МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И КОМПЬЮТЕРНЫХ СЕТЕЙ

УДК 004.72

ИССЛЕДОВАНИЕ ПРОЦЕССОВ БАЛАНСИРОВКИ НАГРУЗКИ В ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЯХ НА ОСНОВЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

Д. А. Перепелкин, д.т.н., доцент, декан ФВТ РГРТУ, Рязань, Россия;
orcid.org/0000-0003-4775-5745, e-mail: dmitryperepelkin@mail.ru
В. Т. Нгуен, магистрант РГРТУ, Рязань, Россия;
orcid.org/0000-0003-2930-5775, e-mail: nguyenvantinrsreu@gmail.com

Программно-конфигурируемые сети (ПКС), как новая сетевая парадигма, позволяют справиться с ограничениями традиционных сетей. ПКС используют управляющий контроллер, имеющий глобальное представление о сети и коммутационных устройствах, которые действуют как оборудование для пересылки пакетов, известные как «коммутаторы OpenFlow». Ограничения сетевых ресурсов и обеспечение требований качества сервиса приводят к важной потребности в балансировке нагрузки между коммутаторами ПКС. ***Цель работы*** *– исследование и анализ процессов балансировки нагрузки в ПКС на основе генетического алгоритма. Для подтверждения эффективности и правильности работы генетического алгоритма в ПКС разработано программное обеспечение моделирования процессов балансировки нагрузки. Результаты моделирования подтвердили эффективность работы генетического алгоритма в ПКС для балансировки и перераспределения сетевого трафика с целью обеспечения требуемого качества сервиса и уменьшения перегрузки в сети.*

Ключевые слова: *программно-конфигурируемые сети, балансировка нагрузки, генетический алгоритм.*

DOI: 10.21667/1995-4565-2021-77-43-57

Введение

Программно-конфигурируемые сети (ПКС) отделяют плоскость управления сетью от плоскости пересылки данных. ПКС продемонстрируют значительные преимущества во многих отношениях по сравнению с традиционными сетями. Однако распределение трафика в ПКС влияет на эффективность и порождает множество других проблем. Например, неравномерное распределение нагрузки в ПКС существенно влияет на производительность сети. Следовательно, было введено несколько методов балансировки нагрузки (БН) для повышения эффективности ПКС. В статье предлагается генетический алгоритм (ГА) для определения кратчайших путей между двумя узлами для равномерного распределения нагрузки в сети.

Программно-конфигурируемые сети (ПКС)

В настоящее время, с быстрым развитием сети, масштабы сети и приложений резко увеличиваются, что приводит к все более и более сложной структуре и функциям. Сеть, основанная на традиционной архитектуре TCP/IP, сталкивается с множеством проблем, особенно с маршрутизатором как ядром сети. Достоверность и эффективность пересылки данных сталкиваются с серьезными проблемами. Для решения этих проблем широкую популярность получила новая парадигма компьютерных сетей – программно-конфигурируемые сети.

ПКС – это новый вид сетевой архитектуры, в которой пересылка и управление разделены и могут управляться напрямую с помощью сетевых приложений. В существующей сети управление трафиком и его пересылка зависят от сетевого оборудования (например, коммутатора, маршрутизатора). ПКС отделяет функцию управления от сетевого оборудования и реализует централизованное управление сетью и программируемость приложений путем стандартизации. Поэтому ПКС полностью меняет традиционную сетевую архитектуру.

Архитектура ПКС состоит из трех уровней:

– плоскость данных: нижний уровень включает в себя устройства пересылки пакетов, такие как Ethernet, оптические и виртуальные коммутаторы, маршрутизаторы и точки доступа;

– плоскость управления: средний уровень включает контроллеры и сетевые службы. Контроллеры несут ответственность за маршрутизацию и принятие решения о пересылке трафика в соответствующие пункты назначения;

– плоскость приложения: самый верхний слой помещается поверх центрального слоя. Этот уровень использует средства программирования высокого уровня для обеспечения некоторых функций, таких как мониторинг безопасности, энергосберегающие сети, управление трафиком и т.д.

Традиционные коммутаторы/маршрутизаторы обновляют таблицы маршрутизации и содержат только информацию о пункте назначения, а маршрутизаторы могут подсчитывать информацию о следующем переходе. Однако контроллер ПКС имеет полный обзор сети. Обновляя информацию о топологии, контроллер ПКС может находить все пути между отдельными источниками к их узлам назначения и их статус загрузки. Сетевая архитектура для системы балансировки нагрузки представлена на рисунке 1.

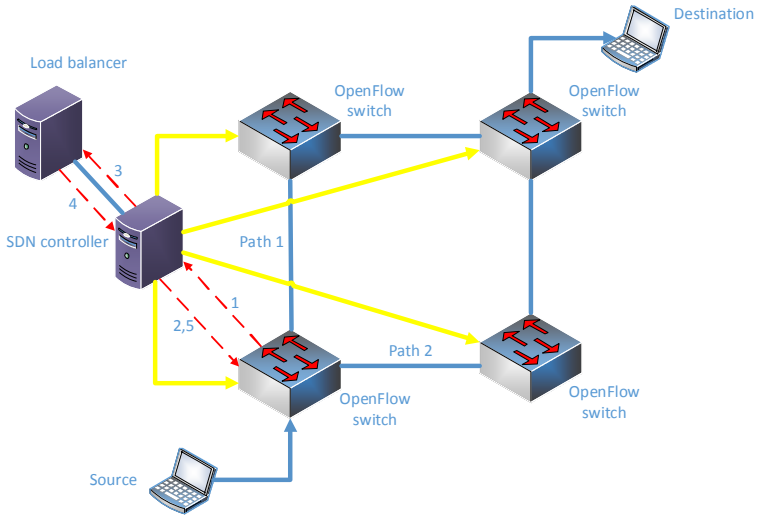

Рисунок 1 – Сетевая архитектура для системы балансировки нагрузки ПКС
Figure 1 – Network architecture for SDN load balancer

Теоретические исследования

Предотвращение перегрузки в сети и балансировка нагрузки

– В работах [1-5] на основе алгоритма парных переходов предложены эффективные алгоритмы адаптивной ускоренной маршрутизации, которые также позволили снизить трудоемкость построения оптимальных маршрутов передачи данных.

– Liu в [6] представляет ГА для решения задачи оптимизации многопутевого распространения с ограниченной полосой пропускания в ПКС. Результаты показывают, что предложенный ГА может глобально, гибко и эффективно определять путь маршрутизации, который минимизирует сетевую задержку при ограничении полосы пропускания в многопутевой ПКС.

