Санкт–Петербургский государственный университет

Лазарев Александр Николаевич

Выпускная квалификационная работа

Алгоритм обнаружения спуфинг-манипуляций на фондовом рынке

Уровень образования: бакалавриат
Направление «Прикладная математика, фундаментальная информатика и программирование»
Кафедра "Математической теории игр и статистических решений"

Научный руководитель:
кандидат физ.-мат. наук, доцент кафедры математической теории игр и статистических решений
Зенкевич Н. А.

Санкт-Петербург
2021 г.

Содержание

Введение

В современных реалиях больших денег финансового рынка, когда существуют огромные технические возможности использовать торговые алгоритмы и помощь роботов, проблема использования недобросовестных техник в своих интересах стоит наиболее остро. Безнаказанность такого поведения может иметь катастрофические последствия, влияя на экономики целых стран, поэтому отслеживание и борьба с манипуляциями на рынке важны и актуальны.

В рамках моей научно-исследовательской работы будет рассмотрена стратегия спуфинга как вида манипуляции на фондовом рынке. После теоретического изучения этой стратегии будет выполнена практическая симуляция спуфинга на биржевых данных.Далее эти результаты будут использованы в качестве обучающей выборки для разработанного алгоритма обнаружения спуфинг-манипуляций.

После подбора параметров алгоритм обнаружения пройдет тесты на реальных данных и, возможно, ему удастся обнаружить манипуляцию на фондовом рынке РФ уже на стадии прототипа.

1. Манипуляции на фондовом рынке

1.1 Актуальность проблемы манипуляции

Рынок ценных бумаг является неотъемлемым институтом во всех развитых государствах и множество игроков, начиная от физических лиц и заканчивая крупными компаниями и государством, выходят на него с целью извлечь выгоду. История рынка ценных бумаг насчитывает более 400 лет, и буквально с момента его появления участники рынка и государство столкнулись с проблемами применения мошеннических схем с целью извлечения наибольшей прибыли на финансовых рынках – проблемами манипуляции.

На сегодняшний день сложно дать четкое определение понятия манипуляции на рынке из-за размытости в юридических и экономических трактовках, а также из-за широкого разнообразия манипулятивных стратегий и схем. Но все определения сводятся к тому, что манипулятор стремится своими действиями повлиять на формирование спроса и предложения и получить выгоду за счет других игроков. Такие действия могут иметь необратимые последствия, вплоть до краха рынка и экономики государства. Поэтому существуют специальные институты, государственные и биржевые, которые отслеживают манипуляции и строго наказывают за подобное мошенничество. К сожалению, эта борьба идет очень тяжело вследствие размытости определения манипуляции в юридических актах и обезличенности данных о биржевых сделках.

Тут необходимо отделить западные рынки от российского рынка. На американском рынке последнее десятилетие идет усиленная борьба с манипуляцией на рынке активов. Например, в августе 2018 года американская комиссия по торговле товарными фьючерсами (CFTC) оштрафовала Deutsche Bank на 30 млн долларов за манипуляцию ценами на золото и серебро с февраля 2008 г. по сентябрь 2014 г. во время торговли фьючерсами. Комиссия доказала, что дочерним предприятием банка была успешно реализована нелегальная схема торговли - спуфинг. Незадолго до этого за спуфинг был оштрафован еще один крупный банк: на этот раз швейцар-

ский UBS – там наказание было более суровым.

В России манипуляция является мошенничеством и классифицируется как уголовное преступление, но от этого громких дел и штрафов за манипуляцию на биржах не становится больше. Сложно поверить, что российские компании ведут исключительно чистую игру. При дальнейшем развитии экономики страны безнаказанность мошенничества на финансовых рынках будет нести все больше угрозы, поэтому вопрос выявления и пресечения манипуляции будет стоять особенно остро.

1.2 Основные определения

В этом разделе представлены базовые определения и понятия, которые лежат в основе дальнейших рассуждений.

Фондовый рынок [1] - это часть финансового рынка, где осуществляется привлечение и перераспределение капитала за счет выпуска и обращения ценных бумаг

Участники рынка ценных бумаг [2] - эмитенты (выпускающие ценные бумаги), инвесторы (заинтересованные в выгодном размещении средств), фондовые посредники (брокеры, дилеры, управляющие компании и банки), структуры осуществляющие организацию торговли (структуры биржевой торговли), структуры ведущие учет прав собственности на ценные бумаги и упрощающие процедуру перехода прав собственности (депозитарии, клиринговые организации, реестродержатели и трансфер-агенты).

Лимитная заявка [3]- это указание на приобретение или продажу конкретного количества лотов инвестиционного актива по заданной цене, которое трейдер даёт брокеру.

Подобные ордера представляют собой публичную оферту для каждого игрока на бирже. Как только найдётся участник торгов, которого устро-

[1] https://www.cbr.ru/securities-market/?utm-source=wutm-content=page

[2] https://fondovyj-rynok.ru/uchastniki-rynka-cennyh-bumag

[3] https://investoriq.ru/trejding/limitnaya-zayavka.html

ит сделанное предложение, выставленная заявка будет реализована.

Далее в работе под словом заявка будем понимать лимитную заявку.

Биржевой стакан [4] - сводная таблица, в которой указывается спрос и предложение на ценную бумагу в актуальное время. Таким образом, данный финансовый инструмент показывает ликвидность конкретного финансового инструмента на данный момент.

Рыночная стратегия [5] - набор правил, по которым трейдер или инвестор заключает сделки на бирже. Эти правила помогают выбрать активы для покупки или продажи, определить лучшую точку входа и рассчитать размер позиции.

Манипуляция рынком - получение дохода вследствие такого изменения цен, которое не продиктовано внешними по отношению к организованным торгам факторами, а является результатом исключительно распространения заведомо ложной информации, совершения операций или выставление заявок на совершение операций на организованных торгах, при наличии существенного отклонения цен от уровня, который бы сформировался без таких действий, а также при условии того, что лицо, получившее доход, действовало умышленно. [3]

1.3 Виды манипуляции на финансовых рынках

По методам манипуляции выделяют три основных вида:

1) манипуляции на основе торговли;

2) манипуляции на основе действий;

3) манипуляции на основе информации.

Манипуляция на основе торговли предполагает влияние на цену акций через торговлю. В манипуляциях на основе информации манипулятор распространяет ложную информацию или слухи о компании, чтобы

[4]https://www.finam.ru/education/likbez/chto-takoe-birzhevoiy-stakan-i-kak-s-nim-rabotat-20210326-17590/

[5]https://iamforextrader.ru/strategii-torgovli-i-investirovaniya-na-fondovom-rynke/

раздуть или понизить цену ее акций. Манипуляция на основе действия предполагает совершение действий, предпринимаемых с целью повлиять на стоимость компании.

Каждая группа манипуляций состоит из широкого разнообразия методов, особенно это касается манипуляций, базирующихся на торговле. Наиболее распространенные стратегии, которая являются рыночными манипуляциями, раскрыты в работе Д. Кумминга [1]. Далее в своей работе мы будем работать лишь с одной торговой стратегией спуфинга.

1.4 Определение спуфинг-манипуляции

Теперь надо разобраться, что такое спуфинг и какую стратегию мы пытаемся отследить. Перед тем как мы сможем перейти к самой стратегии, очень важно понимать, как устроены спрос и предложение на рынке и что такое биржевой стакан. Если рассматривать конкретный актив, то на его покупку и продажу всегда есть заявки, их перечень в виде цена – объем, градуированный по возрастанию, и представляет собой биржевой стакан.

Объем	Цена продажи	
500	105	
420	104	
300	103	
110	102	
50	101	
	99	40
	98	115
	97	500
	96	800
	Цена покупки	Объем

Рис. 1: Стакан заявок.

На рисунке 1 красным отмечены заявки на продажу, а зеленым заявки на покупку. Владельцы актива хотят продать как можно дороже, а покупатели, соответственно, купить дешевле. Разница между лучшей ценой

покупки и лучшей ценой продажи называется спредом. В данном примере спред составляет 2 доллара. Сделка происходит либо когда покупатель повысит цену, либо продавец ее снизит. Как только цены на покупку и продажу совпадут, произойдет сделка на совпадающий объем и исполненные заявки исчезнут. Если же одна из заявок, при этом, не будет исполнена полностью, то произойдет уменьшение объема заявки на приобретенное количество лотов. В целом, так функционирует весь рынок. Есть еще, конечно, много тонкостей касательно видов заявок и условий их исполнения, но не будем сейчас на этом останавливаться. Единственное, отметим, что по правилам биржи игроки могут как ставить, так и отменять заявки в любой момент.

Теперь перейдем к спуфингу. Спуфинг (от английского spoofing) — это стратегия, которая основывается на искусственном занижении и завышении цены на актив за счет выставления заявок на покупку или продажу не с целью приобретения/продажи актива, а для создания мнимого спроса или предложения с последующей отменой заявки до ее исполнения. Гораздо легче понять стратегию, рассмотрев ее на примере.

Рассмотрим актив, рыночная стоимость которого составляет 100, а заявки распределены так же как на рисунке 1. И на данный момент лучшая заявка на покупку составляет 99, а заявка на продажу 101. Сейчас рынок в ожидании, вдруг на него приходит трейдер-спуфер. Он хочет сначала купить актив дешевле, а после продать дороже и извлечь гарантированную выгоду. Для этого ему нужно реализовать две части стратегии.

Игра на понижение. Спуфер начинает первую часть стратегии, играя в стакане (Рисунок 1) на понижение цены. Он размещает заявку на продажу достаточно большого объема по цене 100 услоных единиц, что становится, лучшей заявкой на продажу, но недостаточной для совершения сделки. Своим поведением он стимулирует остальных продавцов уходить ниже его цены, чтобы совершить сделку. Спуфер, как бы, изображает увеличение предложения и роняет цену. Свои заказы спуфер почти незамедлительно отменяет, а остальные заявки остаются, уменьшая тем самым минимальную цену продажи. Тут важно понять, что спуфер не собирается дожидать-

ся исполнения своей заявки —он просто хочет создать ложное впечатление падения цен. Владельцы заявок на покупку, видя такую тенденцию, тоже могут начать снижать цены покупки, но даже если это не произойдет, повторив лже-заявки несколько раз, спуфер может добиться снижения цены продажи до 100 или даже 99. После этого спуфер начинает скупать актив по сниженной цене и переходит ко второй части стратегии.

Игра по повышение. Теперь спуферу необходимо завысить цену на актив, чтобы продать его как можно дороже. То есть, на данный момент у спуфера есть акции для продажи, но лучшая цена покупки составляет 98 у.е, что не устраивает манипулятора. Тогда он идет в стакан заявок на покупку и начинает создавать видимость завышенного спроса: устанавливает крупный заказ на покупку по цене 98,1 у.е, создавая новую лучшую цену покупки, а после сразу отменяет заявку. Такими действиями он заставляет рынок двигаться вверх. По достижению нужной цены покупки в 101 спуфер проводит сделку и продает актив.

