ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Факультет информатики, математики и компьютерных наук**

**Программа подготовки магистров по направлению   
01.04.02 Прикладная математика и информатика**

*Соболев Данил Александрович*

**КУРСОВАЯ РАБОТА**

Прогнозирование финансовых временных рядов

|  |  |
| --- | --- |
|  | Научный руководитель  Кандидат компьютерных наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  С.В. Павлов |

Нижний Новгород, 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc34)

[1. ПРОГНОЗИРОВАНИЕ ФИНАНСОВЫХ РЯДОВ 5](#_Toc35)

[1.1 Определение временного ряда 5](#_Toc36)

[1.2 Требования к исходным данным 6](#_Toc37)

[1.3 Классификация временных рядов 7](#_Toc38)

[1.4 Основные свойства временных рядов 7](#_Toc39)

[1.5 Декомпозиция временного ряда 7](#_Toc40)

[1.6 Типы окон 8](#_Toc41)

[1.7 Типы прогнозов 9](#_Toc42)

[1.8 Простые методы 12](#_Toc43)

[1.8.1 Наивное повторение 12](#_Toc44)

[1.8.2 Скользящая средняя (Simple Moving Average) 12](#_Toc45)

[1.8.3 Экспоненциальная скользящая средняя (EMA) 13](#_Toc46)

[1.8.4 Ансамбль скользящих средних 13](#_Toc47)

[1.9 Статистические модели 14](#_Toc48)

[1.10 Линейные модели 21](#_Toc49)

[1.11 Reccurent Models 22](#_Toc50)

[23](#_Toc51)

[23](#_Toc52)

[23](#_Toc53)

[23](#_Toc54)

[23](#_Toc55)

[23](#_Toc56)

[1.12 Генерация дополнительных признаков временного ряда 24](#_Toc57)

[1.13 VAR 25](#_Toc58)

[1.14 ML 27](#_Toc59)

[1.15 TSAI 33](#_Toc60)

[1.16 FEDOT 34](#_Toc61)

[1.17 Chronos 35](#_Toc62)

[2. Детектирование аномалий и поиск точек перелома в финансовых временных рядах 39](#_Toc63)

[2.1 Генерация аномалий 41](#_Toc64)

[2.2 Статистические методы детектирования аномалий 44](#_Toc65)

[2.3 Прогнозный подход 45](#_Toc66)

[2.4 Isolation Forest 46](#_Toc67)

[2.4 Кластерный подход 46](#_Toc68)

[2.5 Автоэнкодеры 47](#_Toc69)

[2.6 Детектирование многомерных аномалий (PyOD) 48](#_Toc70)

[2.7 Fedot Industrial: библиотека для детектирования аномалий 49](#_Toc71)

[2.8 Детектирование аномалий с помощью LLM 50](#_Toc72)

[3. Оценка новостей 52](#_Toc73)

[4. Разработка комбинированного подхода 53](#_Toc74)

[5. Результаты 53](#_Toc75)

[1.2 Математическая модель задачи о размещениях на плоскости 53](#_Toc76)

[1.3 Подходы для решения задачи о размещениях 54](#_Toc77)

[2. РАЗРАБОТКА КОМБИНАТОРНОГО ПОДХОДА ДЛЯ ПРИБЛИЖЕННОГО РЕШЕНИЯ ЗАДАЧИ О РАЗМЕЩЕНИЯХ 56](#_Toc78)

[2.1 Постановка задачи 56](#_Toc79)

[2.2 Решение для двух объектов 57](#_Toc80)

[2.3 Рассмотрение общего случая 59](#_Toc81)

[2.4 Реализация построения контура прямоугольников при помощи дерева отрезков 61](#_Toc82)

[2.5 Оптимизация алгоритма для построения контура с помощью дерева отрезков 68](#_Toc83)

[2.6 Комбинаторное построение контура с помощью алгоритма плоского заметания 69](#_Toc84)

[2.7 Поиск точек размещения с использованием генетических алгоритмов. 72](#_Toc85)

[3. ПРИБЛИЖЕННОЕ РЕШЕНИЕ ЗАДАЧИ О РАЗМЕЩЕНИЯХ С ПОМОЩЬЮ ЦЛП 76](#_Toc86)

[3.1 Описание подхода 76](#_Toc87)

[3.2 Формулировка модели 77](#_Toc88)

[4. ЭКСПЕРИМЕНТЫ 79](#_Toc89)

[4.1 Применение методологии к задаче размещения двух объектов. 79](#_Toc90)

[4.2 Применение методологии к задаче с большим числом размещаемых объектов 80](#_Toc91)

[ЗАКЛЮЧЕНИЕ 83](#_Toc92)

[СПИСОК ЛИТЕРАТУРЫ 84](#_Toc93)

[ПРИЛОЖЕНИЕ 86](#_Toc94)

# ВВЕДЕНИЕ

Прогнозирование временных рядов является важной и актуальной задачей в различных областях, таких как продажи, техника, производство, медицина и финансы. В современной финансовой теории точность прогнозов напрямую влияет на эффективность принимаемых инвестиционных решений. Корректный прогноз позволяет минимизировать риски, связанные с волатильностью активов, а также оптимизировать стратегию инвестирования. Временные ряды, отражающие динамику цен акций, облигаций, валют и других финансовых инструментов, характеризуются сложной структурой, включающей тренды, сезонность, циклические колебания и случайные флуктуации, что делает задачу прогнозирования особенно нетривиальной и привлекает внимание исследователей со всего мира.

На протяжении нескольких десятилетий исследователи постепенно расширяли арсенал методов для решения этой задачи. Сначала применялись эвристические и наивные модели, затем появились статистические и линейные подходы, позволившие формализовать выявляемые зависимости. С развитием вычислительных мощностей и алгоритмов стали использоваться методы машинного обучения, а затем – рекуррентные нейронные сети, способные учитывать нелинейные зависимости во временных рядах. В свою очередь, глубокие модели, такие как LSTM и трансформеры, продемонстрировали высокую эффективность при выявлении сложных временных паттернов. Параллельно развивались подходы компьютерного зрения, позволяющие анализировать изображения графиков и извлекать статистические закономерности из визуальных паттернов. Каждый новый класс методов открывал дополнительные возможности для повышения точности и устойчивости прогнозов.

Однако в последнее время всё более заметным направлением становится применение больших языковых моделей (Large Language Models, LLM). В отличие от традиционных нейросетевых архитектур, LLM объединяют в себе знания, полученные из обширных текстовых корпусов, и умеют работать с различными типами данных. В промпт можно передать числовой временной ряд и его характеристики, информацию о компании и её финансовые показатели (мультипликаторы, технические индикаторы, отчёты), изображения с графиком цены и отмеченными трейдерами паттернами, а также новостной и социальный контекст. Это даёт LLM возможность учитывать всю информацию, доступную обычному инвестору, и обнаруживать собственные закономерности на её основе. В исследованиях [1]–[7] демонстрируется, как большие языковые модели применяются к различным аспектам прогнозирования финансовых временных рядов.

В рамках данной работы будет проведён сравнительный анализ популярных решений и готовых библиотек для прогнозирования временных рядов. Особое внимание уделяется как одномерным, так и многомерным моделям: от классических статистических алгоритмов до современных нейросетевых и LLM-ориентированных подходов. Для оценки качества прогнозов будут использованы следующие метрики: основная — MAPE (Mean Absolute Percentage Error), дополнительная — RMSE (Root Mean Squared Error). В качестве исходных данных рассматриваются исторические котировки крупных компаний, торгуемых на Московской бирже, а также новостные потоки из открытых источников. Результаты исследования могут быть полезными инструментами для трейдеров и при формировании финансовых стратегий.

Целью данной курсовой работы является демонстрация потенциала современных методов прогнозирования временных рядов с акцентом на большие языковые модели. Ключевая задача исследования — провести эксперимент по прогнозированию временного ряда с помощью LLM с учётом мультимодальной информации, включая выявленные аномалии, новостной контекст и изображения графиков. Таким образом, работа направлена на выявление сильных и слабых сторон разных подходов.

# ПРОГНОЗИРОВАНИЕ ФИНАНСОВЫХ РЯДОВ

## 1.1 Определение временного ряда

Временной ряд – это последовательность значений некоторого показателя, упорядоченных по времени. Каждое значение называет­ся уровнем (отсчетом) ряда, и каждому уровню обязательно соответствует момент времени (или порядковый номер). В отличие от простой выборки, где анализируется только «статическая» статистика (среднее, дисперсия, распределение), при работе с временными рядами учитывается зависимость между соседними измерениями.

В финансовой сфере классическим примером является биржевой курс акции или индекса — например, ежедневная цена закрытия акций «Газпром» на Московской бирже. При анализе таких данных задача часто состоит в определении долгосрочной тенденции (тренда), а также краткосрочных колебаний, чтобы выстроить инвестиционную стратегию или оценить риск в будущие периоды.

## 1.2 Требования к исходным данным

Для получения объективного прогноза временной ряд должен удовлетворять базовым условиям:

1. Равномерный шаг во времени
   1. Все наблюдения должны фиксироваться через одинаковые интервалы (например, ежедневно, еженедельно или ежемесячно)
   2. На финансовых рынках нередко встречаются пропуски — выходные, праздничные или периоды приостановки торгов. В таких случаях нужно решить, будет ли ряд «разреженным» (пропуски оставляем) или заполняем отсутствующие даты (например, приравнивая цену к предыдущему торговому дню).
2. Фиксация в одно и то же время каждого интервала
   1. Если берётся дневная цена закрытия, важно, чтобы она соответствовала именно моменту закрытия торгов. Нельзя, чтобы одни значения были ценами открытия, а другие — средневзвешенными.
3. Полнота исходного ряда
   1. Краткие пропуски (1–2 торговых дня) можно восполнить интерполяцией или переносом предыдущего значения, но длительные «провалы» (например, из-за технических сбоев) требуют более тщательной обработки.

## 1.3 Классификация временных рядов

1. Univariate (одномерный) временной ряд — содержит значения только одного показателя (например, цена закрытия акции «Сбербанк» каждый день).
2. Multivariate (многомерный) временной ряд — одновременно фиксируются несколько показателей, связанных между собой (например, цена и объём торгов той же акции).

## 1.4 Основные свойства временных рядов

При выборе и валидации методов прогнозирования важно учитывать ключевые статистические свойства ряда:

1. Стационарность - свойство ряда, при котором его статистические характеристики (среднее, дисперсия, автокорреляционная функция) не меняются во времени. В строго стационарном ряду распределение любых наборов значений для произвольных моментов времени одинаково, в слабо стационарном сохраняются постоянными только первые два момента: математическое ожидание и ковариация.
2. Автокорреляция - наличие зависимости между значениями ряда на разных лагах (интервалах). Если значения временного ряда непосредственно связаны друг с другом (то есть текущее значение зависит от предыдущих), говорят о наличии автокорреляции. Именно это свойство позволяет многим моделям учитывать «эффект памяти» в данных и строить более точные прогнозы

## 1.5 Декомпозиция временного ряда

Для анализа и построения прогноза временной ряд традиционно представляют как сумму нескольких компонент:

1. Тренд (Trend) - это долгосрочная тенденция (направление) изменения уровня временного ряда. Тренд отражает систематическую (постоянно возрастающую или убывающую) составляющую процесса без учёта сезонных, циклических или случайных колебаний.
2. Цикличность (Cycle) - относительно длительные колебания вокруг тренда, связанные с экономическими, технологическими или другими макро­процессами. В отличие от сезонности, циклы не обязательно повторяются с постоянной длиной (периодом), но обычно охватывают несколько кварталов или лет (например, бизнес-циклы).
3. Сезонность (Seasonality) - это регулярные, периодические колебания уровня временного ряда, связанные с повторяющимися явлениями внутри фиксированного календарного периода (например, квартальные или годовые циклы). Сезонные колебания возникают из-за циклически повторяющихся факторов (праздников, климатических условий, потребительских сезонов и т. д.) и проявляются в виде одинаковых паттернов через каждый сезонный интервал (например, увеличение продаж перед Новым годом или повышение потребления электроэнергии летом).
4. Остатки (Noise/Residual) - это случайные флуктуации и нерегулярные отклонения, не объясняемые ни трендом, ни сезонными/циклическими колебаниями. Остатки характеризуются отсутствием систематической структуры и обычно рассматриваются как «шум», который моделям прогнозирования необходимо учитывать либо отфильтровывать.

## 1.6 Типы окон

Скользящее окно (sliding window) - при этом подходе фиксируется размер окна 𝑤 (например, 𝑤 дней). Для каждой точки прогноза модель обучается (или пересчитывается) на последних 𝑤 наблюдениях:

* Исторический ряд .
* Чтобы спрогнозировать значение в момент 𝑛+1, используем данные .
* Для получения прогноза в момент 𝑛+2 «окно» сдвигается на единицу: (при условии, что прогноз уже известен).

Метод применяется если ряд потенциально не стационарен на длительных отрезках, и старые данные могут уже не отражать современные закономерности. А также когда необходимо «фиксированное» число наблюдений для обучения.

Расширяющееся окно (expanding window) - В этом случае окно «растёт» со временем — на этапе прогноза модель обучается на всех доступных наблюдениях :

* Чтобы спрогнозировать берём и так далее. При расширяющемся окне «никакие» точки не удаляются: каждую новую точку сразу добавляем в обучающую выборку.

Метод применяется если предполагается, что более ранние данные влияют на текущую динамику (ряд относительно стационарен, и имеющиеся наблюдения вносят вклад в прогноз). При наличии скользящих моделей с автоматическим регуляризатором (например, экспоненциальное сглаживание) или когда объём исторических данных не столь велик, чтобы их хранить неэффективно.

## 1.7 Типы прогнозов

После того как мы определились с «окном» данных, встает вопрос, как именно извлечь из них прогноз. Существует два основных подхода: прогноз «точка за точкой» (rolling forecast или one-step-ahead) и мультишаговый прогноз (multi-step-ahead forecast).

Прогноз «точка за точкой» (one-step-ahead forecast) - суть «точечного» прогноза состоит в том, что на каждом временном шаге мы пытаемся спрогнозировать лишь одно следующее значение ряда. Формально это выглядит так: то есть мы обучаем модель на исторических значениях до момента 𝑡 и предсказываем только . Когда фактическое наблюдение становится доступным (утро следующего торгового дня), добавляем его в обучающую выборку («скользящее» или «расширяющееся» окно) и строим прогноз . Процесс повторяется снова и снова: ежедневно модель «скользит» вместе с временным рядом, а её качество измеряется по отклонению прогноза от реального .

Преимущество этого подхода в том, что мы всегда корректируем модель с учётом последнего полученного факта, а значит, прогноз соответствует реальным данным на каждом шаге. Это считается наиболее «честным» методом оценки стабильности модели, поскольку нет накопления ошибок: на каждом новом шаге мы пользуемся лишь истинными наблюдениями, а не предсказанными значениями.

Мультишаговый прогноз (multi-step-ahead forecast) - иногда требуется спрогнозировать сразу несколько шагов вперёд, скажем, на месяц вперёд (где один торговый месяц — примерно 20 дней). Вместо того чтобы ежедневно «скользить» вперёд и запрашивать новые реальные данные, модель генерирует целую цепочку прогнозов: , где ℎ — горизонт прогнозирования (например, ℎ = 20). Существует два основных способа построения мультишагового прогноза.

1. Рекурсивный (autonomous, recursive) способ - поочерёдно прогнозируем каждую следующую точку, используя предыдущие предсказания. Сначала на базе строим прогноз , затем «подставляем» этот прогноз в обучающую последовательность: и предсказываем . Затем — добавляем и так продолжаем до . Проблема этого способа в том, что ошибка надвигается: если на первом шаге модель допустила небольшую погрешность, то при прогнозе второго шага она опирается уже не на чистые данные, а на слегка «испорченный» прогноз, и ошибка нарастает экспоненциально при увеличении ℎ.

2. Одновременный (direct) способ - вместо одной модели, которая рекурсивно «тащит» себя вперёд, строим либо ℎ отдельных моделей, либо одну модель, выводящую целый вектор из ℎ прогнозов. То есть первая подмодель предсказывает на основе , вторая отдельно обучается предсказывать по тем же , третья — и т. д. В результате у нас сразу получается набор прогнозов для всех будущих шагов без рекурсивного накопления ошибки. Минус (помимо сложности обучения сразу ℎ моделей) заключается в том, что каждая из них игнорирует промежуточную информацию: например, модель, предсказывающая , не использует ни реальных, ни смоделированных значений и . В итоге на дальних горизонтах точность может оказаться ниже, чем для рекурсивного подхода, уже потому, что каждая «прямая» модель имеет слишком большие лаги между входными данными и целевым моментом.

Очевидно, что мультишаговый прогноз требуется, когда важно получить сразу серию значений, а не «точку за точкой». Но именно потому, что при каждом подходе (рекурсивном или прямом) накапливаются ошибки либо расходятся зависимости, точность на удалённых горизонтах обычно невысока и нередко хуже, чем у последовательного прогноза «точка за точкой». Как правило, в финансовых задачах, где требуется оценить лишь следующий шаг, выбирают «one-step-ahead». А когда необходимо заложить серию прогнозов — например, для построения сценарного анализа на месяц вперёд — используют multi-step с осторожностью, обязательно проверяя, как далеко «устойчив» прогноз.

1.7 Метрики

## 1.8 Простые методы

Рассмотрены простейшие техники прогнозирования, которые позволяют получить «бенчмарк» для последующего сравнения с более сложными моделями.

### 1.8.1 Наивное повторение

Наивное повторение — самый простой метод прогнозирования временного ряда. Суть его заключается в том, что прогноз на следующий шаг принимается равным последнему известному наблюдению. Формально это записывается так: . То есть если сегодня цена акции равна , то на завтра мы предполагаем те же , не пытаясь учесть какие-либо закономерности.

Этот подход не требует обучения и каких-либо параметров: алгоритм «запоминает» лишь последнее значение и сразу выдает его в качестве прогноза. Несмотря на кажущуюся наивность, метод хорошо работает в ситуациях, когда динамика меняется очень медленно и последних наблюдений достаточно, чтобы «угадать» следующий шаг (например, при относительно спокойном рынке без резких скачков).

Главная роль наивного повторения — выступать в качестве базового уровня для сравнения.

### 1.8.2 Скользящая средняя (Simple Moving Average)

Скользящая средняя (SMA) прогнозирует следующий шаг как среднее последних w наблюдений, сглаживая краткосрочные флуктуации и устраняя высокочастотный шум. Это позволяет выявлять тренды: если текущая цена выше SMA, говорят о восходящем движении, ниже — о нисходящем.

.

Метод не требует обучения: при появлении нового окно смещается, старейшее наблюдение удаляется, и прогноз обновляется тем же выражением. Преимущество SMA — простота и наглядность, недостаток — равная «весовая» нагрузка на все w точек, из-за чего прогноз запаздывает и учитывает устаревшие данные.

### 1.8.3 Экспоненциальная скользящая средняя (EMA)

Экспоненциальная скользящая средняя (EMA) взвешивает более свежие наблюдения сильнее, чем старые, позволяя быстро реагировать на изменения данных. Вместо равных весов, как в SMA, здесь вес α придаётся самому последнему значению, а вес предыдущих точек убывает экспоненциально.

Преимущество EMA заключается в том, что она быстрее реагирует на резкие изменения по сравнению с SMA: вес последнего наблюдения всегда равен α, а влияние через k шагов составляет . Однако хотя старые данные «затухают» экспоненциально, они всё же остаются в расчёте, что может приводить к небольшому запаздыванию в экстремальных ситуациях и требует подбора параметра α.

### 1.8.4 Ансамбль скользящих средних

Ансамбль скользящих средних объединяет несколько EMA (или SMA) с разными «окнами» для получения более устойчивого прогноза, учитывая одновременно краткосрочные, среднесрочные и долгосрочные колебания ряда.