– Ren в [7] разрабатывает ГА для решения проблемы планирования трафика при управлении перегрузкой коммутатора. Он производит выборку коэффициента использования ка-

нала и передает его в ГА, чтобы оптимизировать назначение трафика коммутатора, чтобы избежать перегрузки канала. Результаты показывают, что предложенный ГА делает использование компоновки каналов более сбалансированным и разумным.

– Maniu в [8] применяет ГА для вычисления пути маршрутизации для сетевых потоков, так что динамическое распределение ресурсов может быть оптимизировано для создания самоадаптивной сети с исторической экстраполяцией. Результаты показывают, что предложенный ГА обеспечивает приблизительный оптимальный путь маршрутизации для снижения стоимости канала эффективно.

QoS (Quality of Service) и QoE (Quality of Experience)

– В работах [9-11] предложены подходы динамической балансировки нагрузки в ПКС с QoS. Основное внимание в модели сосредоточено на заданной прокладке оптимальных и резервных маршрутов по заданному показателю QoS, балансировке нагрузки по проложенным маршрутам на основе алгоритма парных переходов, оптимизации и выравниванию нагрузки по индикаторам вариации маршрута и величине джиттера.

– Balaguer в [12] применяет ГА, чтобы найти путь маршрутизации для потока от источника к целевому узлу в ПКС, чтобы максимизировать параллельные потоки без ухудшения QoE. Поскольку мультимедийные потоки должны удовлетворять определенным требованиям к максимальной задержке и минимальной пропускной способности, поиск оптимального пути маршрутизации, который удовлетворяет ограничениям, требует времени, особенно в крупномасштабной сети. ГА предлагает приемлемое приблизительное оптимальное решение с коротким временем сходимости для крупномасштабной сети.

Балансировка нагрузки контроллера и канала

– Kang в [13] предлагают стратегии балансировки нагрузки, которые используют способ мониторинга нагрузки каждого контроллера. Как только обнаруживается дисбаланс нагрузки контроллера, применяется ГА для генерации нового назначения переключения для балансировки нагрузки контроллера.

– Ruelas в [14] также применяет ГА для балансировки нагрузки каждого контроллера в соответствии с набором рабочей нагрузки. По производительности предлагаемый ГА превосходит методы Round-Robin и случайный.

Размещение контроллера

– Sanner в [15] использует ГА для определения расположения контроллера, чтобы минимизировать задержку переключения на контроллер. Он кодирует размещение контроллера как хромосому, но метод кодирования четко не указан. Он оптимизирует только одну цель, и производительность предлагаемого ГА хороша по сравнению с алгоритмом целочисленного линейного программирования.

Задача нахождения кратчайших путей (КП) между двумя заданными узлами

Пусть $G=(V,E)$ – взвешенный граф, где V обозначает множество узлов, а $E \subseteq V \times V$ – множество ребер (дуг или звеньев). Пусть $v_s, v_d \in V(G)$ – исходный и целевой узлы соответственно, а p_i – соответствующий путь без петель от v_s к v_d с весом $w(p_i)$. Предположим, что существует всего n различных путей от v_s до v_d. Пусть $R_n(v_s, v_d)$ представляет набор путей, состоящий из указанных выше n путей с возрастающим порядком сортировки весов, т.е.

$$R_n(v_s, v_d) = \{p_1, p_2, \ldots, p_n | w(p_1) \le w(p_2) \le \ldots \le w(p_n)\}.$$

Тогда традиционная задача поиска кратчайших путей без петель может быть определена для подмножества R_k следующим образом:

$$R_k(v_s, v_d) = \{p_1, p_2, \ldots, p_k | w(p_1) \le w(p_2) \le \ldots \le w(p_k)\}.$$

В практических приложениях может быть несколько путей с одинаковыми весовыми коэффициентами. Согласно вышеприведенному определению, количество путей в полученном наборе путей меньше, чем фактическое количество, то есть некоторые пути, удовлетворяю-

щие условию ограничения, опускаются. Чтобы расширить область действия полученного набора путей, мы расширяем приведенное выше определение и позволяем включать в набор путей все подходящие пути.

Исследование работы генетического алгоритма (ГА)

ГА вдохновлены биологической генетической эволюцией, которая отбирает хромосомы с лучшими показателями приспособленности для создания потомства посредством операций скрещивания и мутации.

Как показано на рисунке 2, процесс генерации новой популяции разделен на подпроцессы скрещивания, мутации и отбора.

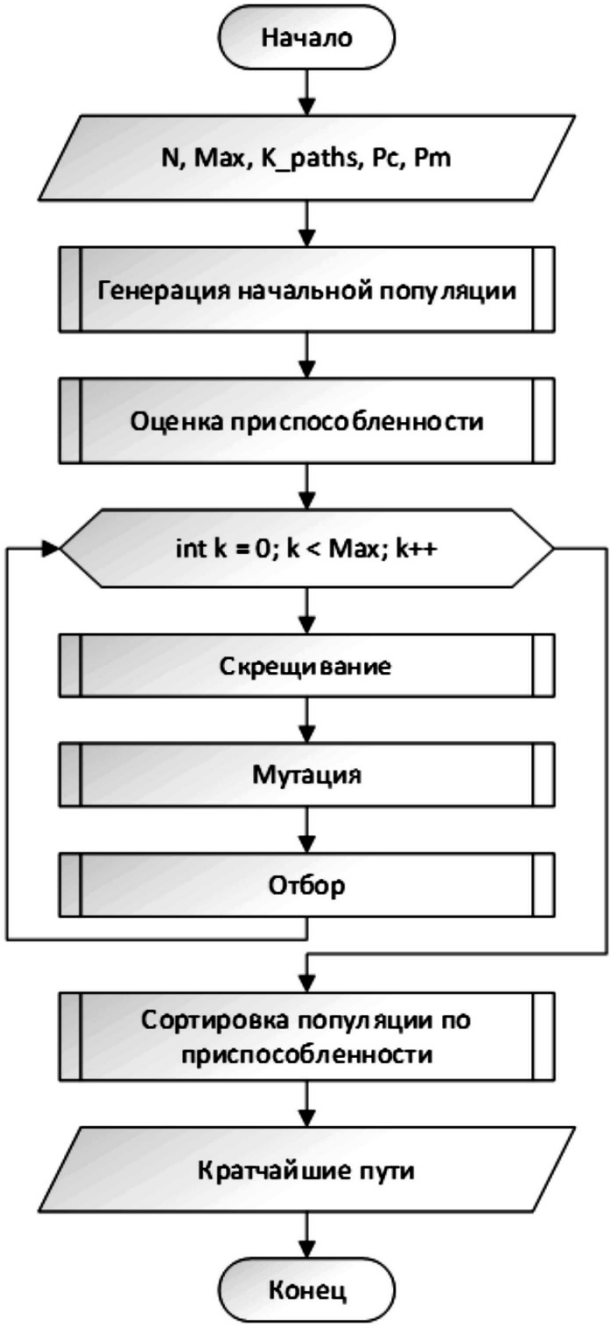

Рисунок 2 – Блок-схема генетического алгоритма
Figure 2 – Flow chart of genetic algorithm

а) генерация начальной популяции

Начальная популяция состоит из группы особей или хромосом. Каждая хромосома представляет собой возможный путь, который генерируется случайным образом и может соединять исходный узел и целевой узел. Для простоты и интуитивного восприятия в этой статье используется прямое кодирование. Каждая хромосома состоит из последовательностей положительных целых чисел, которые представляют идентификаторы узлов, через которые проходит путь. Например: хромосома C обозначается следующим образом: $[v_8, v_4, v_3, v_7, v_6, v_5]$.