Таким образом, манипулятор смог извлечь выгоду в 2 условные единицы с одного актива за несколько минут. Если повторять такую стратегию несколько раз и в нескольких стаканах, доход спуфера может измеряться тысячами долларов в день. Описанная схема является классическим примером спуфинга. На реальных торгах существует несколько разновидностей спуфинга. Одна из них является менее рискованной, так как в ней заказы спуфера выставляются не по лучшей цене, а всегда немного ниже, что исключает возможность исполнения спуферской заявки.

На рисунке 2 представлены графики изменения лучшей цены заявок на покупку и на продажу актива. Если допустить, что увеличение цены покупки обусловлено ложными спуферскими заявками (на покупку большого объема по довольно высокой цене), то на данном графике можно наблюдать как при этом изменяется цена продажи - она тоже будет расти. Именно такая взаимосвязь и делает стратегию спуфинга эффективной.

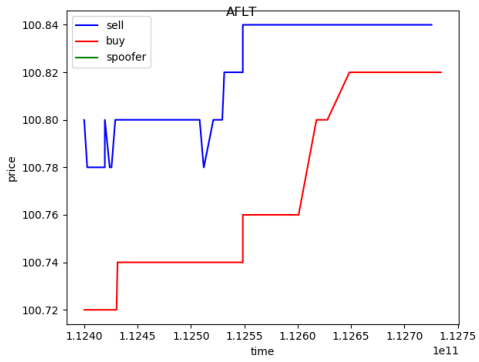


Рис. 2: Изменение цены при игре спуфера на повышение

2. Постановка задачи научно-исследовательской работы

2.1 Проблема спуфинг-манипуляции на российском фондовом рынке

В рамках этой работы сфокусируемся на одной из стратегий рыночной манипуляции – на спуфинге. Выбор в пользу изучения именно этой стратегии был обусловлен следующими причинами.

1. Спуфинг особенно тяжело определить на рынке. Сложность определения спуфинга заключается в том, что опытный мошенник будет действовать с нескольких счетов и выставлять заказы на разные объемы и разные цены.

2. Спуфинг применим на различных рынках. Хоть спуфинг чаще и встречается на фьючерсных рынках, но эта стратегия может быть применима к любому крупному активу, так как воздействует на законы формирования спроса и предложения, которые присущи любому активу на рынке.

3. Спуфинг является популярной стратегией. Популярность мошеннических стратегий, конечно, тяжело оценивать, так как нет подробной статистики по раскрытым мошенническим схемам, но из анализа новостей можно сделать вывод, что именно со спуфингом связаны все самые крупные и громкие манипуляции последних лет. Например,

когда я начинал исследование в этой области, крупнейший штраф за спуфинг-манипуляции был наложен на Deutsche bank в 2013 году в размере нескольких десятков миллионов долларов, а уже в сентябре 2020 года был установлен новый рекорд - JPMorgan обязана выплатить штраф за спуфинг в размере 1 млрд долларов [6].

4. Стратегия спуфинга может быть алгоритмически сформулирована и реализована. Ни для кого не секрет, что торговля давно ведется не людьми, а машинами, которые анализируют, принимают решение по покупке и продаже. Все действия спуфера подвергаются объяснению аппаратом математического анализа, статистики и теории игр и, вследствие чего, могут быть запрограммированы.

5. Наличие реальных примеров безнаказанного применения спуферской стратегии. На спуфинге были замечены многие игроки. Помимо примеров выше есть случаи, когда спуфер так и не понес наказания. Например, в 2012 году американский трейдер Игорь Ойстачер при торговле фьючерсами на нефть на Чикагской товарной бирже (CME) в течение шести часов создал более тысячи заявок на покупку и продажу актива[7]. Органы контроля, вроде, и пришли к выводу, что это было сделано с целью введения других участников рынка в заблуждение и дабы извлечь из этого выгоду, но итоги расследования оказались довольно размытыми, и полновесное обвинение, которое было бы подкреплено неоспоримыми доказательствами злого умысла трейдера, так и не было озвучено[8].

2.2 Формулировка цели исследования

Основной целью исследования является разработка прикладного алгоритма для обнаружения спуфинг манипуляции на российском фондовом рынке акций. Данный алгоритм должен соответствовать нескольким требованиям:

[6] https://www.tinkoff.ru/invest/news/461470/

[7] https://www.rbc.ru/newspaper/2016/11/23/5834252d9a79472d93ac9b33

[8] https://lpgzt.ru/news/2017-09-05/77021.htm

- взаимодействие с реальными данными об истории заявок на покупку и продажу;

- поиск подозрительных заявок, которые могли бы быть частью манипулятивной стратегии спуфинга;

- выдача результата в том же формате, что и входные данные, но с дополнительной отметкой о возможном спуфинге;

- возможность настройки на чувствительность и различные разновидности спуфинга.

Так же хотелось бы упомянуть о глобальной цели исследования - борьбе против спуфинг манипуляций на российском фондовом рынке. В случае успеха данный алгоритм может применяться биржами и судебной властью для привлечения манипуляторов к уголовной ответственности за свои действия.

2.3 Формулировка поставленных задач

Для достижения основной цели исследования необходимо решить ряд задач.

1. Исследовать взаимосвязь с другими работами в этой области. Позволю себе начать работу именно с ответа на вопрос, для чего и почему необходима разработка данного алгоритма, как она соотносится с другими исследованиями в этой области. Мало просто обнаружить спуфинг - необходимо доказать его применение и негативное влияние на других игроков. Лишь в таком случае может последовать наказание. Для этого проводится анализ литературы, чтобы понять ценность и применение алгоритма спуфинга в рамках общей цепочки обнаружения и наказания спуфинга.

2. Подготовить данные для работы алгоритмов и тестов. Это необходимый блок, так как наши исследования, в первую очередь, носят прикладной характер и должны быть осуществимы на реальных данных. Понимая, с чем нам предстоит столкнуться на следующих этапах, необходимо подготовить данные к последующей работе. За основу

взяты данные московской биржи за один день, содержащие в себе информацию о всех заявках на покупку и продажу, а также обо всех сделках.

3. Смоделировать поведение спуфера. Основная причина, по которой необходимо решить эту задачу - отсутствие в открытом доступе реальных примеров применения спуфинг манипуляции в формате биржевых данных (истории заявок). Судить о корректной работе алгоритма невозможно без тестирования, поэтому нам необходимо искусственно создать такие данные, которые бы содержали спуфинг. Так же решение этой задачи поможет закрепить и проверить на практике теоретическое понимание стратегии спуфинга, что подготовит нас к написанию алгоритма обнаружения.

4. Разработать алгоритм обнаружения спуфинга. Это центральная задача научно исследовательской работы, которая должна решить вопрос о поиске подозрительных заявок, которые могли бы быть частью стратегии спуфинга. Алгоритм должен иметь сильную теоретическую базу и практическую реализацию.

5. Провести тестирование алгоритма Для настройки и корректной работы алгоритма необходимо провести несколько тестов. Тесты на искусственно созданных данных дадут информацию о верности или неверности алгоритма и его точности. А тесты на реальных данных дают представление о его практической применимости.

2.4 Ожидаемые результаты

Теперь поговорим о результатах. Я хотел бы выделить два ключевых момента, которые мы должны получить по окончании исследования.

1. Алгоритм на языке Python, способный добавлять в исторические данные биржи суперские заявки и отменять их так, чтобы не возникло отклонение от сложившейся рыночной ситуации и действия спуфера были максимально приближены к реальным и были эффективными с точки зрения спуфера.

2. Алгоритм обнаружения заявок, подозрительных на принадлежность спуфинг-стратегии, реализованный на языке Python. Алгоритм должен быть параметризован, так чтобы подстраиваться под разные стратегии и иметь разную чувствительность.

3. Обзор литературы о манипуляции на финансовых рынках

Поскольку качественных исследований на тему спуфинга не так много, мне бы хотелось подробнее остановиться на научных публикациях одного автора, который на протяжении последних 10 лет занимается этой темой. Речь идет об американском ученом Веллмане (Michael P. Wellman). В этом блоке на примерах работ Веллмана [2] [5] [6] я передам несколько его тезисов.

Всех игроков на рынке можно разделить на две категории: профессионалы (heuristic belief learning) и новички (zero intelligence) [2]. Главное отличие между ними в том, что действия новичков максимально спонтанны и не основываются на анализе открытой информации, то есть порой их действия могут быть иррациональными и непредсказуемыми. Профессионалы же действуют обоснованно: их решения базируются на истории заявок. И вот тезис, который довольно подробно раскрывает Веллман, гласит, что манипулятор оказывает воздействие именно на профессиональных игроков. Иными словами, можно сказать, что спуфер будет оказывать влияние только на тех игроков, которые изучают историю заявок и внимательно отслеживают все тренды которые происходят в биржевых стаканах. Веллман проводит ряд экспериментов, в которых изучает влияние спуфера на рынок из новичков и профессионалов, он исследует разные пропорции этих игроков и проводит симуляции с различным уровнем памяти профессиональных игроков (на сколько долго они отслеживают историю заявок). В результате, автор приходит к выводу, что спуфер может оказывать влияние и реализовывать свою стратегию только на рынке с профессиональными игроками, которые отслеживают историю заявок.

В своей модели Веллман [2] определяет поведение спуфера довольно просто. Спуфер появляется на рынке в определенный момент времени, размещает заказ большого объема по цене на единицу меньше лучшей цены и отменяет заявку, как только лучшая цена изменилась. Затем, он снова размещает заявку того же объема и отменяет ее и так далее. В своей симуляции я буду применять похожий подход, но я бы не сказал, что я ори-

ентировался на стратегию Веллмана в этом случае – скорее обоснованно пришел к такой стратегии сам. Просто использовать указанный алгоритм я посчитал недопустимым, так как Веллман его никак не обосновывает, а лишь описывает саму стратегию.

Исследования Веллмана [2] нацелены на аспект влияния спуфера на выгоду других игроков. Проводя много симуляций торгов без спуфера, он получает усредненную выгоду игроков на рынке по истечении дня. Позже он повторяет симуляции для тех же игроков, но с одним спуфером на рынке и получает уже другое значение прибыли для игроков. Проводя несколько серий экспериментов с разным числом игроков, Веллман приходит к следующим выводам:

- Чем больше опытных игроков на рынке, тем выше отклонение цены от реальной за счет действий спуфера;

- Если на рынке опытных игроков больше, то отклонение цены тем выше, чем дольше спуфер играет в стакане;

- С течением времени отклонение цены сначала увеличивается, а к концу торгов либо стабилизируется, либо уменьшается. То есть, влияние спуфера на цену максимально, когда он начинает манипулировать, а после постепенно снижается.

