МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий и управления в технических системах
(полное название института)

кафедра «Информационные системы»
(полное название кафедры)

Пояснительная записка

к выпускной квалификационной работе бакалавра

на тему Система анализа цен на международных рынках

Выполнил: студент IV курса, группы: ИС/б-43 о

Направления подготовки (специальности) 09.03.02

Информационные системы и технологии
(код и наименование направления подготовки (специальности))

профиль (специализация) Информационные системы и технологии

Гавлюк Андрей Андреевич
(фамилия, имя, отчество студента)

Руководитель Бондарев В. Н. к.т.н. доцент, доцент кафедры «Информационные системы»
(фамилия, инициалы, степень, звание, должность)
Сметанина Т. И. ст. преп. кафедры «Информационные системы»
(фамилия, инициалы, степень, звание, должность)

Дата допуска к защите « ____ » _____ 20 ___ г.

Зав. кафедрой _____ И. П. Шумейко
(подпись) (инициалы, фамилия)

2018 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы. Во все времена вопрос о защите и приумножении собственных средств, всегда являлся актуальным, не зависимо от того это средства простого рабочего или крупной компании. На текущий момент кроме физического риска утраты средств, актуальным является риск утраты экономический, например, падение курса национальной валюты, цены драгоценных металлов, цены на акции компаний и других средств сбережения ценностей. В качестве примера приведём великую депрессию в 1929 году, когда из-за финансового кризиса пострадали многие страны, особенно сильно пострадали такие страны как США, Канада, Великобритания, Германия, Франция. Из-за неправильного инвестирования (а также из-за использования опасного риск-менеджмента) многие люди лишались домов, автомобилей и другого ценного имущества. В России в качестве негативного примера обесценивания валюты можно привести кризис 1998 года, когда за полгода, цена доллара возросла с 6 рублей до более чем 20 рублей за доллар.

Так же в качестве позитивного примера использования спекулятивной торговли можно привести деятельность банков, инвестиционных фондов, хедж-фондов, которые обеспечивают сохранность средств вкладчиков.

Целью данной работы является: Разработка торгового робота для автоматизации торговли:

– изучить современные способы технического анализа рынка;

– выявить (экспериментально) возможность автоматизации, основных торговых стратегий;

– разработать рекомендованные оптимальные параметры для максимизации доходов, при заданном риск-менеджменте.

Предметом и объектом исследования в данной работе является состояние рынка в текущий и прошедшие моменты времени, а также поиск оптимальных параметров для торговли по выбранной торговой стратегии.

Научной новизной данной работы является применение «самооптимизирующегося» алгоритма для создания торгового робота.

Практическим назначением данной работы является создание торгового робота, позволяющего снизить риски при спекулятивной торговле.

Структура работы. Данная работа состоит из пояснительной записки, включающей в себя введение, три раздела, выводы, список использованных источников, приложения и программного модуля на электронном носителе.

Первый раздел посвящён описанию процессов, происходящих при автоматизации торговли, описанию существующих подходов к решению задачи автоматизации торговли и обзору существующих торговых терминалов.

Второй раздел описывает архитектуру торгового робота, основанную на обработке событий торгового терминала. Так же во втором разделе обосновывается выбор критериев и описываются применяемые инструменты технического анализа. Так же даётся краткое математическое обоснование применяемых алгоритмов.

Третий раздел описывает наиболее часто используемые торговые стратегии, рассматриваемые в данной работе. Так же в третьем разделе описывается основные принципы построения генетических алгоритмов.

1 АНАЛИЗ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ В ЗАДАЧЕ АВТОМАТИЗАЦИИ ТОРГОВЛИ С ЦЕЛЬЮ ИССЛЕДОВАНИЯ МЕТОДИК АНАЛИЗА ЦЕН

1.1 Стандарты описания процессов в рамках прикладной предметной области

Алгоритм работы в задаче автоматизации торговли кратко можно описать следующим образом:

1) торговый терминал получает текущее значения котировок;

2) торговый робот производит анализ текущих котировок;

3) в зависимости от собственных настроек, результатов технического анализа текущего состояния рынка и текущего состояния списка открытых сделок возможен один из следующих исходов:

– открытие новой позиции;

– модификация уже существующей позиции;

– частичное закрытие позиции;

– полное закрытие позиции;

– ожидание следующей котировки;

– информирование пользователя, о каком-либо событии;

Для описания процессов в рамках прикладной предметной области будут разработаны такие средства отображения и описания процессов как DFD-диаграммы и IDEF-диаграммы. Для начала автор хотел бы разъяснить некоторые применяемые понятия.

Автоматической торговлей будем называть ту торговлю, при которой пользователь терминала, не принимает никакого участия в принятии решения об установке ордера, а лишь наблюдает за действиями робота, и по усмотрению может закрывать сделки, открытые роботом, которые считает убыточными. Автоматическая торговля, является эффективной, в том случае, если

пользователь не считает необходимым проводить всё время за торговым терминалом, а предпочитает формализовать свой алгоритм торговли и автоматизировать её. Так же автоматическая торговля является эффективной в случае, когда между открытием и закрытием сделки должно пройти всего несколько секунд или доли секунды.

Полуавтоматической торговлей будем называть ту торговлю, при которой пользователь лишь получает извещение от робота о том, что следовало бы в текущий момент открыть сделку по тому или иному инструменту в том или ином направлении. Полуавтоматическая торговля применяется как дополнительный советник при принятии решения о открытии или закрытии сделки, но окончательное решение остаётся за пользователем.

1.1.1 Разработка DFD-диаграммы

Работа торгового робота начинается с того что происходит инициализация, затем происходит проверка запущен ли робот в режиме тестирования или в режиме торговли в зависимости от результатов проверки дальнейшие алгоритмы могут отличаться.

Если происходит запуск в режиме «Робот», то предаются только котировки необходимые для анализа рынка на текущий момент. В этот момент робот проводит анализ текущей ситуации на рынке и может выполнять следующие действия:

– открывать торговые позиции;

– закрывать полностью торговые позиции;

– закрывать частично торговые позиции;

Если происходит запуск в режиме «Тестер», то выполняется тестовая симуляция торговли по историческим данным за указанный временной диапазон, с целью получения результатов эффективности торговой системы с текущими параметрами. Для прохождения тестирования по указанному временному диапазону торговому роботу передаются необходимые исторические данные за указанный период, а также дополнительная информация

необходимые для анализа рынка на каждый из тех моментов времени. Затем после завершения симуляции торговли робот сообщает информацию характеризирующую эффективность тестируемой торговой системы.

При завершении работы торгового робота производится удаление структур, созданных роботом, очищение памяти, восстановление терминала до прежнего состояния.

Для лучшего понимания выше описанных процессов рассмотрим DFD диаграмму (рисунок 1).

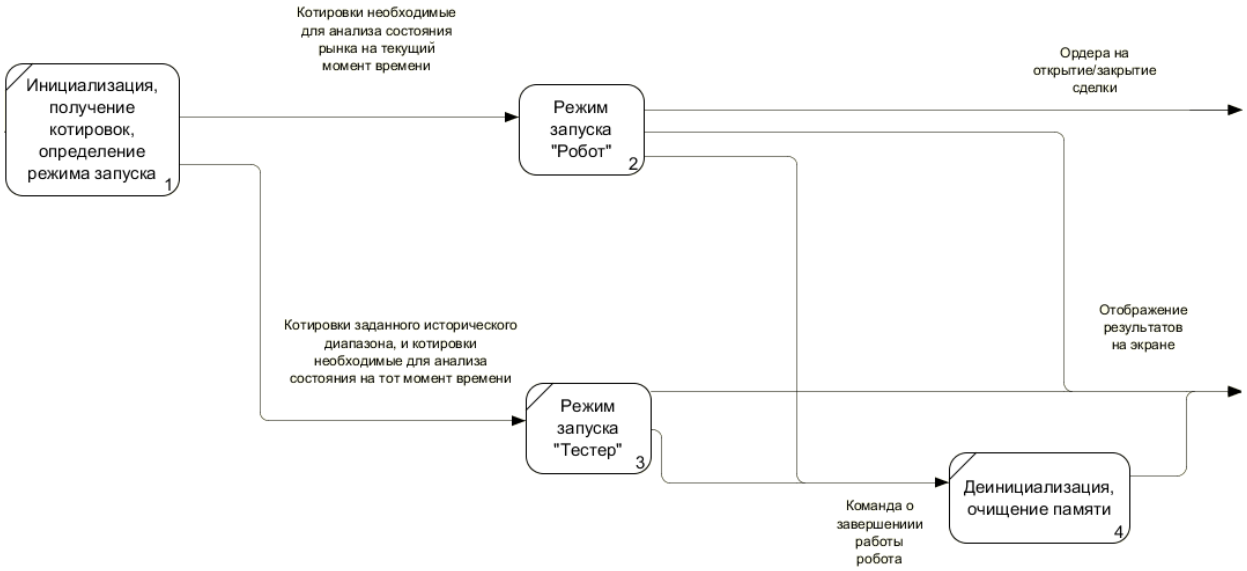

Рисунок 1 – DFD диаграмма основного процесса

Далее более детально будет рассмотрена диаграмма работы в режиме «Робот» (рисунок 2).

Для проведения анализа текущего состояния рынка необходимо произвести некоторые вычисления, как правило, для проведения этих вычислений необходимо владеть информацией о предыдущем состоянии рынка в определённые моменты времени, для этого производится запрос о состояниях рынка в интересующие нас моменты в прошлом.

Затем производится проверка: считает ли пользователь необходимым выполнение автоматической или полуавтоматической торговли

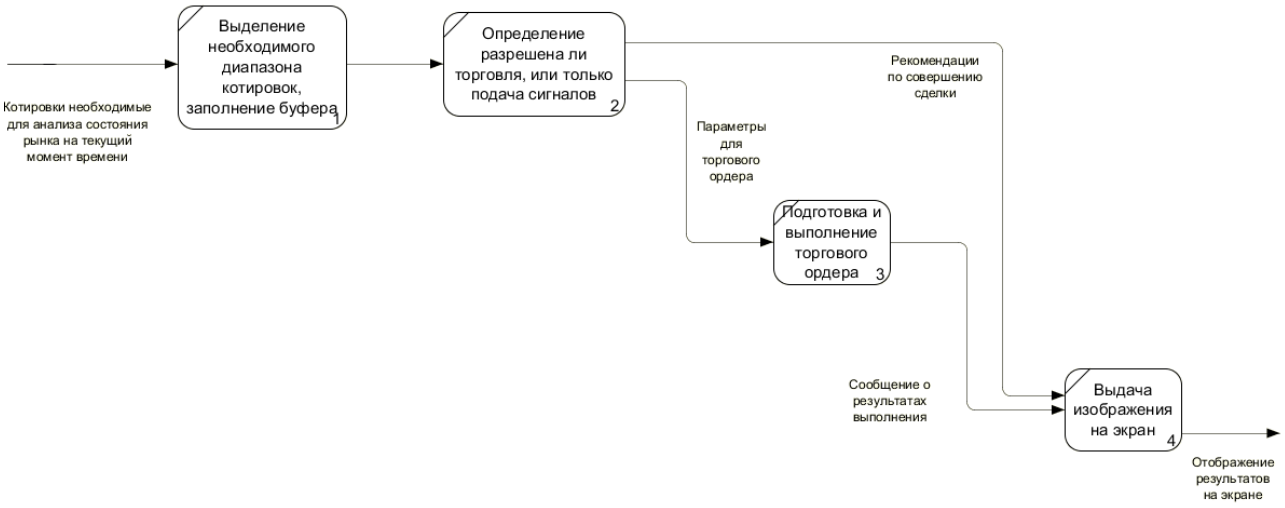

Рисунок 2 – DFD диаграмма процесса запуска в режиме «Робот»

В случае если выбран режим автоматической торговли, то выполняется подготовка и выполнение торгового ордера. Торговый ордер представляет собой своего рода запрос к базе данных на сервере, на котором проводятся торги.

Как можно понять, из всего выше перечисленного, данная функция предоставляет крайне информативный способ передачи ордера-запроса на сервер брокерской компании.

Затем в зависимости от настроек робота пользователь может увидеть одно из следующих сообщений:

– сообщение об открытии позиции;

– сообщение об закрытии позиции;

– сообщение об рекомендации открытии(закрытии) позиции.

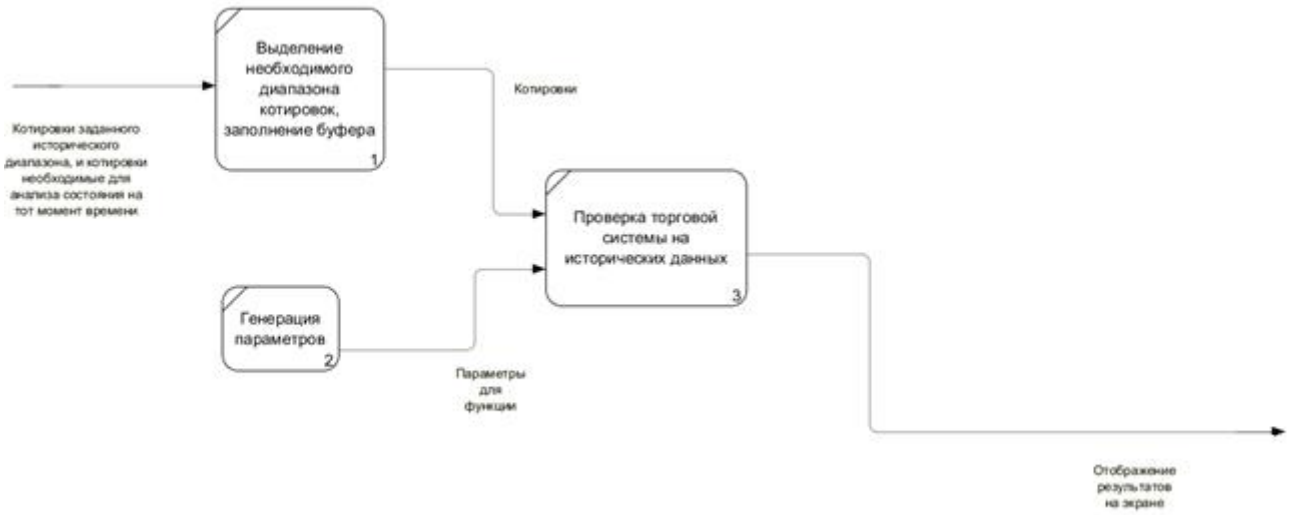

Рисунок 3 – DFD диаграмма процесса запуска в режиме «Тестер»

Теперь опишем запуск в режиме тестера так же, как и в режиме «Робота» (рисунок 3) для запуска в данном режиме, нам также необходимо получить котировки для симуляции торговли. В дальнейшем производится оптимизация параметров данной стратегии и исследование результатов. Оптимизация стратегии достигается за счёт изменения разных параметров и повторной симуляции торговли оптимизация ведётся в сторону увеличения прибыли, полученные данные, отображаются на мониторе.

1.2 Обзор существующих подходов к решению задачи автоматизации торговли

Для решения данной задачи существует несколько подходов. Ниже автор опишет некоторые из них.

Первым подходом, удобным для людей не знакомых с программированием является подход, подразумевающий собой формализацию торговой стратегии, описывающий, когда какие действия следует совершать. Данный подход будем называть созданием торгового робота с помощью конструктора, примером

реализации данного подхода является среда разработки TradeScript, изображённая на рисунке 4.

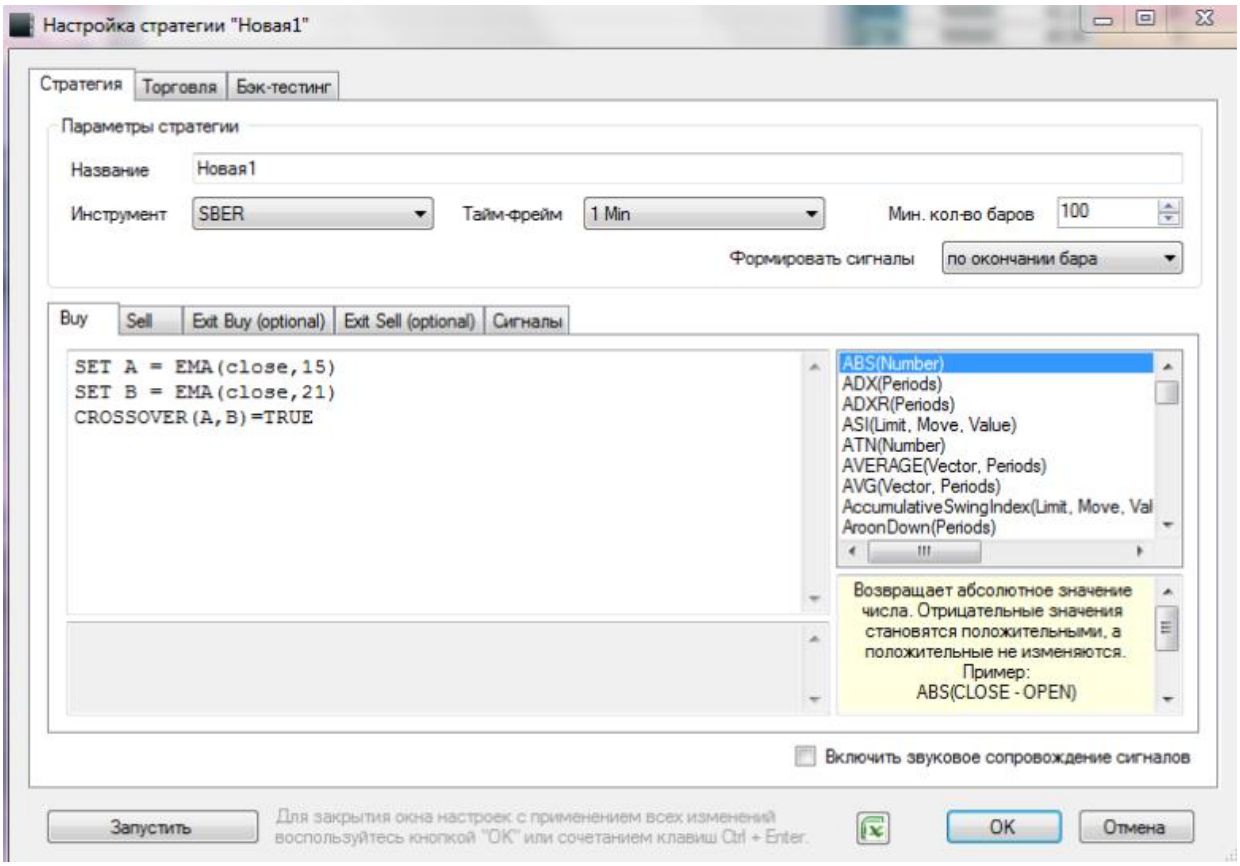

Рисунок 4 – Пример среды разработки TradeScript (конструктор)

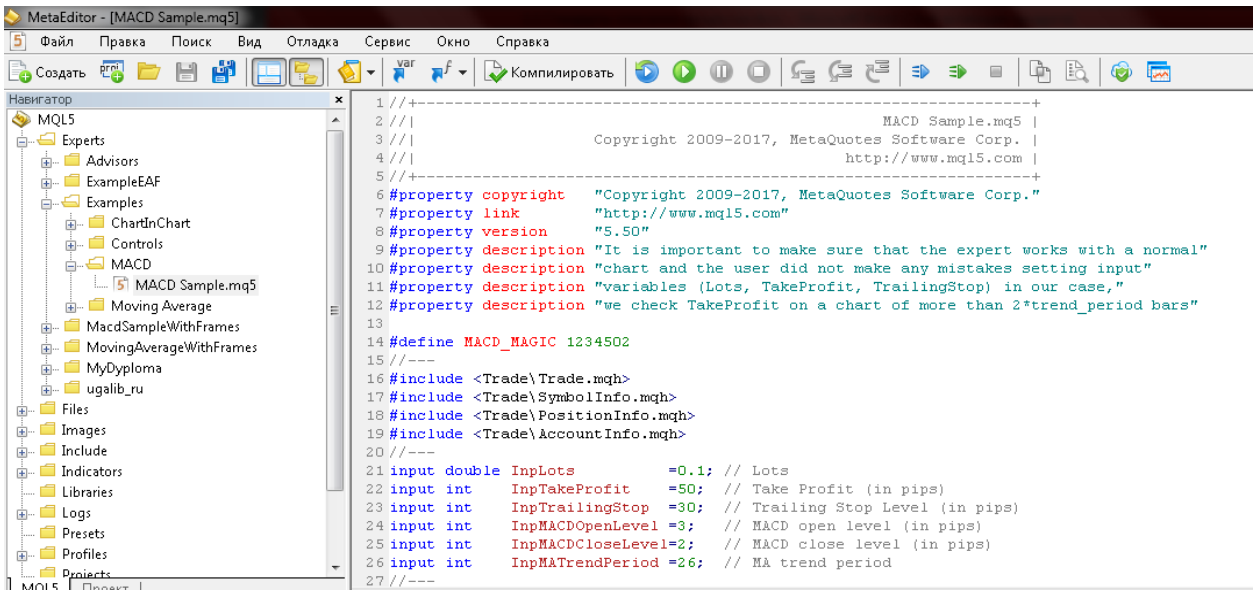

Рисунок 5 – Пример среды разработки MetaQuotes5

Другим популярным подходом для решения задачи автоматизации торговли является полноценное создание программы (торгового робота) для которого необходимо привлекать профессионального программиста, для эффективной работы созданной программы. В качестве примера данного подхода приведём пример среды разработки MetaQuotes5, представленный на рисунке 5.

Из всего перечисленного в данном подразделе можно сделать вывод, что создание робота с помощью конструктора является удобным способом автоматизации торговли для быстрого решения задачи автоматизации торговли, но не позволяет, создавать сложные торговые стратегии. А создание реальной программы позволяет реализовывать более сложные торговые системы.

1.3. Выбор средств и методов решения задачи

На текущий момент существует огромное количество средств и инструментов, обеспечивающих автоматизацию торговли. Ниже мы рассмотрим некоторые из них.

1.3.1 Обзор среды разработки MQL4

MetaTrader4 (MT4) – торговый терминал, разработанный корпорацией MetaQuotes Software Corp. MT4 предназначенная для обслуживания трейдеров на рынках типа FOREX. MT4 представляет собой полный комплекс оборудования необходимого для удалённой торговли вне торгового зала. То есть для организации обслуживания пользователя при наличии у него MT4 нет необходимости в установке дополнительного программного обеспечения. Серверная часть работает только на платформе семейства Windows. Клиентская часть есть в версиях для Windows, Android и iOS.

До конца 2016 года активно использовалась четвёртая версия платформы и выпущена пятая. Первая, вторая и третья версии платформы не используются и не поддерживаются.

MT4 Client Terminal – клиентская часть торговой системы MT4, устанавливается на компьютере трейдера. (рисунок 6) Предназначена для проведения сделок, торговых операций. Так же MT4 Client Terminal обеспечивает возможность технического анализа в режиме реального времени. Данная платформа обеспечивает поддержку исполнения нескольких типов ордеров, так же, позволяет отдавать распоряжения на проведение операций с определёнными условиями (при достижении определённой цены). Кроме этого MT4 Client Terminal позволяет просматривать текущие новости, которые транслирует серверная часть комплекса. Так же в данной торговой платформе реализован внутренний Си-подобный язык программирования – MQL4, данный язык программирования позволяет запрограммировать торговые стратегии, индикаторы, сигналы.