Поскольку путь между двумя узлами состоит из последовательностей узлов, через которые проходит путь, составные узлы различны для разных путей, и, следовательно, значения длины разных путей также могут быть разными. Таким образом, длина хромосомы может варьироваться, но она не должна превышать общее количество узлов в сети.

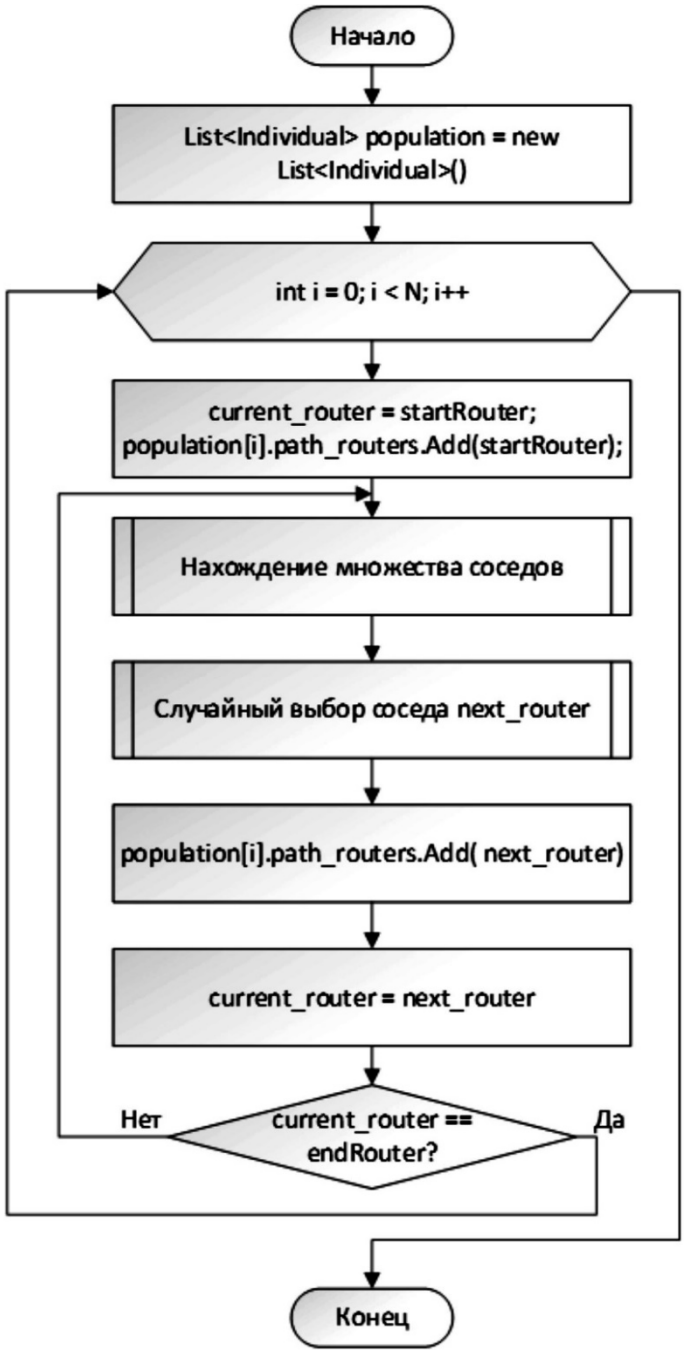

Рисунок 3 – Блок-схема процедуры генерации начальной популяции
Figure 3 – Flow chart of initial population generation procedure

б) оценка приспособленности

В ГА, приспособленность измеряет качества хромосом в популяции и, таким образом, описывает способность хромосомы к выживанию. Соответственно, функция приспособленности обеспечивает конкурентную среду для хромосом в популяции. Без потери общности, пусть вес пути будет критерием оценки, а функция приспособленности определяется как:

$$f(C_i) = w(C_i),$$

где $w(C_i)$ вычисляет значение длины пути соответствующей хромосомы C_i.

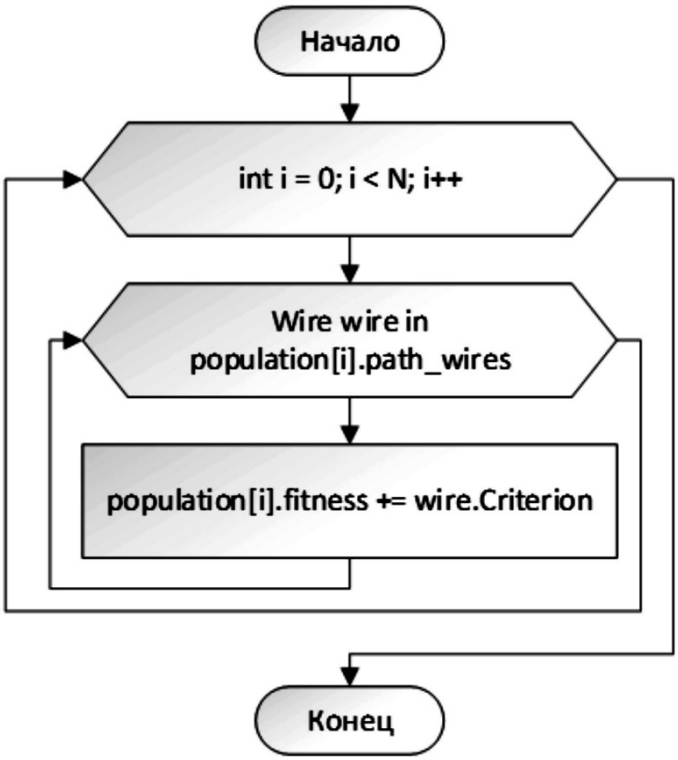

Рисунок 4 – Блок-схема оценки приспособленности
Figure 4 – Flow chart of evaluation of fitness

в) скрещивание

В ГА, оператор скрещивания (также называемый рекомбинацией) генерирует новых потомков для поиска лучших решений путем обмена частичными генами двух родительских хромосом. Скрещивание в задаче КП играет роль обмена каждым подмаршрутом двух выбранных хромосом таким образом, что потомство, производимое оператором скрещивания, представляет собой полный путь. Как правило, существует множество операторов скрещивания, таких как одноточечное, многоточечное и равномерное скрещивание. Операция скрещивания в этой статье выполняется одноточечным оператором скрещивания. Две случайно выбранные родительские хромосомы обнаруживаются в первую очередь перед операцией скрещивания, чтобы гарантировать, что у них есть хотя бы один общий ген, то есть узел, за исключением узлов источника и назначения.