Результаты Веллмана подтверждают влияние, которое оказывается спуфером на рынок [6]. Выкладки из его исследований могут быть использованы для доказательства манипуляции. По российскому законодательству необходимыми условиями манипуляциии являются:

- существенное отклонение цены от той, что сложилась бы без стратегии манипулятора;

- факт нанесения убытков другим игрокам вследствие действий манипулятора.

Оба эти пункта могут быть доказаны на основе алгоритмов Веллмана. Это является очень важным звеном в нашей цепочке к наказанию спуфера. Мы же пока сфокусируемся на задаче выявления спуфинга, а далее, уже учитывая все тонкости уголовного законодательства РФ касательно рыночных манипуляций, подробно рассмотренных в этой статье [3], будем судить о доказанности применения спуфинг-стратегии.

4. Работа с исходными данными

4.1 Спецификация исходных данных

К сожалению, в открытом доступе нет данных, содержащих гарантированную стратегию спуфинга, и заполучить их не представляется возможным. На таких данных мы могли бы обучить наш алгоритм выявлять спуфинг. Поэтому было принято решение провести симуляцию поведения спуфера на основе реальных данных с московской биржи. Нас интересует история заявок на покупку и продажу за день торгов. На официальном сайте[9] такой формат данных называется Full Orders Log (тип A), который имеет спецификацию, представленную на Рис. 3. Для нас ключевыми полями будут являться соответственно цена и объем, а также поля Time и Action, которые позволят вычислять продолжительность заявки для OrderNO. Пример данных можно увидеть на Рис 4.

Еще раз хотел бы обратить внимание, что у нас нет данных о том, принадлежат ли заявки одному участнику рынка или нескольким. Вследствие этого у нас не будет возможности смоделировать стратегию конкретного игрока, а лишь влияние на рынок стратегии спуфинга. Но это для нас не будет препятствием, так как в реальном мире спуфер может действовать не единолично, а в сговоре с другими игроками, и нам важно выявить это самое незаконное воздействие.

4.2 Фильтрация данных с московской биржи

Перед тем, как приступить к моделированию спуфинга необходимо подготовить данные. В первую очередь, нам необходимо разделить заявки по активам, образовав тем самым биржевые стаканы. Таких стаканов было получено 297. Теперь необходимо проанализировать их и выбрать подходящие для внедрения спуфинга. В первую очередь заметим, что нас интересуют активно торгующиеся активы, так как ценами на них легче манипулировать, оставаясь незамеченным, поэтому активы, в которых количество заявок меньше среднего значения были удалены из рассмотрения.

[9] https://www.moex.com/a588

Наименование поля	Тип	Описание
NO	*Int*	Номер записи
SECCODE	*String*	Код инструмента
BUYSELL[1]	*Char*	Признак купли/продажи
		B= Купить
		S= Продать
TIME[2]	*LongInt*	Время в формате **HHMMSSZZZXXX** с марта 2016
		Время в формате HHMMSSZZZ до марта 2016
ORDERNO	*Int*	Номер заявки
ACTION	*Byte*	Тип события:
		0=Снятие заявки
		1=Постановка заявки
		2=Сделка
PRICE[3]	*Float*	Цена заявки
VOLUME	*Int*	Объем
		Для action=1 – видимый объем поставленной заявки
		Для action=2 – объем сделки
		Для action=0 – остаток видимой части заявки
Следующие поля заполняются только для сделок (ACTION=2)		
TRADENO	*Int*	Номер сделки
TRADEPRICE	*Float*	Цена сделки

Рис. 3: Спецификация данных с Московской биржи Full Orders Log

NO	SECCODE	BUYSELL	TIME	ORDERNO	ACTION	PRICE	VOLUME	TRADENO	TRADEPRICE
30818	LSRG	S	100000429850	30789	2	574.9	10	1961689585	575
30819	NLMK	B	100000430356	30790	1	63.21	1780		
30831	ELTZ	S	100000435125	25211	0	824.6	150		

Рис. 4: Пример данных типа Full Orders Log с московской биржи

Среднее значение – 6817 заявок в день. Активов, которые удовлетворяют нашим требованиям осталось только 59. Их можно увидеть на Рис. 5.

Также, из рассмотрения убираем все заявки выставленные в первый и последний час торгов. Такое решение обусловлено тем, что на открытии и закрытии торгов иногда происходят колебания, которые либо связаны со свежими новостями, либо обусловлены иными видами манипуляции, которые применимы именно на открытии и при закрытии биржи. Мы же хотим, чтобы наш эксперимент был максимально свободен от посторонних влияний, поэтому приняли решение не вводить спуфера в эти временные промежутки, а также не искать спуфинг на них. Если в дальнейшем появятся причины для пересмотра такого решения, то всегда можно будет применить алгоритмы к данным, содержащие заявки за весь день. Это не будет препятствием для работы программы - лишь может повлиять на ее

Акция	SBER	TRNFP	SFIN	POLY	MOEX	MVID	GAZP	LKOH	SIBN	VTBR
количечтсво	110586.0	90356.0	87376.0	80122.0	73074.0	70527.0	69970.0	58028.0	55223.0	53768.0
Акция	MTLR	ROSN	MGNT	RUAL	TGKA	SBERP	PLZL	GMKN	ALRS	RASP
количечтсво	50466.0	48994.0	43063.0	41482.0	39304.0	37588.0	36136.0	35512.0	30687.0	28460.0
Акция	TATN	SNGSP	YNDX	MSNG	AKRN	RSTI	MTSS	AFKS	BANEP	RTKM
количечтсво	28004.0	26100.0	25704.0	24810.0	22662.0	22469.0	21652.0	21519.0	20910.0	20726.0
Акция	PHOR	NVTK	AFLT	CHMF	HYDR	NLMK	CBOM	FEES	MTLRP	URKA
количечтсво	20653.0	19252.0	19120.0	18686.0	18563.0	18436.0	17612.0	17161.0	16340.0	16089.0
Акция	IRAO	UPRO	MAGN	OGKB	TATNP	FIVE	SNGS	CHMK	MSST	DSKY
количечтсво	16077.0	15402.0	14482.0	14341.0	14329.0	11861.0	11529.0	9268.0	8587.0	8357.0
Акция	LNTA	KRKOP	APTK	ENPL	IRKT	LSRG	MOBB	BSPB	UWGN	
количечтсво	8269.0	7811.0	7743.0	7559.0	7540.0	7346.0	7332.0	7104.0	7099.0	

Рис. 5: Топ-59 компаний по количеству заявок за день

точность.

4.3 Подготовка данных для внедрения и обнаружения спуфинга

Программная реализация предобработки данных представлена на языке Python в Приложении 1.

Для начала определим формат, в который мы преобразуем нашу исходную таблицу. Так как для анализа спуфинга необходимо изучать спрос и предложение, разделим каждый актив на две таблицы: одна с заявками только на покупку, другая с заявками только на продажу. Далее, каждый раз анализируя или работая с данными, мы будем иметь дело с заявками на покупку или продажу конкретного актива.

Вычисление лучшей цены в каждый момент времени. Как отмечалось ранее, спуфинг-манипуляция тесно связана со спросом и предложением. Для того, чтобы спуфер мог грамотно действовать, нам необходимо знать лучшую цену покупки и продажи в каждый момент времени. Лучшая цена покупки - это максимальная цена покупки, а лучшая цена продажи - это минимальная цена продажи, так как именно на пересечении этих цен и происходит сделка. Знание лучших цен поможет спуферу определить цену заявок, которые он должен установить. Нагляднее всего иллюстрировать значения лучших цен, которые мы добавляем в данные, с помощью графи-

ка. Пример такого графика представлен на Рис. 6. где синим представлен график изменения минимальной цены продажи, а красным – максимальной цены покупки.

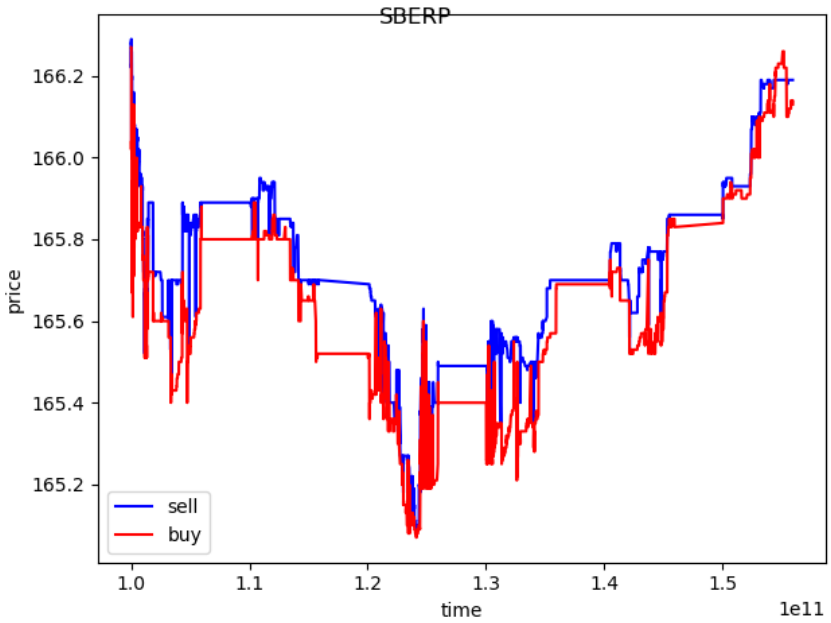


Рис. 6: График изменения наибольшей цены покупки и наименьшей цены продажи.

На данном графике представлены изменения лучшей цены цены продажи и покупки привилегированных акций Сбербанка в промежуток времени с 10 утра до 15 дня. В тех точках, где две кривые пересекаются, происходит сделка, так как в этот момент времени совпадают наименьшая цена продажи и наибольшая цена покупки (продавец готов продать актив по данной цене, а покупатель готов его приобрести).

Такие графики были построены для всех оставшихся активов, подходящих по вышеуказанных параметрам. Они представлены в Приложении 2. Реализовано это за счет алгоритма на языке Python, который построчно считывает данные и сверяет, не изменилась ли лучшая цена (Приложение 1). Причин изменения лучшей цены может быть две:

- заявка с лучшей ценой была отменена

- заявка была реализована (тут так же учитываем объем сделки, который должен равняться объему лучшей заявки)

Чтобы отобразить лучшую цену (BESTPRICE) в данных, необходи-

мо с помощью программного кода (приведен в Приложении 1) проитерироваться по строкам таблиц истории заявок (отдельно для заявок на покупку и продажу), и храня в памяти массив всех активных на данный момент заявок, поставить в соответствие каждой строке таблицы текущую лучшую цену (каждый раз проверяются две возможные причины изменения лучшей цены).

По результатам работы данной программы в таблицы активов был добавлен столбец с названием BESTPRICE, в котором фиксируется лучшая цена в конкретный момент времени. Внешний вид полученных данных представлен ниже на рисунке 7.