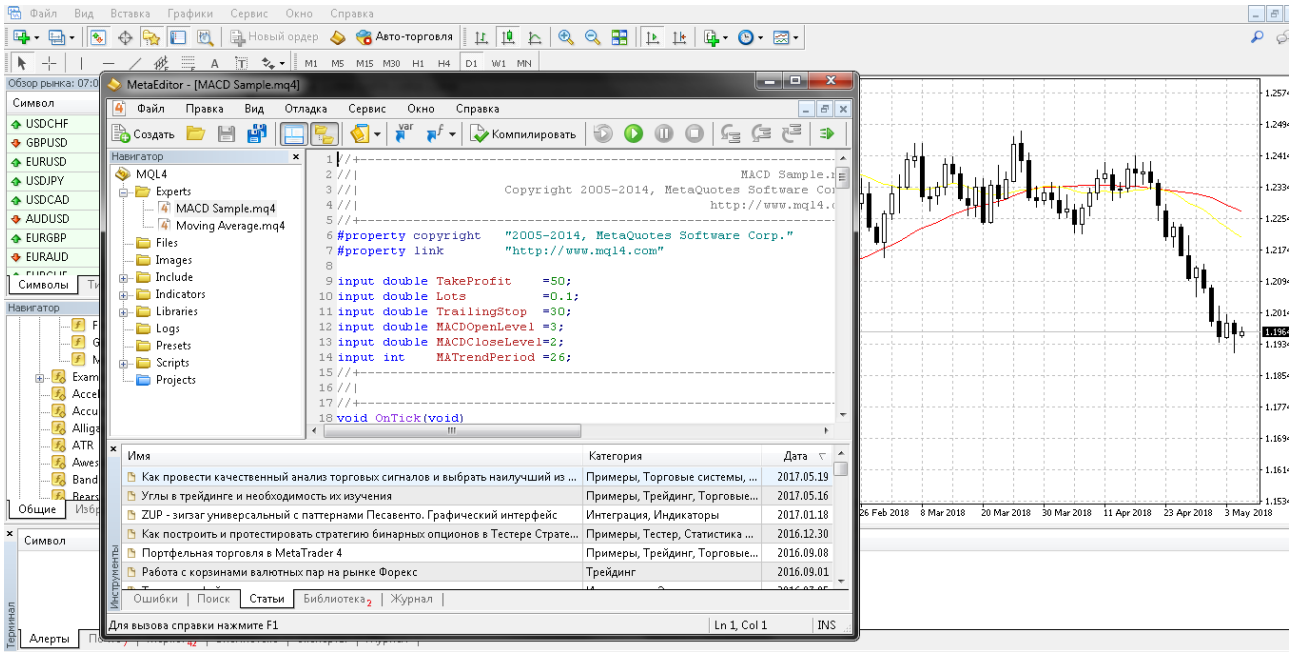

Рисунок 6 – Пример среды разработки MQL4 и торгового терминала MetaTrader4 Client Terminal

Так же имеется возможность обеспечить автоматическую торговлю, когда программа-советник не только выводит рекомендации по торговле в виде изображений и сигналов, но и кроме этого имеет возможность посылать команды на открытие/закрытие сделок. Клиентский терминал может работать абсолютно с любым торговым сервером MetaTrader 4 Server без привязки к определённым предустановленным адресам. Для подключения терминала к серверу необходимо: ввести адрес сервера и выбрать тип счёта, а затем ввести логин и пароль. MT4 Client Terminal поддерживается только на Платформе Windows. Лицензирование для клиентского терминала. MT4 Client Terminal не требуется.

Платформа ориентирована на маржинальную торговлю. Возможна торговля CFD. Но платформа не предназначена для полномасштабной работы на фондовом рынке или для проведения операций без маржинальных условий:

– нет возможности выставить собственную заявку в рынок, чтобы её увидели другие торговцы;

– нет возможности посмотреть список существующих на рынке заявок;

– нет механизмов работы с опционами;

– нет возможности подключить дополнительный источник котировок и новостей;

– нет механизмов работы в национальной валюте (отчёт на клиентском терминале всегда формируется на английском языке с указанием USD в качестве валюты) [18].

1.3.2 Обзор среды разработки QUIK

QUIK (от англ. Quickly Updatable Information Kit – Быстро-Обновляемая Информационная Панель) – это программный комплекс для организации доступа к биржевым торгам (рисунок 7). Совершение сделок с ценными бумагами через интернет называется интернет-трейдингом. Состоит QUIK из серверной части и рабочих мест (терминалов) клиента, взаимодействующих между собой через интернет

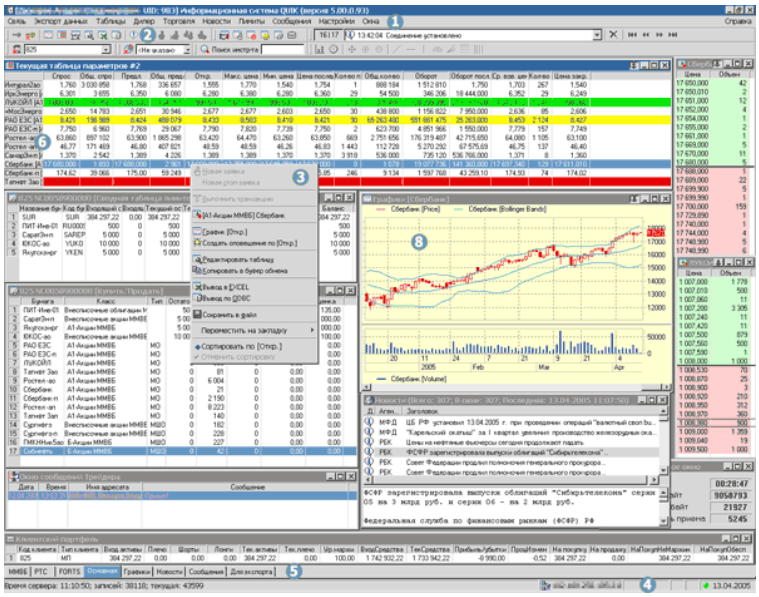

Рисунок 7 – Пример среды QUIK

Изначально QUIK являлся информационной системой, отличавшейся высокой скоростью доставки данных, что и отразилось в названии программы. Сейчас QUIK представляет собой многофункциональную фронт-офисную систему прямого доступа к торгам для организации собственных операций, а также брокерского обслуживания клиентов на финансовых рынках (интернет-трейдинг). QUIK сегодня – это наиболее популярная торговая платформа в России и на Украине для доступа на национальные биржи. QUIK применяется более чем 270 финансовыми организациями для обслуживания десятков тысяч клиентов.

1.3.3 Обзор среды разработки MQL5

В марте 2008 года корпорация MetaQuotes Software Corp. объявила о планировании разработки новой версии платформы MetaTrader – MetaTrader5. Бета-версия вышла 2 сентября 2009, а 1 июня 2010 вышел первая полноценная версия платформы MetaTrader5. Пример интерфейса MetaTrader5, можно увидеть на рисунке 8

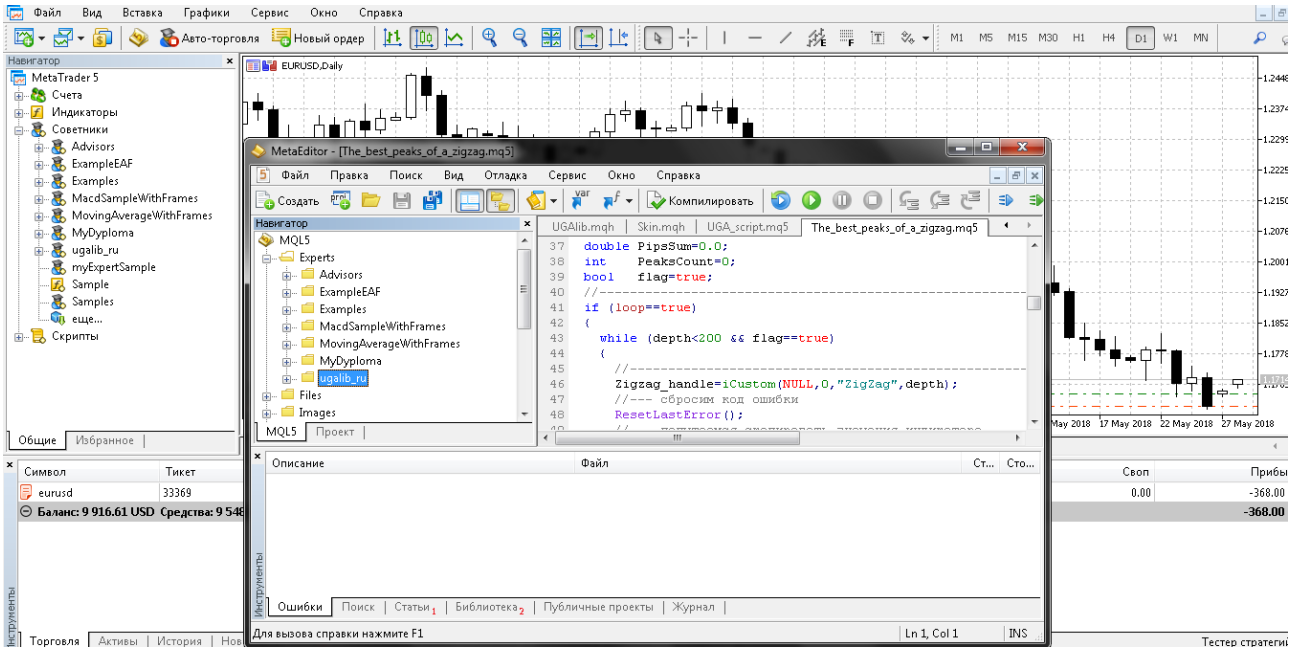

Рисунок 8 – Пример среды разработки MQL5 и торгового терминала MetaTrader5 Client Terminal с демо-счётом

По сравнению с MetaTrader4 Client Terminal, MetaTrader5 Client Terminal имеет целый ряд значительных преимуществ:

– большее количество таймфреймов с в MetaTrader4 их было 9, сейчас их 21;

– кроме перекрытия позиций (возможность открывать сделки одновременно в двух направлениях на покупку и продажу одной и той же валютной пары или другого актива, без их взаимного закрытия) добавлена возможность неттинга (в случае открытия позиций в разных, противоположных направлениях, они обе перекрывается и остаётся только одна результирующая позиция, объём которой больше или происходит взаимное закрытие);

– поддержка котировок уровня L2, так называемого стакана котировок;

– так же добавлен тестер стратегий который обеспечивает возможность тестирования торговых стратегии на реальных массивах котировок, ранее в платформе MetaTrader4 Client Terminal была возможность тестирования лишь на частично смоделированных.

Таким образом можно прийти к заключению что MetaTrader5 Client Terminal является одним из наиболее современных, эффективных и простых

(интуитивно понятных) сред для удалённой торговли, кроме того обеспечивающих простую разработку торговых роботов [19].

1.3.4 Обзор среды TradeScript (SmartX)

В торговом терминале компании ITinvest под названием SmartX есть специальный плагин с конструктором торговых роботов TradeScript (рисунок 4).

С помощью простого, но довольно мощного скриптового языка трейдеры могут создавать механические системы различного уровня сложности.

Существует также модуль бэктестинга, позволяющий оценить продуктивность работы запрограммированной стратегии на исторических данных. Кроме того, предоставлена и возможность тестирования торговой системы «на лету» с использованием текущих биржевых данных, но без вывода приказа на биржу – время виртуальной сделки, цена и получившаяся доходность будут показываться в отдельном окне.

1.4 Постановка задачи

На текущий момент существует огромное количество средств технического анализа рынка. Начиная с первого технического индикатора – осциллятора, созданного ещё в конце 1980-х годов и до текущего момента, было создано огромное количество инструментов технического анализа рынка. Тем не менее процент игроков, которые получали регулярную прибыль со спекулятивной торговли неуклонно снижался, если в 1989 году этот процент составлял 20-25%, то на текущий момент он составляет 2-5%.

Это связано с влиянием информационных технологий на скорость производственных процессов, переговорных процессов и т. д. Таким образом скорость бизнес-процессов неуклонно росла, как следствие и росла скорость изменения тенденций на рынке.

Таким образом задачей данной работы является создание торгового робота способного автоматически подстраиваться под изменяющиеся тенденции рынка.

Вывод по разделу 1

В ходе подготовки данного раздела были выполнены следующие задачи: разработаны DFD-Диаграмма, произведён обзор существующих подходов к решению задачи, произведён выбор средств и методов для решения данной задачи и выполнена постановка задачи.

При помощи анализа с использованием DFD-Диаграммы были описаны разные режимы работы робота автоматический и полуавтоматический. Не смотря на экономию времени при использовании автоматического режима работы существуют риски неправильного вхождения в рынок, поэтому необходимо использовать полуавтоматический режим работы в котором окончательное решение стоит за человеком, а не машиной.

В ходе анализа имеющихся торговых терминалов было выполнено сравнение таких терминалов как MetaTrader4 Client Terminal, QUICK, MetaTrader5 Client Terminal, TradeScript. По таким параметрам как наличие документации, удобство пользования, наличие инструментов технического анализа было принято решение о использовании терминала MetaTrader5 Client Terminal и использовании среды разработки MQL5 (Meta Quotes Language) поставляющейся вместе с данным терминалом.

2 СИСТЕМНЫЙ АНАЛИЗ ИНФОРМАЦИОННОЙ СИСТЕМЫ АТОМАТИЗАЦИИ ТОРГОВЛИ

В данном разделе будет описан анализ информационной системы торгового робота. Будет описана архитектура торгового робота, разработаны критерии эффективности и принципы верификации и тестирования ИС.

2.1 Архитектура ИС

Архитектура данной ИС представляет собой следующую структуру: терминал генерирует события в зависимости от текущего состояния рынка и от настроек робота, события могут быть разными ниже будет приведены основные виды событий.

2.1.1 Виды событий

Пользовательский терминал позволяет генерировать следующие виды событий:

– Init;

– Deinit;

– NewTick;

– Timer;

– Trade;

– Tester;

– TesterInit;

– TesterPass;

– TesterDeinit;

– ChartEvent и др.

Более подробное описание событий можно найти в официальной документации, ниже будут приводнены основные события и их краткое описание.

Обязательными событиями, которые необходимо обработать являются такие события как Init и Deinit, их необходимо обрабатывать для правильной инициализации и деинициализации приложения. Остальные события могут игнорироваться.

Функция OnInit() является обработчиком события Init. Имеет тип void или int, параметров отсутствуют. Событие Init генерируется сразу после загрузки эксперта. Функция OnInit() используется для обработки инициализации. Если OnInit() имеет возвращает значение типа int, то если возвращаемое значение равно нулю это означает неудачную инициализацию, в следствии чего генерируется событие Deinit.

Функция OnDeinit() обрабатывает событие Deinit и вызывается при деинициализации. Объявляется с типом void и имет один параметр типа int, содержащий код причины деинициализации.

Событие NewTick генерируется для экспертов в случае поступления нового тика по символу, к графику которого прикреплен эксперт. Функцию OnTick().

Функция OnTimer() вызывается для обработки события Timer. Периодичность наступления этого события устанавливается при помощи функции EventSetTimer() реагирующей на уведомления о событии Timer.

Функция OnTrade() вызывается для обработки события Trade, которое возникает когда происходит изменение списка открытых сделок, отложенных ордеров, истории ордеров и истории сделок. Таким образом можно сказать что данное событие генерируется при любом действии пользователя связанным с торговлей (открытии/закрытии позиции, выставлении отложенного ордера, установке StopLoss и TakeProfit, срабатывании отложенных ордеров и т.п.) так же изменяется и история ордеров и сделок или список позиций и текущих ордеров.

Функция OnTester() является обработчиком события Tester, которое автоматически генерируется по окончании исторического тестирования эксперта на заданном интервале дат. Функция должна быть определена с типом double, параметров не имеет:

Функция OnTesterInit() обрабатывает событие TesterInit, которое генерируется перед началом оптимизации торгового робота в тестере. Функция определяется с типом void, параметров не имеет. Торговый робот, имеющий обработчик OnTesterDeinit() или OnTesterPass(), при запуске в режиме тестирования на отдельном графике терминала автоматически загружается с указанными символом и периодом, и получает событие TesterInit. Эта функция предназначена для запуска эксперта перед началом оптимизации.

Функция OnTesterPass() обрабатывает событие TesterPass, которое автоматически генерируется при поступлении фрейма во время оптимизации эксперта в тестере стратегий. Функция должна быть определена с типом void, параметров не имеет.

Робот с обработчиком OnTesterPass() загружается на отдельном графике с указанными символом/периодом и получает при оптимизации генерацию ряда событий типа TesterPass, срабатывающих при получении следующего фрейма. Данная функция предназначена для динамической обработки результатов оптимизации, данная функция предназначена для отображения результатов оптимизации, не дожидаясь её окончания. Для добавления фреймов применяется функция FrameAdd(), которую вызвать по окончании прохода в обработчике OnTester().

OnTesterDeinit. Функция OnTesterDeinit() обрабатывает события TesterDeinit, которое генерируется при окончании оптимизации торгового робота в тестере. Функция определяется с типом void, параметров данная функция не имеет. Торговый робот, обрабатывающий событие TesterDeinit() загружается на график, при запуске оптимизации и получает событие TesterDeinit после её завершения. Функция предназначена для конечной обработки результатов оптимизации.

Функция OnBookEvent() обрабатывает события BookEvent. Событие BookEvent генерируется для торговых роботов при изменении состояния стакана цен. Данная функция имеет тип void и один параметр типа string. Чтобы иметь возможность обрабатывать события BookEvent по любому символу, достаточно предварительно уведомить терминал о необходимости получения данных событий для данного символа с помощью функции MarketBookAdd().

OnChartEvent() является обработчиком группы событий ChartEvent: эти события отвечают за обработку событий связанных с графиком, например: нажатие кнопки на клавиатуре, перемещение мыши и т.д.

Затем для эффективного функционирования торгового робота создаётся класс методы которого позволяют совершать обработку данных событий. Методы данного класса позволяют реализовать логику данного класса.

2.2 Критерии оценки эффективности

В этом подразделе будут описаны критерии эффективности торгового робота.

Надежность системы – это свойство системы сохранять в определённый промежуток времени все свои параметры в определённых пределах значений. Так же надёжность позволяет характеризовать способность системы выполнять определённые требованиями функции в заданных режимах работы и условиях.

Надежность информационных систем не является самоцелью, а лишь является средство обеспечивающим своевременный и достоверный вывод правильной информации на ее выходе. Таким образом показатель надёжности функционирования имеет для информационных систем главенствующее значение.

Достоверность функционирования информационной системы – свойство системы, которое обусловливает отсутствие ошибок при произведении ею различных преобразований полученной информации.

Эффективность системы – это свойство системы, заключающееся в качественном выполнении поставленной цели в определённых условиях использования. Показатели эффективности системы позволяют характеризовать приспособленность системы для выполнения поставленных задач и позволяют оценить оптимальность функционирования информационной системы, которая зависит от локальных показателей.

За размещение и отправку ордеров отвечает терминал таким образом можно сказать что за надёжность и достоверность отвечают разработчики терминала MetaTrader5. Задачей разработчика является разработка робота, позволяющего производить автоматизацию торговли.

Ниже будут описаны критерии эффективности системы, которые мы можем применить:

– баланс;

– коэффициент прибыли (profit factor);

– ожидаемый выигрыш (expected payoff);

– общая прибыль (gross profit);

– общий убыток (gross loss);

– просадка (drawdown);

– фактор восстановления (recovery factor).

Баланс – показатель характеризующий количество средств, находящихся на счету у пользователя с учётом открытых и закрытых сделок

Общая прибыль (gross profit) – общая сумма прибыли от всех положительных сделок без учёта затрат на торговлю (с вычетом комиссии)

Общий убыток (gross loss) – общая сумма убытков от всех положительных сделок без учёта затрат на торговлю (с вычетом комиссии)

Коэффициент прибыли (profit factor) – характеристика результата торговли (торгового робота или человека) показывающая отношение общей прибыли от всех положительных сделок к общему убытку от всех отрицательных сделок за определённый промежуток времени, рассчитывается по формуле (2.1):

$$PF = \frac{GP}{GL}, \tag{2.1}$$

где PF – коэффициент прибыли (profit factor)

GP– общая прибыль (gross profit)

GL– общий убыток (gross loss)

Ожидаемый выигрыш (expected payoff) – так же называется математическое ожидание выигрыша, характеристика результата торговли (торгового робота или человека), показывает отношение между суммарным выигрышем и суммарными потерями, рассчитывается по формуле (2.2):

$$EP = \frac{PT}{TT} \cdot \frac{GP}{PT} - \frac{LT}{TT} \cdot \frac{GL}{LT}, \tag{2.2}$$

где EP– ожидаемый выигрыш (expected payoff);

TT – общее количество всех сделок (total trades);

PT – общее количество положительных сделок (profit trades);

LT – общее количество отрицательных сделок (lose trades);

GL – сумма всех убыточных сделок (gross lose);

GP – сумма всех прибыльных сделок (gross profit);

Просадка (drawdown) – характеристика результата торговли, в данном контексте подразумевается максимальная просадка, под этим термином подразумевается разница между максимальным значением баланса и следующем за ним наименьшим значением баланса, за определённый промежуток времени

Фактор восстановления (recovery factor) – характеристика результата торговли, показывающая насколько быстро система восстанавливаются после просадки. Так же фактор восстановления показывает на сколько величина прибыли превышает величину просадки.

2.3 Описание математических средств технического анализа и дальнейшее проектирование торгового робота на основе математического анализа

Любая торговая система имеет ряд параметров, во многих случаях общее количество комбинаций этих параметров крайне велико, ниже будут рассмотрены некоторые торговые стратегии, и их составляющие. Параметрами могут являться такие величины как:

– период графика;

– количество пунктов полученной прибыли, после которого стоит закрывать сделку (выигрышная сделка);

– количество пунктов полученных убытков, после которых стоит закрывать сделку (проигрышная сделка);

2.3.1 Описание средств технического анализа основанных на дополнительных построениях

Существует огромное количество подходов к техническому анализу графиков, но самым распространённым является скользящее среднее (Moving Averange).

Скользящее среднее способствует сглаживанию ряда, целью сглаживания ряда есть отсеивание колебаний уровней этого ряда и выявление максимально устойчивого направления движения.

Любое скользящее среднее – это способ идентификации среднего уровня ряда за некоторый отрезок времени. Термин "скользящее" подразумевает, что среднее значение вычисляется с нуля, каждый раз с появлением нового элемента ряда [6,7].

Рассмотрим для начала общий алгоритм скользящего среднего (moving average, MA).