,

​

Преимущество ансамбля в том, что он сочетает быстрое реагирование короткой EMA, сбалансированность средней и плавность длинной, сглаживая шумы одной компоненты и уменьшая запаздывание другой. Недостаток — немного увеличенная вычислительная нагрузка и необходимость выбора весов , , , но в целом метод остаётся простым и даёт качественный бенчмарк перед более сложными моделями..

1.8.5 Метод дрейфа

Метод дрейфа (Drift) предполагает, что временной ряд «скользит» с постоянным средним темпом изменения: прогноз для (n+1)-го момента равен последнему наблюдению плюс средний «шаг» ​. Таким образом учитывается линейная тенденция за весь период.

Достоинство метода—простейшее учёт общего наклона ряда без обучения; он даёт более адекватный прогноз, чем «наивное» повторение, если динамика близка к линейной. Недостаток—плохо справляется при резких изменениях или нелинейных паттернах, поскольку сохраняет постоянный темп независимо от новых колебаний.

## 1.9 Статистические модели

Для построения адекватных прогнозов временных рядов часто опираются на классические статистические модели (такие как ARIMA, SARIMA, ETS и другие), которые делают ключевым допущением стационарность. Если это условие не соблюдается, параметры модели, оценённые на «старых» данных, перестают быть релевантными для «новых», а прогноз утрачивает устойчивость.

Прежде чем приступать к каким-либо преобразованиям, необходимо проверить стационарность самого ряда. Для этого применяется тест Augmented Dickey–Fuller (ADF). Суть ADF-теста в проверке нулевой гипотезы о наличии единичного корня (то есть о нестационарности ряда). В ходе теста оценивается регрессия вида , где , а p — число добавочных лагов, достаточное для устранения автокорреляции остатков. Если оценённый коэффициент γ оказывается статистически существенно меньше нуля (p-value ниже выбранного уровня значимости, обычно 0,05), мы отвергаем гипотезу о единичном корне и считаем ряд стационарным. В противном случае ряд нужно приводить к стационарности.

Переход к стационарному ряду обычно осуществляется несколькими способами. В первую очередь, временной ряд разбивают на компоненты: тренд, сезонность и остатки. Удаляя из ряда детерминированный тренд (линейный или нелинейный) и/или сезонную составляющую, мы получаем так называемый «остаточный» ряд, в котором уже не наблюдается явно выраженной систематической тенденции или регулярных колебаний. На практике тренд часто убирают с помощью полиномиальной регрессии или скользящего среднего, а сезонность — с помощью классического сезонного декомпозирования (например, метода STL или классического аддитивного/мультипликативного разложения).

Следующий шаг — стабилизация дисперсии. Дисперсия финансовых рядов может быть неустойчивой (гетероскедастичной): периоды «штиль» сменяются резкими всплесками волатильности. Прежде чем приступать к оценке параметров модели, логично «привести» дисперсию к более равномерному виду. Одним из распространённых способов является преобразование Бокса–Кокса, которое задаётся формулой

Правильно подобранная степень λ (обычно оценивается методом максимального правдоподобия) сглаживает сильные колебания и выравнивает дисперсию ряда. После применения этого преобразования «данные становятся ближе к нормальному распределению и более пригодны для линейных моделей».

Если разложение на тренд и сезонность не полностью устраняет нестационарность или если дисперсия после преобразования всё ещё изменяется, применяют разностное преобразование (differencing). На первом порядке разностной разностью называется ряд . Если после первой разности тренд ещё не устранён, допускают вторую разность ,

и так далее. Цель — добиться такого порядка разностей d, при котором полученный ряд будет примерно стационарным. Для сезонных рядов иногда вводят сезонную разность , где s — длина сезонного периода (например, 12 для месячных данных). После каждой итерации разностного преобразования повторяют ADF-тест, чтобы убедиться, что ряд действительно стал стационарным.

Кроме перечисленных преобразований, в финансовых задачах традиционно используют ряд логарифмических доходностей вместо самого ценового ряда. Логарифмическая доходность определяется как , где — цена актива в момент t. Лог-доходности обычно оказываются гораздо более стационарными: их математическое ожидание и дисперсия остаются сравнительно постоянными, а распределение становится более «колоколоподобным». Поэтому вместо моделирования цен часто строят модели на лог-доходностях , прогнозируют будущие , а затем для получения прогноза цены используют обратное преобразование: . Таким образом, сохраняется экономический смысл (мы прогнозируем именно доходности), и при этом автоматически решается частично или полностью проблема нестабильности дисперсии.

В итоге, приведение ряда к стационарному виду сводится к следующему алгоритму:

* Выполнить ADF-тест на исходном ряду .
* Если ряд нестационарен, удалить тренд и сезонность, получив остаточный ряд.
* Применить преобразование Бокса–Кокса для стабилизации дисперсии.
* Провести разностное преобразование необходимого порядка (в том числе сезонные разности), проверяя после каждого шага ADF-тестом.
* Либо сразу перейти к лог-доходностям {}, если такой ряд стационарен.

Только после того, как ряд приведён к приближённо стационарному виду, можно приступать к оценке параметров статистических моделей. Без предварительной обработки, обеспечивающей стационарность, эти методы либо не сойдутся, либо дадут неточные и не устойчивые прогнозы.

1.9.1 ARIMA

ARIMA (Autoregressive Integrated Moving Average) представляет собой классическую модель для прогнозирования временных рядов, объединяющую авторегрессию, интегрирование (разностное преобразование) и скользящее среднее. Основная идея состоит в том, что если исходный ряд не является стационарным, то его приводят к стационарному виду с помощью d-кратных разностей:

Далее стационарный ряд {} описывают как ARMA(p,q)

.

Эквивалентно на уровне это записывают через оператор задержки B в виде

Перед оценкой параметров с помощью максимального правдоподобия или Conditional Sum of Squares сначала определяют d, повторяя тест Augmented Dickey–Fuller для ряда ​ и т. д., пока ряд не станет стационарным. После этого анализ ACF и PACF стационарного предлагает кандидатов для p и q. Комбинации (p,q) сравнивают по информационным критериям и выбирают модель с наименьшим AIC (BIC). Оценённые параметры проверяют на стационарность AR-полинома (корни вне единичного круга) и обратимость MA-полинома.

Для одношагового прогноза по ряду длины T сначала строится прогноз ​ по форме ARMA, затем восстанавливают через обратное интегрирование, например при d = 1 таким образом: . Для дальнейших шагов прогноз рекурсивно использует ранее спрогнозированные значения и остатки.

Главное преимущество ARIMA заключается в гибкости и наличии статистической теоретической базы: при правильно выбранных (p,d,q) она дает интерпретируемые прогнозы с доверительными интервалами. Однако обязательное требование стационарности ограничивает её способность улавливать сложные нелинейные зависимости, а чувствительность к выбросам повышает риск искажений; при яркой сезонности часто приходится переходить на SARIMA.

1.9.2 AUTO-ARIMA

Auto-ARIMA представляет собой автоматизированный алгоритм подбора оптимальных порядков (p,d,q) (и при необходимости сезонных (P,D,Q,s)) для модели ARIMA без ручной идентификации по ACF/PACF. В основе алгоритма лежит последовательное применение теста Augmented Dickey–Fuller для определения минимального d, выбор сезонного порядка D при наличии регулярных циклов и перебор комбинаций (p,q) (и (P,Q) для SARIMA) с оценкой информационных критериев.

Auto-ARIMA экономит время и снижает риск ошибок при выборе параметров, особенно если ряд содержит сложную сезонную структуру: алгоритм автоматически определяет d и D через ADF-тесты, а затем, используя «шаговый» перебор, находит (p,q) (и при необходимости (P,Q)) по минимизации AIC или BIC. Однако при очень длинных рядах или множественных сезонных периодах перебор может стать вычислительно затратным, а оптимизация по AIC/BIC не всегда улавливает тонкие нелинейные зависимости в финансовых данных, требующие ручной донастройки. Несмотря на это, Auto-ARIMA остаётся удобным «базовым» решением для быстрого получения рабочей ARIMA-модели.

1.9.3 TBATS

TBATS (Trigonometric seasonality, Box–Cox transformation, ARMA errors, Trend, and Seasonal components) предназначена для временных рядов с несколькими сезонными циклами и нестабильной дисперсией. Основная идея состоит в том, чтобы описывать каждую сезонность через гармоники, автоматически находить оптимальный параметр Бокса–Кокса для стабилизации дисперсии, моделировать локальный тренд, а незавершённые остатки учитывать через ARMA–процесс.

Сезонная компонента при периоде m задаётся как сумма тригонометрических гармоник::

где m – длина сезонного периода; и оценки амплитуд j-ой гармоники; число учитываемых гармоник .

Перед этим весь ряд трансформируется преобразованием Бокса–Кокса для выравнивания дисперсии:

Параметр λ в TBATS оценивается по максимальному правдоподобию, чтобы остатки были максимально гомоскедастичными.

Объединяя компоненты, модель формулируется как сумма локального линейного тренда (с адаптивной скоростью изменения ), сезонных гармоник и ARMA–ошибок:

α, β — коэффициенты сглаживания тренда, а «ARMA-errors» обозначает остаточную компоненту, смоделированную как ARMA(p,q).

TBATS подходит для рядов с несколькими и даже нецелочисленными сезонными периодами (годовой, квартальный, недельный), когда классические SARIMA–модели не справляются. Благодаря встроенной оптимизации λ преобразование дисперсии становится более гибким, а тригонометрические гармоники позволяют эффективно моделировать сложные циклы без крупных матриц индикаторов. Недостатком являются относительно высокая вычислительная нагрузка и сложность настройки при большом числе сезонностей, а также потребность в достаточном объёме данных для точной оценки всех гармоник и ARMA–компоненты.

## 1.10 Линейные модели

Линейные модели служат базовым инструментом для прогнозирования временных рядов, поскольку они просты в настройке, вычислительно эффективны и легко интерпретируются. В основе любого линейного метода лежит предположение о том, что будущее значение ряда можно представить в виде линейной комбинации заранее выбранных признаков, наиболее часто включающих предыдущие наблюдения.

1.10.1 DLineral

В модели DLinear (Decomposed Linear) временной ряд предварительно разбивается на детерминированные компоненты — тренд и сезонность — и на остатки, после чего остатки прогнозируются линейной регрессией по лагам. Такой подход позволяет отделить влияние крупных циклических колебаний и оставить линейной модели только «чистые» отклонения.

DLinear хорошо подходит для рядов с ярко выраженным трендом и сезонностью, поскольку предварительное вычитание этих компонент облегчает задачу прогнозирования остатка и улучшает точность линейной части. Однако эффективность метода снижается, если декомпозиция компонентов неточна или если сезонность и тренд взаимодействуют с остатками сложным образом.

1.10.2 NLineral

Модель NLinear (Nonlinear Linear) сохраняет линейную зависимость в параметрах, но расширяет пространство признаков за счет включения полиномиальных и взаимодействующих комбинаций исходных лагов. Это позволяет линейному алгоритму аппроксимировать более сложные, «сглаженные» нелинейные зависимости.

NLinear эффективна, когда зависимость ряда от прошлого неуклонно линейная, но имеет полиномиальные или смешанные эффекты. Преимуществами являются сохранение простоты линейной регрессии и гибкость за счет расширенного набора признаков; основным ограничением становится быстрое увеличение числа параметров, что требует жёсткой регуляризации или отбора признаков, чтобы избежать переобучения.

## 1.11 Reccurent Models

Рекуррентные нейронные сети (RNN) и их модификация LSTM предназначены для работы с последовательными данными, где текущее состояние зависит от всех предыдущих. В отличие от классических моделей, они хранят «память» о прошлых значениях, что особенно важно при моделировании временных рядов, демонстрирующих долгосрочные зависимости.

1.11.1 RNN

В стандартной RNN на каждом шаге t вычисляется скрытое состояние как функция текущего входа ​ и предыдущего состояния .

где σ — элемент-wise нелинейность (например, tanh или ReLU), а , , , , — обучаемые параметры. Выход может быть усреднён по всем шагам или браться только финальное значение .

Такая архитектура позволяет сети аккумулировать информацию из всех предыдущих наблюдений, но при этом подвержена проблемам затухающих и взрывающихся градиентов, из-за чего обучение долгосрочных зависимостей становится затруднительным.

1.11.2 LSTM

LSTM (Long Short-Term Memory) решает проблему RNN, вводя специализированные гейты для управления потоком информации и удержания важных сигналов на длительных интервалах. Каждый LSTM-блок содержит входной (i), забыания () и выходной () гейты, а также ячейку состояния .

## 

## 

## 

## 

## 

## 

где ⊙ означает поэлементное произведение. Выход формируется аналогично RNN: .

Благодаря ячейке состояния и гейтам LSTM способен эффективно запоминать или забывать информацию, что делает его предпочтительным для временных рядов с длительными зависимостями. Основные преимущества LSTM — устойчивость к затуханию градиентов и гибкое управление памятью; к недостаткам относятся высокие вычислительные затраты и большое число параметров, что требует аккуратного подбора архитектуры и регуляризации.

## 1.12 Генерация дополнительных признаков временного ряда

В исходном виде финансовые данные обычно представляют собой последовательность «свечей» (OHLCV): для каждого момента времени фиксируются цена открытия (Open), максимальная (High) и минимальная (Low) цены, цена закрытия (Close) и объём торгов (Volume). Однако работа исключительно с этими пятью параметрами часто не позволяет учесть сложную динамику рынка: спорадические всплески ликвидности, изменение волатильности, краткосрочные импульсы и долгосрочные тренды могут оставаться незамеченными. Для получения более информативного описания поведения ряда моделям требуются дополнительные признаки, которые извлекают статистические и экономические закономерности из «сырых» данных. Такой процесс называется feature engineering и превращает одномерный ряд цен в многомерный набор, способный уловить нелинейности и многомасштабные эффекты.

1.12.1 Технические индикаторы

Технические индикаторы представляют собой заранее определённые функции от цен и объёмов, призванные выявлять различные аспекты динамики рынка. Сглаживающие индикаторы, такие как простая и экспоненциальная скользящие средние (SMA и EMA), выделяют направление тренда, убирая высокочастотный шум. Индикаторы момента (Momentum) измеряют изменение цены за фиксированный лаг, отражая силу движения, а осцилляторы (RSI, MACD) позволяют определить зоны перекупленности или перепроданности. Полосы Боллинджера (Bollinger Bands) строятся вокруг скользящей средней с учётом стандартного отклонения, фиксируя периоды повышенной или пониженной волатильности. Индикатор ATR (Average True Range) оценивает средний диапазон колебаний, а объёмные показатели (On-Balance Volume, Chaikin Money Flow) отражают спрос и предложение через сопоставление ценовых изменений и торговых объёмов. Комплекс таких индикаторов, рассчитанный для ряда, существенно обогащает модель информацией о текущей фазе рынка и её вероятных изменениях.

1.12.2 Статистические признаки

Автоматизированное извлечение статистических признаков, например реализуемое библиотекой TSFresh, позволяет получить сотни показателей, описывающих распределение и структуру временного ряда. Среди ключевых категорий — моменты распределения (среднее, дисперсия, асимметрия, эксцесс), автокорреляционные меры (значения ACF и PACF на различных лагах), а также энтропийные показатели, такие как спектральная энтропия после преобразования Фурье и пермутационная энтропия порядка d и лага τ. Дополнительно вычисляются характеристики пиков и впадин (количество экстремумов, амплитуда), длины серий неубывания и невозрастания, статистики выбросов и пр. Полученные признаки обогащают модель квази-динамическими свойствами ряда, позволяя алгоритмам машинного обучения выявлять сложные зависимости во временных паттернах. Применение таких статистических признаков совместно с техническими индикаторами создаёт многомерный описательный вектор для каждой временной точки, что делает прогноз более точным и устойчивым к шуму, по сравнению с использованием «сырых» ценовых данных.

## 1.13 VAR

Vector Autoregression (VAR) представляет собой обобщение авторегрессионных моделей на многомерный случай, позволяя моделировать взаимозависимости нескольких временных рядов одновременно. В отличие от унивариантных моделей, где каждое значение прогнозируется лишь на основе собственной истории, VAR учитывает перекрёстные влияния: текущее значение каждого компонента векторного ряда зависит линейно от прошлых значений всех компонентов. Это делает VAR особенно ценным в финансовом контексте, когда динамика цен разных активов, объёмы торгов и другие индикаторы взаимосвязаны и эволюционируют в единой системе.

Общая формулировка VAR порядка p для векторного ряда записывается как где каждая матрица размера K×K содержит коэффициенты автозависимостей между компонентами, c — вектор констант, а — белый шум с ковариационной матрицей Σ. Оценка параметров {} и c выполняется методом наименьших квадратов при условии стационарности всех компонент.

VAR-модель позволяет строить как одношаговые, так и мультишаговые прогнозы в многомерном пространстве, а также проводить анализ импульсных откликов (impulse response), показывающий, как единичный шок в одной переменной влияет на остальные через временные лаги. Среди преимуществ VAR — гибкость и богатство интерпретаций взаимовлияний, отсутствие необходимости жёстко задавать причинно-следственные связи заранее и возможность выделять структурные шоки методом структурного VAR (SVAR). Основной недостаток — быстрое разрастание числа параметров

(), что требует большого объёма данных и может привести к переобучению; при большом K и p практикуют добавление регуляризации или переход к сокращённым формам (Bayesian VAR). В финансовых приложениях VAR особенно полезен для моделирования совместной динамики нескольких активов, индексных портфелей и макроэкономических индикаторов, где эффекты перекрёстной автокорреляции оказываются существенными.

## 1.14 ML

Методы машинного обучения применяются к временным рядам на основе извлечённых признаков, объединяя исторические значения ряда, технические индикаторы и статистические характеристики в единый многомерный вектор. В отличие от классических статистических моделей, алгоритмы машинного обучения способны автоматически выявлять сложные зависимости и подстраиваться под различную природу данных.

1.14.1 LineralRegression

Линейная регрессия рассматривает прогноз следующего значения ряда как линейную комбинацию набора признаков , включающих, например, лаговые цены или значения технических индикаторов. Ключевая идея заключается в минимизации суммы квадратов ошибок между фактическими значениями ​ и прогнозами , что даёт аналитически вычисляемое решение в виде вектора коэффициентов .

Ключевая формула записывается как

а критерий оценки параметров —

Линейная регрессия характеризуется простотой обучения, прозрачностью модели и возможностью прямой интерпретации каждого коэффициента: показывает силу влияния признака на прогноз. При большом количестве признаков или коллинеарности используется регуляризация (Ridge, Lasso), что предотвращает переобучение и обеспечивает устойчивость оценки. Основное ограничение метода — линейная зависимость между признаками и целевым значением, что не всегда адекватно отражает сложную нелинейную природу финансовых временных рядов. Для преодоления этого недостатка в рамках линейной регрессии применяют полиномиальное расширение признаков или оставляют линейную модель в качестве базового бенчмарка для более сложных алгоритмов..

1.14.2 K-NN

K-Nearest Neighbors регрессия применяет принцип «близости» объектов в пространстве признаков к прогнозированию временных рядов. Для каждого момента времени t формируется вектор признаков , включающий, например, значения ряда в предыдущие p лагов или технические индикаторы. При необходимости прогноза алгоритм находит K ближайших векторов {}, где измеряя расстояние (обычно евклидово) между и каждым , и усредняет соответствующие значения .

Ключевая формула:

K-NN регрессия не требует предварительного обучения параметров и легко адаптируется к нелинейным паттернам, поскольку прогноз формируется на основе реальных точек из истории, близких по многомерному признаковому пространству. Метод эффективен, когда в данных есть повторяющиеся локальные структуры или сезонно-подобные фрагменты. Однако при большом объёме данных и высоком измерении пространства признаков вычислительная сложность поиска соседей может стать критичной, а качество прогноза существенно зависит от выбора K и меры расстояния. Кроме того, K-NN зачастую плохо справляется с шумом и выбросами, так как каждый сосед вносит одинаковый вклад в итоговый прогноз.