	NO	SECCODE	BUYSELL	TIME	ORDERNO	ACTION	PRICE	VOLUME	TRADENO	TRADEPRICE	BEST_PRICE
560	57795	MTLRP	B	100033133662	48244	1	98.75	200	NaN	NaN	99.05
561	57796	MTLRP	B	100033133807	48245	1	99.10	2820	NaN	NaN	99.1
562	57797	MTLRP	B	100033134008	48246	1	99.00	80	NaN	NaN	99.1
563	57799	MTLRP	B	100033134623	48246	0	99.00	80	NaN	NaN	99.1

Рис. 7: Добавление столбца BESTPRICE с лучшей ценой в данные с московской биржи

На рисунке 7 видно, как таблице исходных данных был добавлен новый столбец (крайний справа), содержащий сведения о лучшей цене. Остальные ячейки остались без изменений.

Вычисление объема предложения и спроса в каждый момент времени. Второй важнейший аспект, который важен для спуфинга - это объем спроса и предложения. Логика тут примерно та же, что и с определением лучшей цены. Программный код в Приложении 1 построчно проходит по всем строкам таблицы заявок, храня в памяти суммарный объем активных заявок в каждый момент времени. Каждой заявке ставится в соответствие значение объема всех активных заявок. Напоминаю, что обработка происходит отдельно для таблицы с заявками на покупку и на продажу, поэтому и объем (TOTALVOLUME) для них разный. Объем может меняться в трех случаях:

- либо в результате сделки, тогда суммарный объем уменьшается на величину сделки

- либо из-за отмены заявки, тогда он уменьшается на величину объема этой заявки

- либо при появлении новой заявки, тогда он увеличивается на величину объема заявки.

По результатам работы данной программы в таблицы активов был добавлен столбец с названием TOTALVOLUME, в котором фиксируется значение суммарного объема спроса и предложения в тот момент, когда эта заявка встречается в таблице. Внешний вид полученных данных представлен ниже на рисунке 8.

	NO	SECCODE	BUYSELL	TIME	ORDERNO	ACTION	PRICE	VOLUME	TRADENO	TRADEPRICE	BEST_PRICE	TOTALVOLUME
560	57795	MTLRP	B	100033133662	48244	1	98.75	200	NaN	NaN	99.05	55700
561	57796	MTLRP	B	100033133807	48245	1	99.10	2820	NaN	NaN	99.1	58520
562	57797	MTLRP	B	100033134008	48246	1	99.00	80	NaN	NaN	99.1	58600
563	57799	MTLRP	B	100033134623	48246	0	99.00	80	NaN	NaN	99.1	58520
564	57802	MTLRP	B	100033136350	48249	1	99.05	80	NaN	NaN	99.1	58600
565	57880	MTLRP	B	100033354776	48297	1	74.00	1000	NaN	NaN	99.1	59600
566	57946	MTLRP	B	100033516650	48243	0	99.05	800	NaN	NaN	99.1	58800
567	57947	MTLRP	B	100033517035	48245	0	99.10	2820	NaN	NaN	99.05	55980
568	57949	MTLRP	B	100033517379	48249	0	99.05	80	NaN	NaN	99	55900
569	57950	MTLRP	B	100033517448	48322	1	98.80	2820	NaN	NaN	99	58720
570	57951	MTLRP	B	100033517799	48323	1	98.95	80	NaN	NaN	99	58800
571	58040	MTLRP	B	100033715212	48372	1	98.75	800	NaN	NaN	99	59600
572	58118	MTLRP	B	100033757500	48244	0	98.75	200	NaN	NaN	99	59400
573	58154	MTLRP	B	100033825349	48431	1	98.85	200	NaN	NaN	99	59600
574	58155	MTLRP	B	100033825814	48322	0	98.80	2820	NaN	NaN	99	56780
575	58156	MTLRP	B	100033826343	48432	1	98.90	2820	NaN	NaN	99	59600

Рис. 8: Добавление столбца TOTALVOLUME с суммарным объемом в данные с московской биржи

На рисунке 8 представлен пример добавления значения TOTALVOLUME к таблице данных.

Для оптимизации программного кода значения добавление параметров BESTPRICE и TOTALVOLUME происходило в едином цикле, чтобы уменьшить количество необходимых итераций по строкам таблицы и сократить время работы программы.

Разделение на тестовые и тренировочные данные Теперь нам необходимо обработанные данные разделить на тренировочные и тестовые. На

тренировочной части данных будет сначала применена симуляция поведения спуфера, чтобы после на них можно было проверить алгоритм и настроить параметры. А тестовые данные будут использоваться как контрольные показатели работы алгоритма на реальных данных.

Изначально планировалось разделение на тренировочные и тестовые данные в пропорции 4:1, но позже выяснилось, что вычислительных мощностей не хватает для обработки такого количества тренировочных данных, так как процесс внедрения и спуфинга занимает продолжительное время. Поэтому для тренировочных данных было отобрано случайным образом 10 активов.

Программный код Python, выполняющий эти действия представлен в Приложении 1. В нем использована библиотека pandas для работы с табличными данными.

5. Симуляция спуфинг-стратегии на тренировочных данных

5.1 Описание стратегия спуфинга в симуляции

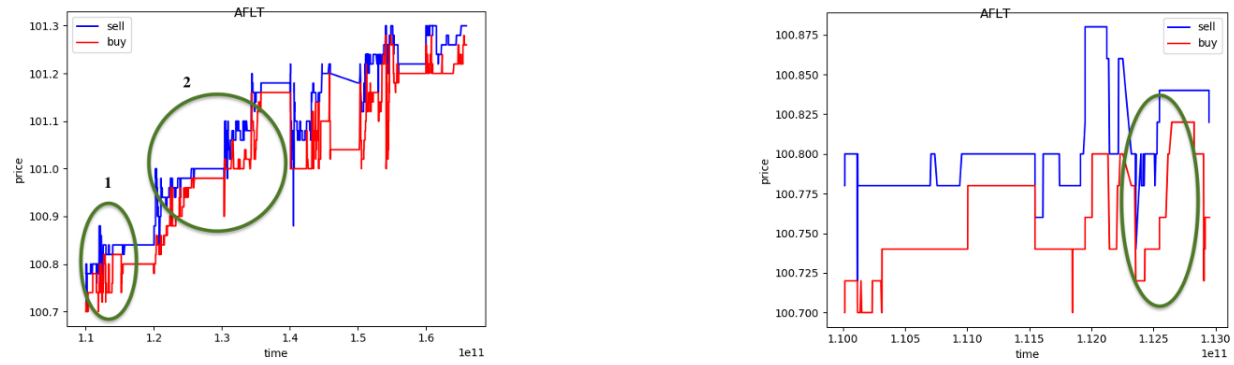
Ранее мы уже определяли поведение спуфера как стратегии влияния на цену путем выставления и скорой отмены крупных заявок на покупку или продажу. Теперь настало время описать точную последовательность действий спуфера для симуляции данных со спуфингом.

Итак, в нашем распоряжении есть данные заявок и сделок за один день, мы отобрали подходящие стаканы для спуфинга. Теперь мы хотим искусственно ввести в эти данные спуфера. Как это сделать? Идея заключается в том, что мы сначала определим промежутки времени, когда могла бы быть реализована стратегия спуфинга, а после реализуем стратегию спуфинга так, чтобы она соответствовала реальным изменениям цены, то есть служила бы ее причиной. Иными словами, выбрав подходящие для спуфинга интервалы мы подберем такие спуферские заявки, которые могли бы стать причиной изменения цены, которое наблюдается на этих интервалах. Далее подходящие интервалы спуфинга будем называть микроколебаниями, так как стратгия спуфинга успешна лишь тогда, когда оказывает краткосрочное влияние на рынок. Объяснять же продолжительное изменение цены спуфингом будет некорректно.

5.1.1 Параметризация подходящего момента времени для введения спуфинга

В какие моменты вводить спуфинг? Заметим, что помимо спуфинга есть ряд нелегальных стратегий именно для манипуляции на закрытии и открытии, поэтому нашего спуфера мы будем вводить с 11 до 17 часов, чтобы повысить чистоту эксперимента. Цены при открытии и закрытии гораздо более изменчивы и подвержены шумам, мы же хотим сделать наши микроколебания максимально зависящими от спуфера, поэтому и рассматриваем такой временной интервал. Определились со временем, теперь что считать подходящими микроколебаниями? Рассмотрим этот вопрос на

Рис. 9: График изменения лучших цен покупки и продажи: общий вид (слева) и приближенный вид (справа)



примере акций Аэрофлота и дальше все основные тезисы будем демонстрировать именно на акциях этой компании, так как визуально они выглядят наиболее подходящими для наших действий.

Изучив левый Рис. 9, который иллюстрирует изменение лучших цен в промежуток с 11 до 17 часов, мы заметим, что на нем есть зоны нескольких видов: зона 1 представляет из себя микроколебания, а зона 2 больше похожа на постепенный рост. С левой стороны Рис. **??** можно увидеть как раз ту зону 1 микроколебания максимальной цены покупки. Это микроколебание можно объяснить спуфингом. Повторю ключевой тезис, что мы находим микроколебания с однонаправленным изменением цены, чтобы объяснить эти изменения спуфинг манипуляцией.

Излагая вышеуказанные тезисы на математическом языке, можно сказать, что спуферскую заявку будем размещать там, где за период времени MICRODELTA секунд лучшая цена изменялась как минимум MICRONUM раз в одном направлении подряд.

5.1.2 Параметризация спуфинг-заявок

Теперь поговорим о том, как выглядит спуфинг заявка которую мы сначала будем искусственно добавлять в данные, а после пытаться обнаружить алгоритмом поиска спуфинга. Из определения спуфинга можно заключить, что суммарно заявки спуфера должны:

- иметь большой объем, чтобы оказывать влияние на рынок
- цена заявок максимально близка к лучшей цене

- заявки будут отменены, как только была улучшена цена

Чтобы описать такую заявку спуфера математически, мы должны ввести три параметра: SPOOFPRICE, SPOOFDELTA и SPOOFVALUE. Тогда условия на объем и цену могут быть сформулированы следующим образом. Заявка спуфера:

1. По цене находятся в диапазоне SPOOFPRICE процентов от лучшей цены.

2. Отменена в окрестности SPOOFDELTA секунд после следующего изменения цены.

3. По объему вместе с другими подобными заявками составляет SPOOFVALUE процентов от TOTALVALUE

5.2 Программная симуляция спуфинг-стратегии

Симуляция реализована в два этапа на языке Python. Программный код доступен в Приложении 3. Хотелось бы обратить внимание, что мы начинаем непосредственно работать с данными, полученными после предварительной обработки. Каждая таблица, по которой будет происходить итерация представляет из себя историю заявок на покупку или продажу конкретного актива.

