Flow Entry Timeouts Optimization over Software Defined Networks Supporting Elephant Flow Classification

Changqing Zhao[1], Ling Xia Liao[1], Han-Chieh Chao[2], Roy Xiaorong Lai[3], Miao Zhang[4*]

[1] School of Electronic Information and Automation, Guilin University of Aerospace Technology, China
[2] Department of Electric Engineering, National Dong Hwa University, Taiwan
[3] Confederal Networks Inc., USA
[4] Quanzhou University of Information Engineering, China
zhaochq@guat.edu.cn, liaolx@guat.edu.cn, hcc@ndhu.edu.tw, roy.lai@ieee.org, zm@qziedu.cn

Abstract

Elephant flow (elephant) classification is significant for network performance management and resource optimization. Classifying elephants over Software Defined Networks (SDNs) often relies on statistics counted by switches and transferred to controllers, leading to a huge control channel bandwidth occupation that degrades network latency and user experience. Classifying elephants using flow packets forwarded to controllers completely avoids moving statistics from switches to controllers, but couples flow entry timeouts to elephant classification accuracy, time efficiency, flow table size, control channel bandwidth usage, and network latency. This paper aims to find the best flow entry timeouts to optimize the objectives by modeling the objectives and formulating a multi-objective optimization problem. The number of objectives is further reduced to 3 and the problem is solved by Machine Learning (ML) approaches. Extensive evaluations are made over real packet traces. To the best of our knowledge, this work is the first effort that explores all the objectives related to flow entry timeouts when using flow packets forwarded to controllers to classify elephants over SDNs.

Keywords: SDN, Elephant flow classification, Flow table size, Control channel bandwidth usage, Flow entry timeout

1 Introduction

Elephant flows (elephants) refer to the flows having large packet counts, byte counts, and time durations, while mice flows (mice) are others. Elephants consume the majority of network bandwidth, and mice are latency sensitive. Although the quantity of elephants is far less than mice, it is significant to identify elephants for network performance management and resource optimization in a global manner, as two elephants sharing the same link may cause network congestion and degrade network Quality of Service (QoS) [1].

Elephant classification over conventional computer networks typically relies on packet traffic/statistics collected by hosts or switches. Since hosts and switches are only able to gather their local traffic/statistics in a network, extra costs and difficulties are needed to gather global traffic/statistics and enable QoS management and resource optimization over the entire network [2].

Software Defined Networks (SDNs) have a layered architecture with separated control and data planes [3]. The control plane consists of controllers and the data plane includes devices and hosts. Devices in the data plane are dumb and rely on flow entries to forward packets. Flow entries are flow-based forwarding rules generated by controllers but stored in flow tables located in the Ternary Content Addressable Memory (TCAM) of switches.

Besides forwarding flow packets, flow entries also count statistics of packets triggering the entries. Therefore, SDN controllers can collect the statistics of each switch using standardized protocols and specifications, and form the global statistics. Based on such statistics, SDN controllers can classify elephants and further optimize network performance and resource usage in a global manner.

However, simply moving such statistics from all the switches to controllers occupies too much control channel bandwidth. Such bandwidth is consumed periodically as the statistics needs to be periodically updated for elephant classification. Since control channel transfers general network management messages, including flow entry installation messages, controller-switch connection establishing messages, and flow statistics polling and response messages, between switches and controllers, the more control channel bandwidth is used to transfer network statistics, the less bandwidth is left for other control messages, leading to a longer network latency degrading network performance and QoS.

To reduce the control channel bandwidth used to transfer network statistics, current research often deploys agents at switches to sample flow packets forwarded to controllers, or allows switches to have the intelligence to sample or aggregate the statistics forwarded to controllers. However, deploying sampling agents often needs to update the software of switches, and aggregating statistics at switches lacks standardized hardware and interfaces to support programmable data planes, raising a big concern on system compatibility, interoperability, and usability in practice [2].

