УДК 004.7

МНОГОПОТОЧНАЯ РЕАЛИЗАЦИЯ ГЕНЕТИЧЕСКОГО АЛГОРИТМА ДЛЯ МНОГОКРИТЕРИАЛЬНОЙ QOS-МАРШРУТИЗАЦИИ В ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЯХ

Д. А. Перепелкин, д.т.н., профессор кафедры САПР ВС РГРТУ, декан ФВТ РГРТУ, Рязань, Россия;
orcid.org/0000-0003-4775-5745, e-mail: perepelkin.d.a@rsreu.ru
А. Н. Сапрыкин, к.т.н., доцент кафедры САПР ВС РГРТУ, Рязань, Россия;
orcid.org/0000-0002-3882-1301, e-mail: saprykin.a.n@rsreu.ru
А. А. Тихонов, студент РГРТУ, Рязань, Россия;
orcid.org/0009-0006-7128-8795, e-mail: at2415216@yandex.ru

Рассматривается задача многокритериальной маршрутизации с обеспечением качества обслуживания (Quality of Service, QoS) в программно-конфигурируемых сетях (ПКС). ***Целью работы*** *являются разработка и оценка эффективности высокопроизводительного алгоритма поиска оптимальных маршрутов, удовлетворяющих совокупности разнородных метрик. В связи с NP-полнотой задачи многокритериальной оптимизации и ограниченной применимостью классических алгоритмов предложена многопоточная реализация генетического алгоритма, ориентированная на эффективное использование вычислительных ресурсов многопроцессорных систем. Для агрегации метрик используется аддитивный критерий с нормализацией и весовым распределением, учитывающим требования различных классов трафика. Проведено сравнительное тестирование с алгоритмами Дейкстры и Multi-Constrained Optimal Path (MCOP) на сетевых топологиях различного масштаба. Показано, что предложенная многопоточная модификация генетического алгоритма обеспечивает существенное ускорение по сравнению с MCOP и сохраняет приемлемое качество маршрутов, особенно в крупномасштабных сетях (500 – 1000 узлов), демонстрируя перспективность применения в современных сетевых инфраструктурах.*

Ключевые слова: *программно-конфигурируемые сети, генетический алгоритм, QoS маршрутизация, многопоточность, многокритериальная оптимизация.*

DOI: 10.21667/1995-4565-2025-94-68-84

Введение

Многокритериальная маршрутизация с обеспечением качества обслуживания (QoS) рассматривается исследователями как ключевой механизм для имплементации нововведений и улучшений в области сетевых технологий в рамках перспективных архитектур, включая программно-конфигурируемые сети (ПКС). Однако поиск подходящего маршрута, удовлетворяющего множественным критериям, представляет собой сложную задачу. Задачи маршрутизации с несколькими ограничениями относятся к классу NP-полных, а большинство существующих алгоритмов QoS-маршрутизации основаны на поддержании полной информации о состоянии сети в каждом узле. Централизованный контроль и глобальное представление о сети в архитектуре ПКС открывают новые возможности для решения этой задачи, однако в условиях современных требований к высокой пропускной способности и низкой задержке подобные подходы зачастую оказываются недостаточно эффективными из-за высокой вычислительной сложности. В связи с этим актуальной задачей является разработка алгоритмов, способных находить оптимальные маршруты, удовлетворяющие множественным критериям, за приемлемое время. Одним из перспективных направлений для этого является использование методов многокритериальной оптимизации, в частности генетических алгоритмов.

В рамках ПКС для повышения эффективности активно исследуются динамические подходы к маршрутизации на основе алгоритма Дейкстры, использующие актуальные

данные о нагрузке [1, 2]. Предварительные результаты таких исследований подтверждают эффективность динамического управления трафиком, однако также выявляют и присущие ему недостатки, такие как потенциальная нестабильность и колебания маршрутов. Помимо прочего, данный подход задействует однокритериальную маршрутизацию, что существенно ограничивает его применимость в средах, где качество обслуживания определяется совокупностью разнородных метрик, таких как задержка, джиттер, потери пакетов и пропускная способность.

В качестве альтернативы для решения задач маршрутизации в ПКС также предлагается применение и других алгоритмов, таких как эвристические и метаэвристические методы, отраженные в работах, посвященных использованию однопоточных реализаций эвристических алгоритмов [3-7]. Однако, как и в случае с классическим подходом, ключевой проблемой остается принципиальная ориентация на оптимизацию по единственному критерию, не позволяющая в полной мере учитывать комплексные требования современных приложений.

Таким образом, несмотря на наличие ряда перспективных подходов, задача разработки высокопроизводительного и устойчивого алгоритма многокритериальной маршрутизации остается актуальной.

В данной работе предлагается многопоточная модификация генетического алгоритма для решения задачи многокритериальной QoS-маршрутизации в программно-конфигурируемых сетях. Предложенная реализация обеспечивает эффективный поиск маршрутов, удовлетворяющих комплексным требованиям по задержке, потерям пакетов и пропускной способности, за счёт использования аддитивной агрегации нормализованных метрик с весовым распределением, адаптированным под тип трафика.

Анализ и постановка задачи

В настоящее время в ПКС алгоритмы маршрутизации с обеспечением качества обслуживания основаны на принципе использования единичного критерия либо же на дополнении данной парадигмы ограничениями на основе других метрик сети (MCOP). Таким образом, оптимизация проводится исключительно по одному целевому показателю качества сетевого обслуживания, в то время как остальные параметры учитываются лишь в качестве ограничений, задающих допустимые диапазоны значений. В качестве альтернативы такому подходу в [8] рассматриваются методы многокритериальной оптимизации, а именно три способа агрегирования критериев для задач данного класса: аддитивный, мультипликативный и минимаксный (максиминный).

Аддитивный критерий подразумевает сложение нормированных значений частных критериев, где нормированное значение представляет собой отношение частного критерия к некоторому нормирующему множителю, измеряющемуся в тех же единицах, что и сам критерий (чтобы в итоге величина стала безразмерной). Тогда целевая функция задачи оптимизации при применении данного критерия имеет вид:

$$F(X) = \sum_{i=1}^{n} c_i f_i(X), \tag{1}$$

где c – весовой коэффициент; а f – нормирующая функция для каждого частного критерия.

Мультипликативный критерий ориентирован на оптимизацию относительных, а не абсолютных изменений значений частных критериев. Результатом справедливой относительной компенсации является формула:

$$F(X) = \prod_{i=1}^{n} F_i^{c_i}(X), \tag{2}$$

где c – весовой коэффициент, равный 1, при условии, что все критерии равноценны.

Критерий максимина реализует стратегию гарантированного результата, которая формально состоит в выборе альтернативы, обеспечивающей максимальное значение наихудшего показателя среди всех частных критериев.

$$F(X^{(0)}) = \max_x \min_i \{f_i(X)\}, i = \overline{1,n}, X = (x_1, ..., x_m). \quad (3)$$

Ввиду особенностей основных метрик каналов в ПКС (пропускная способность [Мбит/с], задержка [мс], процент потери пакетов/PLR [%]) в качестве основы был выбран аддитивный критерий. Данный выбор обусловлен целым рядом преимуществ, а именно: возможностью тонкой настройки весовых коэффициентов для различных метрик, что критически важно для работы с разнородным трафиком (видео, файлы, аудио); использованием в расчётах нормированных значений, обеспечивающим большую надёжность по сравнению с мультипликативным методом; относительно низкой вычислительной сложностью, что позволяет снизить нагрузку на контроллер в сравнении с подходом, основанным на критерии максимина. Следует также отметить, что стратегия максимина, напротив, гарантирует выбор наилучшей альтернативы из набора наихудших, фокусируясь на оптимизации путей исключительно по их самым слабым критериям и практически игнорируя остальные показатели, что не соответствует поставленным задачам.

Определение весовых коэффициентов может быть выполнено методом ранжирования или экспертной оценки с назначением баллов в соответствии с операционными требованиями центра обработки данных или телекоммуникационного оператора. В рамках данного исследования предлагаются следующие нормализованные наборы весов для различных классов трафика: для голосового трафика VoIP – [0.5, 0.4, 0.1]; для видеотрафика – [0.4, 0.4, 0.2]; для файлового трафика – [0.7, 0.2, 0.1], где компоненты векторов соответствуют коэффициентам для задержки, потерь пакетов и пропускной способности соответственно.

Исходные метрики – задержка [мс] и процент потерь пакетов (PLR) [%] – естественным образом подлежат минимизации, в то время как пропускная способность [Мбит/с] требует максимизации. Это решается преобразованием метрики задержки в протоколе OSPF по следующей формуле:

$$OSPF\_bw = \frac{10^8}{bandwidth}, \quad (4)$$

однако с учетом того, что пропускная способность измеряется в Мбит/с, формула (4) принимает иной вид:

$$OSPF\_bw = \frac{10^2}{bandwidth}. \quad (5)$$

В качестве нормирующей функции для преобразования (1) предлагается следующий вариант: $f_i(X) = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$, где $x_{\min}, x_{\max}$ – минимальные и максимальные значения из множества X. Такая нормирующая функция приводит значения в любом диапазоне к числовому промежутку [0, 1], то есть все три метрики, если их не умножать на весовые коэффициенты, будут равнозначны при оценке во время поиска кратчайшего пути в сетевой топологии.