5.2.1 Этап 1: выделение микроколебаний, подходящих для спуфинга

Шаг 1: заполнение вспомогательного массива Trends. Массив Trends носит исключительно вспомогательный характер для отслеживания количества подряд идущих изменений цены в одну сторону (увеличение или уменьшение лучшей цены. На первом этапе необходимо выделить периоды микроколебаний, на которых действия спуфера могли бы оказывать эффект на изменение цены. Для того чтобы найти в данных подходящие периоды микроколебаний необходимо итеративным способом пройти по всем строкам каждой таблицы (разные активы и заявки на покупку и продажу) и изучить особенности изменения цены по следующим правилам:

1. Если цена не изменилась, то переходим к следующей строке
2. Если цена увеличилась, то в массив Trends заносим единицу
3. Если цена уменьшилась, то в массив Trends заносим ноль

Шаг 2: нахождение подходящих временных интервалов Каждый раз, когда когда в массиве Trends происходит изменение нулей на единицы или наоборот, мы заканчиваем микроколебание, которое состояло из нескольких изменений лучшей цены и проверяем условие на соответствие нашим требованиям:

1. Сколько времени прошло с начала микроколебания до его окончания. Если это значение меньше параметра MICRODELTA, то переходим к следующему условию
2. Сколько элементов содержится в массиве Trends. Длина этого массива соответствует количеству односторонних улучшений цены. Если длина массива получилась более MICRONUM, то такое микроколебание нам подходит.

Для тех микроколебаний, что нам подошли, заносим время начала и конца в отдельный массив `Micro_times`.

5.2.2 Этап 2: добавление и удаление спуфинг-заявок

После первого этапа у нас имеется массив состоящий из временных пар начала и конца подходящего микроколебания. Теперь наша задача изучить все заявки между началом и концом каждого микроколебания. Для этого запускаем итерационный процесс по всем парам в массиве `Micro_times` и для каждой пары вновь итерируемся по всем заявкам внутри этого колебания.

Шаг 1: добавление заявок Каждый раз когда происходит улучшение цены, то нам необходимо выставить новые заявки по стратегии спуфинга.

Количество этих заявок - случайное число в пределах от 2 до 10. Каждая новая установленная заявка содержит следующие характеристики:

'NO': 0,

'SECCODE': SECCODE, *(код актива)*

'BUYSELL': BUYSELL, *(код заявки на покупку или продажу)*

'TIME': `time_start` + 100 + i, *(время начала микроколебания)*

'ORDERNO': number, *(номер спуфинг-заявки с префиксом s)*

'ACTION': 1, *(код действия для постановка заявки)*

'PRICE': `prev_best_price` + (2*i2-1)*SPOOFPRICE*`prev_best_price`, *(prevbestprice - предыдущая лучшая цена, i2 - бинарная переменная: 0 - если заявка на покупку, 1 - если на продажу, SPOOFPRICE - параметр алгоритма, обозначает интервал по цене, в котором должны находится заявки, которые выставляются и отменяются во время микроколебания)*

'VOLUME': TOTALVALUE*SPOOFVALUE/n, *(n - количество спуферских заявок, SPOOFVALUE - параметр (процент), показывает какой минимальный объем должен быть у спуферских заявок, чтобы оказать необходимое влияние на других игроков)*

'TRADENO': 0,

'TRADEPRICE': 0,

'BESTPRICE': `prev_best_price`,

'TOTALVOLUME': TOTALVALUE,

'SPOOFER': 1

Когда такая заявка выставляется, то ее номер заносится в специальный массив Cancel.

Шаг 2: удаление заявок Каждый раз когда происходит улучшение цены, нам необходимо удалить выставленные на предыдущей итерации заявки. Удаление происходит по номерам заявок ORDERNO, которые хранятся в массив Cancel.

И так далее до тех пор, пока не будут пройдены все интервалы микроколебаний для конкретного актива.

5.2.3 Значения параметров для симуляции

В общем смысле разные параметры симуляции характеризуют различные стратегии спуфинга. Но при моделировании мы не можем учесть разнообразие стратегий, поэтому необходимо остановиться на конкретном наборе. Следующие значения были определны исходя из теоретических выкладок о спуфинге.

MICRONUM = 5

MICRODELTA = 2000000 (20 секунд)

SPOOFVALUE = 0.4

SPOOFPRICE = 0.01

5.3 Полученные тренировочные данные с симуляцией спуфинга

Результатом разработанного алгоритма стали 42 стакана в которых была искусственно внедрена стратегия спуфинга по всем вышеописанным правилам. Затем эти стаканы были объединены в общую таблицу и получился дата-сет эквивалентный исходному. Всего на рынке за день было поставлено 2473 спуферских заявок. Дальше продолжим разбирать результаты на конкретном примере акций Аэрофлота. Теперь данные выглядят следующим образом, как на Рис. 10. К сожалению пришлось их несколько сократить, чтобы уместить на страницу – полную историю заявок этого примера можно найти в приложении.

Итак разберем что здесь происходит. Спуфер играет на повышение, выставляя и отменяя заказы на покупку. Конечная его цель – повысить цену продажи. На рассматриваемом периоде минимальная цена продажи 100,72. Значит спуферу нужно тянуть цену покупки вверх, имитируя высокий спрос. Для этого он заявку на крупную покупку по цене на 0,1 ниже чем лучшая цена покупки. Как только кто-то разместил заявку по более высокой цене, спуфер отменяет заявку и создает новую. Напомню, что 1 в ACTION – постановка, 0 – снятие заявки. Спуфер выставляет заявку под лучше ценой, а после ее изменения сразу удаляет и ставит новую. И так повторяется несколько раз и конечном итоге спуферу удается довести цену

NO	SECCODE	BUYSELL	TIME	ORDERNO	ACTION	PRICE	VOLUME	TRADENO	TRADEPRICE	BEST_PRICE	TOTALVOLUME
528093	AFLT	B	112400000795	278925	2	100,72	680	2914390041	100,72	100,72	42005
528092	AFLT	S	112400000795	285734	1	100,72	680			100,72	42685
	AFLT	B	112400000796	S-125	0	100,7	16540			100,72	26145
	AFLT	B	112400000797	S-131	1	100,71	17220			100,72	26146
528384	AFLT	S	112402808131	285835	1	100,8	370			100,78	26516
528442	AFLT	B	112403579272	285865	1	90	100			100,72	26616
529685	AFLT	S	112423875692	286417	1	100,78	50			100,78	26666
529763	AFLT	S	112425594569	286455	1	100,84	50			100,78	26716
529940	AFLT	B	112427982353	286535	1	100,72	1000			100,72	27716
530209	AFLT	B	112429107997	286633	1	0	60			100,72	27776
530210	AFLT	B	112429107997	286633	2	0	50	2914390355	100,78	100,72	27926
530212	AFLT	B	112429107997	286633	2	0	10	2914390356	100,8	100,72	27956
530211	AFLT	S	112429107997	286417	2	100,78	50	2914390355	100,78	100,8	28106
530213	AFLT	S	112429107997	284923	2	100,8	10	2914390356	100,8	100,8	28136
530214	AFLT	S	112429108633	285835	0	100,8	370			100,8	27766
530215	AFLT	S	112429110079	286634	1	100,82	370			100,8	28136
530432	AFLT	B	112430297165	286731	1	100,5	100			100,72	28236
530498	AFLT	B	112431211027	286768	1	100,74	50			100,74	28286
	AFLT	B	112431211028	S-131	0	100,71	17220			100,74	11066
	AFLT	B	112431211029	S-139	1	100,73	15980			100,74	27046
530539	AFLT	B	112431810748	286792	1	100,74	50			100,74	27096
536395	AFLT	B	112548613279	290117	2	0	100	2914390845	100,82	100,74	27396
536396	AFLT	S	112548613279	289549	2	100,82	100	2914390845	100,82	100,82	27696
536397	AFLT	S	112548613798	289549	0	100,82	270			100,84	27426
536398	AFLT	B	112548614151	290118	1	100,76	370			100,76	27796
	AFLT	B	112548614152	S-139	0	100,73	15980			100,76	11816
	AFLT	B	112548614153	S-146	1	100,75	16150			100,76	27966
536399	AFLT	S	112548614487	290119	1	100,86	370			100,84	28336
537004	AFLT	S	112558180658	285923	0	101,16	980			100,84	27356
537023	AFLT	S	112558387510	290432	1	101,2	980			100,84	28336
537169	AFLT	B	112600897586	290508	1	100,74	1000			100,76	29336
537353	AFLT	S	112603219401	290596	1	101,96	2000			100,84	31336
538044	AFLT	B	112617615989	290926	1	100,8	80			100,8	31416
	AFLT	B	112617615990	S-146	0	100,75	16150			100,8	15266
	AFLT	B	112617615991	S-154	1	100,79	17150			100,8	32416
538045	AFLT	B	112617616372	290118	0	100,76	370			100,8	32046
538046	AFLT	S	112617616506	284924	0	100,96	1690			100,84	30356
538212	AFLT	B	112620128398	286001	0	100,4	990			100,8	29366
539735	AFLT	S	112644929973	290380	0	101,46	1000			100,84	28366
539737	AFLT	S	112645058786	291753	1	101,42	1000			100,84	29366
539893	AFLT	B	112648529817	291834	1	100,82	40			100,82	29406
	AFLT	B	112648529818	S-154	0	100,79	17150			100,82	12256
	AFLT	B	112648529819	S-160	1	100,81	16150			100,82	28406
539894	AFLT	S	112648530326	291835	1	101,72	1690			100,84	30096
539994	AFLT	S	112651221401	291888	1	102,42	1000			100,84	31096

Рис. 10: Пример данных, извлеченных из модели

продажи до 100,84.

Убедимся, что каждая спуферская заявка выставлена для поддержания лучшей цены и приводит к ее изменению. Эта часть таблицы соответствует такому графику, на котором четко видна стратегия спуфера, способствующая росту рынка.

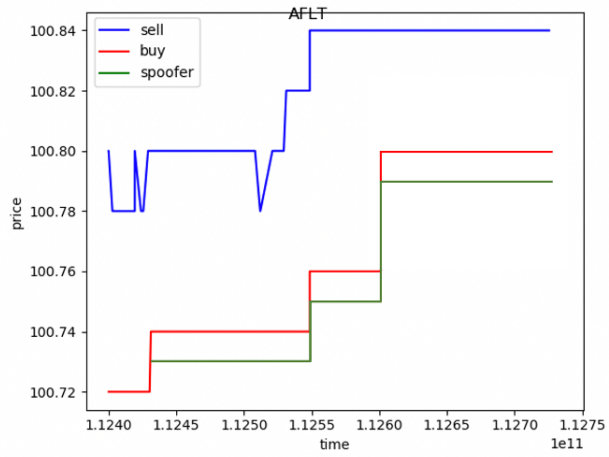


Рис. 11: График стратегии Спуфинга

Зеленая линия спуфинга на Рис. 11 повторяет траекторию изменения максимальной цены покупки, тем самым как бы задавая ее. В то же время заявка спуфера всегда находится под лучшей ценой, чтобы избежать своего исполнения.