Например:

Пусть двумя выбранными родительскими хромосомами для скрещивания будут $C_f = [v_8, v_4, v_3, \boldsymbol{v_7}, v_6, v_5]$ и $C_m = [v_8, v_2, v_{10}, \boldsymbol{v_7}, v_9, v_{11}, v_5]$ соответственно. Точка скрещивания – v_7. Затем два потомка, C_{s1} и C_{s2}, порожденные указанным выше одноточечным оператором скрещивания, обозначаются как:

$$C_{s1} = [v_8, v_4, v_3, \ \boldsymbol{v_7}, \boldsymbol{v_9}, \boldsymbol{v_{11}}, \boldsymbol{v_5}],$$
$$C_{s2} = [v_8, v_2, v_{10}, \ \boldsymbol{v_7}, \boldsymbol{v_6}, \boldsymbol{v_5}],$$

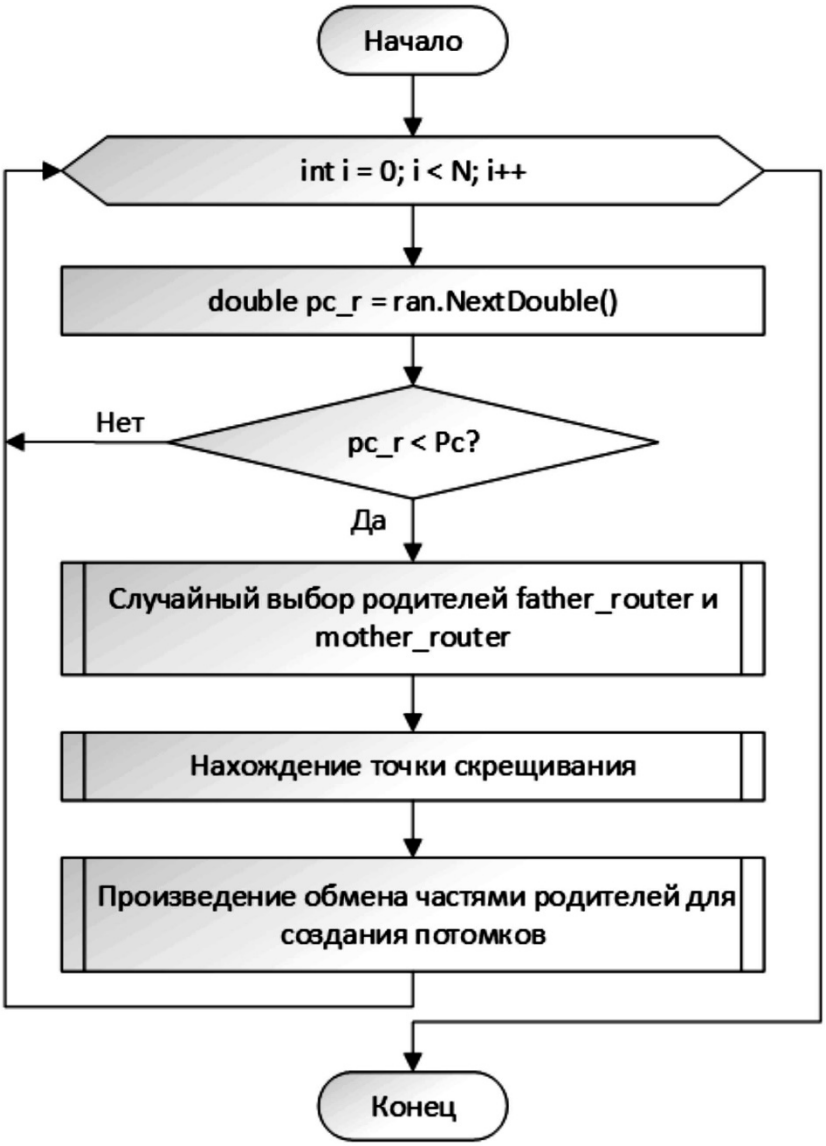

Рисунок 5 – Блок-схема процедуры скрещивания
Figure 5 – Flow chart of crossover procedure

г) мутация

В ГА один или несколько генов хромосомы изменяются случайным образом оператором мутации, добавляя генетическое разнообразие в популяцию. Посредством операции мутации ГА могут искать ранее неисследованные разделы пространства решений и, следовательно, гарантировать, что все пространство поиска связано.

Существуют различные стратегии мутации, такие как одноточечная и многоточечная мутация. В этой статье процесс мутации выполняется одноточечным оператором мутации. Для каждой выбранной хромосомы в операции мутации выбирается случайная точка мутации. Затем хромосомы претерпевают изменения с точки мутации и далее.

Например:

Пусть выбранной хромосомой для мутации будет $C_m = [v_8, v_9, v_{13}, \boldsymbol{v}_{10}, v_6, v_5]$. Точка мутации – v_{10}. Тогда новая хромосома, созданная оператором мутации, будет:

$$C_s = [v_8, v_9, v_{13}, \boldsymbol{v}_{10}, \boldsymbol{v}_7, \boldsymbol{v}_1, \boldsymbol{v}_{12}, \boldsymbol{v}_5],$$