Our previous work proposed to allow controllers to classify elephants using the flow statistics generated by the received packets [4]. In SDNs, switches forward the first packet of each flow to controllers to set up its flow entry when flow entries are generated in a reactive mode. Switches also forward some of the following packets of each flow to

*Corresponding Author: Miao Zhang; E-mail: zm@qziedu.cn
DOI: 10.53106/160792642022092305011

controllers if the matched flow entry has been timed out when the packets arrive in switches. SDNs enforce each flow entry in flow tables to have a timeout to better use the flow table space, because flow tables are stored in the TCAM of switches and the size of TCAM is limited due to its power consumption. Therefore, flow packets forwarded to controllers are actually the packets sampled by the flow entry timeouts. Since elephants often have larger packet counts and longer time duration than mice, elephants have a trend to forward more packets to controllers than mice given flow entry timeouts. Therefore, it is feasible to classify elephants using the flow packets sampled by the flow entry timeouts.

However, such an approach couples flow entry timeouts to elephant classification accuracy and time efficiency. This is because packet sampling based on flow entry timeouts causes packet missing that affects the total byte counts of flows forwarded to controllers, leading to a distribution shift between the sampled and unsampled packet data sets. Therefore, it is significant to optimize flow entry timeouts to provide accurate and time efficient elephant classification when classification is based on the flow packets sampled by the flow entry timeouts.

It is significant to configure flow entry timeouts considering the optimization of flow table size, control channel bandwidth usage, network latency, elephant classification accuracy and time efficiency. This is because flow entry timeouts directly affect such objectives. While flow table and control channel bandwidth are valuable resources in SDNs, network delay, elephant classification accuracy and time efficiency are critical performance measures for SDNs when elephant classification is applied over the flow packets forwarded to controllers. Therefore, this paper considers an optimization problem that finds the best flow entry timeouts to optimize the flow table usage, the control channel bandwidth consumption, the network delay, and the elephant classification accuracy and time efficiency.

This is an optimization problem with 5 objectives. Among them, some objectives are conficted. For instance, shorter timeouts makes flow entries timed out more frequently, leaving more flow table space to new flow entries for incoming flows and leading to a better flow table usage. However, shorter flow entry timeouts enforce more flow packets forwarded to controllers, leading to a higher elephant classification accuracy and time efficiency, larger controller channel bandwidth usage, and longer network delay. To reduce the control channel bandwidth usage and network delay, the value of timeouts has to be larger, making less quantity of flow packets of forwarded to controllers that reduces the elephant classification accuracy. Longer flow entry timeouts also enforce flow entries to live longer in flow tables. As SDNs have more mice than elephants, giving longer timeouts to flow entries make mice' flow entries still active in flow tables but the respective flows have been out of their lifetime in the network, wasting the valuable flow table space and leading to a poor table space usage.

Accordingly, this paper firstly fomulates a multi-objective optmization problem, then simplifies the number of objectives and applies Machine Learning (ML) algorithms to solve the simplified problem. Although our previous work has optimized the flow entry timeouts when classifying elephants over the flow packets forwarded to controllers, this paper considers more related objectives and finds ways to simplify and solve them. The major contributions of this paper are three folds.

1. A multi-objective optimization problem that finds the best set of flow entry timeouts to simultaneously optimize the flow table usage, control channel bandwidth usage, network delay, elephant classification accuracy and time efficiency is formulated.
2. Objectives are simplified and 9 types of flow entry timeouts adjustment strategies are proposed. Nondominated Sorting Genetic Algorithm-II (NSGA-II) [5] and Multi-objective Bayesian Optimization (MBO) [6] algorithms are applied to solve the simplified problem.
3. Extensive evaluations are made to discuss the tradeoff of objectives over real packet traces of networks.

The rest of this paper is organized as follows. While the relate work is summarized in Section 2, the multi-objective optimization problem is formulated in Section 3. Section 4 solves the problem, Section 5 evaluates the proposed approaches followed by the conclusion drawn in Section 6.

2 Related Work and Background

2.1 Single Objective Problem

In SDNs, the major single objective optimization problems typically relate to the optimization of elephant classification, control channel bandwidth usage, and flow table management.

