого горизонта планирования, разработана оптимизационная модель, позволяющая свести к минимуму компромисс между штрафами за задержку по соглашению об уровне обслуживания и затратами на развернутые ресурсы, что ограничивает выполнение запросов экземплярами с достаточным пространством во всех измерениях емкости. Модель применяет штрафные санкции, если среднее время ожидания для конкретного типа запроса превышает порог штрафной задержки для этого конкретного типа запроса в течение периода планирования. Это типичный метод, который облачные провайдеры используют в соглашениях об уровне обслуживания для расчета штрафных санкций за время ожидания, однако он приводит к нелинейному поведению оптимизационной модели, которое, как было

показано, формально неразрешимо. Поэтому нет уверенности, что для решения этой проблемы можно найти какой-либо алгоритм за полиномиальное время. Следовательно, будем использовать два алгоритма (с нулевым ожиданием и с нулевым штрафом) для определения верхней границы количества развернутых экземпляров. Затем разрабатывается несколько свойств проблемы, которые могут быть использованы вместе с верхними границами для разработки структурированной стратегии эффективного решения проблем. Таким образом, удается оптимально решать небольшие и средние задачи, возникающие на практике. Затем формулируется автономный динамический алгоритм (OFD), основанный на решениях по управлению сроками выполнения, чтобы определить почти субоптимальное количество развернутых экземпляров для крупномасштабных задач за относительно короткое время вычислений.

Наконец, систему оптимизации расширяется путем разработки методологии, позволяющей справляться с неопределенностью спроса при поступлении запросов, используя стохастическую версию модели. Используя метод аппроксимации выборочного среднего для решения стохастической модели, подход иллюстрируется двумя различными случайно сгенерированными наборами данных и демонстрируется, что предложенный подход работает лучше, чем существующие подходы к развертыванию облачных мощностей, используемые на практике.

В работе представлены несколько ключевых материалов по облачным вычислениям и исследованиям в области планирования пропускной способности. Сначала разрабатывается модель оптимизации SaaS для планирования пропускной способности и выводятся свойства и верхние границы для этой модели. Используя эти свойства модели и верхнюю границу, разрабатывается структурированная стратегия для оптимального решения небольших и средних задач, возникающих на практике. Во-вторых, для решения крупномасштабных задач предлагается

OFD, который может быть выполнен облачными провайдерами относительно быстро, чтобы скорректировать количество экземпляров до начала периода планирования, но при этом сохранить практически нулевые штрафы за нарушение SLA. В-третьих, расширяется система оптимизации, чтобы справиться с неопределенностью спроса, разработав стохастическую версию модели и продемонстрировав, что модельные решения устойчивы к неопределенности спроса при поступлении запросов. В-четвертых, предоставляется несколько практических рекомендаций для облачных провайдеров по мере проведения ими этого исследования.

1.4.2. Состояние проблемы управления эластичностью

С расширением масштабов внедрения облачных вычислений наблюдается дальнейший рост исследований в области облачных вычислений. Ранние исследования в области облачных вычислений были сосредоточены на внедрении облачных вычислений в качестве прибыльной и эффективной структуры информационных систем [4.9]. В последнее время исследования в области облачных вычислений были сосредоточены на динамике ценообразования между облачными провайдерами и их клиентами, которые приобретают услуги облачных вычислений [4.10, 4.11]. В [4.30] описаны операции, связанные с принятием решений, связанных с облачными вычислениями, тремя участниками: поставщиками, клиентами и брокерами. Чтобы решить некоторые из этих операционных проблем, [4.4, 4.12] исследовали, как облачные провайдеры должны управлять своими новыми выпусками программного обеспечения и исправлениями для оптимального устранения дефектов и угроз безопасности в приложениях облачных вычислений SaaS. Используя аналогичное приложение для облачных вычислений SaaS, исследуем решения о пропускной способности, принимаемые

поставщиком облачных услуг, с использованием управления эластичностью.

Литература о возможностях облачных вычислений

Большая часть литературы о возможностях облачных вычислений посвящена улучшению первоначального размещения входящих запросов в экземплярах для выполнения и оптимальному использованию экземпляров после выполнения запросов. В рамках исследования по первоначальному размещению в [4.40] был представлен алгоритм размещения новых подписчиков SaaS-приложений в новых или активных в данный момент экземплярах с соблюдением требований соглашения об уровне обслуживания для текущих подписчиков, а в [4.59] использовалась эвристика автономного поиска по табу для определения оптимального экземпляра (или набора экземпляров), в котором для первоначального размещения новых входящих запросов в приложении SaaS. Используя стохастические методологии, в [4.14] был использован двухэтапный подход к стохастическим сетям массового обслуживания, которые отслеживают длину очереди ожидающих запросов в каждом экземпляре и итеративный процесс для определения того, куда отправлять запросы для выполнения в сети экземпляров.

Исследования в области использования ресурсов облачных вычислений сосредоточены на перераспределении запросов, чтобы лучше сбалансировать требования к пропускной способности между экземплярами [4.47, 4.58, 4.60] или на том, как снизить энергопотребление экземпляров в центре обработки данных [4.1, 4.5, 4.19, 4.38, 4.42]. Например, в работе [4.54] использовалась стохастическая оптимизационная модель с резервированием и обширными результатами вычислений, чтобы показать, что значительная экономия окружающей среды может быть достигнута путем выборочного включения и

выключения серверов (и соответствующих экземпляров). В [4.23, 4.24] исследовался вопрос о развертывании экземпляров SaaS при рассмотрении экземпляров, развернутых на частных серверах облачного провайдера или арендованных серверах у общедоступного поставщика IaaS, что может привести к сбоям сервера. В [4.3] оптимизированы сроки покупки предварительно настроенных серверных пакетов в долгосрочной перспективе (а не ежедневные/почасовые требования к мощности) для удовлетворения растущего спроса. В отличие от нашего исследования, в котором входящие запросы размещаются на доступных экземплярах в трех измерениях емкости, большинство этих исследовательских работ по облачным вычислениям сосредоточены на том, что экземпляры просто заняты, и не учитывают реальность множественных измерений емкости.

Литература по управлению эластичностью облачных вычислений

Работа посвящена управлению горизонтальной эластичностью для регулировки количества экземпляров с фиксированной емкостью в SaaS-приложениях. В [4.18] был проведен обзор литературы по управлению эластичностью облачных вычислений и обнаружено, что многие исследования посвящены управлению эластичностью в режиме онлайн/реактивному управлению, которое используется в приложениях, где развертывание экземпляра приложения может быть выполнено очень быстро по сравнению со скоростью входящих запросов [4.44]. Методы планирования, основанные на оптимизации, не входят в число классификаций оперативного/реактивного управления эластичностью в [4.44], поскольку такие методы более подходят для автономного планирования, как мы узнаем из литературы по планированию цепочки поставок [4.39].

Основное внимание в исследованиях по управлению автономной эластичностью уделяется прогнозированию входящих запросов и минимизации затрат на развертывание экземпляра при достижении целей SLA [4.22, 4.51, 4.55], а не более сложной задаче планирования пропускной способности, связанной с многомерной пропускной способностью.

Текущие исследования в области управления эластичностью, как правило, ограничивают цели SLA и сводят к минимуму затраты на развертывание экземпляра. При этом игнорируются более сложные проблемы, связанные с денежными штрафами за недостаточно быстрое выполнение запросов. Это может быть разумным предположением для задач управления эластичностью IaaS, но не для SaaS-приложений, таких как проблема, которая исследуется.

Более подходящим примером для исследования является [4.53], где авторы представили модель целочисленного программирования для управления эластичностью. Однако их модель только минимизировала затраты на развертывание экземпляра и определяла оптимальное развертывание на основе пиковой нагрузки. Напротив, предлагаемое исследование сводит к минимуму как затраты на развертывание экземпляра, так и ожидаемые штрафные санкции из-за задержек в выполнении запросов. Это делает проблему более общей и сложной для моделирования и решения. Другая связанная с этим исследовательская работа - [4.45], в которой авторы сформулировали модель стохастического программирования, которая минимизировала ожидаемые затраты на развертывание экземпляра и штрафные санкции за отсутствие достаточного количества активных экземпляров для выполнения запросов. Однако их модель не справляется с реалистичными задачами, поскольку она не моделирует различные типы запросов или динамику ожидания запросов, их выполнения и штрафных санкций, как мы делаем в этом исследовании.

Большинство современных исследований по управлению эластичностью облачных вычислений взяты из литературы по информатике. Таким образом, представленное исследование является важным, в нем эта проблема рассматривается в сообществе исследователей операций, где (учитывая ее актуальность для планирования производственных мощностей) она может быть лучше учтена при принятии фирмами технологических решений по эксплуатации. Некоторыми статьями по управлению эластичностью в литературе по исследованию операций являются работы [4.13, 4.17]. В работе [4.17] был разработан теоретико-игровой подход, в рамках которого изучалось влияние автоматической настройки мощности экземпляра на решения фирм-клиентов о выходе на новый рынок, а также на равновесные цены, потребительский избыток и прибыльность. В [4.13] использовалась упаковка в ячейки для моделирования политики чрезмерных обязательств по продаже ресурсов сверх установленных пределов пропускной способности, поскольку запрашиваемые мощности не всегда используются полностью. Однако в этих статьях не рассматриваются затраты на развертывание экземпляра или штрафы за задержку запроса, как это делается в этом исследовании. В таблице 1.1 приведен сравнительный анализ краткое описание данного исследования в сравнении с наиболее актуальной литературой по облачным вычислениям.

Литература по управлению сроками выполнения работ/составлению расписания

При решении задач управления сроками выполнения/планирования необходимо как установить сроки выполнения поступающих заданий, так и определить последовательность выполнения заданий, чтобы минимизировать общие штрафные санкции за задержку доставки заданий клиентам. Проблема облачных вычислений, рассматриваемая в этой статье, требует планирования поступающих запросов на оптимальные

временные интервалы в пределах горизонта планирования, чтобы минимизировать общую стоимость развертывания мощностей и штрафные санкции за превышение пороговых значений задержки (сроков выполнения) запросов.

Таблица 1.1

Краткое описание различий между наиболее актуальными работами в литературе и представленным исследованием

Атрибуты	**[4.22]**	**[4.55]**	**[4.51]**	**[4.13]**	**[4.53]**	**[4.45]**	**Настоя-щая работа**
IaaS или SaaS?	IaaS	IaaS	IaaS	IaaS	IaaS	SaaS	SaaS
Моделирование на основе оптимизации?	Нет	Нет	Нет	Да	Да	Да	Да
Обработка сложных штрафных санкций за недостаточное предоставление ресурсов?	Нет	Нет	Нет	Нет	Да	Да	Да
Обработка затрат на избыточное предоставление ресурсов?	Да	Да	Да	Нет	Да	Да	Да
Моделирование поступления и выполнения запросов?	Нет	Нет	Нет	Да	Нет	Нет	Да
CS или OR?	CS	CS	CS	OR	CS	CS	OR

Примечание: CS – «computer science», OR – исследование операций

Таким образом, эти процедуры основаны на литературе по управлению сроками выполнения (DDM) и планированию очередей, чтобы определить наилучшие методы планирования элементов, когда они могут быть отложены, и определения приоритетов, аналогично тому, как поступающие запросы помещаются в очередь в контексте облачных вычислений. В большинстве документов по планированию рассматриваются фиксированные параметры пропускной способности, в то

время как в большинстве исследований по планированию пропускной способности запросы спроса рассматриваются в агрегированном виде и не включаются процедуры планирования для них. В отличие от этого, эта статья рассматривает и то, и другое в реальном приложении с нелинейными штрафами за задержку SLA, и, по-видимому, уникальна в этом аспекте.

В [4.35] приведено краткое описание правил планирования DDM и указано, что “Не найдено ни одного правила планирования, которое было бы наилучшим для всех показателей производительности”. Ранние исследования [4.15, 4.16, 4.26] показали, что первое правило с наименьшим временем обработки (SPT) лучше всего подходит для минимизации времени ожидания задания, простоя машины и длины очереди, в то время как первое правило с наибольшим временем обработки (LPT) работает хуже всего. Исследования также показали, что SPT хорошо подходит для минимизации среднего запаздывания, поскольку он имеет тенденцию сокращать среднее время выполнения [4.35]. В [4.36, 4.50] приведена хорошая сводка эвристик планирования очередей для многих приложений. Правила приоритетной диспетчеризации в рамках планирования очередей изучались десятилетиями, и общий вывод заключается в том, что “ни одно правило не работает стабильно лучше, чем все другие правила, при различных конфигурациях магазина, условиях эксплуатации и целях производительности” [4.21].

Именно поэтому в [4.21] был предложен инновационный подход к автономному определению эффективных правил планирования отправки для конкретного приложения. Они предоставили краткое описание многих правил отправки, таких как наиболее ранний срок выполнения, SPT, минимальное время простоя, измененный срок выполнения (MDD) и другие. Они также напоминают о том, что в статичной среде SPT сводит к минимуму среднее время прохождения, среднее опоздание и среднее время

ожидания. В работах [4.6, 4.27] было обнаружено, что использование SPT-сортировки также эффективно для минимизации среднего запаздывания в высоконагруженном ресурсном центре. В [4.48] была разработана модель оптимизации обслуживания для определения оптимальной пропускной способности и указанного времени выполнения заказа с бинарным штрафом за опоздание. Предлагаемая модель отличается от [4.48] тем, что учитывается нелинейный штрафной срок и множество аспектов использования ресурсов. Кроме того, в [4.48] рассматривались только семь временных интервалов в рамках горизонта планирования, что не является разумным предположением для облачных вычислений, где используется 3600 временных интервалов в рамках горизонта планирования. В [4.61] изучались правила приоритета для планирования сроков выполнения многозадачности при различных затратах на обработку. Они использовали модель динамического программирования, которая учитывает фиксированную пропускную способность и штрафные санкции. Однако модель [4.61] применима только для решения небольших задач, и не учитывает сложные нелинейные штрафные санкции за задержку, переменную пропускную способность или множественные параметры пропускной способности, которые включены в данное исследование. В [4.2] построена модель с диспетчеризацией заказов EDD, которая сравнивает гибридную модель мониторинговой системы с более чистыми формами мониторинговых систем "Выполни немедленно" или "Выполни по графику".

1.5. Цель и задачи работы

Целью работы является разработка моделей и алгоритмов управления процессами мониторинга производительности коллективов встраиваемых объектов больших программных систем в облачных архитектурах на основе учета автокорреляции образования и распада части

локальных структур.

Дизайн исследования представлен на рис. 1.3.

Рис. 1.3. Дизайн исследования

Для достижения цели необходимо решить следующие задачи:

1. Разработать модель статистического мониторинга процессов с

большими данными в компьютерных сетях, учитывающую свойства автокорреляции процессов образования и распада части локальных структур на основе контрольных диаграмм.

2. Модифицировать объектно-ориентированную модель временных рядов результатов мониторинга для обеспечения возможности иерархического анализа данных и улучшения интерпретируемости результатов.

3. Создать оптимизационную модель и алгоритмы планирования пропускной способности облачных сервисов SaaS для нахождения компромисса между стоимостью ресурсов и штрафом за задержку выполнения в облачных приложениях "программное обеспечение как услуга" с точки зрения поставщика облачных услуг.

4. Разработать динамический алгоритм определения минимальной верхней границы нулевого штрафа, обеспечивающий получение верхней границы объема вычислительных ресурсов, еще дающей нулевой штраф.

5. Разработать архитектуру программного обеспечения оптимизации облачных сервисов SaaS для решения задач мониторинга, обеспечивающую принятие решений о допустимости конфигурации сервиса с точки зрения штрафов, и также оценку ожидаемого качества обслуживания.

2. ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ СЕТЕВОГО МОНИТОРИНГА НА ОСНОВЕ РАЗДЕЛЯЕМЫХ ВРЕМЕННЫХ ЭКСПОНЕНЦИАЛЬНЫХ МОДЕЛЕЙ СЛУЧАЙНЫХ ГРАФОВ

2.1. Разделяемые временные экспоненциальные модели случайных графов

В этом разделе мы сначала представим модель экспоненциального случайного графа (ERGM) для сетей, не зависящих от времени. Затем мы опишем STERGM для автокоррелированных сетей, которая является основой предлагаемого метода оценки производительности.

2.1.1. Модель экспоненциального случайного графа для сетей, не зависящих от времени

Очень естественно утверждать, что сети “построены” своими локальными структурами, такими как ребра, звезды и треугольники [2.38]. Общая структура сети определяется ее доминирующими локальными структурами и степенью доминирования различных локальных структур. Два модельных примера сетей с 9 узлами показаны на рис. 2.1. В левой преобладают звезды, в то время как в правой преобладают треугольные структуры.

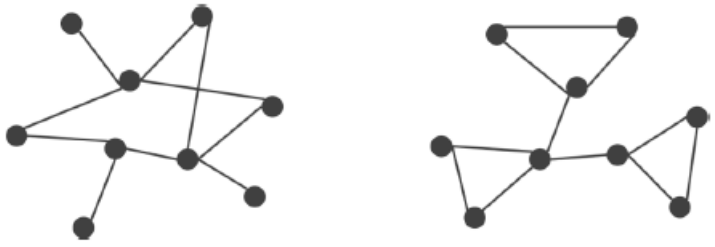


Рис. 2.1. Сети из 9 узлов, в общей структуре которых преобладают звезды (слева) и треугольники (справа)

Сводная статистика локальных структур предоставляет информацию о глобальных свойствах сетей. Например, количество ребер указывает на

общую плотность сети; количество 2-звездочек количественно определяет степень вариабельности (т.е. неоднородность узлов); количество треугольников характеризует свойство транзитивности (часто интерпретируемое как “друзья моих друзей - это мои друзья” в социальной сети) [2.10, 2.31, 2.40].

Обозначим случайную сеть с n узлами через Y и ее наблюдение через y. Запишем граничную переменную между узлами i и j в виде Y_{ij} (i, j)$\in$ {1, 2, …, n}).

В случае бинарных сетей, Y_{ij}=1 указывает на наличие ребра между i и j и Y_{ij}=0 представляет собой не-ребро. Обозначим количество ребер, 2-звездочек и треугольников в неориентированной сети Y через $S_1(Y)$, S_2 (Y) и T(Y). Эти численные параметры рассчитываются так:

$$S_1(Y) = \sum_{1\le i<j\le n} Y_{ij} \text{ - число ребер}$$

$$S_2(Y) = \sum_{1\le i\le n} \binom{Y_{i+}}{2} \text{ - число 2-звезд} \qquad (2.1)$$

$$T(Y) = \sum_{1\le i<j<h\le n} Y_{ij}Y_{ih}Y_{jh} \text{ - число треугольников}$$

Y_{i+} - степень узла i [10].

Запишем сводную статистику локальных структур в сети Y в векторном виде h(Y). Предположим, что Y, зависящий от h(Y), следует распределению экспоненциального семейства. Тогда модель экспоненциального случайного графа для случайной сети Y записывается так:

$$P(Y = y \mid \theta) = \frac{e^{\theta' h(y)}}{k(\theta, \Upsilon)} \qquad (2.2)$$

где θ - вектор параметров, соответствующий статистическому вектору h(y); Υ обозначает пространство, содержащее все возможные наблюдения

случайной сети Y; и $k(\theta, \Upsilon) = \sum_{y^* \in \Upsilon} e^{\theta' h(y^*)}$ - нормализующая константа [2.16, 2.40].

Генерируя случайные наблюдения из ERGM по модели (2.2), можно получить не зависящую от времени сетевую последовательность. Для двух независимых сетей в последовательные моменты времени вероятности структурной статистики одинаковы, а расположение локальных структур полностью случайно. Переход от Y^t к Y^{t+1} происходит через образование и распад локальных структур. Примеры структурного перехода включают формирование и разрыв ребра, формирование (k+1)-звезды из k-звезды и превращение k-звезды в (k-1)-звезду, а также формирование и превращение треугольника из и в 2-звезду.

Рис. 2.2 иллюстрирует формирование треугольника из 2-звездочки и распад 4-звездочки на 3-звездочку. Поскольку сети независимы, структурные переходы случайны, и вероятность Y^t не зависит от Y^{t-1}.

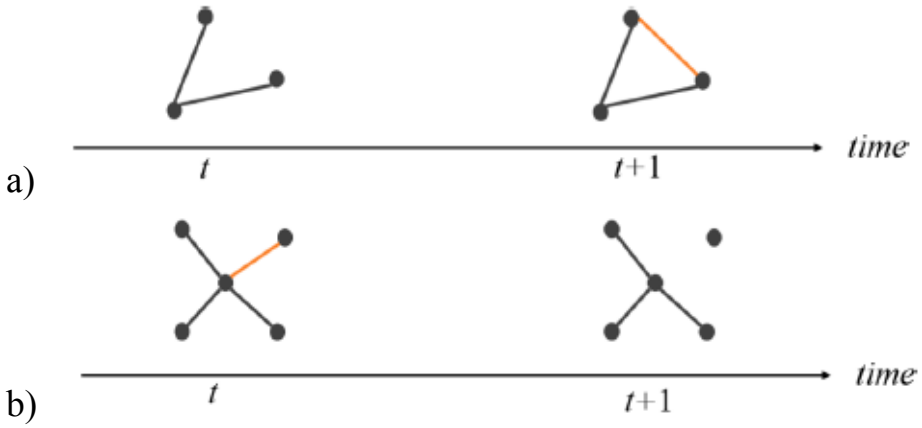


Рис. 2.2. Формирование треугольника из 2-звездочного (верх) и превращение 4-звездочного в 3-звездочный (низ) с течением времени

2.1.2. STERGM для автокоррелированных сетей

Обозначим сетевой временной ряд через $\{Y^1, Y^2, \ldots, Y^t, Y^{t+1}, \ldots\}$. Автокоррелированный сетевой процесс можно рассматривать как комбинацию двух промежуточных процессов, т.е. образования и распада

части локальных структур, при этом остальная часть остается неизменной [2.21].

Пример процессов формирования и распада границ в сети показан на рис. 2.3. Сеть взаимодействия формируется узлами L, M, P, J, D. В момент времени t пять ребер Y^t соответствуют связям между J и любым другим узлом плюс связь между P и D. Со временем шаблон связи меняется на Y^{t+1}, который включает в себя связи между J и любым другим узлом плюс связь между P и L. Такую эволюцию можно рассматривать как комбинацию двух скрытых процессов формирования и распада. Процесс формирования заключается в добавлении дополнительной связи между P и L на основе Y^t, образуя промежуточную сеть Y^+ с 6 ребрами; процесс распада заключается в разрыве связи между P и D из Y^t, пропадая в промежуточной сети Y^- с 4 ребрами.

На рис. 2.3 оператор $\cup$ обозначает объединение сетей Y и Y' как новую сеть $Y \cup Y'$ таким образом, что все ребра (i, j) либо в Y, либо в Y' помещаются в сеть $Y \cup Y'$. Оператор $\cap$ обозначает подстановку ребер (i, j) как в Y, так и в Y' в новую сеть $Y \cap Y'$. Тенденция к тому, что ребро остается неразрешенным, отражает уровень автокорреляции. Рассмотрим связь между P и M. Если их связь, как правило, остается неизменной в течение длительного времени, то их связь сильно автокоррелирована. Такое постоянство, или, скажем, продолжительность такой связи, положительно отражает уровень автокорреляции. Аналогично, автокорреляционная зависимость других свойств сети может быть охарактеризована образованием и растворением соответствующих локальных структур.

Благодаря Y^+ и Y^- процессы образования и распада могут быть непосредственно смоделированы как два ERGM:

$$P(Y^+ = y^+ \mid Y^t = y^t; \theta^+) = \frac{e^{\theta^+ h^+(y^+)}}{k(\theta^+, \Upsilon^+(y^t))}$$
$$P(Y^- = y^- \mid Y^t = y^t; \theta^-) = \frac{e^{(\theta^- h^-(y^-))}}{k(\theta^-, \Upsilon^-(y^t))} \quad (2.3)$$

$Y^+=Y^t \cup Y^{t+1}$ и $Y^-=Y^t \cap Y^{t+1}$ есть случайные промежуточные сети и y^+ и y^- есть их реализации; θ^+ и θ^- есть параметры образования и распада соответственно; $\Upsilon^+(y^t) = \{y' : y^t \in y'\}$ обозначает пространство всех возможных сетей, содержащих y^t как подмножество, и $\Upsilon^-(y^t) = \{y' : y' \in y^t\}$ обозначает пространство всех возможных сетей, которые являются подмножеством в y^t; $k(\theta^+, \Upsilon^+(t))$ и $k(\theta^-, \Upsilon^-(t))$ есть нормализующие константы [2.22]. Здесь, для упрощения обозначений, индекс времени t в сетях формирования и распада $Y^{t,+}$ и $Y^{t,-}$ опущен.

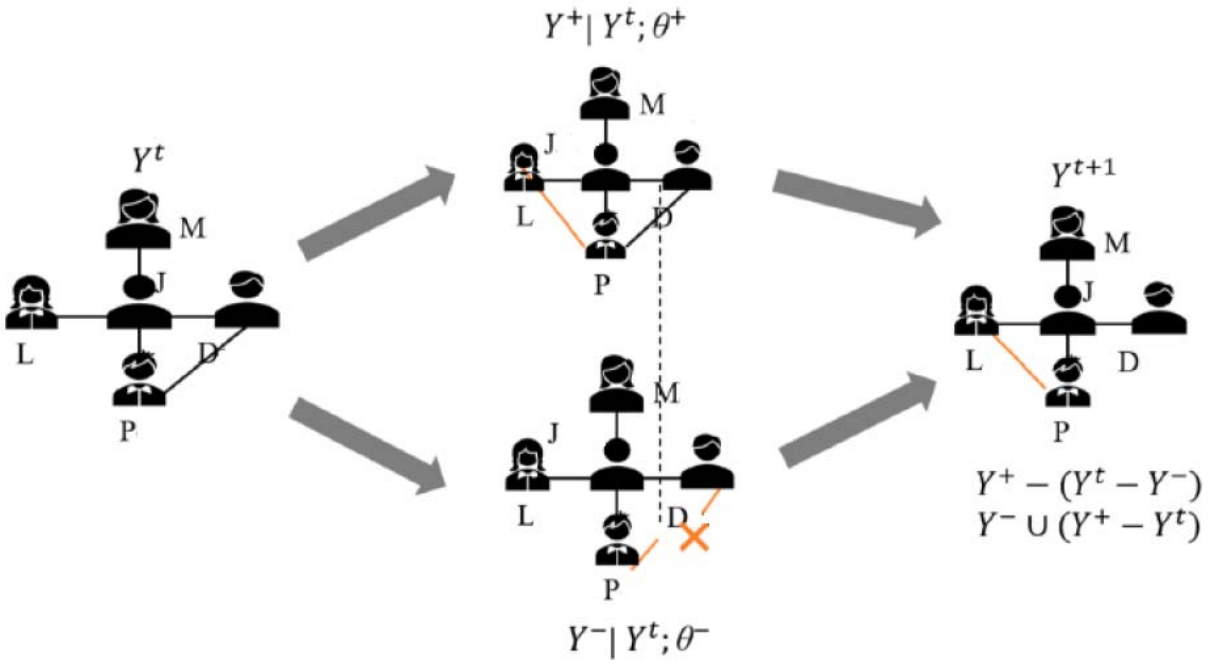


Рис. 2.3. Иллюстрация процесса автокоррелированной сети в виде комбинации процессов создания и распада

Параметры создания и распада указывают на тенденцию соответствующих структур к новому созданию и распаду с течением времени. Следовательно, увеличение параметров создания при

неизменных параметрах распада приводит к увеличению количества локальных структур, а уменьшение параметров формирования приводит к уменьшению количества локальных структур. Путем корректировки значений параметров моделируются сдвиги относительно уровней плотности, взаимности, изменчивости степени и свойств транзитивности сети. Для получения статистических данных, описывающих различные структурные характеристики, можно обратиться к [2.16, 2.31, 2.38, 2.40].

2.2. Оценка параметров

Поскольку нормализующие константы $k(\theta^+, \Upsilon^+(t))$ и $k(\theta^-, \Upsilon^-(t))$ трудно вычислимы и не представляют для нас интереса, параметры могут быть оценены путем повторного выражения модели (2.3) в форме условной логистической регрессии с удалением нормализующего постоянного члена из моделей. Определяя вероятность Y_{ji}^+ в остальной части сети Y_{ji}^{+c}, модель (2.3) эквивалентна

$$\begin{aligned} \log it(Y_{ji}^+ = 1 \mid y_{ji}^{+c}, \theta^+) &= \theta^{+\prime}\delta(y_{ji}^+) \\ \log it(Y_{ji}^- = 1 \mid y_{ji}^{-c}, \theta^-) &= \theta^{-\prime}\delta(y_{ji}^-) \end{aligned} \qquad (2.4)$$

где logit представляет logit-функцию, определенную как $\log it(X=1) = \log \frac{P(X=1)}{P(X=0)}$, которая вычисляет логарифмические коэффициенты двоичной переменной X, принимающей значение 1; θ^+ и θ^- - параметры образования и распада, подлежащие оценке; ^{c} - оператор дополнения, y_{ji}^{+c} представляет оставшуюся структуру в y^+ со статусом ребра y_{ji}^c между узлами i и j; $\delta(y_{ij})$ называется статистикой изменения для пары узлов (i, j), соответствующей изменению статистики h(y), вызванному образованием ребра y_{ji}^c между узлами i и j, заданным в [2.22].

Когда h(y) представляет собой количество ребер, статистика

изменений равна 1, что означает, что количество ребер всей сети увеличивается на 1 из-за существования ребра y_{ij}. Когда h(y) - это количество треугольников, значение $\delta(y_{ij})$ зависит от y^c_{ji} и равно количеству вновь сформированных треугольников с узлами i и j, соединяющимися из состояния без ребер.

Параметры θ^+ и θ^- могут быть оценены методом максимального приближения к совместному распределению вероятностей набора случайных величин. Единственное различие в оценке θ^+ и θ^- заключается в использовании выборок $\{y^{t,+}\}_{t=1,2,\ldots}$ или выборок $\{y^{t,-}\}_{t=1,2,\ldots}$. Обозначим $\pi_{ij}(\theta)$ как условную вероятность наличия ребра между узлами i и j, т.е.

$$\pi_{ij}(\theta) = P(Y_{ij} = 1 \mid Y^c_{ij}, \theta) = \text{logit}^{-1}\left(\sum_{r=1}^{R} \theta_r \delta_r(Y_{ij})\right) \tag{2.5}$$

где r - индекс, соответствующий структурной статистике в векторе h(Y); logit^{-1} - обратная логистическая функция, такая, что $\text{logit}^{-1}(X) = 1/(1+e^{-X})$. Мы рассматриваем временную зависимость первого порядка для сетевого временного ряда $\{Y^1, Y^2, \ldots, Y^{T+1}\}$. Вероятность приближения к совместному распределению вероятностей набора случайных величин равна

$$l(\theta) = \sum_{t=1}^{T} \sum_{i,j} \ln\left[\left(\pi^t_{ij}(\theta)\right)^{Y^t_{ij}} \left(1 - \pi^t_{ij}(\theta)\right)^{1-Y^t_{ij}}\right] \tag{2.6}$$

где Y^t заменено на $y^{t,+}$ or $y^{t,-}$, и максимальное приближение к совместному распределению вероятностей набора случайных (MPLEs) для θ^+ и θ^- получено как [2.25]

$$\hat{\theta}^+ = \arg\max_{\theta^+} l(\theta^+),$$

$$\hat{\theta}^- = \arg\max_{\theta^-} l(\theta^-).$$

Параметры образования и распада θ^+ и θ^- могут быть оценены на основе заданных выборок сети. Также временные ряды сети могут быть

сгенерированы из модели (2.3) с заданными θ^+ и θ^-. Сеть в момент времени t+1 получается путем объединения двух промежуточных сетей в виде

$$y^{t+1} = y^+ - (y^t - y^-) = y^- \cup (y^+ - y^t).$$

2.3. Сравнение схем сетевого управления

Важной частью системы оценки эффективности является построение контрольной диаграммы на основе смоделированных данных. В этом разделе представлены некоторые часто используемые контрольные диаграммы, чтобы проиллюстрировать методы получения контрольных пределов. Хотя для мониторинга сетей с более сложной структурой было предложено много передовых методов, сосредоточимся на контрольных диаграммах для самого основного свойства сети, т.е. плотности. Благодаря этому исследованию простой структуры может быть четко выявлена закономерность автокорреляции, влияющая на показатели мониторинга. Контрольные диаграммы Шухарта, EWMA и CUSUM очень часто использовались в предыдущих исследованиях по мониторингу сети. В этом разделе мы кратко опишем эти контрольные диаграммы для измерения количества ребер.

Также предлагается остаточная контрольная диаграмма с учетом информации об автокорреляции.

2.3.1. Контрольные диаграммы Шухарта для определения плотности сети

Количество ребер h^t для сети y^t отражает свойство плотности сети и обычно не очень мало в сети с частыми связями. Количество ребер h^t можно напрямую отслеживать с помощью контрольной диаграммы Шухарта. В предположении, что h^t приблизительно соответствует нормальному распределению $N(\mu, \sigma^2)$, контрольные пределы для количества ребер h^t равны

$$UCL = \hat{\mu} + L\hat{\sigma}$$
$$LCL = \hat{\mu} - L\hat{\sigma} \qquad (2.7)$$

где $\hat{\mu}$ и $\hat{\sigma}$ - оценки среднего μ и стандартного отклонения σ; L - верхний квантиль $\alpha/2$ в стандартном нормальном распределении с учетом ошибки α типа I [2.30]. Обычно среднее μ оценивается как среднее h^1, h^2, …, h^T, т.е. $\hat{\mu} = \bar{h} = \frac{1}{T}\sum_{t=1}^{T} h^t$. Для оценки $\hat{\sigma}$ могут быть применены различные методы. Здесь используется самый простой из них, т.е. стандартное отклонение выборки

$$\hat{\sigma} = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}(h^t - \bar{h})^2}\,.$$

2.3.2. Контрольная диаграмма EWMA для плотности сети

По сравнению с контрольной диаграммой Шухарта, контрольная диаграмма EWMA более выгодна при обнаружении небольших сдвигов в за счет включения информации из предыдущих данных. Статистика EWMA z_t в момент времени t зависит от текущего количества ребер h^t и его предыдущего значения z_{t-1}, т.е.

$$z_t = \lambda h^t + (1-\lambda) z_{t-1}$$

где λ ($0<\lambda\leq 1$) - весовой параметр, а начальное значение z_t обычно выбирается как $z_0=\mu$ [2.37]. Обычно весовой параметр λ выбирается в качестве одного из нескольких часто используемых значений, таких как 0,05, 0,1 и 0,2.

Контрольными пределами для z_t являются

$$UCL_{ewma} = \mu + L\sigma\sqrt{\frac{\lambda}{2-\lambda}\left[1-(1-\lambda)^{2t}\right]}$$
$$UCL_{ewma} = \mu - L\sigma\sqrt{\frac{\lambda}{2-\lambda}\left[1-(1-\lambda)^{2t}\right]} \qquad (2.8)$$

где L - константа, приводящая к заранее заданной контрольной средней

длине работы (ARL_0) для контрольной диаграммы; μ и σ оцениваются как среднее значение выборки и стандартное отклонение выборки h^t (t = 1, 2, ...,T).

2.3.3. Контрольная диаграмма CUSUM для плотности сети

Аналогично контрольной диаграмме EWMA, контрольная диаграмма CUSUM использует исторические данные и обычно превосходит их также в обнаружении небольших сдвигов. Обозначим статистику CUSUM для обнаружения положительных и отрицательных сдвигов в момент времени t через C_t^+ и C_t^- соответственно. Статистики CUSUM C_t^+ и C_t^- вычисляются как

$$\begin{aligned} C_t^+ &= \max\left\{0, h^t - (\mu + K) + C_{t-1}^+\right\} \\ C_t^- &= \max\left\{0, (\mu + K) - h^t + C_{t-1}^-\right\} \end{aligned} \tag{2.9}$$

где контрольное значение K представляет собой половину обнаруживаемых целевых сдвигов; μ обычно оценивается как среднее значение h^t (t = 1, 2, ...,T) [2.33]. Контрольные пределы для C_t^+ и C_t^- представляют собой постоянную величину d, которая может быть рассчитана на основе моделирования для достижения заранее заданного значения ARL_0.

2.3.4. Диаграмма остаточного контроля с учетом автокорреляции

Основополагающим предположением для приведенных выше трех контрольных диаграмм является то, что выборки независимы с течением времени. Чтобы учесть автокорреляцию между последовательными выборками, предлагается стратегия моделирования автокорреляции с помощью модели временных рядов и мониторинга остатков с помощью контрольных диаграмм Шухарта, которая первоначально была предложена

в [2.2]. Учитывая случай, когда количество ребер для выборок сети приблизительно соответствует модели AR(1), количество ребер в момент времени t зависит от количества ребер в момент времени t-1, т.е.

$h^t-\mu=\phi(h^{t-1}-\mu)+\varepsilon_t$

где ϕ - коэффициент автокорреляции, и $\varepsilon_t \approx N(0,\sigma_e^2)$. Оценка параметра $\hat{\phi}$ может быть получена с помощью метода наименьших квадратов или методов максимального правдоподобия. Мы отслеживаем остатки в момент времени t:

$$e_t = h^t - \hat{\mu} - \hat{\phi}(h^{t-1} - \hat{\mu})$$

Контрольными пределами для остатков являются

$$UCL_{resid} = L_e\hat{\sigma}_e$$
$$LCL_{resid} = -L_e\hat{\sigma}_e$$

где $\hat{\sigma}_e$ оценивается как стандартное отклонение e_t (t = 1, 2, ...,T) от выборок фазы I; L_e может быть получен с помощью моделирования для достижения заранее заданного ARL_0.

2.4. Основанный на STERGM подход к моделированию для оценки производительности

Сетевые контрольные диаграммы предназначены для своевременного обнаружения сетевых аномалий. Для оценки производительности сетевой контрольной диаграммы можно смоделировать временные ряды сети с аномалиями в определенные моменты времени, чтобы проверить способность контрольной диаграммы сигнализировать о таких аномалиях. В этом разделе сначала представлены способы управления автокорреляцией и уровнями сдвига сетей. Затем предоставлена основанная на STERGM структура для оценки производительности сетевых управляющих диаграмм.

2.4.1. Управление уровнями автокорреляции и сдвигами

Генерация сетей из STERGM определяется тремя составляющими: параметрами образования и распада θ^+ и θ^-, а также сетью y^t, на основе которой генерируется y^{t+1}. Чтобы смоделировать временные сети для оценки производительности, обсудим связи между параметрами STERGM и автокорреляцией процесса, а также аномальные типы.