6. Алгоритм обнаружения спуфинга

6.1 Идея алгоритма обнаружения спуфинга

После того, как была решена проблема отсутствия данных для обучения алгоритма за счет симуляции спуфинга самостоятельно, можно переходить к написанию алгоритма обнаружения. Идея алгоритма заключается в том, чтобы обнаружить сначала интервалы времени, когда спуфинг мог оказывать влияние на рынок, а после проверить наличие заявок, которые могли бы быть частью спуфинг-стратегии.

6.1.1 Нахождение микроколебаний

Теперь мы начинаем определять периоды микроколебаний, чтобы уже на них отслеживать возможных спуферов. По параметру MICRONUM (количество однонаправленных изменений лучшей цены) и MICRODELTA (время для микроколебания) определяем , какой промежуток времени максимален для одностороннего движения лучшей цены, чтобы микроколебание подходило под спуфинг-манипуляцию. Алгоритм сначала находит MICRONUM подряд идущих изменений. Затем проверяет количество времени, которое прошло между началом и концом этого изменения и если оно укладывается в значение MICRODELTA, то такой интервал нам подходит.

6.1.2 Нахождение подозрительных заявок

Теперь, когда у нас есть подходящий интервал с подходящим трендом мы проверяем, есть ли на нем выставленные подозрительные новые заявки. Между каждым изменением цены должны быть новые заявки, которые:

- по цене находятся в диапазоне SPOOFPRICE процентов от лучшей цены.

- отменены в окрестности SPOOFDELTA после следующего изменения цены.

Если по объему такие заявки в сумме составляют SPOOFVALUE процентов от TOTALVALUE, то такие заявки обнаружены, то они отмечаются в дополнительном столбце SPOOFER.

Это описан базовый алгоритм, который в теории должен определить подозрительные заявки, которые потенциально могут быть спуферскими. Тут сразу отслеживается не самая простая стратегия спуфера, когда мошенник выставляет несколько заявок по разным ценам, а не самый примитивный вариант с одной крупной заявкой, так как на реальных данных такого никогда не было бы.

6.2 Определения параметров и переменных алгоритма

Продолжая описание алгоритма хотелось бы немного остановится на каждом из параметров , которые используются в нем. Хорошо бы понимать, как выбирается каждое значение, что оно характеризует и что будет происходить при его изменении.

BESTPRICE - переменная (рациональное число, больше 0), характеризует лучшую цену покупки или продажи в данный момент. Если заявка на покупку, то и лучшая цена покупки. Если заявка на продажу, то лучшая цена продажи

TOTALVOLUME - переменная (натуральное число, больше 0), характеризует суммарный объем заявок на покупку или продажу в определенный момент времени

MICRONUM - параметр (натуральное число), обозначает сколько минимально подряд должно быть односторонних изменений цены, чтобы интервал был подвержен влиянию спуфера. Косвенно этот параметр характеризует точность модели через количество интервалов, которое мы анализируем на предмет спуфинга.

MICRODELTA - параметр (вещественное положительное число), связан с предыдущим параметром и показывает максимальное время, за которое может произойти MICRONUM изменений цены, чтобы такое изменение считалось микроколебанием. Начальное значение - 100 000 000(формат времени в наносекундах).

SPOOFPRICE - параметр (процент), обозначает интервал по цене, в котором должны находится заявки, которые выставляются и отменяются

во время микроколебания.

SPOOFDELTA - параметр (положительное рациональное число), показывает в какой интервал времени после постановки суперские заявки должны быть отменены.

SPOOFVALUE - параметр (процент), показывает какой минимальный объем должен быть у спуферских заявок, которые проходят по цене и по отмене, чтобы каждую из таких заявок можно было признать подозрительной на спуфинг. С одной стороны, характеризует стратегию спуфера: сколько ложных заявок он выставляет. А с другой стороны, характеризует точность алгоритма, так как влияет на тот минимальный объем который должен быть достигнут, чтобы заявки были признаны подозрительными.

SPOOFER - итог работы алгоритма (0 или 1). Показывает те заявки, которые были признаны подозрительными.

6.3 Программная реализация алгоритма обнаружения спуфинга

Код алгоритма на языке Python находится в Приложении 4.

6.3.1 Этап 1: выделение микроколебаний подходящих для спуфинга

Данный этап полностью аналогичен этапу 1 при симуляции спуфинга. Советую ознакомиться с подробным алгоритмом действий в блоке программной симуляции спуфинг-стратегии. Здесь же мы лишь напомним основные шаги этого этапа.

1. Заполнение вспомогательного массива Trends
2. Нахождение подходящих временных интервалов, записанных в массив `Micro_times`.

6.3.2 Этап 2: нахождение подозрительных на спуфинг заявок

На втором этапе, располагая массивом микроколебаний `Micro_times`, необходимо итеративно рассмотреть все временные промежутки, которые в

нем содержатся, и перейти к рассмотрению заявок и сделок совершенными во время каждого из таких периодов. Запускаем итерацию по заявкам.

Шаг 1: поиск заявок подходящих по цене. Находим установленные в этот период заявки, которые удовлетворяют следующему условию: цена заявки лежит в интервале [(1-SPOOFPRICE)*`best_price` ; `best_price`].

Шаг 2: проверка отмены заявок. После того, как были отобраны подходящие по цене заявки, необходимо проверить время отмены каждой такой заявки. Если заявка была удалена в течение времени SPOOFDELTA после постановки, то оставляем такую заявку. Если же заявка была удалена после SPOOFDELTA или была исполнена, то она убирается из рассмотрения.

Шаг 3: проверка соответствия объема. Оставшиеся заявки проверяются на соответствие объему. Если суммарный объем оставшихся заявок менее SPOOFVALUE * TOTALVALUE, то в данном микроколебании спуфинга не обнаружено и можно переходить к следующему интервалу. А вот если объем оставшихся заявок более SPOOFVALUE * TOTALVALUE, то всем таким заявкам присваивается статус подозрительных на спуфинг.

На этом разработка алгоритма закончена и можно переходить к тестированию.

7. Тестирование алгоритма обнаружения спуфинга

7.1 Нахождение параметров по тренировочным данным

Теперь, когда в нашем распоряжении есть алгоритм, который теоретически и практически способен определять подозрительные заявки, проблему можно сформулировать следующим образом: какие значения параметров должны быть, чтобы алгоритм работал максимально точно и корректно. Одних теоретических выкладок будет недостаточно, поэтому необходимо протестировать алгоритм на тренировочных данных, которые точно содержат спуфинг.

Основной целью тестирования является определение параметров алгоритма: MICRONUM, MICRODELTA, а также SPOOFPRICE, SPOOFDELTA и SPOOFVALUE. Для каждого параметра мы зададим определенные рамки возможных значений, и для всевозможных комбинаций проведем исследования на тестовых данных. В зависимости от точности нам останется только выбрать наилучшие значения параметров.

Интервалы значения параметров. Сначала определим интервалы допустимых значений для параметров.

MICRONUM = [2, 3, 4, 5, 6, 7, 8, 9, 10]

MICRODELTA = [5сек, 10 сек, 20 сек, 40сек, 1 мин, 2 мин, 5 мин]

SPOOFPRICE = [0.005, 0.01, 0.02, 0.05, 0.1, 0.15]

SPOOFDELTA = [5сек, 10 сек, 20 сек, 40сек, 1 мин, 2 мин, 5 мин]

SPOOFVALUE = [0.01, 0.05, 0.1, 0.15, 0.2]

Границы этих параметров определены исходя из теоретических рассуждений о стратегии спуфинга. Если в процессе тестирования какой-то из параметров будет принимать граничное значение, то интервал допустимых значений будет пересмотрен.

Метрики оценивания эффективности. Для оценивания наилучших параметров будем использовать несколько метрик:

Accuracy - доля правильных ответов алгоритма

Точность (Precision) - доля объектов, названных классификатором подозрительными на спуфинг и при этом действительно являющимися спуфер-заявками

Полнота (Recall) - доля спуферов, которых нашел алгоритм из всех спуферских заявок.

F-мера (F_β) — среднее гармоническое precision и recall. Определяется по формуле 12. Это метрика качества классиффикации, которая применимыа в условиях несбалансированных выборок[10]. В нашем случае как раз спуферских заявок гораздо меньше чем всех остальных, поэтому эта метрика будет полезна. Значение параметра $\beta = 1$.

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

Рис. 12: Формула F-меры

Результаты. Тестирование алогритма проводилось на тренировочных датасетах. В каждый из которых по методу, описанному ранее, были добавлены спуферские заявки. Выбор этих 10 активов был случаен и описан в блоке обработки данных.

Для того, чтобы определить лучшие параметры алгоритма, вычисляем контрольные метрики для каждого набора параметров и для каждого актива, а затем в итоговую таблицу заносим среднее арифметическое значение за эти 10 активов. Так как всевозможных комбинаций параметров MICRONUM, MICRODELTA, SPOOFPRICE, SPOOFDELTA, SPOOFVALUE было 13 230, то хранили в памяти алгоритма только лучшую комбинацию для ключевой метрики . Ключевой метрикой была выбрана F-мера, так как она сочетает в себе точность и полноту. Результат подбора параметров представлен в таблице 13.

По результатам работы алгоритма подбора параметров останавливаем свой выбор на комбинации:

MICRONUM = 5

[10]//bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html

Комбинация								
MICRONUM	MICRODELTA	SPOOFPRICE	SPOOFDELTA	SPOOFVALUE	Accuracy	Precision	Recall	f1-score
5	10	0,01	20	0,4	99,18%	0,99	0,97	0,98

Рис. 13: Лучший набор параметров при максимизации F-меры

MICRODELTA = 10
SPOOFPRICE = 0,01
SPOOFDELTA = 20
SPOOFVALUE = 0,4

На самом деле, это очень одидаемый результата, так как при моделировании спуфинг стратегии мы закладывали именно эти параметры. Алгоритм верно нашел параметры, отличив их от множества других комбинаций. Значения метрик тоже приближены к идеальным. Небольшое отклонение можно объяснить тем, алгоритм может иногда приписывать статус спуфера заявкам, которые не были добавлены искусственно в данные. Это обуславливается либо случайностью, либо тем, что в данных присутствует настоящий спуфер. Также причиной отклонения значений от идеальных является тот факт, что при симуляции стратегии мы не вводили параметр SPOOFDELTA, а привязывали отмену заявок непосредственно к изменению цены.

7.2 Проверка алгоритма на тестовых данных (без симуляции)

Теперь мы проводим тестирование на чистых данных, в которых нет искусственно внедренного спуфинга. В них произведена лишь предварительная предобработка, которая приводит входные данные к формату, с которым алгоритму удобно работать: выделены BESTPRICE и TOTALVOLUME. Сейчас для нас главная задача найти хоть несколько заявок, которые потенциально могут быть частью стратегии спуфинга.