2.1.1 Flow Table Management

Flow table optimization usually optimizes the flow table usage due to the space shortage. The major strategies used consist of dynamic flow entry timeouts and eviction, flow entry aggregation and compression, flow entry splitting, distribution, and caching.

Giving dynamic timeouts to flow entries and designing efficient eviction mechanisms can minimize the number of flow entries in flow tables, and hence reduce the flow table size. Flow entry timeouts typically can be adjusted based on the estimation of the number of flow entries that may appear at the next sampling time in probability [7-8]. Flow entry eviction approaches [9-10] can efficiently remove timed out flow entries to ease the impact of flow table overflowed. Aggregation compresses fine-grained forwarding entries into fewer coarse-grained with slightly larger matching range to reduce the number of entries to be stored [11-12]. By deciding which rules to be placed on which switch while respecting memory constraint and rule dependency, flow entries can be split and distributed over the network to reduce the flow table space used by a single switch [13-14]. Caching timed out flow entries in memory instead of kicking them out enables the efficient use of flow table meanwhile reducing network latency [15-16].

2.1.2 Control Channel Bandwidth

In SDNs, control channel transfers the control requests and responses such as flow entries installation and reactivation messages, network traffic/statistics polling requests and

responses. The total number of flow packets forwarded to controllers are typically controlled by the flow entry timeouts. Distributing the total number of flow packets to multiple controllers reduces the control channel bandwidth usage for each controller in general [17-18].

Regarding the control channel bandwidth used to move network traffic/statistics from switches to controllers, aggregation and sampling are often applied. DevoFlow [19] and iSTAMP [20] applied various aggregation strategies to reduce the volume of flow statistics forwarded to controllers. Traffic/statistics sampling also applied to detect elephants [21], monitor network [22], and measure network latency [23].

2.1.3 Elephant Classification

Studies on elephant classification in current research typically focus on the improvement of classification accuracy, efficiency, and stability. Elephant classifications often can achieve a very high accuracy over the training data sets. However, the accuracy over the testing data sets (the stability) may significantly decreased due to the distribution shift of testing data sets in the reality.

Elephant classification time efficiency is highly relied on the features used to model and the interval during which traffic/statistics are collected. Carefully choosing the collection interval improves the time efficiency of elephant classification. Optimizing the sampling period can reduce the distribution shifts and hence improve the classification accuracy and time efficiency. The typical optimization problem related to elephant classification is to find the best collection or sampling periods to maximize the classification accuracy [24-26].

2.2 Multi-objective Problems

Managing network performance and optimizing network resource usage over SDNs often have to consider at least two objectives due to the dependency. Flow table usage can be presented as the table size, hit ratio, and the number of capacity misses. Such objectives may be jointly optimized to find the best flow entry timeouts [27-29].

Current studies on flow table optimization also jointed with the optimization of network delay and resource usage. As references [30] and [31] reduced the number of flow entries and the load imbalance on link resource utilization; reference [32] optimized the value of idle timeouts of flow entries to jointly minimize the flow table resource usage and the controller computing resource cost; reference [33] formulated an optimization problem that found the best active links to minimize the flow table size and power consumption; and reference [34] tried to find the Pareto frontier of routing routes that optimize both congestion metric and link utilization.

2.3 Algorithms

The algorithms used to solve the optimization problems search the best solution in the given solution space. Exhaustive search, heuristic search, and ML-based search are often used.

2.3.1 Exhaustive Search

Given a solution space, exhaustive search goes through every solution in the solution space. Exhaustive search guarantees global optima but not time acceptable or impossible when the solution space is too big or continuous.

2.3.2 Heuristic Search

Heuristic search refers to a type of search approaches that optimizes a problem by iteratively improving the solution based on a given heuristic function or a cost measure. Heuristic search finds a good or acceptable solution within a reasonable amount of time and memory space instead. Heuristic search is widely used to improve the flow table usage [28], control channel bandwidth consumption [27], and flow classification over SDNs [25-26].

2.3.3 ML-based Search

ML-based algorithms are widely used in current research. Although ML techniques are used in flow classification, flow monitoring, and attack detection, they do not directly solve optimization problems but provide strategy to provide better solutions in various user scenarios [2].