Сетевые структуры эволюционируют с течением времени на основе комбинированных процессов формирования и распада. Параметр формирования θ^+ соответствует склонности доминирующих локальных структур формироваться в скрытой сети Y^+. Исходя из модели (2.4), положительное значение θ^+ указывает на большую вероятность формирования сетевой структуры из состояния несуществования, чем сохранения неизменной. Чем больше θ^+, тем больше вероятность образования структуры. Аналогично, параметр распада θ^- объясняет вероятность сохранения существующей структуры, а не распада. Меньшее значение θ^- указывает на более высокую вероятность распада и меньшую тенденцию к сохранению. Положительная авторегрессия может характеризоваться сохранением сетевых структур [2.25]. Чем долговечнее структура, тем более автокоррелированной является сеть. Как упоминалось в разделе 2.2, параметры формирования и растворения также влияют на уровень сводной статистики для характеристики свойств сети. Таким образом, уровень автокорреляции временных рядов сети и уровень сдвига свойств сети следует хорошо контролировать, регулируя параметры формирования и растворения.

Когда промежуточные сети для процессов образования и распада относятся к одному и тому же распределению (т.е. $Y^+ = Y^- = Y^t = Y^{t+1}$), такой динамический процесс фактически является независимым процессом. Параметры имеют соотношение $\theta^+ = \theta^- = \theta$, где θ - параметр ERGM для сети Y^t ($t = 1, 2, ...,T$). Увеличивая параметр растворения θ^- от θ,

мы устанавливаем автокорреляцию сетевых процессов на разные уровни.

По сути, моделирование временных рядов сети для оценки производительности включает в себя установку уровней автокорреляции и уровней сдвига значений локальной структуры. Поскольку регулировка уровня автокорреляции с помощью параметра распада также изменяет количество выявленных локальных структур, необходимо соответствующим образом установить θ^+ в процессе формирования, чтобы исправить изменения количества структур в процессе растворения. Чтобы поддерживать количество локальных структур на определенном уровне, можно установить параметр формирования θ^+ таким образом, чтобы статистика изменений $\delta(y_{ij}^+) = \delta(y_{ij}^-)$, означающая увеличение числа локальных структур из-за образования, была равна статистике уменьшения из-за распада.

2.4.2. Система оценки эффективности, основанная на STERGM

При статистическом мониторинге процессов ожидается, что хорошо выполненная контрольная схема (1) вызовет ложную тревогу с очень малой вероятностью в состоянии "под контролем" и (2) своевременно обнаружит сдвиги в процессе. Обычно используемым критерием для оценки производительности является средняя продолжительность выполнения (ARL) контрольной диаграммы, которая сигнализирует о выбросе. ARL контрольной диаграммы в статусах "под контролем" и "вне контроля" обычно обозначаются ARL_0 и ARL_1. Ожидается, что значение ARL_0 контрольной диаграммы будет большим, в то время как значение ARL_1, как ожидается, будет небольшим. Для оценки и сравнения характеристик контрольных схем устанавливаются контрольные пределы для разных контрольных схем таким образом, чтобы достичь одинакового значения ARL_0 для процесса, находящегося под контролем, так что их характеристики сравниваются через значения ARL_1 для неконтролируемых

процессов.

Для простоты иллюстрации сосредоточимся на изменении плотности сети в неориентированной сети и смоделируем сети со сдвигами количества ребер. Уровень автокорреляции регулируется с помощью параметра растворения θ^-, а количество ребер поддерживается на целевом уровне с помощью параметра формирования θ^+. Запишем общее количество пар узлов как $C_n^2 = n(n-1)/2$. Параметр формирования ребер задается как

$$\theta^+ = -\log\left(\frac{1+e^{\theta^-}}{m/(C_n^2 - m)} - 1\right) \tag{2.11}$$

для достижения ожидаемого количества ребер m [2.23]. Обозначим количество ребер выборки сети y^t через h^t и статистику построения графиков для мониторинга y^t через z^t. Структура оценки производительности для графиков управления сетью следующая.

Настройки Фазы I (в моменты времени 1, 2, 3, …,T)

(a) *Начальная настройка:*

Зададим количество узлов n и ожидаемое количество ребер m. Случайным образом сгенерируем исходную сеть y^1 с n узлами и m ребрами. Подберем модель (2.2) к y^1 и оценим параметр θ. Установим параметр распада $\theta_0^- = \hat{\theta}$ для независимого процесса или $\theta_0^- > \hat{\theta}$ для процесса с положительной автокорреляцией. В соответствии с (11) определим параметр образования $\theta_0^+ = -\log\left(\frac{1+e^{\theta^-}}{m/(C_n^2 - m)} - 1\right)$ для сетей внутреннего контроля.

(b) *Генерация сети*

Сформируем две случайные промежуточные сети $y^{1,+}$ и $y^{1,-}$ из распределений вероятностей $P(Y^{1,+} = y^{1,+} | Y^1 = y^1;\ \theta_0^+)$ и $P(Y^{1,-} = y^{1,-} | Y^1 = y^1;$

θ_0^-), основанных на модели (3). Получим выборку сети y^2 в момент времени t=2 как $y^2 = y^{1,+} - (y^1 - y^{1,-})$.

Повторим построение промежуточных сетей и генерацию выборок сети y^{t+1} на основе выборки y^t для t = 2, 3, ..., T.

(c) *Оценка пределов контроля*

Вычислим количество ребер для каждой выборки сети и получим последовательность ребер $h^1, h^2, \ldots, h^T$. Оценим контрольные пределы UCL и LCL для статистики построения графика z^t.

Настройки Фазы II (в моменты времени T+1, T+2, T+3, …)

(a) *Начальная настройка:*

Зададим ожидаемое количество ребер как m_*. Сгенерируем сеть y_*^T с n узлами и m_* ребрами. Зададим параметр $\theta_*^- = \theta_0^-$ распада ребер и рассчитаем параметр формирования ребер как $\theta_+^+ = -\log\left(\frac{1+e^{\theta_*^-}}{m_*/(C_n^2 - m_*)} - 1\right)$.

(b) *Генерация сети:*

Создадим две случайные промежуточные сети y_*^{T+} и y_*^{T-} из распределений вероятностей $P(Y^{T,+} = y_*^{T,+} \mid Y^T = y_*^T; \theta_*^+)$ и $P(Y^{T,-} = y_*^{T,-} \mid Y^T = y_*^T; \theta_*^-)$. Получим выборку сети y^{T+1} как $y^{T+1} = y_*^{T,+} - (y_*^T - y_*^{T,-})$.

Вычислим статистику построения графика z^{T+1}. Запишем длину прогона как RL=1, если z^{T+1}>UCL или z^{T+1}<LCL. В противном случае создадим промежуточные сети $y^{T+1,+}$ и $y^{T+1,-}$ на основе модели (2.3) с параметрами θ_*^+, θ_*^- и выборку y^{T+1}.

Повторим построение новых сетей y^t и вычисление z^t для t=T+2, T+3, ... до тех пор, пока z^t>UCL или z^t<LCL и не будет получено значение R=t-T.

(c) *Оценка длин прогонов:*

Повторим шаг генерации сети N раз и получим N длин прогонов. Рассчитаем ARL как $ARL = \frac{1}{N}\sum_{i=1}^{N} RL_i$ для оценки производительности схемы управления сетью.

Эта система оценки производительности на основе STERGM может использоваться для моделирования сдвигов процессов и изменений автокорреляции. Технологический сдвиг количества ребер вводится путем установки $m_* \neq m$. Когда $\theta_*^- \neq \theta_0^-$, имеется изменение автокорреляции количества ребер. В следующем исследовании, посвященном оценке влияния пренебрежения автокорреляцией на производительность мониторинга сети, устанавливаем $\theta_*^- = \theta_0^-$, предполагая, что автокорреляция сетевых структур не меняется.

2.5. Имитационные эксперименты

Чтобы исследовать влияние автокорреляций на контрольные диаграммы сети, используем платформу моделирования на основе STERGM для оценки характеристик контрольных диаграмм Шухарта, EWMA и CUSUM для подсчета ребер в сценариях низкого, среднего и высокого уровней автокорреляции, а также в независимом сценарии. Для контрольных диаграмм EWMA для исследования используются обычно используемые константы 0,05, 0,10 и 0,20 для λ. Для контрольных диаграмм CUSUM используем C_t^+ и C_t^- статистику для отслеживания положительных и отрицательных сдвигов соответственно. В независимом сценарии таблица остаточного контроля эквивалентна контрольной таблице Шухарта и, следовательно, не исследуется.

2.5.1. Параметры эксперимента

Зададим количество узлов равным n=50, а ожидаемое количество ребер равным m=50. На этапе I случайным образом генерируем неориентированную сеть из 50 узлов и 50 ребер в качестве исходной сети y^1, как показано на рис. 2.4(а). Приводим модель (2) к y^1, задавая сетевую статистику в виде количества ребер, т.е. h^1=50. Параметр модели θ оценивается как -3,16 методом оценки максимального правдоподобия с помощью пакета R “ergm” [2.11, 2.17].

а б

Рис. 2.4. Графы а) начальной сети y^1 и б) 1000-й сети на фазе I по независимому сценарию

Устанавливая $\theta_0^+ = \theta_0^- = \theta = -3.16$, получаем независимую сетевую последовательность. Для автокоррелированных сценариев устанавливаем θ_0^-=-1, 0, log(30), что соответствует низкой, средней и высокой нагрзке ребра. Параметры создания рассчитаны как θ_0^+=-3,44, 3,83, 6,95 на основе (11), чтобы обеспечить ожидаемое количество ребер, равное m. Далее мы генерируем сети для анализа фазы I на основе модели (2.3) и оцениваем среднее значение и дисперсию числа ребер в сценариях с различными уровнями автокорреляции. Чтобы уменьшить влияние вариаций, связанных с оценкой параметров, на рабочие характеристики, размер выборки на этапе I устанавливается равным 1000. Сетевые временные

ряды для четырех уровней автокорреляции моделируются с помощью пакета R “tergm” [2.23].

На рис. 2.4(б) показана смоделированная 1000-я сеть в независимом случае. Независимость подсчетов ребер для временных рядов сети с параметром распада -3,16 подтверждается тестом Юнга-Бокса. Коэффициенты частичной автокорреляции первого порядка для подсчета ребер для четырех уровней сетевой автокорреляции равны 0, 0,22, 0,47, 0,97, как указано в табл. 2.1. Чтобы гарантировать сопоставимость этих контрольных карт, получаем пределы контроля путем моделирования сетей внутреннего контроля для достижения ARL_0=200. Точные значения ARL_0 для этих контрольных карт приведены в табл. 2.3. В табл. 2.2 перечислены оценки параметров построения графиков, полученные для четырех типов контрольных графиков в сценариях с отсутствием, низкой, средней и высокой автокорреляциями. Чтобы проиллюстрировать метод получения параметров построения графиков, приводим примеры кодов R в приложении В. В этом примере показан алгоритм вычисления L для контрольной диаграммы Шухарта при сценарии средней корреляции. Используя этот пример в качестве справочного, параметры построения графиков для других контрольных графиков могут быть получены аналогичным образом.

Следующим шагом является оценка характеристик четырех контрольных диаграмм на этапе мониторинга II. Рассчитываются ARL для различных сдвигов процесса. Устанавливаем сдвиги подсчета ребер как δ= ±10% m, ±25% m, ±40% m.

Учитывая как положительные, так и отрицательные сдвиги, случайным образом генерируем сети с ожидаемым количеством ребер 55, 62, 70, 45, 38, 30 в качестве исходных сетей для мониторинга фазы II. Параметры формирования ребер рассчитываются на основе сдвинутых ожидаемых значений количества ребер, при этом параметры распада

остаются такими же, как на этапе I. Для каждой комбинации уровней автокорреляции и уровней количества ребер моделируем временные ряды сети на основе модели (2.3) и повторно подключаем RL. Запускаем моделирование 1000 раз и получаем значения ARL_1 в каждом неконтролируемом сценарии.

Чтобы проиллюстрировать, как контрольные диаграммы обнаруживают сдвиги, моделируем четыре набора данных, соответствующих четырем сценариям автокорреляции, основанным на STERGM. В каждом сценарии случайным образом моделируем 30 сетей без сдвигов для иллюстрации состояния под контролем. Затем моделируем 30 сетей с наибольшим положительным сдвигом, как было установлено в экспериментах по мониторингу фазы II. Параметры построения графиков приведены в таблице 2.2.

На рис. 2.5-2.8 приведены диаграммы Шухарта, EWMA (λ=0,05), CUSUM и остаточного контроля для количества ребер в различных сценариях автокорреляции. На контрольных диаграммах Шухарта можно увидеть скачок после 30-й сети на четырех графиках, особенно в сценарии с высокой автокорреляцией.

Таблица 2.1

Настройки параметров формирования и распада для различных уровней автокорреляции, а также коэффициента частичной автокорреляции первого порядка для подсчета

Сценарий	Создание	Распад	Коэффициент PACF
None	−3.16	−3.16	0
Low	−3.44	−1	0.22
Medium	−3.83	0	0.47
High	−6.59	3.4	0.97

Примечание: PACF - частичная автокорреляционная функция

Аномальные сдвиги обнаруживаются почти сразу во всех сценариях, поскольку сдвиги очень велики. На контрольных диаграммах EWMA и

CUSUM наблюдаем постепенную задержку в обнаружении сдвига с увеличением автокорреляции. На графиках остаточного контроля можно видеть, что обнаружение сдвигов происходит более своевременно по мере увеличения автокорреляции.

Таблица 2.2

Оценки параметров построения графиков для различных контрольных диаграмм в сценариях с отсутствием, низкой, средней и высокой автокорреляциями

Диаграмма	Параметры	Нет	Низкий	Средний	Высокий
Шухарт	UCL	69.00	69.00	69.00	64.00
	LCL	31.00	31.00	31.00	35.99
EWMA-Lσ	λ=0.05	15.83	19.08	23.94	47.76
	λ=0.10	17.10	20.60	25.30	42.14
	λ=0.20	18.24	21.57	25.74	34.00
CUSUM-h	Положительные смещения	4.57	4.48	8.50	15.80
	Отрицательные смещения	4.36	4.26	8.02	16.80
Остаточный	UCL	–	18.99	17.12	5.20
	LCL	–	−18.99	−17.12	−5.20

2.5.2. Анализ результатов эксперимента

Значения ARL диаграмм Шухарта, EWMA, CUSUM и остаточного контроля приведены в табл. 2.3 под каждой комбинацией технологических сдвигов и уровней автокорреляции. Сравнивая значения ARL_1 по уровням автокорреляции в табл. 2.3, можно видеть, что существует общая тенденция к увеличению ARL_1 с увеличением автокорреляции для трех типов контрольных диаграмм, которые не учитывают информацию об автокорреляции. Это указывает на то, что их мощность обнаружения уменьшается с увеличением автокорреляции. В частности, значения ARL_1 контрольных диаграмм Шухарта и EWMA для обнаружения малых и средних сдвигов (т.е. m_*= 55, 62, 45, 38) резко возрастает, когда уровень автокорреляции повышается от среднего до высокого.

Таблица 2.3

ARL различных контрольных диаграмм для обнаружения изменений плотности сети в сценариях с отсутствием, низкой, средней и высокой автокорреляциями

Диаграмма	Уровень	m_*=50	m_*=55	m_*=62	m_*=70	m_*=45	m_*=38	m_*=30
Шухарт	-	200.60	38.19	6.10	1.84	97.30	9.31	1.72
	Низкий	200.38	40.94	6.97	2.04	94.81	10.89	2.06
	Средний	200.45	47.01	8.23	2.36	105.94	13.19	2.24
	Высокий	200.59	116.39	21.66	1.01	144.87	27.90	1.00
EWMA λ=0.05	-	199.99	10.70	3.56	2.30	10.43	3.40	2.20
	Низкий	200.09	15.54	4.49	2.65	15.48	4.48	2.53
	Средний	200.10	22.89	6.27	3.26	23.20	6.15	3.02
	Высокий	199.95	113.70	30.26	8.62	116.20	27.67	7.73
EWMA λ=0.1	none	199.96	11.89	3.88	2.41	11.86	3.71	2.32
	Низкий	200.19	17.37	4.83	2.78	17.96	4.86	2.65
	Средний	200.01	25.22	6.78	3.46	27.45	6.70	3.28
	Высокий	200.12	114.12	29.29	7.18	120.60	27.40	6.45
EWMA λ=0.2	-	200.17	14.12	4.10	2.49	14.45	3.94	2.39
	Низкий	200.20	20.48	5.19	2.90	22.25	5.22	2.76
	Средний	200.28	28.71	7.22	3.52	34.80	7.31	3.35
	Высокий	199.76	111.70	26.59	5.31	125.83	25.96	4.74
CUSUM	-	203.48	12.14	4.05	2.33	4.69	2.68	1.96
	Низкий	202.98	12.50	4.16	2.44	11.77	3.94	2.28
	Средний	199.76	22.77	7.42	4.17	21.95	6.97	3.76
	Высокий	199.48	65.81	16.97	7.21	69.68	16.15	7.32
Остаточный	-	–	–	–	–	–	–	–
	Низкий	202.38	54.66	10.33	2.55	142.68	20.75	2.74
	Средний	200.76	72.83	14.21	3.15	189.73	51.81	4.82
	Высокий	203.97	85.71	1.00	1.00	162.09	1.00	1.00

В целом, контрольные диаграммы CUSUM лучше всего обнаруживают малые и средние сдвиги в сценариях низкой и средней автокорреляции. По сравнению с контрольными диаграммами Шухарта и EWMA негативное влияние высокой автокорреляции на точность контрольных диаграмм CUSUM меньше. Когда сдвиг велик (т.е. m^*= 70, 30), эффект автокорреляции не так силен, поскольку сдвиги достаточно велики, и все контрольные диаграммы могут достаточно хорошо обнаруживать большие сдвиги плотности сети. Характеристики диаграмм

остаточного контроля неблагоприятны для обнаружения малых и средних сдвигов в сценариях с низкой и средней автокорреляцией. При увеличении автокорреляции до более высокого уровня диаграмма остаточного контроля исключительно хорошо обнаруживает как средние, так и большие сдвиги.

Кроме того, обнаружено, что существует эффект взаимодействия между уровнем автокорреляции и весовым параметром на характеристиках контрольных диаграмм EWMA. Как видно из табл. 2.3, в сценариях отсутствия, низкого и среднего уровней автокорреляции значения ARL_1 увеличиваются с увеличением весового параметра λ с 0,05, 0,1 до 0,2. Это означает, что выбор меньшего значения λ приводит к повышению точности. Напротив, в сценарии высокой автокорреляции значения ARL_1 уменьшаются с увеличением λ. В этом случае для мониторинга процесса предпочтителен больший весовой параметр.

Таким образом, автокорреляция оказывает негативное влияние на характеристики контрольных диаграмм Шухарта, EWMA и CUSUM, поскольку они обычно предполагают независимость выборок с течением времени. Контрольные диаграммы EWMA наиболее чувствительны к увеличению автокорреляции при обнаружении сдвигов процесса. Меньший весовой параметр способствует повышению производительности, когда автокорреляция невысока, а в противном случае предпочтительно большее значение. Контрольные диаграммы CUSUM, как правило, лучше обнаруживают малые и средние сдвиги в сценариях с низкой и средней автокорреляцией. При наличии относительно высоких автокорреляций предлагаем использовать остаточные контрольные диаграммы, поскольку они лучше всего выявляют средние и большие сдвиги по сравнению с другими контрольными диаграммами.

2.6. Приложение к сетям электронной почты

В этом разделе мы анализируем сети электронной связи Enron в качестве примера. Версия данных взята из [2.36]. Этот набор данных содержит записи электронной почты 184 уникальных адресов электронной почты от 150 пользователей. Преимущество существует, если между двумя адресами в неделю поступает менее одного электронного письма.

Возьмем первые 20 сетей в качестве данных фазы I, чтобы оценить количество ребер в 20 сетях. Рис. 2.9 - графики автокорреляции и частичной автокорреляции, на которых можно наблюдать сильную автокорреляцию. Показано, что частичные автокорреляции как запаздывания 1, так и запаздывания 2 являются значимыми на графике частичной автокорреляционной функции (pacf) на рис. 2.9. Значение частичной автокорреляции запаздывания 1 превышает 0,6, что является относительно высоким. Поэтому рассматриваем возможность использования диаграммы остаточного контроля для мониторинга сетей Enron. По критерию AIC предпочтительна модель авторегрессии первого порядка. Параметр ϕ модели AR(1) оценивается как 0,62, что является значимым при значении p меньше 0,01. Стандартное отклонение остатков оценивается как 23,05. Диагностические графики для модели AR(1) показаны на рис. 2.10. На графике стандартизированных остатков не наблюдается никаких особых закономерностей. Автокорреляции не являются значимыми на основе графика автокорреляционной функции (acf) для остатков. Кроме того, остатки независимы, поскольку p-значения для статистики Юнга-Бокса превышают 0,05 для 10 лагов, как показано на нижнем графике. Таким образом, модель AR(1) хорошо соответствует данным.

Чтобы рассчитать контрольные пределы для остатков, можно смоделировать временные ряды сети с помощью STERGM, чтобы получить значение ARL_0 около 500 при небольшой вероятности ложной

тревоги. Применяем STERGM к выборочным сетям, начиная с первых 20 недель, в качестве контрольной модели. Можно видеть, что большинство значений pacf из смоделированных выборок близки к эталонному значению, полученному из реальных наблюдений, что показывает удовлетворительное соответствие STERGM данным Enron.

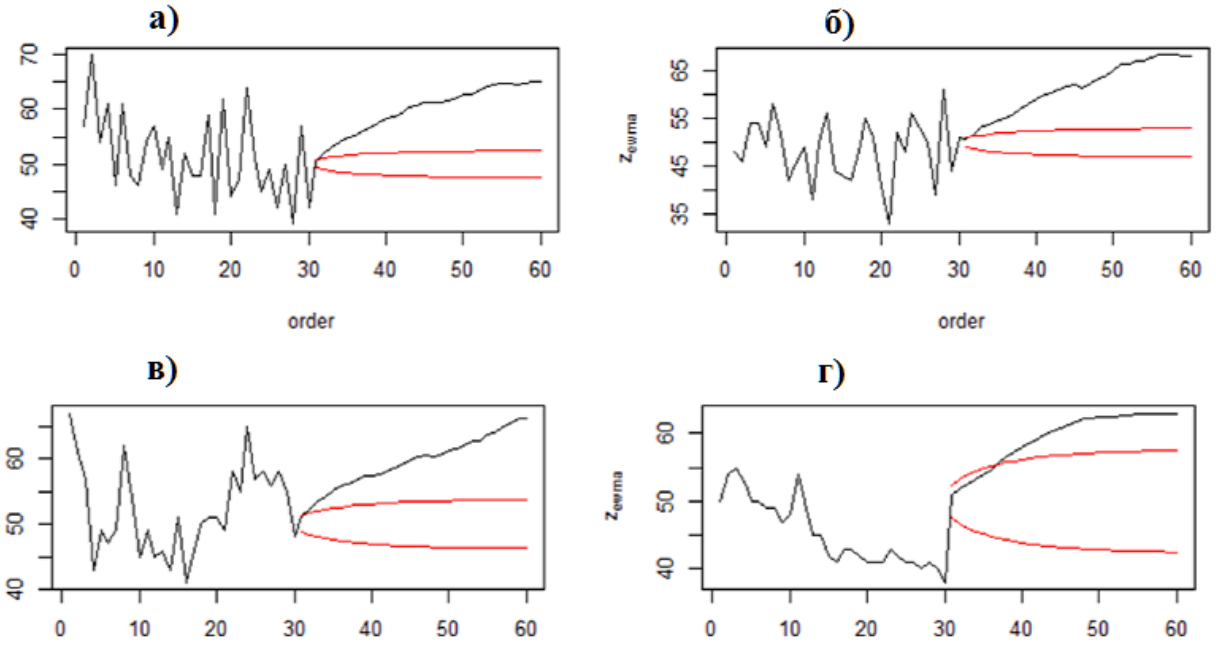


Рис. 2.6. Контрольная диаграмма EWMA для обнаружения положительного сдвига в различных сценариях автокорреляции: а – независимость; б – низкая автокорреляция; в – средняя автокорреляция; г – высокая автокорреляция

Используем STERGM в качестве сетевого генератора и получаем контрольный предел для диаграммы остаточного контроля с ARL_0, равным приблизительно 500, следуя схеме оценки производительности, представленной в разделе 2.4. Рассчитанный контрольный предел составляет 48,41 и -48,41. На рис. 2.12 показаны графики остаточного контроля. Датами выбросов являются недели вокруг 5/21, 5/28, 6/04, 06/11, 07/23, 09/10, 09/17, 09/24, 10/08, 10/22, 11/05, 11/12, 11/19, 12/03, и 12/17. Критические события корректно выявляются с помощью таблицы

остаточного контроля с ограничениями, полученными на основе подхода к оценке эффективности, основанного на STERGM.

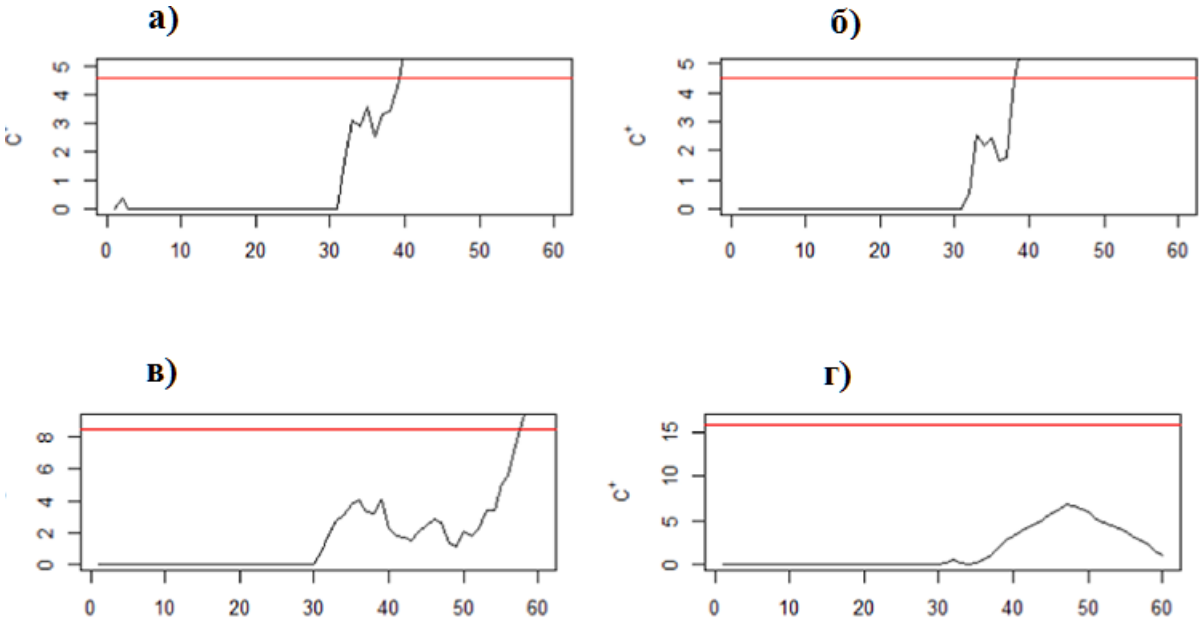


Рис. 2.7. Контрольная диаграмма CUSUM для обнаружения положительного сдвига в различных сценариях автокорреляции

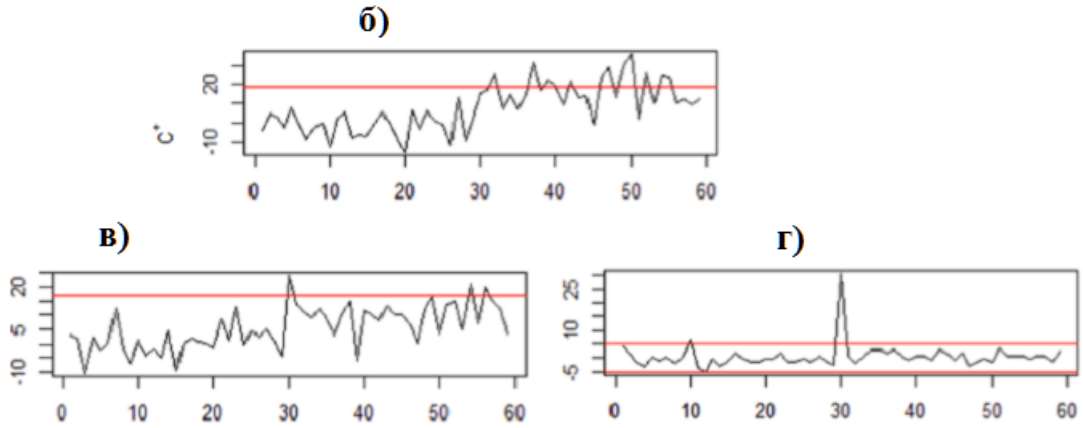
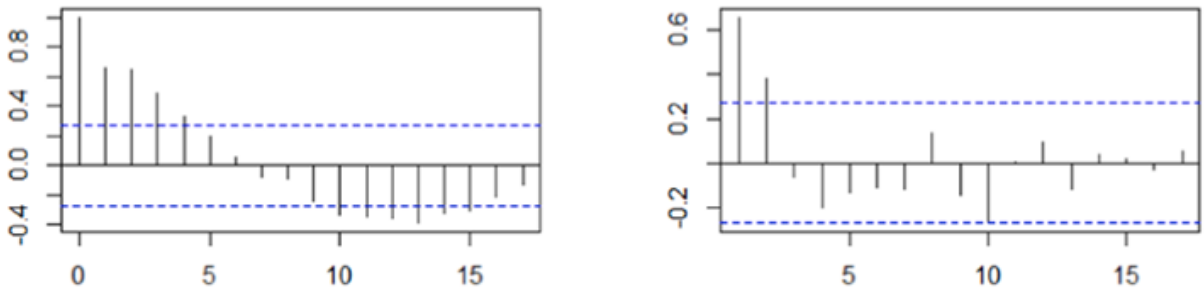


Рис. 2.8. Диаграмма остаточного контроля для обнаружения положительного сдвига в различных сценариях автокорреляции



а) б)

Рис. 2.9. Графики а - ACF и б - PACF для плотности данных Enron

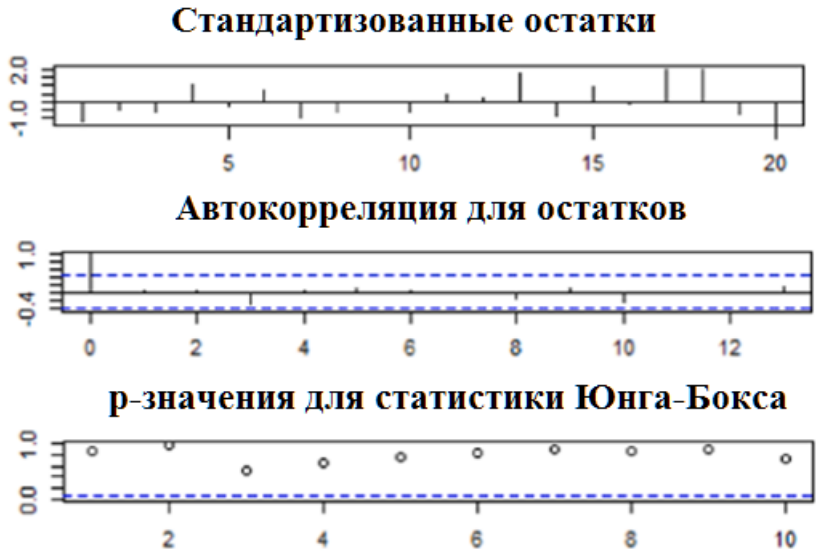


Рис. 2.10. Диагностические графики модели AR(1)

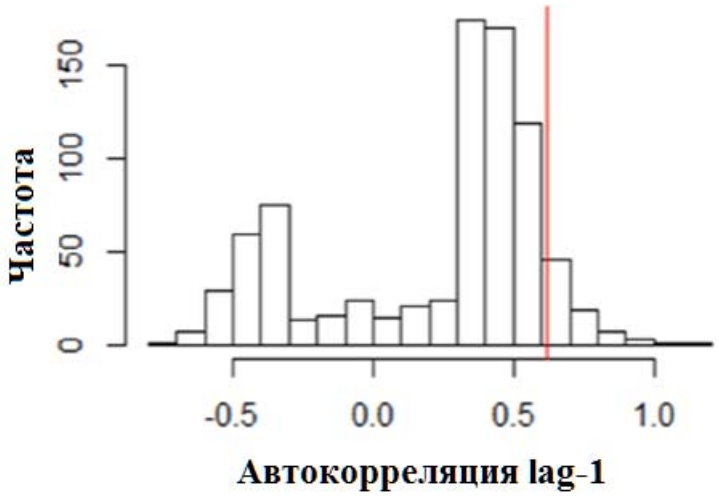


Рис. 2.11. Гистограмма значений автокорреляции lag-1 из смоделированных выборок

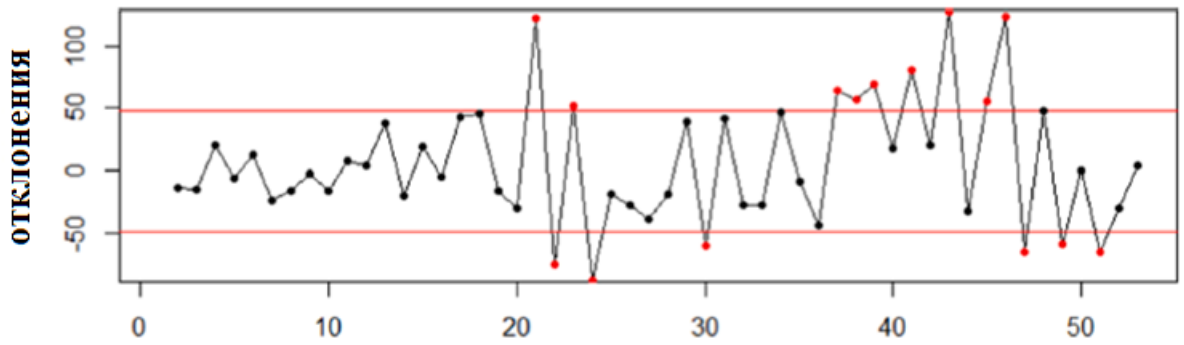


Рис. 2.12. Схема остаточного контроля для сетей электронной почты Enron

2.7. Выводы

Являясь важной категорией “больших данных”, сети повсеместно используются в производстве и сфере услуг. Современные измерительные устройства позволяют собирать данные с высокой частотой и неизбежно вызывают автокорреляции в динамических сетях. Оценка эффективности контрольных карт способствует оценке и выбору эффективного метода мониторинга для автокоррелированных сетей, что, таким образом, облегчает раннее выявление потенциальных сбоев, благоприятных для инженерного менеджмента в эпоху индустрии 3.5. Предложен подход к оценке производительности сетевых управляющих диаграмм, основанный на моделях разделяемых временных случайных графов. Интерпретируя сетевую автокорреляцию как постоянство локальных структур, представлен метод управления уровнем автокорреляции путем настройки параметра распада. Регулируя параметр образования, показан способ поддержания количества локальных структур на постоянном уровне.

Предложенная система моделирования на основе STERGM была применена для изучения влияния автокорреляции на характеристики диаграмм Шухарта, EWMA, CUSUM и остаточного контроля для подсчета ребер. Результаты численных экспериментов показали, что автокорреляции оказывают неблагоприятное влияние на показатели

контрольных диаграмм Шухарта, EWMA и CUSUM. Для контрольных диаграмм EWMA следует выбирать меньшее значение при наличии низкой и средней автокорреляции, в то время как в противном случае предпочтительнее большее значение. Как правило, контрольные диаграммы CUSUM лучше обнаруживают малые и средние сдвиги в сценариях с низкой и средней автокорреляцией. При наличии относительно высоких автокорреляций мы предлагаем использовать остаточные контрольные диаграммы, поскольку они лучше всего обнаруживают средние и большие сдвиги.

Полезность предложенной структуры, основанной на STERGM, и эффективность руководства, полученного в результате исследования эффекта автокорреляции, были проиллюстрированы данными сети электронной связи Enron. В этом реальном случае было показано, что автокорреляция статистики количества ребер находится на уровне от среднего до высокого. Поэтому была принята таблица остаточного контроля. Контрольные пределы были получены с помощью предложенной структуры, основанной на STERGM, для достижения заранее заданного значения ARL_0. Тот факт, что критические события обнаруживаются корректно, показывает практичность нашего метода.

Есть и некоторые ограничения. Во-первых, основное внимание в текущем исследовании уделяется оценке эффективности сетевых контрольных карт при выявлении аномальных изменений плотности сети. Однако на практике формирование сети обычно зависит от множества доминирующих локальных структур, таких как звезды и треугольники, как упомянуто в разделе 2.2.1. Следовательно, совместное влияние автокорреляций нескольких свойств сети заслуживает дальнейшего изучения. Во-вторых, обсуждаась только автокорреляция сетевых структур первого порядка. Сети с автокорреляциями более высокого порядка могут быть исследованы в качестве направления дальнейших исследований. В-

третьих, хотя диаграмма остаточного контроля предлагается для сценариев с высокой автокорреляцией, чтобы уменьшить неблагоприятные последствия пренебрежения автокорреляцией, ее эффективность не так высока, как у других методов в сценариях с низкой и средней автокорреляцией. Следует провести дополнительные исследования методов мониторинга сетей с низкой и средней автокорреляцией.