Таким образом, перед непосредственной работой алгоритма будет происходить предобработка метрик сети, формирующая по формуле (5) из матрицы задержек, а после агрегирующая по формуле (1), используя названные выше весовые коэффициенты, все матрицы метрик, сети в единый набор данных, который далее поступит на обработку в сетевой сервис, реализующий расчёт кратчайшего пути / дерева оптимальных маршрутов методом многопоточной модификации генетического алгоритма.

Анализ существующих решений показывает, что представленная в литературе реализация генетического алгоритма для балансировки нагрузки обладает существенным ограничением, связанным с синхронным (однопоточным) характером вычислений. В работе

[9] демонстрируется возможность значительного повышения производительности подобных алгоритмов за счёт распараллеливания вычислительных процессов. Параллельно с оптимизацией архитектуры для расширения функциональности алгоритма предлагается введение механизма сохранения наилучших особей каждого поколения. Формируемый таким образом «буфер» альтернативных решений служит двум целям: во-первых, обеспечивает возможность выбора из нескольких квазиоптимальных маршрутов, а во-вторых, способствует ускорению сходимости алгоритма за счёт сохранения перспективных генетических материалов.

Принимая во внимание ограничение Global Interpreter Lock (GIL) в интерпретаторе Python, на котором базируется контроллер Ryu, реализация вычислительно нагруженных алгоритмов сопряжена с существенными трудностями. В частности, задачи, зависимые от производительности процессора, при попытке параллельного выполнения в рамках стандартной конфигурации интерпретатора не получают реального ускорения, а в ряде случаев демонстрируют снижение производительности. В связи с этим для реализации многопоточной версии генетического алгоритма был выбран язык программирования Go, изначально разработанный для создания высокопроизводительных параллельных приложений [10]. Данный выбор позволяет обойти ограничения GIL и эффективно использовать ресурсы многопроцессорных систем.

Многопоточная модификация генетического алгоритма

Генетический алгоритм (ГА) представляет собой эвристический метод поиска, применяемый для решения задач оптимизации, в том числе для нахождения кратчайших маршрутов на графе. Алгоритм осуществляет направленный рандомизированный поиск путем формирования популяции потенциальных решений (особей), которые подвергаются операциям, имитирующим биологическую эволюцию: селекции, скрещиванию (кроссинговеру) и мутации. Данный метод относится к классу эволюционных вычислений, характеризующихся использованием принципов естественного отбора для последовательного улучшения качественных характеристик решений на протяжении нескольких поколений.

ГА предполагает случайное скрещивание и мутации существующих особей. Целевой направленностью эволюции является максимизация функции приспособленности (фитнес-функции), количественно выражающей оптимальность особи в контексте решаемой задачи. В рамках данной работы функция приспособленности будет рассчитываться как вес всего пути отдельного индивида: $F_{fit} = \sum_{start}^{end} edge_i$. Индивид в данном случае представляет собой отдельно взятый путь из начальной точки в конечную. Таким образом, значение функции приспособленности прямо пропорционально полезности особи: чем меньше совокупный вес пути, тем выше вероятность отбора данной особи для последующего размножения и передачи генетического материала следующим поколениям.

Для нахождения оптимального пути установка предельных значений для $F_{fit} \to 0$ нецелесообразна, в связи с этим ранняя остановка вычислений предполагается в том случае, если в течение некоторого количества поколений лучшая особь не будет сменяться прочими претендентами.

Также необходимо обеспечить выполнение ряда условий для корректной работы генетического алгоритма:

– каждая особь имеет валидный путь;

– после мутации начальный и конечный узлы графа остаются неизменны, а также не теряется связность пути;

– после скрещивания не появляется петель.

Укрупнённая блок-схема многопоточной модификации генетического алгоритма для нахождения кратчайшего пути на графе представлена на рисунке 1.

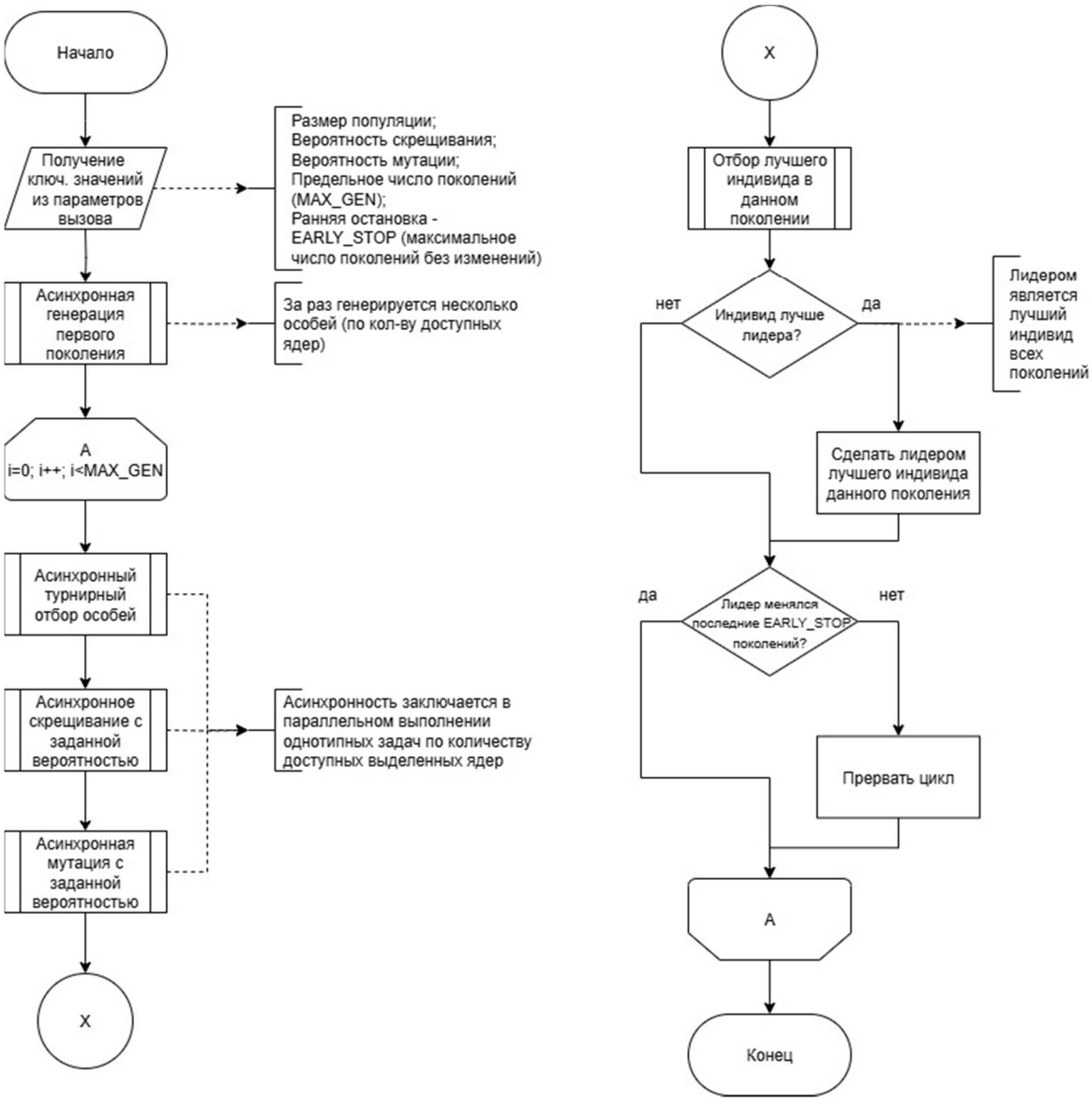

Рисунок 1 – Блок-схема многопоточного генетического алгоритма
Figure 1 – Flow chart of multithread genetic algorithm

Здесь же стоит отметить, что указанное выше приведение данных по формулам (5) и (1) будет производиться перед непосредственной работой алгоритма.

Скрещивание особей представляет собой «переброску» фрагментами пути. Пусть дано 2 хромосомы $chr_1 = [v_1, v_2, ..., v_7, v_N]; chr_2 = [v_1, v_5, ..., v_{11}, v_N]$. Выбирается некий общий узел их пути v_{common} так, что берётся четыре набора узлов:

$$chr_1^{'} = [v_1, v_2, ..., v_{common}]; chr_2^{'} = [v_{common}, ..., v_{11}, v_N];$$

$$chr_1^{''} = [v_{common}, ..., v_7, v_N]; chr_2^{''} = [v_1, v_5, ..., v_{common}]$$

и сливается в узле v_{common} в два новых цельных пути: $chr_{merged1} = [v_1, v_2, ..., v_{11}, v_N]; chr_{merged2} = [v_1, v_5, ..., v_7, v_N]$. В случае образования петли вида $[v_1, v_2, \boldsymbol{v_7}, v_5, v_9, \boldsymbol{v_7}, v_{10}]$ происходит удаление лишнего фрагмента пути, оставляя в итоге набор уникальных узлов: $[v_1, v_2, \boldsymbol{v_7}, v_{10}]$.