Bayesian Optimization (BO) is a type of searching approaches widely used to tune hyperparameters of ML algorithms. For continuous functions, BO typically assumes the unknown function is sampled and maintains a posterior distribution for this function as observations are made. Based on the function evaluation, several potential solutions are picked as the hyperparameters of the next experiment. BO has been applied to tune the parameters of flow classifiers [35] and the routing of SDNs [36]. MBO is the multi-objective version of BO [6], and has been applied in optimizing SDNs [37].

Evolutionary Algorithms (EAs) are search approaches inspired by the natural biological evolution or social behaviors. EAs can be seen as heuristics search algorithms in the sense that such algorithms need to initialize solutions to start the searching iteration. EAs also can be seen as ML-based searching algorithms, since they allow machines to learn the strategy. Genetic algorithms (GAs) are EAs that mimics the natural biological evolution. GAs often solve the single-objective optimization problems, and the multi-objective version, NSGA-IIs, can solve multi-objective optimization problems with higher quality of solutions. GAs and NSGA-IIs can be used to solve any optimization problems in general. In SDNs, GAs have been applied to select features of flows to model flows for accurate flow classification [38], and NSGA-IIs have been involved to optimize various metrics [33-34].

3 Multi-objective Optimization Problem Formulation

This section firstly models the objectives, then formulates the multi-objective optimization problem. Finally, the objectives are discussed and simplified.

Let F be the flow set maintained by controllers to be classified, for each flow $f_i \in F$, J_i be the total number of packets of flow f_i. Let p_{ij} be the size of the jth packet of f_i, and a_{ij} be a binary parameter representing whether or not the jth packet of flow f_i is forwarded to controllers. a_{ij} is 1 if the jth packet of f_i is forwarded to controllers and 0 otherwise. The total byte count of flow f_i forwarded to controllers and the total byte count of all the flows forwarded

to controllers are denoted by P_i and P and they are computed by Equations (1) and (2), respectively.

$$P_i = P_{i1} + \sum_{j=2}^{J_i}(a_{ij}p_{ij}) \quad \textbf{(1)}$$

$$P = \sum_{f_i}^{F}(P_i) \quad \textbf{(2)}$$

To determine a_{ij}, we have to consider the timeout of flow entries. Let $T_{icurrent}$ and t_{i_update} be the current timeout of the flow entry and the current update time of the flow entry of f_i, respectively, we consider the hard and idle types of timeouts. For hard timeouts, let t_{ij} be the timestamp that the jth packet of f_i being forwarded to controllers, then a_{ij} is 1 (being forwarded to controllers) if t_{ij}-t_{i_update}>$T_{i\_current}$ and 0 otherwise (not being forwarded). Regarding idle timeouts, a_{ij} is 1 if t_{ij}-$t_{i(j-1)}$>$T_{i\_current}$ and 0 otherwise.

3.1 Modeling Control Channel Bandwidth Usage

To compute the control channel bandwidth usage *C*, we consider a 1000Mbits SDN and let the total bandwidth of control channel be 1000Mbits per second, then C can be computed by Equation (3), as *P* is total byte counts forwarded to controllers during the interval of T minutes.

$$C = (8 \times \mathrm{P})/(\mathrm{T} \times 60 \times 10^9) \quad \textbf{(3)}$$

3.2 Modeling Flow Table Size

We compute the number of active flow entries in flow table (S) during the given interval of T and take the maximum as the usage of flow table. Let e_i be a binary parameter representing the state of flow entry of f_i in flow tables. We let e_i be 1 if the flow entry is active and 0 otherwise, then for each time spot $\in T$, the total number of active flow entries in flow tables S can be computed by Equation (4)

$$S = \max_{t \in T} \sum_{f_i}^{F} e_i \quad \textbf{(4)}$$

3.3 Network Latency

Since the number of flow packets forwarded to controllers implies the number of interaction between switches and controllers, we directly use the number of flow packets forwarded to controllers *P* to represent the network latency L, as shown in Equation (5).