3. АЛГОРИТМИЗАЦИЯ МОНИТОРИНГА ПРОИЗВОДИТЕЛЬНОСТИ КОЛЛЕКТИВОВ ВСТРАИВАЕМЫХ ОБЪЕКТОВ БОЛЬШИХ ПРОГРАММНЫХ СИСТЕМ НА ОСНОВЕ ВРЕМЕННЫХ РЯДОВ ДЛЯ АНАЛИЗА ЭНЕРГОПОТРЕБЛЕНИЯ

3.1. Моделирование исследования временных рядов

3.1.1. Спецификация модели анализа временных рядов

Предлагаемый метод анализа состоит из 3 этапов: сбор данных, создание объектно-ориентированной модели, поведенческая интерпретация этой модели. Исходные данные (выборки) преобразуются и агрегируются в иерархические объекты с учетом некоторых заданных свойств/показателей. В результате получается высокоуровневая графовая модель, которая облегчает извлечение нетривиальных сведений о поведении системы, в частности о состояниях активности устройств и их последовательности, связанных с потреблением энергии. Это полезно для оценки системы, а также для ее коррекции или уточнения. Временной ряд - это последовательность из N выборок данных TS={$sample_1$, ..., $sample_N$}, которые записываются во временном порядке с фиксированными интервалами. Каждая выборка характеризуется своим значением и временной меткой. Измерительные устройства могут выдавать эти выборки в виде непрерывного потока данных или в виде пакетов с явной или неявной временной меткой (например, в зависимости от периода выборки и начальной временной метки для пакета). TS разбивается на M=N/n меньших частей, называемых сегментами (обозначаемыми как seg), содержащих заданное количество (n) последовательных выборок segi={$sample_{i\cdot n+1}$, ... $sample_{i\cdot n+n}$}. Каждый сегмент характеризуется набором параметров, рассчитанных по всем его выборкам, например: минимальный (min), максимальный (Max), средний (Av), среднеквадратичный (RMS), стандартное отклонение (SD), квартили и т.д. Последующие сегменты

помечаются последующими значениями индекса. Некоторые из последующих сегментов могут быть очень похожи (с точки зрения их показателей). Поэтому целесообразно объединить их в группы. Группа (gk) - это последовательность последующих сегментов, удовлетворяющая следующему условию:

$$g_k = \{seg_i : \forall seg_i \in g_k \exists seg_{i+1} : | metric(seg_i) - metric(seg_{i+1}) | < \varepsilon_g \qquad (3.1)$$

где metric обозначает выбранные параметры сегмента (например, Av или SD), $i \in [1, \| g_k \|)$, а ε_g - предопределенная максимально допустимая разница показателей соседних сегментов в группе. Также могут быть определены другие показатели, определяющие более сложные условия, связанные с набором объектов. Сегменты, не удовлетворяющие условию (3.1), образуют отдельные группы сегментов. Описанный процесс группировки обеспечивает последовательность групп. Он основан на агрегировании последующих и сходных – с точки зрения заданного показателя – сегментов выборок TS. Эта последовательность строго связана с изменениями состояния системы. На данном этапе анализа TS может быть представлена в виде хронологически упорядоченных последовательностей групп. Для повышения точности модели границы между группами могут быть уточнены путем разделения соседних сегментов. Для каждой группы g_k может быть присвоено системное состояние s_x. Правильная идентификация групп, представляющих одно и то же состояние системы в пределах их последовательности, имеет решающее значение для анализа. Хотя все похожие группы в TS связаны с одним и тем же состоянием системы, данное состояние s_x (с меткой x) может быть определено как совокупность всех групп, удовлетворяющих следующему условию:

$$s_x = \{g_i : \forall g_i \in s_x \; \exists g_j \in s_x, i \neq j \wedge | metric(g_i) - metric(g_j) | < \varepsilon_s \qquad (3.2)$$

где ε_s - заданная максимально допустимая разница показателей, относящихся к рассматриваемым группам. Группы, не удовлетворяющие

условию (2), относятся к состояниям одной группы. Также возможны более сложные условия, включающие несколько показателей и различные свойства. Для данного временного ряда вводится ориентированный граф состояний SG={s_{g0}, S, E}, где s_{g0} - это состояние начальной группы TS (относящееся к исходному состоянию системы), S и E - множества вершин (т.е. все состояния) и ребер между вершинами (узлы), соответственно. Набор ребер определяется следующим образом:

$$E = \{(u,v) \in S, u \neq v, \exists g_i \in u \wedge g_j \in v : \exists seg_k \in g_i \wedge seg_l \in g_j, k+1=l\} \quad (3.3)$$

где i, j $\in$ [0, |G|), G - множество всех обнаруженных групп, а k, l$\in$ [0, M). Это условие гарантирует, что ребро соединяет соседние (по времени) группы в TS, которые принадлежат, соответственно, двум разным состояниям u и v. Сегмент seg_k является последним в группе, принадлежащей состоянию u, в то время как сегмент seg_l является первым в группе, принадлежащей состоянию v. Для каждого состояния определены наборы выходных и входных ребер s_x:

$$E_{out(x)}=\{(s_x,v): (s_x,v)\in E\} \quad (3.4)$$

$$E_{in(x)}=\{(u, s_x): (u, s_x)\in E\} \quad (3.5)$$

В зависимости от целей анализа различают конкретные состояния: последовательные состояния, удовлетворяющие условию $|E_{out(x)}|=|E_{in(x)}|=1$, и базовое состояние ($s_B$) с предопределенными свойствами, например, с множеством соответствующих ребер, т.е.:

$$\exists s_b, \forall s_i \in S, \left|E_{out(B)}\right| + \left|E_{in(B)}\right| > \left|E_{out(i)}\right| + \left|E_{in(i)}\right| \quad (3.6)$$

С точки зрения встроенных систем, базовое состояние соответствует состоянию, из которого выполняются все системные действия и в которое система переходит после этих действий (т.е. основному циклу). После определения таких состояний анализ может быть сосредоточен на действиях, выполняемых системой в ответ на различные события

(синхронные и асинхронные – таймеры, прерывания и т.д.). В некоторых приложениях может быть целесообразно определить несколько базовых состояний, однако в рамках данногоисследования и в представленном примере рассматривается только одно базовое состояние. Цикл (C) представляет собой последовательность групп, для которых последующие элементы соединены ребром в системном графе (SG), и первое состояние в этой последовательности соединено ребром, принадлежащим $E_{out(B)}$, а последнее - ребром, принадлежащим $E_{in(B)}$, где s_B - это базовое состояние. Цикл $C=\{g_i, ..., g_p\}$, где $\{g_i, ..., g_p\} \in G$ и i...p - последовательные номера групп (идентификаторы), удовлетворяет следующим условиям:

1) $(g_i \in s_i) \wedge (s_B, s_i) \in E_{out(B)}; (g_p \in s_p) \wedge (s_p, s_B) \in E_{in(B)}$

2) пусть s^{g_x} обозначает состояние, к которому принадлежит группа g_x: $\forall x \in [i,p) : g_x, g_{x+1} \in C, (s^{g_x}, s^{g_{x+1}}) \in E, vs^{g_x} = s^{g_{x+1}}$

3) $\forall x \in [i,p) : s^{g_x} \neq s_B$

Цикл представляет собой последовательность последующих групп, так что ни одна из этих групп не соответствует базовому состоянию (S_B), первой группе в последовательности предшествует группа, помеченная состоянием S_B, а за последней группой следует такая группа. Представленная объектно-ориентированная модель TS предназначена для анализа временных рядов, описывающих спорадические действия устройства, рассеянные по некоторым колебаниям фонового сигнала. Она раскрывает иерархическую структуру с такими компонентами, как сегмент, группа, цикл, класс циклов, элементы графика. Класс циклов - это набор реализаций циклов во времени в пределах TS, которые удовлетворяют заданному условию подобия (например, схожему поведению по энергопотреблению). Идентифицированные объекты могут быть сопоставлены со знаниями о работе устройства, доступными эксперту, или с зарегистрированными журналами событий, что позволяет

провести точную диагностику. Следовательно, соответствующие алгоритмы анализа могут быть сформулированы также с учетом этого представления, что упрощает их спецификацию, а также облегчает реализацию кода. Для получения конкретных свойств рассматриваемых объектов (контейнеров данных) в спецификациях алгоритмов используются следующие функции: avg(x), min(x), max(x), deviation(x), RMS(x), которые обеспечивают среднее, минимальное, максимальное, стандартное отклонение и среднеквадратичное значение всех выборок в пределах рассматриваемого объекта x (например, сегмента, группы) соответственно. В следующем разделе описывается предлагаемая алгоритмическая обработка TS.

3.1.2. Алгоритмы обработки данных – общее представление

Собранные данные, составляющие временные ряды, и предопределенные характеристики компонентов (объектов) используются для создания соответствующей модели. Эта модель включает спецификацию некоторых компонентов (объектов) и связей между ними для разработанных алгоритмов обработки данных. Эти алгоритмы являются гибкими в некоторых диапазонах за счет задания набора параметров. Они включают в себя такие параметры, как начальный размер каждого сегмента, допустимые различия функций подобия (метрик). Обозначения разработанных алгоритмов представлены в разделе 3.2.1, за которыми следуют спецификации алгоритмов (раздел 3.2.2). Некоторые комментарии по особенностям алгоритмов приведены в разделе 3.2.3.

3.2. Алгоритмы обработки данных - детализация

3.2.1. Псевдокодовая нотация

Представленные псевдокоды используют обозначения, частично унаследованные от языка программирования Python. Это объектно-

ориентированная нотация, например, parameters.b обозначает получение атрибута b из экземпляра объекта parameters. Название атрибута определяет его значение, например, параметры.segment_size, параметры.ave_eps, параметры.dev_eps относятся к получению размера сегмента, среднего значения и предельных значений отклонения для показателей подобия соответственно. Для данных типов объектов (например, сегментов, групп, состояний и циклов) определены соответствующие методы, которые могут быть вызваны для их экземпляров объектов, например, GroupA.add(segmentB) обозначает вызов метода, который добавляет segmentB к GroupA. Вызов функции are_similar(segmentB, GroupA) вычисляет показатели для всей группы сегментов внутри группы, связанной с переменными GroupA и segmentB, и возвращает логическое значение, если разница между ними находится в пределах предопределенного диапазона (см. определение группы). Большинство представленных объектов содержат объекты других типов, например, граф содержит набор состояний/узлов и ребер, группа - это последовательность сегментов, сегмент - это последовательность выборок. Итак, используются следующие контейнеры объектов: объект list представляет собой упорядоченную последовательность объектов, в то время как набор содержит неупорядоченные, но уникальные экземпляры объектов (добавление второго объекта с такими же свойствами будет недействительным – в наборе будет сохранен только первый экземпляр). Экземпляры объектов создаются с использованием выражения new object_type, т.е. GroupA=новая группа создает группу и присваивает ее переменной GroupA. Типы объектов предоставляют методам и свойствам понятные названия, соответствующие их значению. Свойство типа объекта может ссылаться на его внутреннюю коллекцию (например, список выборок в группе, наборы ребер и узлов в графе) или на прямое значение (например, list. count - это количество элементов, хранящихся в контейнере

списка). Ссылка на свойства и методы объектов обозначается с помощью оператора точки, как уже объяснялось. Однако, поскольку свойство, на которое ссылается ссылка, может быть другим объектом, последующие точки используются для более глубокой ссылки. Например: узел Out_edges.count ссылается на свойство count коллекции Out_edges экземпляра узла – другими словами, оно указывает количество исходящих ребер узла. Экземпляры объектов, используемых в алгоритмах, также имеют понятные названия, строго соответствующие определениям, приведенным в предыдущем разделе. Итерационные циклы по контейнерам объектов (наборам, последовательностям и т.д.) могут быть построены по схеме foreach:

foreach *variable_name* ***in*** *container* ***do***
    expressions
end foreach

Это относится к итеративному циклу для каждого объекта в пределах указанного контейнера объектов коллекции. На каждой итерации в указанных выражениях может использоваться один объект из этой коллекции. Выражения относятся к данному блоку инструкций на основе создания экземпляра инструкции в коде. Однако для лучшей читаемости алгоритма, вопреки концепции Python, имя переменной может использоваться в выражениях цикла для обозначения не только текущего итерируемого объекта, но и его предшественника или преемника в итерируемом контейнере (используя ссылки variable_name.prev и variable_name.next). Конструкция if...then...else...end if имеет классическую структуру и значение. Условия задаются с помощью базовых операторов связи: =, <, >, ≤, ≥, ≥, и арифметических операторов для математических вычислений: * умножение, / деление + сложение, - вычитание.

3.2.2. Спецификация алгоритмов

Процесс анализа TS включает в себя создание соответствующей модели и отслеживание ее характеристик в соответствии с внедренными алгоритмами. Представлены и систематизированы псевдокоды разработанных алгоритмов, начиная с тех, которые выводят базовые объекты (сегменты, группы последовательностей) и заканчивая более сложными, связанными с генерацией графовой модели (состояний и ребер) и отслеживанием последовательностей состояний или классов циклов,. Набор из 12 разработанных алгоритмов представлен в виде иерархической структуры, состоящей из пяти групп (обозначенных индексами алгоритмов):

{1,2} – создание групп сегментов

{3} – уточнение границ групп сегментов

{4,5,6} – создание модели графа состояний (состояний и ребер)

{7,8} – граф состояний сокращение с помощью слияния последовательностей состояний

{9,10,11,12} – анализ цикла состояний, включая идентификацию классов циклов.

Первый алгоритм в группе использует функции, определенные в последующих элементах группы. Эти алгоритмы используют различные параметры, определяющие свойства рассматриваемых объектов и их взаимосвязи. Результат последней группы алгоритмов, соответствующий наиболее сложной обработке данных, проиллюстрирован на примере последовательности состояний, соответствующей некоторой модели TS.

Создание сегментов - довольно тривиальная задача (простая группировка выборок в наборы фиксированного размера), и она пропускается. Другие объекты более сложны. Алгоритм 1 и алгоритм 2 задают псевдокоды для создания групп сегментов.

Алгоритм 1: Создание групп сегментов

Input: *Segments* – список сегментов
Output: списокгрупп сегментов

```
1:    function create_groups(Segments)
2:        Groups=new list
3:        group=new group
4:        foreach segment in Segments do
5:            if are_similar(segment, group) then
6:                group.add(segment)
7:            else
8:                Groups.add(group)
9:                group=new group
10:               group.add(segment)
11:           end if
12:       end foreach
13:       return Groups
14:   end function
```

Алгоритм 2: Сходство сегмента и группы

Input: segment, group
Output: *True*, если сегмент аналогичен группе, *False* иначе

```
1: function are_similar(segment, group)
2: if (group.segment_count == 0) then
3:     return True
4: end if
5: if ((abs(avg(group)-avg(segment))<parameters.avg_eps)
6:     and (abs(deviation(group)-deviation(segment))<parameters.dev_eps)
7:     and (abs(min(group)-min(segment))<parameters.min_eps)
8:     and (abs(max(group)-max(segment))<parameters.max_eps)
9:     and (abs(RMS(group)-RMS(segment))<parameters.RMS_eps))
10: then
11:    return True
12: else
13:    return False
14: end if
15: end function
```

Функция create_groups() (показанная в алгоритме 1) основывается на ранее созданном списке последующих сегментов (Segments). В цикле foreach (строки 4-12) последующие сегменты (через переменную, называемую segment) повторяются и добавляются в группу (group variable)

с помощью вызова метода group.add (segment) (строка 6) в случае, если рассматриваемый сегмент удовлетворяет условию подобия для группы. Это проверяется с помощью вызова функции are_similar(сегмент, группа) в условном выражении в строке 5. Эта группа расширяется при выполнении условия подобия.

При обнаружении первого сегмента, который не может быть добавлен в текущую группу, созданная группа добавляется в список групп (строка 8), и последующий сегмент может начать создание новой группы (строки 9 и 10). Функция are_similar() (приведенная в алгоритме 2) проверяет указанные условия подобия. Здесь, в качестве примера, параметры рассматриваемого сегмента (среднее значение, отклонение, минимальное, максимальное и среднеквадратичное значение) сравниваются с соответствующими параметрами созданной группы. Выполнение всех условий (проверенных в строке 2 и строках 5-9) возвращает значение True, позволяющее добавить сегмент к текущему объекту группы. Это приводит к созданию упорядоченного набора групп (в соответствии с хронологической последовательностью выборок, затем ее сегментов и, наконец, групп аналогичных соседних сегментов). Стоит отметить, что значительные различия между соседними группами могут привести к созданию промежуточных “узких” групп (с небольшим количеством сегментов). Такие группы соответствуют изменению состояния системы, поскольку их соседние сегменты имеют значительно отличающиеся показатели. Предлагаемый анализ направлен на идентификацию состояний; таким образом, эти группы могут быть разделены на две части и объединены с соответствующими соседними группами. Это выполняется с помощью процедуры уточнения группы increase_accuracy(), указанной в алгоритме 3.

Алгоритм 3: Повышение точности групп из одного сегмента
Input: Groups – список групп сегментов
Output: Groups – список групп с уточненными односегментными группами

```
1: function increase_accuracy(Groups)
2:     foreach group in Groups do
3:         if ((group.segment_count == 1)
4:             and (group.prev.segment_count > 1)
5:             and (group.next.segment_count > 1)) then
6:         group.prev.append_from_beggining(group.first_segment)
7:         group.next.appent_from_end(group.first_segment)
8:         end if
9:     end foreach
10: end function
```

Из-за фиксированного размера сегментов могут возникать промежуточные короткие группы, связанные со значительным изменением значений параметров между двумя соседними более длинными группами (состоящими как минимум из 2 сегментов). Таким образом, границы групп могут быть уточнены. Найдя промежуточную группу (с одиноким сегментом), ее сегмент разбивается на два подсегмента: левый сравнивается с показателями предыдущей группы, а правый - со следующей группой. Найдена самая длинная последовательность выборок (подсегмент) в промежуточном сегменте, удовлетворяющая показателям сходства соседних групп. Если этот подсегмент согласован (в соответствии с указанным показателем сходства) с соседней группой, то он добавляется к этой группе и удаляется из рассматриваемой промежуточной группы (при вызовах методов append_from_beggining и append_from_end, в строках 6 и 7 алгоритма 3, соответственно). Чтобы ускорить работу алгоритма, при идентификации подсегмента можно использовать рекурсивную концепцию "разделяй и властвуй".

После уточнения групп сегментов создается модель состояния TS в рамках трехэтапного процесса. Первым шагом является определение

состояний (узлов графа) – алгоритм 4. Соответствующая процедура detect_states () выполняет итерацию по списку групп (Groups). На каждой итерации (цикл foreach в строке 3) из списка всех групп берется группа (в вызове функции add_to_similar_state(состояния, группа) в строке 4 – функция представлена в алгоритме 5), и состояние (среди всех состояний, идентифицированных на данный момент), соответствующее этой группе, определяется как искал. Эта группа имеет те же показатели, что и состояние, проверяемое вызовом функции are_similar(state, group) в строке 3 алгоритма 5. При поиске состояния для данной группы группа добавляется в состояние (строка 4) и приписывается как член состояния (строка 5). Если какое-либо из существующих состояний не удовлетворяет требованию подобия (ни одно из них не соответствует результату в строке 9 алгоритма 5, проверено в строке 5 алгоритма 4), создается новое состояние, основанное на проанализированной группе (в вызове create_new_state(group) в строке 6 алгоритма 4), и добавляется к состояниям контейнер в строке 7. Объект состояния содержит коллекцию всех групп, относящихся к данному состоянию, а также метку состояния (которая также относится ко всем группам, принадлежащим этому состоянию).

Набор всех состояний, которые рассматриваются как узлы графа состояний (функция detect_states() – алгоритм 4), выводится путем сканирования всех групп.

Переходы между этими состояниями создаются (в функции detect_edges (), показанной в алгоритме 6) путем еще одной итерации по всем обнаруженным группам (уже назначенным определенным состояниям на предыдущем шаге). Таким образом, сканируются последующие группы (строки 3-9), они сопоставляются с ранее идентифицированными состояниями (получаем метку группы с помощью group.StateLabel). Следовательно, создается новое ребро (с помощью выражения new

Edge(group.StateLabel, group.next.StateLabel) в строке 6) и добавляется к набору ребер (экземпляру Edges). Если ребро между заданной парой меток состояний отсутствует в наборе ребер, оно добавляется – контейнер set гарантирует уникальность его элементов. Результатом работы функций, представленных в алгоритмах 4, 5 и 6, является список состояний и набор ребер. На основе этих двух наборов данных создание графа состояния системы является простой задачей. На графе создаются два перехода: один ведет к текущему анализируемому состоянию, связанному с группой, а второй - к следующему состоянию, связанному с последующей группой.

Алгоритм 4: Определение состояний
Input: *Groups* – список групп сегментов
Output: список выявленных состояний

```
1: function detect_states(Groups)
2: States=new list
3: foreach group in Groups do
4:     state=add_to_similar_state(States, group)
5:     if (state is None) then
6:         state=create_new_state(group)
7:         States.add(state)
8:     end if
9: end foreach
10: return States
11: end function
```

Алгоритм 5: Добавление группы в аналогичное состояние
Input: *States* – список состояний, *group* – группа сегментов
Output: **состояние**, на которое похожа группа: группе присваивается метка состояния, и группа добавляется в состояние; **Нет**, если похожее состояние не найдено

```
1: function add_to_similar_state(States, group)
2: foreach state in States do
3:     if (are_similar(state, group)) then
4:         state.add(group)
5:         group.StateLabel=state.Label
6:         return state
7:     end if
8: end foreach
```

```
9: return None
10: end function
```

Алгоритм 6: Определение ребер графа состояния системы

Input: *Groups* – список последующих групп с присвоенными метками состояний

Output: *Edges* – набор границ между состояниями

```
1: function detect_edges(Groups)
2: Edges=new set
3: foreach group in Groups do
4:     If (group.next is not None) then
5:         if (group.StateLabel != group.next.StateLabel) then
6:             Edges.add(new Edge(group.StateLabel, group.next.StateLabel))
7:         end if
8:     end if
9: end foreach
10: return Edges
11: end function
```

На практике производный граф состояний может быть довольно сложным, в частности, содержать длинные последовательности состояний с одним входным и выходным ребрами. Например, всю последовательность загрузки встроенной системы можно рассматривать как последовательность множества детализированных состояний. Но во многих случаях нецелесообразно сосредотачиваться на них – упрощение графика состояний путем объединения последовательности загрузки в одно состояние может облегчить дальнейший анализ. Следовательно, может быть создан сокращенный граф, который объединяет такие последовательности состояний в уникальное обобщенное состояние (в зависимости от целей анализа). Стоит отметить, что если одно и то же состояние появляется также за пределами простой последовательности (в другой группе – в разные моменты времени TS – например, во время последовательности загрузки и во время нормальной работы системы), такое состояние не будет объединено в суперсостояние, поскольку оно не является уникальным, поскольку оно также используется вне

последовательности, подлежащей объединению. Этот процесс реализуется с помощью процедуры merge_simple_sequences() (алгоритм 7). Эта процедура идентифицирует все простые последовательности (т.е. последовательности состояний с одним входящим и выходящим ребрами – первое состояние в графе и последнее может иметь только одно ребро) с помощью рекурсивной функции identify_simple_sequences() (показано в алгоритме 8). Он посещает все узлы (т.е. состояния) графа (строки 14 ÷ 19). Если данный узел имеет одно или меньшее количество входных и выходных ребер, он добавляется к текущей собранной последовательности узлов (строка 4). Узел.In_edges и node.Коллекции Out_edges (строка 3 алгоритма 8) используются для обработки граничных состояний. Если текущий узел не удовлетворяет указанному требованию, объединенная текущая последовательность добавляется к набору последовательностей и создается новый объект current_sequence (строки 11, 12). В результате получается набор списков, содержащих последовательные состояния. Каждый элемент этого набора используется в функции merge_states_and_nodes() для замены соответствующей последовательности состояний на соответствующее состояние (указанное с меткой первого состояния в рассматриваемом списке). После этого шага обновляется вся структура графа: группы из объединенных последовательностей повторно помечаются для присвоения первого состояния последовательности (и эти группы также удаляются из объектов состояния). В действительности, некоторые состояния на графике могут содержать нулевые группы, поэтому такие состояния должны быть удалены из графика. Ребра также могут нуждаться в исправлении. Все эти действия по очистке выполняются при вызове функции merge_states_and_nodes(последовательности, график) (строка 6 алгоритма 7).

Алгоритм 7: Объединение простых последовательностей
Input: *graph* – граф
Output: *graph* – граф с объединенными простыми последовательностями (измененные узлы и состояния)

```
1: function merge_simple_sequences(graph)
2:     graph.Nodes.clear_visited_flags()
3:     sequences=new set
4:     current_sequence=new list
5:     identify_simple_sequences(graph.Nodes.first, sequences,
current_sequence)
6:     merge_states_and_nodes(sequences, graph)
7: end function
```

Алгоритм 8: Идентификация простых последовательностей
Input: *node* – объект узла графа, *sequences* – набор последовательностей, *current_sequence* – обрабатываемая в данный момент последовательность
Output: *sequences* – обновленный набор последовательностей

```
1: function identify_simple_sequences(node, sequences, current_sequence)
2: node.visited=True
3: if (node.In_edges.count <= 1 and node.Out_edges.count <= 1) then
4:     current_sequence.add(node)
5:     if (node.Out_edges.count == 0) then
6:         sequences.add(current_sequence)
7:         current_sequence=new list
8:     end if
9:     else
10:        if (current_sequence.Length > 1) then
11:            sequences.add(current_sequence)
12:            current_sequence=new list
13:        end if
14:    end if
15:    foreach neighbour in node.Neighbours do
16:        if (neighbour.visited is False) then
17:    identify_simple_sequences(neighbour, sequences, current_sequence)
18:        end if
19:    end foreach
20: end function
```

Графовая обработка может быть адаптирована к анализируемым задачам, например, для поиска похожих последовательностей состояний. Похожие последовательности состояний могут появляться в разных частях

анализируемой TS и относиться к похожим внутренним/внешним обстоятельствам/событиям. Это может быть полезно для обнаружения аномалий, определения типичных профилей работы, которые можно сопоставить с журналами приложений или системными журналами. Пример анализа последовательности описан в алгоритме 9 (в нем используются алгоритмы 10-12). Функция detect_base_state() (вызываемая в строке 2 и представленная в алгоритме 10) определяет базовое состояние, которое в дальнейшем используется для извлечения циклов, составляющих это состояние – find_cycles, вызываемая в строке 3 и представленная в алгоритме 11. Базовое состояние - это состояние, которое содержит наибольшее количество входных и выходных данных. Эта характеристика позволяет рассматривать базовое состояние как состояние по умолчанию в рабочем профиле устройства и использовать его в качестве ориентира для обнаружения аномалий. Следовательно, поиск цикла (алгоритм 11) производится путем перебора всех групп (строки 4-17) и нахождения последовательностей групп, которым предшествует группа, присвоенная базовому состоянию, и за которыми следует группа, присвоенная базовому состоянию. На следующем шаге функция find_similar_cycles() (вызываемая в строке 4 алгоритма 9 и представленная в алгоритме 12) вычисляет метрики для каждой последовательности, найденной как цикл, и сравнивает их, чтобы идентифицировать похожие циклы, которые в дальнейшем обозначаются как класс циклов (CC). Каждый класс циклов содержит циклы со схожими свойствами, оцениваемыми вызовом функции are_similar() в строке 10 алгоритма 12, которые, скорее всего, описывают схожие ситуации в анализируемой TS (однако, происходящие в разные периоды времени). Стоит отметить, что сходство циклов необязательно означает одинаковую последовательность состояний, поскольку во встроенной системе некоторые процедуры обработки событий могут меняться в зависимости от условий внешней среды, что приводит к

незначительным различиям состояний в циклах, обрабатывающих один и тот же сценарий.

Алгоритм 9: Поиск всех похожих циклов.

Input: *graph* – граф

Output: *graph* – граф с идентифицированным базовым состоянием и циклами

```
1: function find_all_similar_cycles(graph)
2: detect_base_state(graph)
3: find_cycles(graph)
4: find_similar_cycles(graph)
5: end function
```

Алгоритм 10: Определение базового состояния

Input: *graph* – граф

Output: *graph* – граф с идентифицированным базовым состоянием

```
1: function detect_base_state(graph)
2: max=0
3: foreach node in graph.Nodes do
4:     if (node.Edges.count > max) then
5:         base_state=node.state
6:         max=node.Edges.count
7:     end if
8: end foreach
9: graph.base_state=base_state
10: end function
```

Алгоритм 11: Поиск всех циклов в графе

Input: *graph* – граф

Output: *graph* – граф с идентифицированными циклами

```
1: function find_cycles(graph)
2: cycle=None
3: Cycles=new list
4: foreach g in graph.Groups do
5:     if (g == graph.base_state) then
6:         if (cycle == None) then
7:             cycle=new cycle
8:         else
9:             Cycles.add(cycle)
10:            cycle=None
11:        end if
12:        else
```

```
13:                 if (cycle != None) then
14:                     cycle.Add(g)
15:             end if
16:     end if
17: end foreach
18: graph.Cycles=Cycles
19: end function
```

Алгоритм 12: Поиск похожих циклов в графе

Input: *graph* – граф с идентифицированными циклами

Output: *graph* – граф с идентифицированными похожими циклами

```
1: function find_similar_cycles(graph)
2: Cycle_Classes=new list
3: Cycles.ClearVisitedFlag()
4: foreach c1 in graph.Cycles do
5:     if (c1.visited == False) then
6:         c1.visited=True
7:         current_CC=new list
8:         current_CC.add(c1)
9:         foreach c2 in graph.Cycles do
10:            if ((c2.visited == False) and are_similar(c1,c2)) then
11:                current_CC.add(c2)
12:                c2.visited=True
13:            end if
14:        end foreach
15:        Cycle_Classes.add(current_CC)
16:    end if
17: end foreach
18: graph.Cycle_Classes=Cycle_Classes
19: end function
```

Для обобщения алгоритмов 9-12 рассмотрена наглядная последовательность состояний. Это соответствует идентифицированным последующим группам сегментов $S_i^{g_k}$ (где S_i обозначает экземпляр сегмента i, g - обозначает группу, к которой он принадлежит, и k обозначает метку состояния, к которому он принадлежит) TS, за исключением:

S^{0}_{00} S^{1}_{115} S^{1}_{215} S^{1}_{315} S^{1}_{415} S^{1}_{515} S^{1}_{615} S^{1}_{715} S^{1}_{815} S^{1}_{915} S^{1}_{1015} $\mathbf{S^{2}_{1116}}$ $\mathbf{S^{3}_{1217}}$ $\mathbf{S^{4}_{1316}}$
$\mathbf{S^{5}_{1418}}$ S^{6}_{1515} S^{6}_{1615} S^{6}_{1715} S^{6}_{1815} S^{6}_{1915} S^{6}_{2015} S^{6}_{2115} S^{6}_{2215} S^{6}_{2315} S^{6}_{2415} S^{6}_{2515}
$\mathbf{S^{7}_{2619}}$ $\mathbf{S^{8}_{2717}}$ $\mathbf{S^{9}_{2816}}$ $\mathbf{S^{10}_{2918}}$ $\mathbf{S^{11}_{3020}}$ S^{12}_{3115} S^{12}_{3215} S^{12}_{3315} S^{12}_{3415} S^{12}_{3515} S^{12}_{3615}

Здесь в качестве базового состояния для циклов поиска в TS было выбрано состояние 15. Это состояние содержит максимальное количество входящих и исходящих ребер в соответствующем графе состояний. Выделенные групповые циклы выделены жирным шрифтом и относятся к следующим сегментным группам (верхние индексы): 2 → 3 → 4 → 5 (представляют собой последовательность состояний: 16 → 17 → 16 → 18, указанную третьим индексом), а затем еще к одной: 7 → 8 → 9 → 10 → 11 (здесь создается последовательность состояний 19 → 17 → 16 → 18 → 20). Первый идентифицированный цикл задается как C_1, второй - как C_2 и т.д.

Все обнаруженные последовательности групп, которым предшествует группа, принадлежащая состоянию 15 (т.е. первая последовательность предшествует сегменту S^{1}_{1015} и заканчивается сегментом S^{6}_{1515}, а вторая - между сегментами S^{6}_{2515} и S^{12}_{3115}), составляют так называемые циклы. На следующем шаге эти циклы сравниваются с помощью функции подобия циклов are_similar() и (если они соответствуют заданным требованиям) может быть создан так называемый класс циклов. Функция are_similar(cycle1, cycle2) вычисляет статистические свойства сегментов, включенных в циклы cycle1 и cycle2, в соответствии с заданными пороговыми значениями параметров.

Он аналогичен алгоритму 2, однако учитывает все выборки, относящиеся ко всему циклу, а не к конкретному отдельному сегменту. Это объясняется и иллюстрируется далее (рис. 5.6-5.8).

3.2.3. Особенности алгоритма

Подводя итог, оценим сложность разработанных алгоритмов и

комментируем некоторые вопросы, связанные с проблемой выбора подходящих параметров для определения искомых объектов в модели TS. В таблице 3.1 показана временная и пространственная сложность основных функций в соответствующих алгоритмах (указана в скобках) в зависимости от объема обрабатываемых данных. Здесь в качестве верхнего предела сложности используется обозначение big O. Пессимистическое число сгенерированных групп (с использованием алгоритмов 1 и 2) равно количеству сегментов, поскольку каждая группа содержит по крайней мере один сегмент. Количество сегментов можно рассчитать по формуле: M=N/n, где n - параметр алгоритма определения размера сегмента (функция create_segments). Процедура detect_edges имеет квадратичную сложность, поскольку в процессе определения состояния каждое новое состояние сравнивается с каждой немаркированной группой. Пространственно-квадратичная сложность detect_edges обусловлена возможностью генерации плотного графа. Процедуры merge_simple_sequences и identify_simple_sequences выполняют поиск в графе, как в алгоритме поиска в глубину, что приводит к линейной временной сложности в зависимости от количества состояний и ребер в графовой модели. Процедура определения базового состояния линейна по времени (зависит от количества ребер E). Операция поиска всех циклов также является линейным процессом и может генерировать не более такого количества циклов, как группы. Функция сравнения обнаруженных циклов (find_similar_cycles) показывает квадратичную сложность с числом циклов C. Таблица 3.1 не содержит алгоритм 9, поскольку он состоит из алгоритмов 10-12. Параметры сложности основаны на выборках (N=nM) и количестве сегментов (M). Оценки сложности разработанных функций (алгоритмов) выражены в зависимости от параметров N, P, S, E и C, указанных в третьем столбце таблицы 3.1. Число состояний (S) и групп (P) линейно связано с M. Число ребер в плотном графе обозначим P^2. Это

значение может быть преобразовано (масштабировано) в тот же параметр, например, M (количество сегментов), что приводит к эквивалентной сложности порядка M^2. Следовательно, общая сложность обработки TS во времени равна $O(M) + O(M^2) + O(M) + O(M^2) + O(M) + O(M^2) + O(M^2) + O(M) + O(M^2)$, что приводит к $O(M^2)$. Аналогично, сложность в пространстве также равна $O(M^2)$ из-за сложности компонента $O(P^2)$, соответствующего выполнению алгоритма 6.

Таблица 3.1

Временная и пространственная сложность основных функций

Функция	Размер данных	Временная сложность	Пространственная сложность
create_segments	N - к-во выборок	O(N)	O(N)
create_groups (1,2)	M - к-во сегментов	O(NM)	O(N)
increase_accuracy (3)	P - к-во групп	O(P)	O(1)
detect_states (4,5)	P - к-во групп	$O(P^2)$	O(P)
detect_edges (6)	P - к-во групп	O(P)	$O(P^2)$
merge_simple_sequences (7)	S - к-во состояний	O(S + E)	O(S)
identify_simple_sequences (8)	E - к-во ребер		
detect_base_state (10)	E - к-во ребер	O(E)	O(1)
find_cycles (11)	P - к-во групп	O(P)	O(P)
find_similar_cycles (12)	C - к-во циклов	$O(C^2)$	0

Выбор параметров для алгоритмов может быть выполнен тремя способами:

1) на основе предыдущего опыта анализа (например, взятие параметров из ранее проанализированной версии проекта для проверки свойств следующей версии),

2) на основе практического опыта эксперта, который знает ожидаемые классы циклов (их периодичность, уровни мощности и т.д.),

3) итеративный поиск в экспериментах с репрезентативным

фрагментом изучаемой TS или полным фрагментом (сравните раздел 5).

Последние два подхода основаны на некоторых эвристических свойствах. В целом, размер сегмента n достаточно велик, чтобы охватить профиль энергопотребления базового состояния, в частности периодически выполняемые фоновые процессы. Они не представляют интереса для анализа, в отличие от действий устройства, соответствующих классам циклов – целевым действиям. Начальное значение n может быть рассчитано как $k*T_{max}$/(период выборки), где T_{max} - максимальный период фоновых процессов, множитель k гарантирует, что в один сегмент включено несколько периодов. Верхний предел значения n зависит от самой короткой целевой активности. Слишком большое значение n приводит к включению некоторых выборок базового состояния в целевую активность, что нарушает ее энергетический профиль. Алгоритм отличает базовое состояние от других, используя различия в статистических параметрах профилей потребления энергии. Пороговые параметры (avg_eps, min_eps, max_eps, ...) позволяют различать целевые действия/состояния. Слишком высокие значения могут привести к отключению обнаружения некоторых классов циклов. Целевые действия с короткими пиками тока различаются с помощью параметра max_eps. С другой стороны, энергетические состояния, которые сохраняются в течение длительного времени и приводят к увеличению минимального потребления тока, различаются по параметрам min_eps. При поиске параметров аналитик может взять фрагмент анализируемой TS, содержащий некоторые целевые действия, отметить базовое состояние и эти действия, а затем рассчитать соответствующие уровни мощности и найти значения различий. Расчеты могут поддерживаться с помощью специального программного модуля. Создание различных объектов (сегментов, групп, состояний, графиков) обеспечивает доступ ко всем соответствующим атрибутам (метрикам) и элементам, которые вносят

вклад в рассматриваемый объект (в частности, ко всем соответствующим выборкам). Это облегчает исследование искомых свойств, например, энергопотребления, относящегося к конкретным состояниям, последовательности состояний. Практическая полезность предложенной модели TS и разработанных алгоритмов была подтверждена при анализе энергопотребления электронных устройств. Это проиллюстрировано в следующем разделе, где также говорится о разработанных алгоритмах. Прилагаемые графические схемы помогают понять обработку сигналов с помощью этих алгоритмов, которые также могут быть применены для решения других задач.