Мутация направлена на избежание «зацикливания» алгоритма в локальном оптимальном решении, в данной реализации в пути $chr = [v_1, v_2, ..., v_7, v_N]$ выбирается некий узел v_{mutate}, от которой перестраивается путь до конечной точки маршрута, например: $chr' = [v_1, v_2, \ldots, \boldsymbol{v_6}, v_{12}, v_N]$, где v_6 является точкой перестроения пути.

Турнирная селекция, в свою очередь, предполагает отбор лучшего кандидата для перехода в следующее поколение среди случайного набора индивидов из текущего.

Стоит отметить, что ввиду аппаратных ограничений вычислительных систем по количеству ядер параллельный запуск всех однотипных операций (генерация особей, кроссинговер, селекция и мутация) может привести к снижению производительности из-за переключения одного потока между задачами. Оптимальная стратегия предполагает распараллеливание вычислений с учетом конкретной конфигурации процессора, при котором количество одновременно выполняемых потоков соответствует числу физических или логических ядер системы.

Разработка программной среды для тестирования работы алгоритмов

Для подтверждения эффективности применения предложенной модификации генетического алгоритма в сравнении с классическими подходами, такими, как Дейкстра или MCOP, была спроектирована и разработана визуальная программная система *G_SDN* (рисунок 2 – 6), реализованная на базе фреймворка *Qt – C++*. Она позволяет проектировать карту топологии программно-конфигурируемой сети, открывать терминал контроллера *Ryu*, среды *mininet*, генерировать оболочку для вызова программы по расчёту генетического алгоритма с нужными параметрами, а также отображать результаты работы алгоритма, получая результат его выполнения от контроллера по сети (рисунок 7).

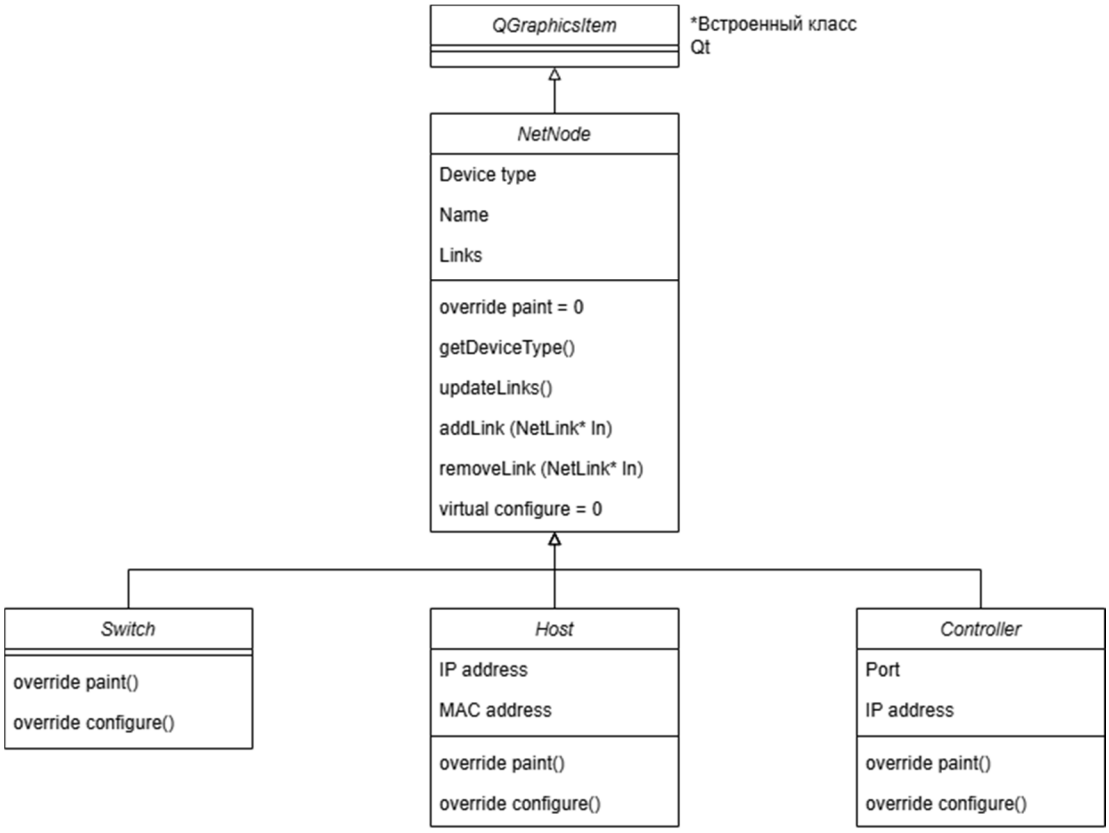

Рисунок 2 – Диаграмма классов для узлов топологии в G_SDN
Figure 2 – Class diagram of network map nodes in G_SDN

Поскольку узлы топологии сети могут представлять разнообразные устройства, был разработан абстрактный класс узла сети (*NetNode*) на базе стандартного класса *QGraphicsItem.* Данный класс предоставляет возможность удобной работы с графикой для отрисовки сетевой карты. Разработанный абстрактный класс реализует общий для всех устройств интерфейс по работе с каналами связи и общие вспомогательные поля и методы, необходимые для логики программы. Наследуемые классы, а именно хост (*Host*), OpenFlow-коммутатор (*Switch*), OpenFlow-контроллер (*Controller*), реализуют конкретные виды сетевых

узлов с соответствующими им интерфейсами настроек, реализующими работу с полями, используемыми в *mininet*.

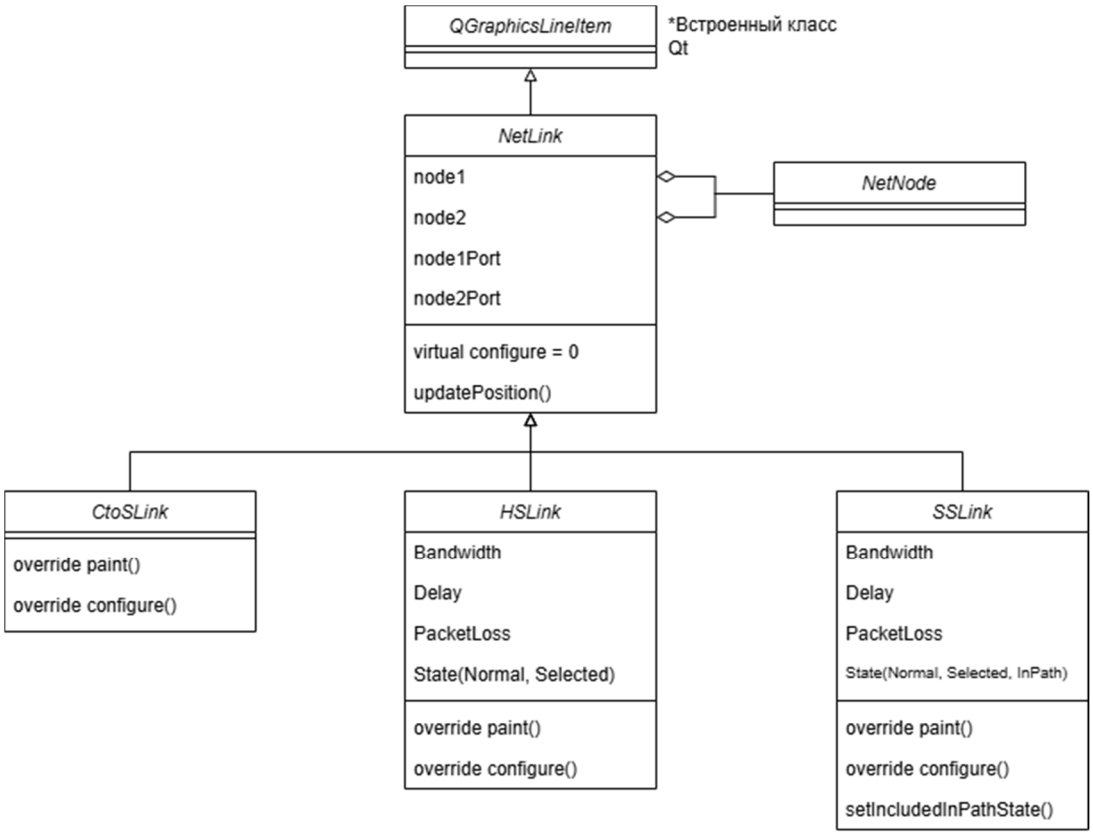

Рисунок 3 – Диаграмма классов для каналов связи топологии в G_SDN
Figure 3 – Class diagram for network map's links in G_SDN

Программно узлы сети представляют собой разные классы, что обусловило необходимость разработать классы разных каналов связи между устройствами: канал контроллер-коммутатор (*CtoSLink*), канал хост-коммутатор (*HSLink*), канал коммутатор-коммутатор (*SSLink*). Для реализации общего интерфейса для всех каналов был разработан абстрактный класс сетевого канала (*NetLink*), хранящий информацию о подключённых узлах сети и о портах подключения к ним. Основным родительским классом для отрисовки был выбран встроенный класс *QGraphicsLineItem*, предоставляющий удобный механизм работы с графикой в *Qt*. В *mininet* нет реализации контроллера, так как он, как и обычная сеть, эмулирует работу стандартной сети, и как следствие метрики каналов типа контроллер-коммутатор не задаются. Остальным классам каналов были добавлены статусы отрисовки в стандартном режиме, при наведении и при присвоении отметки «в пути», для отображения работы алгоритма поиска оптимального маршрута (рисунок 7). Для улучшения опыта использования было предусмотрено, что все коммутаторы по умолчанию подключены к контроллеру, даже если канал связи не был проставлен в редакторе топологии.