1.14.3 XGboost

XGBoost (eXtreme Gradient Boosting) представляет собой продвинутую реализацию градиентного бустинга деревьев решений, специально оптимизированную для высокой производительности и управления переобучением. В задаче прогнозирования временных рядов XGBoost принимает на вход вектор признаков, включающий лаговые значения ряда, технические индикаторы и статистические характеристики, и строит ансамбль последовательно обучаемых деревьев, каждое из которых корректирует ошибки предыдущих.

На каждом шаге добавляется новое дерево, подбираемое так, чтобы минимизировать функцию потерь на остатках предыдущего ансамбля. При этом регуляризующие члены в целевой функции контролируют сложность каждого дерева и предотвращают переобучени.

Итоговый прогноз после m деревьев определяется как:

где η — темп обучения, а каждое дерево выбирается решением задачи

Ω(f) — регуляризационный член, включающий штрафы за глубину дерева и сумму квадратов весов листов.

XGBoost превосходно работает при наличии большого числа разнообразных признаков, обнаруживая сложные нелинейные связи и взаимодействия без явного задания формы этих зависимостей. Ключевыми преимуществами являются встроенный механизм регуляризации, эффективное управление пропущенными значениями и возможность распараллеливания обучения. Тем не менее модель требует тщательной настройки гиперпараметров — скорости обучения, максимальной глубины деревьев, величины регуляризационных штрафов и числа итераций — что делает процесс внедрения более трудоёмким. Кроме того, интерпретируемость ансамбля деревьев слабее по сравнению с простыми линейными моделями, что может осложнять анализ причинно-следственных связей в прогнозах.

1.14.4 CatBoost

CatBoost (Categorical Boosting) — это реализация градиентного бустинга деревьев решений, ориентированная на эффективную работу с категориальными признаками и устранение проблемы сдвига целевой переменной при последовательном обучении. В отличие от классических XGBoost и LightGBM, CatBoost использует технику Ordered Target Statistics, при которой каждое дерево обучается на частичных агрегированных статистиках по целевой переменной, вычисленным только на тех элементах, что предшествуют текущему наблюдению. Это позволяет избежать утечки информации и смещения при работе с временными рядами, где порядок данных критичен.

Ключевой элемент CatBoost — формирование целевых признаков для категориальных переменных через онлайн-агрегацию:

где для каждого наблюдения t сумма целевых значений и число случаев берутся только по тем записям, что предшествуют t, а — сглаживающий параметр («prior») для стабилизации оценки. Остальная часть модели реализует традиционный градиентный бустинг:

где — очередное регрессионное дерево, а η — скорость обучения; но обучение деревьев проходит с учётом Ordered TS.

CatBoost показывает высокую устойчивость к переобучению и превосходит аналоги на наборах с большим числом категориальных признаков, поскольку не требует предварительного one-hot кодирования и автоматически выбирает оптимальные преобразования категорий. При работе с временными рядами CatBoost дополнительно использует специальные схемы временной валидации и может эффективно обрабатывать признаки, основанные на лаговых ценах, технических индикаторах и статистических характеристиках. Главными ограничениями являются значительные требования к памяти и времени обучения для очень больших объёмов данных, а также сложность интерпретации итогового ансамбля деревьев по сравнению с линейными моделями.

1.14.5 RandomForest

Random Forest Regression представляет собой ансамбль декоррелированных деревьев решений, объединённых по принципу усреднения. В задаче прогнозирования временных рядов каждый момент времени t кодируется вектором признаков — обычно это лаговые значения ряда и дополнительные индикаторы — после чего обучается набор M независимых деревьев на разных случайных подвыборках данных и признаков. Итоговый прогноз получается как среднее предсказаний всех деревьев, что способствует снижению дисперсии модели и повышению стабильности прогнозов.

где — предсказание m-го дерева, обученного на случайной подвыборке строк (bootstrap) и случайном подмножестве признаков.

Random Forest эффективен, когда имеются разнотипные признаки (лаги ряда, технические индикаторы, статистические признаки), поскольку не требует масштабирования данных и автоматически оценивает важность каждого признака. Усреднение большого числа слабых моделей позволяет снизить переобучение и обеспечить устойчивый прогноз даже в присутствии шумовых выбросов. Основные ограничения связаны с потерей интерпретируемости на уровне отдельных деревьев и вычислительной нагрузкой при большом числе деревьев и признаков. Кроме того, при наличии сильных временных автозависимостей может потребоваться тщательное формирование лаговых и агрегированных признаков, так как Random Forest сам по себе не учитывает порядок или временную структуру данных.

1.14.6 Prophnet

Prophet — это аддитивная модель временных рядов, предназначенная для гибкого прогнозирования тренда, сезонности и эффектов праздников при наличии выбросов и пропусков данных. Модель представляет ряд как сумму детерминированных компонент и шума:

,

где

* g(t) описывает тренд,
* s(t) — сезонную составляющую,
* h(t) учитывает влияния заранее заданных праздничных и специальных дней,
* — случайный шум.

Тренд g(t) может быть реализован как кусочно-линейная функция с точками изменения скорости роста (changepoints):

,

где k и m — базовые параметры уровня и тренда, a(t) — вектор индикаторных функций для changepoints, а δ, γ — изменения тренда и уровня в этих точках.

Сезонность s(t) моделируется при помощи ряда Фурье:

,

где P — длина сезона (обычно 365.25 для годового и 7 для недельного), а N — число гармоник.

Компонента праздников h(t) задаётся как линейная комбинация индикаторов заранее определённых дат:

,где отражает влияние j-го праздника.

Prophet подходит для бизнес-ориентированных временных рядов благодаря автоматическому обнаружению точек изменения тренда, учёту эффектов праздников и устойчивости к выбросам. Модель проста в использовании и обеспечивает интуитивно понятные результаты. Основным ограничением является то, что аддитивная структура может быть неадекватна при сильном мультипликативном взаимодействии тренда и сезонности; в таких случаях требует предварительного логарифмирования данных.

## 1.15 TSAI

Библиотека tsai создана на базе PyTorch и FastAI для быстрого прототипирования и обучения SOTA-моделей на временных рядах. Она объединяет готовые реализации множества современных архитектур, обеспечивает удобные утилиты для предобработки и гибкую систему колбэков для мониторинга и отладки.

Далее приведён обзор нескольких ключевых архитектур, доступных в tsai:

XCM (Explainable Convolutional Neural Network): Предлагает двухветвевую сверточную структуру: одна ветвь извлекает паттерны вдоль временной оси, другая — вдоль оси признаков. Такое разделение повышает интерпретируемость, позволяя локализовать важные интервалы и признаки для прогноза (Fauvel et al., 2021).

gMLP (Gated Multilayer Perceptron): Упрощённая альтернатива трансформерам без внимания: внутренние линейные слои обогащены сигмоидами в «шлюзах», которые управляют информационным потоком между слоями, что позволяет моделировать зависимости на всём горизонте без рекурсии (Liu et al., 2021).

TSPerceiver: Адаптация Perceiver IO к временным рядам: фиксированное количество «latent» представлений взаимодействует с входом через кросс-внимание, обеспечивая масштабируемость к длинным последовательностям и независимость вычислений от длины ряда (Jaegle et al., 2021).

GatedTabTransformer: Соединяет трансформерные блоки с табличными данными и «шлюзами» (gates) для гибкой обработки разнотипных признаков; применяет механизм позиционного кодирования и селективного внимания к признакам, улучшая работу на временных и табличных рядах (Cholakov et al., 2022).

TSSequencerPlus: Вдохновлён архитектурой Sequencer: полностью линейное, не-сверточное преобразование последовательностей, где каждый токен обрабатывается через слой нормализации и MLP, позволяя эффективно моделировать зависимость на длинных горизонтах без внимания (Tatsunami et al., 2022).

PatchTST: Использует «патчи» временного ряда (непрерывные фрагменты) как токены и трансформерные блоки для их взаимного внимания. Такой подход снижает вычислительную сложность и улучшает представление долгосрочных зависимостей (Nie et al., 2022).

Перечисленные модели лишь небольшая часть представленных решений, но они демонстрируют различные стратегии обработки структур временных рядов — от раздельных свёрток и внимания до линейных и MLP-блоков — и доступны в tsai как готовые компоненты для гибкого экспериментирования и составляет..

## 1.16 FEDOT

FEDOT (Framework for Evolutionary Design of Optimization Techniques) представляет собой платформу для автоматизированного построения конвейеров машинного обучения и моделирования временных рядов с помощью эволюционных алгоритмов. Основная идея заключается в том, чтобы не ограничиваться заранее заданной архитектурой модели, а эволюционно отбирать и комбинировать разнообразные предобработки, алгоритмы прогнозирования и постобработку в едином гибридном пайплайне.

В контексте временных рядов FEDOT выполняет следующие ключевые шаги:

Генерация начальной популяции конвейеров, включающих варианты трансформации признаков (лагирование, дифференцирование, нормализация), базовые модели (ARIMA, Prophet, деревья, линейные регрессии) и ансамблевые методы.

Эволюционный отбор с операторами скрещивания и мутации, которые создают новые пайплайны путём замены, вставки или удаления компонентов.

Многоцелевой критерий оценки, включающий точность прогноза (например, MAPE или RMSE) и сложность модели, что позволяет получать компромисс между качеством и ресурсозатратностью.

Финальный пайплайн, оптимизированный под заданные задачи прогнозирования, автоматически адаптируется к свойствам ряда, будь то мультивариантность, пропуски или сезонные паттерны.

Преимущества FEDOT заключаются в автоматизации экспериментального процесса, гибкости при работе с различными типами данных и прозрачности — итоговая структура пайплайна легко интерпретируется и при необходимости вручную донастраивается. В то же время эволюционный подход требует значительных вычислительных ресурсов и может быть медленным на больших объёмах данных, а результаты зависят от корректного задания пространства возможных операторов и параметров эволюции. Тем не менее FEDOT представляет собой мощный инструмент для прототипирования и сравнения комплексных конвейеров прогнозирования временных рядов без обширных ручных настроек.

## 1.17 Chronos

Chronos — это высокопроизводительная платформа для прогнозирования временных рядов, разработанная в Amazon Science и доступная под открытой лицензией. Основной замысел Chronos состоит в предоставлении единого интерфейса для обучения, оценки и развёртывания современных глубоких моделей прогнозирования, таких как N-BEATS, Transformer (Autoformer, Informer), Temporal Fusion Transformer и LSTM, с учётом особенностей больших и распределённых данных.

Chronos реализует следующие ключевые механизмы:

Модульность и унификация: единая API для всех моделей, позволяющая переключаться между ними без изменения конвейера подготовки данных.

Оптимизированные дата-пайплайны: эффективная загрузка и препроцессинг размеченных и мультивариантных рядов с учётом временных индексов, календарных эффектов и скользящих окон.

Масштабируемость: поддержка обучения на GPU-кластерных средах с автоматическим распределением батчей и синхронизацией градиентов, что обеспечивает линейное ускорение при росте вычислительных ресурсов.

Гибкая настройка: YAML-конфигурации для определения архитектуры моделей, функций потерь, методов оптимизации и расписания обучения, а также встроенные средства для отслеживания метрик (MAE, RMSE, MAPE) и ранней остановки.

Применимость, преимущества и недостатки.

Chronos даёт исследователям и инженерам возможность быстро экспериментировать с передовыми DL-архитектурами прогнозирования без необходимости ручного написания обучающих циклов и предобработки. Платформа особенно ценна в промышленном контексте, где требуется масштабируемость и надёжность при работе с большим числом рядов (например, IoT-данные и финансовые потоки). Однако полнофункциональность Chronos сопровождается высокой кривой обучения и значительными требованиями к аппаратным ресурсам при работе с тяжёлыми моделями и большими наборами данных. Для небольших рядов или ограниченных вычислительных мощностей более подходящими могут оказаться лёгкие модели или AutoML-решения с низкими требованиями.

1.18 Прогнозирование временных рядов с помощью LLM

Недавние исследования демонстрируют, что большие языковые модели (LLM), изначально разработанные для обработки естественного языка, могут быть перенастроены на задачи временных рядов, включая прогнозирование финансовых данных.

В классическом FinLLM подходе основное внимание уделяется анализу новостных текстов и финансовых отчётов, однако такая архитектура не оптимальна для непосредственной работы с числовыми последовательностями цен. Для полноценного использования возможностей LLM требуется выстроить методику, позволяющую трактовать временные ряды как «токены», сохраняя при этом их непрерывный числовой характер и извлекая из них статистические и корреляционные признаки.

1.18.1 Прямой подход

Модели подаётся финансовый временной ряд в промпте, например:

«Ты эксперт по прогнозированию финансовых временных рядов. Мы имеем следующий ряд: 102.5, 103.0, 102.8, 105.2, 104.9, 104.7, 104.3, 104.0, 103.8,

103.6, 103.2, 103.0, 102.7, 102.5, 102.3, 102.0, 101.8. Спрогнозируй следующее значение и верни только число.»

Такой простой подход позвоялет получить от модели предсказание на следующий день.

1.18.2 подход StockTime

В работе StockTime предложена новая архитектура, в которой исторические котировки рассматриваются как последовательность «патчей» — отрезков длины l, каждый из которых представляется как один LLM-токен. Формально ряд цен делится на n=⌊D/l⌋ непересекающихся сегментов

Эти патчи проходят через autoregressive encoder, реализованный на базе LSTM, который преобразует каждый сегмент в скрытое представление .

Параллельно из патчей извлекаются текстовые подсказки: корреляции между акциями, статистические тренды (минимум, максимум, среднее, изменение) и временные метки. Они формируют шаблонный текст, который кодируется через замороженный LLM (например, LLaMA3-8B), давая текстовую эмбеддинговую часть . Затем и объединяются в единую векторную эмбеддинг ​ в пространстве размерности LLM.

Основная предсказательная способность LLM используется на этапе token-level prediction. Объединённые эмбеддинги

{} подаются на вход замороженного LLM, который, обучаясь лишь линейному проекционному слою сверху, «прогнозирует» следующий токен . Этот токен затем развёртывается обратно в числовой сегмент, а целевая функция оптимизации задаётся как среднеквадратичная ошибка между и по всем сегментам: .

Таким образом, LLM выступает в роли авторегрессионного форекастера, свободного от фиксированного окна lookback .

В обширном эксперименте на наборах S&P 100 (дневных и почасовых) и четырёх публичных датасетах с aligned text (Bigdata22/23, ACL18, CIKM18) StockTime показал значимое улучшение MSE и Information Coefficient по сравнению с general LLM, FinLLM и специализированными time-series LLM (Times-LLM, AutoTimes, FPT)

. При этом сохраняются ресурсоэффективность (замороженный бэкбон LLM, обучение лишь эмбеддингов и проекций) и гибкость (любая длина lookback без изменения модели).

Таким образом, StockTime демонстрирует, что при правильной интеграции временных и текстовых признаков LLM могут выступать эффективными инструментами для прогнозирования финансовых рядов, сочетая преимущества autoregressive encoder для числовых данных и богатые контекстные представления LLM. Это направление открывает возможности для дальнейших исследований мультимодальных архитектур и их применения в задачах алгоритмической торговли и риск-менеджмента.

## 2. Детектирование аномалий и поиск точек перелома в финансовых временных рядах

Детектирование аномалий (anomaly detection) и обнаружение точек перелома (change-point detection) являются ключевыми задачами анализа временных рядов, поскольку позволяют выявлять неожиданные или структурные изменения в динамике процессов. В финансовых приложениях такие события часто сигнализируют о смене рыночных режимов, начале кризисных явлений или возможностях для входа и выхода из позиций.

Аномалия — это наблюдение или набор наблюдений, существенно отличающиеся от ожидаемой модели поведения ряда. Обычно выделяют три типа аномалий:

* Точечные (point anomalies) — отдельные выбросы, не вписывающиеся в статистические характеристики ряда.
* Контекстуальные (contextual anomalies) — значения, аномальные только в определённом контексте (например, объём торгов необычен только в часы закрытия биржи).
* Коллективные (collective anomalies) — группы последовательных наблюдений, которые совместно образуют аномалию (например, серия резких падений цены).

Точка перелома (change point) — момент времени, при котором изменяются параметры структуры ряда: среднее, дисперсия, автокорреляция или параметры модели. В финансовом контексте точками перелома могут быть:

1. Смена тренда (переход от восходящего к нисходящему или наоборот).
2. Изменение волатильности (переход к фазе повышенной турбулентности).
3. Сдвиг в корреляции между акциями или секторами.

Обнаружение таких событий имеет практическое значение: смена тренда может служить торговым сигналом, а резкое изменение волатильности — предупреждать о росте рисков.

Методы детектирования аномалий и точек перелома можно разделить на несколько классов:

* Статистические
  + Контрольные карты (Shewhart, CUSUM, EWMA) отслеживают отклонения остатков модели (например, ARIMA) от нулевого уровня.
  + Тесты на единичный корень (ADF, KPSS) и тесты на изменение параметров (Pettitt, Bai–Perron) проверяют однородность статистических характеристик ряда.
* Непараметрические
  + Методы сегментации с оптимизацией (binary segmentation, PELT) минимизируют сумму «стоимости» разбиения (например, сумма квадратов ошибок) при введении точек разрыва.
  + Кернел-базированные (kernel change-point detection) оценивают изменения распределения через меру RKHS.
* Машинное обучение и глубокие сети
  + Кластеризация и методы ближайших соседей (k-NN) обнаруживают выбросы по расстоянию до облака нормальных точек.
  + Автоэнкодеры и вар variational autoencoders реконструируют вход; большие ошибки восстановления указывают на аномалии.
  + Рекуррентные нейросети (LSTM-AE) обучаются предсказывать следующий шаг; значительное расхождение между прогнозом и фактом свидетельствует об аномалии или точке перелома.

Таким образом, детектирование аномалий и поиск точек перелома образует универсальный инструментарий для понимания и управления динамикой финансовых процессов.

## 2.1 Генерация аномалий

Для объективной оценки качества алгоритмов детектирования аномалий во временных рядах часто прибегают к контролируемой генерации «искусственных» выбросов, в тех же финансовых данных. Ниже приводится один из распространённых подходов, позволяющий воспроизводимо внедрять аномалии и затем оценивать работу детекторов по известным «истинным» меткам.

2.1.1 Оценка базовой плотности доходностей

Пусть — исходный финансовый ряд без выраженных аномалий на некотором отрезке . Для обеспечения стационарности мы рассматриваем логарифмические доходности

На участке оценивается эмпирическое распределение доходностей (например, гауссовское с параметрами , или непараметрическое ядровое).

2.1.2 Генерация базового синтетического ряда

Для создания «чистого» (без аномалий) ряда задаём начальное значение и затем рекурсивно строим

.

Таким образом получаем модельный ряд, сохраняющий статистику доходностей исходного ряда, но лишённый оригинальных выбросов.

2.1.3 Внедрение аномалий

Далее в синтетическом ряде выбирают N случайных моментов и «замешивают» аномалии, заменяя соответствующие доходности на «экстремальные» значения. Например, одинарная аномалия может задаваться фиксированным скачком берётся, скажем, из распределения с «толстыми хвостами» (например, )) или в виде заданного множителя (например, для +10 % за один шаг). После этого все последующие значения пересчитывают по формуле экспоненциальной рекурсии, чтобы аномалия отразилась на уровне цен:

.

В итоге в полученном ряде содержатся N известных точек аномалий, что позволяет сравнить их с детектированными алгоритмом.

2.1.4 Метрики оценки: Precision, Recall и NAB Scoring

После применения алгоритма детектирования аномалий мы получаем набор предсказанных меток и имеем «истинные» метки . Классические метрики:

Precision (точность) — доля детектированных точек, которые действительно являются аномалиями.