3.4. Тематические исследования моделей временных рядов

Представленные модель и алгоритмы были использованы при анализе устройств Монитора (небольшие габариты и низкое энергопотребление). Для оптимизации энергопотребления потребовались глубокие исследования тока батареи для различных сценариев тестирования, соответствующих работе устройства. Для этой цели был создан специальный испытательный стенд, содержащий “поддельную батарею” и устройство сбора данных, передающее собранные данные (образцы значений тока, передаваемых от батареи к контролируемому устройству) на управляющий компьютер через Ethernet. Представленные результаты относятся к трем экспериментам. В первом из них объясняется общая идея нашей методологии, а именно определение групп сегментов, их уточнение и создание соответствующего графа состояний в соответствии с используемыми алгоритмами и предполагаемыми параметрами для иллюстративного фрагмента TS. Последующие два эксперимента были посвящены временным рядам, зарегистрированным для двух типов разработанных аппаратов Монитора: Монитор#1 и Монитор #2, отличающихся своей функциональностью и платформами

реализации. В этих экспериментах используются реальные TS, связанные с более длительным периодом наблюдения, в котором задействовано много образцов. Таким образом, проведенный анализ выявил различные объекты (сегменты, группы сегментов), что привело к созданию относительно сложных графиков состояний и получению характерных классов циклов. Представленные рисунки сопоставляют фрагменты выборочных графиков (фоновый график) с идентифицированными состояниями (помеченными метками) и классами циклов, полученными для заданных параметров алгоритма. Влияние параметров алгоритма на полученную модель TS проиллюстрировано и прокомментировано в экспериментах с Монитором №2. Полученные характеристики объектов TS интерпретируются в соответствии с поведением/активностью устройства. Этот процесс сопровождался отслеживанием коррелированных событий журнала в соответствии со схемой анализа, описанной ранее. Управляющий компьютер включает в себя разработанную аналитическую программную систему (TS_AN) с реализованными алгоритмами для создания и исследования соответствующей модели TS. Используемые в алгоритмах структуры данных обеспечили высокую гибкость обработки в TS_AN, включая расширенную визуализацию данных и моделей. Собранные данные TS были обработаны с использованием компьютера следующей конфигурации: процессор INTEL core i5-8600 K 3,6 ГГц (64-разрядный), оперативная память: 32 ГБ, SSD-диск: Samsung 860 EVO объемом 1 ТБ.

3.4.1. Наглядный эксперимент

На рис. 3.1 приведена выдержка TS, относящаяся к току питания устройства (полученному от фальшивой батареи); границы сегментов показаны вертикальными линиями. Сегментация TS предполагает 500 выборок с периодом 200 мс на сегмент. На рис. 3.1 показаны не только значения собранных выборок, но и суммарные значения среднего текущего

значения (красная линия), т.е. среднее значение, рассчитанное по всем выборкам с начала TS до текущего времени на оси x. Группы сегментов были сгенерированы с помощью алгоритма 1. В этом процессе были определены следующие параметры распознавания (используемые при групповом и сегментном сравнении алгоритма 2): (1) avg_eps, (2) dev_eps, (3) min_eps, (4) max_eps как: 3 мА, 5 мА, 1 мА, 10 мА соответственно. Эти значения были выбраны для обеспечения возможности обнаружения 5 состояний. Значение avg_eps в 3 мА отличает состояние 1 от 4, min_eps (1 мА) отличает состояния 2 и 5 от других, и, наконец, параметр dev_eps гарантирует, что состояния 3 и 5 будут различимы. Таким образом, на рис. 3.1 были выделены 6 групп сегментов, выделенных разными цветами (на основе алгоритмов 1 и 2). Процесс уточнения группы показан на рис. 3.2 (на основе алгоритма 3). В результате зеленая группа была расширена за счет подсегмента следующей синей группы (теперь уточненный подсегмент обозначается зеленым). Последний сегмент зеленой группы был соседним с последующей группой одиночных сегментов, поэтому эта группа была разделена, и идентифицированная левая часть была добавлена к зеленой группе. В этом процессе было учтено условие подобия (включая указанные выше параметры).

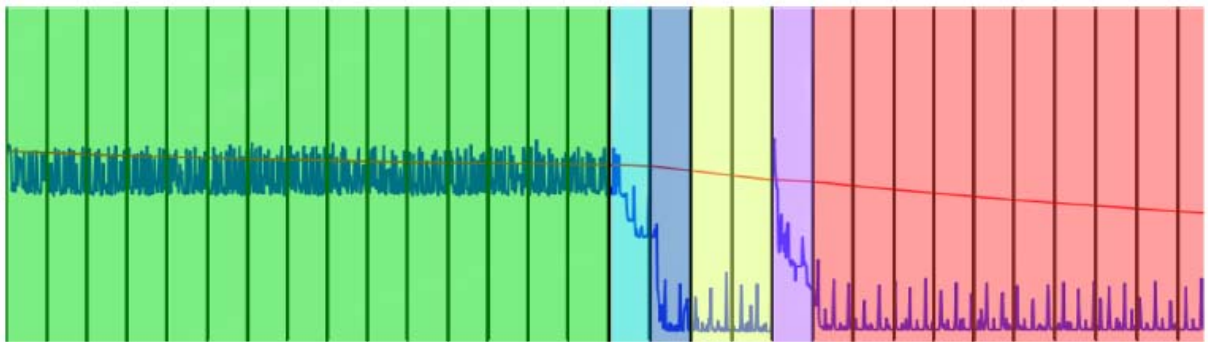


Рис. 3.1. Идентификация групп сегментов

Следующим шагом является присвоение состояний (маркировка – в соответствии с алгоритмами 4 и 5). С учетом ранее рассчитанных

характеристик идентифицированных групп было создано 5 состояний: S1-S5. Метки состояний присваиваются последующим группам в порядке возрастания. Для каждой группы проверяется, существует ли аналогичная группа. Если это так, то метка выхода присваивается следующей аналогичной группе. Это произошло в случае последней красной группы (шестой группы) на рис. 2, поскольку она аналогична (по отношению к вычисленным параметрам) желтой группе (группа 4), которая была назначена состоянию S4, следовательно, красная группа также получает метку состояния S4. Для рассматриваемого фрагмента TS с идентифицированными состояниями выводится соответствующий граф состояний (алгоритмы: 4 – для состояний и 6 – для ребер), показанный на рис. 3.3. Состояния 1-3 могут представлять собой последовательность состояний, поэтому их можно объединить в одно агрегированное состояние 1 (ср. алгоритмы 7 и 8).

Для реальных TSs полученные графики обычно довольно сложны. Алгоритмы, описанные ранее, были включены в разработанный анализатор TSS TS_AN, который использовался при анализе энергопотребления. Эта программа принимает в качестве входных данных файл, содержащий собранные образцы данных, относящиеся к цели анализа, и применяет разработанные алгоритмы для обнаружения соответствующих объектов TS, создания соответствующих структур данных с вычисляемыми свойствами и графиком состояния. Кроме того, он предоставляет интерактивный графический интерфейс с широкими возможностями визуализации, например, графические образцы TS с отмеченными обнаруженными состояниями. Можно выбрать визуализацию требуемого временного диапазона, получить подробные данные за указанные моменты времени (с помощью курсора) и т.д. Это обеспечивает полезные функции навигации по графику, такие как: возврат к исходному виду, изменение вида, ручное перемещение по графику для

выбора вида искомого фрагмента, увеличение/уменьшение вида и т.д. TS_AN написан на Python с включенными библиотеками: Matplot (навигация и представление графиков) и Numpy (статистические вычисления и обработка памяти для больших наборов данных). Возможности этого инструмента, в частности эффективность разработанных алгоритмов, продемонстрированы в комплексных исследованиях TS, собранных с реальных устройств. Производные объекты TS проиллюстрированы соответствующими фрагментами графиков TS, и они завершаются общей статистикой или комментариями по интерпретации.

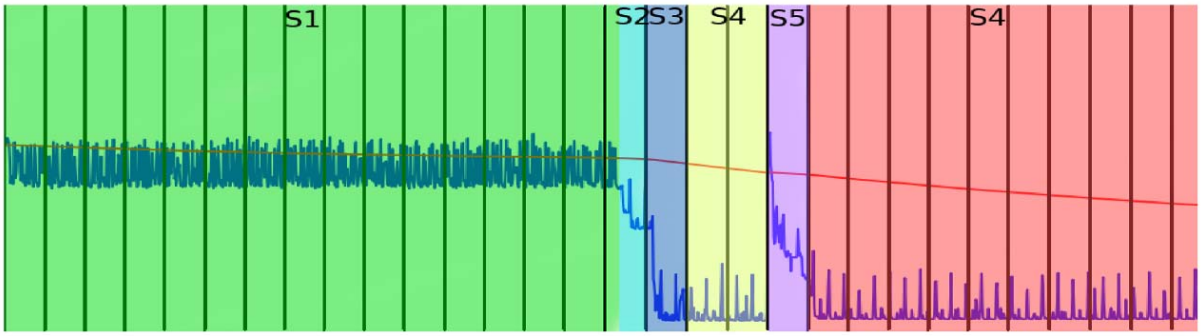


Рис. 3.2. Уточнение групп и обозначение состояний

Рис. 3.3. Граф состояния, соответствующий TS, за исключением приведенного на рис. 3.2.

3.4.2. Анализ временных рядов по Монитору №1

Монитор №1 позволяет отслеживать график состояния объекта в течение более длительного времени, например, одного месяца, и отправлять обобщенные результаты или обнаруженные аномалии в центр обработки по GSM-сети [3.14]. Более тщательный анализ в центре обработки также может привести к предупреждению объекта или даже автоматическому вызову бригады стабилизации. Монитор#1 работает под

управлением специально разработанного прикладного программного обеспечения в операционной системе Android и использует модель распределенного энергопотребления.

Система управления питанием процессора (CPU) разделена на области, которые могут находиться как минимум в одном из четырех различных состояний энергопотребления: ВКЛЮЧЕНО, ВЫКЛЮЧЕНО, УДЕРЖАНИЕ, ЗАВЕРШЕНИЕ работы. Он может работать при различных уровнях напряжения (от 1,2 В до 1,8 В) и тактовых частотах (до 800 МГц). Большинство трансиверов и других микросхем могут питаться по требованию или в режиме ожидания. Все повышающие и понижающие резисторы портов ввода/вывода могут быть активированы в разных состояниях.

Время выполнения периодических задач, таких как считывание данных с аналого-цифрового преобразователя (АЦП), выполнение алгоритма классификатора потока, запись данных потока в энергонезависимую память и передача данных на сервер центра обработки, оказывает огромное влияние на средний ток и определяет, сможет ли устройство работать в течение 20 или 40 часов непрерывной работы. Конфигурация питания для такого устройства включает в себя множество параметров, принимающих различные значения в различных состояниях устройства, например, во время передачи пакетов данных или пребывания в состоянии спящего режима.

Работа устройства предполагает взаимодействие программных и аппаратных компонентов, поэтому его исследование может быть поддержано с помощью внедренной иерархической модели TS и разработанного инструмента. Проведенный анализ репрезентативной TS охватывает 2 часа 50 минут наблюдения, что соответствует 4 644 164 выборкам данных (период выборки 2 мс).

Создание модели TS заняло около 10 минут на одном ПК,

конфигурация которого уже была приведена ранее. Результаты анализа временных рядов, относящихся к току питания, обеспечиваемому поддельной батареей, описывают типичный сценарий работы устройства. Выходной ток батареи контролировался прибором сбора данных Keysight, который отправлял (через Ethernet) собранные образцы данных на компьютер PC с установленной программой TS_AN.

Представленные результаты относятся к эксперименту, который начался с исследования в 13:04 и закончился в 15:40. Период выборки был установлен равным 2 мс, диапазон измеряемого тока - от 0 до 3 А с 262 144 уровнями выборки. Эксперимент включал в себя такие действия, как включение и загрузка устройства, включение/выключение дисплея устройства, оповещение медицинского центра о происходящих событиях и проведение мониторинга потока данных (сигналы датчика поступали из предварительно зарегистрированного источника данных).

Собранные необработанные данные (484 МБ) были отправлены на анализ TS_AN со следующими параметрами: количество выборок в сегменте n=500; параметры сравнения: 1) avg_eps=0,01A, 2) dev_eps=0,009A, 3) min_eps=0,007A, 4) max_eps=0,050A.

Значение параметра n зависит от частоты дискретизации измерительного прибора и знаний пользователя о поведении тестируемой системы. Параметру avg_eps было присвоено значение 0,01A, что гарантировало стабильное определение базового состояния. С другой стороны, низкое значение min_eps позволяет обнаруживать даже кратковременные аномалии. Значения параметров были установлены в ходе 9 запусков экспериментального алгоритма.

После каждого выполнения пользователь оценивает качество определения базового состояния (например, проверяет, не перекрывает ли оно некоторые известные и видимые аномалии). В дальнейшем полученные результаты иллюстрируются и интерпретируются.

Полученные объекты TS (например, состояния, классы циклов) сопоставляются с применяемыми алгоритмами анализа и соответствующими последовательностями выборок в пределах представленных участков графика TS (с использованием меток и других цветовых обозначений). Некоторые параметрические характеристики также включены в комментарии.

На рис. 3.4 показана полученная TS-запись первых 5 минут операции Монитора. Первая часть относится к определенной последовательности состояний (обозначенной как S0), соответствующей включению устройства до завершения последней активации услуги системной программы. Фактически, это состояние представляет собой последовательность идентифицированных последовательных состояний S0-S14 (без разветвления, один вход, один выход), объединенных алгоритмами 7 и 8. В следующем состоянии, S15, потребление тока находится на самом низком уровне - 36 мА. Выявленные состояния могут быть соотнесены с ходом выполнения программы (сценария) Монитора.

Этот процесс может быть поддержан отслеживанием событий журнала, что облегчает интерпретацию поведения устройства в соответствии с действиями его компонентов. В состоянии S16 более высокий импульс тока возник в результате взаимодействия с сервером протокола динамической настройки хоста (DHCP) для получения адресов Интернет-протокола (IP) и системы доменных имен (DNS) для устройства. В 13:08:36 устройство обнаружило отсутствие ответов на запросы модема Universal Serial Bus (USB).

Это привело к обновлению количества USB-устройств, и Монитор получил подключение к Интернету через название точки доступа (APN) оператора мобильной связи стандарта Long Term Evolution (LTE). Одним из элементов состояния S17 было возобновление связи с DHCP-сервером, которое было видно в виде импульса тока 460 мА. Перейдя в спящий

режим (состояние S15), Монитор пробудился (состояние S20) и освободил оперативную память (активированный сборщик мусора виртуальной машины Dalvik в операционной системе Android). Последующие состояния – S21, S23 и S24 – описывают момент повторного нарушения связи между процессором и модемом LTE.

Для попытки восстановления связи требуется повторное перечисление USB и инициирование стандартного обмена командами Attention (внимание). Модем не ответил на первую команду AT. Процедура восстановления связи с модемом настраивает инфраструктуру USB и модем LTE с помощью сигналов, управляющих питанием USB и модема.

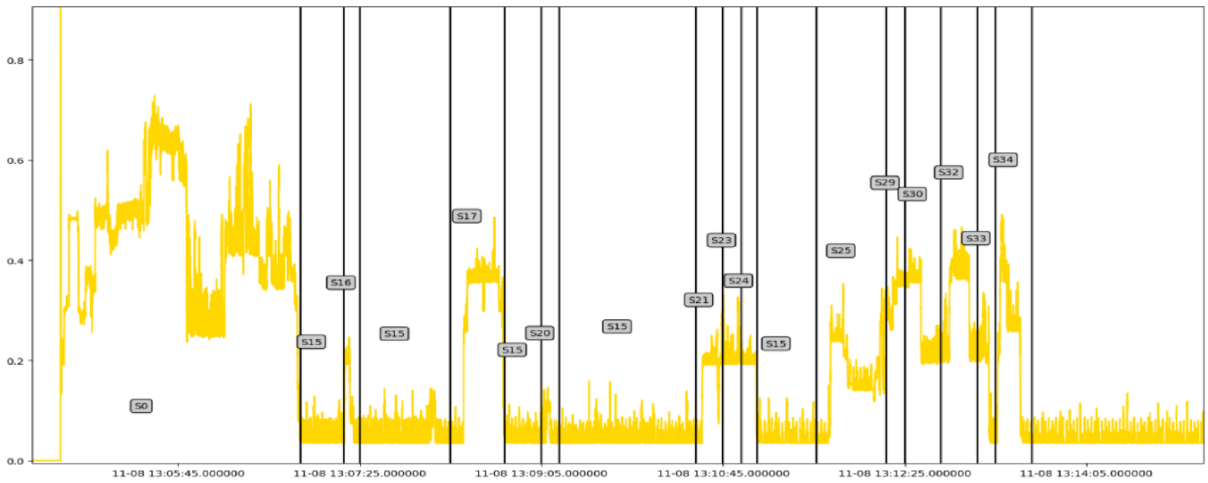


Рис. 3.4. Начальная часть рассматриваемого временного ряда.

Это приводит к установке режима питания, отключающего эти компоненты для перехода в спящий режим. Следовательно, средняя нагрузка по току для трех рассмотренных состояний составляет 223 мА. В ожидании ответа (30 секунд) модем был отключен от питания, и устройство вернулось в состояние S15. По истечении следующих 40 секунд модем был подключен повторно, и Монитор инициировал процедуру настройки модема с помощью AT-команд (включая получение IP-адресов и DNS-адресов из службы DHCP). Этот процесс приводит к текущей нагрузке, заданной состояниями: S25, S29, S30, S32, S33 и S34.

После такой настройки Монитор переходит в режим S15 (USB, процессор и модем находятся в спящем режиме, а затем процессор может быть активирован). Аналогичным образом, остальные части зарегистрированного TS можно отследить, обратившись к зарегистрированным журналам регистрации.

Для проанализированной TS сгенерированный граф состояний содержал 99 состояний. Примерно для 50 из этих состояний количество входных и выходных ребер было равно 1, следовательно, при использовании алгоритма 7 первичный граф был сокращен до 49 состояний и показан на рис. 3.5. Стоит отметить, что, несмотря на относительно большое количество состояний, сложность графа средняя. Для нередуцированного графа были получены следующие значения: мощность ребер и узлов |E|=140 и |S|=88, диапазон входных значений=1-7, диапазон выходных значений=1-7, плотность графа |E|/(|S|)2=0,02. На уменьшенном графике эти параметры следующие: E=|70|, |S|=49, плотность 0,28. Более того, только два состояния содержат 20 ребер, а остальные имеют гораздо меньше значений. Следовательно, их анализ может быть выполнен вручную. Щелчок по указанному состоянию на визуализированном графическом графике TS отображает соответствующий фрагмент TS. На этом графике некоторые состояния соответствуют агрегированным последовательностям состояний (группам) из основного графика (например, S15, S29). Стоит отметить, что для этих последовательностей некоторые параметры систематически увеличивались. Разработанная система позволяет нам выявлять эти состояния и отслеживать соответствующие фрагменты TS для проверки. Алгоритм 11 выполняет поиск групп последовательностей состояний (циклов), в качестве базового состояния он принимает состояние S49. Базовое состояние состоит из множества низкоуровневых и коротких периодических импульсов (фоновая активность) – сравните рис. 3.6-3.8.

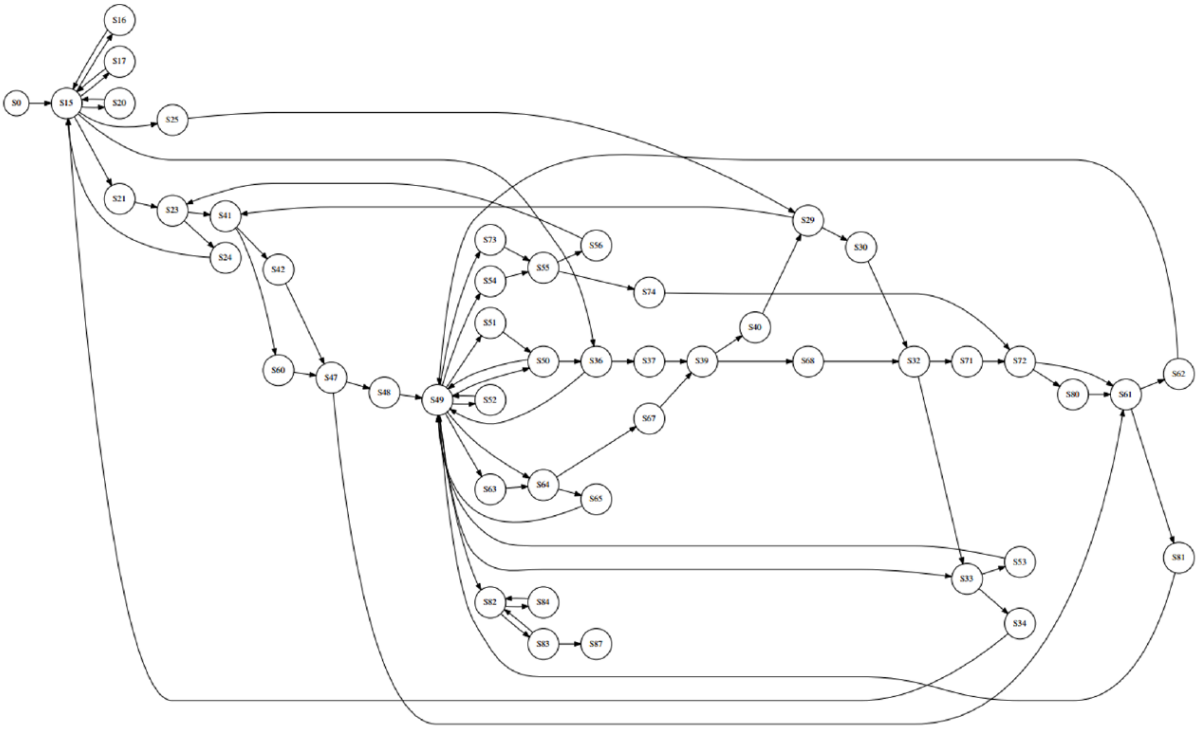


Рис. 3.5. Граф состояний анализируемого временного ряда

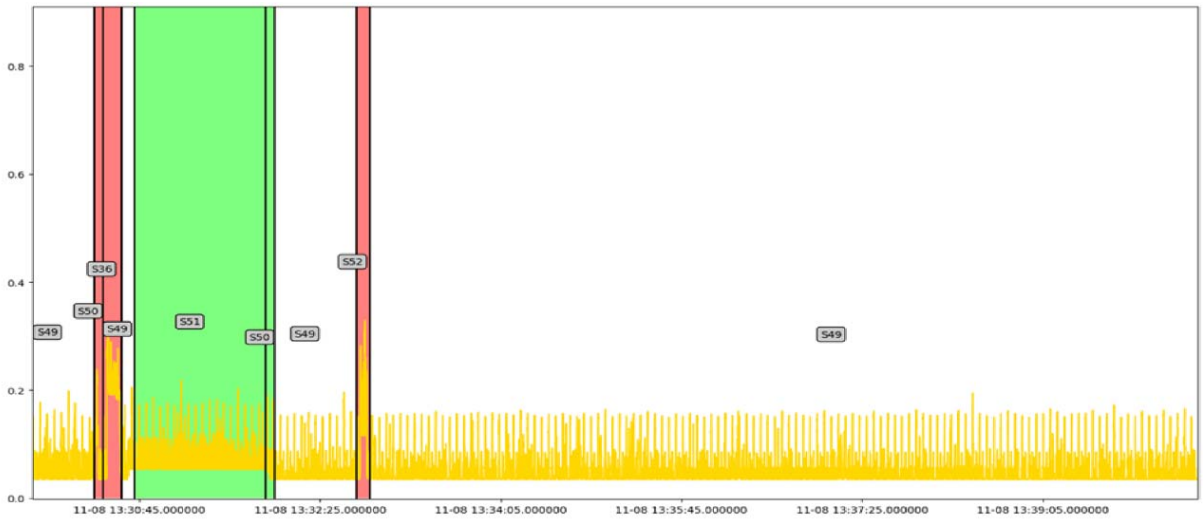


Рис. 3.6. Цикл класса CC1 (красный) и CC2 (зеленый)

Первый обнаруженный цикл (C1) отмечен на рис. 3.6 красным цветом. Его первое появление (создание экземпляра) включает состояния S50 и S36 (15–секундный скачок тока, возникающий в результате включения экрана Монитора для отображения сообщения). Следующий цикл (C2 – отмечен зеленым цветом) включает состояния S50 и S51

(повышенный уровень тока на 52 мА из-за пробуждения модема в ответ на событие радиосвязи). Третий обнаруженный цикл C3 также отмечен красным, поскольку он относится к тому же классу циклов CC1, что и цикл C1. На рис. 3.7 и 3.8 показаны два последующих цикла класса CC3 (синим цветом). На рис. 3.7 он содержит состояния S54, S55, S56, S23, S41, S60, S47, S61 и S62. Аналогичная последовательность приведена на рис. 3.8.

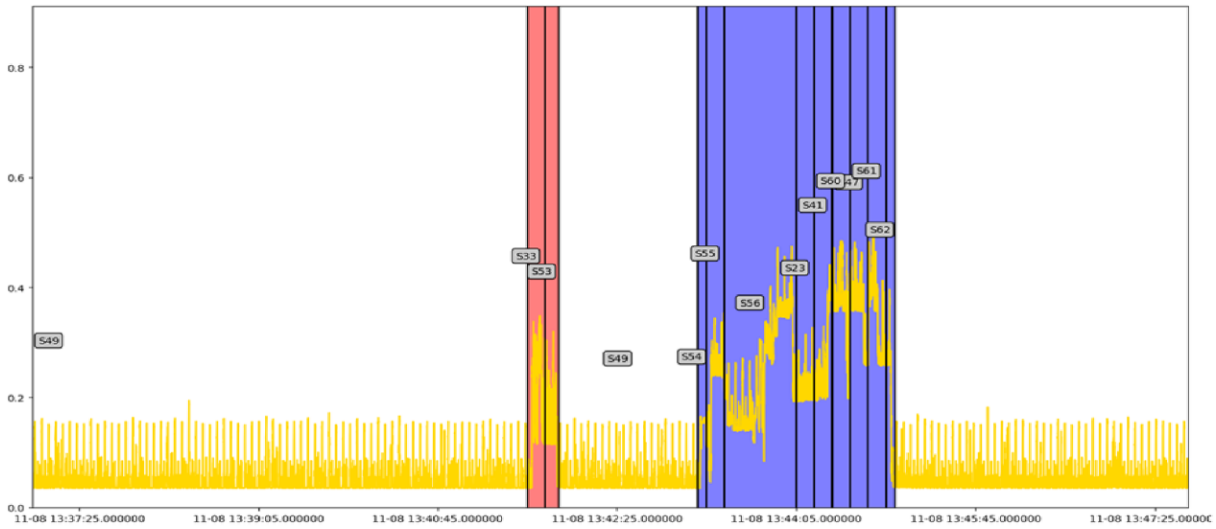


Рис. 3.7. Первое появление цикла класса CC3 (синий)

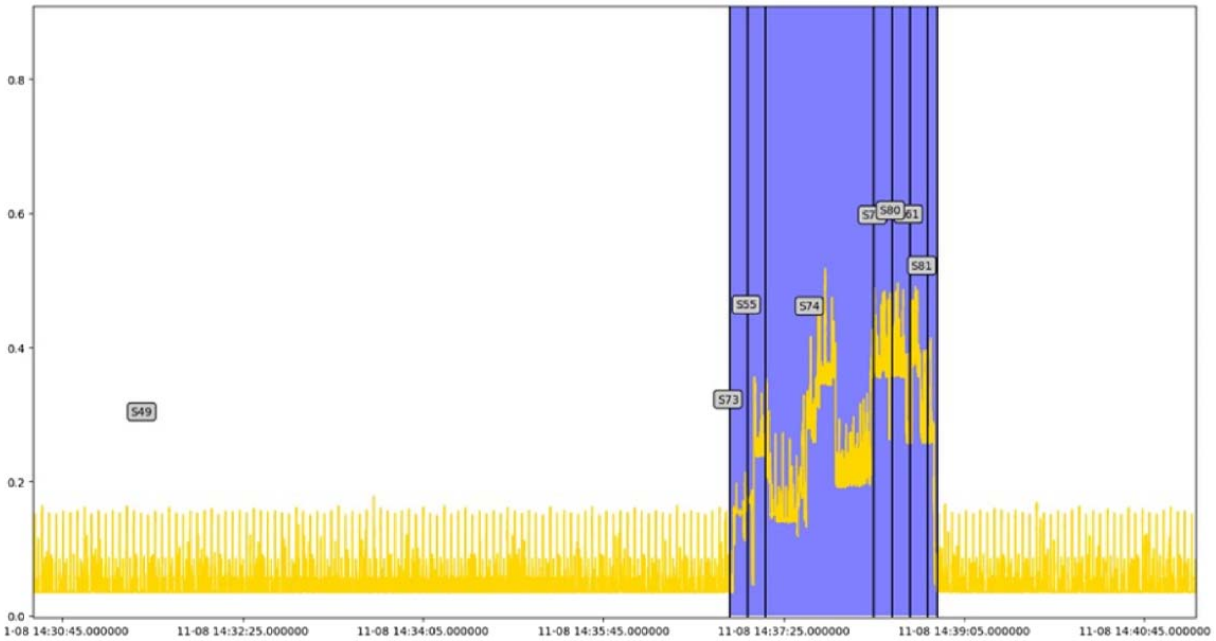


Рис. 3.8. Второе появление цикла класса CC3 (синий)

Эти последовательности содержат общие состояния S61 и S62,

связанные с активностью модема. Третий экземпляр класса CC3 появился в течение периода (14:36:40, 14:40:00), ему предшествовал и за ним следовал более длительный период, относящийся к состоянию S49. Разнообразие включенных состояний является результатом колебаний состояния сети, чередования потоков программного обеспечения и т.д. Все состояния, относящиеся к CC3, связаны с последовательностями событий одного типа, такими как настройка модема LTE, сетевых интерфейсов и отправка данных на удаленный сервер. Тем не менее, потребление энергии аналогично.

3.4.3. Анализ временных рядов по Монитору№2

Монитор#2 - это разработанное гибкое устройство Монитор Patch, которое наклеивается непосредственно на объект для мониторинга сердца по одному каналу и обычно используется в течение 14 дней наблюдения. В отличие от предыдущего устройства, обработка данных упрощена. Его работа контролировалась (на том же испытательном стенде, что и для Монитора №1) по сценарию тестирования, включающему форму сигнала потокаданных, соответствующую примерно 19 часам наблюдения (35809091 выборка необработанных данных, период выборки 5 мс).

Система анализа (TS_AN) позволила нам получить сокращенный график состояний, состоящий из 35 состояний, при этом базовое состояние включало 17 и 14 входных и выходных ребер соответственно (для большинства состояний это было меньше 8 в общей сложности). Базовые состояния состоят из множества импульсов (с периодом 150 мс), предполагающих амплитуду около 1,8 мА (длительность 25 мс) при токе отключения питания 40 мкА. Алгоритм 12 идентифицировал 14 классов циклов (CC0-CC13) с одним доминирующим (CC2) – с наибольшей частотой. Количество экземпляров классов циклов составило 2104.

Монитор №2, по сравнению с предыдущим прибором, показывал

больше классов циклов, и эти классы также были более разнообразными, они включали больше состояний. К сожалению, этот прибор не регистрировал события, поэтому интерпретация требовала больших усилий. Сосредоточение внимания на цикле доминирующего класса показало, что он создает более длинные последовательности со встроенным циклом CC6. Этому циклу предшествует один цикл, относящийся к обмену данными через интерфейс NFC (один из набора: CC5, CC8, CC10, CC11), например, CC2, CC22, CC5, CC6, CC2, CC22, - эти циклы разделены базовыми состояниями.

Однако за последовательностью, содержащей CC5, не последовало CC6, что было аномалией, вызвавшей увеличение тока на более длительное время (2 с вместо 120 мс). Это произошло из-за ошибки в программном модуле связи. Это стало причиной некоторого дополнительного энергопотребления. Более того, в одном учебном цикле были выявлены некоторые временные отклонения. Это может быть обнаружено благодаря компактности агрегированной модели, ее прозрачности и возможности получения подробных данных на более низком уровне (например, о состоянии).

Классические методы обнаружения знаний также могут быть использованы при анализе поведения устройств на объектах более высокого уровня (например, классах циклов). Проведенные эксперименты с аппаратом Монитора№2 касались следующих параметров: размер сегмента n=225, среднее значение eps=0,40, dev_eps=0,35, минимальное значение eps=0,50 и максимальное значение eps=4,5 (в мА). Эти параметры были выбраны экспертом, изучившим фрагмент TS, относящийся примерно к 15 минутам, в соответствии с подсказками, приведенными в ранее.

В частности, выявленный максимальный период фоновой периодической активности составил 150 мс, время выборки - 2 мс, а

предположение о k=3 привело к параметру n=225. Пожалуйста, обратите внимание, что Монитор#2 реализован по технологии, отличной от Монитор#1, он работает на более низком уровне мощности, поэтому значения параметров eps также значительно ниже.

Модель TS была создана в течение 27 минут (максимальный размер резидентного набора: 2,64 ГБ). Эксперименты были повторены для различных значений параметров, чтобы проверить их влияние на результаты алгоритма. Изменение параметра n для тех же остальных параметров показало, что увеличение значения n уменьшает количество классов циклов. Количество обнаруженных аномалий изменяется медленнее, чем количество состояний с n значениями. Если n установлено на какое-то высокое значение, это приводит либо к неправильному определению базового состояния, либо к маскировке аномалий/событий.

Например, при n=10000 число экземпляров классов циклов сократилось с 2104 до 1473, при этом были выделены только 5 состояний и 5 классов циклов. Предполагая, что n=675 привело к 20 состояниям, 9 классам циклов с 2082 экземплярами; а для n=32 анализ показал 44 состояния, 17 классов циклов с 2090 экземплярами.

Время, необходимое для построения модели TS, составило 11 минут (2,64 ГБ) и 1 час 24 минуты (3,4 ГБ) соответственно. Время выполнения обратно пропорционально размеру сегмента (как показано в таблице 3.1). Дальнейшее снижение n привело к значительному увеличению классов циклов (что не соответствует профилю работы устройства). Для фиксированного значения n=225 и значения eps_dev=0,35 было изучено влияние остальных параметров. Для значений avg_eps=0,20, min_eps=0,25 и max_eps=2,25 проведенный анализ выявил 51 состояние, 29 классов циклов и 134 их экземпляра. Для avg_eps=0,80, min_eps=1,50 и max_eps=9 это составило 14 состояний, 15 классов циклов и 2106 их экземпляров. Увеличение значений параметров avg_eps, min_eps, max_eps, при

некотором уровне значений следующее увеличение оказывает небольшое влияние на результаты. Уменьшение значений *_eps в описанном случае привело к увеличению числа состояний.

Когда система обнаруживает ложные состояния, возможно, алгоритм показывает неверное базовое состояние. Неправильное базовое состояние, в свою очередь, приводит к обнаружению ложных классов циклов. Это наблюдалось в первом случае (avg_eps=0.2) – количество экземпляров классов циклов снизилось с 2104 до 134 из-за неправильного определения базового состояния. Проверку правильности базового состояния можно легко выполнить, сопоставив его с некоторой начальной частью анализируемой TS.

3.5. Сопоставление модели временных рядов с логарифмами

Полученная модель TS дает общее представление о работе системы, ориентированной на исследуемые параметры производительности (в рассматриваемом примере – энергопотребление). Более глубокое понимание соответствующих программных/аппаратных процессов требует отслеживания системных/прикладных журналов. Здесь важным вопросом является сопоставление модели TS с этими журналами. Собранные образцы сигналов и производные объекты модели TS задаются временными метками, привязанными к временной шкале системы мониторинга. Системные журналы/ журналы приложений содержат записи о событиях/отчетах с локальными временными метками контролируемого устройства (в нашем случае Монитор#1), они основаны на локальных часах, иногда синхронизируются с сетевыми часами, настраиваются пользователями и т.д. Более того, точность обеих временных шкал может отличаться, также время от времени могут появляться ложные временные метки. Как следствие, обе временные шкалы должны быть сопоставлены. Другой проблемой является извлечение соответствующих записей

журнала, связанных с интересующими объектами, например, последовательностями состояний (циклами). Этим вопросам уделяется мало внимания в литературе по логарифмическому анализу [3.15, 3.18].

Для корреляционного анализа были введены некоторые специальные алгоритмы. Они используют нормализованную временную шкалу с исходным временем 0 – начальной точкой эксперимента. Алгоритм корреляции состоит из 3 шагов:

1. Каждая запись журнала преобразуется в единый набор классифицированных слов, каждое из которых представлено парой <слово, класс слова>, где класс слова может быть одним из следующих: источник события, идентификатор процесса, отметка времени, слово сообщения, числовое слово.

2. Чтобы найти коррелирующие события в анализируемом классе циклов, извлекаются временные интервалы последующих реализаций циклов (C_i) в этом классе.

3. Сопоставление последовательностей пакетов слов с классом циклов.

Самым сложным является третий шаг. Он начинается с поиска похожих наборов слов на основе предопределенной метрики сходства, здесь указываются соответствующие веса для разных классов слов (в записях журнала). Показатель сходства определяется как взвешенная сумма количества общих (равных) слов, деленная на взвешенную сумму всех слов. Два набора слов считаются похожими, если показатель сходства превышает заданный порог $\varepsilon_{similarity}$. Наборы похожих пакетов создаются путем сканирования списка пакетов со словами. Алгоритм берет первый пакет слов и создает новый набор пакетов с этим первым пакетом. Все остальные пакеты слов сравниваются с каждым из уже включенных элементов текущего созданного набора - если они соответствуют критериям сходства, они добавляются в набор. Наконец, созданный набор

похожих пакетов слов добавляется в контейнер. Процедура повторяется до тех пор, пока все пакеты слов не будут распределены по некоторым наборам.

Каждый набор похожих пакетов слов (SBW) разбивается на последовательности пакетов. Последовательность можно рассматривать как список, состоящий из отсортированных пакетов (в соответствии с временными метками) слов, которые удовлетворяют следующему критерию: для каждого элемента последовательности разница между его временной меткой и временной меткой каждого другого элемента, находящегося вне последовательности, превышает предопределенный параметр ε_T. Можно заметить, что если похожие записи были зарегистрированы в течение небольшого периода времени, то вполне вероятно, что они были вызваны одним и тем же событием/причиной. Процедура генерации последовательности пытается найти такие записи в наборах похожих пакетов. На основе этих последовательностей алгоритм создает массив временных интервалов между последовательными пакетами (в пределах каждого SBW). Для созданного массива генерируются статистические параметры с соответствующими временными метками (медиана, среднее значение, стандартное отклонение).

Затем используются статистические параметры для выбора значимых наборов пакетов. Обычно встроенные системы имеют множество периодических задач, которые можно рассматривать как обычные, за исключением тех, в которых пропущены или сдвинуты элементы. Другие аномалии могут возникать случайным образом во времени. Следовательно, набор похожих словосочетаний с большим разбросом во времени между последовательностями словосочетаний с большей вероятностью коррелирует с аномальным поведением.

Например, два набора пакетов из проанализированных TS были

классифицированы как несущественные из-за их временной статистики (медиана, среднее значение, стандартное отклонение времени в секундах и количество элементов в наборе): 1) (0, 0, 0, 1267) - только одна последовательность и 2) (336.4, 339.2, 67.0, 522) - операции периодической системы.

Значения разброса невелики по сравнению с набором пакетов слов, который классифицируется как значимый - выборочные значения разброса: (449.9, 931.8, 984.0, 102).