MA вычисляется по формуле (2.3):

$$MA = \sum_{i=1}^{n} w_i x_i \quad , \tag{2.3}$$

где MA – простое скользящее среднее;

x_i– цены за i-ое время;

w_i – вес с которым i-ая цена входит в формулу;

n – количество цен учитываемых при расчёте МА, так же называется периодом МА, при этом необходимо чтобы выполнялось следующее условие (формула 2.4):

$$\sum_{i=1}^{n} w_i = 1 \, , \tag{2.4}$$

Из этого условия видно, что необходимо что бы сумма всех весов была равна 1, так же это известно, как правило нормирования.

Различают несколько типов скользящих средних, а именно:

– простая;

– экспоненциальная;

– взвешенная;

$$SMA = \sum_{i=1}^{n} \frac{1}{n} x_i \, , \tag{2.5}$$

где SMA – значение простой скользящей средней (simple moving average);

x_i – цены за i-ое время;

n – период МА

Простая скользящая средняя является средним арифметическим ряда вычисляется по следующей формуле (2.5)

Примеры простых скользящих средних можно увидеть та рисунке В.1 Приложения В где, красным показано простое скользящее средне с периодом равным 5 дней (среднее значение за неделю), а зелёным с периодом в 21 день (среднее за месяц).

Первым недостатком простой скользящей средней является то что веса, которые относятся к разным дням имеют одинаковое значения, хотя интуитивно понятно, что цена, которую актив имел 20 дней назад менее актуальна и менее влияет на текущее положение чем цена, которую актив имел вчера.

Так же можно рассмотреть рекуррентную формулу (2.6) вычисления простой скользящей средней:

$$SMA_k = SMA_{k-1} + \frac{1}{n} x_k - \frac{1}{n} x_{k-n}, \quad (2.6)$$

где k – номер текущей итерации;

n – период расчёта SMA;

SMA_k – значение простой скользящей средней текущем расчётном периоде;

SMA_{k-1} – значение простой скользящей средней на предыдущем расчетном периоде;

x_k – значение цены актива за новую «добавляемую» часть периода, которая добавилась на текущей итерации

x_{k-n}– значение цены актива за старую «вычитаемую» часть периода, которую мы перестаём учитывать, начиная с текущего этапа, эта то значение, которое мы добавили n итераций назад или 1 период назад

При рассмотрении рекуррентной формулы SMA можно показать, что SMA реагирует на каждое значение два раза, один раз когда это значение входит в SMA, что вполне правильно и необходимо для верного анализа цен, но так же оно реагирует и второй раз когда оно «выходит» из расчётного интервала, но это никак не связано с текущим состоянием рынка, поэтому это может оказывать отрицательное влияние на результаты анализа.

Для более качественного анализа на практике применяют взвешенное скользящее среднее (weighted moving average WMA), как видно из названия для определения значения применяются веса в простейшем случае можно применить линейный ряд т.е. для самого последнего применить минимальные значения веса, а для новых значений применить более высокие значения (таблица 1).

Исследовав таблицу, видно, что сумма весов не будет равна 1, а значит необходимо ввести нормирующий множитель, рассчитаем его значение по формуле (2.7):

$$1/\sum_{i=1}^{n} i = \frac{2}{n(n+1)}\ , \tag{2.7}$$

Таблица 1 – Цена и её вес

Цена	x_k	x_{k-1}		x_{k-n+1}	x_{k-n}
вес цены	1	2	…	n-1	N

Отсюда можно получить следующую формулу (2.8) для вычисления:

$$WMA_k = \frac{2}{n(n+1)} \sum_{k=t-n+1}^{t} (i - t + n) \cdot x_k\ , \tag{2.8}$$

где k – номер текущей итерации;

n – период расчёта WMA;

t – текущий момент времени

x_k – цена в текущий момент времени;

WMA_k – значение взвешенной скользящей средней на текущем расчётном периоде [6,7].

Для демонстрации построим простую среднюю скользящую и взвешенную скользящую среднюю, посмотрим на рисунок В.2 приложения В. На данном

рисунке изображена простая скользящая средняя (красная) и взвешенная скользящая средняя (зелёная), обе эти скользящие имеют период равный 5-ти дням, как видно из рисунка взвешенная скользящая средняя более сильно реагирует на новые входящие значения, таким образом это позволяет раньше и чётче определять образовавшиеся тенденции.

Так же одной из разновидностей, скользящих средних является экспоненциальная скользящая средняя. Но в данной средней скользящей средней веса уменьшаются не линейно, а по экспоненте, таким образом рекуррентная формула (2.9) будет иметь следующий вид:

$$EMA_k = \alpha x_k + (1 - \alpha) \cdot EMA_{k\text{-}1}\ , \qquad (2.9)$$

где EMA_k – значение экспоненциальной скользящей средней на текущем расчётном периоде;

EMA_{k-1} - значение экспоненциальной скользящей средней на предыдущем расчётном периоде;

x_k – цена в текущий момент времени;

α – показательный процент, определяющий степень сглаживания, на данный показатель назначается ограничения: $0 < \alpha \leq 1$;

Про показательный процент α необходимо так же сказать что при $\alpha=1$, значение будет равно текущей цене.

Из формулы EMA_k видно что экспоненциальная скользящая средняя (EMA) в отличии от линейной взвешенной скользящей средней (WMA) и простой (SMA), не обладает недостатком влияния выходящего временного промежутка, поскольку на момент выхода из расчётного промежутка поздние временные интервалы обладают крайне малыми весами.

Как видно из рекуррентной формулы EMA, для данной скользящей средней не определён период, в качестве аналога периода используется показательный процент α, так же можно рассчитать период по следующей формуле (2.10) :

$$T=\frac{2}{\alpha}-1\ , \tag{2.10}$$

где T – период графика;

α – показательный процент;

Так же, на основании рекуррентной формулы, для экспоненциальной средней скользящей можно построить можно построить формулу (2.11) на основе ряда как для линейной взвешенной скользящей средней (WMA) и простой скользящей средней (SMA):

$$\begin{gathered} EMA_k = \alpha x_k + (1-\alpha) EMA_{k-1} = \\ = \alpha x_k + \alpha(1-\alpha) x_{k-1} + \alpha(1-\alpha)^2 EMA_{k-2} = \ldots \\ \ldots = \alpha x_k + \alpha(1-\alpha) x_{k-1} + \ldots + \alpha(1-\alpha)^T EMA_0\ , \end{gathered} \tag{2.11}$$

где EMA_k – значение экспоненциальной скользящей средней в текущий момент;

EMA_{k-1}, EMA_{k-2} … – значение экспоненциальной скользящей средней в прошедшие моменты времени;

EMA_0 – последняя учитываемая цена на промежутке;

T – период графика;

x_k, x_{k-1} – значения цены в соответствующие моменты времени;

α – показательный процент;

Или если упростить полученное значение, то получим следующую формулу (2.12):

$$EMA_k = \alpha \sum_{k=0}^{T-1} (i - t + n)\, x_k\ , \tag{2.12}$$

Изучив график экспоненциальной скользящей средней (рисунок В.3 приложения В) можно прийти к заключению, что экспоненциальная скользящая средняя более качественно выполняет сглаживание цены.

Ещё одним, не менее полезным инструментом анализа являются так называемые фракталы торговой системы Била Вильямса. Торговая система Билла Вильямса, состоит из 5 ступеней, на каждой из ступеней имеется свой инструмент, технического анализа, более подробно данная торговая стратегия будет рассмотрена в пунктах ниже.

Фракталы Билла Вильямса бывают 2 видов: верхние и нижние. Фрактал ставится над данным баром и называется верхним если значение его максимальной цены (high) выше чем у 2 ближайших соседей с права и 2 ближайших соседей слева. Аналогично нижним фрактал называется в том случае, когда его минимальное значение цены (low) будет ниже чем у 2 ближайших соседей с права и 2 ближайших соседей слева.

Для более детальной наглядности рассмотрим приведённый график (рисунок В.4 приложения В), как видно из графика, что при завершении образования фрактала образуется локальный минимум или максимум. Кроме этого можно заметить, что при образовании фрактала, чем раньше совершится повторное достижение значения ближайшего локального экстремума, тем больше вероятность образования более сильной тенденции на дальнейшее движение по тренду, что является сигналом для открытия торговли.

Как видно из приведённого выше описания торговых индикаторов скользящие среднее и фракталы крайне эффективно могут применятся в тех рынках в которых преобладает активная смена возрастающих и спадающих тенденций, но при отсутствии данных тенденций, и при «пассивном» боковом движении данные индикаторы будут иметь низкую эффективность. Инструменты технического анализа, которые при применении в боковом движении будут иметь более высокую эффективность будут рассмотрены ниже.

2.3.2 Описание средств технического анализа основанных на преобразовании графика

Описанные выше индикаторы строились непосредственно на самом графике, кроме них также существуют другие способы анализа. Суть таких методов заключается в построении каких-либо других графиков на основании какого-либо преобразования с уже имеющимся графиком, такие анализаторы имеют специальное название – осцилляторы.

При отсутствии явно выраженной тенденции скользящие средние могут давать ложные сигналы о появлении тенденции. В отличии от скользящих средних осцилляторы не привязаны к графику, а значит позволяют более объективно оценивать образовавшиеся тенденции, то есть могут позволить явно показать ускоряется или замедляется рост, или падение цены.

Одним из наиболее распространённых осцилляторов является индекс относительной силы, изображённый на рисунке В.5 приложения В (Relative strength index, RSI). Судя из названия индекс должен сравнивать показания одного актива относительно другого, но это не совсем так данный индекс показывает непосредственно силу относительно самого актива [7,8].

Данный индекс строится на графике где осью абсцисс является временная ось, а осью ординат является показатель относительной силы цены и принимает значения от 0 до 100, согласно рекомендациям автора данного индекса, Уэллса Уайлдера, значения индекса выше 70 следует интерпретировать как сигнал о восходящей тенденции, а значения ниже 30 – как сигнал о нисходящей тенденции. Что же касается значений из интервала от 30 до 70, то они являются неопределёнными. Для расчёта данного индекса так же используется расчетный период времени. Данный период показывает размер временного диапазона, который используется для расчёта [17].

Для расчёта относительного индекса цен применяется формула (2.13):

$$RSI = 100 - \frac{100}{1 - \frac{U}{D}}, \quad (2.13)$$

где RSI – значение индекса относительной силы для данного актива;

U – среднее движение цены верх за выбранный период;

D – среднее движение цены вниз за выбранный период

Значения U и D рассчитываются как среднее движение цены за выбранный период.

Направление движения цены считается вверх, при условии, если разница между ценами закрытия двух соседних баров положительна при условии, что из более поздней цены вычисляется более ранняя. Затем все положительные движения цены складываются и делятся на значение периода, таким образом мы получим значение U.

Направление движения цены считается вниз, при условии, если разница между ценами закрытия двух соседних баров отрицательна при условии, что из более поздней цены вычисляется более ранняя. Затем все отрицательные движения цены складываются и делятся на значение периода, таким образом мы получим значение D.

Так же при дальнейшем рассмотрении рисунка В.5 приложения В можно заметить, что индекс относительных цен в некоторой мере выравнивает данный график, при этом те места, которые являлись локальными минимумами или максимумами на ценовом графике, так же являются минимумами и максимумами соответственно, и на RSI.

Кроме RSI так же следует упомянуть о таком инструменте технического анализа как Stochastic oscillator. Stochastic oscillator так же как и RSI является способом «выровнять» график и спроецировать его ценовую составляющую на числовые значения от 0 до 100. Различие заключается в том, что для при построении RSI строится одна линия, а при построении Stochastic oscillator, строится 2 линии. Эти линии строятся в 2 этапа в начале определяются начальные массивы значений по следующей формуле (2.14):

$$K_t = \frac{C_t - L_T}{H_T - L_T} \cdot 100\,, \qquad (2.14)$$

где K_t – значение быстрого стохастика в момент времени t , быстрым стохастиком называется первая линия оптируемого торгового инструмента;

C_t – значение цены закрытия бара в момент времени t;

T – временной период из которого будет выбираться минимальное и максимальное значение;

L_T – минимальное значение цены закрытия на выбранном временном интервале;

H_T – минимальное значение цены закрытия на выбранном временном интервале;

Затем к полученному быстрому стохастику применяется метод сглаживания основанный на скользящих средних. Проще говоря строится скользящая средняя с малым периодом, относительно периода T, по значениям, полученным при построении быстрого стохастика. Так будет получен медленный стохастик.

На рисунке В.6 приложения В изображён инструмент технического анализа Stochastic oscillator. Синяя линяя соответствует значениям быстрого стохастика, красная пунктирная – медленного. Так же следует упомянуть о том, что в отличии от RSI где сигнальными уровнями являются уровни 30 и 70, для Stochastic oscillator, эти значения будут равны 20 и 80 [14].

Существует несколько типов сигналов которые данный инструмент технического анализа может подать:

1. Первый тип заключается в том, что, когда один из стохастиков опустится ниже уровня 20, а затем поднимется выше него, следует открывать сделку на покупку. Сигналом на продажу будет иметь место если один из стохастиков сначала преодолеет уровень 80, а затем опустится ниже него.

2. Второй тип сигналов заключается в определении тренда самого стохастика. Если значения быстрого стохастика больше чем значения медленного, то мы имеем сигнал на покупку. Если значения быстрого стохастика меньше чем значения медленного, то мы имеем сигнал на продажу.

3. Третий тип заключается в одновременном анализе графика цен и графика Stochastic oscillator. Если значения локальных максимумов и минимумов графика цен, на протяжении анализируемого интервала времени растут, а соответствующие им значения Stochastic oscillator находятся на одном уровне, то это является следствием образовавшейся восходящей тенденции, что в свою очередь сигнализирует о возможности открытия сделки на покупку. Если значения локальных максимумов и минимумов графика цен, на протяжении анализируемого интервала времени падают, а соответствующие им значения Stochastic oscillator находятся на одном уровне, то это является следствием образовавшейся нисходящей тенденции, что в свою очередь сигнализирует о возможности открытия сделки на продажу [13].

Опираясь на выше изложенную информацию можно сказать что Stochastic oscillator и RSI являются крайне эффективными техническими индикаторами рынка. Но кроме этого следует упомянуть что сами по себе, отдельно друг от друга они могут давать ложные сигналы, поэтому их следует применять вместе друг с другом, либо с другими инструментами технического анализа, для фильтрации полученных сигналов.

В торговой системе Билла Вильямса так же имеются осцилляторы: Awesome Oscillator (AO) и Accelerator Oscillator (AC).

Awesome Oscillator – так же известный как «чудесный» осциллятор, применяется для оценки сил покупателей и продавцов относительно текущего направления тренда.

Билл Вильямс взял для расчета инструмента технического анализа Awesome Oscillator (AO) максимальную (high) и минимальную (low) цену за дневной промежуток.

Максимум показывает на сколько за анализируемый промежуток были сильны покупатели, минимум показывает силу продавцов, но Вильямс взял средину этого периода по формуле (2.15):

$$med=\frac{high}{low}, \tag{2.15}$$

Далее, используя вычисления, были найдены оптимальные периоды простых скользящих средних (SMA), которые был построены по точкам ранее усредненных экстремумов. Эти периоды были 5 и 34. Для получения значений данного осциллятора из быстрой простой скользящей средней автор вычел медленную, полученные результаты были отображены на графике.

Рассмотрим график изображённый на рисунке В.7 приложения В, синим цветом изображена простая скользящая средняя с периодом 34, построенная по усреднённым ценам. Красным цветом изображена простая скользящая средняя с периодом 5, так же построенная по усреднённым ценам. Под ценовым графиком построена гистограмма осциллятора АО.

Значения АО выше 0 означают что простая скользящая средняя с периодом 5, находится выше простой скользящей средняя с периодом 34, что свидетельствует о наличии положительного движения. Значения АО ниже 0 означают что простая скользящая средняя с периодом 5, находится ниже простой скользящей средняя с периодом 34, что свидетельствует о наличии отрицательного движения.

Столбцы гистограммы, окрашенные в красный цвет, свидетельствуют об изменении разницы между простой скользящей средней с периодом 5 и простой скользящей средней с периодом 34 в отрицательную сторону, что может являться признаком нисходящей тенденции. Столбцы гистограммы, окрашенные в зелёный цвет, свидетельствуют об изменении разницы между простой скользящей средней с периодом 5 и простой скользящей средней с периодом 34 в положительную сторону, что может являться признаком восходящей

тенденции. Если на предыдущем временном интервале бар имеет противоположный цвет по сравнению с предыдущим, то это может свидетельствовать об окончании или развороте тренда в соответствующую сторону.

Так же особенностью данного технического индикатора является наличие дополнительных сигналов на увеличение объёма сделки или на открытие ещё одного ордера по направлению тренда. Сигналом для дополнительного ордера на покупку является наличие временного «замедления» восходящей тенденции, когда AO всё ещё больше 0, и имеется последовательность из 3 столбцов гистограммы в которой имеется 1-ый столбец, 2-ой ниже первого и 3-ий выше первого. Такая комбинация свидетельствует о неудачной попытке разворота рынка в сторону падения, и о значительном ускорении движения цены вверх. Сигналом для дополнительного ордера на продажу является наличие временного «замедления» нисходящей тенденции, когда AO всё ещё меньше 0, и имеется последовательность из 3 столбцов гистограммы в которой имеется 1-ый столбец, 2-ой выше первого и 3-ий ниже первого. Такая комбинация свидетельствует о неудачной попытке разворота рынка в сторону роста, и о значительном ускорении движения цены вниз. Таким образом данный индикатор в гораздо меньшей мере подвержен влиянию тех недостатков, которые имеют Stochastic oscillator и RSI и может применятся сам по себе [2].

Так же был Биллом Вильямсом был разработан технический индикатор Accelerator Oscillator (AC) изображённый на рисунке В.8 приложения В, который так же является осциллятором. Accelerator Oscillator отображает как сильно нарастает (ускоряется) или спадает (замедляется) движущая сила торгуемого актива.

При разработке Accelerator Oscillator, в его основу был положен принцип увеличения (ускорение) или снижения (замедление) скорости движения цены. При развороте ценового движения его скорость замедлятся, а при развитии рыночного движения растет и ускорение цены. Заметив данные особенности при исследовании ценовых рядов, можно заметить, что для того что бы наглядно

продемонстрировать изменение скорости движения цены, можно применять индикатор Accelerator Oscillator (AC) для распознавания этого изменения, с целью использования таких особенностей для определения точек открытия и закрытия торговых позиций.

Осциллятор Accelerator Oscillator рассчитывается по приведённой ниже формуле:

$$AC = AO - SMA_{AO}, \tag{2.16}$$

где AC – значение осциллятора Accelerator Oscillator;

AO – значение осциллятора Awesome Oscillator;

SMA_{AO} – значение простой скользящей средней построенной по значениям Awesome Oscillator;

Применение осциллятора Accelerator Oscillator имеет существенные различия от использования похожих гистограммных индикаторов. Одно из таких отличий – пересечение барами гистограммой нулевой отметки не является сигналом для открытия сделки, а лишь предпосылкой для открытия сделки. Для индикатора Accelerator Oscillator такое пересечение не показывает никаких сигналов для открытия сделок ни на покупку, ни на продажу.

Сигналами для открытия сделок являются определенные последовательности баров индикатора Accelerator Oscillator. Ниже будет рассмотрено два типа сигналов для открытия сигналов.

Первый тип сигналов основывается на пересечении нулевой линии. Сигналом на покупку является последовательность из 2 возрастающих баров, образованных после пересечения нулевой отметки и имеющих положительные значения. Открывать сделку на покупку необходимо при завершении 2-го бара и открытии 3-го бара. Сигналом на продажу является последовательность из 2 спадающих баров, образованных после пересечения нулевой отметки и имеющих отрицательные значения. Открывать сделку на продажу необходимо при завершении 2-го бара и открытии 3-го бара. Данный тип сигналов основан

на распознавании, усиливающихся тенденций, следующий тип основан на распознавании зарождающихся тенденций.

Второй тип сигналов основан изменении движения графика и зарождении новой тенденции. Сигналом для покупки является последовательность из 3 возрастающих баров. Открывать сделку на покупку следует только после завершения сигнальной конструкции, а именно в начале 4-го бара.

Таким образом индикатора Accelerator Oscillator может показывать различные сигналы на покупку и продажу и является полезным инструментом для проведения технического анализа рынка [3].

2.4 Принципы верификации торгового робота

Верификация – процесс который состоит в подтверждении которого получаются подтверждения что заданные спецификацией требования целиком и полностью выполнены. В процессе вариации могут быть обнаружена информация о наличии каких-либо недостатков, которые затем необходимо будет скорректировать.

Для начала описания процесса верификации торгового робота необходимо описать стратегию верификации. Верификация торгового робота предусматривает тестирование робота на исторических данных, в качестве тестовых примеров будут взяты исторические данные котировок за 2017-2018 года. Источником котировок будут служить сервера компании MetaQuotes, разработчика платформы MetaTrader. В качестве источника котировок была выбрана именно данная компания, поскольку она занимается исключительно разработкой ПО и не имеет каких-либо интересов в «изменении» исторических данных, в чем уличали компании предоставляющие дилинговые услуги.

После получения исторических данных будет выполнен запуск симуляции торговли на исторических данных. Показателем качества выполнения будет являться увеличение дохода по сравнению с стандартными параметрами.

Выводы по разделу 2

В ходе подготовки данного раздела были выполнены следующие задачи: описана архитектура торгового робота, основанная на обработке событий, генерируемых торговым терминалом, описаны критерии эффективности, применяемые при разработке торгового робота, описаны основные торговые индикаторы технического анализа.

При выборе критериев оптимизации эффективности наибольшую значение следует уделить такому критерию как баланс счёта, поскольку именно он является наиболее практическим и полезным критерием с точки зрения практической полезности робота. Затем внимание следует уделить фактору восстановления (recovery factor), поскольку он характеризует возможность восстанавливать просадки, что является следствием неправильных сигналов робота.

Так же на основании описанных инструментов технического анализа можно сказать что имеет место огромное место различных способов анализа рынка, каждый из которых имеет ряд преимуществ и ряд недостатков. Поэтому для более эффективного анализа необходимо применять несколько торговых индикаторов: один основной, и несколько «фильтрующих»

3 СТРУКТУРНЫЙ СИНТЕЗ СИСТЕМЫ АВТОМАТИЗАЦИИ ТОРГОВЛИ

Данный раздел посвящён разработке торгового робота, использующего самооптимизирующийся генетический алгоритм. Так же в данном разделе будет приведено описание основных торговых стратегий, выявлению в них параметров необходимых для оптимизации.

3.1 Разработка торгового робота

Выше в первом разделе была описана общая DFD-структура разрабатываемого робота. Во втором разделе была описана общая архитектура приложения и описана система обработки событий терминала торговым роботом. На основании материалов, описанных в предыдущих частях будет выполнена разработка торгового робота

3.1.1 Теоретическое описание торговых стратегий на основе описанных ранее инструментов технического анализа

Ранее были описаны, наиболее часто используемые инструменты технического анализа. На основании этих инструментов строятся торговые стратегии.

Одна из таких систем основывается (рисунок В.9 приложения В) на применении к графику цен трёх известных нам инструментов технического анализа рынка.