Алгоритм будет тестироваться на 49 датасетах из топ-59 российских компаний по количеству заявок в день. Напомню, что выбраны компании были произвольным образом. Результатам работы алгоритма будет количество обнаруженных спуферских заявок. Рассматривать результаты наибо-

лее интересно сопоставляя их с общим количеством заявок за день торгов и количеством выявленных микроколебаний. Изучим таблицу 14.

Акции	Количество заявок	количество микроколебаний	спуфинг-заявки	Акции	Количество заявок	количество микроколебаний	спуфинг-заявки
AFKS	21519.0	680	8	MSNG	24810.0	599	0
AFLT	19120.0	762	38	MSST	8587.0	396	0
AKRN	22662.0	321	6	MTLR	50466.0	867	872
ALRS	30687.0	775	0	MTSS	21652.0	656	0
APTK	7743.0	379	14	NLMK	18436.0	721	120
BANEP	20910.0	397	0	NVTK	19252.0	606	266
BSPB	7104.0	325	0	OGKB	14341.0	458	0
CBOM	17612.0	284	0	PHOR	20653.0	447	22
CHMF	18686.0	623	0	PLZL	36136.0	668	0
CHMK	9268.0	310	0	POLY	80122.0	825	8
DSKY	8357.0	433	0	RASP	28460.0	652	4
ENPL	7559.0	327	0	RSTI	22469.0	494	0
FEES	17161.0	641	0	RTKM	20726.0	649	0
FIVE	11861.0	313	0	RUAL	41482.0	813	0
GAZP	69970.0	974	786	SBER	110586.0	991	1238
GMKN	35512.0	829	238	SFIN	87376.0	473	0
HYDR	18563.0	649	32	SNGS	11529.0	528	0
IRAO	16077.0	476	12	SNGSP	26100.0	762	0
IRKT	7540.0	338	0	TGKA	39304.0	558	0
LKOH	58028.0	901	0	TRNFP	90356.0	278	0
LNTA	8269.0	383	9	UPRO	15402.0	306	0
LSRG	7346.0	428	0	URKA	16089.0	368	0
MAGN	14482.0	523	0	UWGN	7099.0	162	0
MOBB	7332.0	367	38	YNDX	25704.0	787	76
MOEX	73074.0	917	21				

Рис. 14: Количество выявленных подозрительных заявок

Спуфинг был найден в 7 активах из 49. На изображении они отмечены желтым. Остальные подозрительные заявки маловероятно окажутся спуфингом, так как из недостаточно для реализации полноценной стратегии спуфинга. Скорее всего это можно назвать погрешностями алгоритма. Такой результат вполне правдоподобен, так как спуфинг был обнаружен не во всех стаканах и количество подозрительных заявок в пределах разумного и соответствует теоретическому подходу к стратегии спуфинга.

Интересно, что нет прямой связи между количеством заявок и количеством найденных микроколебаний, хотя разумно было предполагать, что чем больше заявок в стакане актива, тем больше возможностей для спуфера применить свою стратегию. Это можно объяснить волатильностью акций: некоторые активы имеют большую волатильность и более подвержены посторонним влияниям и изменения, другие же, наоборот, почти не

растут и не падают в цене длительный промежуток времени. Конечно, для спуфера более волатильный актив подходит лучше.

Сбербанк, ожидаемо, получился лидером по количеству подозрительных заявок на спуфинг, так как за торговый день было выставлено более 110 тысяч заявок, из которых 1238 потенциально могут быть частью нелегальной стратегии спуфинга.

В целом, это очень приятный результат, который свидетельствует о корректной работе алгоритма.

Выводы

Подводя итог проделанной работы, хотелось бы отметить, что поставленная цель исследования достигнута. Нам необходимо было разработать алгоритм, способный обнаружить спуфинг манипуляцию на фондовом рынке. Такой алгоритм получен. Тестирование показало его корректность - значит, проделанный путь был не напрасен.

Помимо глобальной цели, в научной-исследовательской работе был определен ряд задач, решение которых можно сформулировать следующим образом

1. Проведен обзор литературы и выявлены общие точки соприкосновения с другими исследованиями. Так как задача обнаружения манипуляции является лишь звеном в цепочке борьбы с манипуляцией, то необходимо сочетать ее решения с другими элементами. Технически еще необходимо доказать, что выявленные подозрительные заявки являются именно спуфингом - здесь будут полезны работы Веллмана и исследования уголовно-правовых нормы.

2. Успешно проведена подготовка данных с московской биржи. Мы убедились, что формат данных Full Orders Log подходит для работы со спуфингом как для симуляции поведения нарушителя, так и для разработки алгоритма по его обнаружению.

3. Реализована симуляция стратегии спуфинга. Реализован алгоритм для моделирования стратегии спуфинга, который позволяет с помощью параметров задавать различные разновидности спуфинга и экспериментировать с ними. Получены данные, которые доказали свою корректность благодаря тестированию и верной настройке алгоритма обнаружения. И данная симуляция дала возможность провести настройку и подбор параметров для алгоритма обнаружения. Но из-за вычислительных ограничений не удалось провести симуляцию большего количества данных.

4. Разработан алгоритм обнаружения спуфинга. В его основу легли теоретические выкладки о стратегии спуфинга и опыт моделирования поведения спуфера в биржевом стакане. Для практического применения, алгоритм был реализован на языке Python, что позволяет использовать его с целью пост-контроля реальных биржевых данных на спуфинг.

5. Успешно проведено тестирование алгоритма. Алгоритм показал достойные результаты естирования как на тренировочных данных с искусственно внедренным спуфером, так и на тестовых данных с биржи, в которых было обнаружено 3808 подозрительных заявок.

В заключение, хотелось бы немного рассказать о дальнейших перспективах. Алгоритм обнаружения спуфинга, который был разработан после более тщательных тестов, может найти применение как в государственных структурах, заинтересованных в наказании спуфинга, так и на биржах с целью контроля и пресечения нелегальных и опасных действий. А еще есть вопрос, над которым тоже стоит подумать: как предотвратить применение стратегии спуфинга? Возможно, есть способы, чтобы такая стратегия стала неэффективной или невозможной... На этой интересной идее я хотел бы и завершить научно исследовательскую работу.

Список литературы

[1] Cumming, D., and S. Johan, 2008, Global market surveillance, American Law and Economics Review 10, 454-506.

[2] Xitong Wang, Michael P. Wellman. 2017. Spoofing the Limit Order Book: An Agent-Based Model //The AAAI-17 Workshop on Computer Poker and Imperfect Information Games – WS-17-06 – University of Michigan – P. 367-375

[3] Бобков О. В. 2017. Манипулирование рынком: проблемы эффективности уголовно-правового запрета // Юридическая наука и правоохранительная практика 2 (40) 2017 - Стр. 206-211

[4] Albert S. Kyle (University of Maryland, Robert H. Smith School of Business), S. Viswanathan (Duke University, Fufua School of Business). How to Define Illegal Price Manipulation. Draft: January, 2008

[5] Xitong Wang, Yevgeniy Vorobeychik, Michael P. Wellman. 2018. A Cloaking Mechanism to Mitigate Market Manipulation //Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18) – P. 541-547

[6] Elaine Wah, Mason Wright, Michael P. Wellman. 2017. Welfare Effects of Market Making in Continuous Double Auctions // Journal of Artificial Intelligence Research 59 (2017) P.613-650

[7] Yevgeniy Vorobeychik, Christopher Kiekintveld, Michael P. Wellman – University of Michigan Computer Science Engineering. Conference Paper – January 2006. Empirical Mechanism Design; Methods, with Application to a Supply-Chain Scenario

[8] AES Analysis. 2012. High Freequency Trading – Measurment, Detection and Response

Приложение 1 Програмный код предобработки данных

```
\texttt{import numpy as np
import pandas as pd
import datetime
import matplotlib.pyplot as plt
import seaborn as sns
import random
from sklearn.model_selection import train_test_split
data=pd.read_csv('OrderLog20181229.csv', sep=';')

Test = []
Train = []
DROPSTART = 10000000000
DROPEND = 10000000000

# отбираем топ-59 активов по количеству заявок за день
top = data.groupby('SECCODE')['PRICE'].describe().round(2).
    sort_values(by = 'count',ascending=False).head(59)
# раделяем данные на тестовые - проверка алгоритма
index_list = list(top.index)
# train_seccode, test_seccode  = train_test_split(np.array(index_list)
train_seccode, test_seccode  = train_test_split(np.array(index_list),
for seccode in index_list:
    # выделяем конкретный актив
    st = data[(data.SECCODE == seccode)].reset_index(drop=True)

    # рассматриваем заявки на покупку
    st_b = st[(st.BUYSELL == "B")].reset_index(drop=True)
    # удаляем данные открытия и закрытия торгов
    st_b = st_b.loc[st_b['TIME'] > 100000000000 + DROPSTART]
    st_b = st_b.loc[st_b['TIME'] < 184500000000 - DROPEND]
```

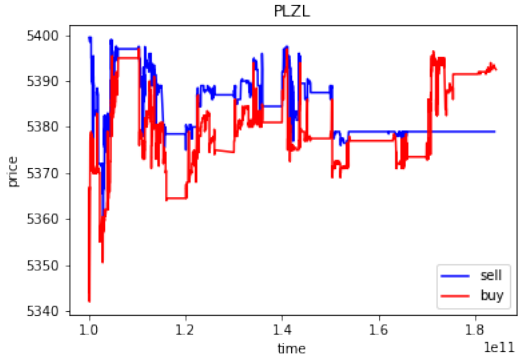
```
# добавляем столбец 'BEST_PRICE'
st_b['BEST_PRICE'] = ''
st_b['TOTALVOLUME'] = ''
D = {}
total_volume = 0

for i, j in st_b.iterrows():
    action = st_b.at[i, 'ACTION']
    price = st_b.at[i, 'PRICE']
    volume = st_b.at[i, 'VOLUME']
    number = st_b.at[i, 'ORDERNO']

    if action == 1:
        total_volume = total_volume + volume
        D.update({number:[price, volume]})

    elif action == 0:
        D.pop(number)
        total_volume = total_volume - volume

    elif action == 2:
        total_volume = total_volume - volume
        old_volume = D[number][1]
        if volume == old_volume:
            D.pop(number)
        else:
            D.update({number:[price, old_volume - volume]})

    max_price = max(list(D.values()))[0]
    st_b.at[i, 'BEST_PRICE'] = max_price
    st_b.at[i, 'TOTALVOLUME'] = total_volume

# рассматриваем заявки на продажу
```