Recall (полнота) — доля «истинных» аномалий, успешно обнаруженных.

Более продвинутую оценку даёт NAB score (Numenta Anomaly Benchmark). Этот метод присваивает каждому обнаружению весовую функцию, зависящую от близости предсказания к истинной точке аномалии, по формуле

,

где функция отклика максимальна, если , плавно спадает при удалении от τ в заданном «окне допуска» и допускает штраф за слишком ранние или слишком поздние детекции, а ложные срабатывания на «чистых» участках получают отрицательные вклады. Финальная метрика нормируется так, чтобы случайное детектирование давало 0, идеальное — +1, а худшее — −1.

Использование NAB score позволяет учесть не только факт обнаружения, но и своевременность, что критично в финансовых приложениях, где раннее предупреждение об изменении режима может быть более ценным, чем запоздалое обнаружение.

Таким образом, синтетическая генерация аномалий через выборку доходностей и внедрение экстремальных скачков обеспечивает контролируемую среду для тестирования алгоритмов, а сочетание классических метрик Precision/Recall и NAB score позволяет всесторонне оценить их эффективность в задачах финансового анализа.

## 2.2 Статистические методы детектирования аномалий

Введём несколько классических детекторов, применяемых к одномерным временным рядам. Во всех случаях требуется задать окно длины w (для «скользящих» детекторов) или использовать весь доступный набор данных (для глобальных).

OutlierDetector

Выявляет выбросы на основе межквартильного размаха (IQR). Пусть — 25-й и 75-й процентили распределения ряда . Тогда межквартильный размах

, а пороги детекции задаются как , обычно k = 1.5. Точка считается аномалией, если .

MeanAnomalyDetector

Использует скользящее среднее и, опционально, скользящее стандартное отклонение на окне длины w: .

Для «great\_of\_mean\_detector» (lower=False) сигнал аномалии генерируется при ,

для «least\_of\_mean\_detector» (lower=True) — при , где α — задаваемый порог (по умолчанию α = 3).

DistributionBasedAnomalyDetector

Параметрически аппроксимирует распределение всего ряда (например, нормальным и вычисляет p-значение для каждой точки: . Точка ​ считается аномалией, если , где β — уровень значимости (обычно 0.05).

RareDistributionDetector

На каждом шаге строит эмпирическое распределение последних w значений , сортирует их и помечает в качестве аномалий n самых экстремальных элементов в каждой «хвостовой» части: . Параметр n определяет «глубину» хвоста, а «окно» w — локальный контекст. Варианты с n=1,2,3 и разными w позволяют балансировать между чувствительностью и устойчивостью к шуму.

Каждый из перечисленных методов имеет свою область применения: IQR–детектор прост и не требует параметризации, но чувствителен к несимметричным распределениям. Скользящее среднее улавливает временные тренды, однако требует надёжной оценки . Параметрический детектор выгоден при действительно нормальном характере данных, а редкие распределения (RareDistribution) эффективно выявляют строго экстремальные события, оставаясь нечувствительными к умеренным колебаниям. В совокупности эти методы образуют базовый инструментарий для статистического анализа аномалий во временных рядах.

## 2.3 Прогнозный подход

В этом методе на каждом шаге строится прогноз цены любым методом, например экспоненциальной скользящей средней (EMA), а затем вычисляется ошибка прогноза в виде MAPE. Точка t помечается как аномалия, если превышает заранее заданный порог δ. Такой подход прост в реализации и автоматически адаптируется к локальной динамике, однако требует аккуратного выбора порога и может пропускать аномалии при медленном нарастании ошибки.

## 2.4 Isolation Forest

Isolation Forest — метод детектирования аномалий, основанный на идее, что аномальные точки легче «изолировать» в случайных деревьях разбиения. Алгоритм строит ансамбль «изолирующих деревьев» (iTrees), в каждом из которых на узлах случайно выбирается признак и случайный порог для разбиения. Аномалии обычно оказываются в ближних к корню листьях, поскольку для их изоляции требуется меньшее число разбиений.

Для объекта x его аномальный «путь» в дереве определяется длиной h(x) от корня до листа. Средняя длина пути по дереву нормируется относительно ожидаемой длины для случайного образца размера n: , где — гармонический номер. Чем ближе s(x,n) к 1, тем более аномален объект.

Isolation Forest эффективно выявляет выбросы в высокоразмерных данных без предположений о распределении, масштабах или предобработке признаков. Основные преимущества — линейная сложность по объёму данных и гиперпараметрам, а также отсутствие необходимости задавать пороги вручную: достаточно оценить аномальные объекты по их низкому среднему пути. К недостаткам относится чувствительность к выбору числа деревьев и глубины, а также невозможность явно учесть временную структуру без добавления лаговых признаков..

## 2.4 Кластерный подход

В кластерных методах аномалии определяются как точки, плохо вписывающиеся в образованные группы («кластерные выбросы»).

K-means

Сначала пространство признаков (например, лаговых цен и технических индикаторов) может быть снижено через PCA до d главных компонент, сохраняющих большую часть дисперсии:

где столбцы W — собственные векторы ковариационной матрицы. Затем алгоритм K-means разбивает на K кластеров путём минимизации внутрикластерного SSE. Для каждого объекта вычисляют расстояние до своего центра : . Точки с выше заданного порога считаются аномальными.

K-NN (кластеризация через плотность соседей)

Для каждого момента t вычисляют среднее (или медианное) расстояние до K ближайших соседей в исходном признаковом пространстве:

Если превышает порог, точка объявляется аномалией. Этот метод не требует предварительного задания числа кластеров и естественно адаптируется к локальной плотности данных.

Оба подхода выявляют объекты, находящиеся «вне» основной массы данных, и хорошо демонстрируют свою эффективность на временных рядах, предварительно преобразованных в многомерный приз­наковый вектор.

## 2.5 Автоэнкодеры

Автоэнкодер — это нейросетевая архитектура, состоящая из двух частей: энкодера , сжимающего входной вектор признаков в низкоразмерное латентное пространство , и декодера , восстанавливающего из аппроксимацию .

Ключевая функция потерь при обучении:

.

После обучения на «нормальных» данных реконструкционная ошибка используется как мера аномалии: если она превышает порог δ, точка считается выбросом.

Преимущества автокодеров в том, что они автоматически выявляют наиболее информативные признаки и эффективно моделируют нелинейные зависимости. Основной недостаток — потребность в достаточном объёме «чистых» данных для обучения и чуткость к выбору архитектуры (глубина, размер латента) и порога ошибки..

## 2.6 Детектирование многомерных аномалий (PyOD)

В задачах многомерного анализа аномалий удобно использовать библиотеку PyOD, предоставляющую широкий набор алгоритмов. Ниже перечислены основные модели:

* HBOS (Histogram-based Outlier Score): Оценивает аномальность каждого признака на основе частотных гистограмм и объединяет независимые «оценки» через произведение или сумму.
* ABOD (Angle-based Outlier Detection): Вычисляет для каждой точки дисперсию углов между векторами, соединяющими её с парами других точек; аномалии имеют низкую угловую вариативность.
* LOF (Local Outlier Factor): Сравнивает плотность локального соседства точки с плотностью её соседей; аномалии находятся в менее плотных регионах по сравнению с окружением.
* CBLOF (Cluster-based Local Outlier Factor): Разбивает данные на кластеры, затем оценивает аномальности по размеру и удалённости кластера: точки из небольших и отдалённых кластеров считаются выбросами.
* Isolation Forest: «Изолирует» объекты случайными разбиениями признакового пространства; аномалии, требующие меньше уровней разбиения, получают высокую оценку.
* KNN (k-Nearest Neighbors): Оценивает среднее (или медианное) расстояние до k ближайших соседей; большие расстояния соответствуют потенциальным аномалиям.
* MCD (Minimum Covariance Determinant): Строит робастную оценку ковариации на подмножестве данных с минимальным детерминантом; точки, плохо соответствующие этой модели, считаются выбросами.
* PCA (Principal Component Analysis): Реконструирует объекты из первых d главных компонент и измеряет ошибку восстановления; аномалии дают большие остатки.
* LSCP (Locally Selective Combination in Parallel): Создаёт ансамбль детекторов, затем для каждой точки выбирает несколько наиболее надёжных базовых моделей на локальном подмножестве для итоговой оценки.

Такое разнообразие методов позволяет быстро проверить гипотезы и найти подходящий детектор, который поможет находить аномалии даже в многомерных рядах.

## 2.7 Fedot Industrial: библиотека для детектирования аномалий

Fedot Industrial (Fedot.Industrial) — расширение AutoML-фреймворка FEDOT, предназначенное для решения задач промышленного анализа, включая детектирование аномалий во временных рядах и изображениях. Через единый высокоуровневый API (FedotIndustrial) она позволяет быстро настраивать и запускать пайплайны, объединяющие предобработку, извлечение признаков и алгоритмы обнаружения выбросов.

Ниже перечислены основные методы детектирования аномалий, реализованные в Fedot Industrial:

* Z-score Detector — классический детектор выбросов по отклонению от скользящего среднего
* IQR-based Outlier Detector — поиск выбросов через межквартильный размах
* Isolation Forest — ансамбль изолирующих деревьев для выявления точек, легко «изолируемых»
* Local Outlier Factor (LOF) — сравнительный анализ локальной плотности соседних точек
* K-Nearest Neighbors (KNN) Detector — оценка аномальности по расстоянию до K ближайших соседей
* One-Class SVM — метод опорных векторов для выявления «области нормальности»
* Autoencoder Detector — реконструкция входного вектора через сжатое латентное пространство и порог на ошибку восстановления
* Change-Point Detector — сегментация ряда на участки с разными статистическими свойствами (например, PELT)
* Prophet-based Detector — аддитивная модель для выявления неожиданных отклонений от тренда и сезонности

Каждый из этих детекторов легко интегрируется в эволюционно оптимизируемый пайплайн — Fedot Industrial автоматически подбирает лучшую комбинацию предобработки, типа детектора и его гиперпараметров по заданному критерию качества (MAPE, RMSE, Precision/Recall и др.).

## 2.8 Детектирование аномалий с помощью LLM

Современные большие языковые модели (LLM) демонстрируют способность обрабатывать не только текст, но и числовые последовательности, представленные в виде текстовых промптов. В задаче обнаружения аномалий во временных рядах можно выделить два принципиально разных подхода: пакетное (batch) детектирование и «реальное время» (online) с использованием прогноза.

2.8.1 Пакетный (batch) анализ через промпт

В этом сценарии мы формируем единый текстовый запрос, в котором передаём LLM всю историю ряда и инструкцию действовать как эксперт по аномалиям. Пример промпта:

«Ты — специалист по анализу временных рядов. Вот последовательность ежедневных цен закрытия за последние 20 дней:

«102.5, 103.0, 102.8, 105.2, 104.9, 150.0, 104.7, 104.3, 104.0, 103.8,

103.6, 200.0, 103.2, 103.0, 102.7, 102.5, 102.3, 300.0, 102.0, 101.8.

Определи, есть ли в этом ряде аномалии, и если да, приведи их индексы (номера дней).»

Модель, интерпретируя выбросы 150.0, 200.0 и 300.0 как аномалии, может вернуть:

“Обнаружены аномалии на позициях 6, 12 и 18.”

тот подход прост в реализации и позволяет единовременно проверить всю серию, однако его точность и воспроизводимость зависят от качества формулировки промпта и способности LLM оперировать длинными списками чисел без потери контекста.

2.8.2 Детектирование в реальном‐времени (online) с использованием прогноза

На каждом шаге мы делаем прогноз следующего значения ряда лучшей моделью. А затем в LLM передаём предыдущие значения, прогноз и реальное значение

«Ты — эксперт по аномалиям.

Предыдущий ряд: 103.0, 102.7, 102.5, 102.3, 101.8, 101.0, 100.8

Прогноз для текущего дня был 100.0, фактическое значение — 105.0. Считаешь ли ты это аномалией? Ответь «Да» или «Нет», и если «Да», объясни причину.»

В случае ответа «Да» такая модель отреагирует на резкий отрыв факта от прогноза и выдаст интерпретацию превышения допустимой погрешности.

Преимущества этого метода — адаптивность к меняющейся динамике: модель постоянно «видит» свои собственные ошибки и может корректировать чувствительность к аномалиям. Недостатки — высокая стоимость запросов и накопление задержки из-за двойного взаимодействия для каждого шага.

Оба подхода демонстрируют, как LLM можно использовать вне классического NLP-контекста для детектирования аномалий во временных рядах: пакетная проверка подходит для ретроспективного анализа, а онлайн-метод с прогнозом обеспечивает динамическое обнаружение и более своевременные сигналы.

## 3. Оценка новостей

В работе [7] показана эффективность LLM при оценке новостных заголовков. Для оценки заголовков необходимо получать новостные заголовки классифицированные по отношению к конкретной компании. Далее очищенные от дубликатов заголовки подаются в LLM со следующим промптом: «Ты финансовый эксперт. Твоя задача оценивать потециальное влияние новостей на стоимость акцией компаний. Далее тебе будет передан запрос в формате: «Оцени влияние новости {заголовк} на цену акций компании N. Отвечай ПОЛОЖИТЕЛЬНОЕ, если после данной новости цены акций компаний вырастут, НЕЙТРАЛЬНО, если влияние будет слабым или неочевидно, и НЕГАТИВНОЕ, если цены акций компании упадут. Возвращать только одно слово.» Далее оценка от LLM переводится в число 1,0,-1 все значения новостей за день складываются и усредняются. Из лучших компаний по оценке формируется портфель на следующий день. Такой подход показывает неплохие результаты.

## 4. Разработка комбинированного подхода

Для улучшения предсказательных способностей LLM будем использовать комбинированный подход. За базовый метод примем «прямой метод» для предсказания.

4.1 Переход к многомерным рядам

В начале обновим промпт так чтобы начать учитывать не только цену закрытия, но и другие свойства рядов. Попробуем добавить свойства определяемые регрессией, как важные фичи. Так промпт примет вид:

«Ты эксперт по прогнозированию финансовых временных рядов. Мы имеем следующий характеристики:

* Цены: 102.5, 103.0, 102.8, ... , 101.8.
* Объёмы торгов: 10, 11, 15, ..., 8.
* Доходности (в %): -0.05, 0.05, -0.02, ... -0.04
* ...
* MACD индикатор: 0, 0, 0, ..., 1

Спрогнозируй следующее значение ряда цена для компании N и верни только число.»

В При передаче запроса важно учитывать контекстное окно модели, иначе можно передать слишком много данных и тогда модель отбросит часть информации.

4.2 Учёт точек перелома и аномалий

Для расширения информации модели будет полезно учитывать аномалии в ряду, так добавление ряда: «Аномалия. Поможет модели грамотно трактовать смену тренда и точнее определять направление дальнейшего движения цены.

4.3 Учёт новостного фона

Далее логичным шагом будет передача новостного контекста акции. Рассмотрим два варианта:

1. Основан на предварительной оценке новостей и передаче числовой оценки в виде ряда «Новостной фон (-100 – крайне отрицательный / 100 крайне положительный)».

2. Передача новостных заголовков в промпте:

«Ты эксперт по прогнозированию финансовых временных рядов. Мы имеем следующий характеристики:

* Цены: 102.5, 103.0, 102.8, ... , 101.8.
* Объёмы торгов: 10, 11, 15, ..., 8.
* Доходности (в %): -0.05, 0.05, -0.02, ... -0.04
* ...
* Аномалия: нет, нет, нет, ..., да

Новости за последние дни:

* 19.06 Компания планирует выплатить дивиденды.
* 15.03 В компании заявили о смене генерального директора.

Спрогнозируй следующее значение ряда цена для компании N и верни только число.»

4.4 Передача изображения ряда

Современные LLM умеют не только обрабатывать текст, но и работать с изображениями. Поэтому дополнительными данными для модели может служить передача графика цен за последние дни, а также графика с отображенными технических и индикаторов и паттернов используемых трейдерами.

Итоговый запрос к модели:

«Ты эксперт по прогнозированию финансовых временных рядов. Мы имеем следующий характеристики:

* Цены: 102.5, 103.0, 102.8, ... , 101.8.
* Объёмы торгов: 10, 11, 15, ..., 8.
* Доходности (в %): -0.05, 0.05, -0.02, ... 0.10
* ...
* Аномалия: нет, нет, нет, ..., да
* Новостной фон (-100 – крайне отрицательный / 100 крайне положительный): 20, -30, 30, ..., 90

Спрогнозируй следующее значение ряда цена для компании N и верни только число.»

## 5. Результаты

Комбинированный подход позволил улучшить точность прогнозов модели на 30% от базового показателя. Такой подход расходует приблизительно 100 токенов на один прогноз. Метрики подхода соответствуют и даже превосходят классические методы прогнозирования рядов.

# ЗАКЛЮЧЕНИЕ

В данной работе проведён всесторонний анализ методов прогнозирования финансовых временных рядов, детектирования аномалий и оценки новостного фона с использованием как классических статистических и машинных алгоритмов, так и современных подходов на базе больших языковых моделей (LLM). Были рассмотрены эвристические и наивные методы (SMA, EMA, Drift, Fourier), классические статистические модели (ARIMA, SARIMA, TBATS), линейные архитектуры (DLinear, NLinear), рекуррентные сети (RNN, LSTM), подходы на основе автоматизированного машинного обучения (Auto-ARIMA, FEDOT) и специализированные библиотеки (tsai, Chronos). Для задачи детектирования аномалий исследованы статистические детекторы (IQR, z-score, distribution-based), алгоритмы на основе прогноза (EMA + MAPE), а также многомерные и глубинные решения (Isolation Forest, кластеризация, автоэнкодеры, PyOD, Fedot Industrial). Кроме того, подробно рассмотрены методы оценки новостей через LLM-промпты и расчёт агрегированных новостных индексов.