Для отрисовки сетевой карты был разработан класс *NetworkView*, реализующий механизмы подсчёта узлов разного типа для их наименования, функции отображения топологии, загруженной из собственного файла **.sdn*, а также обёрточные методы создания узлов сети, необходимые для передачи графическому объекту начальных координат. Для удобства работы с графикой родительским классом был выбран стандартный класс *QGraphicsView*, а для отображения – *QGraphicsScene*.

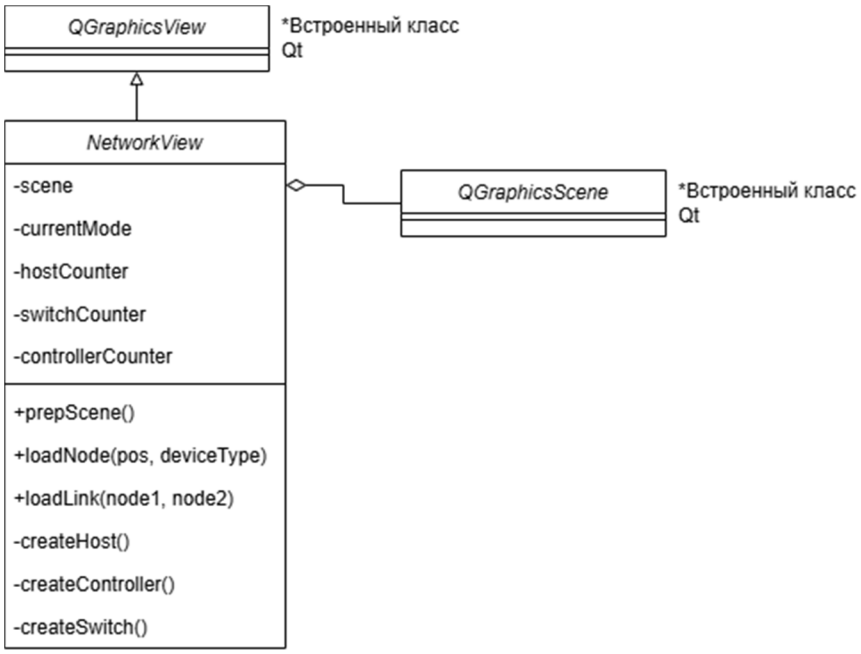

Рисунок 4 – Диаграмма классов для отрисовщика топологии в G_SDN
Figure 4 – Class diagram for the network map's painter in G_SDN

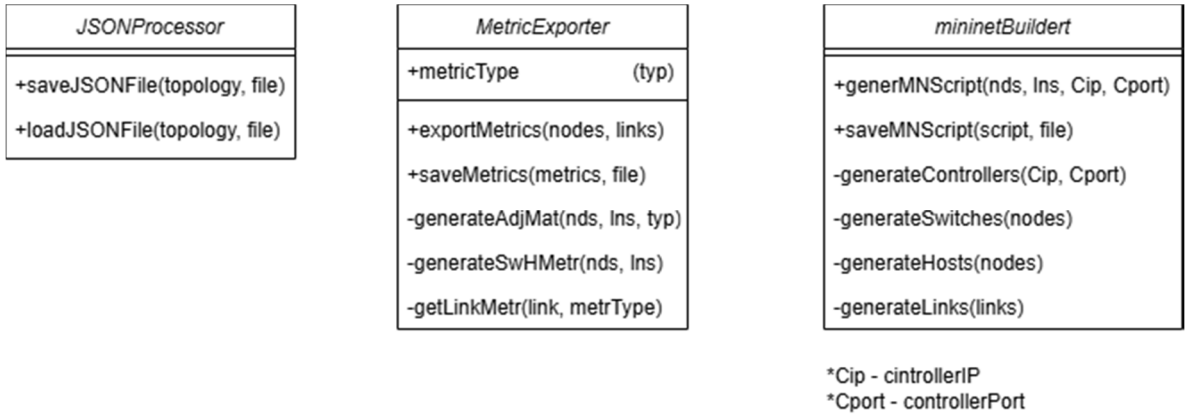

Рисунок 5 – Диаграмма классов для работы с файлами в G_SDN
Figure 5 – Class diagram for file processing in G_SDN

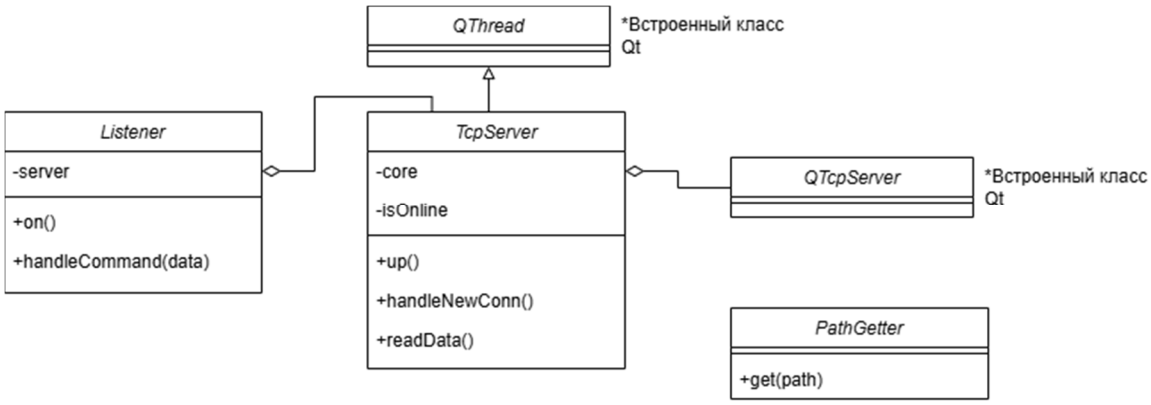

Рисунок 6 – Диаграмма классов для работы с сетью в G_SDN
Figure 6 – Class diagram for network communication in G_SDN

Для работы с файлами было разработано три класса: *JSONProcessor* – для сохранения и загрузки проекта **.sdn* топологии через интерфейс классов элементов сети; *MetricExporter* – для сохранения метрик каналов связи топологии в виде матрицы смежности (для метрик

коммутатор-коммутатор) и словаря «ключ-значение» (для метрик коммутатор-хост) в отдельный файл **.txt*, работающий с сетью через интерфейс абстрактного класса канала связи; *mininetBuilder* – для построения исполняемого скрипта *mininet* вида **.py*, также работающего с сетью через упомянутый интерфейс.

Для работы с сетью был создан класс *Listener*, необходимый для того, чтобы получать от контроллера *Ryu* информацию о найденном оптимальном маршруте и отображать его в графическом приложении. Он агрегирует в себе ещё один разработанный класс – *TcpServer*, используемый как прослойка между встроенным *QTcpServer* и программой. Для обеспечения асинхронности работы он наследуется от встроенного класса *QThread*. Для извлечения пути из полученных данных реализован класс со static-методом по получению маршрута из байтовой последовательности, поступающей на сервер от контроллера.