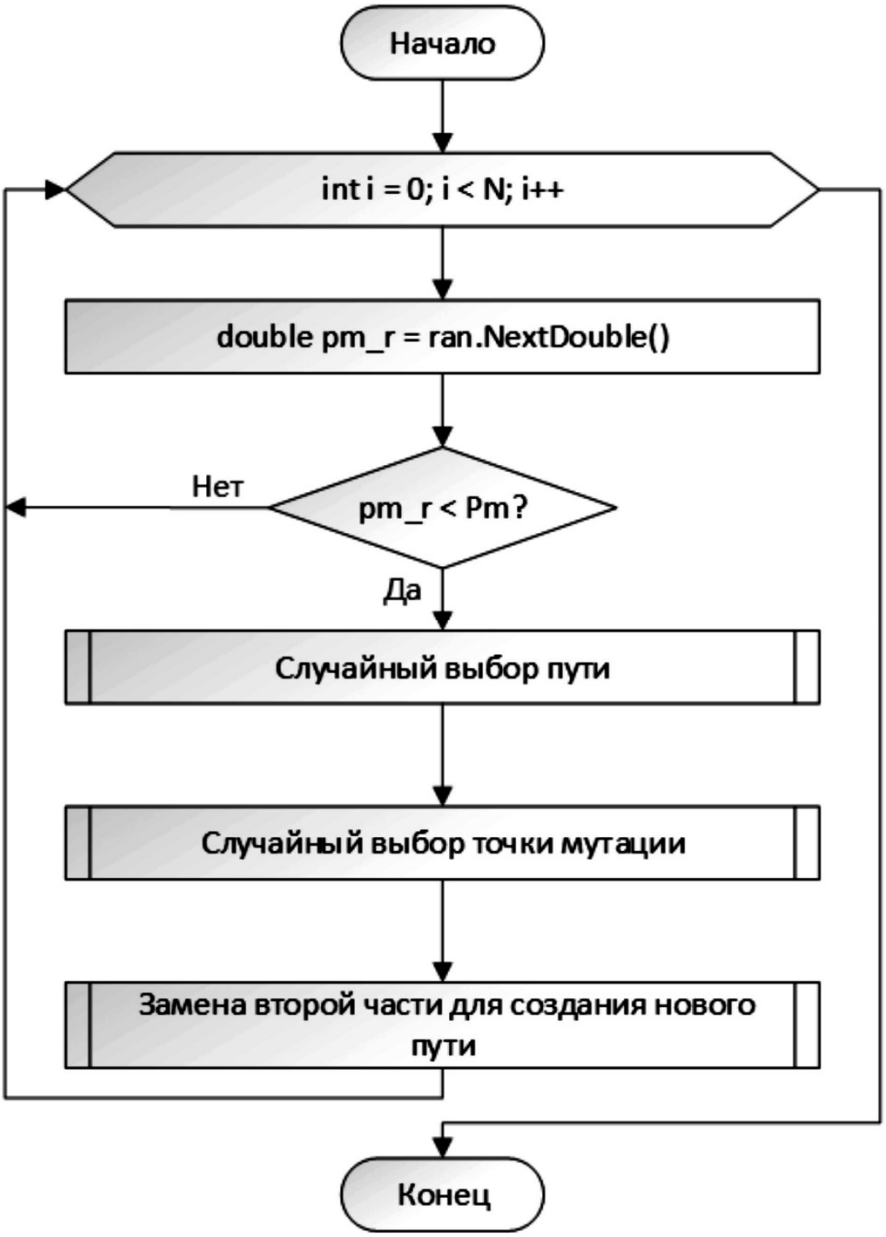

Рисунок 6 – Блок-схема процедуры мутации
Figure 6 – Flow chart of mutation procedure

д) отбор

В ГА ожидается, что операция отбора улучшит качество популяции, дав хромосомам лучшего качества больше шансов выжить в следующем поколении. Существует два типа стратегий отбора: пропорциональный и порядковый. Пропорциональный отбор выбирает хромосомы на основе их значений приспособленности относительно приспособленности других хромосом в популяции, что включает выбор колеса рулетки, стохастический выбор остатка и стохастический универсальный выбор. При порядковом отборе хромосомы выбираются не на основе их значений приспособленности, а на основе их ранжирования в популяции. Выбор турнира, выбор с усечением и выбор линейного ранжирования – вот некоторые примеры метода выбора на основе порядкового номера. Для решения практической задачи поиска нескольких оптимальных решений в предложенном алгоритме принят метод выбора с усечением.

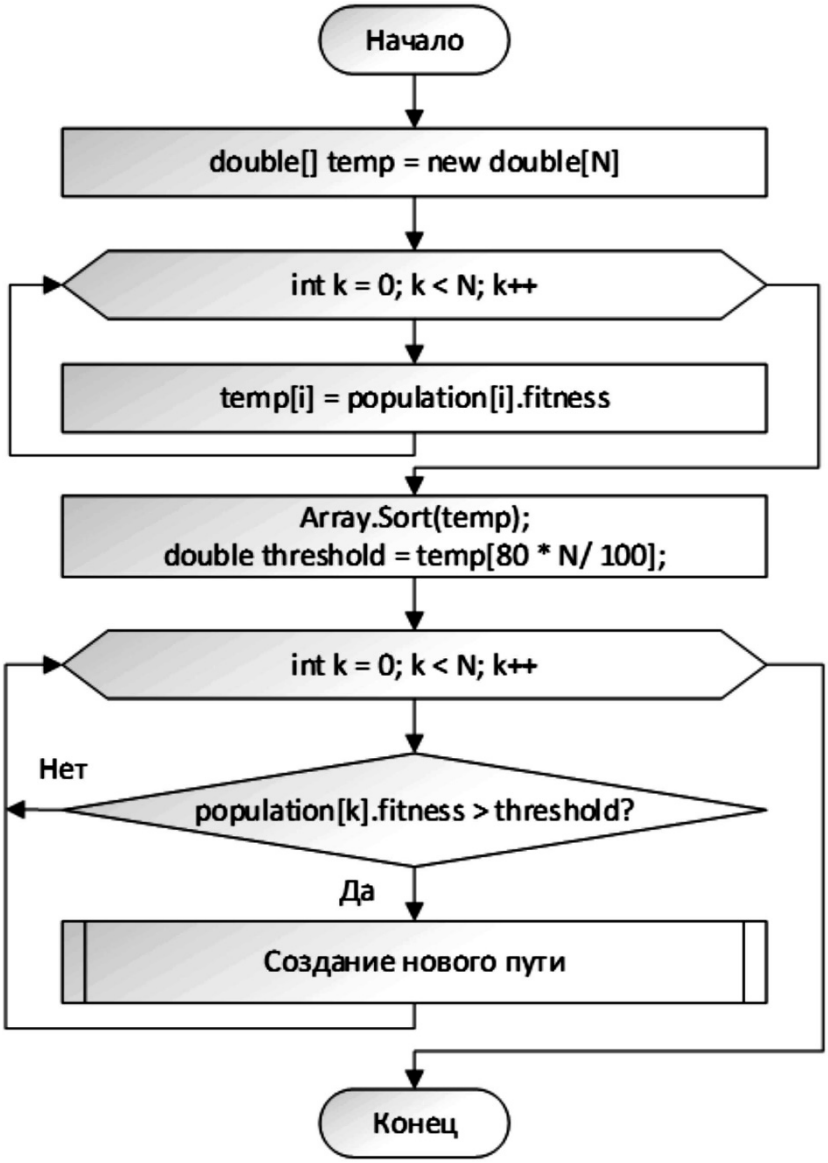

Рисунок 7 – Блок-схема процедуры отбора
Figure 7 – Flow chart of selection procedure

Экспериментальное исследование

Для подтверждения правильности предложенного алгоритма разработано программное обеспечение моделирования процессов балансировки нагрузки в программно-конфигурируемой сети (ПКС). Приложение разработано на языке программирования C# с использованием платформы .NET Framework 4.5.