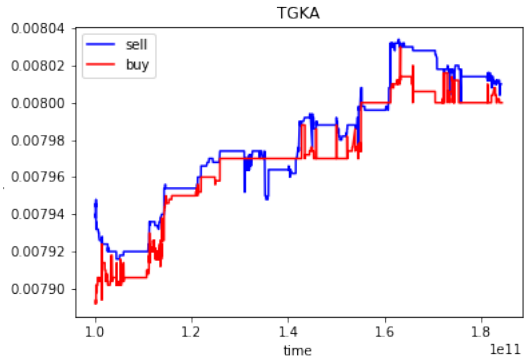
```
st_s = st[(st.BUYSELL == "S")].reset_index(drop=True)
st_s = st_s.loc[st_s['TIME'] > 100000000000 + DROPSTART]
st_s = st_s.loc[st_s['TIME'] < 184500000000 - DROPEND]
# добавляем столбец 'BEST_PRICE' и 'TOTALVOLUME'
st_s['BEST_PRICE'] = ''
st_s['TOTALVOLUME'] = ''
D = {}
total_volume = 0
for i, j in st_s.iterrows():
    action = st_s.at[i, 'ACTION']
    price = st_s.at[i, 'PRICE']
    volume = st_s.at[i, 'VOLUME']
    number = st_s.at[i, 'ORDERNO']

    if action == 1:
        D.update({number:[price, volume]})
        total_volume = total_volume + volume

    elif action == 0:
        D.pop(number)
        total_volume = total_volume - volume

    elif action == 2:
        old_volume = D[number][1]
        total_volume = total_volume - volume
        if volume == old_volume:
            D.pop(number)
        else:
            D.update({number: [price, old_volume - volume]})
    min_price = min(list(D.values()))[0]
    st_s.at[i, 'BEST_PRICE'] = min_price
    st_s.at[i, 'TOTALVOLUME'] = total_volume
```

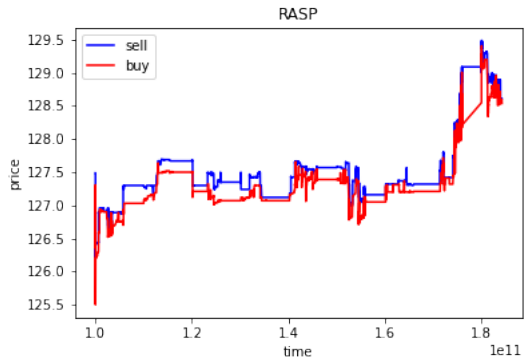
```
if seccode in train_seccode:
    Train.append([st_b, st_s])
    print('успех\n')
elif seccode in test_seccode:
    Test.append([st_b, st_s])
    print('успех\n')
else:
    print('error')
```

Приложение 2. Примеры графиков лучшей цены и покупки

Графики наибольшей цены покупки и наименьшей цены продажи для некоторых активов







Приложение 3. Програмный код симуляции спуфинг-стратегии

```
    # определяем параметры
MICRONUM = 5
MICRODELTA = 100000000

# Этап 1
Micro_times = []
for train in Train:

    Micro_times_st = []
    for st in [st_b, st_s]:
        st['SPOOFER'] = 0  #
        prev_best_price = 0
        Trends = []
        Times = []
        micro_is_ok = 0
        for i, j in st.iterrows():
            best_price = st.at[i, 'BEST_PRICE']
            time = st.at[i, 'TIME']

            if prev_best_price < best_price:
                trend = 1
            elif prev_best_price > best_price:
                trend = 0
            else:
                continue

            if len(Trends) == 0:
                start_time = time
                Trends.append(trend)
```

```
            elif Trends[-1] == trend:
                if time - start_time < MICRODELTA:
                    Trends.append(trend)
                else:
                    if micro_is_ok == 0:
                        start_time = 0
                    else:
                        Times.append([start_time, time])
                        micro_is_ok = 0
                    Trends = []

            elif Trends[-1] != trend:
                if micro_is_ok == 0:
                    start_time = 0
                else:
                    Times.append([start_time, time])
                    micro_is_ok = 0
                Trends = []

            if len(Trends) == MICRONUM:
                micro_is_ok = 1
        print(len(Times))
        Micro_times_st.append(Times)

    Micro_times.append(Micro_times_st)

# Этап 2

count = 0
prev_best_price = 0
count = 0
Train_new = []
SPOOFVALUE = 0.4
```

```
SPOOFPRICE = 0.01
for i1 in range(len(Micro_times)):  # seccodes
    print('Начинаю актив', i1)
    Train_new_st = []
    for i2 in range(len(Micro_times[i1])):
        train_new = Train[i1][i2].copy()
        SECCODE = train_new.at[1, 'SECCODE']
        BUYSELL = train_new.at[2, 'BUYSELL']
        for times in Micro_times[i1][i2]:  # time pairs
            st_micro = Train[i1][i2].copy()
            st_micro = st_micro.loc[st_micro['TIME'] >= times[0]]
            st_micro = st_micro.loc[st_micro['TIME'] <= times[1]]

            k = 0
            Cancel = []
            for i, j in st_micro.iterrows():

                best_price = st_micro.at[i, 'BEST_PRICE']
                volume = st_micro.at[i, 'VOLUME']
                time = st_micro.at[i, 'TIME']
                TOTALVALUE = st_micro.at[i, 'TOTALVOLUME']

                if k == 0:
                    k = 1
                    prev_best_price = best_price
                    time_start = time

                # ищем все заявки установленные в данный промежуток
                /времени, подходящие по цене
                if best_price != prev_best_price:
                    if k != 1:  # отменяем предыдущие
                        for number in Cancel:
                            new_row =
```

```
            {'NO': 0,
            'SECCODE': SECCODE,
            'BUYSELL': BUYSELL,
            'TIME': time + 100 + int(number[2:]),
            'ORDERNO': number,
            'ACTION': 0,
            'PRICE': int(train_new.loc
                [train_new.loc
                [train_new.ORDERNO == number]
                .index, 'PRICE']),
            'VOLUME': int(
                          train_new.loc
                          [train_new.loc
                          [train_new.ORDERNO
                          == number].index,
                          'VOLUME']),
            'TRADENO': 0,
           'TRADEPRICE': 0,
            'BEST_PRICE': best_price,
            'TOTALVOLUME': TOTALVALUE,
            'SPOOFER': 1}

            train_new = train_new.append(new_row,
            ignore_index=True)
            Cancel.remove(number)

k = 2
n = random.randint(2, 10)
for i in range(n):
    count = count + 1
    number = 's_' + str(count)
    new_row = {
    'NO': 0,
```

```
                    'SECCODE': SECCODE,
                    'BUYSELL': BUYSELL,
                    'TIME': time_start + 100 + i,
                    'ORDERNO': number,
                    'ACTION': 1,
                    'PRICE': prev_best_price + (2 *
                                i2 - 1) * SPOOFPRICE
                                * prev_best_price,
                    'VOLUME': TOTALVALUE * SPOOFVALUE / n,
                    'TRADENO': 0,
                    'TRADEPRICE': 0,
                    'BEST_PRICE': prev_best_price,
                    'TOTALVOLUME': TOTALVALUE,
                    'SPOOFER': 1}

                    train_new = train_new.append(
                    new_row, ignore_index=True)
                    Cancel.append(number)
    train_new.sort_values(by=['TIME', 'NO']).reset_index()
    Train_new_st.append(train_new)
Train_new.append(Train_new_st)
```

Приложение 4 Програмный код алгоритма обнаружения спуфинга

```
    # Этап 1
MICRONUM = 5
MICRODELTA = 1100000000
Micro_times = []

for trains_st in [Test[0]]:
    Micro_times_st = []
    for st in trains_st:
        st['SPOOFER'] = 0
        prev_best_price = 0
        Trends = []
        Times = []
        micro_is_ok = 0
        for i, j in st.iterrows():
            best_price = st.at[i, 'BEST_PRICE']
            time = st.at[i, 'TIME']

            if prev_best_price < best_price:
                trend = 1
            elif prev_best_price > best_price:
                trend = 0
            else:
                continue

            if len(Trends) == 0:
                start_time = time
                Trends.append(trend)

            elif Trends[-1] == trend:
                if time - start_time < MICRODELTA:
```

```
                    Trends.append(trend)
                else:
                    if micro_is_ok == 0:
                        start_time = 0
                    else:
                        Times.append([start_time, time])
                        micro_is_ok = 0
                    Trends = []

            elif Trends[-1] != trend:
                if micro_is_ok == 0:
                    start_time = 0
                else:
                    Times.append([start_time, time])
                    micro_is_ok = 0
                Trends = []

            if len(Trends) == MICRONUM:
                micro_is_ok = 1
        print(len(Times))
        Micro_times_st.append(Times)

    Micro_times.append(Micro_times_st)


# Этап 2
SPOOFPRICE = 0.01
SPOOFDELTA = 200000000 # 2 min
SPOOFVALUE = 0.4
st_b['CHECK_SPOOFER'] = ''
count = 0
prev_best_price = 0
Spoof_numbers = []
```

```
for i1 in range(len(Micro_times)): #seccodes
    print('Начинаю актив', i1)
    Spoof_numbers_st = []
    for i2 in range(len(Micro_times[i1])):# buy and sell
        Spoof_numbers_st_t = []
        for times in Micro_times[i1][i2]: # time pairs
            st_micro = Train[i1][i2].copy()
            st_micro = st_micro.loc[st_micro['TIME'] >= times[0]]
            st_micro = st_micro.loc[st_micro['TIME'] <= times[1]]

            k = 0
            for i, j in st_micro.iterrows():
                best_price = st_micro.at[i, 'BEST_PRICE']
                volume = st_micro.at[i, 'VOLUME']
                time = st_micro.at[i, 'TIME']
                TOTALVALUE = st_micro.at[i, 'TOTALVOLUME']

                if k == 0:
                    k = 1
                    prev_best_price = best_price
                    time_start = time

                if best_price != prev_best_price:
                    # ищем все заявки установленные в
                    данный промежуток времени, подходящие
                    по цене
                    table = st_micro.loc[st_micro['TIME']
                    >= time_start]
                    table = table.loc[table['TIME']
                    <= time]
                    table = table.loc[table['ACTION']
                    == 1]
                    table = table.loc[table['PRICE']
```

```
                    >= (1-SPOOFPRICE)*best_price]

                    time_start = 0

                    # проверяем отмену заявок
                    for l, m in table.iterrows():
                        table_2 = Train[i1][i2].copy()
                        number = table.at[l, 'ORDERNO']
                        put_time = table.at[l, 'TIME']
                        table_2 = table_2.loc[table_2['ORDERNO']
                        == number]
                        table_2 = table_2.loc[table_2['ACTION']
                        == 0]
                        try:
                            cancel_time = int(table_2['TIME'])
                            # try потому что заявка могла
                            быть исполнена
                        except:
                            continue

                        if cancel_time - put_time < SPOOFDELTA:
                            volume_set = table_2['VOLUME'].sum()
                            if volume_set > SPOOFVALUE*TOTALVALUE:
                                Spoof_numbers_st_t.append(number)
                                count = count + 1
    Spoof_numbers_st.append(Spoof_numbers_st_t)
Spoof_numbers.append(Spoof_numbers_st)
```