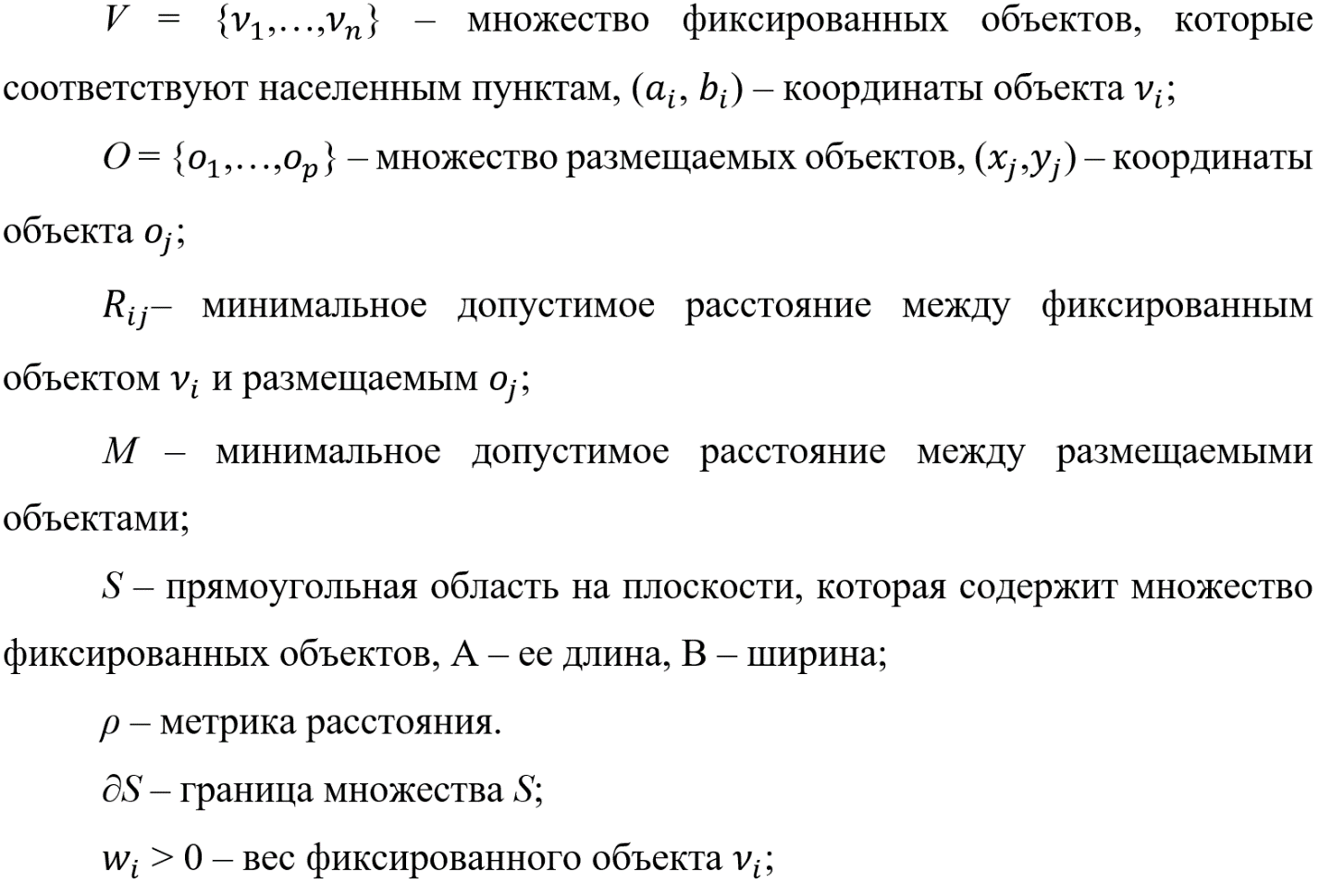
На основе синергии перечисленных подходов был предложен комбинированный метод, включающий передачу в LLM расширенного контекста: многомерных признаков ряда (цены, объёмы, доходности, технические индикаторы), меток точек перелома и аномалий, а также числовых и текстовых описаний новостного фона. Экспериментально показано, что такой подход повышает точность прогноза на 30 % относительно базового «прямого» метода EMA и укладывается в разумный лимит токенов.

Следующим этапом предполагается разработка AI-агента, способного имитировать работу реального трейдера: он будет оперативно обрабатывать поступающие котировки, новости, сигналы из соц. сетей и графические данные, фундаментальные данные о компаниях, хранить объёмный контекст, сопоставимый с человеческим, и на его основе самостоятельно принимать торговые решения. Такой агент должен объединить в себе все модули доступные обычному человеку, а также вычислительные мощности машины. Что позволит перейти к построению полноценной интеллектуальной системе алгоритмической торговли.

Ссылка на код работы: <https://github.com/SodaWay1617/Financial-Time-Series-Forectsting>

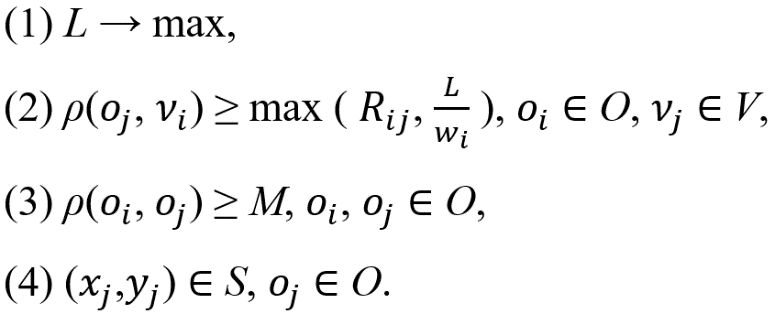
## 1.2 Математическая модель задачи о размещениях на плоскости

Введем следующие обозначения:



Требуется найти оптимальное расположение объектов из в области с учётом всех ограничений. Для максимизации взвешенного минимального расстояние между размещаемыми и фиксированными объектами введем дополнительную переменную .

Тогда модель имеет вид:



Задача (1)–(4) относится к классу NP-трудных задач [2]. Также следует учесть, что допустимая область может быть несвязной и иметь невыпуклую форму.

## 1.3 Подходы для решения задачи о размещениях

В [18] представлено историческое обоснование разработки нового комбинаторного алгоритма, в основе которого лежит ряд исследований по данной теме. Приводится описание подхода для решения задачи в метрике с помощью комбинаторного подхода и линейного программирования [2]. Для проблемы размещения двух объектов предлагается использовать, при различных условиях, или алгоритм ветвей и границ, или решать серию задач линейного или целочисленного программирования [16]. Был введён переход от непрерывной задачи в дискретную с метрикой [5]. Приводится ссылка на работу с исследованием параллельных алгоритмов размещения при в метриках и [11]. Было доказано, что если множество допустимых решений не пусто, то найдется оптимальное решение, в котором по крайней мере один объект находится в угловой точке этого множества. На основе этого был разработан алгоритм, представленный в [18], для размещения объектов в метрике , где ≠ и - произвольные положительные вещественные числа.

Множество других исследований также посвящено решению задачи о размещении. Ниже приведены некоторые значимые работы, которые способствовали развитию теории этой проблемы.

Так, Simpson в своей работе [10] предложил геометрическое решение упрощённой версии задачи Вебера, известной как "проблема Вебера с двумя объектами". Решение основывается на построении перпендикулярных биссектрис между парами точек и определении точки пересечения этих биссектрис.

Модель Вебера [13] была разработана с целью минимизации транспортных затрат между поставщиками, производителями и потребителями. Вебер предложил формулу для определения оптимального местоположения объекта, исходя из минимизации транспортных расходов.

Впоследствии Brimberg и его соавторы предложили новую математическую модель для размещения нескольких объектов на плоскости, используя критерий максимина и прямоугольные расстояния между ними. Они разработали смешанную целочисленную формулировку для этой модели и эвристический алгоритм, основанный на комбинации жадного случайного адаптивного поиска и метода локального поиска [2].

В [15] представлен эвристический алгоритм для решения задачи размещения объектов на плоскости, применяя критерий максимина и минимальные допустимые расстояния между ними. Алгоритм основан на итеративном добавлении наиболее благоприятных точек к множеству решений и позволяет получить приближённое оптимальное решение проблемы размещения.

Также существуют и другие методы, такие как алгоритм разделения слоев (SDA) [4], генетические алгоритмы [19] и нейронные сети [7], которые также могут быть использованы для решения данной задачи.

Ожидается, что использование изучаемого комбинаторного алгоритма приведет к получению приближенных решений за более короткое время по сравнению с классическими методами.

# РАЗРАБОТКА КОМБИНАТОРНОГО ПОДХОДА ДЛЯ ПРИБЛИЖЕННОГО РЕШЕНИЯ ЗАДАЧИ О РАЗМЕЩЕНИЯХ

## 2.1 Постановка задачи

Пусть имеются начальные условия, соответствующие модели (1)-(4). Основная идея алгоритма заключается в выборе некоторого и проверке существования допустимого решения. Введём вспомогательную задачу – допустимое размещение (ДР), определяет возможность размещения объектов в области вне запрещенных зон, учитывая ограничение на минимальное расстояние между ними, выполняется с применением декомпозиции. При изменении запрещенные зоны последовательно расширяются или сужаются до нахождения допустимого значения с заданной точностью.

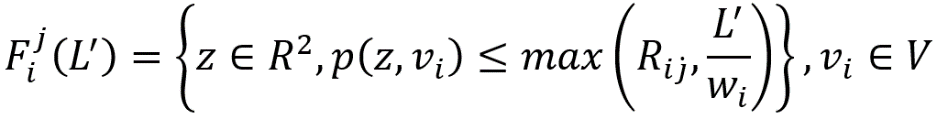
## 2.2 Решение для двух объектов

Значения параметра выбираются с использованием метода дихотомии. Начальный интервал, содержащего оптимальное значение , определяется из следующего неравенства:



Для метода дихотомии применяются следующие правила: если существует ДР, то метод делает шаг к правому концу интервала; в противном случае, к левому.

Для решения ДР необходимо найти контуры объединения изотетических прямоугольников ограничений. Это осуществляется с помощью метода плоского заметания (ПЗ), используя дерево отрезков [9]. Для каждого изотетичные прямоугольники – это область:



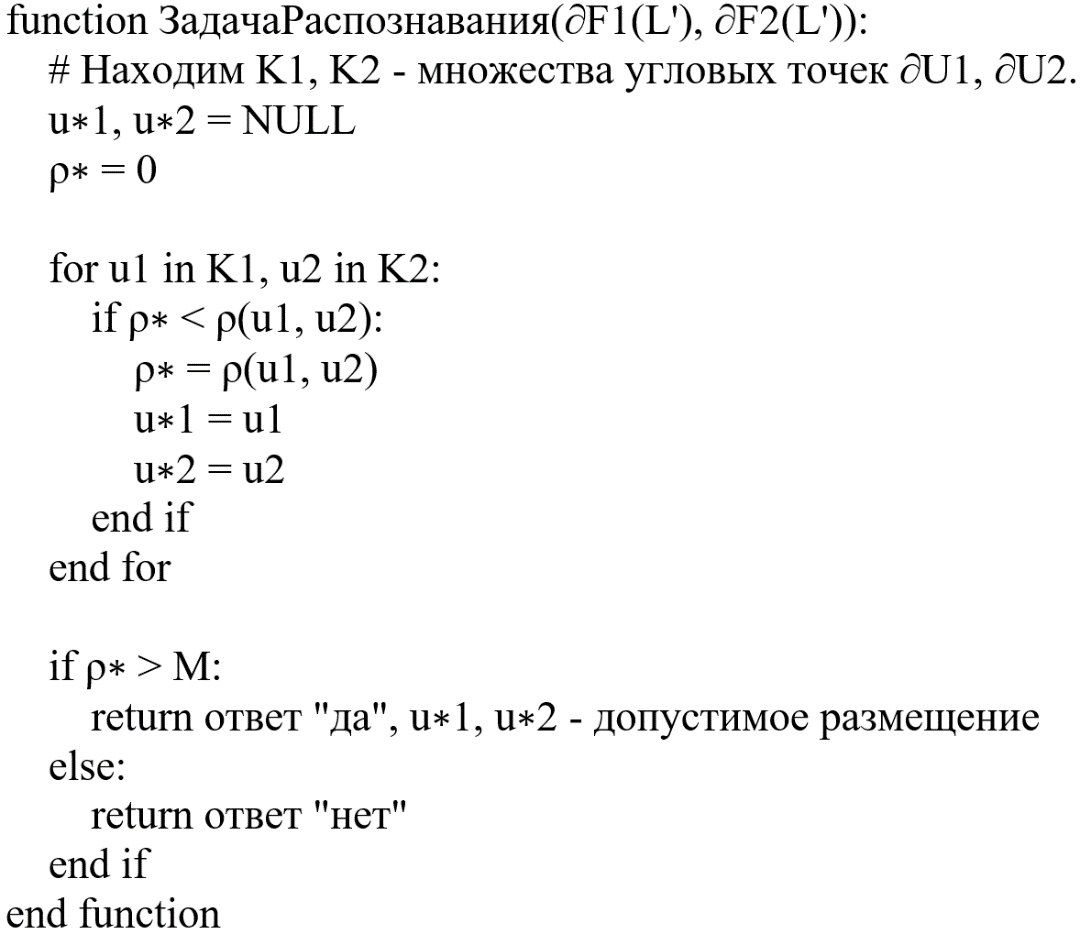
Контур ∂() состоит из ориентированных вертикальных и горизонтальных ребер, формирующих непересекающиеся циклы. Ориентация ребер определяется таким образом, что область объединения квадратов находится слева. Также контур обрамляет и возможные внутренние полости области объединения.

Запрещенные зоны соответствуют открытым множествам ). Допустимая область для размещения объекта обозначается как и определяется как разность между областью и int().

Для определения и , следует принять во внимание следующие точки:

1. Угловые точки множества, которые не входят в (L′), а также угловые точки в и угловые точки *.*
2. Угловые точки с наибольшими и наименьшими значениями координат.
3. Угловые точки из множества, которые представляют собой наиболее отдалённые точки при условии, что = [5].

Псевдокод алгоритма 1 для определения допустимости размещения с 2-мя размещаемыми объектами при фиксированном :



Количество итераций алгоритма определяется как . Трудоёмкость построения контура объединения квадратов составляет [9]. Соответственно, общая трудоёмкость алгоритма 1 оценивается как

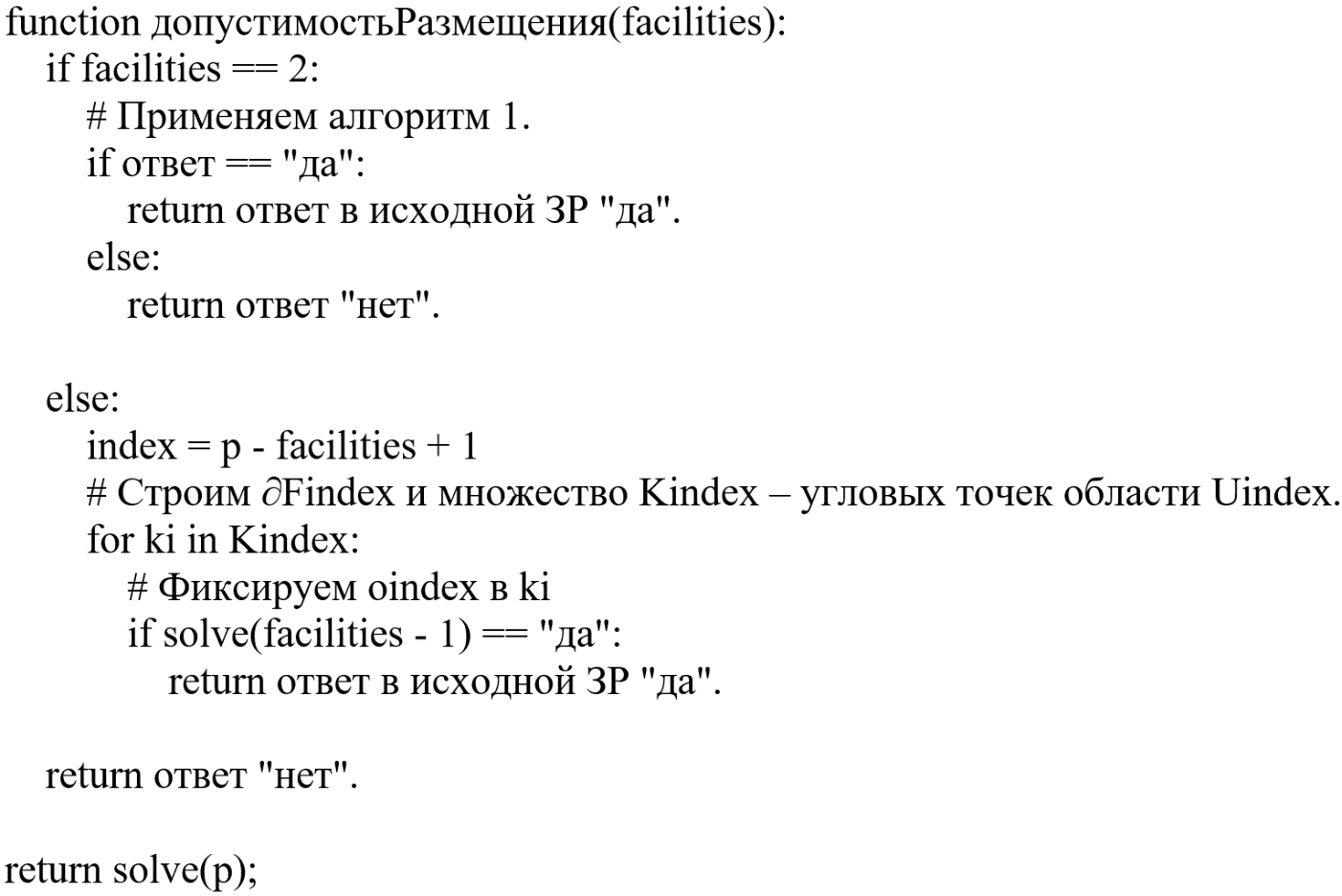
## 2.3 Рассмотрение общего случая

Выбор значений параметра осуществляется согласно описанному выше подходу. При решении задачи допустимости размещения для рекомендуется использовать рекурсию, проверяя доступность размещения объектов.

В работе [18] представлены концепции "сдвига влево" и блокирования, на основе которых была сформулирована и доказана следующая теорема:

“Если существует допустимое размещение для заданного , то существует допустимое решение задачи (1)-(4), в котором по крайней мере один объект располагается в угловой точке допустимой области ∂.”

Исходя из этой теоремы, при решении ДР для p объектов один из объектов можно предполагать расположенным в угловой точке, имеющей запрещенную зону радиуса M, таким образом, производится редукция задачи. Решение ДР достигается путем фиксации объекта в угловой точке его допустимых расположений и перехода к решению задачи с меньшим количеством объектов. Применяя описанную рекурсию, можно вывести алгоритм решения ДР для :



Трудоемкость алгоритма 2 не превосходит , при этом используемая память *.* [18]. Весь алгоритм для нахождения решения задачи с заданной точностью имеет сложность где *D*(*S*) - диаметр множества *S*.

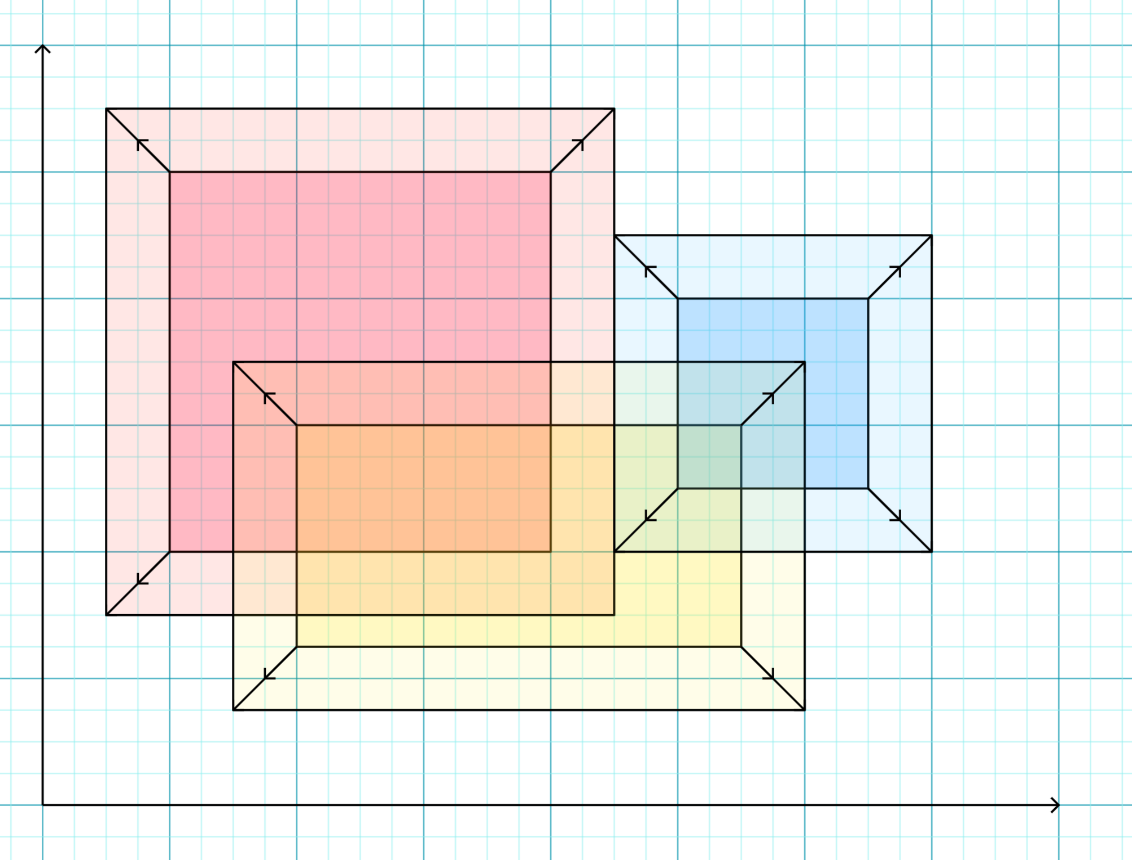


Рис.1 Пример расширения зон ограничений при увеличении параметра L.