На рисунке 8 представлена экспериментальная топология ПКС, состоящая из 20 узлов связи и 40 ребер. Полученные пути отмечены разными цветами. Параметры алгоритма задаются следующим образом: размер популяции $N = 100$; число итераций $Max = 100$; число кратчайших путей $K_{paths} = 4$; вероятность скрещивания $P_c = 0,8$; вероятность мутации $P_m = 0,1$.

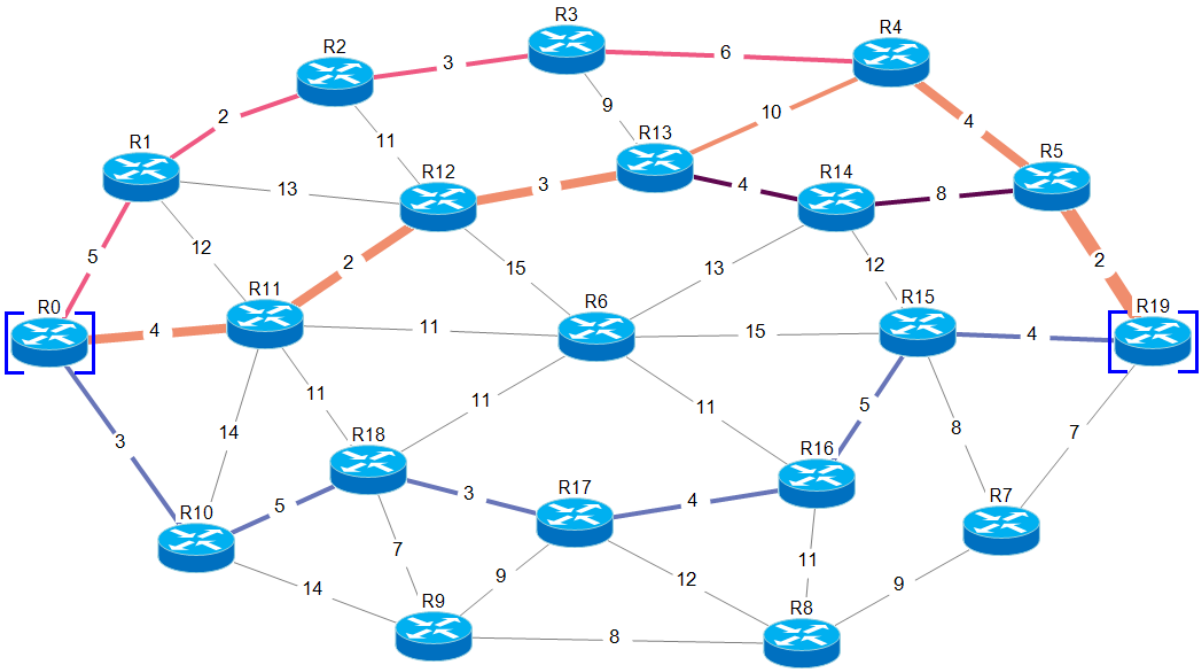

Рисунок 8 – Результат работы алгоритма в топологии ПКС из 20 узлов и 40 ребер
Figure 8 – The result of the algorithm in topology SDN of 20 nodes and 40 edges

В таблицах 1-5 приведены результаты работы генетического алгоритма для рассматриваемой топологии ПКС: Ds – длина маршрута, I – доля информации, проходящей через маршрут, AV – среднее значение каналов, входящих в машрут, SD – квадратичное отклонение каналов, входящих в маршрут, MxVL – максимальное значение канала в маршруте, MnVL – минимальное значение канала в маршруте, J – отклонение значения длины текущего маршрута от длины оптимального маршрута.

Таблица 1 – Результат работы генетического алгоритма ($N = [50;100;500]$; $Max = 100$; $K_{paths} = 4$; $P_c = 0,8$; $P_m = 0,1$)

Table 1 – The result of the genetic algorithm ($N = [50;100;500]$; $Max = 100$; $K_{paths} = 4$; $P_c = 0,8$; $P_m = 0,1$)

N	Ds	I, %	AV	SD	MxVL	MnVL	J, %
50	22	31,7	3,7	1,5	6	2	37,1
	25	27,9	4,2	2,7	10	2	
	34	20,5	8,5	4,7	15	4	
	35	19,9	5	3	10	2	
100	22	26,7	3,7	1,5	6	2	12
	23	25,5	3,8	2	8	2	
	24	24,4	4	0,8	5	3	
	25	23,4	4,2	2,7	10	2	
500	22	26,7	3,7	1,5	6	2	12
	23	25,5	3,8	2	8	2	
	24	24,4	4	0,8	5	3	
	25	23,4	4,2	2,7	10	2	

Таблица 2 – Результат работы генетического алгоритма ($N = 100$; $Max = [50;100;500]$; $K_{paths} = 4$; $P_c = 0,8$; $P_m = 0,1$)

Table 2 – The result of the genetic algorithm ($N = 100$; $Max = [50;100;500]$; $K_{paths} = 4$; $P_c = 0,8$; $P_m = 0,1$)

Max	Ds	I, %	AV	SD	MxVL	MnVL	J, %
50	22	28,4	3,7	1,5	6	2	31,3
	23	27,1	3,8	2	8	2	
	25	25	4,2	2,7	10	2	
	32	19,5	4,6	2,9	11	2	
100	22	26,7	3,7	1,5	6	2	12
	23	25,5	3,8	2	8	2	
	24	24,4	4	0,8	5	3	
	25	23,4	4,2	2,7	10	2	
500	22	28,1	3,7	1,5	6	2	24,1
	24	25,8	4	0,8	5	3	
	25	24,8	4,2	2,7	10	2	
	29	21,3	4,8	3,3	12	2	

Таблица 3 – Результат работы генетического алгоритма $N = 100$; $Max = 100$; $K_{paths} = [4; 6; 8]$; $P_c = 0,8$; $P_m = 0,1$)

Table 3 – The result of the genetic algorithm $N = 100$; $Max = 100$; $K_{paths} = [4; 6; 8]$; $P_c = 0,8$; $P_m = 0,1$)