$$L = P \quad \textbf{(5)}$$

3.4 Modeling Elephant Classification Inaccuracy

Let $label_i$ be the label of f_i, and it is 1 if flow f_i is marked as a real elephant and 0 otherwise. We let plabel$_i$ be the prediction flow f_i made by the model. Since each packet received by controllers can trigger an action of classification, while each flow may forward multiple packets to controllers, $plabel_i$ is the final prediction made by the classifier. Then, the F1-score (*F1*) of elephant classification can be computed using Equation (6), in which Rec and Pre are the recall and precision that can be computed using Equations (7) and (8), respectively. We use Equation (9) to present the elephant classification inaccury.

$$F1 = 2(Pre \times Rec)/(Pre + Rec) \quad \textbf{(6)}$$

$$Rec = \sum_{f_i}^{F}(plabel_i \times label_i)/\sum_{f_i}^{F} label_i \quad \textbf{(7)}$$

$$Pre = \sum_{f_i}^{F}(plabel_i \times label_i)/\sum_{f_i}^{F} plabel_i \quad \textbf{(8)}$$

$$\mathrm{A} = 1/\mathrm{F1} \quad \textbf{(9)}$$

3.5 Elephant Classification Time Inefficiency

Regarding the elephant classification time cost *E*, we let $Telephant_i$ be the accumulated time when flow f_i is classified as an elephant, and T_i be the life span of f_i. Then $Telephant_i/T_i$ computes the classification time cost of f_i, and the overall classification time cost E is the average classification time cost for all real elephants being classified correctly, as shown in Equation (10). We simply use E to model the elephant classification time inefficiency.

$$E = \sum_{f_i}^{F}(plabel_i \times label_i \times Telephant_i/T_i)/ \sum_{f_i}^{F}(plabel_i \times label_i) \quad \textbf{(10)}$$

3.6 Multi-objective Optimization Problem Formulation

Let $T_{icurrent\_}$ be the current timeout of $f_i \in$ F, objectives of *A*, *C*, *L*, *E*, and *S* are the functions of $T_{icurrent\_}$. Therefore, the proposed multi-objective optimization problem can be formulated to find the best set of timeout $T_{icurrent\_}$ for each $f_i \in$ F such that the active flow table size $S(T_{icurrent\_})$, the control channelbandwidth usage $C(T_{icurrent\_})$, the network latency $E(T_{icurrent\_})$, and elephant classification inaccuracy $A(T_{icurrent\_})$ and time inefficiency $E(T_{icurrent\_})$ are jointly minimized, as illustrated by Equation (11), TT is the largest value of timeout that a flow entry can have. Table 1 lists the symbols and their descriptions used in this paper

$$\underset{T_{icurrent} \in (0,TT)}{\text{minimize}} \; fun(\mathrm{S}(T_{icurrent}), \mathrm{C}(T_{icurrent}), \mathrm{L}(T_{icurrent\_}), \mathrm{A}(T_{icurrent\_}), \mathrm{E}(T_{icurrent\_})) \quad \textbf{(11)}$$

3.7 Simplifying Optimization Problem

Optimization problems with 3 more objectives are very hard to be solved. Since some of the objectives modeled above are not conflicted, they can be simplified. Particularly, when classifying elephants based on the flow packets sampled by the flow entry timeouts, optimizing flow table usage S is to optimize the elephant classification inaccuracy *A*. This is because decreasing the value of flow entry timeouts can send more flow packets to controllers and save more flow table space for incoming flows, leading to smaller S and A simultaneously. Similarly, more flow packets sent to controllers consumes higher control channel bandwidth, and also means more interaction between switches and controllers, leading to bigger C and longer *L*. Therefore, optimizing control channel bandwidth usage C is to optimize the network latency

L. However, improving *S* and *A* does worsen *C* and *L*, because shorter timeout value decreases *S* and *A* but increases *C* and *L*. Accordingly, the proposed optimization problem can be simplified as a multi-objective optimization problem that finds the best flow entry timeouts to minimize the controller channel bandwidth usage *C*, elephant classification inaccuracy *A*, and elephant classification inefficiency *E*, as shown in Equation (12). It should be noticed that classification time efficiency *E* seems to have the same optimization direction with *S* and *A*, because shorter flow entry timeouts also reduce *E*, the time cost of elephants in the given data set being correctly classified. However, we keep *E* in Equation (12) considering its importance.