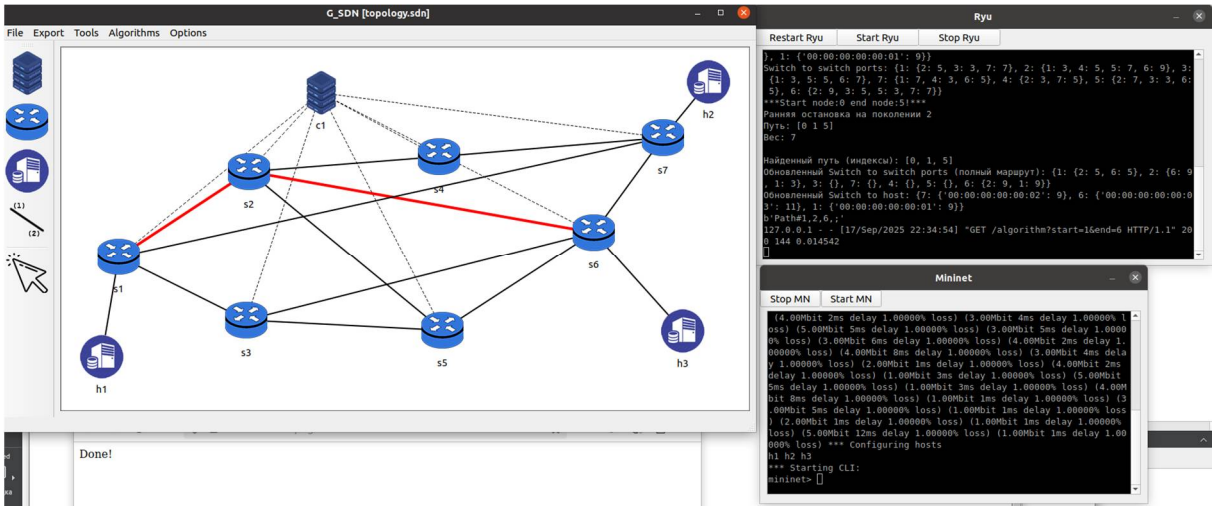

Рисунок 7 – Пример нахождения кратчайшего пути в сетевой топологии в среде G_SDN
Figure 7 – Example of finding the shortest path in network topology in G_SDN environment

Предложенная тестовая топология (рисунок 7) используется для проверки работоспособности (рисунок 8) и оценки эффективности разработанного алгоритма. Следует подчеркнуть, что в рамках данного исследования алгоритмы, ориентированные на двуметрическую оптимизацию (такие как LARAC и CSP), не включаются в сравнительный анализ. Это обусловлено их принципиальной ограниченностью работы с двумя метриками (например, задержка и процент потери пакетов), тогда как предложенный подход предназначен для многокритериальной оптимизации с поддержкой двух, трех и более метрик при сохранении вычислительной эффективности. Таким образом, для комплексной оценки эффективности предлагается проведение сравнительного тестирования в двух аспектах: во-первых, сравнение с алгоритмом MCOP, также ориентированным на многокритериальную оптимизацию, и во-вторых, сравнение с алгоритмом Дейкстры , применяемым к предварительно преобразованным данным с использованием формул (1) и (5) для приведения разнородных метрик к единой целевой функции.

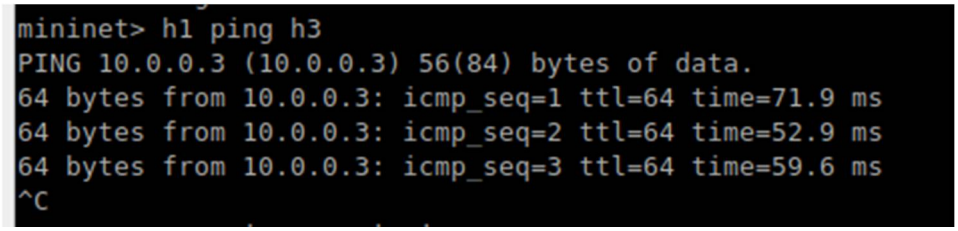

Рисунок 8 – Результат выполнения команды ping между хостами h1и h3
Figure 8 – Result of executing ping command between hosts h1 and h3

Для описания работы данной программной системы можно привести следующую схему (рисунок 9):

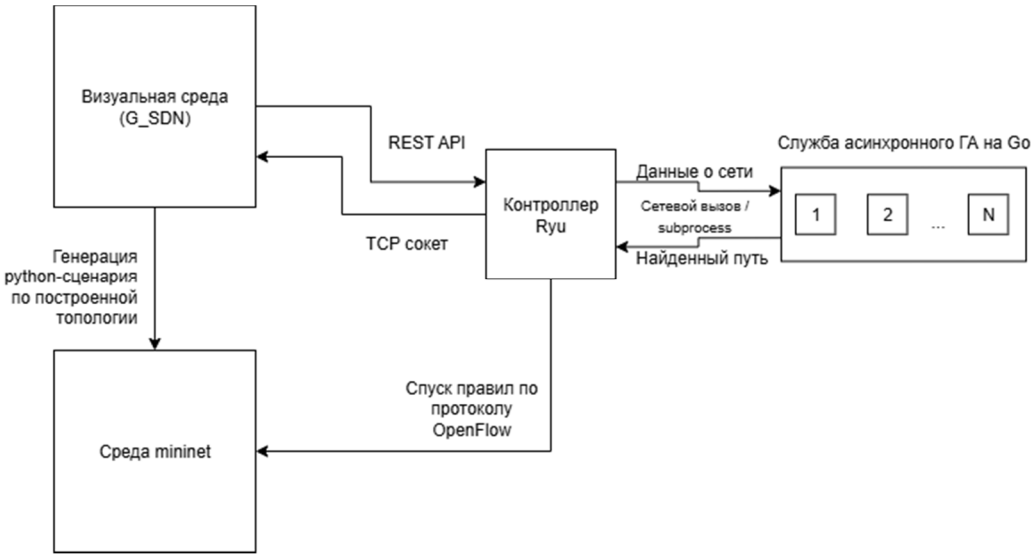

Рисунок 9 – Взаимодействие визуальной среды, контроллера Ryu и эмулятора mininet
Figure 9 – Communication of visual environment, Ryu controller and mininet emulator

Экспериментальное исследование

Сравнительный анализ эффективности алгоритмов выполнен на экспериментальной топологии (рисунок 10). На первом этапе исследована и оценена применимость разработанного алгоритма, после чего проведено нагрузочное тестирование. Результат работы модифицированного генетического алгоритма представлен на рисунке 11, а МСОР – на рисунке 12.