## 2.4 Реализация построения контура прямоугольников при помощи дерева отрезков

Дерево отрезков, известное также как Segment Tree, это структура данных, обеспечивающая эффективное решение задач, связанных с операциями на интервалах в массиве. Это инструмент, позволяющий быстро добавлять, удалять и осуществлять запросы информации на определенных отрезках.

Данная структура представляет собой бинарное дерево, где каждый внутренний узел представляет собой отрезок массива, а листья соответствуют индивидуальным элементам массива. Значения внутренних узлов обычно вычисляются на основании значений их дочерних узлов посредством агрегирующей функции. Более подробное описание дерева отрезков приведено в работе [9]. Далее мы определим основные операции с деревом отрезков, используемые для вычисления контура объединения прямоугольников.

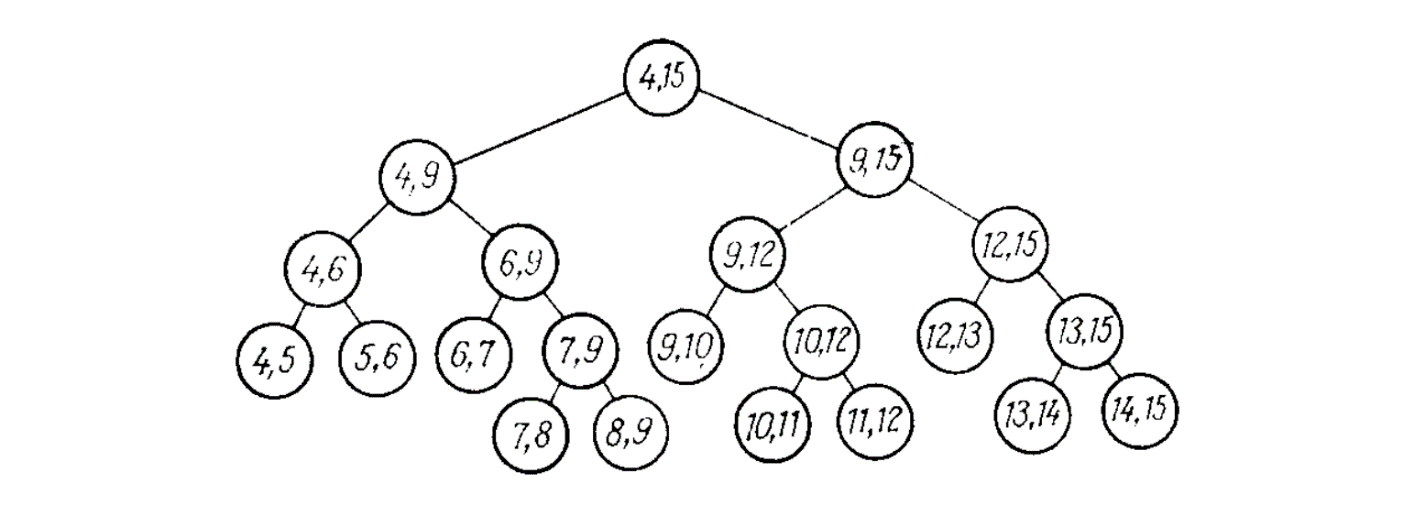


Рис.1.1 Дерево отрезков для интервала [4,15].

Пример структуры данных узла:

struct SegmentTreeNode

{

SegmentTreeNode\* leftChild;

SegmentTreeNode\* rightChild;

long l;

long r;

int C;

Status status;

}

Рекурсивная инициализация дерева:

function buildSegmentTree(l, r):

subtree = new SegmentTreeNode(l, r)

if (r - l) > 1:

subtree.leftChild = buildSegmentTree(l, (l + r) / 2)

subtree.rightChild = buildSegmentTree((l + r) / 2, r)

return subtree

Необходимо инициализировать массив равный длине отрезка , тогда сложность .

Вставка отрезка в дерево:

function insert(left, right):

if left <= l and r <= right:

C += 1

else:

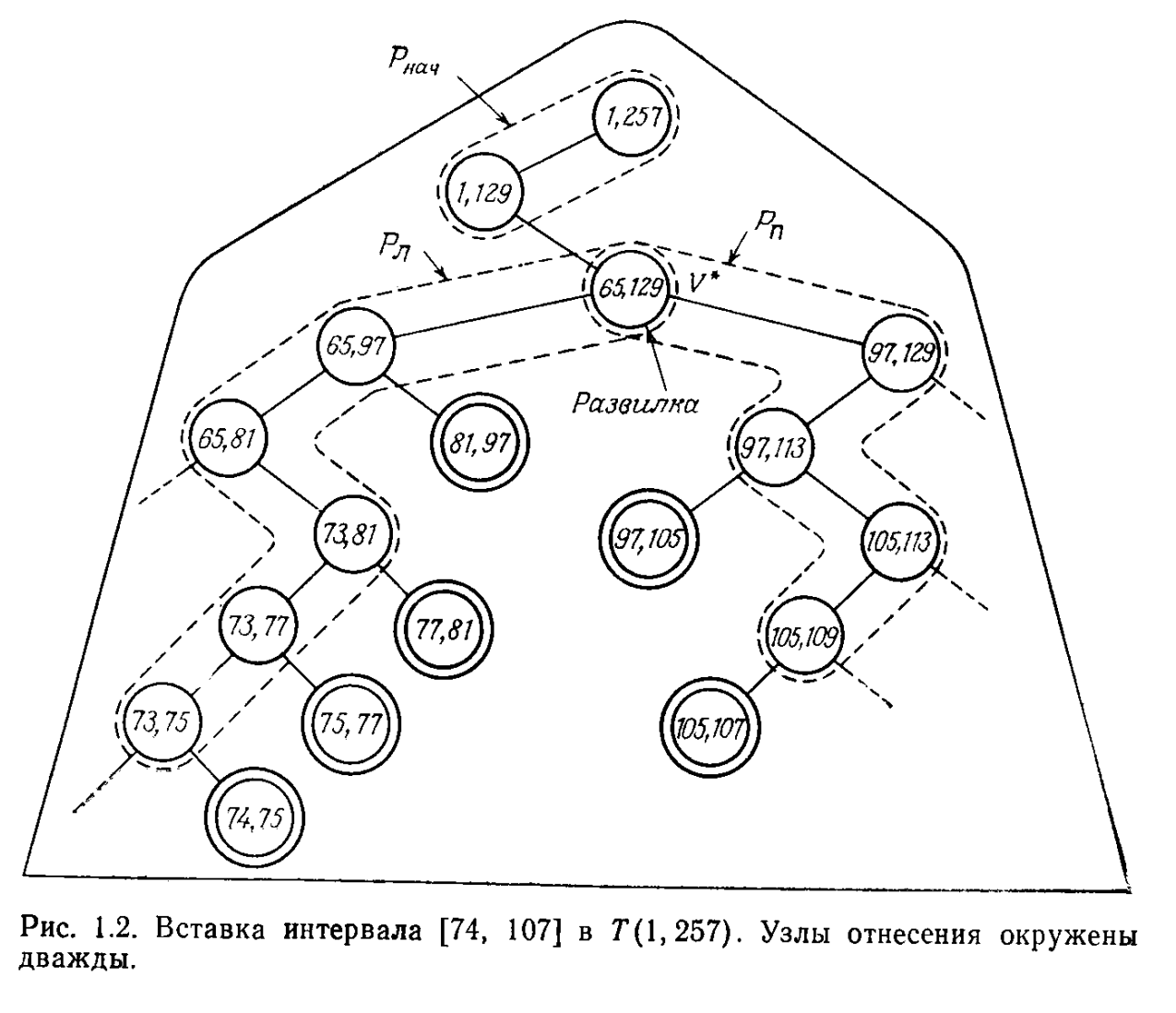
if left < (l + r) / 2 and leftChild:

leftChild.insert(left, right)

if (l + r) / 2 < right and rightChild:

rightChild.insert(left, right)

update()



Сложность вставки и удаления .

Удаление отрезка из дерева:

function remove(left, right):

if left <= l and r <= right:

C -= 1

else:

if left < (l + r) / 2 and leftChild:

leftChild.remove(left, right)

if (l + r) / 2 < right and rightChild:

rightChild.remove(left, right)

update()

Теперь обратимся к алгоритму поиска контура объединения, представленному в работе [9]. Для его понимания необходимо ввести следующие понятие.

Алгоритм плоского заметания, также известный как алгоритм сканирования, основывается на последовательном сканировании плоскости по горизонтальным линиям или сканирующей прямой. Этот алгоритм определяет взаимное расположение объектов в процессе сканирования и выполняет соответствующие операции с ними.

Основная идея алгоритма заключается в следующем: на первом шаге, с использованием дерева отрезков и плоского заметания, формируется множество вертикальных рёбер контура. В этом процессе решается задача пересечения и объединения интервалов, образованных вертикальными сторонами исходных прямоугольников. После формирования вертикальных рёбер, прямым методом вычисляются горизонтальные рёбра контура.

Для каждого ребра создается тройка, состоящая из абсциссы, конца ребра и ссылки на противоположный конец. Эти тройки затем сортируются лексикографически по возрастанию. В конечном итоге, за один проход алгоритма генерируются все горизонтальные рёбра.

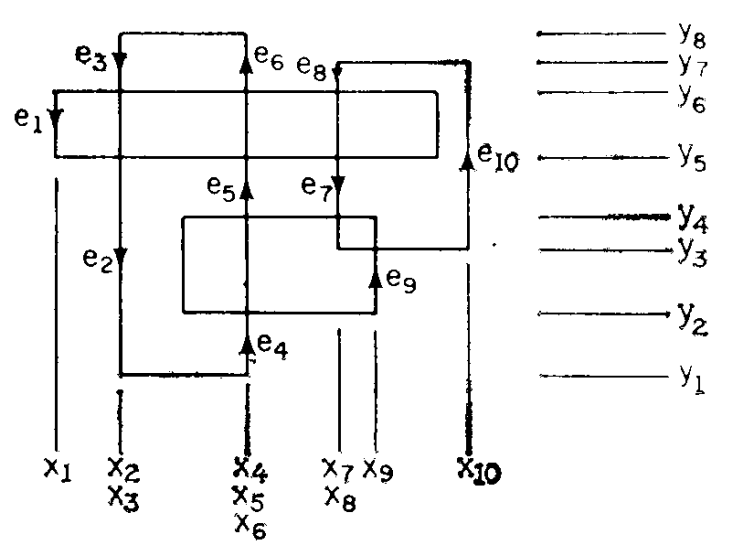
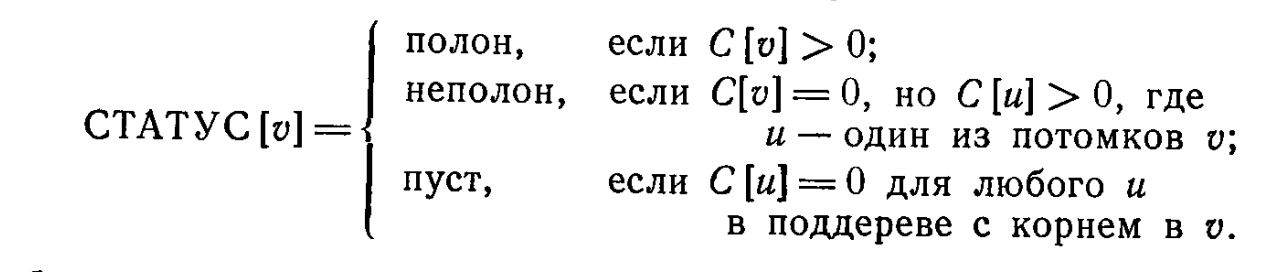
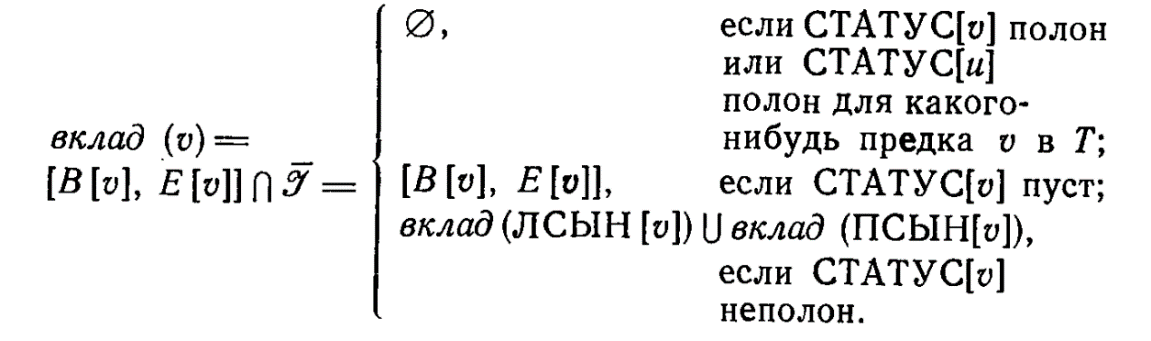


Рис.1.3 Пример контура объединения прямоугольников.

Для реализации алгоритма введём классификацию меры пересечения текущего сечения (объединения всех отрезков вертикальных рёбер) и узла дерева отрезков.



Также вводится вклад узла в сечение:



Процесс объединения смежных интервалов реализуется с использованием внешнего стека. Верхний элемент стека всегда содержит верхний конец последнего вставленного интервала. Если нижний конец следующего интервала, предназначенного для добавления в стек, совпадает с верхним элементом стека, последний удаляется из стека. Затем добавляется верхний конец предстоящего интервала, осуществляя объединение интервалов.

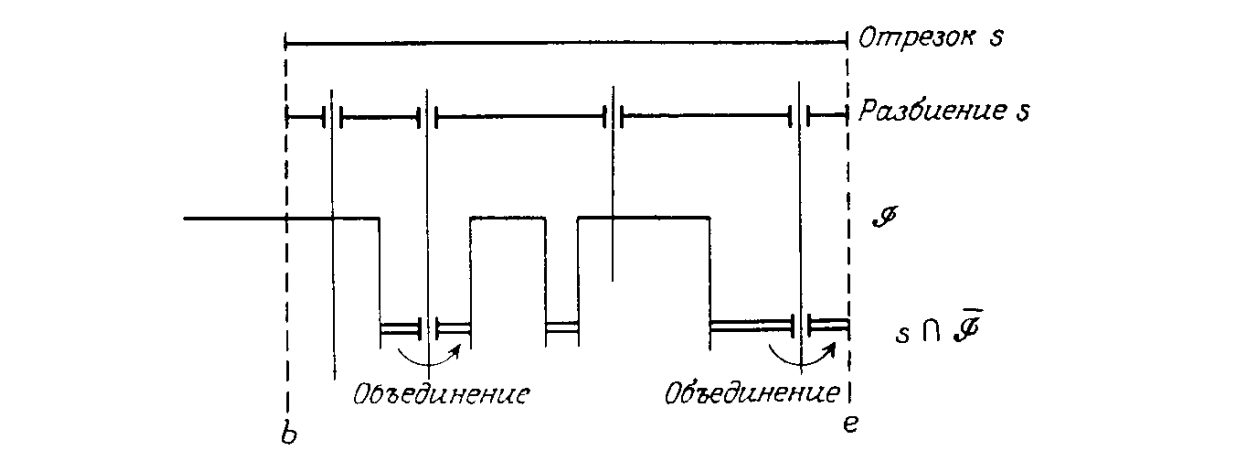
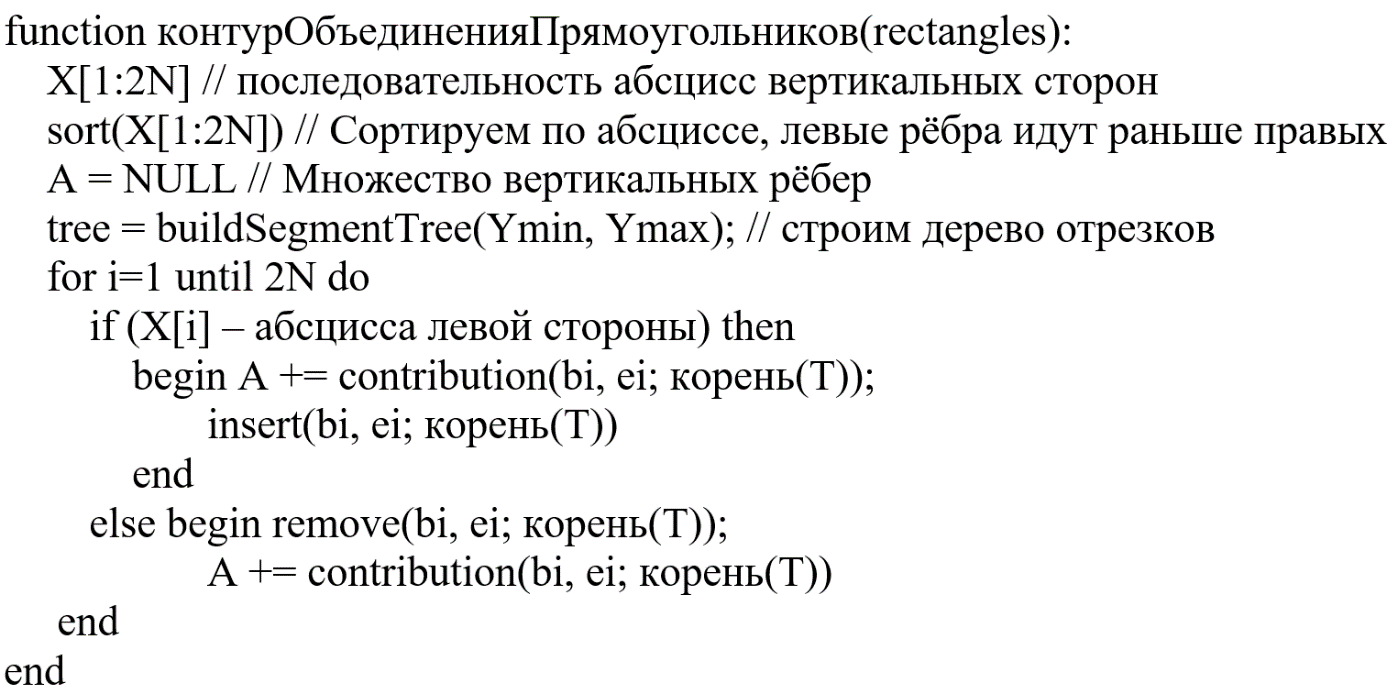


Рис.1.4 Объединение смежных интервалов при вычислении вклада вертикальной стороны s в сечение.

Теперь может быть представлен псевдокод алгоритма для поиска вертикальных рёбер контура объединения прямоугольников. Его достаточно, чтобы получить угловые точки контура.