Алгоритм пытается найти наборы с более высокими значениями разброса и предоставляет их для дальнейшего анализа. Идентифицированные последовательности для каждого набора похожих пакетов слов затем сопоставляются с временными диапазонами, полученными на шаге 2. В процессе сопоставления создается список кандидатов. Каждый кандидат представляет собой список последовательностей пакетов, в котором все временные метки корректируются по смещенному значению времени.

Процедура генерации кандидатов является итеративным процессом. Каждая итерация включает в себя вычисление нового времени смещения, смещение каждой временной метки в последовательностях и проверку того, могут ли все последовательности быть сопоставлены с временными метками рассматриваемого класса циклов (CC). Точность и количество кандидатов можно настроить с помощью параметра $\varepsilon_{\text{T-tolerance}}$, который определяет значение шага увеличения времени смещения. Чем меньше значение $\varepsilon_{\text{T-tolerance}}$, тем больше кандидатов генерируется, что приводит к более широкому набору кандидатов, передаваемых на следующий шаг. Затем для заданных наборов кандидатов алгоритм находит значение смещения, которое позволяет выбрать кандидатов для большинства наборов пакетов. Эти последовательности с метками, связывающими журналы записей с данными экземплярами CC (и, как следствие,

описывающими событие аномалии), являются результатом работы алгоритма и могут быть использованы при более глубоком анализе. TS содержал 19716 зарегистрированных записей в журнале, относящихся к периоду времени [11-08 07:05:04.093; 11-08 09:43:38.651]. Для отслеживания аномального цикла класса CC3 были рассмотрены следующие временные диапазоны (сравните раздел 6, фиг. 7 и 8): {(13:43:10, 13:45:00), (14:07:25, 14:09:15), (14:36:40, 14:40:00)} – указывается дата, час, минута, секунда (местное время). При сопоставлении журналов событий с этими диапазонами были приняты следующие параметры (настроенные экспериментально):

1) порог сходства слов $\varepsilon_{similarity}$=0,3;

2) ε_{T}=2 мин 8 с;

3) $\varepsilon_{T\text{-}tolerance}$=1 с;

4) веса классов слов WT(x)={word=1,0; pid=2,0; source=3,0; non_word=0,0}. В результате извлечения журнала было получено 1602 записи (в виде пакетов слов). Они относились к 11 наборам похожих словарей, которые были сгенерированы программными модулями со следующими названиями: AT, RILJ, DHCP, MobileDataStateTracker, SocketSenderThread и DCT. AT отвечает за форматирование и разбор передаваемых/принимаемых сообщений GSM-модема, RILJ является сервисным компонентом Android, управляющим беспроводной связью, и т.д. Служба DHCP реализует обработку протокола DHCP. MobileDataStateTracker - это компонент системы Android, входящий в состав Network Manager, который управляет и настраивает интерфейсы сетевых пакетов ядра. Модуль ведения журнала SocketSenderThread является частью библиотек пользовательского пространства, отвечающих за сетевые сокеты. Дискретное косинусное преобразование (DCT) - это сервис, отвечающий за отслеживание пропускной способности соединения. Выявленные текущие изменения относятся к изменениям

состояния связи Монитор#1 в сетях LTE и Internet. Внедренная объектно-ориентированная модель TS и логарифмический корреляционный анализ позволили выявить некоторые недостатки в потреблении энергии и внести соответствующие улучшения в программное обеспечение, что привело к экономии электроэнергии примерно на 20% при типичных сценариях. Представленная модель TS с соответствующим графиком состояний и выборочными объектами предоставляет возможность анализировать общее поведение системы (отслеживание состояния), а также переключаться на детальный анализ, относящийся к представленным объектам или даже выборкам в выбранных диапазонах моделей TS (по времени или состояниям). Схемы корреляции журналов значительно сужают список интересных записей, более того, они могут быть адаптированы к требованиям TS или специфике системы.

3.6. Анализ результатов

Разработанная методология анализа временных рядов была вдохновлена реальной проблемой проектирования довольно сложных встраиваемых устройств (Монитор), а именно оптимизацией энергопотребления. Энергосбережение является важной проблемой в устройствах Интернета вещей.

В этом исследовании предлагается метод получения знаний о потреблении энергии, основанный на текущих измерениях. Внедренная многоуровневая объектно-ориентированная модель временных рядов, поддерживаемая соответствующими алгоритмами обработки данных, обеспечивает лучшее понимание энергетических зависимостей и характеристик работы устройства, включая вопросы реализации. Полученные функции облегчают идентификацию проблем проектирования/реализации и выполнение соответствующих улучшений.

Алгоритмы, описанные в разделе, позволили создать иерархическую

модель TS с четко выделяемыми функциями (объектами), которые объединяются в конечный автомат. Он сопоставляет поведение системы во времени с этими объектами, позволяет анализировать частоту их появления (экземпляры), корреляции с другими объектами (например, последовательности классов циклов), корреляции с журналами событий и т.д. Необработанные данные TS преобразуются в представление более высокого уровня, которое упрощает интерпретацию для эксперта.

Проанализированные TS аппаратов Монитора выглядели как мешанина различных импульсов, появляющихся спорадически на фоне некоторого низкоуровневого шума, соответствующего базовому состоянию. Это в некоторой степени может напоминать сигналы специфический поток данных, однако последний состоит из некоторых известных регулярных импульсов с возможными девиациями в случае стабильных отклонений. Такой простой регулярности не хватает в рассматриваемых TS, которые можно отнести к электронно-лучевым аппаратам. Тем не менее, в этой путанице импульсов можно обнаружить некоторую упорядоченность, приводящую к определенным состояниям TS или последовательностям классов циклов, которые могут быть дополнительно интерпретированы или проанализированы экспертом. Это агрегированное и разложенное представление TS упрощает поиск характерных признаков TS для проведения более глубокого анализа.

Модели TS были сжаты до 100 состояний и 12 классов циклов, поэтому интерпретацией работы устройства можно управлять, отслеживая компоненты модели в соответствии с зарегистрированными журналами или другими свойствами устройства. Более того, декомпозированная модель TS создает возможность применения алгоритмов обнаружения знаний на уровне производных объектов (а не исходных данных), что может повлиять на их эффективность.

Разработанная методология и инструментарий были успешно

использованы для обнаружения и устранения различных аномалий в разработанных и выпускаемых версиях тестов. Это привело к снижению энергопотребления.

Введенная декомпозиция временных рядов выполняется автоматически с помощью разработанного программного средства TS_AN, однако есть некоторые проблемы, которые необходимо задать с помощью предопределенных параметров в алгоритмах, некоторые из них являются явными (например, связаны с количеством выборок, размером сегментов), другие могут косвенно влиять на результат декомпозиции, например, они могут основываться на некоторых предварительных знаниях о работе системы (например, на уровне спецификации системы) или на выполнении калибровочных тестов. Однако параметры, проверенные в более старых версиях устройства, могут быть адаптированы к его последующим версиям.

Сложность создания сегмента линейно зависит от количества рассмотренных выборок TS. Другие алгоритмы более сложны в зависимости от количества сегментов (M) с верхней границей $O(M^2)$, на практике она ниже из-за низкой плотности графа TS. Таким образом, проведенный анализ реальных показателей TSs, относящихся к Монитор №1 и Монитор №2, занял 10 и 10-80 минут соответственно. Здесь стоит отметить, что оптимизация мощности более старой версии прибора проводилась вручную.

Результатом этого опыта стала концепция разработки представленной модели TS и аналитических алгоритмов. Список объектов, представляющих состояния и события, удобнее сопоставлять с объектами, извлеченными из классических журналов (сгенерированных с помощью хорошо известных инструментов: logcat и ftrace). Объединение значений потребляемого тока с производными объектами в TS позволяет лучше понять энергопотребление в каждом состоянии и ускоряет процессы

отладки или настройки для повышения производительности устройства.

Высокая точность выполняемого анализа обеспечивается за счет использования специального внешнего оборудования для сбора данных. Здесь рассматривается проблема объединения двух временных шкал наблюдений: оборудования для мониторинга и тестируемого устройства. Следовательно, корреляция журналов событий с производными функциями модели TS была относительно сложной из-за независимого временного масштаба мониторинга и контролируемых устройств, что довольно часто встречается на практике.

Предложенная схема сопоставления и объединения отчетов о событиях, генерируемых внутри контролируемого устройства, с данными о производительности, измеряемыми извне, также может быть применена к другим тестируемым системам. В случае устройств с механизмами совместной синхронизации этот процесс может быть упрощен. С другой стороны, в некоторых мобильных устройствах регистрация событий не предусмотрена из-за требований к миниатюризации, поэтому для интерпретации модели TS может потребоваться более активное участие эксперта. Так было в случае с Монитор №2. Несмотря на это неудобство, один из проектировщиков успешно выявил некоторые аномалии в поведении на основе нашей модели TS, которые не были выявлены во время стандартных процедур тестирования и сертификации.

При анализе TS учитываются три цели:

1) проверка проектных допущений и выявление возможных нарушений,

2) сравнение рабочего профиля с эталонным,

3) идентификация удаленных объектов (сегментов, экземпляров класса циклов или последовательностей) или неправильных временных соотношений. В проведенных экспериментах эти проблемы были связаны с особенностями энергопотребления.

Выявив избыточное энергопотребление (по сравнению с предположениями) для рассматриваемого сценария TS, тестировщик устройств может выполнить поиск объектов с более высоким потреблением (более высоким средним уровнем тока). Например, уменьшение параметра min_eps позволяет нам агрегировать объекты с более высоким сходством, что, как следствие, позволяет выявить больше различных классов объектов, которые могут быть кандидатами для более глубокого изучения. Таким образом, были обнаружены некоторые подозрительные последовательности циклов в приборах Монитора, которые не обеспечивали минимальный ток в неактивных состояниях (из-за некоторой ошибки программного обеспечения).

В случае разработки новой версии устройства сгенерированная модель TS может быть сравнена с производной моделью для предыдущей версии. Здесь анализ направлен на выявление отличающихся характеристик одних и тех же объектов (например, последовательности состояний, продолжительности) или их последовательностей, появления новых объектов и т.д.

На основе сгенерированных моделей TS можно получить профили соответствующих объектов, например, периодичность создания экземпляров класса циклов (чрезмерные задержки, перерывы), распределение последовательностей объектов. Эти функции могут быть использованы при наличии экспертных знаний о работе устройства и облегчать принятие решения о приемке или необходимость детального анализа (например, с использованием логарифмической корреляции).

Такой анализ на более высоком уровне иерархических объектов является более систематичным, чем работа с необработанными данными, эти объекты могут быть дополнительно обработаны с помощью алгоритмов интеллектуального анализа данных. Необходимы некоторые комментарии относительно возможных расширений и ограничений нашего

подхода. Они касаются трех аспектов: гибкости алгоритма, специфичности источника TS и целей анализа.

Разработанные алгоритмы зависят от предполагаемых параметров, которые должны быть адаптированы к рассматриваемой TS и исследуемой проблеме. Они могут быть получены тестировщиком/экспертом с учетом некоторой интуиции и общего представления о TS или найдены итеративным путем. Это не критично из-за относительно небольших временных затрат на автоматическую генерацию модели.

Представленная модель декомпозиции ориентирована на схемы временных рядов с преобладанием некоторой фоновой активности и конечным набором других, более интенсивных классов активности (классов циклов). Эти предположения выполняются во многих практических встроенных устройствах и устройствах Интернета вещей.

Представленный анализ может быть расширен для других свойств объекта, например, для проверки временных соотношений, периодичности. Возможно дальнейшее расширение для исследования корреляций/сходств нескольких сигналов (TSS), передаваемых одним и тем же устройством, на уровне соответствующих моделей (графиков и т.д.). Поскольку эксперименты проводятся с использованием некоторых тестовых сценариев, важной задачей является выбор репрезентативных сценариев для устройства или конкретных сценариев, связанных с некоторой неоднозначностью работы устройства или сомнениями пользователя/разработчика, нестандартными условиями окружающей среды и т.д., которые необходимо исследовать.

Предложенная методика была проверена при анализе энергопотребления некоторых аппаратов Монитора с использованием TS, представляющих текущее потребление устройства. Тем не менее, ее применимость может быть шире благодаря ее гибкости. Его можно использовать для аналогичного анализа других устройств, работа которых

согласуется с моделями систем SRT, IRT и CRT.

Анализ энергопотребления важен на быстрорастущем рынке различных устройств Интернета вещей. Представленная схема анализа TS может быть адаптирована для других целей анализа, таких как мониторинг показателей производительности (например, использование процессора, памяти, длины очереди обслуживания, нагрузки на передачу данных), анализ сигналов активации устройств. Буквенный регистр важен для узлов беспроводных сенсорных сетей (WSN).

В модель могут быть введены другие определения агрегации выборок, объектов, их свойств, показателей распознавания и иерархических структур, а также альтернативные параметры профилирования в алгоритмах.

Интерпретация поведения устройства в соответствии с полученной моделью TS может быть улучшена путем сопоставления журналов событий или приложений с выявленными характерными чертами или отклонениями от нормы. Условия подобия также могут быть заданы по-разному, например, количество похожих состояний в цикле, временные характеристики некоторых объектов (длительность, интервалы между объектами). Кроме того, в программу устройства могут быть вставлены соответствующие инструкции по регистрации, чтобы сфокусироваться на исследуемых проблемах. Еще одним важным моментом является выбор подходящих сигналов для наблюдения. Они могут подаваться непосредственно в систему мониторинга с наблюдаемого устройства или регистрироваться в ней и выводиться для анализа после выполнения задачи.

3.7. Выводы

Основным вкладом стала разработка оригинальной модели анализа временных рядов путем введения иерархических объектов с

предопределенными свойствами, относящимися к их структуре и статистическим характеристикам включенных выборок. Для создания этой модели требуется разработка эффективных алгоритмов. Этот новый взгляд на временные ряды облегчает оценку свойств поисковой системы/устройства, обнаружение аномалий, проверку действий по улучшению и т.д. Эти процессы могут быть автоматизированы. Эта концепция была реализована в разработанной аналитической системе.

Полезность представленной методики была подтверждена при анализе энергопотребления серийно выпускаемых аппаратов Монитора. Здесь стоит отметить, что возросшая сложность доступных микроконтроллеров и разработанного программного обеспечения делает анализ возможных проблем более сложным. Тем более что многие проблемы зависят от различных особенностей, которые трудно заметить без глубокого анализа, более того, довольно часто спецификации аппаратного или программного обеспечения не выполняются на практике и могут быть выявлены в ходе экспериментов. Из-за сложности и обилия собираемых данных требуется соответствующее моделирование и алгоритмическая поддержка.

Проблема энергопотребления приобретает все большее значение в связи с широким распространением различных устройств Интернета вещей и беспроводных сенсорных сетей. Разработанный инструмент планируется использовать в проектах, связанных с промышленными беспроводными сетями. Представленная методика, проверенная на временных рядах, описывающих ток, отводимый от аккумулятора, может быть использована для решения более широкого спектра задач. В частности, те, которые связаны с анализом производительности системы, например, использование ядра процессора, памяти, возможностей передачи данных, мониторинг физических параметров (температура, давление, влажность).

Разработанные алгоритмы обеспечивают определенную гибкость,

которая достигается за счет объектно-ориентированных псевдокодов и параметризованных критериев подобия, причем эти критерии могут быть сформулированы по-разному. Выбор параметров для конфигурации системы может осуществляться итеративным способом или быть результатом определенных знаний о работе системы. Представленное исследование может быть объединено с методами обнаружения знаний, имеющими дело с представленными модельными объектами.

4. УПРАВЛЕНИЕ ЭЛАСТИЧНОСТЬЮ ДЛЯ ПЛАНИРОВАНИЯ МОЩНОСТЕЙ В ОБЛАЧНЫХ СРЕДАХ ТИПА "ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КАК УСЛУГА", ОБСЛУЖИВАЮЩИХ СИСТЕМЫ МОНИТОРИНГА

Области применения облачных вычислений расширяются по мере того, как компании переходят от локальных ИТ-сред к общедоступным, частным или гибридным облакам. Следовательно, поставщики облачных услуг используют планирование производительности для поддержания производительности вычислительных ресурсов (экземпляров), необходимой для удовлетворения динамического характера спроса (запросов). Однако существует компромисс между развертыванием слишком большого количества дорогостоящих экземпляров и развертыванием слишком малого количества экземпляров и выплатой штрафных санкций за невозможность своевременной обработки запросов.

У экземпляра есть несколько измерений ресурсов, и выполнение запроса требует использования нескольких измерений емкости экземпляра. Это подробное многомерное управление ресурсами облачных вычислений известно как управление эластичностью и является важной проблемой, с которой сталкиваются все облачные провайдеры. Определение оптимального количества экземпляров, необходимого для данного горизонта планирования, является сложной задачей из-за комбинаторного характера задачи оптимизации. Разрабатывается оптимизационная модель и связанные с ней алгоритмы, чтобы найти компромисс между стоимостью ресурсов и штрафом за задержку выполнения в приложениях "программное обеспечение как услуга" с точки зрения поставщика облачных услуг. Разрабатывается точный подход для решения задач малого и среднего размера и эвристические методы для решения больших задач. Затем оценивается их производительность с помощью обширного

компьютерного анализа с использованием реальных данных и современных подходов облачных провайдеров. Также разрабатывается стохастическая структура и методология для решения проблемы неопределенности спроса в задачах мониторинга и, используя два разных случайно сгенерированных набора данных (представляющих примеры проблем на практике), демонстрируется, что могут быть получены надежные решения.

4.1. Постановка задачи оптимизации пропускной способности облака и оптимизационная модель

Раздел начинается с определения задачи планирования мощностей для области исследований в подразделе 4.1.1. Определив непрерывную общую задачу DQ_g, она преобразуется в дискретную оптимизационную задачу DQ_1 в подразделе 4.1.2.

Затем формулируется детальная оптимизационная задача DQ_1 в разделе 4.1.3, которая позволяет понять структуру и оптимально решить небольшие и средние версии задачи. В подразделе 4.1.4 представлены примеры для задачи DQ_1. В подразделе 4.1.5 разработана структурированная стратегия для решения задачи DQ_1 путем перебора в ограниченном пространстве решений, что повышает эффективность получения оптимального решения.

4.1.1. Постановка проблемы пропускной способности облака

Следуя [4.63], определим задачу планирования мощностей для задач мониторинга. Исходя из различных аппаратных и серверных ограничений, каждый экземпляр комбинированного программного и аппаратного обеспечения SaaS, управляемого поставщиком облачных услуг, имеет определенный объем ресурсов, который может быть выделен для выполнения клиентских запросов. Поскольку большинство облачных

провайдеров не хотят настраивать свои экземпляры на постоянной основе, они сосредотачиваются на настройке экземпляров в начале коротких периодов планирования. Поэтому предполагаем, что горизонт планирования составляет 1 час (его можно легко настроить) и что количество развернутых экземпляров остается неизменным на протяжении всего периода планирования.

Облачные провайдеры могут использовать различные горизонты планирования для учета любых закономерностей поступления циклических запросов, таких как время суток или день недели. Кроме того, горизонт планирования - это период времени, в течение которого начисляются штрафы за задержку запроса, поэтому выбирается горизонт планирования в качестве основного ориентира модели. Облачный провайдер может использовать результаты исследований для каждого горизонта планирования и комбинировать планы горизонта планирования, чтобы сформировать план пропускной способности на определенный день (или неделю) для каждого из своих клиентов. Таким образом, модель оптимизации для одного горизонта планирования превращается в план полной пропускной способности клиента для облачного провайдера.

Чтобы более полно понять нюансы этой проблемы облачных вычислений, ниже более подробно описано взаимодействие между спросом (запросами) и пропускной способностью (экземплярами), а также то, как применяются штрафные санкции за задержку. Спрос на облачного провайдера - это случайное поступление запросов, которые являются запросами на выполнение определенной задачи обработки данных мониторинга. Для выполнения каждого запроса требуется определенное количество ресурсов. Экземпляр может выполнять более одного запроса одновременно, но для параллельного выполнения запросов должно быть достаточно оперативной памяти, мощности процессора, емкости хранилища и других требований к ресурсам. Если все экземпляры для

конкретного клиента заняты, то запрос будет ждать, пока не будет развернут другой экземпляр или не завершатся другие запросы и не станут доступны ресурсы для текущего развернутого экземпляра.

Чтобы определить штрафные санкции за задержку для конкретного клиента, облачные провайдеры сравнивают среднее время ответа для конкретного типа запроса с пороговым значением штрафной задержки для конкретного типа запроса на каждом горизонте планирования. Если среднее время ответа на запрос определенного типа ниже порогового значения для данного типа запроса на горизонте планирования, штраф не может быть наложен. Если среднее время ответа для конкретного клиента превышает пороговое значение для данного типа запроса, за каждую секунду превышения порогового значения будет начисляться линейный штраф. Стоимость штрафных санкций для конкретного запроса и пороговое время их применения оговариваются в рамках соглашения об уровне обслуживания с каждым клиентом.

Существует множество других реальных примеров SaaS, которые должны обеспечивать автономное управление горизонтальной эластичностью.

В различных приложениях запросы могут поступать в любое время и в любой комбинации. Определим эту непрерывную общую задачу для заданного пути выборки как задачу DQ_g. В следующем разделе покажем, как эта непрерывная общая задача может быть преобразована в дискретную математическую модель для оптимизации.

4.1.2. Преобразование общей задачи DQ_g в оптимизационную модель DQ_1

Преобразуем общую задачу DQg с поступлением запросов через случайные промежутки времени в задачу DQ_1, предполагая, что все поступления запросов происходят в начале множества небольших

дискретных временных интервалов в пределах общего горизонта планирования. Чем меньше длительность временного интервала, τ, тем ближе решение DQ_l будет к решению DQ_g. Однако по мере уменьшения величины горизонта планирования решение задачи оптимизации становится все более трудоемким. Таким образом, оптимальная продолжительность временного интервала τ зависит от приложения и зависит от объема и времени выполнения запросов, частоты поступления запросов и продолжительности горизонта планирования. В приложении для обработки данных мониторинга τ установлено равным 1 секунде, поскольку выполнение всех запросов занимало более 1 секунды, а время поступления данных фиксировалось в секундах.

Общая задача DQ_g сформулирована с целым числом γ_t^j запросов типа j, поступающих в течение временного интервала t. Учитывая DQ_g, преобразуем ее в задачу DQ_l, перемещая все запросы, поступающие в течение каждого временного интервала t, в начало временного интервала t. Пусть оптимальным решением задачи DQ_g будет Π_g при значениях решения (X_i^g, Y_{it}^{gj}), а оптимальным решением задачи DQ_l будет Π при значениях решения $(X_i,\ Y_{it}^j)$ (обозначения приведены в таблице 4.2 [4.63]).

Таблица 4.2

Параметры и переменные оптимизационной модели

Параметры	
n	Количество временных интервалов на горизонте планирования.
τ	Продолжительность временного интервала в секундах.
k	Количество типов запросов.
λ_j	Количество целочисленных временных интервалов, необходимых для выполнения запроса типа j.
b_j	Стоимость штрафа за секунду выполнения запроса типа j, если он не завершает выполнение в течение заданного среднего порога времени штрафа μ_j, где j=1, 2, ..., k.
μ_j	Порог штрафной задержки, при котором может быть наложен штраф, если среднее время ожидания и выполнения запросов типа j превышает это значение. Измеряется в целом количестве временных интервалов.
c_l^j	Единицы объема оперативной памяти, необходимые для запроса типа j.

c_2^j	Единицы мощности процессора, необходимые для выполнения запроса типа j.
c_3^j	Единицы емкости дискового хранилища, необходимые для запроса типа j.
A	Затраты на единицу экземпляра для горизонта планирования.
C_1	Единицы объема оперативной памяти, доступные в каждом экземпляре.
C_2	Единицы мощности процессора, доступные в каждом экземпляре.
C_3	Единицы емкости дискового хранилища, доступные в каждом экземпляре.
γ_t^j	Целое число запросов типа j, поступающих в начале временного интервала t.
u_j	Общее количество запросов типа j, поступивших в течение периода планирования. $u_j = \sum_{t=1}^{n} \gamma_t^j + I_0^j$, где I_0^j - запросы, оставшиеся с предыдущего периода планирования.
m	Верхняя граница количества экземпляров, необходимых для горизонта планирования.
M	Большое количество для обеспечения надлежащей функциональности индикаторной переменной, V_j.
Переменные	
X_i	X_i=1, если экземпляр i развернут в течение периода планирования; в противном случае X_i=0, i=1, 2, ..., m.
Y_{it}^j	Количество запросов типа j, размещенных в экземпляре i за временной интервал t, j=1, 2, ..., k; i=1, 2, ..., m; t=1, 2, ..., n.
I_t^j	Количество запросов типа j, потребность в мощности которых не удовлетворена во временном интервале t и которые будут ожидать начала выполнения доступной мощности экземпляра, где I_0^j - запросы, оставшиеся с предыдущего горизонта планирования.
L_{it}^j	Общее суммарное количество запросов типа j, выполняемых в экземпляре i за временной интервал t.
P_j	Среднее время ожидания для каждого запроса типа j в течение периода планирования.
Q_j	Штрафные санкции за задержку для каждого запроса типа j на горизонте планирования.
V_j	Индикаторная переменная, которая принимает значение 1, когда запросы типа j превышают свой штрафной порог; в противном случае - 0.
П	Минимизированная общая стоимость - общая стоимость штрафных санкций за задержку + общая стоимость развертывания экземпляров.

Лемма 4.1 [4.63]**.** Решение задачи DQ_l является нижней границей общей задачи DQ_g. То есть $П_g \geq П$.

Доказательство. Запросы, поступающие в каждый временной интервал, одинаковы по количеству и типу как для задач D_q, так и для

задач DQ_1. Единственное различие заключается в том, что предполагается, что запросы, поступающие в течение любого временного интервала t в DQg, поступают в начале временного интервала t в DQ_1. Кроме того, значения решения (X_i^g, Y_{it}^{gj}) применимы к задаче DQ_1 со значением целевой функции G, поскольку запросы поступают в DQ_g позже, чем они поступили в DQ1. Таким образом, оптимальное решение П для DQ_1 не может быть хуже допустимого решения (X_i^g, Y_{it}^{gj}) задачи DQ_g, то есть $П_g \geq П$.

Доказательство завершено.

Поскольку важна минимизация затрат в задаче DQ_g, необходимо убедиться, что задача оптимизации DQ_1 дает нижнюю границу для задачи DQ_g (лемма 4.1).

4.1.3. Формулировка оптимизационной модели

Используем смешанную постановку для модели оптимизации [4.63], задачи DQ_1, поскольку количество экземпляров может быть только целым, но штрафные условия непрерывны.

Задача DQ_1 сводит к минимуму затраты ресурсов на развертывание экземпляра и любые штрафные санкции за задержку выполнения запроса, как указано в каждом соглашении об уровне обслуживания между поставщиком облачных услуг и клиентом. Эта модель ориентирована на количество развертываемых клиентских экземпляров, а не на общий оптимальный размер фермы серверов или количество ферм серверов, необходимых в различных регионах. Сначала представим детерминированную модель. Это позволяет понять структуру проблемы и разработать структурные свойства для эффективного решения задач малого и среднего размера. Эта модель также обеспечивает основу для сравнения, чтобы оценить производительность наших OFD. В подразделе

4.2 на основе детерминистической модели разработана стохастическая модель для решения проблемы неопределенности при поступлении запросов. В [4.30] высказано предположение, что эти типы моделей программирования являются хорошими подходами для определения требований к производительности в приложениях облачных вычислений.

Наиболее важным параметром является параметр поступления запроса (требования), γ_t^j, который представляет собой целое число запросов типа j, поступающих в начале временного интервала t. В этой матрице будет одна строка для каждого временного интервала t и один столбец для каждого типа запроса j. Если за определенный промежуток времени не поступит ни одного запроса, вся строка γ_t^j будет заполнена нулями. Однако за определенный промежуток времени может поступить несколько запросов, и в каждом столбце γ_t^j могут быть ненулевые целые значения. Параметр γ_t^j определяется на основе исторических данных или создается случайным образом на основе известных распределений частоты поступления запросов каждого типа. Таким образом, каждый горизонт планирования имеет уникальное значение γ_t^j для каждого случайно сгенерированного случая (известного как путь выборки).

При согласовании соглашения об уровне обслуживания будет задокументировано количество типов запросов (k), а также все параметры, связанные с запросами. Это включает время, необходимое для выполнения каждого запроса (λ_j), и пороговое время задержки каждого запроса (μ_j). Оба эти значения выражены в виде целого числа временных интервалов, необходимых для достижения их значений. Поскольку они являются целыми значениями, длительность временного интервала, τ, определяет минимальное время выполнения и время штрафной задержки в рамках модели. Пороговое время штрафа за задержку, μ_j, является ключевым моментом при заключении соглашения об уровне обслуживания,

поскольку величина, на которую оно превышает время выполнения, λ_j, определяет среднее время ожидания, приемлемое для запроса типа j, прежде чем будет начислен штраф.

Время выполнения каждого запроса, λ_j, зависит от многих факторов, в том числе от используемой вычислительной мощности (CPU). Это одна из причин, по которой параметры требований к производительности для выполнения j-го запроса каждого типа также будут указаны в соглашении об уровне обслуживания. В модели используются три параметра: оперативная память c_1^j (RAM), центральный процессор c_2^j (CPU) и дисковое хранилище c_3^j (disk storage), но можно легко добавить дополнительные параметры. Последний параметр запроса, оговоренный в соглашении об уровне обслуживания, - это стоимость штрафа за секунду среднего времени ожидания, b_j, которая превышает порог штрафной задержки. Облачный провайдер будет согласовывать b_j на основе порога штрафа μ_j, а также цены, которую клиент платит за облачные услуги.

Остальные параметры в модели зависят от поставщика облачных услуг и типа серверов, которые они используют для размещения клиентских экземпляров. Емкость, доступная во всех трех измерениях ресурсов (C_1, C_2, C_3), а также стоимость каждого экземпляра за горизонт планирования (α) являются факторами структуры затрат и настройки оборудования серверных ферм облачного провайдера. Стоимость экземпляра, a, включает в себя энергозатраты, техническое обслуживание, накладные расходы и капитальные затраты, которые распределяются в зависимости от использования.

Приведем формулировку задачи минимизации для DQ_1. Основной переменной для принятия решения является количество развернутых экземпляров $\sum_{i=1}^{m} X_i$, где X_i=1, если экземпляр i развернут в течение периода

планирования; в противном случае X_i=0. Ограничиваем возможное количество экземпляров верхней границей m, рассчитанной в разделе 4.1.4. Переменная Y_{it}^j - это количество запросов типа j, размещенных в экземпляре i за временной интервал t. Она определяет, когда конкретный запрос начнет выполняться.

Переменная I_t^j представляет собой количество запросов типа j, требования к емкости которых не удовлетворены в течение временного интервала t. I_t^j может интерпретироваться как очередь, в которой запросы типа j ожидают, когда доступная емкость экземпляра начнет выполняться. Эти переменные и все параметры модели описаны в таблице 4.2.

Задача DQ_1 [4.63]**:**

$$\text{Min } \Pi\left(X_i, Y_{it}^j\right) = \sum_{j=1}^{k} Q_j + \alpha \sum_{i=1}^{m} X_i \tag{4.0}$$

с ограничениями:

$$I_t^j = I_{t-1}^j + \gamma_t^j - \sum_{i=1}^{m} Y_{it}^j,\ j = 1,2,...,k;\ t = 1,2,...,n \tag{4.1}$$

$$Y_{it}^j \le u_j X_i,\ j = 1,2,...,m;\ j = 1,2,...,k;\ t = 1,2,...,n \tag{4.2}$$

$$L_{it}^j \ge \sum_{h=0}^{\lambda_j - 1} Y_{i,t-h}^j,\ j = 1,2,...,k;\ i = 1,2,...m;\ t = 1,2,...,n \tag{4.3}$$

$$\sum_{j=1}^{k} c_r^j L_{it}^j \le C_r X_i,\ r = 1,2,3;\ i = 1,2,...m;\ t = 1,2,...,n \tag{4.4}$$

$$P_j \ge \frac{\sum_{t=0}^{n} \tau I_t^j}{u_j},\ j = 1,2,...,k \tag{4.5}$$

$$P_j + \tau\lambda_j \le \tau\mu_j + V_j M,\ j = 1,2,...,k \tag{4.6}$$

$$Q_j \ge b_j u_j (P_j + \tau\lambda_j - \tau\mu_j) - (1 - V_j)M,\ j = 1,2,...,k \tag{4.7}$$

$$Y_{it}^j, I_t^j, L_{it}^j > 0,\ i = 1,2,...,m;\ j = 1,2,...,k;\ t = 1,2,...n \tag{4.8}$$

$$P_j, Q_j \geq 0,\ j = 1,2,...,k \tag{4.9}$$

$$X_i, V_j \in \{0,1\},\ i = 1,2,...,m;\ j = 1,2,...,k \tag{4.10}$$

Целевая функция минимизирует затраты на задержку выполнения всех запросов типа j, а также затраты на ресурсы развернутых экземпляров. Стоимость ресурсов развернутого экземпляра - это стоимость на горизонте планирования экземпляра, α, умноженная на количество развернутых экземпляров $\sum_{i=1}^{m} X_i$. Нелинейная природа штрафных издержек, Q_j, является более сложной и требует определения нескольких ограничений.

Набор ограничений (4.1) отслеживает количество запросов типа j, требования к производительности которых не выполнены в течение временного интервала t, и добавляет их в I_t^j. I_0^j используется для переноса любых невыполненных запросов с предыдущего горизонта планирования на текущий горизонт планирования. Набор ограничений (4.2) присваивает Y_{it}^j запросов типа j экземпляру i во временном интервале t, если экземпляр i развернут. Набор ограничений (4.3) подсчитывает общее количество запросов типа j, назначенных экземпляру i во временном интервале t. Набор ограничений (4.4) гарантирует, что запросы назначаются каждому экземпляру в определенный промежуток времени таким образом, чтобы не нарушались ограничения пропускной способности экземпляра.

Набор ограничений (4.5) вычисляет среднее время ожидания, P_j, для каждого типа j запроса в течение горизонта планирования. Набор ограничений (4.6) определяет, превышает ли среднее время ожидания, P_j, плюс время выполнения, $\lambda_j\tau$, порог штрафной задержки, $\mu_j\tau$. Если порог штрафа превышен, то индикаторная переменная V_j=1, что означает, что по запросам типа j будет начисляться штраф, пропорциональный величине, на которую среднее время ожидания плюс время выполнения превышает порог задержки штрафа типа j.

Если среднее время ожидания плюс время выполнения меньше порогового значения штрафа, то $V_j=0$ и штраф за задержку не начисляется для запросов типа j, даже если произошло некоторое ожидание. Набор ограничений (4.7) вычисляет фактическую стоимость штрафа Q_j для каждого типа запроса j, когда индикаторная переменная $V_j=1$, и штраф для запроса типа j разрешен.

Вычислительная сложность задачи DQ_1

Задача DQ_1 отличается от традиционных задач с упаковкой в ячейки, поскольку запросы поступают динамически в разные периоды и должны быть упакованы в экземпляры (ячейки) определенной емкости. Кроме того, запрос покидает ячейку после выполнения, что создает пространство в ячейке для других запросов, ожидающих упаковки. Кроме того, целевые функции этих двух задач различны, и за превышение ожидания запроса в DQ_1 взимается штрафная плата за задержку. Таким образом, формально исследуется вычислительная сложность задачи DQ_1, поскольку она имеет несколько особенностей, отличных от задачи об упаковке контейнеров.

Теорема 4.1 [4.63]**.** Задача DQ_1 является сильно NP-сложной.

Доказательство.

Для упрощения выбрана задача о 3-х разбиениях из [4.62].

Пример: Даны целые положительные числа s, B и набор целых чисел $A=\{a_1, a_2, ..., a_s\}$ с и $B/4<a_j<B/2$ для $1\le j\le 3s$. Существует ли разбиение A на три набора элементов $\{A_1, A_2, \ldots, A_s\}$ такое, что $\sum_{a_j\in A_i} a_j = B$, $1\le i\le s$?

Решение: Найти разбиение A на три набора элементов $\{A_1, A_2, \ldots, A_s\}$ такое, что $\sum_{a_j\in A_i} a_j = B$, $1\le i\le s$. Учитывая произвольный случай 3-разбиения, построим следующий пример задачи DQ_1.

• Горизонт планирования состоит из n=s периодов. Количество типов запросов k=3s+1. Количество запросов типа j, j=1, 2, ..., 3, поступивших в начале периода 1, равно 1, а количество запросов типа j, j=1, 2, ..., 3, поступивших во все остальные периоды, равно 0. Количество запросов типа j =3s+1, поступивших в начале каждого периода t, равно 1, t=1, 2, ..., s. Таким образом, γ_1^j=1, j=1, 2, ..., 3s; γ_t^j=0, j=1, 2, ..., s, t=2, 3, ..., s; и $\gamma_t^{3s+1} = 1$, t=1, 2, ..., s.

• Из приведенных выше данных о поступлении запроса получаем u_j =1, j=1, 2, ..., 3s; и u_{3s+1}=s.

• Емкость экземпляра: C_1 =2B, C_2 =0 и C_3 =0.

• Стоимость штрафа за секунду выполнения запроса типа j, b_j=B, j=1, 2, …, 3s, 3s + 1; и стоимость за единицу экземпляра, α=B.

• Количество целочисленных временных интервалов, необходимых для выполнения запроса типа j, λ_j =1, j=1, 2, …, 3s, 3s + 1. Зададим длину временного интервала τ=1.

• Пороговое значение штрафного времени, при котором может быть наложен штраф (в количестве временных интервалов) $\mu_1=\mu_2 =\ldots\ \mu_{3s}$ =s, j=1, 2, …, 3s, j=1, 2, ..., 3s и μ_{3s+1} =1.

• Пропускная способность запроса типа j=1, 2, ..., 3c: $c_1^j = \alpha_j$, $c_2^j = 0$, $c_3^j = 0$, j=1, 2, ..., 3c.

• Пропускная способность запроса типа j=1, 2, …, 3s: $s_1^j = \alpha_j$, $s_2^j = 0$, $s_3^j = 0$, j=1, 2, …, 3s.

• Общая стоимость П и пороговое значение, D=B.

Для примера задачи DQ_1, построенной выше, рассмотрим следующий вопрос: существует ли выполнимое назначение Y_{it}^j запросов $\sum_{i=1}^{m} X_i$ экземплярам с общей стоимостью $\Pi(X_i, Y_{it}^j) \le D=B$?