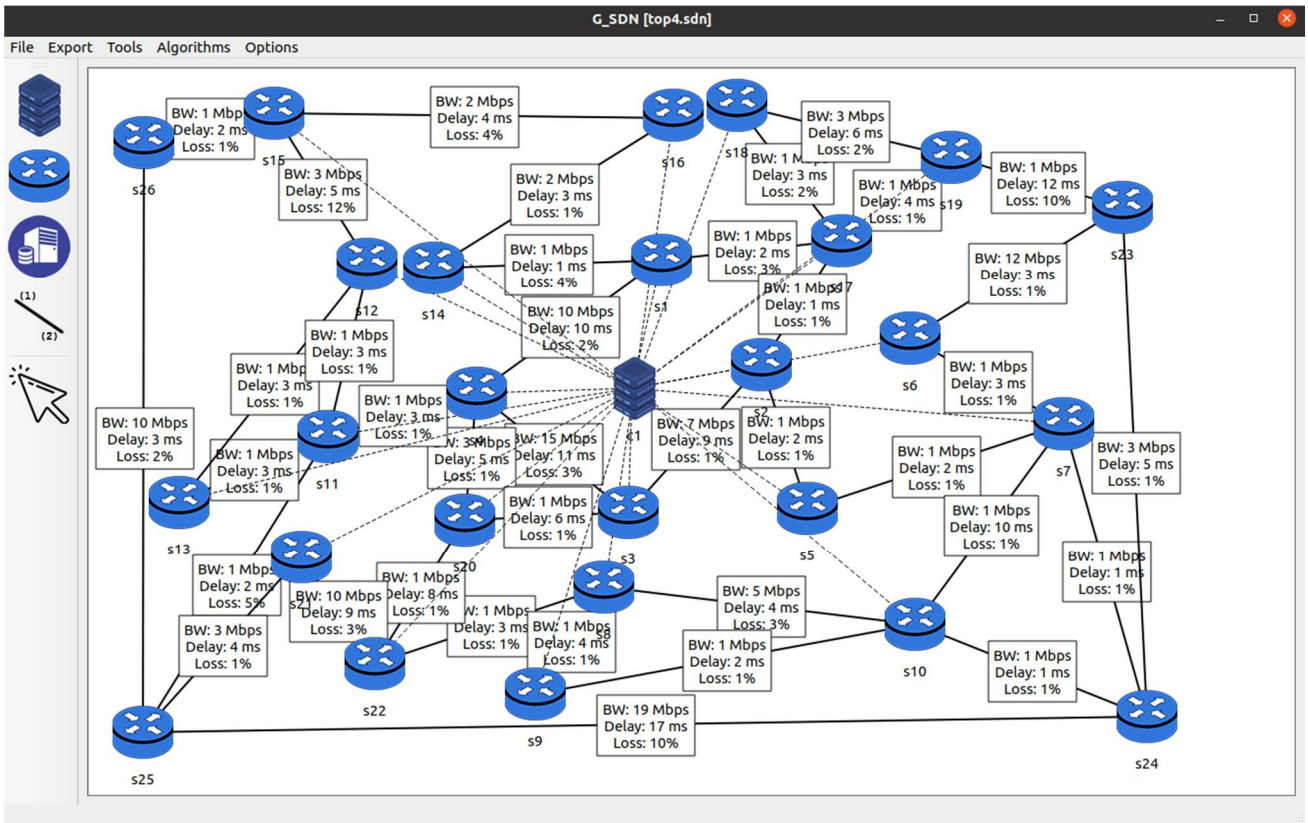

Рисунок 10 – Экспериментальная топология
Figure 10 – Experemental network map

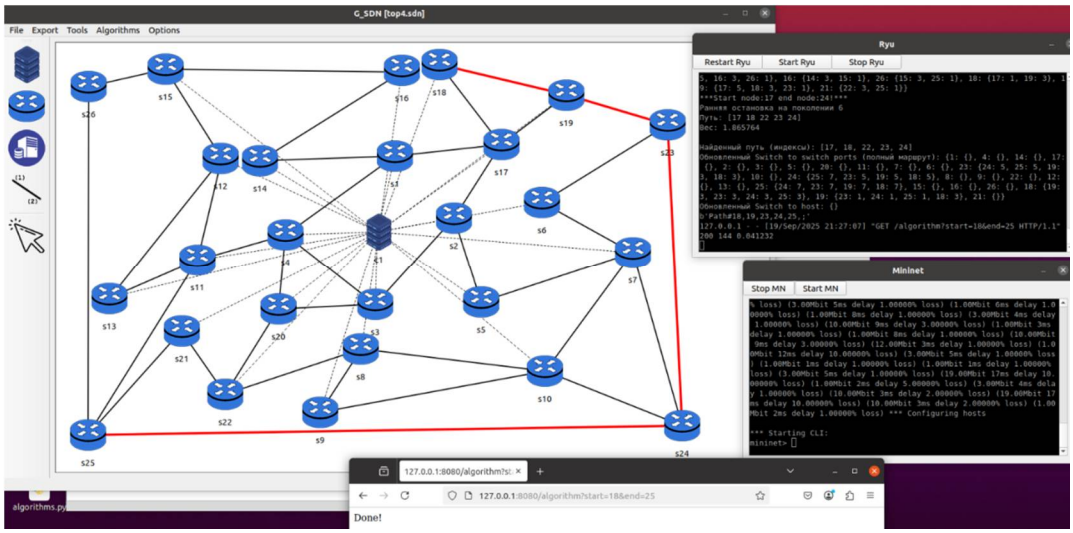

Рисунок 11 – Результат работы генетического алгоритма на экспериментальной топологии
Figure 11 – Result of genetic algorithm execution on experimental network map

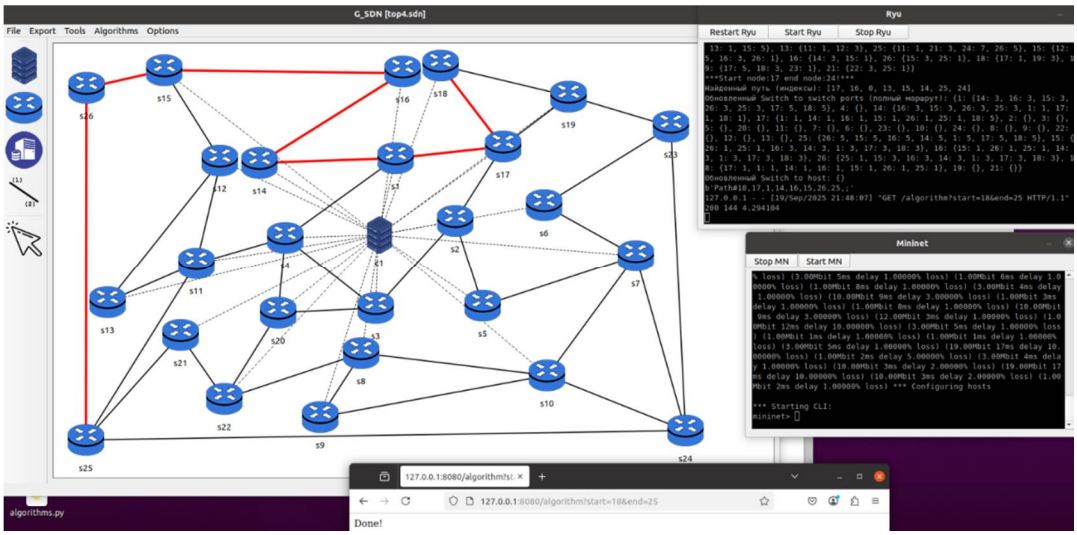

Рисунок 12 – Результат работы алгоритма MCOP на экспериментальной топологии
Figure 12 – Result of MCOP execution on experimental network map

Из пробного запуска видно, что модифицированный ГА нашёл путь со следующими характеристиками (пропускная способность, задержка, процент потерь): 1, 40, 23; в то же время характеристики пути, найденного через MCOP, составили 1, 18, 17.

В качестве целевого типа трафика был выбран файловый, соответственно ГА использовал при приведении метрик коэффициенты, названные выше, а MCOP оптимизировал, в первую очередь, пропускную способность. Путь, обнаруженный на рисунке 11, демонстрирует близкую к оптимальной, но не наилучшую эффективность по сравнению с решением, представленным на рисунке 12. Указанное отклонение обусловлено стохастической природой генетического алгоритма, который в рамках заданного числа итераций мог не сгенерировать глобально оптимальное решение вследствие ограниченности исходной популяции или недостаточной эффективности операторов кроссовера и мутации. Данная особенность может быть нивелирована путем корректировки параметров алгоритма, такой как увеличение размера популяции, повышение вероятности мутаций или расширение лимита генераций.

Однако проведенный эксперимент демонстрирует существенное превосходство генетического алгоритма по временным показателям: время работы составило 0.085 секунды против 6.4 секунды у алгоритма MCOP, что обеспечивает ускорение в 75 раз при сохранении

приемлемого качества решения. Для проведения комплексного сравнительного анализа дополнительно предлагается определить оптимальный маршрут с использованием алгоритма Дейкстры (рисунок 13), предварительно объединив матрицы тем же методом, что и при нахождении пути с помощью ГА. Заметим, что путь, найденный таким образом, соответствует пути с рисунка 11, а общее время его работы составило 0.003 секунды, что в 28 раз быстрее многопоточной реализации генетического алгоритма и в 2133 раз быстрее MCOP. Так, можно сделать два промежуточных вывода: нахождение менее оптимального пути связано с коэффициентами, выбранными для объединения матриц; использование классического алгоритма Дейкстры ожидаемо более выгодно для маленьких графов. Для проведения сравнительного анализа производительности алгоритмов маршрутизации исследование проводится на сетевых топологиях различного масштаба: 10, 50, 100, 250, 500, 750, 1000.

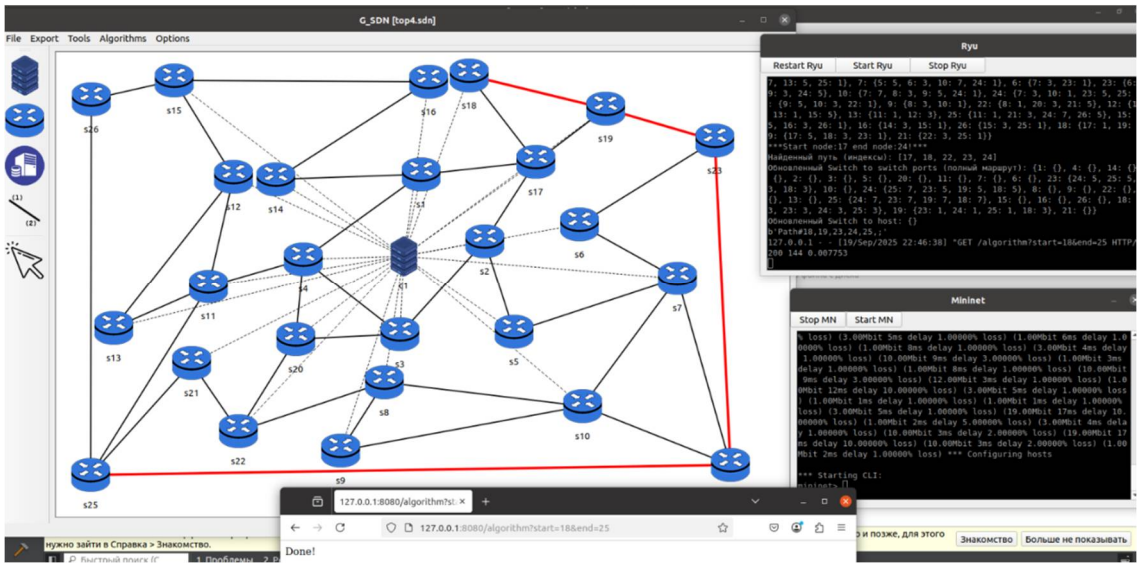

Рисунок 13 – Результат работы алгоритма Дейкстры на экспериментальной топологии
Figure 13 – Result of Dijkstra algorithm execution on experimental network map

Для начала отметим, что скорость работы генетического алгоритма прямо пропорциональна количеству генерируемых поколений. Так, например, решение задачи нахождения оптимального пути на экспериментальной топологии может быть получено за разное время, с разным шансом на достижение оптимального результата (таблица 1). При оценивании скорости работы алгоритмов время преобразования матрицы топологии не учитывалось, поскольку эта операция является общей для всех сравниваемых методов. Тестирование проводилось на основе матрицы задержек. Расчеты были выполнены с помощью программного обеспечения, описанного в работе [11].