Первым инструментом, применяемым для анализа рынка, в рассматриваемой торговой является 2 экспоненциальных скользящих средних с разным периодом. Эти скользящие средние применяются для определения основной направленности рынка, а также, при их перекрещивании проявляется

образование тренда. От этого индикатора в список параметров данной торговой стратегии войдут такие величины как: период первой скользящей средней и период второй скользящей средней [12].

Вторым элементом данной торговой стратегии является осциллятор Stochastic oscillator. Данный индикатор применяется для отсечения опасных сигналов, в тех случаях, когда рынок находится в состоянии перекупленности и перепроданости. В данном случае состоянием перекупленности будем считать те временные интервалы в которых хотя бы одна из линий Stochastic oscillator имеет значение более 80. Состояние перекупленности подразумевает тот факт, что большая часть участников рынка открыли позицию по покупке актива, в следствии чего произошёл сильный рост цены и входить в позицию уже поздно. Состоянием перепроданости будем считать те временные интервалы в которых хотя бы одна из линий Stochastic oscillator имеет значение менее 20. Состояние перепроданости подразумевает тот факт, что большая часть участников рынка открыли позицию по продаже актива, в следствии чего произошёл сильный спад цены и входить в позицию уже поздно. От этого индикатора в список параметров данной торговой стратегии войдут такие величины как: период стохастика, период скользящей средней построенной по Stochastic oscillator, величина значения перепроданости, величина значения перекупленности [15].

Третьим элементом данной торговой системы является стохастический осциллятор RSI. Назначение данного осциллятора в этой торговой стратегии схожее с назначением осциллятора Stochastic oscillator – защита от опасных сделок. Рекомендованный уровень для покупки – выше 50, для продажи – ниже 50. От этого индикатора в список параметров данной торговой стратегии войдут такие величины как: период RSI, рекомендованный уровень для покупки, рекомендованный уровень для продажи [16].

Выше описана торговая стратегия и определены её параметры, основанные на входящих в неё индикаторах. Помимо параметров, основанных на индикаторах, непосредственно существуют параметры самой стратегии. Для данной стратегии этими параметрами будут следующие значения: кол-во

пунктов убытков после которого следует закрыть торговую сделку (Stop Loss), кол-во пунктов прибыли, после которой следует закрыть торговую сделку (Take Profit), кол-во пунктов прибыли, после которой следует перенести (Stop Loss) для данной сделки в зону без убытков (Tralling stop)

В таблице А.1 приложения А перечислены параметры, которые необходимо оптимизировать для данной стратегии, а также, интервалы на которых будет выполнятся поиск оптимальных значений.

Следующей торговой стратегией, которая будет рассмотрена будет стратегия, основанная на индикаторе Ишимоку Кинко Хайо (Ichimoku Kinko Hyo) которая позволяет определить рыночный тренд, уровни поддержки и сопротивления и позволяет производить генерацию сигналов для покупки и продажи. Данный индикатор разрабатывался в 1930-х годах японским аналитиком Гоичи Хосода и более 30 лет совершенствовался автором вплоть аж до 1960-ых годов. Несмотря на то что данная стратегия разрабатывался для анализа японского индекса Nikkei 225 индекса 225 наиболее крупных компаний Японии, стратегия успешно применяется для торговли [9, 10].

С помощью индикатора Ишимоку имеется возможность определить рыночный тренд, так же имеется возможность определения уровней образования тренда или разворота тренда такие уровни имеют название: уровни поддержки и уровни сопротивления. Так же данный индикатор позволяет генерировать сигналы на покупку и продажу актива.

Индикатор Ишимоку Кинко Хайо изображён на рисунке В.10 приложения В, состоит из 5 линий:

1. Tenkan-sen (Красная) показывает среднее значение цены за период времени T1. Tenkan-sen определяется как сумма максимума и минимума за это время, поделенная на два;

2. Kijun-sen (Синяя) показывает среднее значение цены период времени T2;

3. Senkou Span A (Красный пунктир) показывает середину расстояния между предыдущими двумя линиями, сдвинутую вперед на величину T2;

4. Senkou Span B (Синий пунктир) показывает среднее значение цены за период времени T3, кроме того данное значение необходимо сдвинуть вперед на величину T2;

5. Chinkou Span (Зелёная) показывает цену закрытия текущей японской свечи, сдвинутую назад на величину второго временного интервала.

В таблице А.2 приложения А перечислены параметры, которые необходимо оптимизировать для данной стратегии, а также, интервалы на которых будет выполнятся поиск оптимальных значений.

Третьей и последней рассмотренной стратегией будет торговая стратегия Билла Вильямса.

Торговая система Б. Вильямса [2,20] (рисунок В.11 приложения В) состоит из четырёх составляющих:

1. Аллигатор. Аллигатор представляет собой комбинацию 3 средних скользящих:

– синяя линия – SMA с периодом 13, и с сдвигом на 8 баров. Так же называется челюстями аллигатора. Отображает то, где бы находилась цена при отсутствии хаоса. Величина диапазона между челюстями и реальной ценой говорит об интерпретации игроками этой новых входных данных;

– красная линия – SMA с периодом 8, и с сдвигом на 5 баров. Так же называется губами аллигатора. Эта линия отображает равновесие, для меньшего периода. данный период равен пятой части от временного периода графика. То есть, если вы торгуете на дневном графике, линия будет примерно отображать ситуацию на четырехчасовом графике;

– зеленая линия – SMA с периодом 5, и с сдвигом на 3 баров. Так же называется зубами аллигатора. Опять же, зеленая линия соответствует примерно одной пятой временной структуры зубов (красная линия) [4,5].

Аллигатор – более точный индикатор трендового движения чем простая скользящая средняя. Очевидно, чтобы изменить значение синей линии нужно получить больше информации о изменении цены, чем для того, чтобы изменить значение красной или зеленой линии [4,5].

2. Awesome Oscillator (AO) – один из осцилляторов Билла Вильямса

3. Accelerator Oscillator (AC) – ещё один из осцилляторов Билла Вильямса

4. Фракталы Вильямса;

В таблице А.3 приложения А перечислены параметры, которые необходимо оптимизировать для данной стратегии, а также, интервалы на которых будет выполнятся поиск оптимальных значений.

В этом пункте были рассмотрены одни из самых известных торговых стратегий, описаны их параметры, и выбраны интервалы на которых будет проходить оптимизация.

3.1.2 Описание принципов построения генетических алгоритмов

Оптимизационные задачи заключаются в нахождении минимума (максимума) заданной функции. Такую функцию имеет название целевая функция. Как правило, данная функция является сложной функция, зависящая от многих входных параметров. В задаче оптимизации необходимо найти значения входных параметров, при которых целевая функция достигает экстремального (в данной задаче максимального) значения [1, 11].

Существует целый набор методов, позволяющих производить оптимизацию. Эти оптимизационные методы можно разделить на несколько типов:

– методы, использующие понятие производной (градиентные методы) и

– методы стохастические (например, методы группы Монте-Карло).

С помощью их имеется возможность найти экстремальное значение целевой функции. Не всегда применяя данные методы можно гарантировать получение значения глобального экстремума. Когда происходит нахождение локального экстремума вместо нахождения глобального экстремума это называется преждевременной сходимостью. Кроме проблемы преждевременной сходимости имеет место другая проблема – значительные временные затраты при проведении вычислений. Как показывает практика методы гарантирующие более точные результаты вычислений имеют высокое время работы. Для

решения описанных проблем и проводится поиск новых оптимизационных алгоритмов.

Генетические алгоритмы основаны на принципах естественного отбора описанных Ч. Дарвином. Генетические алгоритмы относятся к стохастическим методам. Данные алгоритмы имеют успешное применение в различных областях деятельности. Так же следует упомянуть о Существовании различных модификаций генетических алгоритмов.

Опишем идею применения генетических алгоритмов, допустим у нас есть две торговые стратегии, основанные на таких инструментах технического анализа как Stochastic oscillator и RSI (рисунки В.12 и В.13). Каждая из выбранных стратегий является своего рода решением задачи оптимизации, не факт, что оптимальным.

Основной идеей генетических алгоритмов является организация «естественного отбора» среди различных решений задачи.

Как известно, принцип естественного отбора заключается в том, что в конкурентной борьбе выживает наиболее приспособленный. В нашем случае приспособленность особи определяется доходностью стратегии: чем больше значение целевой функции (прибыль, полученная в результате торговли по торговой стратегии с данными параметрами), тем более приспособленной является особь.

Так как генетические алгоритмы применяют биологические аналогии, то как следствие используемая терминология напоминает биологическую. Таким образом, одно пробное решение, мы будем называть особью или хромосомой, а набор всех пробных решений – популяцией.

Приступим к процессу размножения: попробуем на основе исходной популяции создать новую. Для этого сформируем из исходной популяции брачные пары (в данном случае одну) для скрещивания [11].

Согласно идее генетического алгоритма, некоторые из членов популяции не будут участвовать в процессе размножения, так как образуют пару сами с собой. Какие-то члены популяции примут участие в процессе размножения

неоднократно с различными особями популяции. Процесс размножения (рекомбинация) заключается в обмене участками хромосом между родителями.

Приведём пример скрещивания для наших торговых стратегий, для этого рассмотрим таблицу 2.

Таблица 2 – Пример скрещивания

Номер особи	Stochastic oscillator (Ген №1)	Stochastic oscillator EMA (Ген № 2)	RSI (Ген № 3)
1	14	3	14
2	20	4	5
3	14	3	5
4	20	4	14

Таблица 3 – Пример мутации

Номер особи	Stochastic oscillator (Ген №1)	Stochastic oscillator EMA (Ген № 2)	RSI (Ген № 3)
1	14	3	14
2	20	4	5
3	14	3	5
4	20	4	14
5	50-14=36	3	5
6	20	10-4=6	14

В данной таблице под номерами 1 и 2 находятся родительские особи, а под номерами 3 и 4 их дочерние особи, которые образовались в результате скрещивания 1 и 2 особи. В данной колонии, на начальном этапе имеет место быть всего 2 особи, в реальном поиске решений на начальном этапе при решении задач оптимизации может быть, например, 10, 20, 50 или более особей, данное поколение будет называется первым поколением, начальным поколением или протопопуляцией.

Следующим шагом в работе генетического алгоритма являются мутации, т.е. случайные изменения полученных в результате скрещивания хромосом. Пусть вероятность мутации равна 0,3. Для каждого потомка возьмем случайное число на отрезке [0;1], и если это число меньше 0,3, то изменяем случайно выбранный ген на значение равное максимальному минус значение данного

Пример мутации мы можем видеть на приведённой выше таблице 3, особи №5 и №6. Пятая особь является результатом мутации особи № 3 гена №1. Шестая особь является результатом мутации особи № 4 гена №2.

В результате будет получено новое поколение, которое представлено на таблице 3. Получившуюся популяцию можно будет вновь подвергнуть скрещиванию, мутации и отбору особей в новое поколение. Таким образом, через несколько поколений мы получим популяцию из похожих и наиболее приспособленных особей.

Следует разъяснить что под отбором следует понимать выбор некоторого количества наиболее приспособленных особей.

3.1.3 Описание интерфейсов разработанных классов и функций

В ходе разработки торгового робота был реализован ряд классов и функций. Некоторые из них будут описаны ниже:

```
void geneticAlgo (
double ReplicationPort, //Доля Репликации.
double NMutationPort,   //Доля Естественной мутации.
double AMutatPort, //Доля Искусственной мутации.
double GenoMergPort, //Доля Заимствования генов.
double CrossOverPort,//Доля Кроссинговера.
double NMutProbab//Вероятность мутации каждого гена в %
);
```

Листинг 1 – Основная функция генетического алгоритма

geneticAlgo – основная функция (Листинг 1) запускающая генетический алгоритм, получает основные параметры, которые будут описаны ниже, необходимые для запуска генетического алгоритма, данными параметрами

являются значения в диапазоне от 0 до 1, которые характеризуют вероятность наступления того или иного события

ProtopopulationBuilding – данная функция позволяет создать первичную протопопуляцию, от которой затем будут наследоваться все остальные потомки.

```
void CycleOfOpers(
double &historyHromosomes[][10000],//---
double    ReplicationPort,
double    NMutatPort,
double    AMutatPort,
double    GenoMergPort,
double    CrossOverPort,
double    ReplicationOffset,
double    NMutProbab )
```

Листинг 2 – Функция итерации генетического алгоритма

CycleOfOpers – данная функция (Листинг 2) запускается после генерации протопопуляции, а затем на протяжении каждой итерации, для изменения особей в популяции. Опишем параметры приведённой функции:

&historyHromosomes[] – ссылка на массив особей с хранящимися значениями параметров и функции приспособленности;

double ReplicationPort – параметр принимающей значение от 0 до 1 и характеризующий вероятность того что особь создаст полную копию себя;

double NMutatPort – параметр принимающей значение от 0 до 1 и характеризующий вероятность того что в особи произойдёт естественная мутация, т. е. у нее появится ген случайный ген не присутствующий у её родителей; из произвольного промежутка.

double AMutatPort – параметр принимающей значение от 0 до 1 и характеризующий вероятность того что в особи произойдёт искусственная мутация, т. е. у нее появится ген случайный ген не присутствующий у её родителей; из промежутка максимально удалённого от её родителей.

double GenoMergPort – параметр принимающей значение от 0 до 1 и характеризующий вероятность того что в особи проявится возможность взять гены от разных (более 2) родителей в популяции.

double CrossOverPort – параметр принимающей значение от 0 до 1 и характеризующий вероятность того что в особи проявится возможность скрещивать гены от 2 родителей в популяции.

NMutProbab – параметр принимающей значение от 0 до 1 и характеризующий вероятность того что в особи проявится мутация в произвольном гене.

3.2 Тестирование торгового робота и получение оптимальных параметров для торговых стратегий

Для проведения тестирования был разработан специальный визуализатор тестов (рисунок 9, 10), который позволит наблюдать за оптимизацией, при запуске данной программы отображается 2 графика: верхний и нижний. Верхний график отображает текущее состояние колонии, в данном случае колония состоит из 10 особей. На верхнем графике по оси ординат отображается значение параметра баланс, или попросту количество средств на счету. По оси абсцисс отображается количество сделок. Сами линии отображают результат торговли, красные линии отображают торговлю в убыток (неэффективные Торговые Стратегии), зелёные линии отображают торговлю в прибыль.

Нижний график отображает общую историю всей популяции. На ней по оси абсцисс обозначено общее количество «живших» особей, можно сказать что этот показатель характеризует номер поколения. По оси ординат обозначено сколько каждая особь «заработала» при жизни, другими словами на ней обозначено значение целевой функции или функции приспособленности.

Изучим состояние популяции в начале оптимизации (рисунок 9) и в конце (рисунок 10). Можно заметить, что, в начале оптимизации значения целевой функции особей принимают значения, характеризующиеся низкой прибылью и даже более того не редки случаи, когда результат торговли – убыток.

Рисунок 9 – Визуализация оптимизации (начало) стратегии EMA + Stochastic oscillator + RSI для пары доллар евро

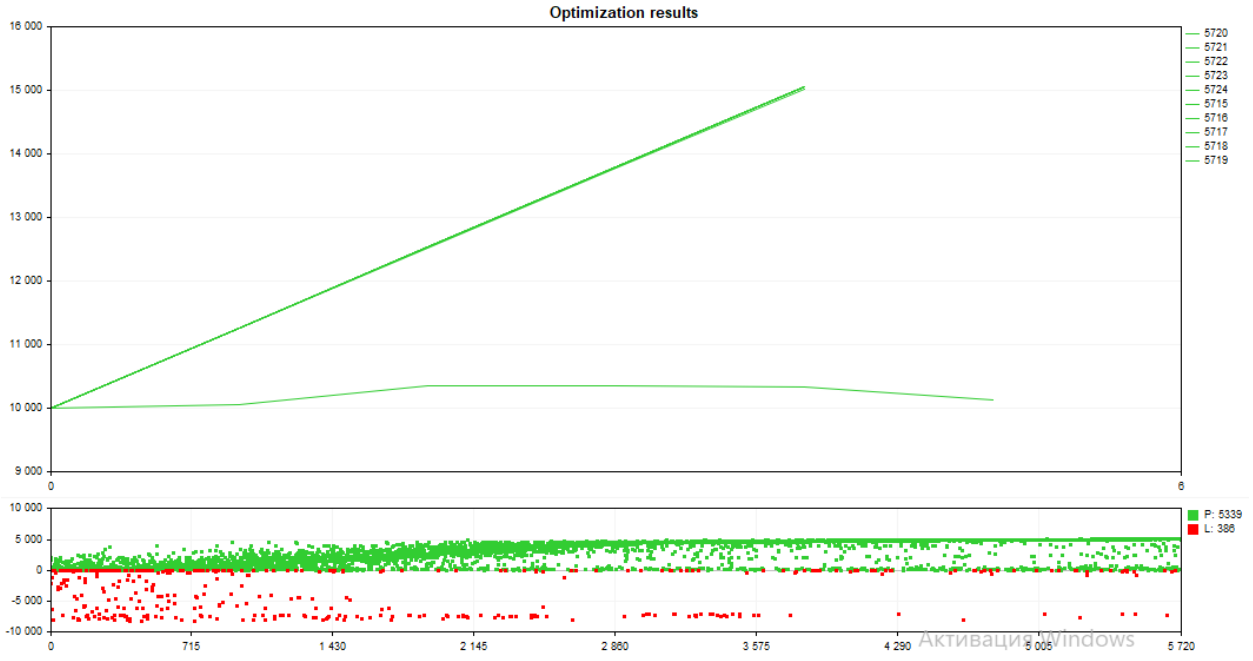

Рисунок 10 – Визуализация оптимизации (конец) стратегии EMA + Stochastic oscillator + RSI для пары доллар евро

Рассмотрев график, изображённый на рисунке 9, для большей наглядности можно вычислить profit factor по формуле (2.17).

$$PF = \frac{75}{22} \approx 3.4\,, \tag{2.17}$$

Проанализировав график изображённый на рисунке 10, описывающий конечное состояние популяции можно заметить, что прибыльность сделок возросла, а количество убыточных сделок уменьшилось, для большей наглядности можно вычислить profit factor по формуле (2.18).

$$PF = \frac{5339}{386} \approx 13{,}8\,\mathbf{,} \tag{2.18}$$

Сравнив profit factor в начале оптимизации и конце оптимизации можно сделать вывод об успешной оптимизации торговой стратегии. Более подробные результаты оптимизации, а именно оптимальные значения для торговых стратегий, по состоянию на май 2018 года, представлены в приложении Б.

Вывод по разделу 3

При выполнении данного раздела были исследованы особенности построения генетических алгоритмов, а также, выполнено тестирование оптимизации генетического алгоритма на следующих валютных парах: евро к доллару США (EUR/USD), доллар США к фунту стерлингов (USD/GBP), доллар США к японской иене (USD/JPY), доллар США к швейцарскому франку (USD/CHF), доллар США к канадскому доллару (USD/CAD), австралийский доллар к доллару США (AUD/USD). В результате проведённого тестирования были получены оптимальные параметры для каждой из 6 валютных пар, для каждой из 3 рассмотренных в данной работе стратегий. С результатами данного тестирования можно ознакомится в приложении Б

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была изучена архитектура, основные принципы построения торговых роботов, потоки данных при функционировании торговых роботов.

В ходе выполнения первого раздела работы было разработаны DFD-диаграмма, произведён обзор существующих подходов к решению задачи, выполнена постановка задачи и выбраны средства и методы для решения поставленной задачи. При помощи анализа с использованием DFD-диаграммы были описаны разные режимы работы робота. В ходе анализа имеющихся торговых терминалов было выполнено сравнение основных наиболее часто применяемых торговых терминалов. В ходе анализа было принято решение о использовании терминала MetaTrader5 Client Terminal и использовании среды разработки MQL5 (Meta Quotes Language) поставляющейся вместе с данным терминалом.

При выполнении второго раздела были решены следующие задачи: описана архитектура торгового робота, описаны критерии эффективности, применяемые при разработке торгового робота, описаны основные торговые индикаторы технического анализа. Так же был произведён выбор критериев оптимизации эффективности. Такими критериями били выбраны баланс счёта и фактор восстановления (recovery factor).

В третьем разделе на основании описанных инструментов технического анализа были описаны наиболее популярные торговые стратегии. Так же при выполнении данного раздела были исследованы особенности построения генетических алгоритмов, а также, выполнено тестирование оптимизации генетического алгоритма на основных валютных парах.

Результатом проделанной работы имеется готовый торговый робот реализующий автоматическую подстройку под текущие условия рынка.

Дальнейшее улучшение данного торгового робота возможна в двух направлениях:

1.Разработка дополнительных модулей для других торговых стратегий. Описанные в данной работе стратегии не являются единственными и уникальными, а являются лишь малой долей в огромном многообразии различных торговых стратегий, основанных на технических индикаторах. Каждую из этих стратегий можно математически формализовать, разработать алгоритм и интегрировать в данный торговый робот.

2. Доработка генетического алгоритма. Генетический алгоритм – это не строго определённый один какой-либо алгоритм. Генетические алгоритмы – это целое семейство различных алгоритмов, имеющих ряд характерных особенностей. Такие алгоритмы разрабатываются и модернизируются и по сей день. Поэтому и рассмотренный в работе генетический алгоритм можно улучшать и дорабатывать под свои нужды.

Таким образом, в результате выполнения работы был разработан готовый программный продукт, позволяющий автоматизировать обнаружение торговых сигналов, необходимых для совершения прибыльных сделок.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бондарев В.Н. Искусственный интеллект: Учеб. пособие для вузов/ В.Н. Бондарев, Ф.Г. Аде.- Севастополь: Изд-во СевНТУ, 2002. – 615 с.: ил.

2. Вильямс Билл. Новые Измерения в Биржевой Торговле, 2015 – 156 с.: ил.

3. Вильямс Билл. Торговый хаос: Увеличение прибыли методами технического анализа / Джастин Грегори-Вильямс и Билл Вильямс ; Пер. с англ. – 4-е изд. – М. : Альпина Паблишер, 2015. – 310 с.

4. Вильямс Билл. Торговый хаос 2. Джастин Грегори Вильямс и Билл Вильямс - М.: ИК Аналитика, 2005. – 237 с.

5. Вильямс Билл. Хозяева рынков Вильямс - М.: ИК Аналитика, 2010. – 181 с.

6. Булашев С.В. Статистика для трейдеров. -М.: Компания Спутник, 2003. – 245 с.

7. Джон Дж. Мэрфи Технический анализ фьючерсных рынков: теория и практика. – Пер. с английского: Новицкая О., Сидоров В. Науч. ред. к. э. н. Самотаев И. М.: Сокол, 1996. – 592 с.

8. Ларри Вильямс. Долгосрочные секреты краткосрочной торговли. –М.: ИК Аналитика. 2001. – 312 с.

9. Лукас Д.В Компьютерный анализ фьючерсных рынков: Пер. с англ. - М.: Издательский Дом "АЛЬПИНА", 1998. – 304 с.

10. Нисон Стив. Японские свечи: графический анализ финансовых рынков. Перевод с англ. Дозорова Т., Волкова М. М.: Издательство «Диаграмма», 1998. – 336 с.