Таблица 4.3

γ_t^j, количество запросов типа j, поступивших в начале временного интервала t в задаче принятия решения

Запрос	j=1	j=2	j=3	j=4	…	j=3s	j=3s+1
γ_1^j	1	1	1	1	…	1	1
γ_2^j	0	0	0	0	…	0	1
γ_3^j	0	0	0	0	…	0	1
…	0	0	0	0	…	0	1
…	0	0	0	0	…	0	1
…	0	0	0	0	…	0	1
γ_{3s-1}^j	0	0	0	0	…	0	1
γ_{3s}^j	0	0	0	0	…	0	1

Задача принятия решения явно относится к классу NP-сложных. Кроме того, легко проверить, что построение задачи принятия решения может быть выполнено за полиномиальное время. Теперь покажем, что существует такое назначение, что $\Pi(X_i, Y_{it}^j) \le D=B$ тогда и только тогда, когда существует решение задачи о трех разделах.

Предположим, что существует разбиение A на три набора элементов $\{A_1, A_2, \ldots, A_s\}$, такое, что $\sum_{a_j \in A_i} a_j = B$, $1 \le i \le s$. Предположим, что $A_1=\{a_1, a_2, a_3\}$, $A_2=\{a_4, a_5, a_6\}$, …, $A_i=\{a_{3i-2}, a_{3i-1}, a_{3i}\}$, …, $A_s=\{a_{3s-2}, a_{3s-1}, a_{3s}\}$. Заметим, что $a_{3i-2}+a_{3i-1}+a_{3i}=B$, i=1, 2, …, s. Предполагаем задание только с одним экземпляром, $\sum_{i=1}^{m} X_i$, как показано ниже. Заметим, что $X_1=1$ и $X_i=0$, i=2, 3, ..., m.

Предлагается следующее определение Y_{1t}^j:

• Период 1: $Y_{11}^1=1$, $Y_{11}^2=1$, $Y_{11}^3=1$ и $Y_{11}^{3s+1}=1$; другие $Y_{11}^j=0$.

• Период 2: $Y_{12}^4=1$, $Y_{12}^5=1$, $Y_{12}^6=1$ и $Y_{12}^{3s+1}=1$; другие $Y_{12}^j=0$.

• Период i: $Y_{1i}^{3i-2}=1$, $Y_{1i}^{3i-1}=1$, $Y_{1i}^{3i}=1$ и $Y_{1i}^{3s+1}=1$; другие $Y_{1i}^j=0$, i =3, 4, …, s-1.

• Период s: $Y_{1s}^{3s-2}=1$, $Y_{1s}^{3s-1}=1$, $Y_{1s}^{3s}=1$ и $Y_{1s}^{3s+1}=1$; другие $Y_{1s}^{j}=0$.

Поскольку $a_{3j-2}+a_{3j-1}+a_{3j}=B$, j=1,2,…,s, емкость экземпляра, соответствующего назначению, может быть полностью использована. Заметим, что в приведенном выше задании имеем $P_j \leq s-1$, j=1,2,…,s, и $P_{3s+1}=0$.

Таким образом, стоимость штрафа для запроса типа j на горизонте планирования Q_j =0, j=1,2, …,3s, 3s+1 и $\Pi(X_i, Y_{it}^{j})=\alpha=B$. Коэффициент использования ресурсов экземпляра, соответствующего назначению Y_{1t}^{j}, можно найти в таблице 4.4.

Таблица 4.4

Использование емкости экземпляров, соответствующее 3-D разбиению

Использование емкости экземпляра, $C_1=2B$	Тип запроса 3i-2	Тип запроса 3i-1	Тип запроса 3i	Тип запроса 3s+1	Общая использованная емкость
Период i=1	a_1 (1 экз.)	a_2 (1 экз.)	a_3 (1 экз.)	B (1 экз.)	2B
Период i=2	a_4 (1 экз.)	a_5 (1 экз.)	a_6 (1 экз.)	B (1 экз.)	2B
…	…	…	…	…	…
…	…	…	…	…	…
Период i	a_{3i-2} (1 экз.)	a_{3i-1} (1 экз.)	a_{3i} (1 экз.)	B (1 экз.)	2B
…	…	…	…	…	…
…	…	…	…	…	…
Период i=s	a_{3s-2} (1 экз.)	a_{3s-1} (1 экз.)	a_{3s} (1 экз.)	B (1 экз.)	2B

Предположим, что существует назначение (X_i, Y_{it}^{j}) для решения задачи $\Pi(X_i, Y_{it}^{j}) \leq D=B$.

Теперь покажем, что существует решение задачи о 3D-разбиении.

• Поскольку $\Pi(X_i, Y_{it}^{j}) \leq B$, у нас не может быть более одного экземпляра, поскольку a=B. Таким образом, $\Pi(X_i, Y_{it}^{j})= B$. Это означает, что $\sum_{i=1}^{m} X_i = 1$ и стоимость штрафа Q_j=0, j=1,2, …,3s,3s+1. Поскольку у нас

есть ровно один экземпляр такой, что $X_1=1$ и $X_i=0$, i=2,3,…,m. Заметим, что общая емкость, доступная для этого экземпляра в течение горизонта планирования в пересчете на C_1, равна 2sB.

• Поскольку $\sum_{j=1}^{3s} a_j = sB$, общая потребность в ресурсе для запросов типа j, j=1,2,...,3s на горизонте планирования равна sB. Поскольку $\gamma_i^{3s+1}=1$, i=1,2,...,s, общая потребность в мощности для запросов типа j=3s+1 на горизонте планирования равна sB. Это означает, что существует назначение всех запросов одному экземпляру без каких-либо простаивающих мощностей.

• Поскольку $Q_{3s+1}=0$, тип запроса 3s+1 должен присваиваться экземпляру немедленно в каждый период i=1,2,...,s. Таким образом, запрос типа 3s+1 потребляет емкость B в каждый период i, i=1,2, ...,s. Оставшаяся емкость B в каждый период должна быть полностью израсходована запросом типа j=1,2,...,3s, поступившим в начале периода 1.

• Поскольку оставшаяся емкость экземпляра B (в каждом периоде) должна быть использована запросом типа j, имеющим требования к емкости, aj, j=1,2,...,3s и $\sum_{j=1}^{3s} a_j = sB$, эти запросы должны быть полностью упакованы в s периодов, каждый из которых имеет оставшуюся емкость B в каждом периоде. Поскольку $a_i<B/2$, два запроса, упакованные в период, создадут незанятую емкость. Поскольку $a_i>B/4$, четыре запроса не могут не поместиться в оставшуюся емкость B за период. Таким образом, ровно три запроса упаковываются в период, полностью используя емкость B. Это означает, что существует решение проблемы с 3-мя разделами. Можно легко проверить, что $Q_j=0$, j= 1, 2, ..., 3s, 3s+1 в задаче.

Теорема доказана.

4.1.4. Вычисление верхней границы

Чтобы определить верхнюю границу для задачи DQ_1, вычисляется количество развернутых экземпляров, которые приводят к нулевым задержкам запросов [4.63]. Увеличение емкости экземпляра сверх количества экземпляров с нулевым ожиданием увеличило бы затраты ресурсов и не могло бы еще больше снизить штрафные санкции за задержку, поскольку они уже равны нулю. Следовательно, число экземпляров с нулевым ожиданием, m, определенное по алгоритму 5, становится начальной верхней границей для задачи оптимизации DQ_1. Вычисляем значение нулевого ожидания на основе алгоритмов уменьшения первого соответствия, первоначально разработанных для одномерной упаковки в ячейки [4.7, 4.33]. Более поздние исследования упаковки в векторные контейнеры предполагают, что все размеры независимы друг от друга и геометрическая интерпретация предметов отсутствует. Поскольку векторная ячейковая упаковка является NP-сложной [4.32], большая часть исследований по этой проблеме сосредоточена на сравнении эффективности различных методов аппроксимации [4.8, 4.49, 4.57].

Начиная с верхней границы нулевого ожидания, алгоритм 1 с нулевым штрафом повторно анализирует уменьшающееся количество развернутых экземпляров, пока не случится штраф. Алгоритм с нулевым штрафом уменьшает фиксированное количество экземпляров H на значение калибровочного параметра ZeroPenJump. При первом появлении штрафа Алгоритм 1 переходит ко второму циклу repeat/until, увеличивая H на один фиксированный момент времени, пока штраф не вернется к нулю. Значение H на данный момент является минимальным числом случаев, необходимых для достижения нулевого штрафа, и становится верхней границей, m. Алгоритм 1 с нулевым штрафом использует ожидающую очередь неисполненных запросов, поэтому необходимо использовать

критерий сортировки, чтобы определить, какие запросы должны искать доступную емкость в первую очередь.

Алгоритм 1 [4.63]**. Определение минимальной верхней границы нулевого штрафа**

```
1 procedure Zero-Penalty (m, n, k, γ_t^j)
2       H=m
3       repeat
4               Call Assign-Queries-In-Queue
5               H=⌊H*(1−ZeroPenJump)⌋,
                где ZeroPenJump - процентное уменьшение
6       until Penalty > 0
7       repeat
8               H=H+1
9:              Call Assign-Queries-In-Queue в Алгоритме 2
10      until Penalty=0
11      m=H,
        минимальное количество случаев, необходимое для достижения
        нулевых штрафных санкций.
12      return m
13 end procedure
```

Для решения этой задачи Алгоритм 1 с нулевым штрафом вызывает Алгоритм 2 назначения запросов в очереди, который также используется в OFD. Алгоритм 2 помещает все поступающие запросы в очередь ожидания, а затем сортирует ожидающие запросы по SPT, как предложено в литературе по DDM. Можно использовать и другие критерии сортировки, но мы считаем, что SPT также лучше всего работает в нашем приложении для обработки данных. Если в экземпляре есть доступная емкость, то назначаются запросы, а если нет, то запросы остаются в очереди с соответствующей задержкой и рассчитанными штрафными издержками.

Значение, полученное с помощью алгоритма 1, является верхней границей проблемного сценария, поскольку увеличение числа экземпляров выше этого значения m приведет к дополнительным затратам ресурсов экземпляра, но штрафные санкции не могут быть снижены ниже нуля.

Следовательно, оптимальное решение никогда не будет превышать эту верхнюю границу нулевого штрафа.

Алгоритм 2. Назначение запросов из очереди для начала выполнения

```
1 procedure Assign-Queries-In-Queue(H, n, k, γ_t^j)
2     for t=1 to n do
3         Добавляем все вновь поступающие запросы в очередь
          ожидания.
4         Сортируем очередь запросов, сначала выполняя запросы с
          наименьшим временем обработки.
5         for j=1 to k do
6             Если запрос типа j ожидает, пробуем поместить его в
              один из его экземпляров. Если в экземпляре найдено
              место во всех трех измерениях емкости, добавим
              требования к емкости этого запроса для этого экземпляра
              на некоторое количество временных интервалов в
              будущем, равное времени его выполнения.
7         end for
8         Запишем все запросы, которые не могут быть назначены
          экземпляру, в очередь ожидания.
9     end for
10    Рассчитаем потенциальные штрафные санкции Q_j, используя наборы
      ограничений (4.5)-(4.7).
11    return Q_j
12 end procedure
```

4.1.5. Структурированная стратегия для решения задачи DQ_1

При установлении верхней границы m задача DQ_1 может быть решена оптимально. Однако, за исключением небольших практических задач, пространство решений по-прежнему настолько велико, что обычно на крупных серверах или виртуальных машинах, используемых для анализа пропускной способности облачных провайдеров, быстро заканчивается объем памяти. Поэтому разрабатывается структурированная стратегия, которая решает проблему с фиксированным экземпляром (которая аналогична первоначальной задаче, за исключением того, что количество экземпляров фиксировано и задано, как описано более подробно ниже), начиная с верхней границы m количества экземпляров и уменьшая количество фиксированных экземпляров r, пока оптимальное

решение не начнет увеличиваться. Поскольку оптимальное решение модели фиксированной емкости с r экземплярами не приводит к увеличению числа фиксированных экземпляров (см. лемму 4.2), для достижения решения обычно не требуется рассматривать все возможности фиксированных экземпляров.

Чтобы упростить этот структурированный стратегический подход, определим две отдельные задачи $(DQ_l^r, \overline{D}\overline{Q}_l^r)$, аналогичные задаче DQ_l, с дополнительными ограничениями на общее количество развернутых экземпляров $\sum_{i=1}^{m} X_i$. В $\overline{D}\overline{Q}_l^r$ число экземпляров фиксировано на уровне r, тогда как в DQ_l^r число экземпляров меньше или равно r. Для простоты обозначим оптимальное значение целевой функции $\Pi^r(X_i, Y_{it}^j)$ задачи DQ_l^r как Π_r, а соответствующее значение $\sum_{j=1}^{k} Q_j$ (соответственно, $\sum_{i=1}^{r} X_i$) - как S_q^r (соответственно, Z_x^r). Заметим, что $Z_x^r \le r$, $S_q^m = 0$ и $S_q^{m+1} = 0$. Кроме того, мы обозначаем оптимальное значение целевой функции $\sum_{j=1}^{k} Q_j$ задачи $\overline{D}\overline{Q}_l^r$ как $\overline{S}_q^r$. Поскольку $S_q^m = 0$ и $S_q^{m+1} = 0$, имеем $\overline{S}_q^m = 0$ и $\overline{S}_q^{m+1} = 0$. В структурированной стратегии решаем задачу $\overline{D}\overline{Q}_l^r$ с фиксированным числом экземпляров r, начиная со среднего значения, r=(1+m)/2. Затем итеративно увеличиваем или уменьшаем значение r, чтобы достичь улучшенного верхнего граничного значения, m*, как определено в лемме 4.3 ниже), а затем находим наименьшее количество экземпляров, которые приводят к минимальной общей стоимости, m_0, как определено в лемме 4.5 ниже.

Конкретные шаги этой структурированной стратегии описаны в алгоритме 3. Итеративно решаем задачу поиска m*, используя два цикла

while в строках 3-11 алгоритма 3. Затем итеративно решаем задачу поиска m_0, используя третий цикл while в строках 12-18.

Таблица 4.5

Сравнение оптимального для полной модели (DQ_1), оптимального для структурированной стратегии (алгоритм 3) и автономного динамического алгоритма 4 времени решения для выбранных горизонтов планирования

Горизонт планирования	120	166	168	67	119	155	156	163	133	136	160
Задача DQ_1	0.3	0.4	0.3	2.4	5.4	42.0	252.1	661.9	35.4	>24 ч	704.6
Оптимальный алгоритм 3	0.1	0.2	0.2	0.5	0.6	0.5	4.8	31.1	20.5	22.1	23.2
Динамический алгоритм 4	<1	<1	<1	<1	<1	<1	<1	<1	<1	<1	<1

Примечание: все значения указаны в минутах, за исключением периода >24 часов, когда модель не может быть решена по истечении 24 часов

Горизонт планирования	127	121	167	122	96	162	26
Задача DQ_1	>24 ч	>24 ч	>24 ч	>24 ч	>24hr	>24 ч	>24 ч
Оптимальный алгоритм 3	97.4	31.6	71.3	48.8	75.5	63.8	121.6
Динамический алгоритм 4	<1	<1	<1	<1	<1	<1	<1
Горизонт планирования	48	164	146	151	25	131	132
Задача DQ_1	>24 ч	>24 ч	>24 ч	>24 ч	>24hr	>24 ч	>24 ч
Оптимальный алгоритм 3	134.8	208.5	231.9	1376.2	118.8	103.5	176.6
Динамический алгоритм 4	<1	<1	<1	<1	<1	<1	<1

Таким образом, алгоритм 3 с фиксированными ограничениями на экземпляры позволяет решать более масштабные задачи без нарушения ограничений вычислительной памяти. Сокращение времени вычислений показано в таблице 4.5. Это иллюстрирует, что все больше и больше проблем решаются за разумное время с помощью структурированного подхода в алгоритме 3, в отличие от непосредственного решения исходной модельной задачи DQ_1.

Задача DQ_1^r, r=1, 2, …,m+1:

$$\min \Pi^r\left(X_i, Y_{it}^j\right) = \sum_{j=1}^{k} Q_j + \alpha \sum_{i=1}^{r} X_i$$

Ограничения: (4.1)-(4.10)

$$\sum_{i=1}^{r} X_i \le r \qquad (4.11)$$

Алгоритм 3. Структурированная стратегия решения задачи DQ_1

1 **procedure Structured-Strategy** $(\bar{D}\bar{Q}_1^r, \bar{D}\bar{Q}_1^{r+1})$
2 $r = \left\lfloor \frac{1+m}{2} \right\rfloor$
3 Решаем задачи $(\bar{D}\bar{Q}_1^r, \bar{D}\bar{Q}_1^{r+1})$, находя решения $\bar{S}_q^r$, $\bar{S}_q^{r+1}$ соответственно
4 **while** $\bar{S}_q^r > \bar{S}_q^{r+1}$ **do**
5 r=r+1
6 Решаем задачу $\bar{D}\bar{Q}_1^{r+1}$ и находим решение $\bar{S}_q^{r+1}$
7 **end while**
8 **while** $\bar{S}_q^r = \bar{S}_q^{r+1}$ **do**
9 r=r-1
10 Решаем задачу $\bar{D}\bar{Q}_1^r$ и находим решение $\bar{S}_q^r$
11 **end while**
12 $m^* = r$
13 $m^0 = m^*$
14 Решаем задачи $DQ_1^{m_0-1}$, $DQ_1^{m_0}$ и находим решения Π^{m_0-1}, Π^{m_0} соответственно
15 **while** $\Pi^{m_0-1} = \Pi^{m_0}$ do
16: $m_0 = m_0 - 1$
17: Решаем задачу $DQ_1^{m_0-1}$ и находим решение Π^{m_0-1}
18: **end while**
19: **return** m_0, Π^{m_0}
20: **end procedure**

Примечание: При реализации ограничение (4.11) (и приведенное ниже ограничение (4.12)) можно было бы удалить и заменить индексированными значениями от 1 до r. Однако эти ограничения включены, чтобы сделать различия в описанных задачах более очевидными.

Лемма 4.2. Оптимальное значение целевой функции Π^r задачи DQ_1^r не увеличивается с увеличением r при $r \le m$. Отсюда $\Pi^1 \ge \Pi^2 \ge \ldots \Pi^m \ge \Pi^{m+1}$, r=1, 2, …, m+1.

Доказательство. Предположим, что условие нарушено при r =1. То есть $\Pi^r < \Pi^{r+1}$. Оптимальным решением задачи DQ_1^r является выполнимое решение задачи DQ_1^{r+1}. Таким образом, мы имеем $\Pi^r \ge \Pi^{r+1}$, что

противоречит нашему предположению, что $\Pi^r<\Pi^{r+1}$. По индукции условие $\Pi^1\geq\Pi^2\geq\ldots\Pi^m\geq\Pi^{m+1}$, r=1, 2, …, m+1, истинно.

Доказательство завершено.

Рассмотрим теперь задачу $\overline{D}\overline{Q}_l^r$, r=1, 2, …, m+1:

$$\min \sum_{j=1}^{k} Q_j$$

Ограничения: (4.1)-(4.10)

$$\sum_{i=1}^{r} X_i = r \qquad (4.12)$$

Лемма 4.3. В оптимальных решениях задачи $\overline{D}\overline{Q}_l^r$, r=1, 2, ..., m+1, существует индекс m^* такой, что $\overline{S}_q^1 > \overline{S}_q^2 > \ldots > \overline{S}_q^{m^*}$ и $\overline{S}_q^{m^*} = \overline{S}_q^{m^*+1} = \ldots = \overline{S}_q^m = \overline{S}_q^{m+1}$, где $1\leq m^*\leq m$.

Доказательство.

Первоначальное увеличение числа экземпляров может привести к уменьшению значения $\sum_{j=1}^{k} Q_j$. Однако это значение штрафа остается неизменным после достижения определенного порогового значения m^*. Предположим, что условие $\overline{S}_q^m = \overline{S}_q^{m^*}$ нарушено при $r=m^*$. То есть $\overline{S}_q^r < \overline{S}_q^{r+1}$ и значение $\overline{S}_q^r$ не может быть больше $\overline{S}_q^{r+1}$ по определению. Оптимальное решение задачи с m^* экземплярами является выполнимым решением задачи $\overline{D}\overline{Q}_l^{r+1}$. Таким образом, имеем $\overline{S}_q^r \geq \overline{S}_q^{r+1}$. Это противоречит нашему предположению о том, что $\overline{S}_q^r < \overline{S}_q^{r+1}$. По индукции условие $\overline{S}_q^{m^*} = \overline{S}_q^{m^*+1} = \ldots = \overline{S}_q^m = \overline{S}_q^{m+1}$ истинно.

Доказательство завершено.

Лемма 4.4. $S_q^r \geq \overline{S}_q^r$, r=1, 2, …, m+1.

Доказательство. Цель - развернуть наименьшее количество экземпляров m_0 таким образом, чтобы $\Pi^{m_0} = \min\left\{\Pi^1, \Pi^2, ..., \Pi^m, \Pi^{m+1}\right\}$. Таким образом, определяем m_0 как наименьший индекс, такой, что $\Pi^{m_0} = \min\left\{\Pi^1, \Pi^2, ..., \Pi^m, \Pi^{m+1}\right\}$, $1 \le m_0 \le m+1$. Из определения m_0 и леммы 4.2 следует, что $\Pi^{m_0-1} > \Pi^{m_0} = \Pi^{m_0+1} = \Pi^m = \Pi^{m+1}$, где мы устанавливаем $\Pi^0 = \infty$.

Предположим, что $S_q^r < \overline{S}_q^r$ для конкретного r, где $1 \le r \le m+1$. При оптимальном решении задачи DQ_1^r могут возникнуть следующие два варианта.

1. $Z_x^r < r$,

2. $Z_x^r = r$.

В случае 1 оптимальным решением является выполнимое решение задачи $\overline{D}\overline{Q}_1^r$. Таким образом, мы имеем $S_q^r \ge \overline{S}_q^r$. Это противоречит нашему предположению о том, что $S_q^r < \overline{S}_q^r$. В случае 2 оптимальное решение DQ_1^r может быть реализовано в $\overline{D}\overline{Q}_1^r$ с лучшим значением целевой функции, чем $\overline{S}_q^r$. Это противоречит тому факту, что $\overline{S}_q^r$ является оптимальным.

Доказательство завершено.

Лемма 4.5. $\Pi^{m_0-1} > \Pi^{m_0} = \Pi^{m_0+1} = \Pi^m = \Pi^{m+1}$ и $m_0 \le m^* \le m$.

Доказательство. Из леммы 4.3 следует, что $m^* \le m$. Необходимо доказать, что $m_0 \le m^*$. Пусть $\mu = \overline{S}_q^{m^*-1} - \overline{S}_q^{m^*}$ и $\mu > 0$ в соответствии с леммой 4.3. Пусть $r = m^*+1$ и рассмотрим два следующих варианта:

1. $\mu < a$,

2. $\mu \ge a$.

Случай 1 подразумевает, что существует выполнимый график для задачи DQ_1^r, имеющей m^*-1 экземпляров, значение целевой функции

$\overline{S}_q^{m^*-1} + a(m^* - 1)$ которой лучше значения $\overline{S}_q^{m^*} + am^*$, соответствующего развертыванию ровно m^* экземпляров. Таким образом, будет найдено такое решение DQ_1^r, при котором $m_0 \leq m^*$.

В случае 2 существует выполнимый график для задачи DQ_1^r, имеющей ровно m^* экземпляров, значение целевой функции $\overline{S}_q^{m^*} + am^*$ лучше значения $\overline{S}_q^{m^*+1} + a(m^* + 1)$, соответствующего развертыванию ровно m^*+1 экземпляра, поскольку $\overline{S}_q^{m^*} = \overline{S}_q^{m^*+1}$ (лемма 4.3). DQ_1^r позволяет найти такое решение, при котором $m_0 \leq m^*$. Следовательно, $m_0 \leq m^*$.

Из леммы 4.2 и приведенных выше аргументов следует, что $\Pi^{m_0-1} > \Pi^{m_0} = \Pi^{m_0+1} = \Pi^m = \Pi^{m+1}$ и $m_0 \leq m^* \leq m$.

Доказательство завершено.

Теорема 4.2. Алгоритм 3 находит оптимальное количество экземпляров, требуемых в задаче DQ_1.

Доказательство. Шаги 3-11 алгоритма 3 определяют значение m* на основе результатов, приведенных в лемме 4.3.

Шаги 12-18 алгоритма 3 определяют значение m_0 на основе результатов, приведенных в лемме 4.5. Из леммы 4.2 и леммы 4.5 следует, что значение m_0, полученное с помощью алгоритма 3, является оптимальным числом экземпляров, требуемым в задаче DQ_1.

Доказательство завершено.

С помощью структурированной стратегии, описанной в алгоритме 3, который использует свойства, установленные в приведенных выше леммах, практические задачи малого и среднего размера могут быть решены оптимально. Типичными реальными примерами таких проблем малого и среднего масштаба являются внутренние приложения компании, где количество пользователей ограничено и, следовательно, количество входящих запросов невелико. Сюда относятся приложения для учета или

ценообразования, предназначенные для конкретной фирмы, доступ к которым имеют только сотрудники этих фирм. Практические задачи большого размера обычно не могут быть решены до тех пор, пока не закончится память или не превысятся временные рамки. Поэтому в разделе 4.2 предлагаются субоптимальные OFD для быстрого решения более крупных задач реального мира.

4.2. Автономные динамические алгоритмы

Разработаны автономные динамические алгоритмы для воспроизведения результатов оптимизационной модели в более сжатые сроки. Минимальное количество экземпляров всегда находится между единицей и верхним граничным числом экземпляров, m, которое определяется с помощью алгоритма 1 с нулевым штрафом, рассмотренного в разделе 4.1.4. Следовательно, OFD4 перебирает все возможное фиксированное количество экземпляров от единицы до верхней границы, m. Вместо оптимизации числа идеальное размещение каждого запроса в наилучшем месте в наилучший временной интервал алгоритм размещает запросы по мере их поступления в очередь. Затем очередь сортируется по критерию, описанному ранее, чтобы изменить порядок, в котором различные типы запросов ищут доступную емкость в фиксированном количестве экземпляров. Если емкость экземпляра недоступна, запросы остаются в очереди и начисляются соответствующие штрафы за задержку. Затем определяем минимальную общую стоимость (штраф за задержку плюс затраты на развертывание экземпляра) для всего возможного фиксированного количества экземпляров на каждом горизонте планирования.

Разрабатывается OFD4, используя метод сортировки очереди в алгоритме 2, чтобы упростить внедрение облачным провайдером. Поскольку OFD просто назначают запросы экземпляру (или помещают их

в очередь) по мере поступления запросов, облачные провайдеры могут использовать этот метод в качестве критерия принятия решений о том, где и когда размещать входящие запросы в развернутых экземплярах в режиме реального времени, в дополнение к использованию его в качестве инструмента планирования пропускной способности для решения задач большого размера. Это означает, что результаты анализа должны точно отражать реальные сценарии, поскольку алгоритм 4 (OFD4) принимает решения в режиме реального времени и может быть непосредственно реализован поставщиками облачных услуг. Используя разработанные алгоритмы, облачный провайдер может значительно снизить затраты по сравнению с существующими подходами.

Выходные данные OFD4 могут различаться в зависимости от типа сортировки, выполняемой в процедуре назначения запросов в очереди (алгоритм 2). Как обсуждалось ранее, в литературе по планированию очереди предлагается множество критериев сортировки/отправки для различных приложений. Протестировано несколько различных критериев сортировки, которые обобщены в подразделе 4.2.1. Затем в подразделе 4.2.2 проведен анализ того, почему некоторые процедуры сортировки работали с лучше, чем другие.

4.2.1. Описание процедур проверки критериев сортировки

Критерии сортировки могут основываться на фиксированных значениях, таких как размер или время обработки различных типов запросов, или динамических значениях, таких как накопленный штраф или количество запросов, ожидающих в очереди. Могут даже существовать сложные процедуры сортировки, которые начинают сортировку по одному измерению, но переопределяют приоритет определенного типа запроса при превышении различных динамических пороговых значений. Пока поступления запросов остаются случайными, ни одна процедура сортировки не будет идеальной во всех сценариях.

Алгоритм 4 OFD4). Определение минимального количества экземпляров, близкого к оптимальному

```
1 procedure OFD(m, n, k, γ_t^j)
2       for H=1…m do
3             Call Assign-Queries-In-Queue из Алгоритма 2
4       end for
5       Находим наименьшую общую стоимость штрафных санкций плюс
        развертывание экземпляров на фиксированном количестве от 1 до m
        экземпляров.
6       Установить значение H* равным количеству экземпляров, связанных
        с этой наименьшей общей стоимостью.
7       return H*
8 end procedure
```

Алгоритм 5. Определение верхней границы нулевого ожидания

```
1 procedure Zero-Wait (n, γ_t^j)
2       m= 1
3       for t =1,n do
4             Удалить полностью выполненные запросы из всех экземпляров
5             Поместить запросы, поступающие в t, в первый экземпляр
              среди m экземпляров с достаточной оставшейся емкостью
6             while остаются неназначенные запросы do
7                   m=m+1
8                   Разместить неназначенные запросы в экземпляре m до
                    тех пор, пока у него остается достаточная емкость
9             end while
10            Обновить время выполнения запросов, назначенных каждой
              ячейке, на τ
11      end for
12      return m
13 end procedure
```

Проанализировано шесть процедур сортировки по SPT, LPT, наибольшей суммарной задержке, наибольшему суммарному штрафу за задержку, наибольшему количеству ожидающих и наибольшему следующему штрафу. SPT имеет ожидающий тип запроса с наименьшим временем обработки, который сначала ищет доступный развернутый экземпляр, в то время как LPT имеет самый длительный по времени обработки поиск запроса. Наибольшая суммарная задержка определяется

типом запроса с наибольшей текущей суммарной задержкой поиска в первую очередь, тогда как наибольший суммарный штраф за задержку просто умножает наибольшую суммарную задержку на значение штрафа для каждого типа запроса. Это означает, что запрос с большим штрафным типом сначала выполняет поиск вакансии, если время задержки у них одинаковое. Наибольшее количество ожидающих запросов подсчитывает количество запросов, ожидающих выполнения, и запускает поиск доступных развернутых экземпляров с типом запроса, у которого самая длинная очередь. Наконец, следующим по величине штрафом будет тип запроса с наибольшим штрафом за задержку поиска. Установлено, что SPT лучше всего работает практически на всех горизонтах планирования поступления запросов в данных клиентских приложений, что подтверждает текущую литературу по DDM.

4.2.2. Обсуждение выбора процедуры сортировки

Понимание наилучшей процедуры сортировки для приложения очень важно для практиков, поскольку это сэкономит много времени и снизит сложность принятия решений. Кроме того, облачный провайдер хотел бы использовать наилучшую процедуру сортировки для назначения запросов экземплярам в режиме реального времени по мере поступления запросов. Установлено, что более длинные и сложные типы запросов оказывают большое влияние на количество требуемых экземпляров, поэтому можно было бы подумать, что запрет на то, чтобы определенный тип запросов всегда блокировался и никогда не выполнялся, уменьшит общие штрафные санкции.

Однако, после анализа данных облачного провайдера установлено, что сортировка SPT имела наименьшее значение целевой функции в 92,8% случаев на всех горизонтах планирования и находилась в пределах 1% от наименьшего значения целевой функции в 97,0% случаев.

Комбинированный анализ целевой функции и минимального количества экземпляров в большей степени благоприятствует SPT, поскольку несколько раз, когда SPT не имел наименьшего значения целевой функции, он все равно рекомендовал минимальное количество экземпляров.

В целом, существует только пять (из 168) горизонтов планирования, где значение целевой функции сортировки SPT было более чем на 5% выше самого низкого значения целевой функции сортировки. В трех из этих периодов планирования сортировка по наибольшему количеству ожидающих была самой низкой, в то время как сортировка по наибольшему количеству следующих штрафных санкций была самой низкой в двух других. На всех пяти из этих горизонтов планирования было всего несколько сложных запросов (типы 15-25), которые выполнялись близко друг к другу, и наибольшее количество процедур сортировки ожидающих и следующих штрафных санкций позволило определить приоритетность этих сложных запросов и снизить их штрафные значения. Поскольку сложные запросы на этих горизонтах планирования возникали нечасто, дополнительная задержка, возникавшая при выполнении несложных запросов (при ожидании выполнения сложных запросов), была компенсирована многими запросами такого типа без задержек. Таким образом, среднее время выполнения для несложных типов запросов было ниже порога штрафной задержки, и штрафные санкции не начислялись. Кроме того, есть только два горизонта планирования (127 и 162), на которых процедура SPT не имела минимального значения целевой функции или минимального количества экземпляров. В обоих этих горизонтах планирования большинство сложных запросов поступало в начале или в конце горизонта планирования, и ни один запрос не поступал в середине. Однако в обоих горизонтах планирования SPT рекомендовал использовать три экземпляра, в то время как другие процедуры сортировки

рекомендовали два экземпляра. Количество развернутых экземпляров было уменьшено только на один экземпляр, использующий SPT.

Таким образом, результаты показывают, что в клиентском приложении лучше сначала запускать все несложные типы запросов с более коротким сроком выполнения с помощью процедуры сортировки SPT. Более сложные типы запросов могут быть отложены, но, вероятно, они все равно были бы отложены. Таким образом, лучше никогда не откладывать выполнение несложных типов запросов (которые встречаются чаще) и немного дольше откладывать выполнение более сложных типов запросов.

Практический совет 1. При прочих равных условиях облачные провайдеры предпочли бы сначала выполнять несложные запросы с более коротким сроком выполнения, поскольку они выполняются быстрее и одновременно может быть обработано больше запросов по сравнению с более сложными запросами, выполнение которых занимает больше времени. Это понимание усиливается при наличии буфера между средним временем задержки и пороговым значением штрафа, поскольку это означает, что в оптимальном решении произойдет некоторое ожидание, поскольку небольшое время ожидания не будет наказуемо и потребуется меньше ресурсов.

Для снижения эффективности процедуры сортировки SPT потребуется кардинально изменить приложение для обработки данных. Это может произойти, если у облачного провайдера в сложных запросах значения штрафных санкций в 50-100 раз выше, чем в простых запросах, или если время буферизации между временем выполнения и пороговым значением штрафных санкций очень велико для простых запросов и очень мало для сложных запросов. Это возможно, поскольку соглашение об уровне обслуживания является предметом переговоров. Однако это не типично, поскольку клиенты облачных вычислений не хотят ждать

обработки простых (как правило, более объемных) запросов и на самом деле более терпимо относятся к некоторой задержке при выполнении сложных запросов, поскольку они уже ожидают, что эти запросы займут много времени.

4.3. Анализ данных численного приложения

Используя данные от облачного провайдера (описанные в подразделе 4.3.1), оценивается эффективность результатов разработанного алгоритма по сравнению с текущими подходами к управлению горизонтальной эластичностью облачных провайдеров, описанными в разделах 4.3.2 и 4.3.3. В заключение приведены общие выводы из данных, раздела 4.3.4.

4.3.1. Описание вычислительных данных

Данные получены из облачного веб-приложения для мониторинга объектов интернет-вещей, которое обслуживает тысячи пользователей клиента. В таблице 4.6 приведены данные о параметрах для всех 25 типов запросов, а на рисунке 4.1 показано общее количество запросов и количество сложных запросов (типы запросов 15-25) для каждого горизонта планирования. Каждый тип запроса требует разного времени выполнения и объема вычислительных ресурсов, таких как фиксированное количество виртуальных процессоров (vCPU), памяти (GiB) и дискового хранилища (GB). Существует также согласованный порог штрафной задержки для каждого типа запроса, который определен в соглашении об уровне обслуживания между поставщиком облачных услуг и клиентом.

По данным поставщика облачных услуг ресурсоемкость каждого развертываемого экземпляра составляет 10 виртуальных процессоров, 25 ГБ оперативной памяти и 80 ГБ дискового хранилища. Стоимость развертывания каждого отдельного экземпляра в течение часа составляет

250 руб. Все стоимостные параметры были уменьшены на константу для обеспечения конфиденциальности, без потери структуры реальных данных.

Таблица 4.6

Входные параметры для каждого типа запроса

ID запроса	Время исполнения, с	vCPU	Объем ОЗУ, Гб	Объем памяти, Гб	Штраф за секунду
1	1.1	0.25	0.2	0	0.001736
2	1.25	0.25	0.4	0	0.001736
3	1.35	0.5	0.8	0	0.003472
4	1.45	0.5	1.5	6	0.006944
5	1.55	0.5	1.6	0	0.006944
6	1.69	0.5	3.05	6	0.008146
7	1.89	1	1.5	8	0.008146
8	2.11	1	3	8	0.008013
9	2.36	1	6.1	50	0.043403
10	2.62	1	12.2	0	0.043403
11	2.87	2	3	12	0.032585
12	3.22	2	6.4	0 0.	017094
13	3.74	2	12.2	25	0.032585
14	4.47	2	24.4	48	0.065171
15	5.44	4	6	32	0.065171
16	6.51	4	6	0	0.027778
17	7.46	4	12.8	0	0.034188
18	8.5	4	24.4	64	0.065171
19	9.51	4	24.4	50	0.065171
20	22.07	4	24.4	25	0.065171
21	35.69	8	12	24	0.065171
22	67.79	8	12	64	0.055556
23	119.41	8	24.4	64	0.065171
24	199.25	8	25.2	0	0.067308
25	357.2	9	12	0	0.067308

Наконец, есть данные о поступлении запросов за целую неделю. То есть выдаются день, время и идентификатор запроса для каждого запроса, запрошенного любым пользователем системы мониторинга за эту неделю.

Оптимизационная модель, а также алгоритмы реализованы C++. Для получения результатов оптимизации использован пакет CPLEX 12.7.1

[4.29]. Все реализации алгоритмов запускались на серверах с 24-ядерным процессором Intel(R) Xeon(R) CPU E5-2683 v4 с тактовой частотой 2,1 ГГц, 64 МБ кэш-памяти и 256 ГБ оперативной памяти, работающих под управлением операционной системы Ubuntu 16.04.3.