Таблица 1 – Результаты работы генетического алгоритма с разными размерами популяций на экспериментальной топологии

Table 1 – Results of execution of genetic algorithm with different population sizes on experimental network map

Размер популяции	МО времени (мс)	Дисперсия времени	Количество запусков	МО длины пути	Дисперсия длины пути
10	1.29	0.06	10	24.25	61.94
24	1.46	0.12	10	19.8	29.96
50	3.6	0.74	10	18.2	26.36
100	9.94	3.4	10	17	16
200	44.38	132.68	10	13.8	5.76
400	180.33	12.18	10	13	0

Таким образом, можно сделать вывод о том, что время работы ГА в значительной степени зависит от размера популяций, которыми он оперирует в ходе работы. При этом размер популяции в 400 особей гарантирует 100 % сходимость к глобальному оптимуму на графе размерностью 25×25 узлов, тогда как для достижения сопоставимой точности с незначительным снижением вероятности нахождения оптимального пути достаточно размера популяции в 200 особей. Однако при решении задач для крупномасштабных топологий (1000 узлов и более) стратегия установки лимита поколений, кратного количеству узлов, становится неоптимальной. Несмотря на сохраняющееся преимущество в скорости перед алгоритмом MCOP, такое решение может уступать в эффективности алгоритму Дейкстры, применяемому к предварительно преобразованным матрицам смежности. Это обусловлено тем, что с ростом размера графа требуемое количество поколений для гарантированной сходимости генетического алгоритма существенно увеличивает вычислительную нагрузку, снижая тем самым первоначальное временное преимущество. Для проведения сравнительного анализа производительности алгоритмов в условиях крупномасштабных сетей было выполнено дополнительное исследование, количественные результаты которого представлены в таблице 2. Стоит учитывать, что работа генетического алгоритма зависит от правильного подбора коэффициентов [12]. В рамках экспериментальной проверки установлены следующие параметры генетического алгоритма: вероятность скрещивания – 85 %, вероятность мутации – 15 %, размер популяции – 50 особей для топологий до 250 узлов и 200 особей для крупных сетей (свыше 250 узлов). Влияние указанных параметров на производительность алгоритма отражено в таблице 1. При сравнительном анализе времени выполнения генетического алгоритма и алгоритма Дейкстры учитывается также продолжительность этапа преобразования метрик.

Таблица 2 – Исследование эффективности работы рассматриваемых алгоритмов в сетях разного размера

Table 2 – Research of considered algorithms efficiency in networks of different sizes

Размер топологии (кол-во узлов)	ГА (мат. ожидание длины пути)	ГА (мат. ожидание времени, мс)	Дейкстра (длина пути)	Дейкстра (мат. ожидание времени, мс)	MCOP (мат. ожидание времени, мс)
10	22	1.73	22	0.1	47.83
50	86.8	8.61	36	4.2	52109.87
100	77.7	24.75	12	18.65	-
250	24.2	314.72	7	154.79	-
500	28.5	915.25	5	838.42	-
750	27.7	1604.91	3	3422.34	-
1000	58.2	2497.09	7	5181.34	-

Следует отметить, что использование модуля маршрутизации, реализованного на Go, в составе контроллера Ryu, написанного на Python, создает дополнительные временные затраты, обусловленные межпроцессным взаимодействием и преобразованием данных между различными программными средами. Если использовать библиотеку *subprocess*, и хранить исполняемый файл генетического алгоритма на сервере с *Ryu*, то задержка на вызов будет составлять порядка 0.3 – 0.4 секунд (рисунок 14). Если же установить расчётный модуль как сетевую службу, то накладные расходы будут зависеть от того, где он находится.

Таким образом, полученные результаты (таблица 2, рисунок 15) формируют комплексное представление о времени работы рассматриваемых алгоритмов. Согласно полученным экспериментальным данным, можно заключить, что в рамках решаемой задачи многокритериальной оптимизации применение генетического алгоритма оправдано только при больших топологиях (500+ узлов), что вполне рационально для современных ПКС-контроллеров, которые уже поддерживают подключение до 1000 узлов. Более классический и надежный алгоритм MCOP при работе с тремя метриками показывает недопустимо

большое время работы уже на топологии, содержащей 50 узлов, обрабатывая ее медленнее, чем ГА, или, чем алгоритм Дейкстры обрабатывает топологию из 1000 узлов.

```
Поколение 69: 26 [0 914 231 7]
Поколение 69: 26 [0 914 231 7]
Поколение 70: 26 [0 914 231 7]
Поколение 71: 26 [0 914 231 7]
Время выполнения: 0.6795 секунд


Return code: 0
Return code: 0
1.0073151588439941
```

Рисунок 14 – Задержка вызова генетического алгоритма через подпроцесс
Figure 14 – Delay of executing genetic algorithm via subprocess

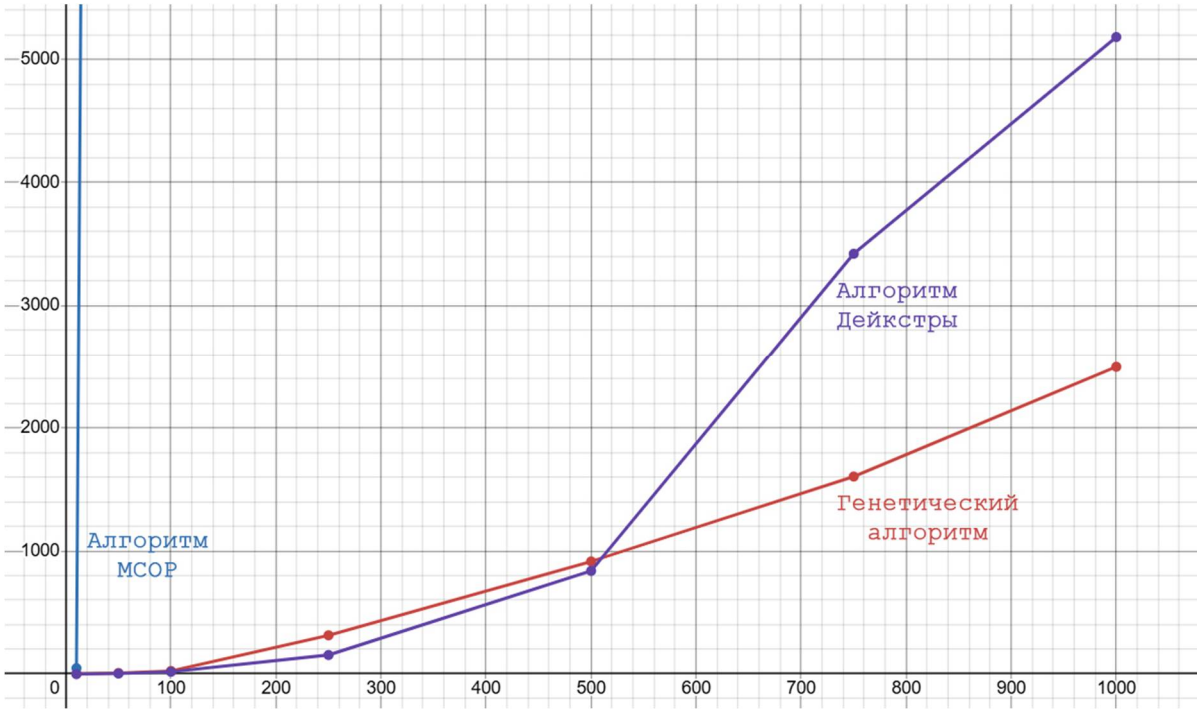

Рисунок 15 – Сравнение времени работы рассматриваемых алгоритмов
Figure 15 – Comparison of considered algorithms operating time

Следует также отметить, что, как показано в исследованиях [4, 12], производительность генетического алгоритма может быть существенно улучшена как в аспекте качества решения, так и в отношении времени вычислений путем оптимизации параметров его работы. Представленные в работе результаты могут быть дополнительно улучшены при проведении целенаправленного параметрического анализа и тонкой настройки коэффициентов алгоритма.

Несмотря на современное ограничение в 1000 узлов на один контроллер ПКС, быстрое развитие данного направления позволяет прогнозировать увеличение масштабируемости сетей. В этом контексте выявленная тенденция генетического алгоритма к более плавному росту времени выполнения с увеличением размера сети может стать определяющим фактором при выборе метода поиска оптимальных маршрутов в перспективных крупномасштабных сетевых инфраструктурах. То же можно сказать и про представленную методику расчёта метрик для оптимизации сразу нескольких параметров найденного маршрута, так как её применение позволяет находить удовлетворительные пути за

приемлемое количество времени, а преимущество данного подхода показано на примере сравнения времени работы генетического алгоритма и алгоритма Дейкстры с предобработкой данных сети с MCOP.