11. Панченко, Т. В. Генетические алгоритмы [Текст]: учебно-методическое пособие / под ред. Ю.Ю. Тарасевича. – Астрахань: Издательский дом «Астраханский университет», 2007. – 87 с.

12. Смит К. Как стабильно зарабатывать на рынке FOREX / Кортни Смит; Пер. с англ. – М.: Альпина Паблишер, 2012. – 224 с.

13. Стивен Б. Алексис. Технический анализ от А до Я. / Стивен Б. Алексис – Диаграмма, 1999 г. – 278 с.

14. Швагер Джек. Технический анализ. Полный курс. – М.: Альпина Паблишер, 2001. – 768 с.

15. Эдвин Лефевр : Воспоминания биржевого спекулянта. – М.: Альпина Паблишер, 2012. – 103 с.

16. Элдер А. Э45 Как играть и выигрывать на бирже: Психология. Технический анализ. Контроль над капиталом / Александр Элдер. – 4-е изд., перераб. и доп. – М.: Альпина Бизнес Букс, 2007. – 472 с.

17. J.Welles Wilder Jr. New concepts in technical trading systems. – Hunter Publishing Company, North Carolina 1978 . – 129 pp.

18. MQL 4 трейдинг: сайт проекта MQL 4. – 2005. [Электронный ресурс]. Дата обновления: 10.05.2018 URL: https://www.mql4.com. (дата обращения: 15.05.2018)

19. MQL 5 трейдинг: сайт проекта MQL 5. – 2010. [Электронный ресурс]. Дата обновления: 15.05.2018 URL: https://www.mql5.com. (дата обращения: 15.05.2018)

20. Profitunity Trading Group : сайт проекта MQL 5. – 2010. [Электронный ресурс]. Дата обновления: 20.05.2018 URL: http://www.profitunity.com. (дата обращения: 20.05.2018)

ПРИЛОЖЕНИЕ А

Таблицы значений параметров торговых стратегий

Таблица А.1 – Список параметров и оптимизационных интервалов для первой торговой стратегии

Название параметра	Значение по умолчанию	Минимальное значение	Максимальное значение
Take Profit	50	20	500
Stop Loss	30	5	100
Tralling stop	30	5	500
Период первой скользящей средней (EMA)	10	5	100
Период второй скользящей средней (EMA)	5	2	50
Период стохастика Stochastic oscillator	14	5	34
Период MA построенной по Stochastic oscillator	3	2	10
Значение перепроданости по Stochastic oscillator	80	50	90
Значение перекуплености по Stochastic oscillator	20	10	50
Сигнальное значение RSI на покупку	50	50	90
Сигнальное значение RSI на продажу	50	10	50

Таблица А.2 – Список параметров и оптимизационных интервалов для первой торговой стратегии основанной на индикаторе Ишимоку Кинко Хайо

Название параметра	Значение по умолчанию	Минимальное значение	Максимальное значение
Период Tenkan-sen	52	20	90
Период Kijun-sen	26	10	50
Период Senkou Span A	52	20	90
Сдвиг Senkou Span A	26	10	50
Период Senkou Span B	26	10	50
Сдвиг Senkou Span B	26	10	50
Период Chinkou Span	9	5	25
Take Profit	50	20	500
Stop Loss	30	5	100
Tralling stop	30	5	500

Таблица А.3 – Список параметров и оптимизационных интервалов для первой торговой стратегии Билла Вильямса

Название параметра	Значение по умолчанию	Минимальное значение	Максимальное значение
Период челюсти аллигатора	13	8	80
Сдвиг челюсти аллигатора	8	5	50
Период зубов аллигатора	8	5	50
Сдвиг зубов аллигатора	5	3	30
Период губ аллигатора	5	3	30
Сдвиг губ аллигатора	3	2	20
Период быстрой МА для АО	5	2	50
Период медленной МА для АО	34	10	200
Период медленной МА для АС	5	2	20
Take Profit	50	20	500
Stop Loss	30	5	100
Tralling stop	30	5	500

ПРИЛОЖЕНИЕ Б

Результаты оптимизации торговых параметров (на 1 мая 2018 г.)

Таблица Б.1 – Результаты оптимизации для торговой стратегии EMA + Stochastic Oscillator + RSI

Название параметра	USD /CHF	AUD/ USD	USD/ CAD	USD /JPY	USD/ GBP	EUR/ USD
Take Profit	44	45	77	84	72	53
Stop Loss	19	21	30	41	39	32
Tralling stop	10	10	15	20	24	25
Период первой скользящей средней (EMA)	9	14	2	11	17	15
Период второй скользящей средней (EMA)	4	8	5	5	6	7
Период стохастика Stochastic	11	17	14	15	17	20
Период MA построенной по Stochastic	3	4	3	5	3	5
Значение перепроданости по Stochastic	36	31	35	37	27	23
Значение перекуплености по Stochastic	64	69	65	63	73	77
Сигнальное значение RSI на покупку	53	61	62	55	65	60
Сигнальное значение RSI на продажу	47	39	38	45	35	40

Таблица Б.2 – Результаты оптимизации для торговой стратегии Ишоку Кинко Хайо

Название параметра	EUR/ USD	USD/ GBP	USD/ JPY	USD/ CAD	AUD /USD	USD/ CHF
Период Tenkan-sen	50	57	63	45	43	42
Период Kijun-sen	22	29	21	27	22	23
Период Senkou Span A	54	58	60	45	40	41
Сдвиг Senkou Span A	24	29	27	26	25	24
Период Senkou Span B	23	26	25	28	29	22
Сдвиг Senkou Span B	24	25	28	20	21	22
Период Chinkou Span	10	11	19	17	12	13
Take Profit	45	77	72	77	45	44
Stop Loss	19	21	30	39	30	21
Tralling stop	10	10	15	24	15	10

Таблица Б.3 – Результаты оптимизации для торговой стратегии Билла Вильямса

Название параметра	EUR/ USD	USD/ GBP	USD/ JPY	USD/ CAD	AUD /USD	USD/ CHF
Период челюсти аллигатора	13	14	13	15	13	14
Сдвиг челюсти аллигатора	8	8	9	9	8	8
Период зубов аллигатора	8	8	9	9	8	8
Сдвиг зубов аллигатора	5	6	7	6	5	7
Период губ аллигатора	5	6	7	6	5	7
Сдвиг губ аллигатора	3	3	5	3	3	4
Период быстрой МА для АО	5	3	3	4	6	7
Период медленной МА для АО	34	33	37	34	35	39
Период медленной МА для АС	5	4	3	4	7	8
Take Profit	72	77	45	44	84	53
Stop Loss	39	30	21	19	41	32
Tralling stop	24	15	10	10	20	25