Определим вспомогательную функцию для обновления актуального состояния узлов для дерева отрезков:

function update():

if C > 0:

status = Status.FULL

else:

if leftChild and rightChild:

if leftChild.status == rightChild.status == Status.EMPTY:

status = Status.EMPTY

else:

status = Status.PARTIAL

else:

status = Status.EMPTY

Будем использовать внешний стек для определения вертикальных рёбер, его содержание возвращается при вызове функции ВКЛАД (contribution).

function contribution(left, right, stack):

if status != Status.FULL:

if left <= l and r <= right and status == Status.EMPTY:

if stack and l == stack[-1]:

stack.pop()

else:

stack.append(l)

stack.append(r)

else:

if leftChild and left < (l + r) / 2:

leftChild.contribution(left, right, stack)

if rightChild and (l + r) / 2 < right:

rightChild.contribution(left, right, stack)

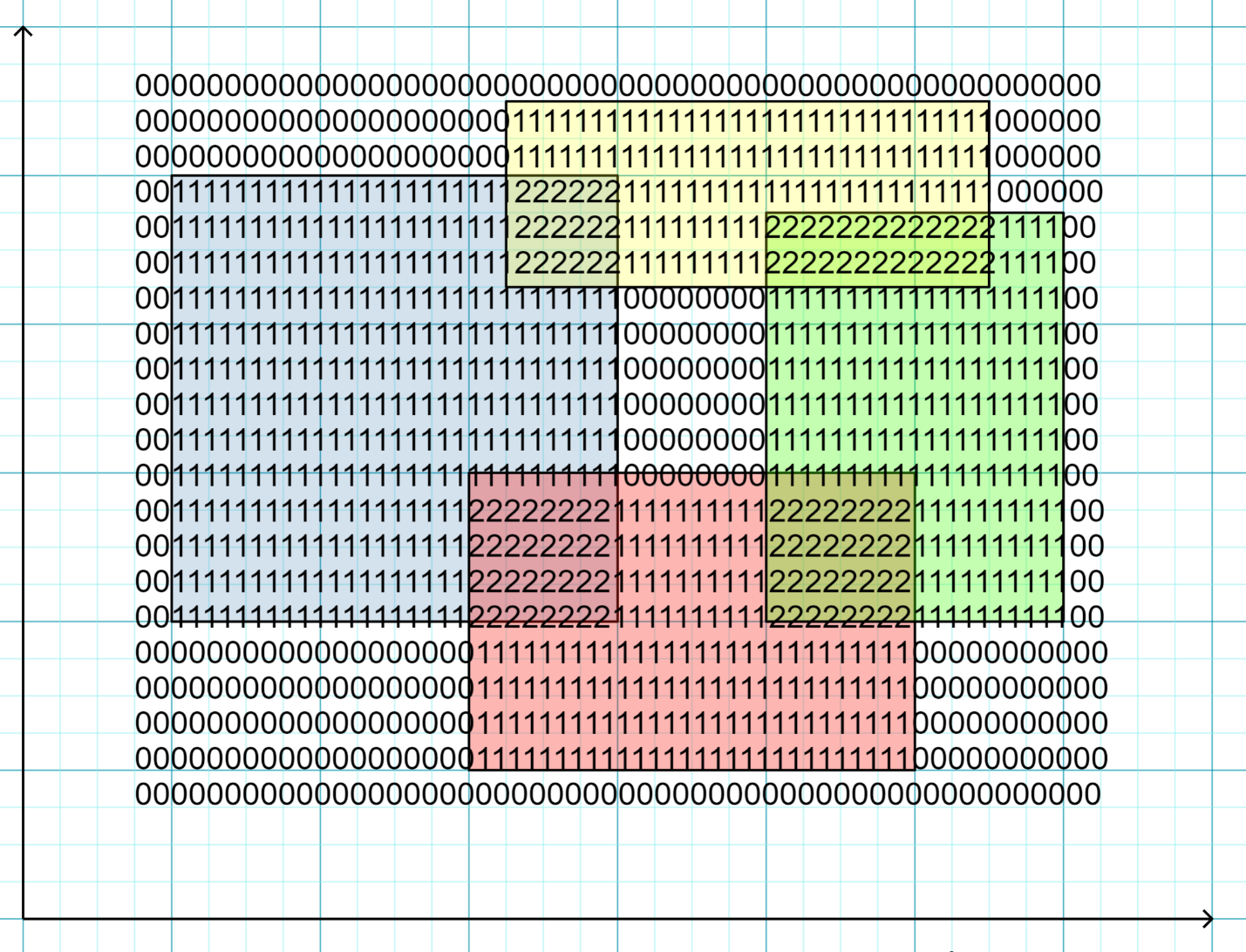


Рис. 1.5 Пример покрытия объединения прямоугольников параметром C дерева отрезков при работе алгоритма ПЗ.

**Теорема.** Сложность получения контура объединения N изотетичных прямоугольников, состоящего из p рёбер . [9]

В работе [18] показано, что эта сложность для поиска контуров квадратов сводится к

## 2.5 Оптимизация алгоритма для построения контура с помощью дерева отрезков

Полученный алгоритм имеет хорошую алгоритмическую сложность. Тем не менее, после реализации алгоритма на C++ в соответствии с псевдокодом, с помощью программного обеспечения Intel VTune Profiler был проведен анализ производительности. Результаты исследования показали, что алгоритм затрачивает более 20% времени на операции с памятью.

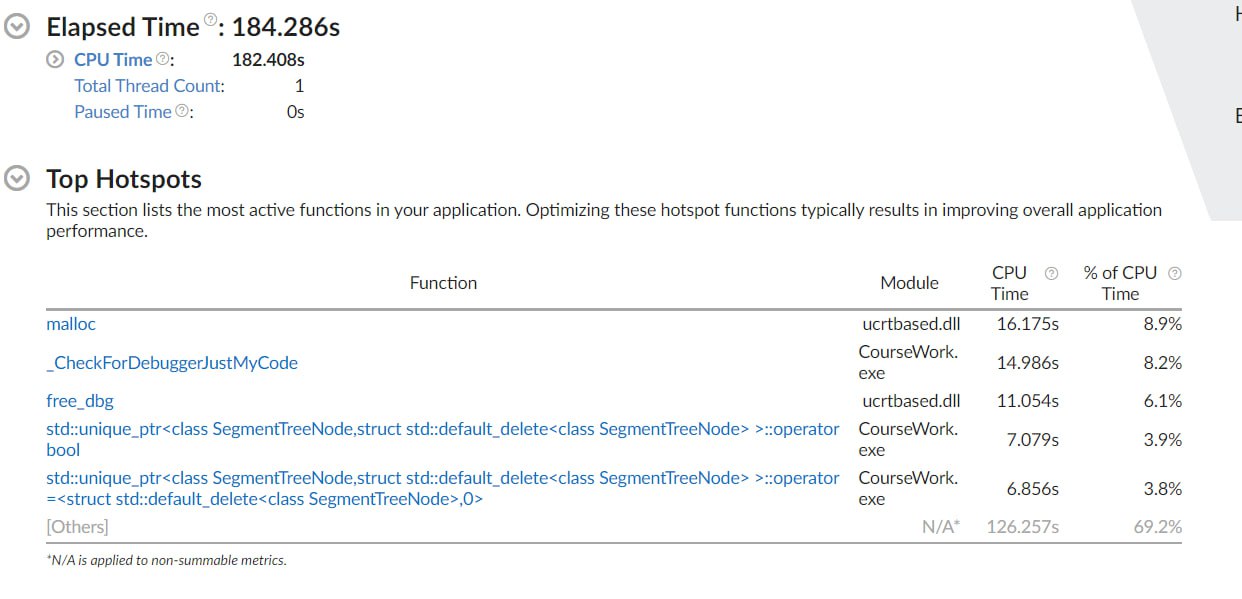


Рис. 1.6 Пример анализа производительности алгоритма поиска приближенного решения задачи размещения.

Это в основном связано с тем, что для построения каждого контура мы создаем дерево отрезков и затем очищаем его. Если мы не используем потенциал многопоточности, оптимизацией может быть создание дерева один раз и использование его для всех вызовов процедуры поиска контура прямоугольника.

Если для последующего контура максимальное и минимальное значения Y вертикального ребра превышают размеры корневого интервала, дерево должно быть обновлено путем расширения интервала. Таким образом, мы выделяем память один раз, и в худшем случае будем расширять дерево на каждой итерации, что эквивалентно созданию нового дерева с точки зрения обхода в глубину, но более эффективно с точки зрения операций выделения памяти.

## 2.6 Комбинаторное построение контура с помощью алгоритма плоского заметания

Для сравнения разработаем и реализуем комбинаторный алгоритм для поиска точек контура без использования дерева отрезков. Для этого введем следующие структуры данных:

struct Segment

{

int bottom;

int top;

}

struct ActiveSegment

{

Segment activeSegment;

vector<Segment> innerSegments;

}

За основу будет также взят алгоритм, основанный на плоском заметании, для поиска точек контура с помощью дерева отрезков. Однако, чтобы оптимизировать работу с памятью, мы будем хранить намного меньше информации. А именно, массив ActiveSegment и массив точек, которые образуют контур на шаге . Псевдокод для алгоритма может выглядеть следующим образом:

function контурОбъединенияПрямоугольников(rectangles):

X[1:2N] – последовательность абсцисс вертикальных сторон

sort(X[1:2N]) // Сортируем по абсциссе и левые рёбра идут раньше правых  
 activeSegments = NULL // Множество активных покрывающих отрезков

points = NULL; // Множество точек образовывавших контур

A = NULL; // Множество точек контура

for i=1 until 2N do

if (X[i] – абсцисса левой стороны) then

begin for j=1 until len(activeSegments) do

if (bi,ei) full in activeSegments[j] then

activeSegments[j].add(bi,ei)

break

if (bi,ei) in activeSegments[j] then

activeSegments[j].add(bi,ei)

update(activeSegment[j]) // границы сегмента

break

if (bi, ei) no in any activeSegments then activeSegments.add(bi,ei)

if (bi, ei) расширил какой-либо сегмент then пробуем слить сегменты;

end

else begin for j=1 until len(activeSegments) do

if (bi,ei) full in or in activeSegments[j] then

// Удаляем внутренний интервал

// При необходимости разделяем интервалы

end

if (X[i] != X[i+1]) then

// Обновляем список points и A

end

Алгоритм последовательно анализирует предварительно отсортированные абсциссы прямоугольников, всегда начиная с левых. Для каждой абсциссы он итерируется по списку активных сегментов, чтобы определить их взаимное расположение.

Если левая вертикальная грань не принадлежит ни одной активной области, она формирует новый сегмент. Если левая грань находится внутри активной области, она просто добавляется к сегменту. Если левая грань пересекает сегмент, она добавляется к сегменту, и границы обновляются. Если левая грань обновила границы какого-либо сегмента, то проверяется новое взаимное расположение сегментов и по необходимости они сливаются.

Когда мы достигаем правой грани, мы ищем область, к которой она относится, и она удаляется из внутренних интервалов. Затем следует затратная операция построения новых сегментов из оставшихся внутренних сегментов, что приводит либо к обновлению исходной области, либо к набору новых областей с соответствующими им внутренним интервалам.

Финальным шагом является определение списка текущих точек контура. При изменении абсциссы определяется список точек, которые были удалены или добавлены. В основном мы ищем список точек, которые стали контурными, и те, которые были точками контура, но больше ими не являются. Особое внимание следует уделить поиску точек при слиянии и разделении сегментов.

Не трудно заметить, что оценка наихудшего случая сложности алгоритма составляет из-за необходимости в слиянии и разделении активных областей. Одно из преимуществ алгоритма — его одинаковая эффективность как для целочисленных задач, так и для ограничений с плавающей точкой.

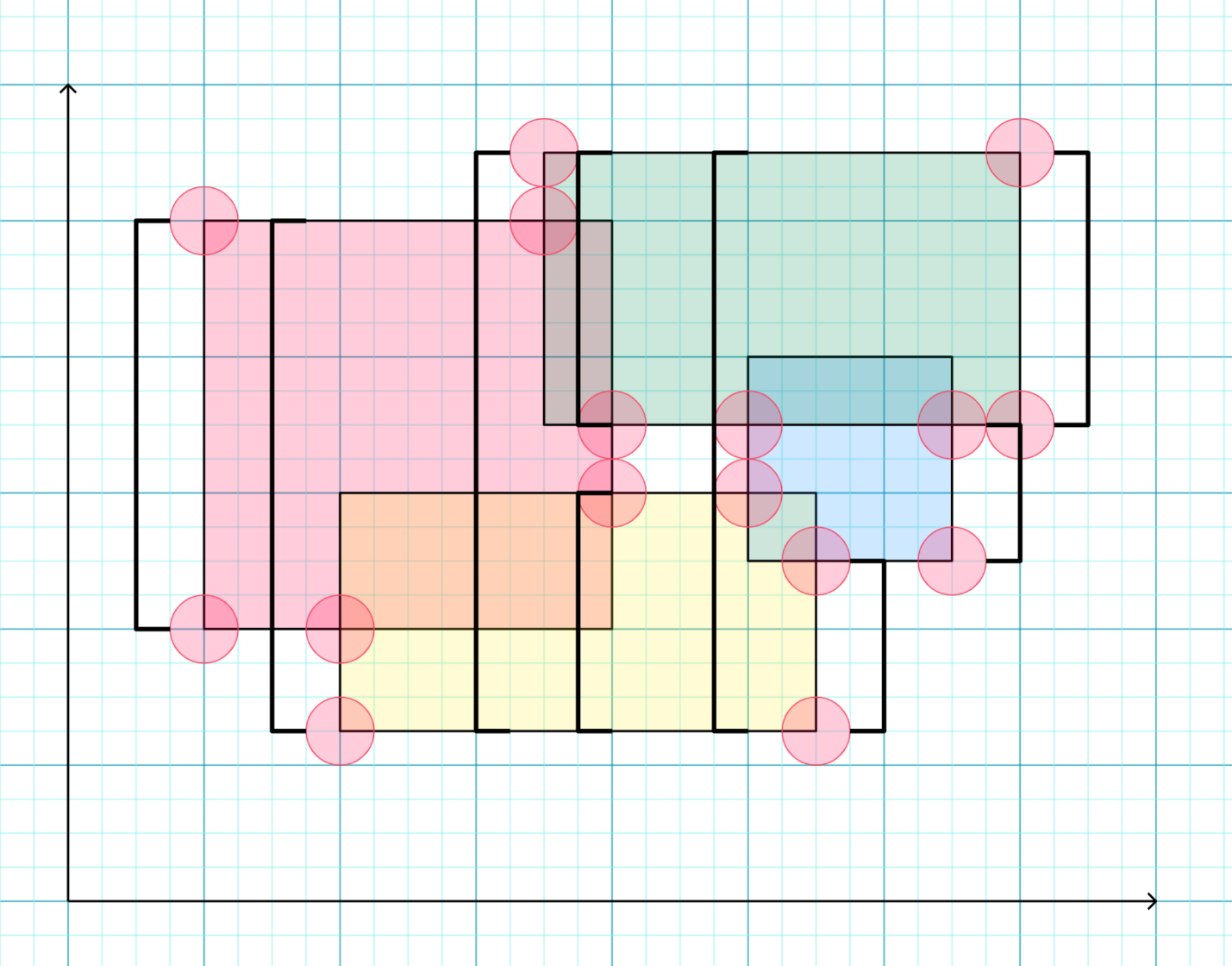


Рис. 1.7 Пример покрытия контура сегментами и определения точек контура.

## 2.7 Поиск точек размещения с использованием генетических алгоритмов.

Для сравнения разработаем и реализуем генетический алгоритм (ГА) для поиска точек контура объединения прямоугольников. Прежде всего, введем понятие генетических алгоритмов.

Генетические алгоритмы — это методы оптимизации и поиска, вдохновленные принципами эволюции и генетики. ГА основаны на концепции естественного отбора и генетической мутации. Алгоритм начинается с создания случайной популяции особей, каждый из которых представляет потенциальное решение оптимизационной задачи. Затем новые поколения создаются путем комбинирования и изменения особей с помощью операций скрещивания и мутации.

Каждый индивидуум в популяции представлен генетическим кодом, или хромосомой, часто представляемым в виде битовой строки. Часто используются операции скрещивания, которые объединяют хромосомы двух родительских особей, чтобы создать потомство. Мутация, в свою очередь, случайным образом изменяет часть генетического кода.

Процесс эволюции в ГА основан на принципе "выживания сильнейших". Качество индивидуумов оценивается с помощью функции приспособленности, которая измеряет степень соответствия каждого индивидуума требованиям оптимизационной задачи. Особи с более высокой приспособленностью имеют больше шансов передать свои гены следующему поколению.

Процесс эволюции продолжается до достижения определенного критерия остановки, такого как достижение определенного значения функции приспособленности или истечение заранее заданного числа поколений. В конечном итоге, лучший индивидуум, обладающий наиболее оптимальным решением, выбирается в качестве результата ГА.

ГА обладают способностью обнаруживать глобальные оптимумы и эффективно справляться с проблемами большого масштаба и сложными ограничениями. При использовании генетических алгоритмов важно правильно настроить параметры алгоритма, такие как размер популяции, вероятности скрещивания и мутации, а также выбрать подходящую функцию приспособленности.

Генетический алгоритм (ГА) может быть использован для прямого решения задачи, определенной уравнениями (1)-(4). Чтобы решить эту проблему, может быть использована следующая стратегия: каждый ген обозначает положение объекта на плоскости и представляется в виде пары действительных чисел (x, y). Хромосома представляет собой набор генов для всех объектов, которые должны быть размещены. Предположим, что функция приспособленности является суммой результатов последовательного размещения генов. Если объект может быть размещен в позиции гена, соблюдая все ограничения, к окончательному значению приспособленности добавляется одно очко. Через правильный выбор, скрещивание и мутации можно достичь приближенного решения задачи размещения, которое удовлетворяет всем ограничениям.

Однако реализации эффективного генетического алгоритма для прямого решения задачи следует посвятить отдельное исследование. В контексте приближённого решения с заданной точностью будем использовать ГА для поиска точек допустимого размещения объектов.

Построим простую модель поиска ГА. Определим ген как положение точки в пространстве, например, x. В этом случае хромосома состоит из двух генов. Вместе с приспособленностью хромосома также будет иметь параметр «доступности», указывающий, что положение точки удовлетворяет всем ограничениям. Простым примером функции приспособленности может быть сумма функций определяющих выполнения ограничений для всех прямоугольников. Приспособленность увеличивается на 1, если текущая точка находится вне прямоугольника, на 2, если она находится на стороне прямоугольника, и на 3, если она является угловой точкой ограничения. Однако приспособленность не может превышать , где - количество прямоугольников в ограничении. Эта функция обеспечивает сходимость в допустимой области, с фокусом на поиске угловых точек. Однако эта функция имеет очевидный недостаток, так как учитываются все прямоугольники, то преимущество будут иметь не только угловые точки, но и точки лежащее одновременно на нескольких гранях прямоугольников. Селекция в данном случае должна быть направлена на отбор и сохранение лучших точек. При кроссовер происходит размещение потомка в позицию, связанную с средневзвешенным соотношением приспособленности родительских особей. Таким образом, потомок будет ближе к лучшему кандидату. Функция поиска контура должна возвращать набор “доступных” точек с максимальной приспособленностью. Основой является стандартный ГА с особями и шагами.

Идея алгоритма заключается в том, что со временем популяция будет сходиться к допустимым точкам контура прямоугольника, и мы сформируем наше приближенное решение из них. По мере увеличения количества точек и шагов вероятность сходимости к оптимальному решению повышается. Из-за высокой алгоритмической сложности для большого числа шагов и особей, необходимых для нахождения оптимального решения - в контексте конкретной реализации это, этот алгоритм не был подробно изучен из-за альтернатив с более низкой сложностью. Однако такой алгоритм может быть применен для быстрой оценки метрики , даже в сложных задачах.