Заключение

Предложено решение задачи многокритериальной QoS-маршрутизации в программно-конфигурируемых сетях на основе модифицированного генетического алгоритма с многопоточной реализацией. Разработанный подход обеспечивает эффективный поиск оптимальных маршрутов с учётом совокупности ключевых метрик качества обслуживания – задержки, потерь пакетов и пропускной способности – за счет применения аддитивного метода агрегации нормализованных показателей с весовыми коэффициентами, адаптированными под различные классы трафика (голосовой, видео, файловый). Внедрение механизма сохранения лучших особей каждого поколения способствует ускорению сходимости алгоритма и повышению стабильности решений.

Экспериментальное исследование показало, что предложенная многопоточная реализация генетического алгоритма демонстрирует значительное преимущество по времени выполнения по сравнению с алгоритмом MCOP. По сравнению с алгоритмом Дейкстры, примененным к преобразованным метрикам, генетический алгоритм уступает в скорости на малых топологиях, однако его временная сложность растет более плавно, что делает его перспективным для использования в крупномасштабных ПКС-инфраструктурах.

Таким образом, предложенный подход сочетает высокую производительность, масштабируемость и гибкость в условиях многокритериальной оптимизации, что делает его перспективным инструментом для реализации эффективной QoS-маршрутизации в современных и будущих программируемых сетях.

Библиографический список

1. **Grbac T.G, Domazet N.** On the Applications of Dijkstra’s Shortest Path Algorithm in Software Defined Networks // Intelligent Distributed Computing XI. 2018. С. 39-45.

2. **Tirupathi V., Sagar K.** A SDN-Based Load Balancing Algorithm for IoT Traffic Data and Network Performance Evaluation // SSRG International Journal of Electronics and Communication Engineering.Vol. 10, Issue 8. 2023. С. 108-117.

3. **Перепелкин Д.А., Нгуен В.Т.** Исследование процессов балансировки нагрузки в программно-конфигурируемых сетях на основе генетического алгоритма // Вестник Рязанского государственного радиотехнического университета. 2021. № 77. С. 43-57.

4. **Перепелкин Д.А., Нгуен В.Т.** Исследование и анализ процессов многопутевой маршрутизации и балансировки потоков данных в программно-конфигурируемых сетях на основе генетического алгоритма // Вестник Рязанского государственного радиотехнического университета. 2022. № 79. С. 31-48.

5. **Перепелкин Д.А., Иванчикова М.А., Нгуен В.Т.** Интеллектуальная многопутевая маршрутизация в программно-конфигурируемых сетях на основе алгоритма миграции стаи птиц // Вестник Рязанского государственного радиотехнического университета. 2022. № 82. С. 44-59.

6. **Yilan Liu, Yun Pan, Muxi Yang, Wenqing Wang, Chi Fang, Ruijuan Jiang**. The Multi-Path Routing Problem in the Software Defined Network, in 11th International Conference on Natural Computation (ICNC). 2015. DOI: 10.1109/ICNC.2015.7377999.

7. **Корячко В.П., Перепелкин Д.А.** Программно-конфигурируемые сети. Учебник для вузов. М.: Горячая линия – Телеком, 2020. 288 с.

8. **Корячко В.П., Курейчик В.М., Норенков И.П.** Теоретические основы САПР. Энергоатомиздат, 1987. 400 с.

9. **Скворцов С.В., Дьяков М.С.** Ускорение генетического алгоритма решения транспортной задачи средствами многопоточного программирования // Вестник Рязанского государственного радиотехнического университета. 2023. № 84. С. 99-107.

10. A Tour of Go [Электронный ресурс] URL: https://go.dev/tour/concurrency/1 (дата обращения: 15.09.2025).

11. **Тихонов А.А.** Разработка программного модуля вычислений и визуализации результатов при изучении отдельных тем математической статистики // IT Open 2024 Материалы IV регионального конкурса студенческих научно-исследовательских работ в области информационных и вычислительных технологий. 2025. С. 168-177.

12. **Федоренко К.В., Оловянников А.Л.** Исследование основных параметров генетического алгоритма применительно к задаче поиска оптимального маршрута // Вестник государственного университета морского и речного флота им. адмирала С.О. Макарова. 2017. Том 9. № 4. С. 714-723.

UDC 004.7

MULTITHREADED IMPLEMENTATION OF GENETIC ALGORITHM FOR MULTI-OBJECTIVE QOS ROUTING IN SOFTWARE-DEFINED NETWORKS

D. A. Perepelkin, Dr. in technical sciences, Professor, CAD Department, Dean of Computer Engineering Faculty, RSREU, Ryazan, Russia;
orcid.org/0000-0003-4775-5745, e-mail: dmitryperepelkin@mail.ru
A. N. Saprykin, Ph.D. (in technical sciences), associate professor, CAD Department, RSREU, Ryazan, Russia;
orcid.org/0000-0002-3882-1301, e-mail: saprykin.a.n@rsreu.ru
A. A. Tikhonov, student, RSREU, Ryazan, Russia;
orcid.org/0009-0006-7128-8795, e-mail: at2415216@yandex.ru

The problem of multi-objective Quality of Service (QoS) routing in software-defined networks (SDN) is considered. The aim of the work is to develop and evaluate the performance of a high-performance algorithm for finding optimal routes that satisfy a set of heterogeneous metrics. Due to NP-completeness of multi-objective optimization problem and limited applicability of classical algorithms, multithreaded implementation of genetic algorithm is proposed, focused on the efficient use of multiprocessor systems computational resources. An additive criterion with normalization and weight distribution considering the requirements of different traffic classes is used for metric aggregation. A comparative testing with Dijkstra's and Multi-Constrained Optimal Path (MCOP) algorithms on network topologies of various scales is conducted. The proposed multithreaded modification of genetic algorithm is shown to provide a significant speedup compared to MCOP and to maintain acceptable route quality, especially in large-scale networks (500 – 1000 nodes), demonstrating prospects for application in modern network infrastructures.

Keywords*: software-defined networks, genetic algorithm, QoS routing, multithreading, multi-objective optimization.*

DOI: 10.21667/1995-4565-2025-94-68-84

References

1. **Grbac T.G, Domazet N.** On the Applications of Dijkstra’s Shortest Path Algorithm in Software Defined Networks. *Intelligent Distributed Computing XI.* 2018, pp. 39-45.

2. **Tirupathi V., Sagar K.A** SDN-Based Load Balancing Algorithm for IoT Traffic Data and Network Performance Evaluation. *SSRG International Journal of Electronics and Communication Engineering,* 2023, vol. 10, iss. 8, pp. 108-117.

3. **Perepelkin D.A., Nguyen V.T.** Issledovanie processov balansirovki nagruzki v programmno-konfiguriruemyh setyah na osnove geneticheskogo algoritma. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta*. 2021, no. 77, pp. 43-57. (in Russia).

4. **Perepelkin D.A., Nguyen V.T.** Issledovanie i analiz processov mnogoputevoj marshrutizacii i balansirovki potokov dannyh v programmno-konfiguriruemyh setyah na osnove geneticheskogo algoritma. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta*. 2022, no. 79, pp. 31-48. (in Russia).

5. **Perepelkin D.A., Ivanchikova M.A., Nguyen V.T.** Intellektual'naya mnogoputevaya marshrutizaciya v programmno-konfiguriruemyh setyah na osnove algoritma migracii stai ptic. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta*. 2022, no. 82, pp. 44-59. (in Russia).

6. **Yilan Liu, Yun Pan, Muxi Yang, Wenqing Wang, Chi Fang, Ruijuan Jiang.** The Multi-Path Routing Problem in the Software Defined Network. *11th International Conference on Natural Computation (ICNC)*. 2015. DOI: 10.1109/ICNC.2015.7377999.

7. **Koryachko V.P., Perepelkin D.A.** Programmno-konfiguriruemye seti. Uchebnik dlya vuzov. M.: *Goryachaya liniya – Telekom*. 2020. 288 p.

8. **Koryachko V.P., Kurejchik V.M., Norenkov I.P**. Teoreticheskie osnovy SAPR. *Energoatomizdat*. 1987. 400 p.

9. **Skvortsov S.V., D'yakov M.S.** Uskorenie geneticheskogo algoritma resheniya transportnoj zadachi sredstvami mnogopotochnogo programmirovaniya. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta*. 2023, no. 84, pp. 99-107. (in Russia).

10. A Tour of Go. [*Elektronnyj resurs*] URL: https://go.dev/tour/concurrency/1 (data obrashcheniya: 15.09.2025).

11. **Tikhonov A.A.** Razrabotka programmnogo modulya vychislenij i vizualizacii rezul'tatov pri izuchenii otdel'nyh tem matematicheskoj statistiki. **IT Open 2024 Materialy IV regional'nogo konkursa studencheskih nauchno-issledovatel'skih rabot v oblasti informacionnyh i vychislitel'nyh tekhnologij**. 2025, pp. 168-177.

12. **Federenko** K**.V., Olovyannikov A.L.** Issledovanie osnovnyh parametrov geneticheskogo algoritma primenitel'no k zadache poiska optimal'nogo marshruta. *Vestnik gosudarstvennogo universiteta morskogo i rechnogo flota im. admirala S.O. Makarova*. 2017, vol. 9, no. 4, pp. 714-723. (in Russia).