ПРИЛОЖЕНИЕ В

Графики торговых индикаторов

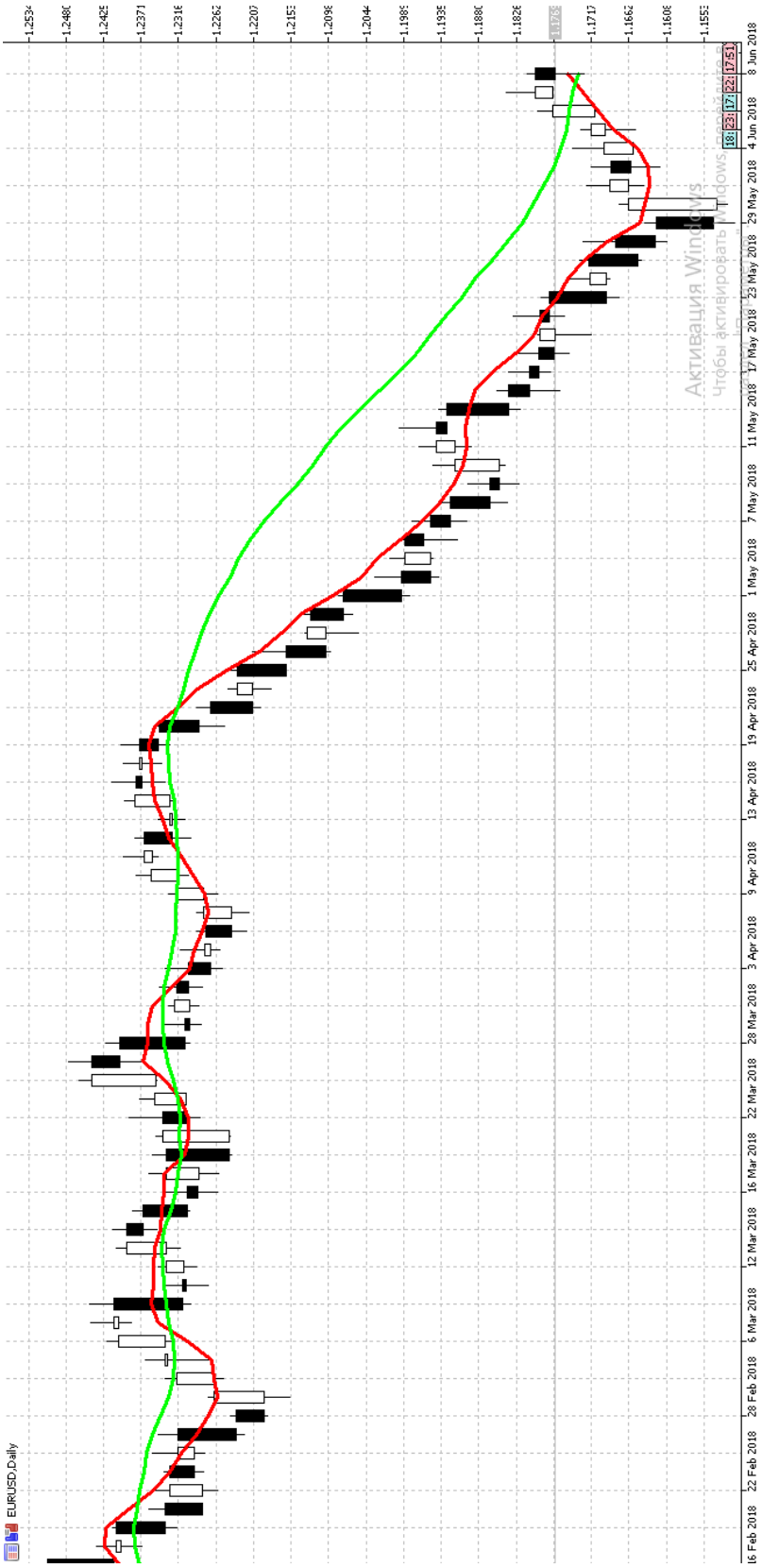

Рисунок В.1 – Простые скользящие средние с разным периодом

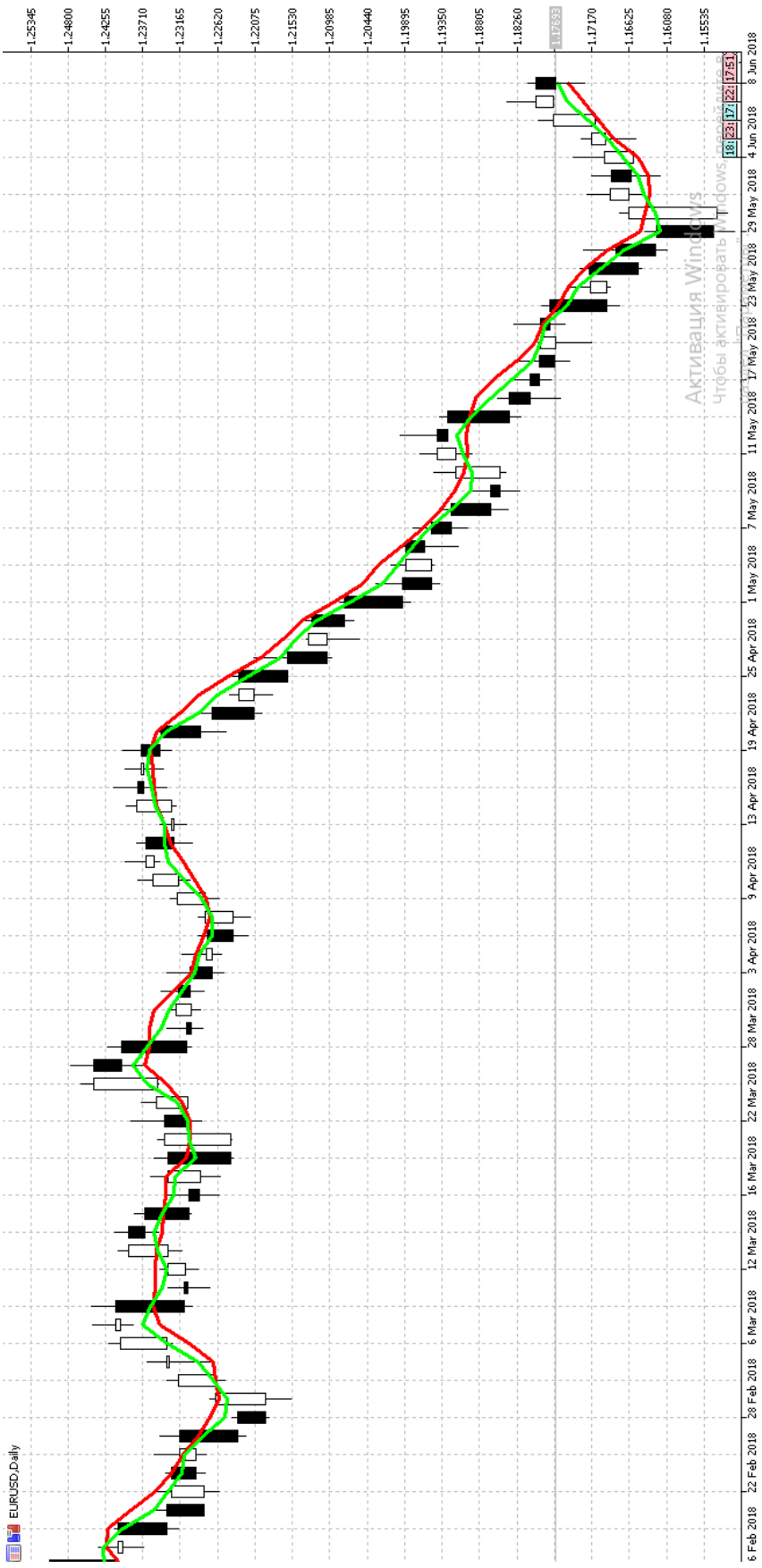

Рисунок В.2 – Простое скользящие средние и взвешенное скользящее среднее

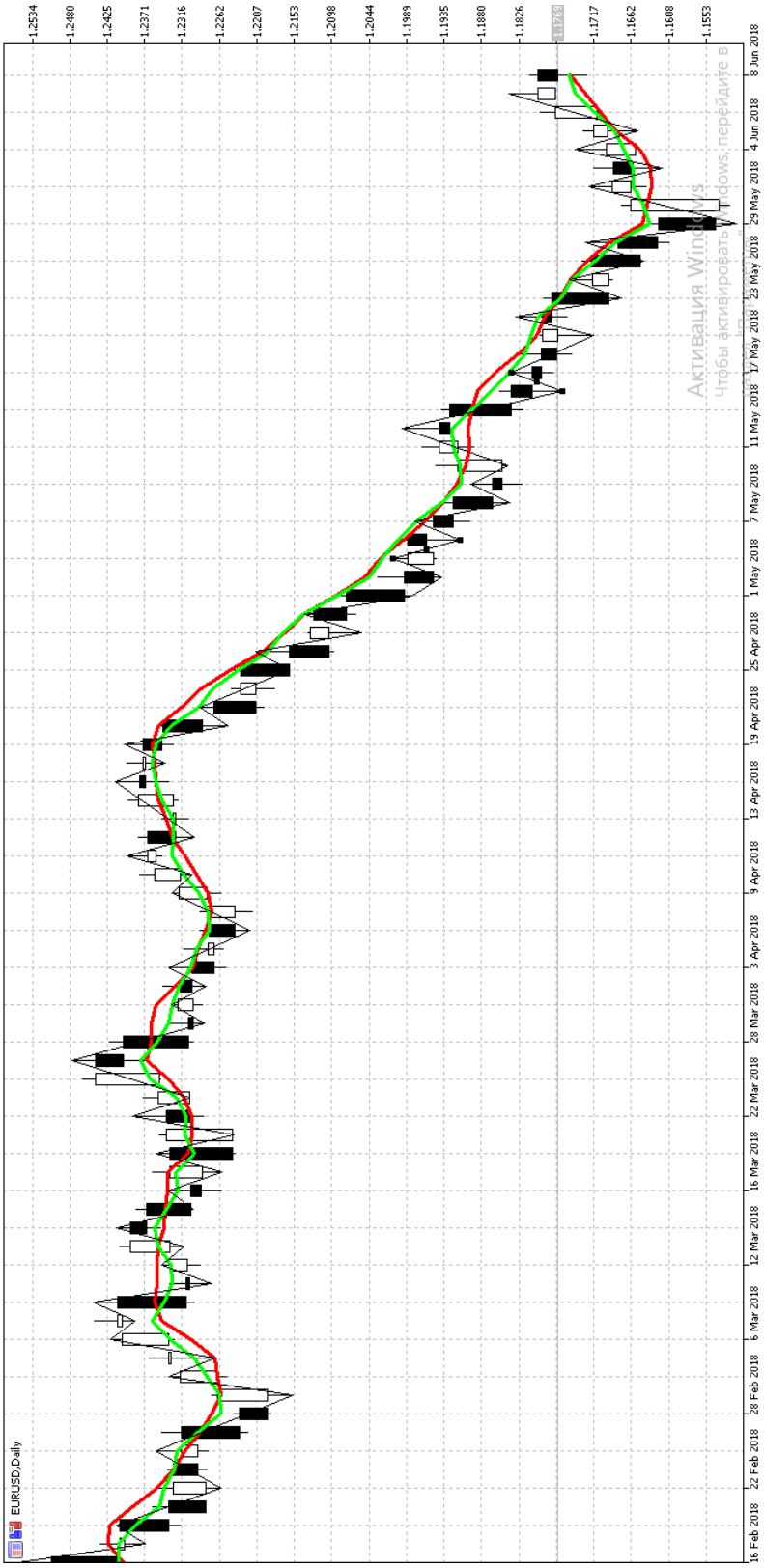

Рисунок В.3 – Простое скользящие средние и экспоненциальной скользящее среднее

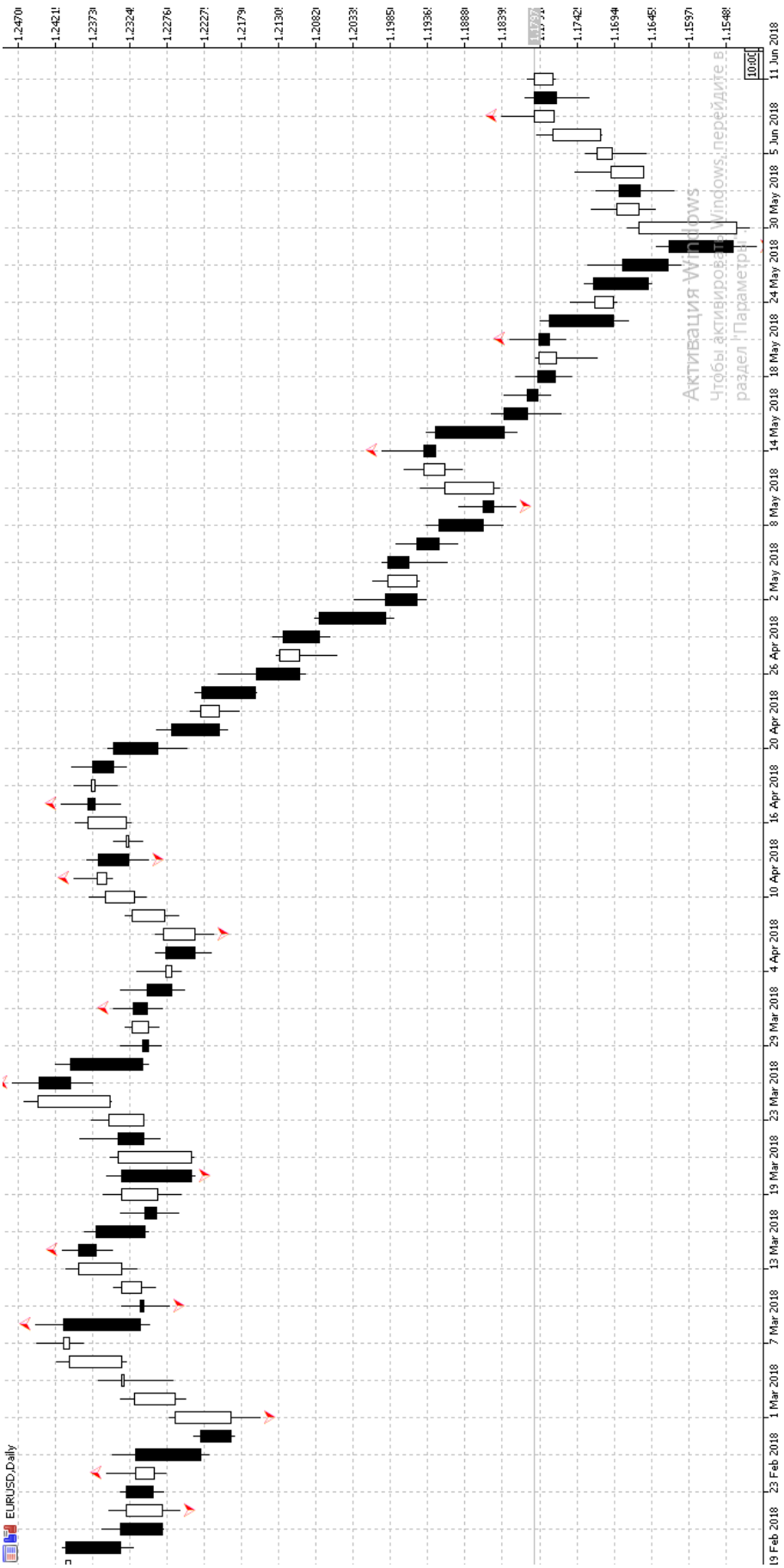

Рисунок В.4 – Фракталы

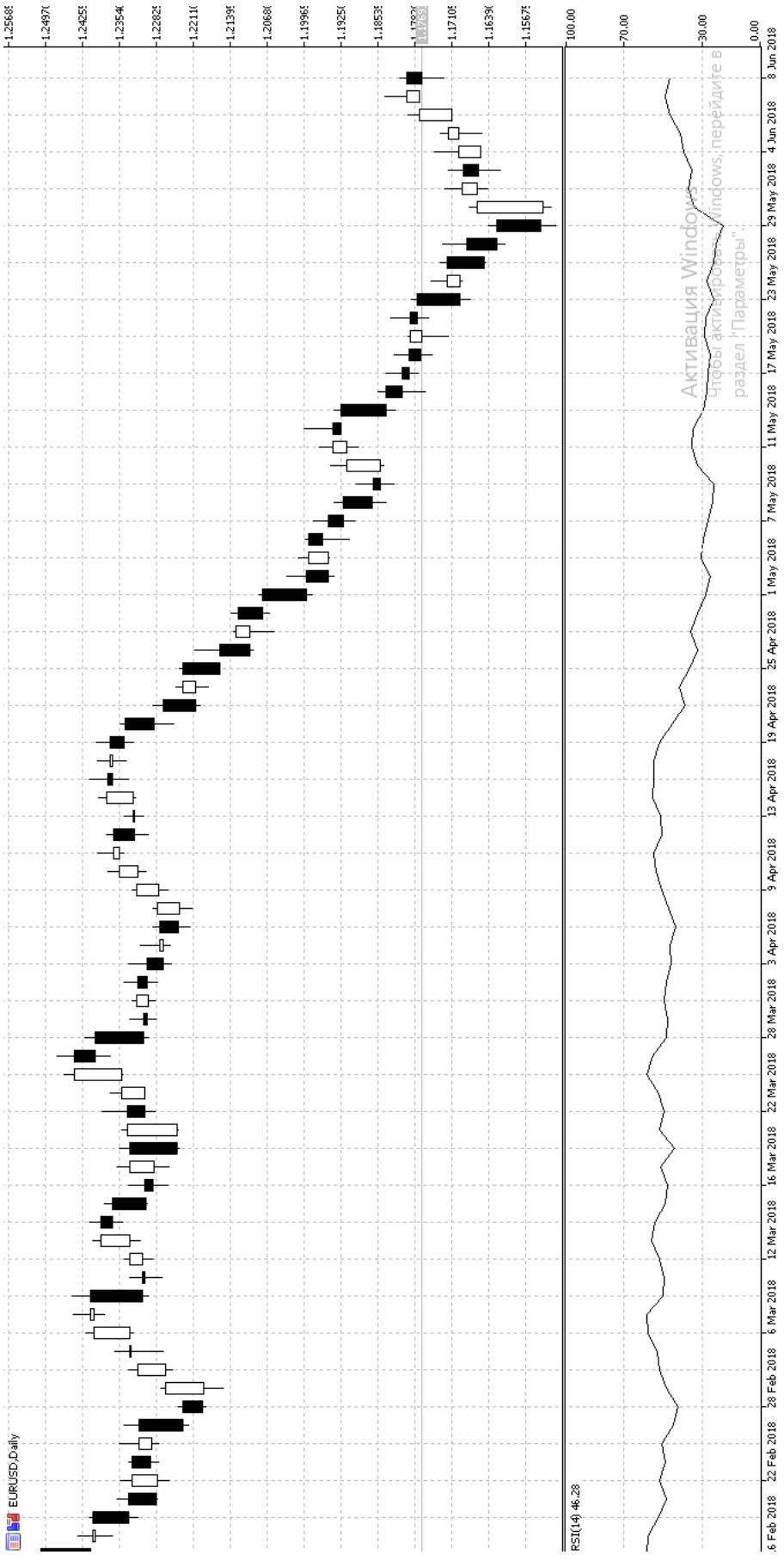

Рисунок В.5 – График цен и RSI

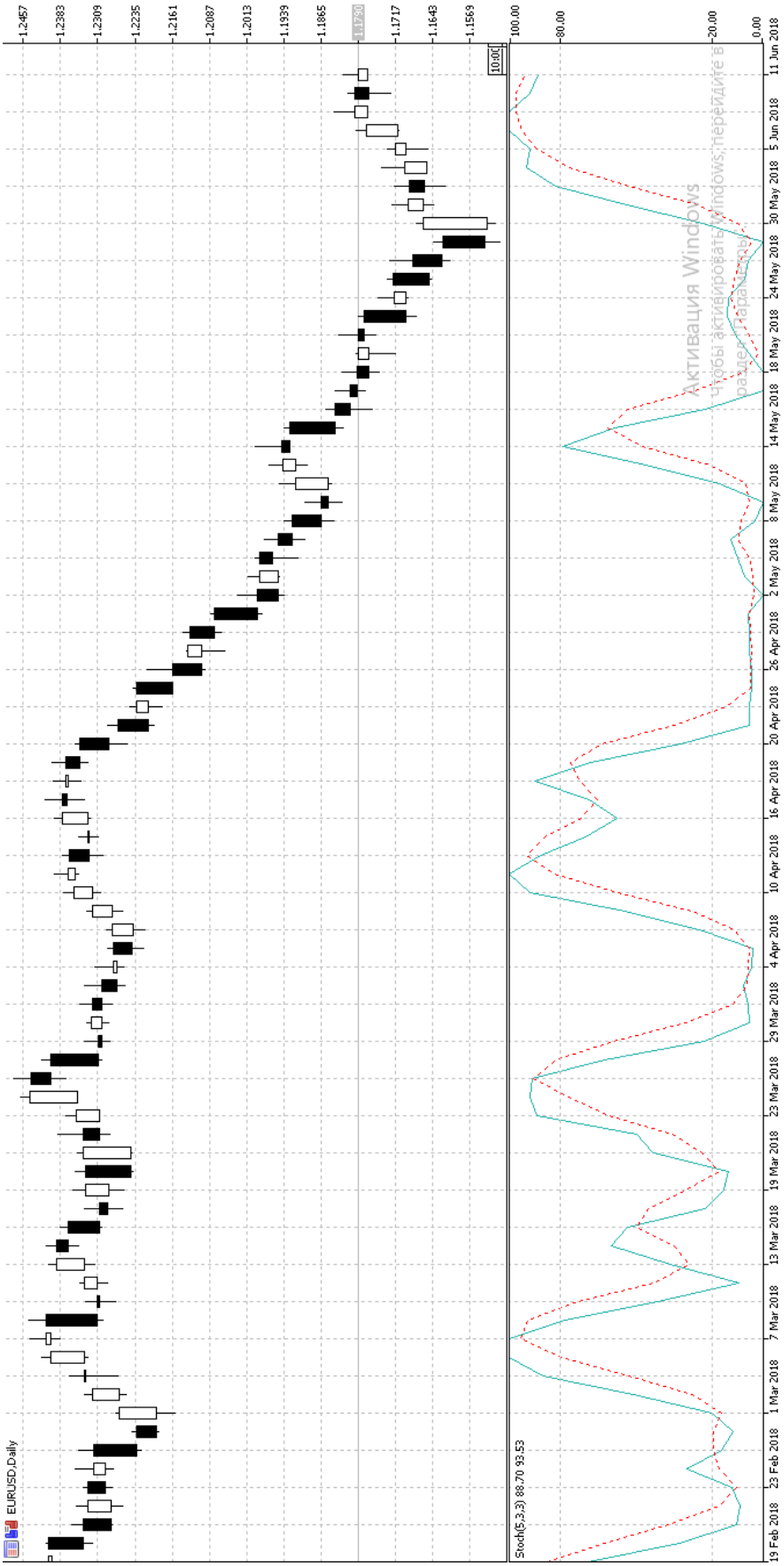

Рисунок В.6 – График цен и Stochastic oscillator

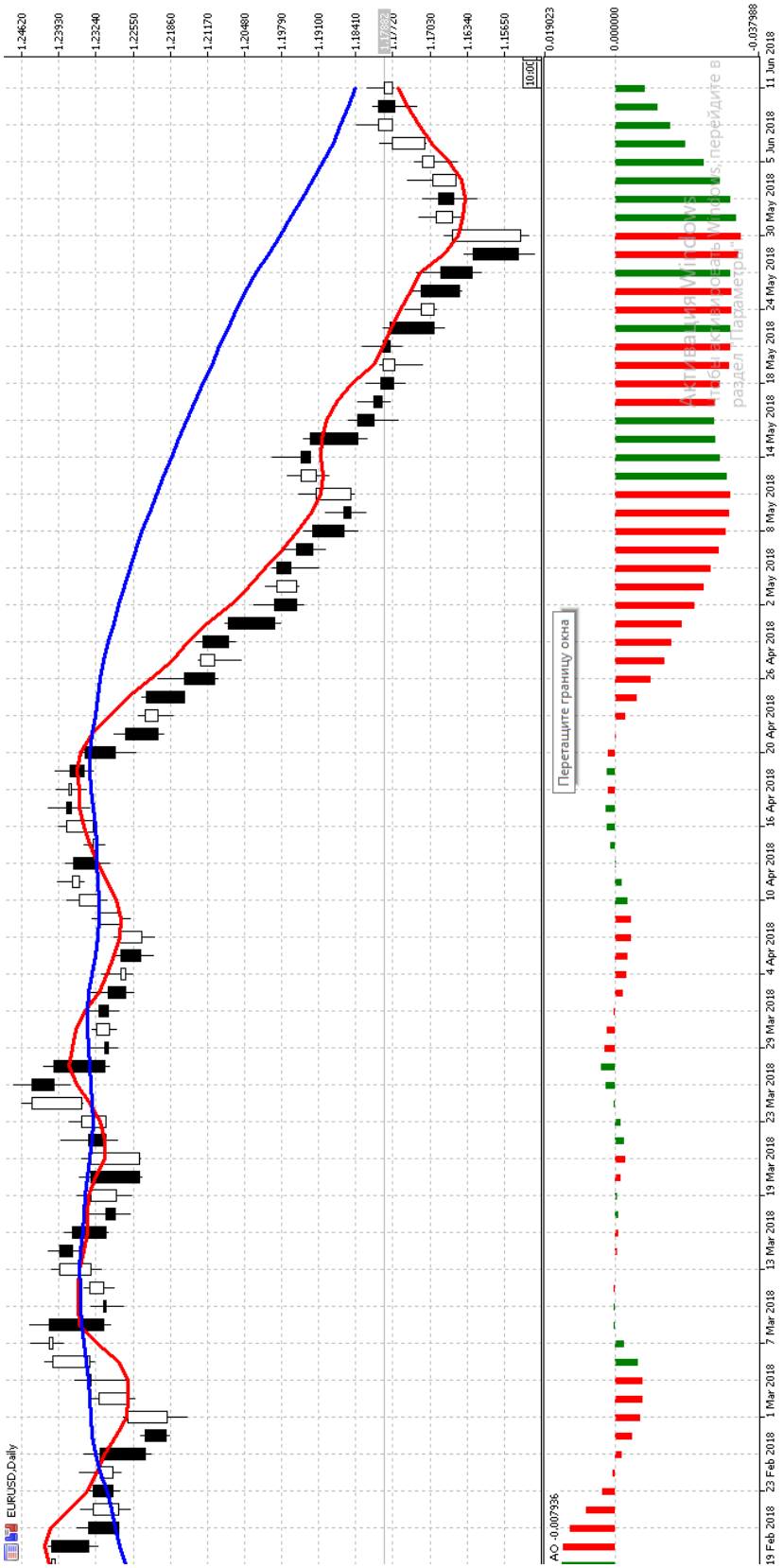

Рисунок В.7 – Построение Awesome Oscillator (AO) на основе скользящих средних