K_{paths}	Ds	I, %	AV	SD	MxVL	MnVL	J, %
4	22	26,7	3,7	1,5	6	2	12
	23	25,5	3,8	2	8	2	
	24	24,4	4	0,8	5	3	
	25	23,4	4,2	2,7	10	2	
6	22	19,3	3,7	1,5	6	2	29
	23	18,4	3,8	2	8	2	
	24	17,6	4	0,8	5	3	
	25	16,9	4,2	2,7	10	2	
	30	14,1	4,3	2,3	9	2	
	31	13,7	5,2	2,7	11	3	
8	22	15,3	3,7	1,5	6	2	33,3
	23	14,6	3,8	2	8	2	
	24	14	4	0,8	5	3	
	25	13,4	4,2	2,7	10	2	
	30	11,2	4,3	2,3	9	2	
	31	10,8	5,2	2,7	11	3	
	32	10,5	4,6	2,9	11	2	
	33	10,2	4,7	2,6	9	2	

Таблица 4 – Результат работы генетического алгоритма $N = 100;\ Max = 100;\ K_{paths} = 4;\ P_c = [0,2;\ 0,5; 0,8];\ P_m = 0,1)$

Table 4 – The result of the genetic algorithm $N = 100;\ Max = 100;\ K_{paths} = 4;\ P_c = [0,2;\ 0,5; 0,8];\ P_m = 0,1)$

P_c	Ds	I, %	AV	SD	MxVL	MnVL	J, %
0,2	22	28,3	3,7	1,5	6	2	26,7
	24	26	4	0,8	5	3	
	25	24,9	4,2	2,7	10	2	
	30	20,8	4,3	2,3	9	2	
0,5	22	32,8	3,7	1,5	6	2	35,3
	30	24,1	4,3	2,3	9	2	
	33	21,9	4,7	3,2	12	2	
	34	21,2	8,5	4,7	15	4	
0,8	22	26,7	3,7	1,5	6	2	12
	23	25,5	3,8	2	8	2	
	24	24,4	4	0,8	5	3	
	25	23,4	4,2	2,7	10	2	

Таблица 5 – Результат работы генетического алгоритма $N = 100;\ Max = 100;\ K_{paths} = 4;\ P_c = 0,8;\ P_m = [0,1; 0,4; 0,8])$

Table 5 – The result of the genetic algorithm $N = 100;\ Max = 100;\ K_{paths} = 4;\ P_c = 0,8;\ P_m = [0,1; 0,4; 0,8])$

P_m	Ds	I, %	AV	SD	MxVL	MnVL	J, %
0,1	22	28	3,7	1,5	6	2	26,7
	23	26,8	3,8	2	8	2	
	25	24,7	4,2	2,7	10	2	
	30	20,5	4,3	2,3	9	2	
0,4	22	29,7	3,7	1,5	6	2	26,7
	25	26,1	4,2	2,7	10	2	
	29	22,5	4,8	3,3	12	2	
	30	21,7	4,3	2,3	9	2	
0,8	22	31,6	3,7	1,5	6	2	42,1
	23	30,2	3,8	2	8	2	
	35	19,9	7	3,3	11	4	
	38	18,3	5,4	3,7	12	2	

Таблица 1 показывает, что при размере популяции $N = \mathbf{100}$ и $N = \mathbf{500}$ предложенный алгоритм давал лучшие результаты, чем при $N = \mathbf{50}$. Таким образом, можно сказать, что при достаточно большом размере популяции получаются результаты, близкие к оптимальным. Из таблицы 2 видно, что при количестве итераций $Max = \mathbf{100}$ результаты лучше, чем при $Max = \mathbf{50}$ и $Max = \mathbf{500}$. Следовательно, необходимо выбрать подходящее значение Max для каждой топологии сети. Таблица 3 показывает, что с увеличением значения K_{paths} увеличивается и значение джиттера. Поскольку ясно, что длина кратчайшего пути постоянна, а по мере увеличения значения K_{paths} длина самого длинного из этих K_{paths} путей также увеличивается. Таблица 4 показывает, что при вероятности скрещивания $P_c = \mathbf{0,8}$ генетический алгоритм давал решения близкие к оптимальным результатам, чем при

$P_c = 0,2$ и $P_c = 0,5$. Данные таблицы 5 показывают, что при вероятности мутации $P_m = 0,1$ алгоритм давал лучшие результаты, чем при $P_m = 0,4$ и $P_m = 0,8$. Потому что, когда значение P_m слишком велико, многие хорошие хромосомы разрушаются.

Заключение

В работе проведены исследование и анализ процессов балансировки нагрузки в ПКС на основе генетического алгоритма. Для кодирования хромосом принята интуитивно понятная и прямая схема. Специфичные для представления операции скрещивания, мутации и отбора предназначены для выполнения генетических операций. Генетический алгоритм был применен к топологии ПКС из 20 узлов связи и 40 ребер для получения кратчайших путей от одного исходного узла к узлу назначения. Результаты моделирования подтвердили, что использование генетического алгоритма в ПКС способствует повышению эффективности сети (способности обрабатывать более интенсивный трафик) и увеличению качества сервиса для предоставления наилучшего обслуживания за счет уменьшения перегрузки сети и доли потери пакетов.

Библиографический список

1. **Перепелкин Д. А., Перепелкин А. И.** Разработка алгоритмов адаптивной маршрутизации в корпоративных вычислительных сетях // Вестник Рязанского государственного радиотехнического университета. 2006. № 19. С. 114-116.

2. **Перепелкин А. И., Перепелкин Д. А.** Разработка алгоритма динамической маршрутизации на базе протокола OSPF в корпоративных вычислительных сетях // Вестник Рязанского государственного радиотехнического университета. 2009. № 28. С. 68-72.

3. **Перепелкин Д. А.** Алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF при динамическом добавлении элементов корпоративной сети // Вестник Рязанского государственного радиотехнического университета. 2010. № 34. С. 65-71.

4. **Перепелкин Д. А.** Алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF при динамическом отказе элементов корпоративной сети // Вестник Рязанского радиотехнического государственного университета. 2011. № 37. С. 53-58.

5. **Перепелкин Д. А., Перепелкин А. И.** Повышение качества функционирования корпоративных сетей на базе протокола OSPF // Качество. Инновации. Образование. 2010. № 12 (67). С. 51-56.

6. **Liu Y., Pan Y., Wang M.** The multi‑path routing problem in the software defined network, in 11th International Conference on Natural Computation (ICNC), Zhangjiajie, China, 2015. DOI: 10.1109/ICNC.2015.7377999.

7. **Ren H., Li X., Geng J.** A SDN‑based dynamic traffic scheduling algorithm, in IEEE International Conference on Cyber‑Enabled Distributed Computing and Knowledge Discovery (CyberC), Chengdu, China, 2016. DOI: 10.1109/CyberC.2016.103.