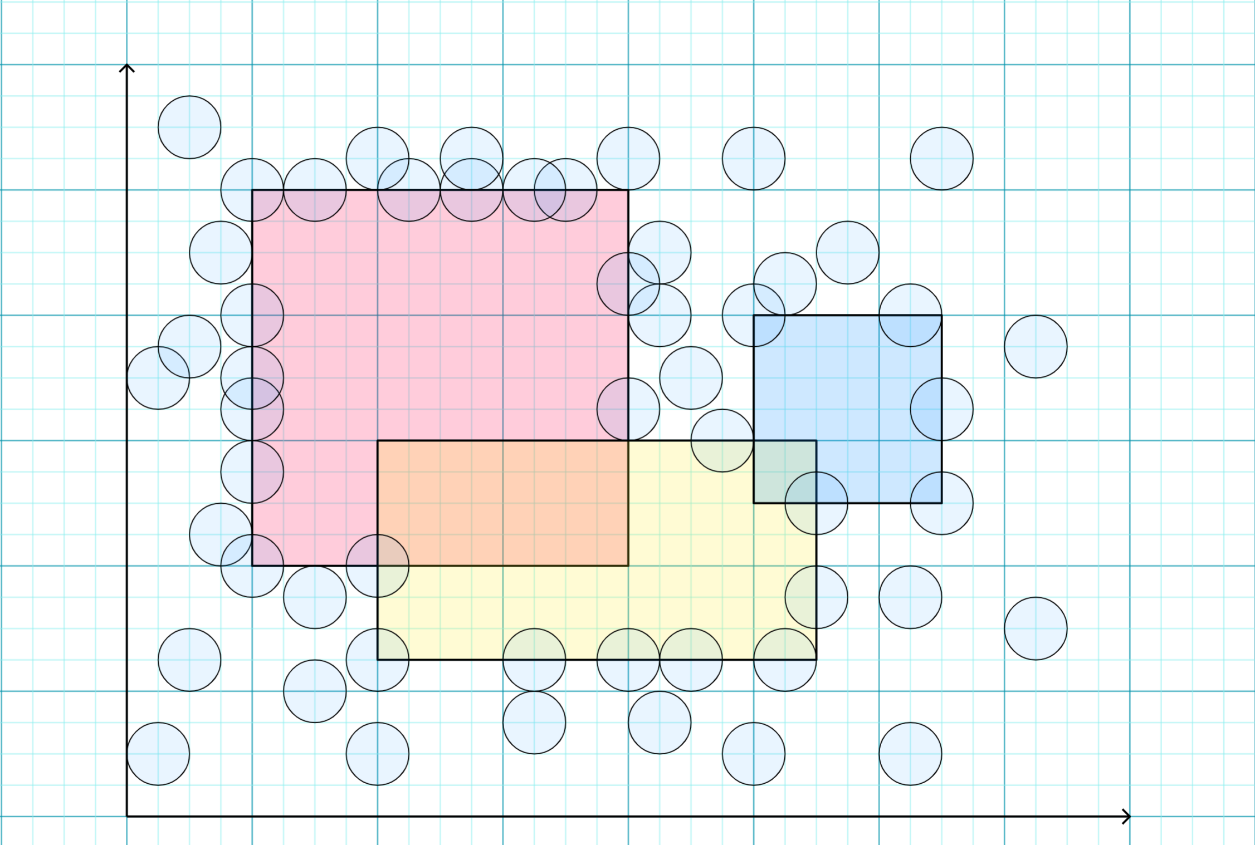


Рис. 1.7 Пример покрытия контура популяцией.

# ПРИБЛИЖЕННОЕ РЕШЕНИЕ ЗАДАЧИ О РАЗМЕЩЕНИЯХ С ПОМОЩЬЮ ЦЛП

## 3.1 Описание подхода

Целочисленное линейное программирование (ЦЛП) является разделом математического программирования, который занимается решением оптимизационных задач с линейной целевой функцией и ограничениями, требующими, чтобы переменные принимали только целочисленные значения, что является его основной особенностью.

Однако требования целочисленности приводит к значительным изменениям в алгоритмах и методах решения задач линейного программирования, также это приводит к более сложной вычислительной задаче в отличие от задач линейного программирования с непрерывными переменными.

Для решения задач ЦЛП используются различные методы и алгоритмы, такие как метод ветвей и границ, метод северо-западного угла и другие. В зависимости от размера и сложности задачи, а также требуемой точности решения, выбирается наиболее подходящий метод.

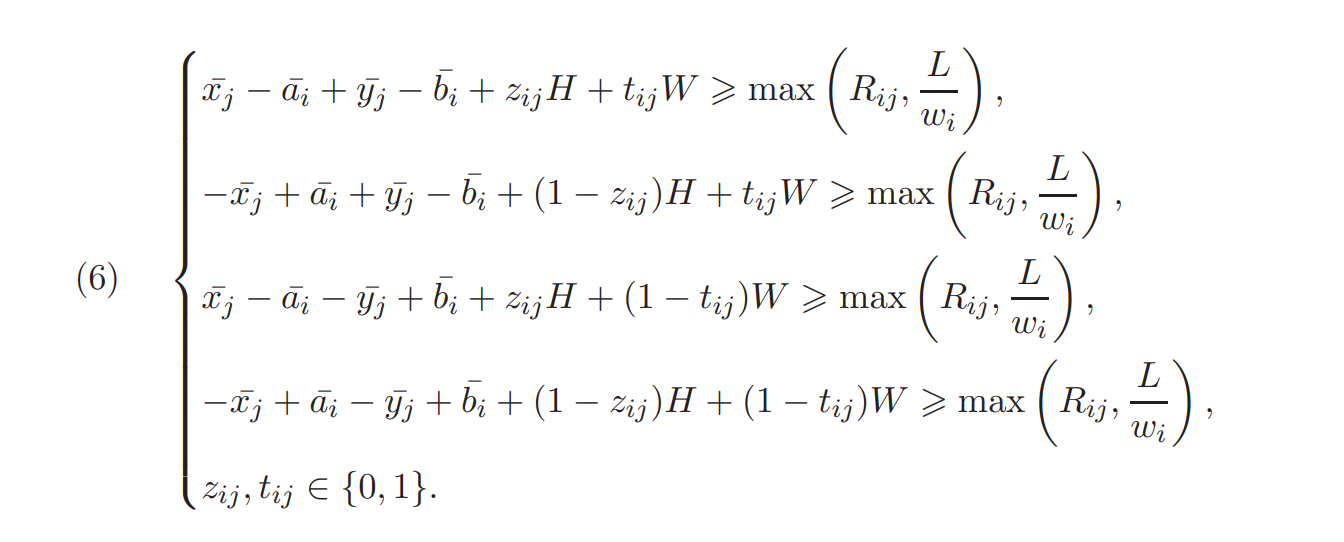
## Формулировка модели

Модель ЦЛП для данной задачи основывается на свойстве, связывающем решения задачи в метрике и , следует, что при рассмотрении оптимальных решений и в соответствующих метриках, справедливы следующие соотношения:

*T*( )= ,

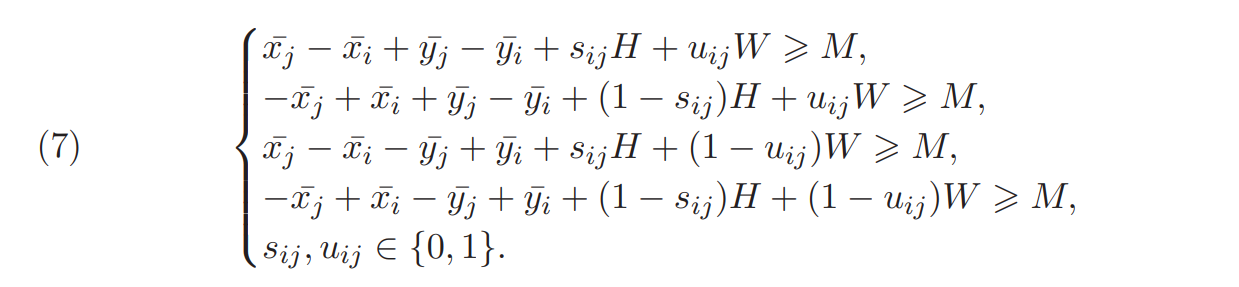
= T−1(), где T – отображение вида .

Запишем задачу (1)–(4) в метрике , где (,) представляют собой T−1 (, ), а (,) представляют T−1 (,), где *i* ∈ {1,…, *n*} и *j* ∈ {1, …, *p*}. Введем константы *W* и *H*, которые являются достаточно большими положительными значениями, например, *W*=*H*=2(*A*+*B*). Для описания взаимного расположения объектов введем булевы переменные и , где =1, если >= , и = 0 в противном случае; а = 1, если ≤ , и = 0 в противном случае. Ограничения (2) для ∈*O* и ∈*V* могут быть записаны следующим образом:

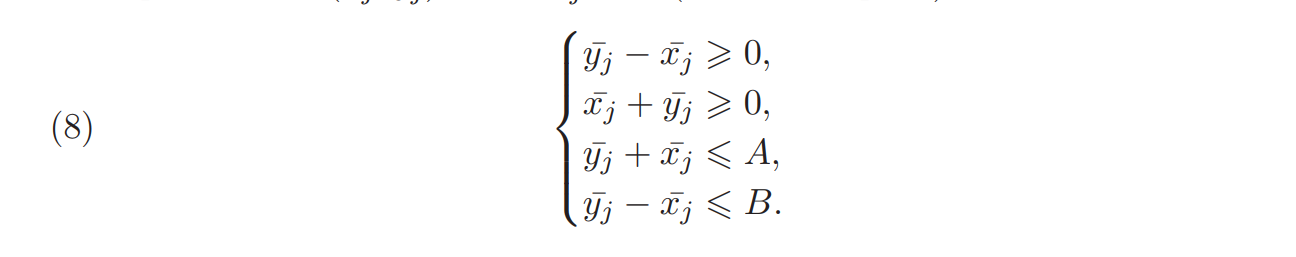


Также мы вводим булевы переменные и для определения взаимного расположения объектов. Переменная равна 1, если ≤ , и равна 0 в противном случае. Аналогично, переменная равна 1, если ≤ , и равна 0 в противном случае.

Ограничения (3) для ,∈ *V*:



Ограничения после поворота (,) ∈ *S* и ∈*O*:



Модель целочисленного линейного программирования (ЦЛП) (1), (6)–(8) может быть получена, если каждое из ограничений (6) будет представлено в виде двух неравенств. Для реализации будут использованы программные продукты Gurobi Optimization и IBM CPLEX Optimization Studio.

# ЭКСПЕРИМЕНТЫ

## 4.1 Применение методологии к задаче размещения двух объектов.

На языке программирования C++ был реализован комбинаторный алгоритм для поиска приближенного решения с заданной точностью, включающий в себя четыре алгоритма для выбора точек контура. Они представлены ниже:

* SegmentTree (ST) – это базовый алгоритм для поиска точек контура с использованием дерева отрезков.
* SharedSegmentTree (SST) – это ST, но с оптимизацией построения дерева отрезков.
* GeneticAlgorithm (GA) – это генетический алгоритм.
* CoveringSegments (CS) – это прямой комбинаторный алгоритм для поиска точек контура.

Для проведения вычислительных экспериментов использовались модель ЦЛП (1), (6)–(8) и программные продукты Gurobi Optimization с реализацией модели на языке программирования С++, а также IBM CPLEX Optimization Studio с реализацией модели на языке программирования OPL.

Для каждого набора параметров было проведено 50 экспериментов, после чего было вычислено среднее время. Для определения точности GA вычислялось отношение полученного суммарного значения L к сумме ответов, полученных при помощи SST.

Вычисления выполнены на Intel Core i7 9750H 2600 МГц.

Таблица 1 отражает отношение времени выполнения (в секундах) к количеству фиксированных объектов. Параметры задачи были следующими: и = ≤ 25 генерировались случайным образом. Параметры для GA были выбраны для максимально быстрого получения ответа:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n | CPLEX | Gurobi | SST | ST | GA | CS |
| 10 | 0,08 | 0,081 | 0,007 | 0,078 | 3,1 (84%) | 0,018 |
| 20 | 0,08 | 0,105 | 0,010 | 0,083 | 3,078 (85%) | 0,040 |
| 30 | 0,159 | 0,171 | 0,013 | 0,090 | 3,239 (83%) | 0,080 |
| 40 | 0,283 | 0,210 | 0,017 | 0,121 | 3,307 (78%) | 0,145 |
| 50 | 0,404 | 0,236 | 0,022 | 0,099 | 3,45 (83%) | 0,196 |

Табл. 1

## 4.2 Применение методологии к задаче с большим числом размещаемых объектов

Таблица 2 отображает отношение времени выполнения (в секундах) к количеству размещаемых объектов. Параметры задачи были следующими: и = ≤ 25 генерировались случайным образом. Параметры для GA были выбраны для максимально быстрого получения ответа:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| p | CPLEX | Gurobi | SST | ST | GA | CS |
| 3 | 0,306 | 0,213 | 0,014 | 0,085 | 3,896 (87%) | 0,110 |
| 4 | limit | 0,276 | 0,017 | 0,094 | 5,127 (82%) | 0,106 |
| 5 | limit | 0,432 | 0,031 | 0,225 | 3,239 (83%) | 0,152 |
| 6 | limit | 0,635 | 0,043 | 0,344 | 7,067 (84%) | 0,170 |
| 7 | limit | 0,648 | 0,048 | 0,278 | 4,44 (78%) | 0,259 |
| p=8 n=10 | 80+ | 1,1 | 0,058 | 0,168 | 10,936 (90%) | 0,098 |

Табл. 2

Для CPLEX начиная с начали действовать ограничения бесплатной версии.

Анализ результатов экспериментов позволяет сделать некоторые выводы о производительности и эффективности рассматриваемых алгоритмов. Программный продукт IBM CPLEX Optimization Studio, к примеру, не является оптимальным выбором для решения данной задачи, что обусловлено как ограничениями использования, так и низкой скоростью работы для задач с большим количеством размещаемых объектов. Это становится особенно очевидным при сравнении его производительности с таковой у Gurobi и комбинаторного алгоритма, несмотря на многопоточность, которую предлагает IBM CPLEX Optimization Studio. Рассмотренные в данном исследовании комбинаторные алгоритмы используют всего один поток выполнения.

Gurobi Optimization чаще всего демонстрировал высокую скорость работы, однако он также уступал комбинаторным алгоритмам, а также он имеет ограничения в бесплатной версии.

Комбинаторные алгоритмы SST/ST, основанные на использовании дерева отрезков для построения контура, показали наиболее эффективные результаты. Версия SST с оптимизацией в среднем оказалась в 8 раз быстрее алгоритма из оригинальной работы. Стоит отметить, что есть потенциал для дальнейшего ускорения этого алгоритма путем использования многопоточности.

Алгоритм CS оказался средним по времени работы и весьма сложным для реализации, требуя особого внимания к деталям.

Как и ожидалось, генетический алгоритм (GA) оказался самым медленным. Но при этом следует учитывать, что существуют возможности для создания более эффективной модели и реализации, при которых скорость и точность GA могут быть приемлемыми. Для задач с большим количеством объектов размещения GA может быть использован для быстрой оценки параметров, что, вероятно, окажется более эффективным, чем использование продукта IBM CPLEX Optimization Studio.

Итак, можно сделать вывод, что для решения задачи размещения предпочтительно использовать либо оптимизированный комбинаторный алгоритм, использующий дерево отрезков для поиска контура, либо полную версию программного продукта Gurobi Optimization, который также решает подобные задачи достаточно эффективно.

# ЗАКЛЮЧЕНИЕ

В рамках данного исследования была осуществлена детальная работа над максиминной задачей размещения объектов в прямоугольной области на плоскости, с учётом ограничений, вносимых фиксированными объектами. В контексте поставленной задачи были учтены ограничения на минимальное расстояние как между фиксированными и размещаемыми объектами, так и между самими размещаемыми объектами. Дополнительно рассмотрен вариант, при котором минимальные расстояния зависят от типа размещаемых объектов.

В ходе исследования были разработаны и протестированы различные модификации алгоритма, способного находить приближенное решение с заданной точностью. Был проведён вычислительный эксперимент с целью сравнения эффективности комбинаторного алгоритма (с четырьмя модификациями), исследующего приближенное решение с заданной точностью, с возможностями продустов продукт IBM CPLEX Optimization Studio и Gurobi Optimization для решения задач целочисленного линейного программирования.

Результаты эксперимента свидетельствуют о том, что с использованием разработанного алгоритма можно достигать решений, близких к оптимальным по значениям целевой функции. При этом затраты вычислительных ресурсов значительно ниже по сравнению с использованием продуктов IBM CPLEX Optimization Studio и Gurobi Optimization.

Полученные данные подтверждают универсальность и эффективность анализируемого алгоритма, а также свидетельствуют о его потенциале для применения в системах для размещения объектов на плоскости.

Ссылка на код работы: <https://github.com/danek0100/Aproximation-Solution-of-the-Maximin-Assignment-Problem>

# СПИСОК ЛИТЕРАТУРЫ

[1] Ageev, A.A., Gimadi, E.Kh., Kurochkin, A.A. (2003). Polynomial algorithm for solving the placement problem on chains with equal production capacities of enterprises. Moscow: University Press.

[2] Brimberg, J., Mehrez, A. (1994). Multi-Facility Location Using a Maximin Criterion and Rectangular Distances. New York: Springer.

[3] Fowler, R. (1981). Optimal packing and covering in the plane are NP-complete. New York: ACM.

[4] Hwang, R.Z., Lee, R.C.T., Chang, R.C. (1993). The slab dividing approach to solve the Euclidean P-Center problem. New York: Springer.

[5] Katz, M.J., Kedem, K., Segal, M. (2002). Improved Algorithms for Placing Undesirable Facilities. Amsterdam: Elsevier.

[6] Kochetov, Y.A., Pashchenko, M.G., & Plyasunov, A.V. (2010). On the complexity of local search in the p-median problem. Dordrecht: Springer.

[7] Padmanabhan, S. (2021). Optimal placement of public electric vehicle charging stations using deep reinforcement learning. ArXiv.

[8] Panyukov, A.V., Pelzwerger, B.V. (1990). Polynomial algorithms for the finite Weber problem on a tree network. Amsterdam: Elsevier.

[9] Preparata, F., Shamos, M. (1988). Computational geometry. Introduction. Moscow: Mir.

[10] Simpson, T. (1776). The Doctrine and Application of Fluxions. London: Nourse.

[11] Tamir, A. (2006). Locating Two Obnoxious Facilities Using the Weighted Maximin Criterion. Amsterdam: Elsevier.

[12] Toussaint, G. (1983). Computing largest empty circles with location constraints. Ottawa: National Research Council of Canada.

[13] Weber, A. (1922). Über den Standort der Industrien. Tübingen: J.C.B. Mohr.

[14] Weber, A. (1929). Theory of the Location of Industries. Chicago: University of Chicago Press.

[15] Zabudskii, G.G. (2006). Model Building and Location Problem Solving in a Plane with Forbidden Gaps. New York: Springer.

[16] Zabudskii, G.G., Markhotskaya, N.V. (2008). Solving the Maximin Placement Problem on the Plane with Minimum Admissible Distances. In XIV Baikal International School-Seminar "Optimization Methods and Their Applications". Irkutsk: ISE SB RAS.

[17] Zabudskii, G.G., Amzin, I.V. (2012). Search Region Contraction of the Weber Problem Solution on the Plane with Rectangular Forbidden Zones. New York: Springer.

[18] Zabudskii, G.G., Koval, A.A. (2014). Searching for a Solution with a Given Accuracy for the Maximin Placement Problem on the Plane. Irkutsk: ISE SB RAS.

[19] Zergane, S., Smaili, A. (2018). Optimization of wind turbine placement in a wind farm using a new pseudo-random number generation method. Amsterdam: Elsevier.

# ПРИЛОЖЕНИЕ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод | RMSE | MAPE | DA | TIME (ms) |
| Native Repeat | 30.49 | 1.80% | 46.81% | 0.05 |
| SMA | 66.76 | 4.31% | 48.84% | 1 |

Таблица 1. Усреднённые результаты по 10 тикерам для одномерных временных рядов.