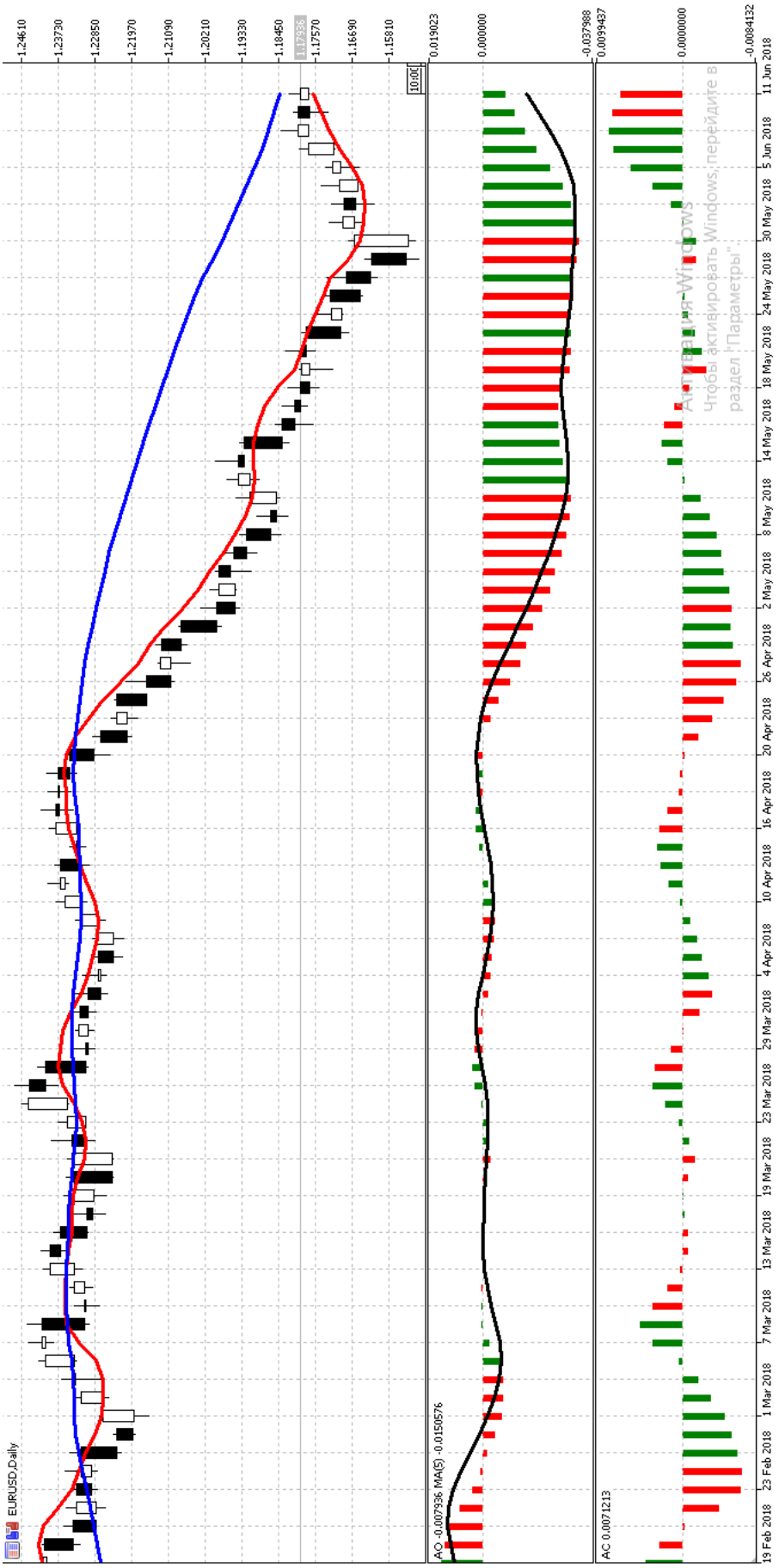

Рисунок В.8 – Построение АС на основе скользящих средних и АО

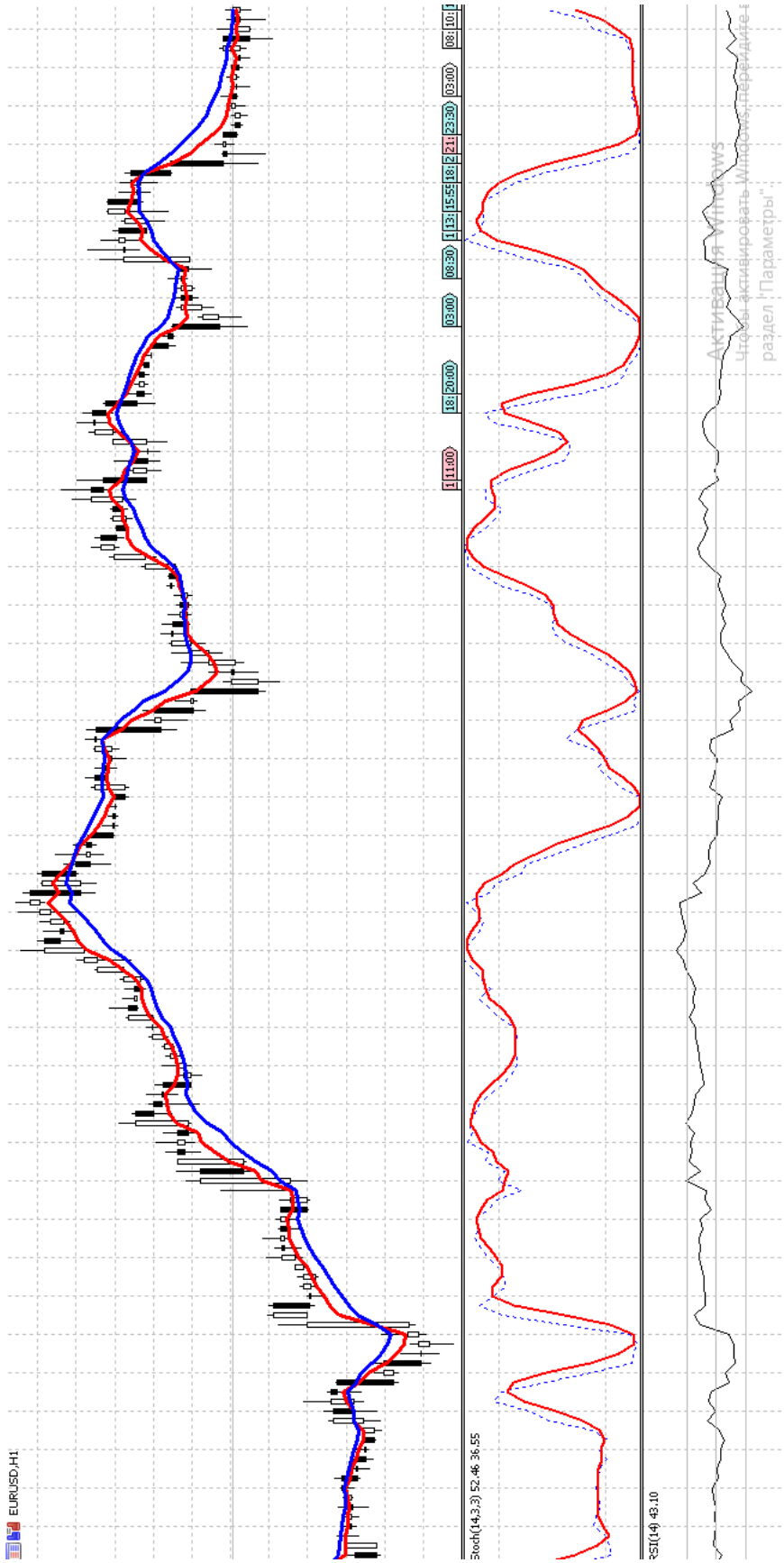

Рисунок В.9 – Торговая стратегия, основанная на EMA, Stochastic oscillator и RSI

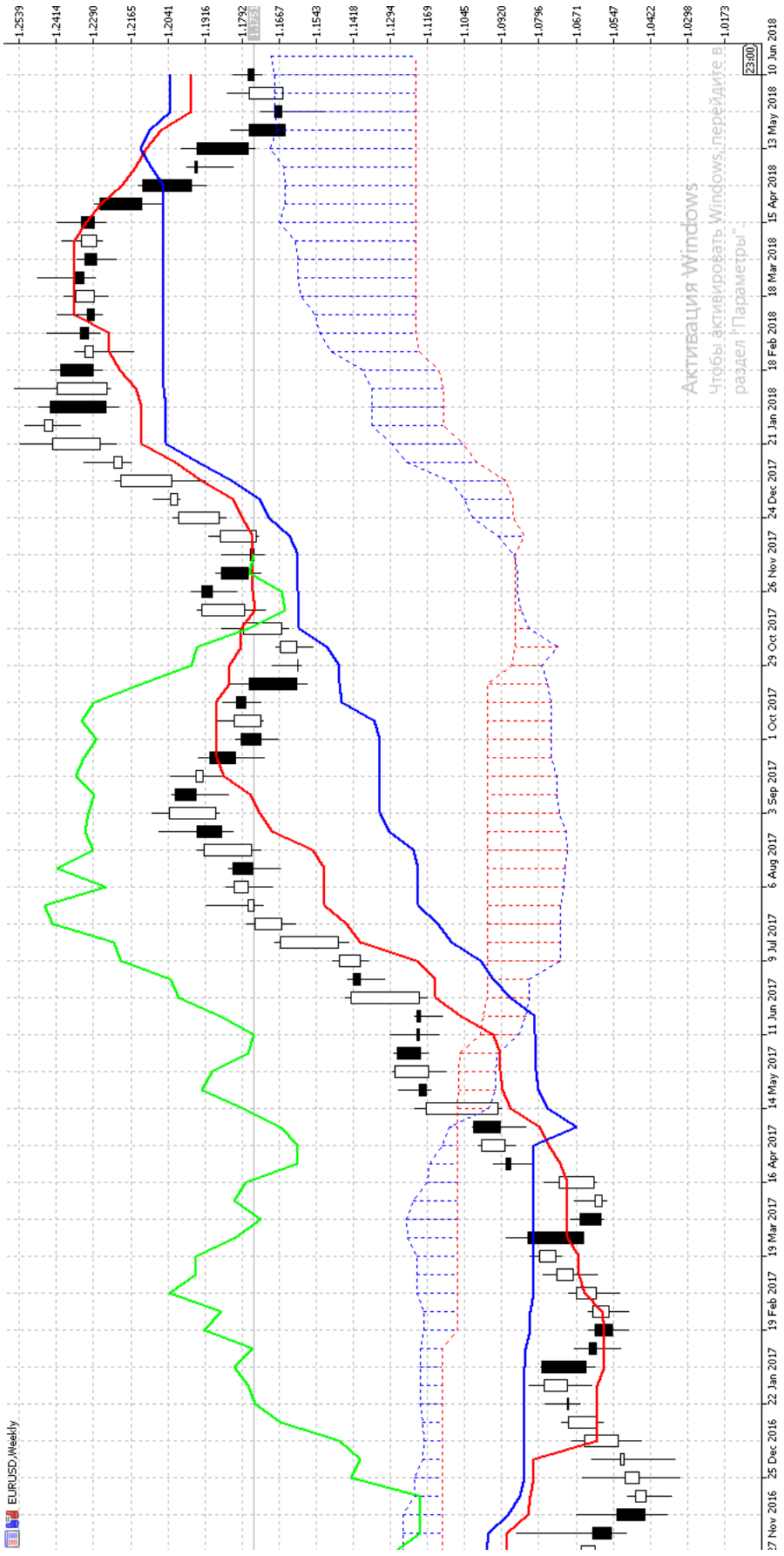

Рисунок В.10 – Торговая стратегия, основанная на индикаторе Ишимоку Кинко Хайо

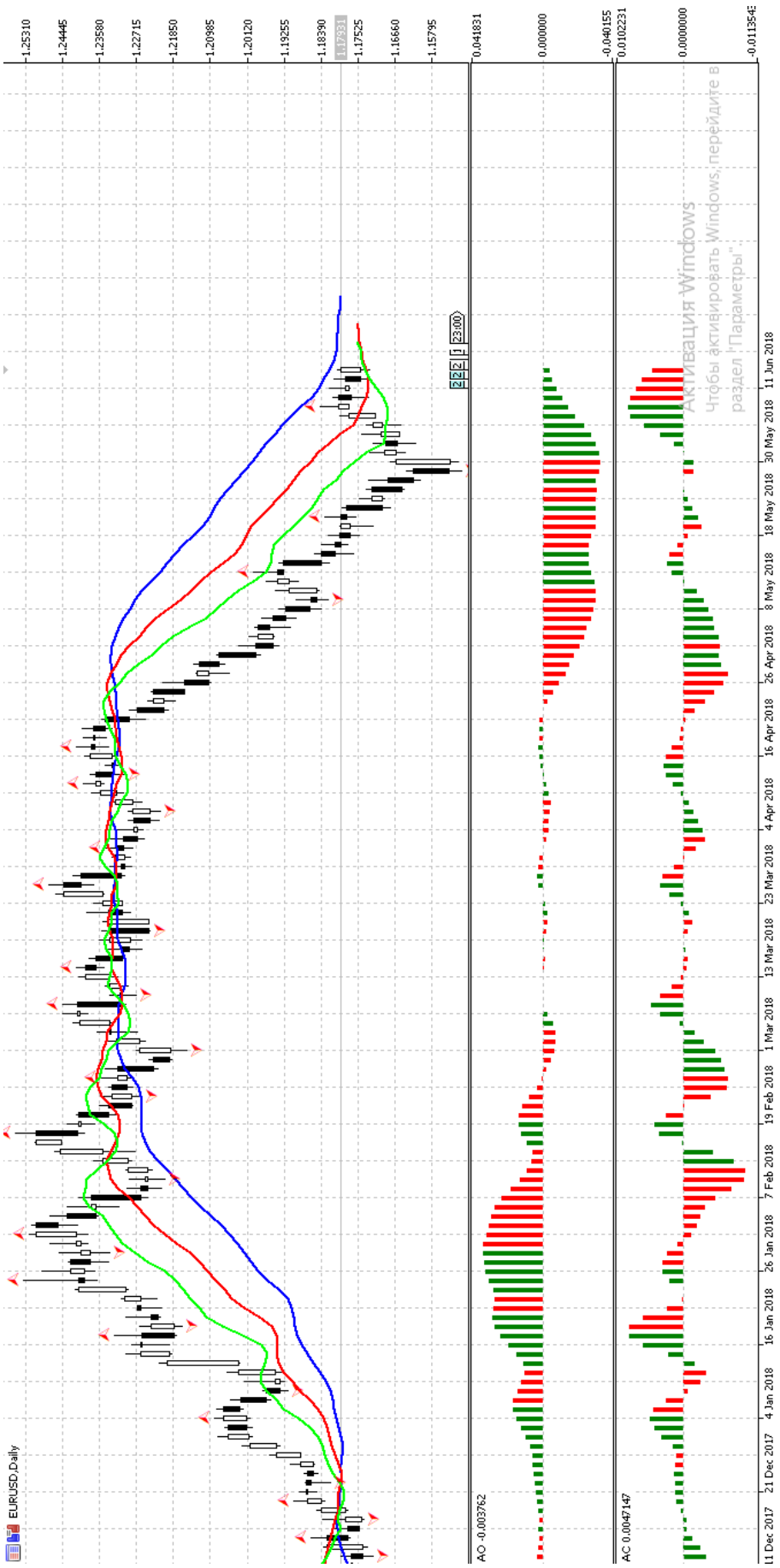

Рисунок В.11 – Торговая стратегия Билла Вильямса

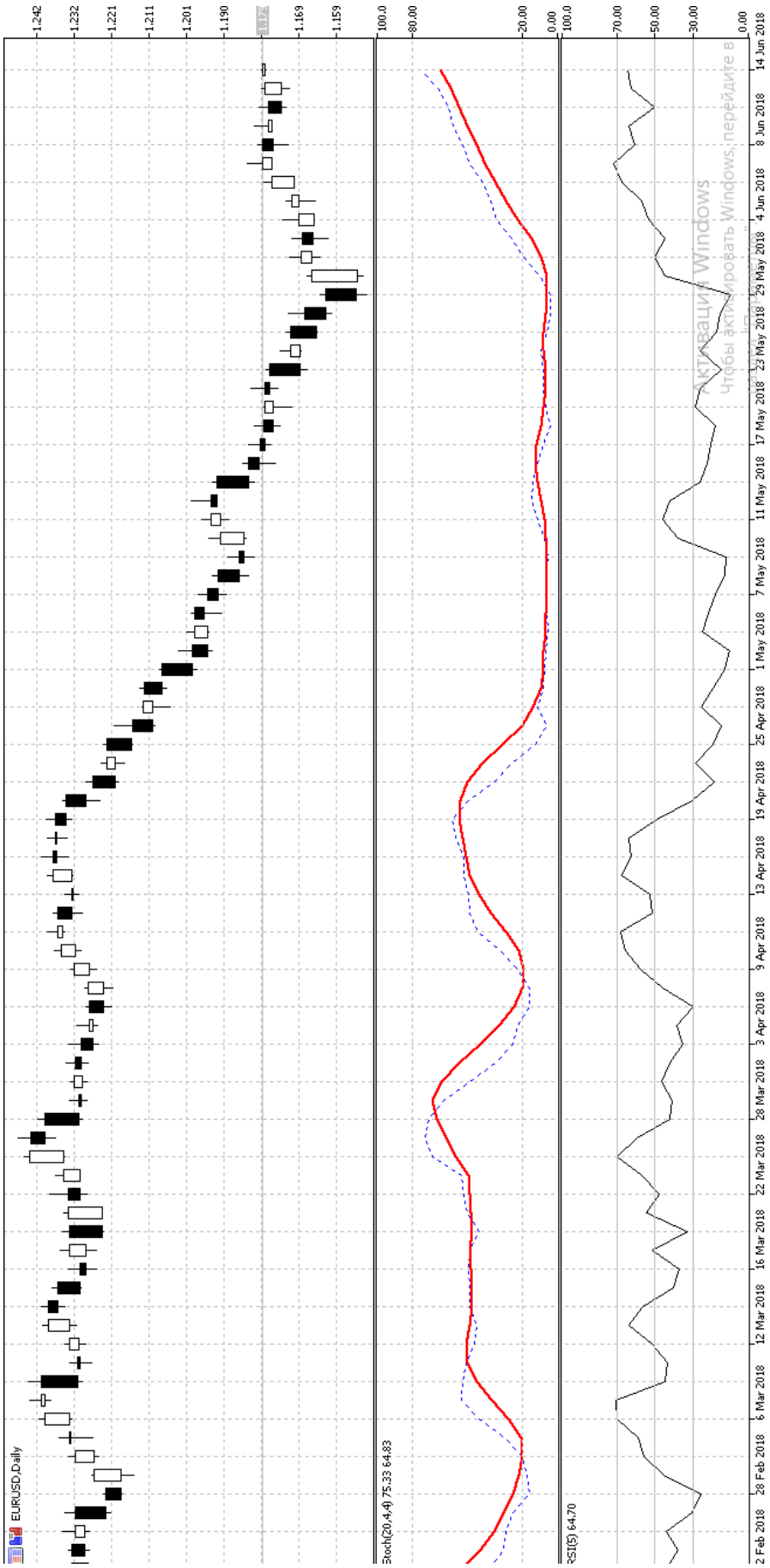

Рисунок В.12 – Торговая стратегия Stochastic oscillator (20,4) и RSI (5)

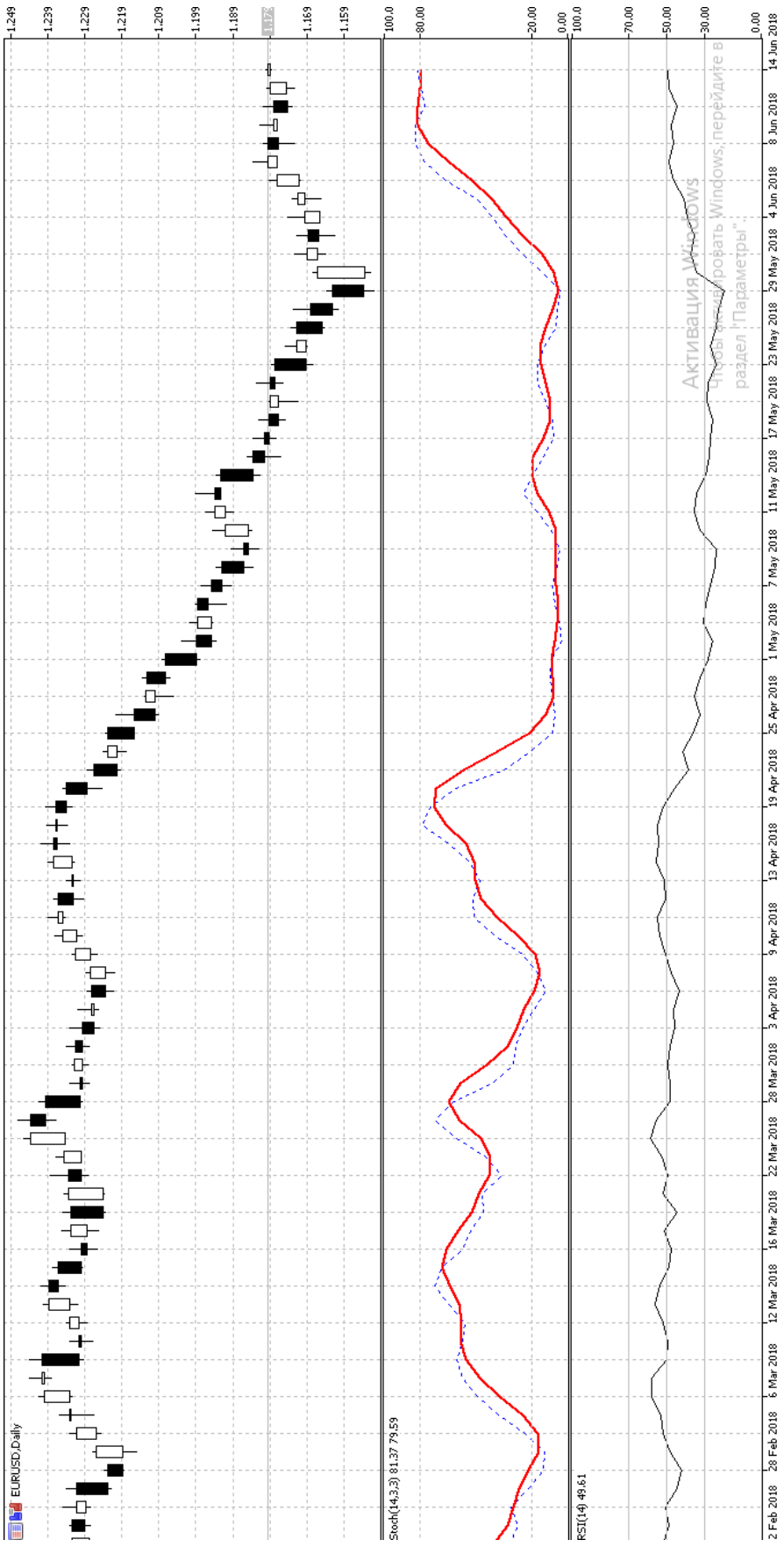

Рисунок В.13 – Торговая стратегия Stochastic oscillator (14,3) и RSI (14)