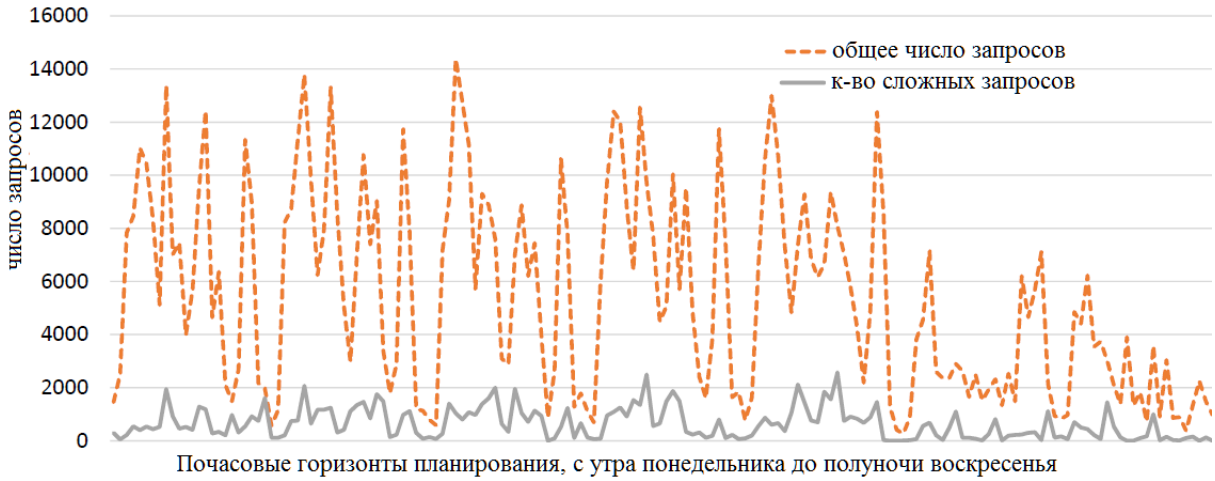


Рисунок 4.1. Общее число запросов и сложных запросов (типа 15-25) для каждого часового горизонта планирования, начиная с утра понедельника (слева) и заканчивая полуночью воскресенья (справа от оси x)

4.3.2. Результаты по верхней границе

Улучшение верхней границы для алгоритма с нулевым штрафом по сравнению с алгоритмом с нулевым ожиданием связано с тем, что среднее время ожидания должно превышать пороговое значение задержки для начисления штрафных санкций. Таким образом, возможны небольшие задержки, за которые не начисляются штрафные санкции.

Практический совет 2. Использование алгоритма с нулевым штрафом может снизить количество развернутых экземпляров в среднем на 16,8% по сравнению со значением нулевого ожидания. Это иллюстрирует важность использования зависящих от контекста вычислений верхней границы, которые улучшают стандартные процедуры упаковки в ячейки в алгоритме с нулевым ожиданием.

Сложное время поступления запроса является основной причиной того, что алгоритм с нулевым штрафом заметно уступает алгоритму с нулевым ожиданием на некоторых горизонтах планирования. Например, для горизонта планирования №15 верхняя граница сокращена всего на 1,7% благодаря алгоритму с нулевым штрафом, поскольку все 139 запросов типа 22 (сложных) поступают в течение первых 92 секунд, а дополнительные запросы типа 22 не поступают.

Следовательно, значение нулевого штрафа может быть лишь в несколько раз меньше значения нулевого ожидания. В отличие от этого, для горизонта планирования №26 верхняя граница сокращена на 37,0% благодаря алгоритму нулевого штрафа. На этом горизонте планирования существует период интенсивного поступления сложных запросов, который задает значение нулевого ожидания. Однако многие более сложные запросы поступают на протяжении всего горизонта планирования небольшими пакетами, и все эти запросы могут выполняться без задержек с меньшим количеством экземпляров, чем в период интенсивного поступления. Таким образом, более длительные задержки в период загруженности запросов компенсируются более сложными запросами без задержек. Это позволяет снизить среднюю задержку для каждого типа запроса при меньшем количестве обращений, что позволяет получить нулевое значение штрафа на 37,0% ниже, чем при нулевом значении ожидания. Для дальнейшего улучшения результатов определения верхней границы используются алгоритмы оптимизации, позволяющие найти наилучший баланс между минимальными штрафами за задержку и низкими затратами на развертывание экземпляра.

4.3.3. Сравнение данных

Используя верхние граничные значения из раздела 4.5.2, программа OFD4 может определить рекомендуемое количество развернутых

экземпляров для всех 168 еженедельных горизонтов планирования. В этом разделе рассмотрен иллюстративный набор выборочных траекторий поступления запросов (по одной для каждого горизонта планирования), которые наблюдались в данных. В разделе 4.4.2 проводится аналогичное сравнение с использованием стохастической модели и используем миллионы случайно сгенерированных выборочных траекторий, чтобы охарактеризовать, как модель и OFD4 реагируют на неопределенность спроса. Современный подход Practice Max20 основан на использовании максимального количества развернутых экземпляров, необходимого для любого временного интервала на горизонте планирования, и увеличении этого значения на 20%. Подход Avg 20 рассчитывается путем использования среднего количества развернутых экземпляров за все периоды времени в пределах горизонта планирования и увеличения этого значения на 20%.

В этом иллюстративном примере пути поступления запроса Practice Max20, по-видимому, развертывается во многих случаях на всех горизонтах планирования. Фактически, практический подход Avg20 использует в среднем на 47% меньше экземпляров на всех 168 горизонтах планирования, чем предложенные алгоритмы.

Однако основную причину, по которой специалисты-практики обычно не используют подход Practice Avg 20, можно увидеть на рисунке 4.2, где сравниваются общие затраты (штрафная задержка и затраты на развертывание экземпляра) для разных подходов. В этом примерном сценарии траектории, основанном на данных, подход Practice Avg20 снижает затраты на 70,8% горизонтов планирования по сравнению с Practice Max20.

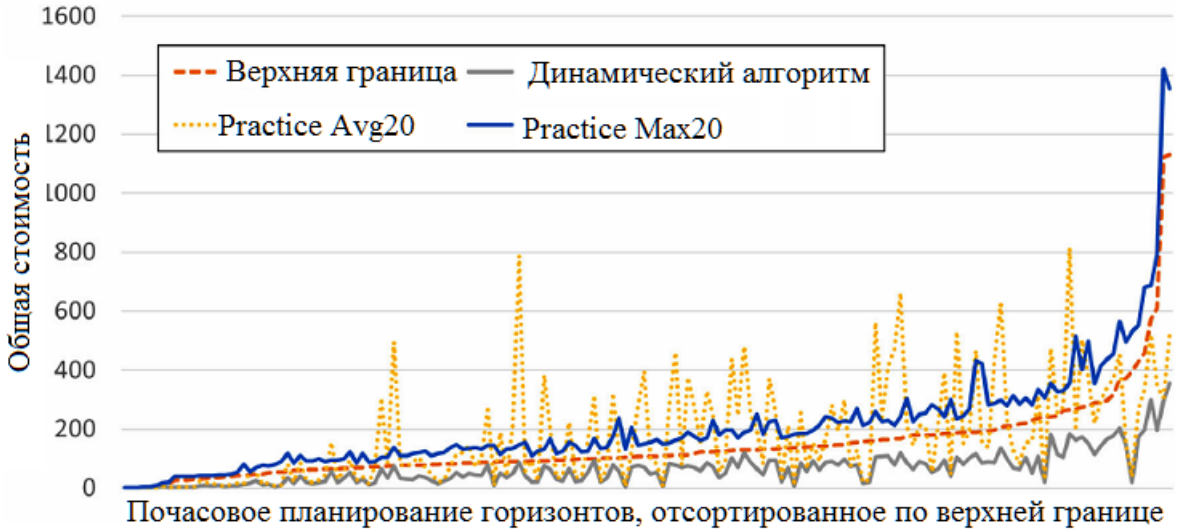


Рисунок 4.2. Сравнение итоговых затрат целевой функции по верхней границе и OFD с текущими практическими подходами к управлению потенциалом горизонтальной эластичности. Горизонты планирования отсортированы по их верхней границе

В среднем переход от подхода Practice Max20 к подходу Practice Avg20 снижает затраты на 14,8% на всех горизонтах планирования. Интересно, однако, что на некоторых этапах планирования использование подхода Practice Avg20 приводит к четырехкратному увеличению затрат на внедрение Practice Max20. Сравнение синей сплошной линии “Practice Max20” с желтой пунктирной линией “Practice Avg20” на рисунке 4.2 показывает, что незначительное количество горизонтов планирования с высокими штрафными санкциями может быть очень дорогостоящим. Таким образом, большинство поставщиков облачных услуг предпочитают более стабильные затраты на развертывание Practice Max20 чрезмерным штрафным расходам, которые могут возникнуть при использовании подхода Practice Avg20 в сценарии, подобном этому. Кроме того, клиенты облачных сервисов предпочитают практиковать уровни развертывания Max20, поскольку они практически не сталкиваются с задержками в выполнении.

По мере того как облачные сервисы становятся все более конкурентоспособными, а поставщики облачных услуг стремятся улучшить использование ресурсов, им потребуются решения, подобные нашему, для сокращения развертывания экземпляров и ограничения больших штрафных санкций (и недовольства клиентов), которые возникают в результате чрезмерных задержек выполнения. На рисунке 4.2 показано, что серая линия динамического алгоритма всегда приводит к наименьшим общим затратам для каждого горизонта планирования в этом примере сценария с данными о траектории. Установлено, что OFD4 приводит к снижению общих затрат в среднем на 53,6% и 68,0% по сравнению с верхним пределом и Pracice Max20, соответственно. Что еще более важно, затраты всегда ниже на всех горизонтах планирования, а OFD приводит к общему снижению затрат более чем на 60% по сравнению с Pracice Max20 более чем на 75% горизонтов планирования.

OFD также снижает общие затраты как минимум на 28% по сравнению с верхней границей нулевого штрафа более чем на 90% горизонтов планирования.

4.3.4. Дополнительная информация о данных

В ходе анализа, рассмотренного ранее, установлено, что дополнительные 1000 запросов типа 1 на горизонте планирования не оказывают существенного влияния на результаты. Однако добавление 30-40 более сложных запросов может оказать заметное влияние. Для целей данного обсуждения предполагаем, что запросы типов 1-14 являются несложными, а запросы типов 15-25 - сложными.

Это означает, что решение всех несложных запросов занимает менее 5 секунд и требует использования двух или менее VCPU, в то время как решение сложных запросов занимает более 5 секунд и требует от 4 до 9 VCPU. Все шесть горизонтов планирования с наименьшими верхними

границами не содержат сложных запросов. Как правило, процент сложных запросов увеличивается на горизонтах планирования при меньшем общем количестве запросов, но неизменной верхней границе. Это можно лучше понять, рассмотрев временные показатели использования ресурсов для различных запросов. Например, 10 запросов типа 20 потребляют 88,3 vCPU-секунды, в то время как 300 запросов типа 1 потребляют всего 82,5 vCPU-секунды. Поскольку сложные запросы также требуют памяти и ресурсов хранилища для увеличения времени обработки, легко понять, почему для одного сложного запроса могут потребоваться ресурсы, аналогичные более чем 30 несложным запросам. Количество сложных запросов также влияет на значения OFD, особенно на величину уменьшения по сравнению с верхней границей. В данных наших клиентских приложений мы находим, что 86% горизонтов планирования были сокращены на 42% или более по сравнению с верхним граничным значением. Из 12 файлов данных, количество которых сократилось менее чем на 42%, в трех уже было нулевое значение ожидания, равное единице, и дальнейшее сокращение было невозможно. В остальных девяти файлах процент сложных запросов был выше (12.2%, 25.4%, 43.0%, 65.5%, 81.6%, 23.88%, 52.0%, 10.3%, 38.0% чей средний показатель составил 39,7%) по сравнению с общим средним процентом сложных запросов, который составил всего 13,5%.

Сложные запросы также, по-видимому, оказывают значительное влияние на время решения оптимальной модели. Именно увеличение времени обработки сложных запросов приводит к усложнению процесса оптимального решения. Например, горизонт планирования №1 имеет значение нулевого ожидания, равное 40 экземплярам, и значение OFD, равное 14 экземплярам.

Решая задачу DQ_1 для горизонта планирования №1 на компьютере с 32 ГБ оперативной памяти, можно было добраться только до первой

строки таблицы ветвей и привязок CPLEX, прежде чем закончится объем памяти. Сохранение всего остального без изменений, но сокращение времени выполнения всех запросов до 1 секунды (или одного временного интервала) позволило тому же компьютеру решить задачу DQ_1 для горизонта планирования № 1 примерно за 6 часов.

Таким образом, предполагается, что более сложные запросы (с более длительным временем выполнения) увеличивают сложность оптимизации и время решения коммерческих задач решателями.

Практический совет 3. Сложные запросы с более длительным временем выполнения и более высокими требованиями к пропускной способности оказывают большее влияние на верхние границы, OFD, оптимальные значения и оптимальное время решения, чем большее количество несложных запросов. Поэтому облачным провайдерам следует уделять больше внимания согласованию условий соглашения об уровне обслуживания для сложных запросов и настаивать на более мягких пороговых значениях штрафных санкций и/или величинах штрафных санкций за сложные запросы.

Поскольку сложные запросы оказывают большее влияние, им следует уделить дополнительное внимание при заключении соглашения об уровне обслуживания. Во-первых, поскольку сложные запросы приводят к значительному потреблению ресурсов, они могут иметь меньшие штрафные санкции в секунду, когда среднее время ожидания превышает порог штрафной задержки. Это позволило бы дольше задерживать выполнение некоторых сложных запросов без больших штрафных санкций. Таким образом, было бы развернуто меньше экземпляров, поскольку задержка сложных запросов обходилась бы не так дорого.

Во-вторых, для сложных запросов могут быть установлены более высокие пороговые значения задержки. Если бы можно было увеличить время, необходимое для перевода другого экземпляра в оперативный

режим, плюс время выполнения запроса, то автономное управление горизонтальной эластичностью могло бы сосредоточиться только на несложных запросах типов 1-14, а сложные запросы могли бы просто привести к увеличению числа развернутых экземпляров в реальном времени при поступлении сложных запросов. Однако это может привести к чрезмерному циклическому включению/выключению экземпляров. Наконец, если клиент настаивает на сокращении времени выполнения сложных запросов, можно использовать более крупные экземпляры, способные обрабатывать несколько сложных запросов одновременно. В целом, при заключении соглашения об уровне обслуживания следует учитывать некоторые из этих условий для сложных запросов, чтобы помочь поставщику облачных услуг лучше планировать свои развертываемые экземпляры в соответствии с требованиями клиента.

4.4. Стохастическая реализация с неопределенностью спроса

В этом разделе мы демонстрируем, как модели оптимизации и OFD могут быть реализованы в условиях неопределенного горизонта планирования для получения надежного решения. При неопределенности спроса общая стоимость меняется в зависимости от выбранного пути поступления запросов. Поэтому желательно минимизировать ожидаемые затраты, когда спрос является стохастическим, как показано в задаче DQ_2. В разделе 4.4.1 показано, как метод аппроксимации выборочного среднего (SAA) может быть использован для решения стохастической версии задачи. В разделе 4.4.2 мы представлены результаты использования метода SAA, которые показывают, что оптимизационная модель и коэффициенты устойчивы к неопределенности спроса.

Предполагаем, что запросы имеют определенное распределение вероятности поступления. Таким образом, будет множество сценариев

спроса (т.е. примерных путей) для поступления запросов в течение горизонта планирования.

Пусть γ_t^{sj} - целочисленное значение запросов типа j, поступающих в начале временного интервала t сценария спроса s, где t=1, 2, ..., n и Π^{s^*} является оптимальным целевым значением для оптимального решения для сценария спроса s. Оптимальное решение для данного сценария s может быть получено путем решения задачи DQ_1 для каждого сценария s, где s=1, 2, ..., K и K - количество сценариев спроса. Обратите внимание, что K может быть очень большим числом. Пусть u_j^s - общее количество запросов типа j в сценариях, поступивших в течение периода планирования ($u_j^s = \sum_{t=1}^{n} \lambda_t^{sj}$). Аналогично, следующая запись $\left(Y_{it}^{sj}, I_t^{sj}, \gamma_t^{sj}, L_{it}^{sj}, P_j^s, Q_j^s, V_j^s\right)$, соответствующая таблице 4.2, может быть определена для выборочного пути s. Пусть ρ^s - вероятность реализации сценария s. Находим оптимальное возможное решение, которое минимизирует ожидаемые затраты.

Задача DQ_2:

$$\text{Min } \Psi_2 = \sum_{s=1}^{K} \rho^s \Pi^s(X_i, Y_{it}^{sj})$$

Ограничения:

$$\Pi^s(X_i, Y_{it}^{sj}) = \sum_{j=1}^{k} Q_j^s + \alpha \sum_{i=1}^{m} X_i, \quad s = 1,2,...K \tag{4.13}$$

$$I_t^{sj} = I_{t-1}^{sj} + \gamma_t^{sj} - \sum_{i=1}^{m} Y_{it}^{sj}, \; s = 1,2,...,K; \; j = 1,2,...,k; t = 1,2,...,n \tag{4.14}$$

$$Y_{it}^{sj} \le u_j^s X_i \; s = 1,2,...,K; i = 1,2,...,m; \; j = 1,2,...,k; \; t = 1,2,...,n \tag{4.15}$$

$$L_{it}^{sj} \ge \sum_{r=0}^{\lambda_j - 1} Y_{i,t-r}^{sj} \; s = 1,2,...,K; i = 1,2,...,m; \; j = 1,2,...,k; \; t = 1,2,...,n \tag{4.16}$$

$$\sum_{j=l}^{k} c_r^j L_{it}^{sj} \le C_r X_i,\ s=1,2,...,K; i=1,2,...,m; t=1,2,...,n; r=1,2,3 \qquad (4.17)$$

$$P_j^s \ge \frac{\sum_{t=0}^{n} \tau I_t^{sj}}{u_j^s},\ s=1,2,...,K; j=1,2,...,k \qquad (4.18)$$

$$P_j^s + \lambda_j \ge \mu_j + V_j^s M,\ s=1,2,...,K; j=1,2,...,k \qquad (4.19)$$

$$Q_j^s \ge b_j u_j^s \left(P_j^s + \lambda_j - \mu_j\right) - \left(1 - V_j^s\right) M,\ s=1,2,...,K; j=1,2,...,k \qquad (4.20)$$

$$Y_{it}^{sj}, I_t^{sj}, Ls_{it}^{j} > 0,\ i=1,2,...,m; j=1,2,...,k; t=1,2,...n; s=1,2,...,K \qquad (4.21)$$

$$P_j^s, Q_j^s \ge 0,\ j=1,2,...,k; s=1,2,...,K \qquad (4.22)$$

$$X_i, V_j^s \in \{0,1\},\ i=1,2,...,m; j=1,2,...,k; s=1,2,...,K \qquad (4.23)$$

Лемма 4.6. Если учесть все возможные сценарии K для представления неопределенности, то минимальные ожидаемые затраты Ψ_2 могут быть получены путем решения задачи DQ_2.

Доказательство.

Решение Π^{S^*} получается путем решения задачи DQ_1 для каждого сценария s. Поскольку затраты Ψ_2 включают в себя все затраты, соответствующие всем возможным выборкам, задача DQ_2 получает минимальную ожидаемую стоимость.

Доказательство завершено.

4.4.1. Метод SAA

Чтобы решить стохастическую версию задачи, сформулированную выше, нужно рассмотреть экспоненциальное число путей выборки в задаче DQ_2. Чтобы устранить эту проблему, используем метод SAA [4.37]. Таким образом, предлагается формулировку SAA (с решением $\left(X_i, Y_{it}^{sj}\right)$) для решения задачи DQ_2 в задаче SAA. Определяем подходящий размер выборки, k, который обеспечивает статистически малый разрыв между

оптимальностью DQ_2 и соответствующей задачей SAA. Используя распределения вероятностей для данных о спросе (поступлении запросов), процесс выборки определяет размер выборки (k) и количество повторений проверенных выборок (v). Значение v увеличивается при заданном значении k до тех пор, пока не будет достигнута требуемая точность. Если эта точность не достигнута, то увеличивается размер выборки, k, и повторяется процесс до тех пор, пока не будет достигнута желаемая точность.

Задача DQ_1 может быть эффективно решена с помощью существующих решателей, таких как CPLEX, для небольших клиентских приложений. Однако, поскольку потенциально необходимо решить миллионы сценариев (т.е. примеры путей), используем OFD для получения данных Π^{s^*} для каждого сценария s. В частности, для заданного распределения запросов начинаем со случайной генерации k путей выборки с γ_t^{sj} входящими запросами для каждого пути выборки. Затем решаем задачу SAA и получаем оптимальное целевое значение для первой выборки,

$$\pi_k^1(y) = \frac{1}{k}\prod_s \left(X_i, Y_{it}^{sj}\right)$$

Аналогично, получаем $\pi_k^2(y)$, $\pi_k^3(y)$, …, $\pi_k^v(y)$ для v повторений, где y - оптимальное решение SAA с целевым значением $\pi_k(y)$ и $y=(x_i)$.

Оценка оптимального целевого значения задачи DQ_2 задается формулой

$$\overline{\pi}_k^v = \frac{\pi_k^1(y^1) + \pi_k^2(y^2) + ... + \pi_k^v(y^v)}{v} \qquad (4.25)$$

где $\pi_k^r(y^r)$ обозначает оптимальное целевое значение r-й репликации SAA, r=1, 2, …, v.

Можно рассчитать дисперсию выборки $V_k^v = \frac{\sum_{r=1}^{v}\left[\pi_k^r\left(y^r\right) - \overline{\pi}_k^v\right]^2}{v}$.

После v повторений приблизительный доверительный интервал в 100 (1-p)% для ожидаемого объективного значения (т.е. $E\left[\Psi_2^*\right]$) определяется по формуле

$$\overline{\pi}_k^v \pm t_{v-1,1-p/2}\sqrt{\frac{V_k^v}{v-1}} \tag{4.26}$$

Половина длины этого заданного доверительного интервала обозначается

$$\zeta(v,p) = t_{v-1,1-p/2}\sqrt{\frac{V_k^v}{v-1}} \tag{4.27}$$

Если оценка $\overline{\pi}_k^v$ удовлетворяет

$$\frac{\overline{\pi}_k^v - E\left[\Psi_2^*\right]}{E\left[\Psi_2^*\right]} = \xi \tag{4.28}$$

тогда можно утверждать, что относительная погрешность $\overline{\pi}_k^v$ равна ξ. Если увеличивать количество повторений до тех пор, пока $\frac{\zeta(v,p)}{\overline{\pi}_k^v}$ не станет меньше или равно ξ, то относительная погрешность $\overline{\pi}_k^v$ будет не более чем $\frac{\xi}{1-\xi}$ с вероятностью приблизительно 1-p [4.41]

Следовательно, чтобы получить оценку $E\left[\Psi_2^*\right]$ с относительной погрешностью ξ и доверительным уровнем 100(1-p) процентов, следуем процедуре [4.41], чтобы определить размер v для заданного размера выборки k.

Ниже перечислим шаги процедуры, описанной в [4.41].

Шаг 1: Установить v_0=5 случайных повторов (в каждом из которых k=100 путей выборки) и установить v=v_0.

Шаг 2: Вычислить $\overline{\pi}_k^v$ и $\zeta(v,p)$ из π_k^1, π_k^2, π_k^3, …, π_k^v с уровнем достоверности 95%.

Шаг 3: Если $\frac{\zeta(v,p)}{\overline{\pi}_k^v} < \frac{\xi}{1+\xi}$, то использовать $\overline{\pi}_k^v$ в качестве точечной оценки для Ψ_2^* и остановиться. В противном случае заменить v на v+1, выполнить еще одну репликацию моделирования и перейти к Шагу 2.

Шаг 4: Если $v>v^{max}$, увеличить k путей выборки на k^{step}=20, установить v=v_0=5 и повторить шаги 2-3, описанные выше.

4.4.2. Результаты SAA

Чтобы проиллюстрировать надежность оптимизационных моделей и OFD4 для определения неопределенности прибытия, применяем подход SAA на всех 168 горизонтах планирования для двух разных случайно сгенерированных наборов данных (а именно, RSP#1, RSP#2) и сравниваем результаты с типичными подходами практиков для определения количества развернутых экземпляров.

Чтобы обеспечить устойчивость предложенного подхода к случайным изменениям, используем два метода для генерации случайных путей поступления запросов. Используя метод генерации пути случайной выборки #1, случайным образом генерируем набор данных RSP#1. Сначала вычисляем среднее количество входящих запросов каждого типа для каждого горизонта планирования в наших данных. Затем мы используем эти средние значения в качестве параметров в распределениях Пуассона, из которых выбираем количество поступлений запросов в каждый временной интервал каждого горизонта планирования для каждого типа запросов, используя соответствующее среднее значение. Генерируем эти

выборки, используя функцию C++ для выборки с распределением Пуассона. Это создает пути выборки со случайным числом поступлений запросов каждого типа в каждом временном интервале для каждого горизонта планирования.

Поскольку также были пути выборки данных со многими временными интервалами, содержащими нулевые поступления запросов, за которыми следуют временные интервалы с большим количеством поступлений запросов, был создан второй набор путей случайной выборки (RSP#2), который объединяет поступления запросов в меньшее количество временных интервалов. Использование этой второй рандомизации помогает обеспечить устойчивость разработанного подхода к большей неопределенности поступления запросов. В RSP#2 сначала вычисляем процент временных интервалов с нулевым поступлением запросов на каждом горизонте планирования в наших данных. Также рассчитывается среднее количество поступлений запросов для каждого типа запросов на каждом горизонте планирования, пренебрегая временными интервалами с нулевыми поступлениями (по сравнению со средними значениями, рассчитанными в RSP#1, где не пренебрегали временными интервалами с нулевыми поступлениями). Затем генерируем пути случайной выборки для RSP#2 следующим образом: сначала определяем, должен ли временной интервал иметь нулевые поступления или нет, путем выборки случайного числа из равномерного распределения между 0 и 1.

Если выбранное число меньше процента временных интервалов с нулевыми поступлениями запросов, который рассчитан ранее, то считаем, что в этот временной интервал поступило ноль запросов, и переходим к следующему временному интервалу. Если равномерное случайное число превышает этот процент, то считаем, что в этот временной интервал поступило ненулевое количество запросов. Для этих последних временных интервалов выбираем количество поступлений запросов в них для каждого

типа запросов из распределения Пуассона с вышеупомянутыми средними параметрами.

Сами расчеты SAA выполняются с ξ=0,002, что означает, что есть 95%-ная уверенность в том, что ошибка составляет не более 0,2%. Установлен предел репликации, v_{max}, равным 40 (минимум 5), а количество путей выборки для каждой репликации - 180 (минимум 100). В RSP#1 (RSP#2) установлено, что 94,0% (88,7%) горизонтов планирования достигли погрешности χ=0,002. На тех горизонтах планирования, которые не достигали допустимого уровня ошибок, количество ошибок по-прежнему составляло менее 0,052%, и итоговое решение о развертывании экземпляра не менялось. Таким образом, по этим причинам и для полноты анализа включены все горизонты планирования в расчеты результатов, представленные в таблице 4.7 и на рисунках 4.3-4.6.

Таблица 4.7

Сравнение общей стоимости подхода OFD (Динамический алгоритм) 4 SAA с несколькими практическими подходами для определения количества развернутых экземпляров на будущих горизонтах планирования с использованием двух разных случайно сгенерированных наборов данных (RSP#1 и RSP#2). Общая стоимость - это средняя стоимость за горизонт планирования

	RSP#1		RSP#2	
Подход к принятию решений	Общая стоимость	% больше чем дин. алг. 4	Общая стоимость	% больше чем дин. алг. 4
Динамический алгоритм 4	1967		2020	
Practice Max20	5118	160.19	5701.182	23
Practice Avg20 Up	2060	4.73	2098	3.87
Practice Avg20 Down	2234	13.56	2395	18.59

Рисунок 4.3. Сравнение изменений в развернутых экземплярах при использовании автономного динамического алгоритма с текущими практическими подходами к управлению потенциалом горизонтальной эластичности с использованием случайного набора данных RSP#1

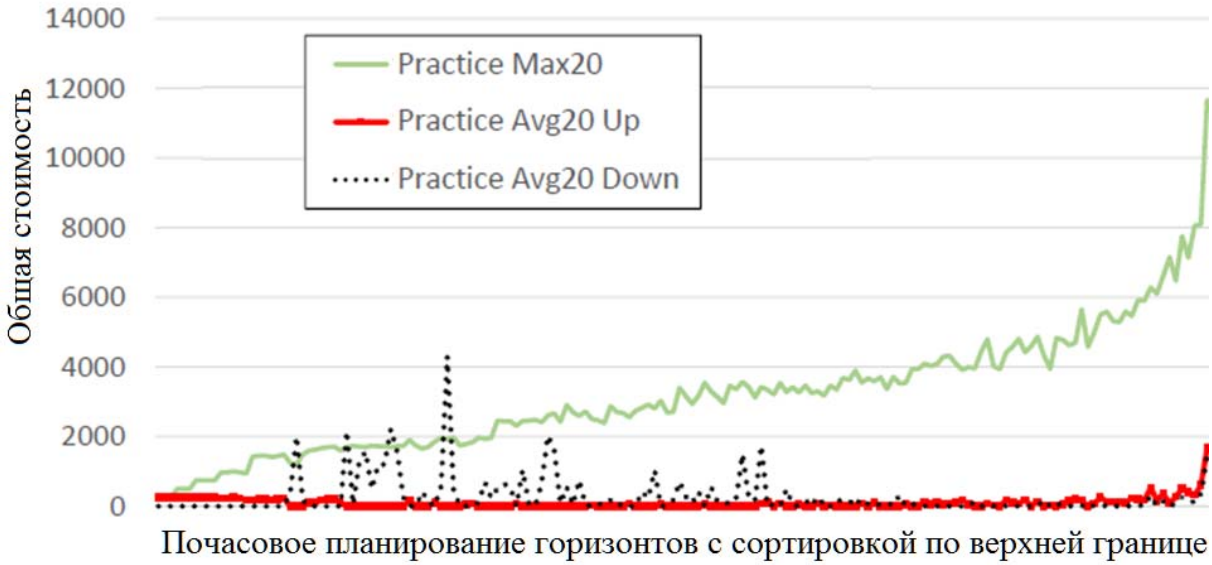


Рисунок 4.4. Сравнение изменения общих затрат при использовании автономного динамического алгоритма с текущими практическими подходами к управлению производительностью по горизонтальной эластичности с использованием случайного набора данных RSP#1

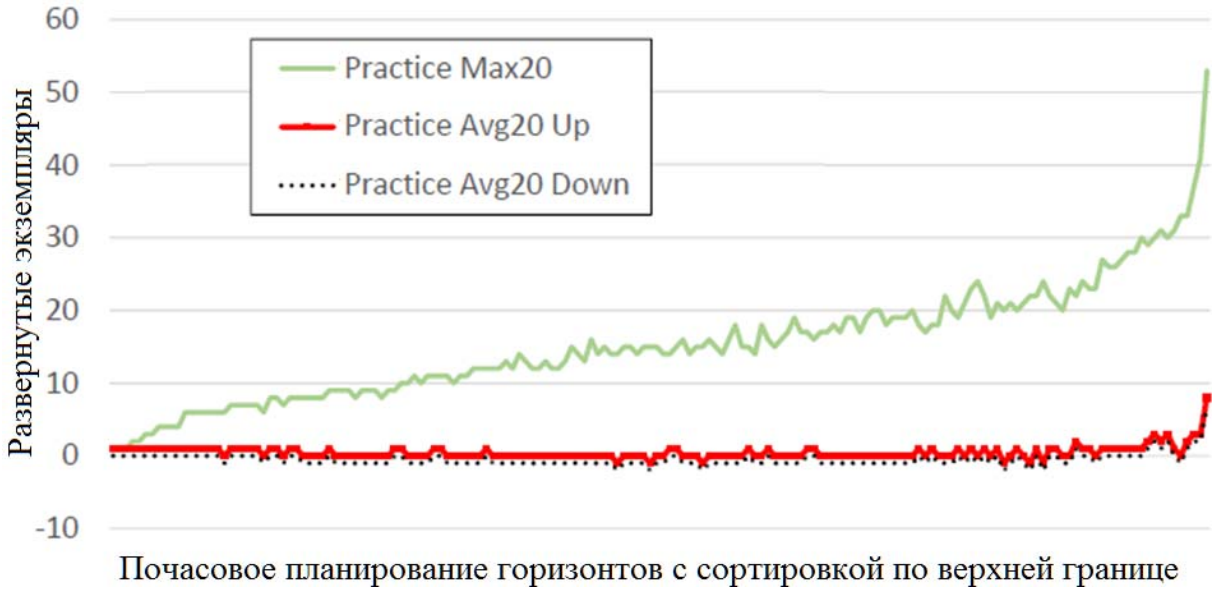


Рисунок 4.5. Сравнение изменений в развернутых экземплярах при использовании автономного динамического алгоритма с текущими практическими подходами к управлению потенциалом горизонтальной эластичности с использованием случайного набора данных RSP#2

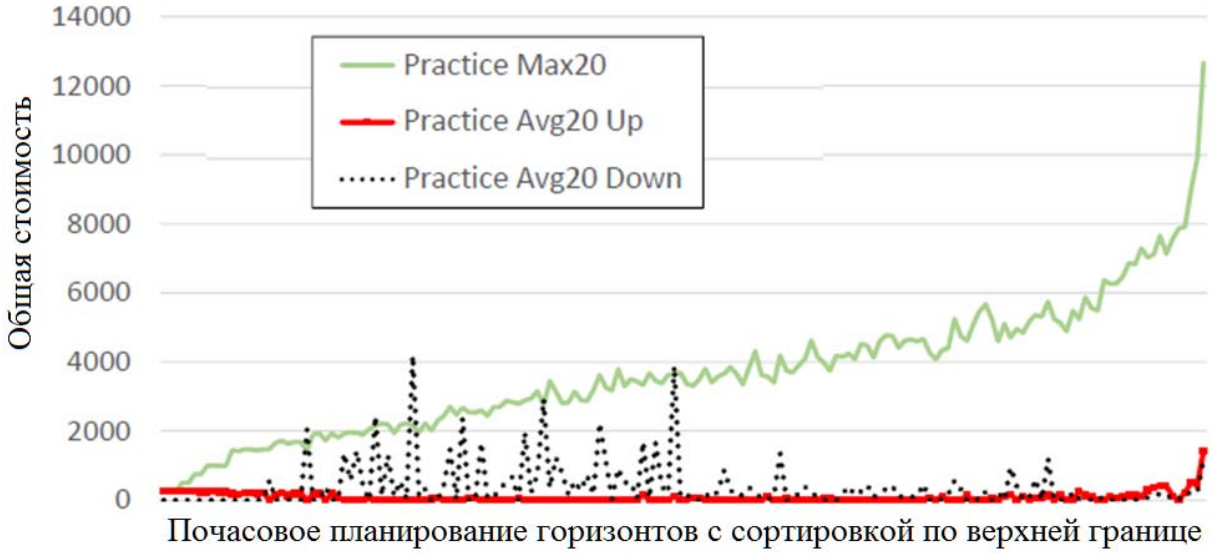


Рисунок 4.6. Сравнение изменения общих затрат при использовании автономного динамического алгоритма с текущими практическими подходами к управлению производительностью по горизонтальной эластичности с использованием случайного набора данных RSP#2

В следующем разделе произведено сравнение подходов для определения количества развернутых экземпляров и их общей стоимости на основе анализа SAA.

4.4.3. Емкость экземпляра SAA

В среднем количество развернутых экземпляров OFD4 на 69,1% меньше по сравнению с типичным значением Max20. В результате OFD более чем на 95% горизонтов планирования рекомендуемое количество развернутых экземпляров сокращается как минимум на 50% по сравнению с Practice Max20. Например, горизонт планирования №157 сократился на 87,5% с Practice Max20 в восемь экземпляров до развертывания OFD в одном экземпляре. По сравнению с Practice Avg20, OFD приводит к сокращению числа развернутых экземпляров на 12,6% и увеличению числа экземпляров на 19,5% за счет подходов Practice Avg20 Up (округление средних экземпляров в большую сторону) и Practice Avg20 Down (округление средних экземпляров в меньшую сторону) соответственно. Графики рекомендуемого количества развернутых экземпляров для всех 168 горизонтов планирования можно увидеть на рисунках 4.3 и 4.5.

Хотя может показаться, что подход к снижению производительности Practice Avg20 лучше справляется с сокращением потребностей в мощности, в следующем разделе показано, что иногда это приводит к чрезмерному сокращению мощности, что приводит к увеличению штрафных санкций.

Практический совет 4. Использование OFD4, описанного в работе, может сократить рекомендуемое количество развернутых экземпляров по крайней мере на 60% по сравнению с Practice Max20 более чем на 81% горизонтов планирования.

4.4.4. Сравнение затрат SAA

Из результатов, приведенных в предыдущем разделе, следует, что практикующим специалистам следует перейти от подхода Practice Max20 к некоторой вариации подхода Practice Avg20, что приводит к гораздо меньшему количеству развертываний экземпляров и более высокому использованию ресурсов. Однако более важным критерием является минимизация общей стоимости развертывания в контексте управления производительностью. Когда анализируются результаты SAA для каждого горизонта планирования, установлено, что OFD4 всегда приводит к решению с наименьшими затратами, как с помощью случайно сгенерированных наборов данных RSP#1, так и с помощью RSP#2 (см. рис. 4.7 и 4.9). Таким образом, несмотря на то, что OFD4 рекомендует большее количество развернутых экземпляров, чем в случае с обычным сокращением на 20%, OFD является лучшим выбором для обеспечения баланса между затратами на развертывание ресурсов и штрафными санкциями. Как показано в таблице 4.7, среднее снижение затрат при использовании OFD4 по сравнению с практическим подходом к снижению на 20% составляет 13,56% для RSP#1 и 18,59% для RSP#2. Кроме того, OFD4 обеспечивает снижение затрат на 4,73% (3,87% для RSP#2) по сравнению с Practice Avg20 Up и позволяет использовать меньше задействованных ресурсов. Наконец, OFD4 обеспечивает значительное снижение затрат на 160,19% для RSP#1 (на 182,23% для RSP#2) по сравнению с наиболее популярным подходом Practice Max20.

Несмотря на то, что улучшение OFD4 по сравнению с Practice Avg20 Up, приведенным в таблице 4.7, относительно невелико, оно принесет значительные преимущества по нескольким причинам. Во-первых, значения затрат в работе уменьшены на константу в целях обеспечения конфиденциальности, при этом сохраняется структура затрат и количество поступающих запросов.

Рисунок 4.7. Структура и функции прототипа программного обеспечения оптимизации облачных сервисов SaaS для решения задач мониторинга

Во-вторых, приведенные общие затраты являются средними по всем 168 горизонтам планирования для одного клиента. Таким образом, эта сумма, умноженная на 168 часов в неделю, умноженная на 52 недели в

году, умноженная на всех клиентов облачного провайдера, составляет существенную сумму. В-третьих, эти значения являются средними по миллионам путей выборки с использованием метода SAA. В разделе 4.3.3 рассмотрен один путь выборки из данных, где подход OFD работает намного лучше по сравнению с практическими подходами.

Разработан прототип программного обеспечения оптимизации облачных сервисов SaaS для решения задач мониторинга (рис. 4.7).