8. **Maniu R. and Dumitru L. A.** Exploring the possibilities of a self‑regulating SDN controller, Scientific Bulletin «Mircea cel Batran» Naval Academy, vol. 18, no. 1, pp. 58-61, 2015.

9. **Перепелкин Д. А.** Концептуальный подход динамического формирования трафика программно-конфигурируемых телекоммуникационных сетей с балансировкой нагрузки // Информационные технологии. 2015. Т. 21. № 8. С. 602-610.

10. **Koryachko V. P., Perepelkin D. A., Byshov V. S.** Approach of Dynamic Load Balancing in Software Defined Networks with QoS. Proceedings of 6th Mediterranean Conference on Embedded Computing (MECO), 11-15 June 2017, Bar, Montenegro.

11. **Koryachko V. P., Perepelkin D. A., Byshov V. S.** Enhanced Dynamic Load Balancing Algorithm in Computer Networks with Quality of Services, in Automatic Control and Computer Sciences, 2018, Vol. 52, No. 4, pp. 268-282.

12. **Balaguer R.** Flow embedding algorithms for software defined audio networks [Online].

13. **Kang S. B. and Kwon G. I.** Load balancing of software‑defined network controller using genetic algorithm, Contemporary Engineering Sciences, vol. 9, no. 18, pp. 881-888, 2016. DOI: 10.12988/ces.2016.66105.

14. **Ruelas M. R. and Rothenberg C. E.** Implementation of neural switch using OpenFlow as load balancing method in data center [Online].

15. **Sanner J. M., Ouzzif M., Hadjadj‑Aoul Y.** Evolutionary algorithms for optimized SDN controllers & NVFs'placement in SDN networks [Online].

UDC 004.72

RESEARCH OF LOAD BALANCING PROCESSES IN SOFTWARE-DEFINED NETWORKS BASED ON GENETIC ALGORITHM

D. A. Perepelkin, Dr. Sc. (Tech.), Full Professor, Dean of the Faculty of Computer Engineering, RSREU, Ryazan, Russia; orcid.org/0000-0003-4775-5745, e-mail: dmitryperepelkin@mail.ru
V. T. Nguyen, post-graduate student, RSREU, Ryazan, Russia;
orcid.org/0000-0003-2930-5775, e-mail: nguyenvantinrsreu@gmail.com

As a new network paradigm, software-defined networks (SDN) are able to cope with the limitations of traditional networks. SDNs use a management controller with a global view of the network and switching devices that act as packet forwarding equipment, known as «OpenFlow switches». Network resources limitations and ensuring quality of service requirements lead to an important need for load balancing between SDN switches. ***The purpose of work –*** *research and analysis of load balancing processes in SDN based on genetic algorithm. To confirm the effectiveness and correctness of the genetic algorithm in SDN, the software for modeling the load balancing processes has been developed. The simulation results confirmed the efficiency of the genetic algorithm in SDN for balancing and redistributing network traffic in order to ensure the required quality of service and reduce network congestion.*

Key words: *software-defined networks, load balancing, genetic algorithm.*

DOI: 10.21667/1995-4565-2021-77-43-57

References

1. **Perepelkin D. A., Perepelkin A. I.** Razrabotka algoritmov adaptivnoy marshrutizatsii v korporativnykh vychislitel'nykh setyakh. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta*. 2006, no. 19, pp. 114-116. (in Russian).

2. **Perepelkin A. I., Perepelkin D. A.** Razrabotka algoritma dinamicheskoy marshrutizatsii na baze protokola OSPF v korporativnykh vychislitel'nykh setyakh. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta*. 2009, no. 28, pp. 68-72. (in Russian).

3. **Perepelkin D. A.** Algoritm adaptivnoy uskorennoy marshrutizatsii na baze protokola OSPF pri dinamicheskom dobavlenii elementov korporativnoy seti. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta*. 2010, no. 34, pp. 65-71. (in Russian).

4. **Perepelkin D. A.** Algoritm adaptivnoy uskorennoy marshrutizatsii na baze protokola OSPF pri dinamicheskom otkaze elementov korporativnoy seti. *Vestnik Ryazanskogo radiotekhnicheskogo gosudarstvennogo universiteta*. 2011, no. 37, pp. 53-58. (in Russian).

5 **Perepelkin D. A., Perepelkin A. I.** Povyshenie kachestva funkcionirovanija korporativnyh setej na baze protokola OSPF. *Kachestvo. Innovacii. Obrazovanie*. 2010, no. 12 (67), pp. 51-56. (in Russian).

6. **Liu Y., Pan Y., Wang M.** The multi‑path routing problem in the software defined network, in 11th International Conference on Natural Computation (ICNC), Zhangjiajie, China, 2015. DOI: 10.1109/ICNC.2015.7377999.

7. **Ren H., Li X., Geng J.** A SDN‑based dynamic traffic scheduling algorithm, in IEEE International Conference on Cyber‑Enabled Distributed Computing and Knowledge Discovery (CyberC), Chengdu, China, 2016. DOI: 10.1109/CyberC.2016.103.

8. **Maniu R. and Dumitru L. A.** Exploring the possibilities of a self‑regulating SDN controller. *Scientific Bulletin «Mircea cel Batran» Naval Academy*, vol. 18, no. 1, pp. 58-61, 2015.

9. **Perepelkin D. A.** Konceptual'nyj podhod dinamicheskogo formirovanija trafika programmno-konfiguriruemyh telekommunikacionnyh setej s balansirovkoj nagruzki. *Informacionnye tehnologii.* 2015, vol. 21, no. 8, pp. 602-610. (in Russian).

10. **Koryachko V. P., Perepelkin D. A., Byshov V. S.** Approach of Dynamic Load Balancing in Software Defined Networks with QoS. Proceedings of 6th Mediterranean Conference on Embedded Computing (MECO), 11-15 June 2017, Bar, Montenegro.

11. **Koryachko V. P., Perepelkin D. A., Byshov V. S.** Enhanced Dynamic Load Balancing Algorithm in Computer Networks with Quality of Services, in Automatic Control and Computer Sciences, 2018, vol. 52, no. 4, pp. 268-282.

12. **Balaguer R.** Flow embedding algorithms for software defined audio networks [Online].

13. **Kang S. B. and Kwon G. I.** Load balancing of software‐defined network controller using genetic algorithm, Contemporary Engineering Sciences, vol. 9, no. 18, pp. 881-888, 2016. DOI: 10.12988/ces.2016.66105.

14. **Ruelas M. R. and Rothenberg C. E.** Implementation of neural switch using OpenFlow as load balancing method in data center [Online].

15. **Sanner J. M., Ouzzif M., Hadjadj‐Aoul Y.** Evolutionary algorithms for optimized SDN controllers & NVFs'placement in SDN networks [Online].