ПРИЛОЖЕНИЕ Г

Примеры программного кода

```
//+------------------------------------------------------------------+
//|                                                      Robot2.mq5 |
//|                                  Copyright 2018, Andrey Gavluck |
//|                                           gavl.andr96@gmail.com |
//+------------------------------------------------------------------
--+
#property copyright "Copyright 2018, Andrey Gavluck"
#property link      "gavl.andr96@gmail.com"
#property version   "1.00"
#include "MyRobot.mqh"
MyMyRobot myRobot;
//+------------------------------------------------------------------
--+
//| Expert initialization function
|
//+------------------------------------------------------------------
--+
int OnInit(void)
  {
//--- Инициализация программы
   if(!myRobot.OnInitEvent())
     {
      ::Print(__FUNCTION__," > Failed to initialize!");
      return(INIT_FAILED);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------
--+
//| Expert deinitialization function
|
//+------------------------------------------------------------------
--+
void OnDeinit(const int reason)
  {
   myRobot.OnDeinitEvent(reason);
  }
//+------------------------------------------------------------------
--+
//| Expert tick function
|
//+------------------------------------------------------------------
--+
void OnTick(void)
  {
   myRobot.OnTickEvent();
  }
//+------------------------------------------------------------------
--+
//| Timer function
|
```

```
//+------------------------------------------------------------------
--+
void OnTimer(void)
  {
   myRobot.OnTimerEvent();
  }
//+------------------------------------------------------------------
--+
//| ChartEvent function
|
//+------------------------------------------------------------------
--+
void OnChartEvent(const int id,const long &lparam,const double
&dparam,const string &sparam)
  {
   myRobot.ChartEvent(id,lparam,dparam,sparam);
  }
//+------------------------------------------------------------------
--+
//| Tester function
|
//+------------------------------------------------------------------
--+
double OnTester(void)
  {
   return(myRobot.OnTesterEvent());
  }
//+------------------------------------------------------------------
--+
//| TesterInit function
|
//+------------------------------------------------------------------
--+
void OnTesterInit(void)
  {
   myRobot.OnTesterInitEvent();
  }
//+------------------------------------------------------------------
--+
//| TesterPass function
|
//+------------------------------------------------------------------
--+
void OnTesterPass(void)
  {
   myRobot.OnTesterPassEvent();
  }
//+------------------------------------------------------------------
--+
//| TesterDeinit function
|
//+------------------------------------------------------------------
--+
void OnTesterDeinit(void)
  {
   myRobot.OnTesterDeinitEvent();
  }
```

```
//+------------------------------------------------------------------
--+


//+------------------------------------------------------------------+
//|                                                     MyRobot.mqh |
//|                                  Copyright 2018, Andrey Gavluck |
//|                                          gavl.andr96@gmail.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2018, Andrey Gavluck"
#property link      "gavl.andr96@gmail.com"
#include <EasyAndFastGUI\WndEvents.mqh>
#include "Strategy.mqh"
#include "FrameGenerator.mqh"
#include "FormatString.mqh"

class MyMyRobot : public CWndEvents
  {
private:
   //--- Стратегия
   Strat         m_strat;
   //--- Генератор фреймов
   CFramGen       m_frame_gen;

   //--- Окно
   CWindow           m_window1;
   //--- Статусная строка
   CStatusBar        m_status_bar;
   //--- Поля ввода
   CTextEdit         m_curves_total;
   CTextEdit         m_sleep_ms;
   //--- Кнопки
   CButton           m_reply_frames;
   //--- Таблицы
   CTable            m_table_stat;
   CTable            m_table_param;
   //--- Графики
   CGraph            m_graph1;
   CGraph            m_graph2;
   //--- Индикатор выполнения
   CProgressBar      m_progress_bar;
   //---
public:
                     MyMyRobot(void);
                    ~MyMyRobot(void);
   //--- Инициализация/деинициализация
   bool              OnInitEvent(void);
   void              OnDeinitEvent(const int reason);
   //--- Обработчик события "новый тик"
   void              OnTickEvent(void);
   //--- Обработчик торгового события
   void              OnTradeEvent(void);
   //--- Таймер
   void              OnTimerEvent(void);
   //--- Тестер
   double            OnTesterEvent(void);
   void              OnTesterPassEvent(void);
```

```
   void              OnTesterInitEvent(void);
   void              OnTesterDeinitEvent(void);
   //---
protected:
   //--- Обработчик событий графика
   virtual void      OnEvent(const int id,const long &lparam,const
double &dparam,const string &sparam);
   //---
public:
   //--- Создаёт графический интерфейс для работы с фреймами в
режиме оптимизации
   bool              CreateFrameModeGUI(void);
   //---
private:
   //--- Форма
   bool              CreateWindow(const string text);
   //--- Статусная строка
   bool              CreateStatusBar(const int x_gap,const int
y_gap);
   //--- Таблицы
   bool              CreateTableStat(const int x_gap,const int
y_gap);
   bool              CreateTableParam(const int x_gap,const int
y_gap);
   //--- Поля ввода
   bool              CreateCurvesTotal(const int x_gap,const int
y_gap,const string text);
   bool              CreateSleep(const int x_gap,const int
y_gap,const string text);
   //--- Кнопки
   bool              CreateReplyFrames(const int x_gap,const int
y_gap,const string text);
   //--- Графики
   bool              CreateGraph1(const int x_gap,const int y_gap);
   bool              CreateGraph2(const int x_gap,const int y_gap);
   //--- Индикатор выполнения
   bool              CreateProgressBar(const int x_gap,const int
y_gap,const string text);
   //---
private:
   //--- Доступность интерфейса
   void              IsAvailableGUI(const bool state);
   void              IsLockedGUI(const bool state);

   //--- Расчёт соотношения положительных и отрицательных исходов
   void              CalculateProfitsAndLosses(void);

   //--- Обновление статистической таблицы
   void              UpdateStatTable(void);
   //--- Обновление таблицы параметров
   void              UpdateParamTable(void);
   //--- Обновление графика
   void              UpdateBalanceGraph(void);

   //--- Просмотреть результаты оптимизации
   void              ViewOptimizationResults(void);
  };
```

```
//+------------------------------------------------------------------
--+
//|
Strategy.mqh |
//|                                        Copyright 2018, Andrey
Gavluck |
//|
gavl.andr96@gmail.com |
//+------------------------------------------------------------------
--+
#property copyright "Copyright 2018, Andrey Gavluck"
#property link      "gavl.andr96@gmail.com"


//---
#include <Trade\Trade.mqh>
#include <Trade\SymbolInfo.mqh>
#include <Trade\PositionInfo.mqh>
#include <Trade\AccountInfo.mqh>
//---
//---
int ExtTimeOut=10; // time out in seconds between trade operations
//+------------------------------------------------------------------
--+
//+------------------------------------------------------------------
--+
class Strat
  {
protected:
   double            m_adjusted_point;             // point value
adjusted for 3 or 5 points
   CTrade            m_trade;                      // trading
object
   CSymbolInfo       m_symbol;                     // symbol info
object
   CPositionInfo     m_position;                   // trade
position object
   CAccountInfo      m_account;                    // account info
wrapper
   //--- indicators

   //--- indicator buffers

   //--- indicator data for processing

public:
                     Strat(void);
                    ~Strat(void);
   //---
   bool              OnInitEvent();
   void              OnTickEvent(void);
   //---
   bool              Init(void);
   void              Deinit(void);
   bool              Processing(void);
```

```
protected:
   bool              InitCheckParameters(const int digits_adjust);
   bool              InitIndicators(void);
   bool              LongClosed(void);
   bool              ShortClosed(void);
   bool              LongModified(void);
   bool              ShortModified(void);
   bool              LongOpened(void);
   bool              ShortOpened(void);
  };

//+------------------------------------------------------------------
--+
//|
FrameGenerator.mqh |
//|                                     Copyright 2018, Andrey
Gavluck |
//|
gavl.andr96@gmail.com |
//+------------------------------------------------------------------
--+
#property copyright "Copyright 2018, Andrey Gavluck"
#property link      "gavl.andr96@gmail.com"
//+------------------------------------------------------------------
--+
//|
Strategy.mqh |
//|                                     Copyright 2018, Andrey
Gavluck |
//|
gavl.andr96@gmail.com |
//+------------------------------------------------------------------
--+
#property copyright "Copyright 2018, Andrey Gavluck"
#property link      "gavl.andr96@gmail.com"
class CFramGen
  {
private:
   //--- Указатели на графики для визуализации данных
   CGraphic         *m_graph_balance;
   CGraphic         *m_graph_results;
   //--- Переменные для работы с фреймами
   string            m_name;
   ulong             m_pass;
   long              m_id;
   double            m_value;
   double            m_data[];
   //--- Параметры эксперта
   string            m_param_data[];
   uint              m_par_count;
   //--- Баланс результата
   double            m_balance[];
   //--- Количество серий
   uint              m_curves_total;
   //--- Индекс текущей серии на графике
   uint              m_last_serie_index;
   //--- Счётчик фреймов
```

```
   ulong             m_frames_counter;
   //--- Всего фреймов
   ulong             m_frames_total;
   //--- Для определения максимальной серии
   double            m_curve_max[];
   //--- Данные положительных и отрицательных исходов
   double            m_loss_x[];
   double            m_loss_y[];
   double            m_profit_x[];
   double            m_profit_y[];
   //--- Массив со статистическими показателями
   string            m_stat_data[];
   //---
public:
   //---
                     CFramGen(void);
                    ~CFramGen(void);
   //--- Установка количества серий для отображения на графике
   void              SetCurvesTotal(const uint total);
   //--- (1) Количество фреймов, (2) текущий индекс проверяемого
фрейма, (3) текущий фрейм
   ulong             FramesTotal(void)  { return(m_frames_total);
}
   ulong             CurrentFrame(void) { return(m_frames_counter);
}
   ulong             CurrentPass(void)  { return(m_pass);
}
   //--- Количество отрицательных/положительных исходов
   int               LossesTotal(void)  {
return(::ArraySize(m_loss_y));   }
   int               ProfitsTotal(void) {
return(::ArraySize(m_profit_y)); }

   //--- Обработчики событий тестера стратегий
   void              OnTesterEvent(const double on_tester_value);
   void              OnTesterInitEvent(CGraphic
*graph_balance,CGraphic *graph_results);
   void              OnTesterDeinitEvent(void);
   bool              OnTesterPassEvent(void);
   //---
public:
   //--- Перебор фреймов
   bool              ReplayFrames(void);

   //--- Возвращает статистические показатели в переданный массив
   int               CopyStatData(string &dst_array[]) {
return(::ArrayCopy(dst_array,m_stat_data));  }
   //--- Возвращает параметры эксперта в переданный массив
   int               CopyParamData(string &dst_array[]) {
return(::ArrayCopy(dst_array,m_param_data)); }
   //---
private:
   //--- Получает данные баланса
   int               GetBalanceData(void);
   //--- Получает статистические данные
   void              GetStatData(double &dst_array[],double
on_tester_value);
```

```
   //--- Освободить массивы
   void              ArraysFree(void);
   //--- Сохранить статистические данные
   void              SaveStatData(void);

   //--- Обновить график результатов
   void              UpdateResultsGraph(void);
   //--- Обновить график балансов
   void              UpdateBalanceGraph(void);

   //--- Финальный пересчёт данных со всех фреймов после
оптимизации
   void              FinalRecalculateFrames(void);

   //--- Добавляет (1) отрицательный и (2) положительный результат
в массивы
   void              AddLoss(const double loss);
   void              AddProfit(const double profit);
  };

//+------------------------------------------------------------------
--+
//|
GENETICALGO lib.mqh |
//|                                        Copyright 2018, Andrey
Gavluck |
//|
gavl.andr96@gmail.com |
//+------------------------------------------------------------------
--+
//Библиотека "Универсального Генетического Алгоритма GENETICALGO
lib"             |
//использующего представление хромосомы вещественными числами.
|
//+─────────────────────────────────────────────────────────────────
──────+

//----------------------Глобальные переменные----------------------
-------
double Persone[];            //Набор оптимизируемых аргументов
функции - генов
                                //(например: веса нейронной сети и
т.д.)-хромосома
int    PersoneCount     =0;  //Максимально возможное количество
хромосом в колонии
int    TotalOfPersonesInHistory=0;//Общее количество хромосом в
истории
int    ChrCountInHistory   =0;  //Количество уникальных хромосом в
базе хромосом
int    GeneCount           =0;  //Количество генов в хромосоме

double RMin     =0.0;//Минимум диапазона поиска
double RMax        =0.0;//Максимум диапазона поиска
double Precision           =0.0;//Шаг поиска
int    OptMet      =0;  //1-минимум, любое другое - максимум
```

```
double Population   [][1000];   //Популяция
double Colony       [][500];    //Колония потомков
int    PopulPersCount   =0;  //Текущее количество хромосом в
популяции
int    Epoch               =0;  //Кол-во эпох без улучшения
int    AmountStartsFF=0;        //Количество запусков функции
приспособленности
//---------------------------------------------------------------------
-------

//---------------------------------------------------------------------
-------
//Основная функция GENETICALGO
void GENETICALGO
(
double ReplicP, //Доля Репликации.
double NMutP,   //Доля Естественной мутации.
double ArtiMut, //Доля Искусственной мутации.
double GMPort, //Доля Заимствования генов.
double GMPort,//Доля Кроссинговера.
//---
double ReplicOf,  //Коэффициент смещения границ интервала
double NMutProbab//Вероятность мутации каждого гена в %
)
{
  //сброс генератора, производится только один раз
  MathSrand((int)TimeLocal());
  //-----------------------Переменные-----------------------------
-------
  int    pers=0, gene  =0;//индексы хромосом и генов
  int    resetCounterFF   =0;//счетчик сбросов "Эпох без улучшений"
  int    currEpo     =1;//номер текущей эпохи
  int    SumOfCurrEpo=0;//сумма "Эпох без улучшений"
  int    MinOfCurrEpo=Epoch;//минимальное "Эпох без улучшений"
  int    MaxOfCurrEpo=0;//максимальное "Эпох без улучшений"
  int    epochGlob        =0;//общее количество эпох
  // Колония [количество признаков(генов)][количество особей в
колонии]
  ArrayResize    (Population,GeneCount+1);
  ArrayInitialize(Population,0.0);
  // Колония потомков [количество признаков(генов)][количество
особей в колонии]
  ArrayResize    (Colony,GeneCount+1);
  ArrayInitialize(Colony,0.0);
  // Банк хромосом
  // [количество признаков(генов)][количество хромосом в банке]
  double          historyHromosomes[][100000];
  ArrayResize    (historyHromosomes,GeneCount+1);
  ArrayInitialize(historyHromosomes,0.0);
  //---------------------------------------------------------------
-------
  //--------------Проверка корректности входных параметров---------
-------
  //...количество хромосом должно быть не меньше 2
  if (PersoneCount<=1)  PersoneCount=2;
  if (PersoneCount>500) PersoneCount=500;
```

```
  //-------------------------------------------------------------
-------

//=============================================================
=====
  // 1) Создать протопопуляцию
------1)
  ProtopopulationBuilding ();

//=============================================================
=====
  // 2) Определить приспособленность каждой особи
------2)
  //Для 1-ой колонии
  for (pers=0;pers<PersoneCount;pers++)
    for (gene=1;gene<=GeneCount;gene++)
      Colony[gene][pers]=Population[gene][pers];

  GetFitness(historyHromosomes);

  for (pers=0;pers<PersoneCount;pers++)
    Population[0][pers]=Colony[0][pers];

  //Для 2-ой колонии
  for (pers=PersoneCount;pers<PersoneCount*2;pers++)
    for (gene=1;gene<=GeneCount;gene++)
      Colony[gene][pers-PersoneCount]=Population[gene][pers];

  GetFitness(historyHromosomes);

  for (pers=PersoneCount;pers<PersoneCount*2;pers++)
    Population[0][pers]=Colony[0][pers-PersoneCount];

//=============================================================
=====
  // 3) Подготовить популяцию к размножению
-----3)
  RemovalDuplicates();

//=============================================================
=====
  // 4) Выделить эталонную хромосому
------4)
  for (gene=0;gene<=GeneCount;gene++)
    Persone[gene]=Population[gene][0];

//=============================================================
=====
  ServiceFunction();

  //Основной цикл генетического алгоритма с 5 по 6
  while (currEpo<=Epoch)
  {

//=============================================================
===
```

```
    // 5) Операторы GENETICALGO
-----5)
    CycleOfOperators
    (
    historyHromosomes,
    //---
    ReplicP, //Доля Репликации.
    NMutP,   //Доля Естественной мутации.
    ArtiMut, //Доля Искусственной мутации.
    GMPort, //Доля Заимствования генов.
    GMPort,//Доля Кроссинговера.
    //---
    ReplicOf,  //Коэффициент смещения границ интервала
    NMutProbab//Вероятность мутации каждого гена в %
    );

//=================================================================
===
    // 6) Сравнить гены лучшего потомка с генами эталонной
хромосомы.
    // Если хромосома лучшего потомка лучше эталонной,
    // заменить эталонную.
-----6)
    //Если режим оптимизации - минимизация
    if (OptMet==1)
    {
      //Если лучшая хромосома популяции лучше эталонной
      if (Population[0][0]<Persone[0])
      {
        //Заменим эталонную хромосому
        for (gene=0;gene<=GeneCount;gene++)
          Persone[gene]=Population[gene][0];
        ServiceFunction();
        //Сбросим счетчик "эпох без улучшений"
        if (currEpo<MinOfCurrEpo)
          MinOfCurrEpo=currEpo;
        if (currEpo>MaxOfCurrEpo)
          MaxOfCurrEpo=currEpo;
        SumOfCurrEpo+=currEpo; currEpo=1; resetCounterFF++;
      }
      else
        currEpo++;
    }
    //Если режим оптимизации - максимизация
    else
    {
      //Если лучшая хромосома популяции лучше эталонной
      if (Population[0][0]>Persone[0])
      {
        //Заменим эталонную хромосому
        for (gene=0;gene<=GeneCount;gene++)
          Persone[gene]=Population[gene][0];
        ServiceFunction();
        //Сбросим счетчик "эпох без улучшений"
        if (currEpo<MinOfCurrEpo)
          MinOfCurrEpo=currEpo;
        if (currEpo>MaxOfCurrEpo)
```

```
          MaxOfCurrEpo=currEpo;
        SumOfCurrEpo+=currEpo; currEpo=1; resetCounterFF++;
      }
      else
        currEpo++;
    }

//=================================================================
===
    //Прошла ещё одна эпоха....
    epochGlob++;
  }
  Print("Прошло всего эпох=",epochGlob," Всего
сбросов=",resetCounterFF);
  Print("Мин.эпох без улучш.=",MinOfCurrEpo,
        " Средн.эпох без улучш.=",

NormalizeDouble((double)SumOfCurrEpo/(double)resetCounterFF,2),
        " Макс.эпох без улучш.=",MaxOfCurrEpo);
  Print(ChrCountInHistory," - Уникальных хромосом");
  Print(AmountStartsFF," - Общее кол-во запусков FF");
  Print(TotalOfPersonesInHistory," - Общее кол-во хромосом в
истории");
  Print(NormalizeDouble(100.0-((double)ChrCountInHistory*100.0/

(double)TotalOfPersonesInHistory),2),"% дубликатов");
  Print(Persone[0]," - Лучший результат");
}


//-----------------------------------------------------------------
-------
//Создание протопопуляции
void ProtopopulationBuilding()
{
  PopulPersCount=PersoneCount*2;
  //Заполнить популяцию хромосомами со случайными
  //...генами в диапазоне RMin...RMax
  for (int pers=0;pers<PopulPersCount;pers++)
  {
    //начиная с 1-го индекса (0-ой -зарезервирован для VFF)
    for (int gene=1;gene<=GeneCount;gene++)
      Population[gene][pers]=

SelectInDiscreteSpace(RNDfromCI(RMin,RMax),RMin,RMax,Precision,3);
    TotalOfPersonesInHistory++;
  }
}
//-----------------------------------------------------------------
-------

//-----------------------------------------------------------------
-------
//Получение приспособленности для каждой особи.
void GetFitness
(
double &historyHromosomes[][100000]
```

```
)
{
  for (int pers=0;pers<PersoneCount;pers++)
    CheckHistoryPersones(pers,historyHromosomes);
}
//---------------------------------------------------------------------
-------

//---------------------------------------------------------------------
-------
//Проверка хромосомы по базе хромосом.
void CheckHistoryPersones
(
int     pers,
double &historyHromosomes[][100000]
)
{
  //-----------------------Переменные-----------------------------
-------
  int   Ch1=0;  //Индекс хромосомы из базы
  int   Ge =0;  //Индекс гена
  int   cnt=0;  //Счетчик уникальных генов. Если хоть один ген
отличается
                //- хромосома признается уникальной
  //---------------------------------------------------------------
-------
  //Если в базе хранится хоть одна хромосома
  if (ChrCountInHistory>0)
  {
    //Переберем хромосомы в базе, чтобы найти такую же
    for (Ch1=0;Ch1<ChrCountInHistory && cnt<GeneCount;Ch1++)
    {
      cnt=0;
      //Сверяем гены, пока индекс гена меньше кол-ва генов и пока
попадаются одинаковые гены
      for (Ge=1;Ge<=GeneCount;Ge++)
      {
        if (Colony[Ge][pers]!=historyHromosomes[Ge][Ch1])
          break;
        cnt++;
      }
    }
    //Если набралось одинаковых генов столько же, можно взять
готовое решение из базы
    if (cnt==GeneCount)
      Colony[0][pers]=historyHromosomes[0][Ch1-1];
    //Если нет такой же хромосомы в базе, то рассчитаем для неё
FF...
    else
    {
      FitnessFunction(pers);
      //.. и если есть место в базе сохраним
      if (ChrCountInHistory<100000)
      {
        for (Ge=0;Ge<=GeneCount;Ge++)

historyHromosomes[Ge][ChrCountInHistory]=Colony[Ge][pers];
```

```
        ChrCountInHistory++;
      }
    }
  }
  //Если база пустая, рассчитаем для неё FF и сохраним её в базе
  else
  {
    FitnessFunction(pers);
    for (Ge=0;Ge<=GeneCount;Ge++)
      historyHromosomes[Ge][ChrCountInHistory]=Colony[Ge][pers];
    ChrCountInHistory++;
  }
}
//------------------------------------------------------------------
-------

//------------------------------------------------------------------
-------
//Цикл операторов GENETICALGO
void CycleOfOperators
(
double &historyHromosomes[][100000],
//---
double    ReplicP, //Доля Репликации.
double    NMutP,   //Доля Естественной мутации.
double    ArtiMut, //Доля Искусственной мутации.
double    GMPort, //Доля Заимствования генов.
double    GMPort,//Доля Кроссинговера.
//---
double    ReplicOf,  //Коэффициент смещения границ интервала
double    NMutProbab//Вероятность мутации каждого гена в %
)
{
  //----------------------Переменные-----------------------------
-------
  double          child[];
  ArrayResize    (child,GeneCount+1);
  ArrayInitialize(child,0.0);

  int gene=0,pers=0, border=0;
  int    i=0,u=0;
  double p=0.0,start=0.0;
  double          fit[][2];
  ArrayResize    (fit,6);
  ArrayInitialize(fit,0.0);

  //Счетчик посадочных мест в новой популяции.
  int T=0;
  //------------------------------------------------------------
-------

  //Зададим долю операторов GENETICALGO
  double portion[6];
  portion[0]=ReplicP; //Доля Репликации.
  portion[1]=NMutP;   //Доля Естественной мутации.
  portion[2]=ArtiMut; //Доля Искусственной мутации.
  portion[3]=GMPort; //Доля Заимствования генов.
```

```
  portion[4]=GMPort;//Доля Кроссинговера.
  portion[5]=0.0;
  //----------------------------
  if (NMutProbab<0.0)
    NMutProbab=0.0;
  if (NMutProbab>100.0)
    NMutProbab=100.0;
  //----------------------------
  //-------------------------Цикл операторов GENETICALGO ---------
  //Заполняем новую колонию потомками
  while (T<PersoneCount)
  {
    //============================
    for (i=0;i<6;i++)
    {
      fit[i][0]=start;
      fit[i][1]=start+MathAbs(portion[i]-portion[5]);
      start=fit[i][1];
    }
    p=RNDfromCI(fit[0][0],fit[4][1]);
    for (u=0;u<5;u++)
    {
      if ((fit[u][0]<=p && p<fit[u][1]) || p==fit[u][1])
        break;
    }
    //============================
    switch (u)
    {
    //---------------------
    case 0:
      //-------------------------Репликация-------------------------
-------
      //Если есть место в новой колонии, создадим новую особь
      if (T<PersoneCount)
      {
        Replication(child,ReplicOf);
        //Поселим новую особь в новую колонию
        for (gene=1;gene<=GeneCount;gene++)
Colony[gene][T]=child[gene];
        //Одно место заняли, счетчик перемотаем вперед
        T++;
        TotalOfPersonesInHistory++;
      }
      //------------------------------------------------------------
----
      break;
      //---------------------
    case 1:
      //---------------------Естественная мутация-----------------
-------
      //Если есть место в новой колонии, создадим новую особь
      if (T<PersoneCount)
      {
        NaturalMutation(child,NMutProbab);
        //Поселим новую особь в новую колонию
        for (gene=1;gene<=GeneCount;gene++)
Colony[gene][T]=child[gene];
```

```
        //Одно место заняли, счетчик перемотаем вперед
        T++;
        TotalOfPersonesInHistory++;
      }
      //-------------------------------------------------------
----
      break;
      //---------------------
    case 2:
      //----------------------Искусственная мутация-----------------
-------
      //Если есть место в новой колонии, создадим новую особь
      if (T<PersoneCount)
      {
        ArtiMut(child,ReplicOf);
        //Поселим новую особь в новую  колонию
        for (gene=1;gene<=GeneCount;gene++)
Colony[gene][T]=child[gene];
        //Одно место заняли, счетчик перемотаем вперед
        T++;
        TotalOfPersonesInHistory++;
      }
      //-------------------------------------------------------
----
      break;
      //---------------------
    case 3:
      //-------------Образование особи с заимствованными генами----
-------
      //Если есть место в новой колонии, создадим новую особь
      if (T<PersoneCount)
      {
        GenoMerging(child);
        //Поселим новую особь в новую колонию
        for (gene=1;gene<=GeneCount;gene++)
Colony[gene][T]=child[gene];
        //Одно место заняли, счетчик перемотаем вперед
        T++;
        TotalOfPersonesInHistory++;
      }
      //-------------------------------------------------------
----
      break;
      //---------------------
    default:
      //---------------------------Кроссинговер--------------------
-------
      //Если есть место в новой колонии, создадим новую особь
      if (T<PersoneCount)
      {
        CrossingOver(child);
        //Поселим новую особь в новую  колонию
        for (gene=1;gene<=GeneCount;gene++)
Colony[gene][T]=child[gene];
        //Одно место заняли, счетчик перемотаем вперед
        T++;
        TotalOfPersonesInHistory++;
```

```
      }
      //-------------------------------------------------------------
----

      break;
      //---------------------
    }
  }//Конец цикла операторов GENETICALGO --

  //Определим приспособленность каждой особи в колонии потомков
  GetFitness(historyHromosomes);

  //Поселим потомков в основную популяцию
  if (PopulPersCount>=PersoneCount)
  {
    border=PersoneCount;
    PopulPersCount=PersoneCount*2;
  }
  else
  {
    border=PopulPersCount;
    PopulPersCount+=PersoneCount;
  }
  for (pers=0;pers<PersoneCount;pers++)
    for (gene=0;gene<=GeneCount;gene++)
      Population[gene][pers+border]=Colony[gene][pers];

  //Подготовим популяцию к следующему размножению
  RemovalDuplicates();
}//конец ф-ии
//————————————————————————————————————————————————————————————
———————

//————————————————————————————————————————————————————————————
———————
//Репликация
void Replication
(
double &child[],
double  ReplicOf
)
{
  //-----------------------Переменные-----------------------------
-------
  double C1=0.0,C2=0.0,temp=0.0,Maximum=0.0,Minimum=0.0;
  int address_mama=0,address_papa=0;
  //-------------------------------------------------------------
-------
  SelectTwoParents(address_mama,address_papa);
  //------------------Цикл перебора генов-------------------------
-------
  for (int i=1;i<=GeneCount;i++)
  {
    //----определим откуда мать и отец --------
    C1 = Population[i][address_mama];
    C2 = Population[i][address_papa];
    //----------------------------------------
```

```
    //-------------------------------------------------------------
-----
    //....определим наибольший и наименьший из них,
    //если C1>C2, поменяем их местами
    if (C1>C2)
    {
      temp = C1; C1=C2; C2 = temp;
    }
    //-------------------------------------------
    if (C2-C1<Precision)
    {
      child[i]=C1; continue;
    }
    //-------------------------------------------
    //Назначим границы создания нового гена
    Minimum = C1-((C2-C1)*ReplicOf);
    Maximum = C2+((C2-C1)*ReplicOf);
    //-------------------------------------------
    //Обязательная проверка, что бы поиск не вышел из заданного
диапазона
    if (Minimum < RMin) Minimum = RMin;
    if (Maximum > RMax) Maximum = RMax;
    //-------------------------------------------------------------
--
    temp=RNDfromCI(Minimum,Maximum);
    child[i]=
    SelectInDiscreteSpace(temp,RMin,RMax,Precision,3);
  }
}
//---------------------------------------------------------------------
-------

//---------------------------------------------------------------------
-------
//Естественная мутация.
void NaturalMutation
(
double &child[],
double  NMutProbab
)
{
  //-----------------------Переменные-----------------------------
-------
  int    address=0;
  //-------------------------------------------------------------
-------

  //-----------------Отбор родителя-----------------------
  SelectOneParent(address);
  //------------------------------------
  for (int i=1;i<=GeneCount;i++)
    if (RNDfromCI(0.0,100.0)<=NMutProbab)
      child[i]=

SelectInDiscreteSpace(RNDfromCI(RMin,RMax),RMin,RMax,Precision,3);
    else
```

```
      child[i]=Population[i][address];
}
//------------------------------------------------------------------
-------

//------------------------------------------------------------------
-------
//Искусственная мутация.
void ArtiMut
(
double &child[],
double  ReplicOf
)
{
  //----------------------Переменные-----------------------------
-------
  double C1=0.0,C2=0.0,temp=0.0,Maximum=0.0,Minimum=0.0,p=0.0;
  int address_mama=0,address_papa=0;
  //------------------------------------------------------------
-------
  //-----------------Отбор родителей-----------------------
  SelectTwoParents(address_mama,address_papa);
  //-------------------------------------------------------
  //-------------------Цикл перебора генов-------------------------
-----
  for (int i=1;i<=GeneCount;i++)
  {
    //----определим откуда мать и отец --------
    C1 = Population[i][address_mama];
    C2 = Population[i][address_papa];
    //-----------------------------------------

    //------------------------------------------------------------
-----
    //....определим наибольший и наименьший из них,
    //если C1>C2, поменяем их местами
    if (C1>C2)
    {
      temp=C1; C1=C2; C2=temp;
    }
    //-----------------------------------------
    //Назначим границы создания нового гена
    Minimum=C1-((C2-C1)*ReplicOf);
    Maximum=C2+((C2-C1)*ReplicOf);
    //-----------------------------------------
    //Обязательная проверка, что бы поиск не вышел из заданного
диапазона
    if (Minimum < RMin) Minimum = RMin;
    if (Maximum > RMax) Maximum = RMax;
    //------------------------------------------------------------
--
    p=MathRand();
    if (p<16383.5)
    {
      temp=RNDfromCI(RMin,Minimum);
      child[i]=
      SelectInDiscreteSpace(temp,RMin,RMax,Precision,3);
```

```
    }
    else
    {
      temp=RNDfromCI(Maximum,RMax);
      child[i]=
      SelectInDiscreteSpace(temp,RMin,RMax,Precision,3);
    }
  }
}
//------------------------------------------------------------------------
-------

//------------------------------------------------------------------------
-------
//Заимствование генов.
void GenoMerging
(
double &child[]
)
{
  //----------------------Переменные-----------------------------
-------
  int  address=0;
  //-------------------------------------------------------------
-------
  for (int i=1;i<=GeneCount;i++)
  {
    //-----------------Отбор родителя-----------------------
    SelectOneParent(address);
    //------------------------------------------------------
    child[i]=Population[i][address];
  }
}
//------------------------------------------------------------------------
-------

//------------------------------------------------------------------------
-------
//Кроссинговер.
void CrossingOver
(
double &child[]
)
{
  //----------------------Переменные-----------------------------
-------
  int address_mama=0,address_papa=0;
  //-------------------------------------------------------------
-------
  //-----------------Отбор родителей-----------------------
  SelectTwoParents(address_mama,address_papa);
  //-------------------------------------------------------
  //Определим точку разрыва
  int address_of_gene=(int)MathFloor((GeneCount-
1)*(MathRand()/32767.5));

  for (int i=1;i<=GeneCount;i++)
```

```
  {
    //----копируем гены матери--------
    if (i<=address_of_gene+1)
      child[i]=Population[i][address_mama];
    //----копируем гены отца--------
    else
      child[i]=Population[i][address_papa];
  }
}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
//Отбор двух родителей.
void SelectTwoParents
(
int &address_mama,
int &address_papa
)
{
  //-----------------------Переменные-----------------------------
  int cnt=1;
  address_mama=0;//адрес материнской особи в популяции
  address_papa=0;//адрес отцовской особи в популяции
  //--------------------------------------------------------------
  //----------------------------Отбор родителей-------------------
  //Десять попыток выбрать разных родителей.
  while (cnt<=10)
  {
    //Для материнской особи
    address_mama=NaturalSelection();
    //Для отцовской особи
    address_papa=NaturalSelection();
    if (address_mama!=address_papa)
      break;
    cnt++;
  }
  //--------------------------------------------------------------
}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
//Отбор одного родителя.
void SelectOneParent
(
int &address//адрес родительской особи в популяции
)
{
  //-----------------------Переменные-----------------------------
```

```
  address=0;
  //-----------------------------------------------------------------
-------
  //----------------------------Отбор родителя----------------------
-----
  address=NaturalSelection();
  //-----------------------------------------------------------------
------
}
//+------------------------------------------------------------------+

//+------------------------------------------------------------------+
//Естественный отбор.
int NaturalSelection()
{
  //----------------------Переменные-------------------------------
-------
  int    i=0,u=0;
  double p=0.0,start=0.0;
  double          fit[][2];
  ArrayResize    (fit,PopulPersCount);
  ArrayInitialize(fit,0.0);
  double delta=(Population[0][0]-Population[0][PopulPersCount-
1])*0.01-Population[0][PopulPersCount-1];
  //-----------------------------------------------------------------
-------

  for (i=0;i<PopulPersCount;i++)
  {
    fit[i][0]=start;
    fit[i][1]=start+MathAbs(Population[0][i]+delta);
    start=fit[i][1];
  }
  p=RNDfromCI(fit[0][0],fit[PopulPersCount-1][1]);

  for (u=0;u<PopulPersCount;u++)
    if ((fit[u][0]<=p && p<fit[u][1]) || p==fit[u][1])
      break;

  return(u);
}
```