$$\underset{T_{icurrent}\in(0,\mathrm{TT})}{\text{minimize}} \; fun(\mathrm{C}(T_{icurrent}), \mathrm{A}(T_{icurrent}), \mathrm{E}(T_{icurrent})) \tag{12}$$

Table 1. Symbols used and the description

Symbols	Descriptions
F	flow set
i	flow identifier
j	packet identifier
f_i	flow i in F
A	elephant classification inaccuracy
E	elephant classification time efficiency
S	flow table size
L	network latency
C	control channel bandwidth usage
t_{ij}	timestamp of f_i's jth packet
t_{i_update}	update time of f_i's flow entry
$T_{icurrent\_}$	f_i's current flow entry timeout
p_{ij}	size of f_i's jth packet
a_{ij}	f_i's jth packet forwarded or not
P_i	byte count of fi forwarded
P	total byte count of flows forwarded
$\mathrm{J_i}$	f_i's packet count
e_i	f_i's flow entry active or not
T	time interval
t	time spot in T
$label_i$	label of flow f_i
$plabel_i$	prediction of flow f_i
$F1$	elephant classification F1-score
Rec	elephant classification recall
Pre	elephant classification precision
T_i	Time duration of flow f_i
$Telephant_i$	f_i's duration when detected as elephant
TT	timeout's upper bound

4 Solving Optimization Problem

This section firstly develops strategies adjusting the flow entry timeouts, then NSGA-II and MBO are applied to solve the simplified problem for each strategy.

4.1 Timeout Adjustment Strategies

Given the interval T during which the objectives are computed, the flow entry timeouts can be fixed or dynamically changed. While fixed idle or hard timeouts are often given to flow entries for the simplicity, dynamical timeouts are involved to reduce the flow table size, although they are complicated and may add overhead on controllers. To reduce the solution space of the proposed optimization problem, we consider the following timeout strategies: fixed hard and idle timeouts, hard and idle timeouts dynamically adjusted based on their averaged variances, hard and idle timeouts dynamically changed with a fixed ratio, and hybrid timeouts, as listed in Table 2.

- Fixed timeouts refer to hard or idle timeouts with fixed value during the interval T.
- Variance-averaged strategy allows each flow entry to have a hard or idle timeout with an initial value. Such value remains unchanged until the corresponding flows are identified as elephants. The flow entry timeout of an elephant classified is adjusted to the sum of the mean and the variance of packet inter-arrival time of the flow. This strategy has a high probability to make the flow entry of an elephant live to the arrival of the next packet of the flow, leading to a reduced control channel usage and flow table size.
- Fixed-ratio strategy allows each flow entry to have a hard or idle timeout with an initial value and remains the value unchanged until the corresponding flow is classified as an elephant. After flows are classified as elephants, the value of their timeouts is increased with a fixed ratio whenever their flow entry is timed out. Fixed-ratio strategy considers elephants forward more packets to controllers, reducing the number of packets elephants forward to controllers is to reduce the control chanel bandwidth usage.
- Hybrid timeouts dynamically adjust the type and value of flow entry timeouts during the interval T. It gives hard timeouts to flow entries and switched hard timeouts to idle timeouts after flows are classified as elephants. It balances the elephant classification inaccuracy, flow table size, and control channel bandwidth usage. The value of timeouts can be fixed or dynamically adjusted using variance -averaged, fixed-ratio, or many more.