4.5. Выводы по главе 4

Исследование посвящено решению практической задачи, связанной с с поставщиком облачных услуг, чтобы помочь определить, как развернуть мощности системы управления мониторингом на перспективу планирования с использованием модели облачных вычислений SaaS. Это исследование обеспечивает основу для анализа требований к пропускной способности в средах SaaS и потенциально может быть использовано в различных аналогичных контекстах облачных вычислений.

Удовлетворение ожиданий клиентов при одновременном снижении эксплуатационных расходов является общей целью всех поставщиков облачных услуг. В этой среде эффективное управление пропускной способностью важно для провайдеров, чтобы максимизировать их прибыль. Поэтому, поскольку облачные вычисления становятся все более распространенными, необходимо понимать, как принимаются оперативные решения, необходимые для предоставления облачных сервисов с оптимальным соотношением затрат и технологической эффективности.

ЗАКЛЮЧЕНИЕ

В процессе выполнения диссертационного исследования получены следующие основные результаты:

1. Разработана модель статистического мониторинга процессов с большими данными в компьютерных сетях, учитывающая свойства автокорреляции процессов образования и распада части локальных структур на основе контрольных диаграмм.

2. Представлены модифицированные алгоритмы мониторинга производительности коллективов встраиваемых объектов больших программных систем, обеспечивающие возможности иерархического анализа данных и улучшающая интерпретируемость результатов.

3. Создана оптимизационная модель и алгоритмы планирования пропускной способности облачных сервисов SaaS, позволяющая найти компромисс между стоимостью ресурсов и штрафом за задержку выполнения во встраиваемых облачных приложениях "программное обеспечение как услуга" с точки зрения поставщика облачных услуг.

4. Разработан динамический алгоритм определения минимальной верхней границы нулевого штрафа, обеспечивающий получение верхней границы объема вычислительных ресурсов, еще дающей нулевой штраф. Среднее снижение затрат при использовании алгоритма по сравнению с существующим составляет 4,73% и позволяет использовать меньше задействованных ресурсов.

5. Разработана структура программного обеспечения оптимизации облачных сервисов SaaS для решения задач мониторинга, обеспечивающая принятие решений о допустимости конфигурации сервиса с точки зрения штрафов, и также оценку ожидаемого качества обслуживания.

6. Компоненты программного обеспечения зарегистрированы в Роспатенте.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аграновский А.В., Скуратов А.К., Тихонов А.Н., Хади Р.А. Информационная безопасность в RUNNet. Труды XI Всероссийской научно-методической конференции «Телематика 2004», 7-10 июня 2004 г., СПб., том 1, стр.66-68.

2. Амоа Куадио-кан Армел Жеафруа, Сидоренко Е.В., Рындин Н.А. Мониторинг состояния коммуникационных сетей на основе облачных вычислений в режиме реального времени// Моделирование, оптимизация и информационные технологии. 2025;13(1). URL: https://moitvivt.ru/ru/journal/pdf?id=1809 DOI: 10.26102/2310-6018/2025.48.1.014

3. Афанасьев В.Н., Юзбашев М.М. Анализ временных рядов и прогнозирование. М.: Финансы и статистика; 2001, 228 с.

4. Афанасьев К.Е., Домрачев В.Г., Ретинская И.В., Скуратов А.К., Стуколов С.В. Многопроцессорные системы: построение, развитие, обучение / Под ред. Тихонова А.Н. М.: КУДИЦ-ОБРАЗ, 2005. - 224 с.

5. Барашкин Р.Л., Бугай А.И. Программное средство для анализа сетевого трафика. Новые информационные технологии: Тезисы докладов XI Междунар. конф. – М., 2022.

6. Безрукова Е. Г., Руденчик Е. А. Прогнозирование статистических временных рядов. М-во общ. и проф. образования РФ. Яросл. гос. техн. ун-т Ярославль, 1997, 94 с.

7. Бугай А.И., Гугель Ю.В., Калинина Э.В., Ретинская И.В., Скуратов А.К. Применение агрегирующих и разностных операторов для анализа потоков информации в сетях. "Вестник РГРТА", Выпуск №13, Рязань, 2003г. стр.41-46.

8. Варламов О.О., Чжан С., Балдин А.В., Мышенков К.С.,

Сидоренко Е.В. Разработка миварной экспертной системы для планирования ресурсов цеха и анализа отклонений. Моделирование, оптимизация и информационные технологии. 2024; 12(3). https://moitvivt.ru/ru/journal/pdf?id=1641. DOI: 10.26102/2310-6018/2024.46.3.017

9. Городецкий А.Я., Заборовский В.С. Фрактальные процессы в компьютерных сетях. Изд-во СПбГТУ, 2000, 101 с.

10. Домрачев В.Г., Безрукавный Д.С., Калинина Э.В., Ретинская И.В., Скуратов А.К. Нечеткие методы в задачах мониторинга сетевого трафика. Ж. Информационные технологии. №3 2006 с. 2-10

11. Журбенко И.Г. Спектральный анализ временных рядов. 1982, 168 с.

12. Камиль Висам Абдуладим Камиль. Задача оптимизации пропускной способности облака SAAS с учетом штрафных санкций// Сб. тр. VI Всеросс. НПК «Информационные технологии в экономике и управлении». – Махачкала, 2024. С. 83-89.

13. Кильдишев Г.С., Френкель А.А. Анализ временных рядов и прогнозирование. М.: Статистика, 1973 г.

14. Козлов В.А. Открытые информационные системы, Финансы и статистика, 1999, 224 с.

15. Коноплев В.В. Организация центра учета, классификации и мониторинга сетевого трафика: Автореферат диссертации на соискание ученой степени канд. техн. наук : 05.13.11. М., 2002,18 с.

16. Кузнецов С.Е., Халиев В.А. Обзор специализированных статистических пакетов по анализу временных рядов: Науч. Отчет. М.: СтатДиалог, 1993 г.

17. Майнагашев С.М., Попков В.К. Задача о максимальном потоке в нестационарных сетях связи. В сб. Моделирование в информатике и вычислительной технике. Сб. трудов ВЦ СО РАН, 1988, с. 64-69.

18. Мельников Д.А. Информационные процессы в компьютерных сетях. М.: Кудиц-Образ, 1999,256 с.

19. Милославская Н.Г., Толстой А.И. Интрасети: обнаружение вторжений. Юнити, 2001,587 с.

20. Олифер В.Г., Олифер Н.А. Основы сетей передачи данных. ИНТУИТ.ру, 2003, 248 с.

21. Программа интерактивного управления очередью системы мониторинга/ Е.В. Сидоренко, М.В. Кочегаров, В.А.К. Камиль, К.А.Ж. Амоа, О.А. Ющенко. Свидетельство о регистрации программы для ЭВМ № 2025615513 от 12.02.2025. М.: ФИПС, 2025.

22. Сажин Ю. В., Катынь А. В., Басова В. А., Сарайкин Ю. В. Статистические методы прогнозирования на основе временных рядов. Саранск : Изд-во Морд, ун-та, 2000, 113 с.

23. Сидоренко Е.В. Аппаратно-программное обеспечение средств управления большими данными наблюдаемых энергетических систем// Системы управления и информационные технологии, №4(98), 2024. С. 92-98.

24. Сидоренко Е.В. и др. Реализация элементов программно-алгоритмического комплекса оптимизации технологических процессов на базе адаптивных методов// Энергетические системы. – 2019.– № 1. – С. 169-175.

25. Сидоренко Е.В. Моделирование экспоненциального случайного графа для сетей, не зависящих от времени как основа метода оценки производительности// Информационные технологии моделирования и управления, №1(135), 2024. – С. 52-60.

26. Сидоренко Е.В. Проблемы мониторинга сетевой инфраструктуры на основе типовых решений// Информационные технологии моделирования и управления, №4(134), 2023. – С. 311-320

27. Сидоренко Е.В., Рындин Н.А. Результаты влияния

автокорреляций на контрольные диаграммы мониторируемой сети // Наука и технологии: перспективы развития и применения: сб. статей VII Междунар. НПК. - Петрозаводск: МЦНП «НОВАЯ НАУКА», 2024. - С. 23-30.

28. Сидоренко Е.В., Рындин Н.А., Корнеев А.М. Рационализация выбора оптимальной виртуальной машины для реализации специальных запросов в облачной среде// Системы управления и информационные технологии, №4(94), 2023. С. 41-45

29. Скуратов А.К. Алгоритмы анализа и мониторинга телекоммуникационной сети с использованием статистических методов / Вестник УГАТУ, 2005, т.6, №1(12). с.212-226.

30. Скуратов А.К. Мониторинг функционирования телекоммуникационных сетей с использованием методов статистического анализа. Известия Тульского государственного университета. Серия Информационные технологии. Системы управления. Выпуск 1. Вычислительная техника. -Тула: ТулГУ, 2005, с. 113-122.

31. Тихонов А.Н., Скуратов А.К., Домрачев В.Г., Ретинская И.В. Новый этап развития и использования высокопроизводительных вычислительных ресурсов в научно-образовательной среде. Новости искусственного интеллекта (AI News), 4 (52) 2002 г., стр. 35- 45

32. Al-Daoud, H., Al-Azzoni, I. and Down, D.G. (2012) Power-aware linear programming based scheduling for heterogeneous computer clusters. Future Generation Computer Systems, 28(5), 745–754.

33. Al-Kofahi, M. M., Mohammad, Y., Al-Shorman, M. Y., & Al-Kofahi, O. A. (2019). Toward energy efficient microcontrollers and Internet-of-Things systems. Computers & Electrical Engineering, 79, 106457.

34. Altendorfer, K. and Minner, S. (2014) A comparison of make-to-stock and make-to-order in multi-product manufacturing systems with variable due dates. IIE Transactions, 46(3), 197–212.

35. Alwan, L. C. (1992). Effects of autocorrelation on control chart performance// Communications in Statistics, 21, 1025–1049.

36. Alwan, L. C., & Roberts, H. V. (1988). Time-series modeling for statistical process control// Journal of Business & Economic Statistics, 6, 87–95.

37. Arbabian, M.E., Chen, S. and Moinzadeh, K. (2020) Capacity expansions with bundled supplies of attributes: An application to server procurement in cloud computing. Manufacturing & Service Operations Management. https://doi.org/10.1287/msom.2019.0827.

38. August, T., Niculescu, M.F. and Shin, H. (2014) Cloud implications on software network structure and security risks. Information Systems Research, 25(3), 489–510.

39. Aydin, N., Muter, I. and Birbil, S. (2020) Multi-objective temporal bin packing problem: An application in cloud computing. Computers & Operations Research, 121, 1–16.

40. Bendre, M., & Manthalkar, M. (2019). Time series decomposition and predictive analytics using MapReduce framework. Expert Systems with Applications, 116, 108–120.

41. Blackstone, J.H., Phillips, D.T. and Hogg, G.L. (1982) A state-of-the-art survey of dispatching rules for manufacturing job shop operations. The International Journal of Production Research, 20(1), 27–45.

42. Bondu, A., Gay, D., Lemaire, V., Boull´e, M., & Cervenka, E. (2019). FEARS: a FEature And Representation Selection approach for time series classification. In Proceedings of The Eleventh Asian Conference on Machine Learning, PMLR, Nagoya, Japan (pp. 379–394).

43. Brown, D.J. (1979) A lower bound for on-line one-dimensional bin packing algorithms. Coordinated Science Laboratory Report no. R-864 (UILU-ENG 78-2257), Urbana, IL.

44. Caprara, A. and Toth, P. (2001) Lower bounds and algorithms for the 2-dimensional vector packing problem. Discrete Applied Mathematics, 111(3),

231–262.

45. Carroll, A., & Heiser, G. (2010). An analysis of power consumption in a smartphone. In Proceedings of the 2010 USENIX Annual Technical Conference, Boston; US (pp. 271–284).

46. Chen, H., & Liu, H. (2016). A remote electrocardiogram monitoring system with good swiftness and high reliablility. Computers & Electrical Engineering, 53, 191–202. https://doi.org/10.1016/j.compeleceng.2016.02.004

47. Chen, P.Y. and Wu, S.Y. (2012) The impact and implications of ondemand services on market structure. Information Systems Research, 24(3), 750–767.

48. Chen, S., Lee, H. and Moinzadeh, K. (2019) Pricing schemes in cloud computing: Utilization-based vs. reservation-based. Production and Operations Management, 28(1), 82–102.

49. Cheng, H.K., Li, Z. and Naranjo, A. (2016) Research notecloud computing spot pricing dynamics: Latency and limits to arbitrage. Information Systems Research, 27(1), 145–165.

50. Chien, C. F., Chou, C. W., & Yu, H. C. (2016). A novel route selection and resource allocation approach to improve the efficiency of manual material handling system in 200-mm wafer fabs for industry 3.5// IEEE Transactions on Automation Science and Engineering, 13, 1567–1580.

51. Chien, C. F., Hong, T. Y., & Guo, H. Z. (2017). An empirical study for smart production for tft-lcd to empower industry 3.5// Journal of the Chinese Institute of Engineers, 40, 552–561.

52. Chien, C. F., Lin, Y. S., & Lin, S. K. (2020). Deep reinforcement learning for selecting demand forecast models to empower industry 3.5 and an empirical study for a semiconductor component distributor// International Journal of Production Research. https://doi.org/10.1080/00207543.2020.1733125.

53. Choudhary, V. and Zhang, Z. (2015) Research notepatching the

cloud: the impact of saas on patching strategy and the timing of software release. Information Systems Research, 26(4), 845–858.

54. Cohen, M.C., Keller, P.W., Mirrokni, V. and Zadimoghaddam, M. (2019) Overcommitment in cloud services: Bin packing with chance constraints. Management Science, 65(7), 3255–3271.

55. Costa, A., & Fichera, S. (2017). Economic statistical design of ARMA control chart through a Modified Fitness-based Self-Adaptive Differential Evolution. Computers & Industrial Engineering, 105, 174–189.

56. Dieker, A., Ghosh, S. and Squillante, M.S. (2016) Optimal resource capacity management for stochastic networks. Operations Research, 65(1), 221–241.

57. Dong, H., Chen, N., & Wang, K. (2020). Modeling and change detection for countweighted multilayer networks// Technometrics, 62(2), 184–195.

58. Eilon, S. and Hodgson, R. (1967) Job shops scheduling with due dates. The International Journal of Production Research, 6(1), 1–13. IISE TRANSACTIONS 17.

59. Elvers, D.A. (1973) Job shop dispatching rules using various delivery date setting criteria. Production and Inventory Management, 14(4), 62–69.

60. Farahani, E. M., & Kazemzadeh, R. B. (2019). Phase I monitoring of social network with baseline periods using Poisson regression// Communications in Statistics - Theory and Methods, 48, 311–331.

61. Fazli, A., Sayedi, A. and Shulman, J.D. (2018) The effects of autoscaling in cloud computing. Management Science, 64(11), 5149–5163.

62. Fotuhi, H., Amiri, A., & Maleki, M. R. (2018). Phase I monitoring of social networks based on Poisson regression profiles// Quality & Reliability Engineering International, 34, 572–588.

63. Frank, O., & Strauss, D. (1986). Markov graphs// Journal of the American Statistical Association, 81, 832–842.

64. Galante, G. and Bona, L.C.E. (2012) A survey on cloud computing elasticity, in Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing, IEEE, pp. 263–270.

65. Gandhi, A., Gupta, V., Harchol-Balter, M. and Kozuch, M.A. (2010) Optimality analysis of energy-performance trade-off for server farm management. Performance Evaluation, 67(11), 1155–1171.

66. Garey M.R., Johnson D.S. (1979) Computers and Intractability: A guide to the theory of np-completeness (series of books in the mathematical sciences). Freeman, San Francisco.

67. Gartner (2019) Gartner forecasts worldwide public cloud revenue to grow 17.5 percent in 2019. https://www.gartner.com/en/newsroom/press-releases/2019-04-02-gartner-forecasts-worldwide-public-cloud-revenue-to-g (accessed 9 September 2020).

68. Geiger, C.D., Uzsoy, R. and Aytu_g, H. (2006) Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach. Journal of Scheduling, 9(1), 7–34.

69. Gong, Z., Gu, X. and Wilkes, J. (2010) Press: Predictive elastic resource scaling for cloud systems, in Proceedings of the International Conference on Network and Service Management, IEEE Press, pp. 9–16.

70. Gullhav, A.N. and Nygreen, B. (2016) A branch and price approach for deployment of multi-tier software services in clouds. Computers & Operations Research, 75, 12–27.

71. Gullhav, A.N., Cordeau, J.F., Hvattum, L.M. and Nygreen, B. (2017) Adaptive large neighborhood search heuristics for multi-tier service deployment problems in clouds. European Journal of Operational Research, 259(3), 829–846.

72. Guo, Z., Li, J. and Ramesh, R. (2019) Optimal management of virtual infrastructures under flexible cloud service agreements. Information Systems Research, 30(4), 1424–1446.

73. Hall, N.G. (1986) Scheduling problems with generalized due dates. IIE Transactions, 18(2), 220–222.

74. Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., Krivitsky, P. N., & Morris, M. (2018). ergm: Fit, simulate and diagnose exponential-family models for networks// The Statnet Project. R package version, 3(9), 4.

75. Haupt, R. (1989) A survey of priority rule-based scheduling. Operations-Research-Spektrum, 11(1), 3–16.

76. He, Q. P., Wang, J. (2017). Statistical process monitoring in the era of smart manufacturing// 2017 American Control Conference (ACC) (pp. 4797–4802).

77. He, Z., Wang, Z., Tsung, F., & Shang, Y. (2016). A control scheme for autocorrelated bivariate binomial data// Computers & Industrial Engineering, 98, 350–359.

78. Herbst, N.R., Kounev, S. and Reussner, R.H. (2013) Elasticity in cloud computing: What it is, and what it is not, in Proceedings of the 10th International Conference on Autonomic Computing, pp. 23–27.

79. Hills, J., Lines, J., Baranauskas, E., Mapp, J., & Bagnall, A. (2014). Classification of time series by shapelet transformation. Data Mining and Knowledge Discovery, 28, 851–881. https://doi.org/10.1007/s10618-013-0322-1

80. Hmouz, R Al-, Pedrycz, W., & Balamash, A. (2015). Description and prediction of time series: A general framework of Granular Computing. Expert Systems with Applications, 42(1015), 4830–4839. https://doi.org/10.1016/j.eswa.2015.01.060

81. Hosseini, S. S., & Noorossana, R. (2018). Performance evaluation of EWMA and CUSUM control charts to detect anomalies in social networks using average and standard deviation of degree measures// Quality & Reliability Engineering International, 34, 477–500.

82. Hsu, C. Y., Chen, W. J., & Chien, J. C. (2020). Similarity matching

of wafer bin maps for manufacturing intelligence to empower industry 3.5 for semiconductor manufacturing// Computers & Industrial Engineering. 142(3):106358. https://doi.org/10.1016/j.cie.2020.106358.

83. Hunter, D. R. (2007). Curved exponential family models for social networks// Social Networks, 29, 216–230.

84. Hunter, D. R., Handcock, M. S., Butts, C. T., Goodreau, S. M., & Morris, M. (2008). Ergm: A package to fit, simulate and diagnose exponential-family models for networks// Journal of Statistical Software, 24, 1–29.

85. IBM ILOG CPLEX (2016) CPLEX 12.7 user's manual. https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.studio.help/pdf/usrcplex.pdf (accessed 9 September 2020).

86. Iyoob, I., Zarifoglu, E. and Dieker, A. (2013) Cloud computing operations research. Service Science, 5(2), 88–101.

87. Jamrus, T., Wang, H. K., & Chien, C. F. (2020). Dynamic coordinated scheduling for supply chain under uncertain production time to empower smart production for industry 3.5// Computers & Industrial Engineering. https://doi.org/10.1016/j.cie.2020.106375.

88. Jennings, B. and Stadler, R. (2015) Resource management in clouds: Survey and research challenges. Journal of Network and Systems Management, 23(3), 567–619.

89. Johnpaul, C., Prasad, Munaga V. N. K., Nickolas, S., & Gangadharan, G. R. (2020). Trendlets: A novel probabilistic representational structures for clustering the time series data. Expert Systems with Applications, 145. https://doi.org/10.1016/j. eswa.2019.113119

90. Johnson, D.S. (2016) Vector bin packing, in Encyclopedia of Algorithms, Springer, New York, NY, pp. 2319–2323.

91. Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R. and Graham, R.L. (1974) Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM Journal on Computing, 3(4), 299–325.

92. Jula, A., Sundararajan, E. and Othman, Z. (2014) Cloud computing service composition: A systematic literature review. Expert Systems with Applications, 41(8), 3809–3824.

93. Kamalzadeh, H., Ahmadi, A., & Monsour, S. (2017). A shape-based adaptive segmentation of time-series using particle swarm optimization. Information Systems, 67, 1–18.

94. Katircioglu-¨Oztürk, D., Altay Güvenir, H., Ravens, U., & Baykal, N. (2017). A window-based time series feature extraction method. Computers in Biology and Medicine, 89, 466–486. https://doi.org/10.1016/j.compbiomed.2017.08.011

95. Keskinocak, P. and Tayur, S. (2004) Due date management policies. Handbook of Quantitative Supply Chain Analysis, Springer, Boston, MA, pp. 485–554.

96. Khan, J. Y. M. R. (2019). Yuce Internet of things (IoT), systems and applications (1st ed.). New York: Jenny Stanford Publishing.

97. Khedmati, M., & Niaki, S. T. A. (2016). Monitoring simple linear profiles in multistage processes by a MaxEWMA control chart// Computers & Industrial Engineering, 98, 125–143.

98. Kleinrock, L. (1976) Queueing Systems, Volume 2: Computer Applications, Wiley, New York, NY.

99. Kleywegt, A.J., Shapiro, A. and Homem-de Mello, T. (2002) The sample average approximation method for stochastic discrete optimization. SIAM Journal on Optimization, 12(2), 479–502.

100. Komolafe, T., Quevedo, A. V., Sengupta, S., & Woodall, W. H. (2019). Statistical evaluation of spectral methods for anomaly detection in networks// Network Science, 7, 319–352.

101. Kozłowski, A., & Sosnowski, J. (2018). Evaluating energy consumption in wireless sensor networks. Proceedings of SPIE: Photonics Applications in Astronomy, Communications, Industry, and High-Energy

Physics Experiments, 10808, 1739–1748. https://doi.org/ 10.1117/12.2501347

102. Kozłowski, A., & Sosnowski, J. (2019). Energy efficiency trade-off between duty-cycling and wake-up radio techniques in IoT Networks. Wireless Personal Communications, 107, 1951–1971. https://doi.org/10.1007/s11277-019-06368-0

103. Kramer, H.H., Petrucci, V., Subramanian, A. and Uchoa, E. (2012) A column generation approach for power-aware optimization of virtualized heterogeneous server clusters. Computers & Industrial Engineering, 63(3), 652–662.

104. Kravets O.Ja., Atlasov D. I., Gorshkov A.V., Sidorenko E.V., Mutina E.I., Aksenov I.A., Redkin Yu.V. Designing the architecture of a distributed system for information monitoring of IoT and IIoT infrastructures traffic. International Journal on Information Technologies and Security, vol. 16, no. 1, 2024, pp. 49-56. https://doi.org/10.59035/BTBI7690 (WoS).

105. Kreipl, S. and Pinedo, M. (2004) Planning and scheduling in supply chains: An overview of issues in practice. Production and Operations Management, 13(1), 77–92.

106. Krivitsky, P. N., & Goodreau, S. M. (2019). Stergm – separable temporal ergms for modeling discrete relational dynamics with statnet. - http://www.vps.fmvz.usp.br/CRAN/web/packages/tergm/vignettes/STERGM.pdf.

107. Krivitsky, P. N., & Handcock, M. S. (2014). A separable model for dynamic networks// Journal of the Royal Statistical Society: Series B (Statistical Methodology), 76, 29–46.

108. Krivitsky, P. N., & Handcock, M. S. (2018). tergm: Fit, simulate and diagnose models for network evolution based on exponential-family random graph models// The Statnet Project. R package version, 3(5), 2.

109. Krosman, K., & Sosnowski, J. (2019). ESD problems in embedded systems. Proceedings of SPIE: Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments, 11176, 546–

553. https://doi.org/10.1117/12.2536623

110. Ku, C.-C., Chien, C.-F., & Ma, K.-T. (2020). Digital Transformation to Empower Smart Production for Industry 3.5 and An Empirical Study for Textile Dyeing. Computers & Industrial Engineering, 106297. doi:10.1016/j.cie.2020.106297.

111. Kubacki, M., & Sosnowski, J. (2017). Holistic processing and exploring event logs. In A. Romanovsky, & E. Troubitsyna (Eds.), Software Engineering for Resilient Systems. SERENE 2017. Lecture Notes in Computer Science (pp. 184–200 Cham, Springer. https://doi.org/10.1007/978-3-319-65948-0_12.

112. Kubacki, M., & Sosnowski, J. (2019). Exploring operational profiles and anomalies in computer performance logs. Microprocessors and Microsystems, 69, 1–15. https://doi. org/10.1016/j.micpro.2019.05.007

113. Kwok, T. and Mohindra, A. (2008) Resource calculations with constraints, and placement of tenants and instances for multi-tenant SaaS applications, in International Conference on Service-Oriented Computing, Springer, pp. 633–648.

114. Law, A.M. (2015) Simulation Modeling and Analysis, Mcgraw-Hill, New York, NY.

115. Lef_evre, L. and Orgerie, A.C. (2010) Designing and evaluating an energy efficient cloud. The Journal of Supercomputing, 51(3), 352–373.

116. Leifeld, P., Cranmer, S., & Desmarais, B. (2018). Temporal exponential random graph models with btergm: Estimation and bootstrap confidence intervals// Journal of Statistical Software, 83, 1–36.

117. Li, B. and Kumar, S. (2018) Should you kill or embrace your competitor: Cloud service and competition strategy. Production and Operations Management, 27(5), 822–838.

118. Li, G., Yan, W., & Wu, Z. (2019). Discovering shapelets with key points in time series classification. Expert Systems with Applications, 132, 76–

86. https://doi.org/10.1016/ j.eswa.2019.04.062

119. Li, X., Kang, Y., & Li, F. (2020). Forecasting with time series imaging. Expert Systems with Applications, 160. https://doi.org/10.1016/j.eswa.2020.113680

120. Lorido-Botran, T., Miguel-Alonso, J. and Lozano, J.A. (2014) A review of auto-scaling techniques for elastic applications in cloud environments. Journal of Grid Computing, 12(4), 559–592.

121. Lubba, C. H., Sethi, S. S., Knaute, P., Schultz, S. R., Fulcher, B. D., & Jones, N. S. (2019). catch22: CAnonical Time-series CHaracteristics. Data Mining and Knowledge Discovery, 33, 1821–1852. https://doi.org/10.1007/s10618-019-00647-x

122. Maleki, M. R., Amiri, A., & Castagliola, P. (2018). An overview on recent profile monitoring papers (2008–2018) based on conceptual classification scheme// Computers & Industrial Engineering, 126, 705–728.

123. Maragah, H. D., & Woodall, W. H. (1992). The effect of autocorrelation on the retrospective X-chart// Journal of Statistical Computation & Simulation, 40, 29–42.

124. Martinez, B., Mont´on, M., Vilajosana, I., & Prades, J. D. (2015). The Power of Models: Modeling Power Consumption for IoT Devices. IEEE Sensors Journal, 15(10), 5777–5789. https://doi.org/10.1109/JSEN.2015.2445094

125. Mazaheri, V., & Khodadadi, H. (2020). Heart arrhythmia diagnosis based on the combination of morphological, frequency and nonlinear features of ECG signals and metaheuristic feature selection algorithm. Expert Systems with Applications, 161.

126. McCulloh, I., & Carley, K. M. (2011). Detecting change in longitudinal social networks// Journal of Social Structure, 12, 1–37.

127. Megahed, A., Mohamed, M. and Tata, S. (2017) A stochastic optimization approach for cloud elasticity, in Proceedings of the IEEE 10th

International Conference on Cloud Computing, IEEE Press, pp. 456–463.

128. Megahed, F. M., & Jones-Farmer, L. A. (2015). Statistical perspectives on big data// In S. Knoth, & W. Schmid (Eds.). Frontiers in statistical quality control 11. - Cham: Springer International Publishing, pp. 29–47.

129. Mell, P. and Grance, T., (2011) The NIST definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg.

130. Meng, X., Isci, C., Kephart, J., Zhang, L., Bouillet, E. and Pendarakis, D. (2010) Efficient resource provisioning in compute clouds via VM multiplexing, in Proceedings of the 7th international Conference on Autonomic Computing, ACM, pp. 11–20.

131. Montgomery, D. C. (2009). Introduction to Statistical Quality Control. - Hoboken, NJ: John Wiley & Sons.

132. Morris, M., Handcock, M., & Hunter, D. (2008). Specification of exponential-family random graph models: Terms and computational aspects// Journal of Statistical Software, 24, 1–24.

133. Nguyen, T.H. and Wright, M. (2015) Capacity and lead-time management when demand for service is seasonal and lead-time sensitive. European Journal of Operational Research, 247(2), 588–595.

134. Noorossana, R., & Vaghefi, S. J. M. (2006). Effect of autocorrelation on performance of the MCUSUM control chart// Quality & Reliability Engineering International, 22, 191–197.

135. Page, E. S. (1954). Continuous inspection schemes// Biometrika, 41, 100–115.

136. Panigrahy, R., Talwar, K., Uyeda, L. and Wieder, U. (2011) Heuristics for vector bin packing. research.microsoft.com.

137. Park, Y., Priebe, C. E., & Youssef, A. (2013). Anomaly detection in time series of graphs using fusion of graph invariants// IEEE Journal of Selected

Topics in Signal Processing, 7, 67–75.

138. Perry, M. B. (2020). An EWMA control chart for categorical processes with applications to social network monitoring// Journal of Quality Technology, 52, 182–197.

139. Pilastre, B., Boussouf, L., D'Escrivan, St, & Tourneret, J.-Y. (2020). Anomaly detection in mixed telemetry data using a sparse representation and dictionary learning. Signal Processing, 168. https://doi.org/10.1016/j.sigpro.2019.107320

140. Pinedo, M. (2016) Scheduling: Theory, Algorithms, and Systems, Springer, Cham, Switzerland.

141. Priebe, C. E., Conroy, J. M., Marchette, D. J., & Park, Y. (2005). Scan statistics on Enron graphs// Computational & Mathematical Organization Theory, 11, 229–247.

142. Qin, Y., Tang, B., & Mao, Y. (2016). Adaptive signal decomposition based on wavelet ridge and its application. Signal Processing, 120, 480–494. https://doi.org/10.1016/j. sigpro.2015.09.032

143. Raj, S., & Ray, K. Ch (2018). Sparse representation of ECG signals for automated recognition of cardiac arrhythmias. Expert Systems with Applications, 1051, 49–64. https://doi.org/10.1016/j.eswa.2018.03.038

144. Rajagopalan, V., & Ray, A. (2006). Symbolic time series analysis via wavelet-based partitioning. Signal Processing, 86(11), 3309–3320. https://doi.org/10.1016/j. sigpro.2006.01.014

145. Roberts, S. W. (1959). Control chart tests based on geometric moving averages// Technometrics, 1, 239–250.

146. Robins, G., Pattison, P., Kalish, Y., & Lusher, D. (2007). An introduction to exponential random graph (p*) models for social networks// Social Networks, 29, 173–191.

147. Roy, N., Dubey, A. and Gokhale, A. (2011) Efficient autoscaling in the cloud using predictive models for workload forecasting. IEEE International

Conference on Cloud Computing, IEEE Press, pp. 500–507.

148. Ruiz, L. G. B., et al. (2020). A time-series clustering methodology for knowledge extraction in energy consumption data. Expert Systems with Applications, 160.

149. San-Segundo, R., Montero, J. M., Barra-Chicote, R., Fern´andez, F., & Pardo, J. M. (2016). Feature extraction from smartphone inertial signals for human activity segmentation. Signal Processing, 120, 359–372. https://doi.org/10.1016/j. sigpro.2015.09.029

150. Sch¨afer, P., & Leser, U. (2017). Fast and Accurate Time Series Classification with WEASEL. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM ’17) (pp. 637–646). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3132847.3132980.

151. Sedaghat, M., Hernandez-Rodriguez, F. and Elmroth, E. (2013) A virtual machine re-packing approach to the horizontal vs. vertical elasticity trade-off for cloud autoscaling, in Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference, ACM, pp. 1–10.

152. Sengupta, S., & Woodall, W. H. (2018). Discussion of ”statistical methods for network surveillance”// Applied Stochastic Models in Business and Industry, 34, 446–448.

153. Sharma, U., Shenoy, P., Sahu, S. and Shaikh, A. (2011) A cost-aware elasticity provisioning system for the cloud, in Proceedings of the 31st International Conference on Distributed Computing Systems, IEEE Press, Piscataway, NJ, pp. 559–570.

154. Shen, S. and Wang, J. (2014) Stochastic modeling and approaches for managing energy footprints in cloud computing service. Service Science, 6(1), 15–33.

155. Shen, Z., Subbiah, S., Gu, X. and Wilkes, J. (2011) Cloudscale: Elastic resource scaling for multi-tenant cloud systems, in Proceedings of the

2nd ACM Symposium on Cloud Computing, ACM, pp. 1–14.

156. Sidorenko E.V. An overview of methods for evaluating network monitoring performance based on shared time exponential random graph models// Modern informatization problems in the technological and telecommunication systems analysis and synthesis (MIP-2024'AS): Proceedings of the XXIX-th International Open Science Conference. - Yelm, WA, USA: Science Book Publishing House, 2024. Pp. 216-222

157. Sidorenko E.V. et al. Elements realization of software algorithmic system optimization of technological processes based on adaptive methods// IOP Conference series: Materials science and engineering, 2020, 791(1):012034 (Scopus).

158. Sidorenko E.V., Kravets O.Ja. Algorithmization of time series processing as the results of the work of teams of embedded objects of large software systems of the monitoring subsystem// Modern informatization problems in simulation and social technologies (MIP-2025'SCT): Proc. of the XXX-th Int. Open Science Conf. - Yelm, WA, USA: Science Book Publishing House, 2025. – pp. 96-110.

159. Simmon, E. (2018) Evaluation of cloud computing services based on NIST SP 800-145. NIST Special Publication, 500, p. 322.

160. Snijders, T. A. B., Pattison, P. E., Robins, G. L., & Handcock, M. S. (2006). New specifications for exponential random graph models// Sociological Methodology, 36, 99–153.

161. Soleimani, P., Noorossana, R., & Amiri, A. (2009). Simple linear profiles monitoring in the presence of within profile autocorrelation// Computers & Industrial Engineering, 57, 1015–1021.

162. Sperl, R. E., & Chung, S. M. (2019). Two-step anomaly detection for time series data. In 2019 International Conference on Data and Software Engineering (ICoDSE), Pontianak, Indonesia (pp. 1–5). https://doi.org/10.1109/ICoDSE48700.2019.9092751

163. Stillwell, M., Schanzenbach, D., Vivien, F. and Casanova, H. (2010) Resource allocation algorithms for virtualized service hosting platforms. Journal of Parallel and Distributed Computing, 70(9), 962–974.

164. Van der Gaast, J.P., Rietveld, C.A., Gabor, A.F. and Zhang, Y. (2014) A tabu search algorithm for application placement in computer clustering. Computers & Operations Research 50, 38–46.

165. Van, H.N., Tran, F.D. and Menaud, J.M. (2009) SLA-aware virtual resource management for cloud infrastructures, in Proceedings of the ninth IEEE International Conference on Computer and Information Technology, IEEE Press, pp. 357–362.

166. Vanhatalo, E., & Kulahci, M. (2015). The effect of autocorrelation on the Hotelling T2 control chart// Quality & Reliability Engineering International, 31, 1779–1796.

167. Wang, H., Tang, M., Park, Y., & Priebe, C. E. (2014). Locality statistics for anomaly detection in time series of graphs// IEEE Transactions on Signal Processing, 62, 703–717.

168. Wang, Y. H. T., & Huang, W. H. (2017). Phase II monitoring and diagnosis of autocorrelated simple linear profiles// Computers & Industrial Engineering, 112, 57–70.

169. Wang, Z., Lu, Z. and Ye, T. (2016) Multi-neighborhood local search optimization for machine reassignment problem. Computers & Operations Research 68, 16–29.

170. Wilson, J. D., Stevens, N. T., & Woodall, W. H. (2019). Modeling and detecting change in temporal networks via the degree corrected stochastic block model// Quality & Reliability Engineering International, 35, 1363–1378.

171. Woodall, W. H., Zhao, M. J., Paynabar, K., Sparks, R., & Wilson, J. D. (2017). An overview and perspective on social network monitoring// IISE Transactions, 49, 354–365.

172. Xu, F., Liu, Y., Li, Q., & Zhang, Y. (2013). V-edge: Fast Self-

constructive Power Modeling of Smartphones Based on Battery Voltage Dynamics. In 10th USENIX Symposium on Networked Systems Design and Implementation, NSDI (pp. 43–55). Lombard, IL: USENIX Association.

173. Xu, Y., Shi, C. and Duenyas, I. (2016) Priority rules for multi-task duedate scheduling under varying processing costs. Production and Operations Management, 25(12), 2086–2102.

174. Yahyaoui, H., & Al-Daihani, R. (2019). A novel trend based SAX reduction technique for time series. Expert Systems with Applications, 13015, 113–123. https://doi.org/ 10.1016/j.eswa.2019.04.026

175. Yin, S., & Kaynak, O. (2015). Big data for modern industry: Challenges and trends [point of view]// Proceedings of the IEEE (pp. 143–146).

176. Yu, L., Woodall, W. H., & Tsui, K. L. (2018). Detecting node propensity changes in the dynamic degree corrected stochastic block model// Social Networks, 54, 209–227.

177. Zhao, J., & Itti, I. (2016). Decomposing Time Series with application to Temporal Segmentation. In 2016 IEEE Winter Conference on Applications of Computer Vision (WACV). https://doi.org/10.1109/WACV.2016.7477722

178. Zhao, M. J., Driscoll, A. R., Sengupta, S., Fricker, R. D., Jr., Spitzner, D. J., & Woodall, W. H. (2018). Performance evaluation of social network anomaly detection using a moving window-based scan method// Quality & Reliability Engineering International, 34, 1699–1716.

179. Zheng, D., Li, F., & Zhao, T. (2016). Self-adaptive statistical process control for anomaly detection in time series. Systems with Applications, 57, 324–336.

180. Zou, N., & Li, J. (2017). Modeling and change detection of dynamic network data by a network state space model// IISE Transactions, 49, 45–57.