Table 2. Flow entry timeouts adjustment strategies

Types	Strategies	Descriptions	#
Fixed	Hard	Value not changed	1
	Idle	Value not changed	2
Var-ave	Hard	Value changed after ants elephant classified based on the mean packet inter-arrival time	3
	Idle		4
Fixed-ratio	Hard	Value increased proportionally after elephants classified	5
	Idle		6
Hybrid of Hard/Idle	Fixed	Switch type fix value	7
	Var-ave	Type/value changed based on the mean packet inter-arrival time after elephants classified	8
	Fixed-ratio	Type/value proportionally changed after elephants classified	9

4.2 Solving Problems

After applying the timeout adjustment strategies, the proposed problem is converted to find the best initial value of timeouts to minimize the control channel bandwidth usage, the elephant classification inaccuracy, and the elephant classification time inefficiency. To solve this problem, we apply NSGA-II and MBO algorithms. The result is not unique. It is a Pareto frontier that consists of a set of Pareto optimal solutions in the objective space, each of which is at least as good as the others in at least one but not all dimensions. The final Pareto frontier is the mix of the frontiers generated by NSGA-II and MBO.

4.2.1 NSGA-II

NSGA-II is a multi-objective evolutionary algorithm that uses a fast nondominated sorting procedure, an elitist-preserving approach, and a parameterless niching operator [5]. Given a population size N and the number of objectives M, NSGA-II has the overall computation complexity of $O(MN^2)$ to find the Pareto-optimal frontier among all nondominated levels of frontiers, while a naive sorting procedure requires $O(MN^3)$ comparisons. NSGA-II also can maintain a good spread of solution in the Pareto-optimal frontier by computing the crowding distance to estimate the density of solution surrounding a particular solution in the population, and selecting the solutions with lower (better) nondomination rank and located in a lesser crowded region to form the Pareto-optimal frontier. The crowding distance is the average distance of two solutions on either side of this solution along each of the objective.

Since NSGA-II has been widely used in current research, we apply the existing NSGA-II algorithm implemented in Python[1] to solve the problem. In particular, we let the value of timeouts varying in (0,10s) due to 91% of the elephants and 88% of the mice having mean packet inter-arrival time between 0 and 10 seconds. We let each population consisting of 30 solutions and the total of 50 generations are evolved, which are the default configuration in many NSGA-II algorithms. NSGA-II is a procedure that finds the Pareto-optimal frontier iteratively. Given the population initialized randomly, NSGA-II starts to generate its child generation using crossover and mutation operators. The fitness of each solution in both parent and child generations is calculated and sorted. The solutions with the best nondomination ranks and crowding distance are chosen to form the next generation of parent population and restart the iteration again until the number of generations evolved hits the maximum. The fitness of a solution can be calculated using the objective functions. For the proposed three-objective optimization problem, each solution has a fitness value corresponding to an objective computed by Equations (3), (9), and (10).

4.2.2 Multi-objective BO

As similar as NSGA-II the multi-objective version of GA for multi-objective optimization problems, MBO algorithm is the multi-objective version of BO algorithm. Unlike a GA using selection, crossover, and mutation to generate next generation of population, BO uses the information extracted from the entire set of promising solutions. A BO algorithm often starts from a randomly generated population of solutions, then applies Bayesian network to estimate the distribution of the selected set of solutions. New solutions are selected according to the estimation and added to the original population to form the offspring population. The process is repeated until the termination criteria are met. The MBO incorporates the selection method of NSGA-II into BO. Therefore, the MBO has the same computation complexity of $O(MN^2)$ as NSGA-II, although a BO algorithm may be faster than a GA. Applying MBO to solve the proposed problem is due to: (1) MBO is highly efficient regarding the number of objective function evaluation, and (2) it does not require any analytical knowledge of the objectives, allowing the methods to perform well with black-box functions [39].

In particular, MBO methods start from randomly generating n solutions (parent population), then build a probabilistic model (Bayesian network) of the promising solutions. After new solutions (offspring population) are sampled based on the Bayesian network, both the parent and the offspring population are combined to perform a non-dominated sorting based on the crowding distance and nondomination ranks. The n best (based on the rank and crowding distance) solutions are selected to form the next generation of parent population to repeat the process till some convergence criteria are satisfied.

[1] github.com/anyoptimization/pymoo

4.2.3 Solutions with Given Timeouts

As NSGA-II and MBO are both algorithms have populations initialized randomly, they can only generate approximation Pareto optimal frontiers for multi-objective optimization problems. To further improve the quality of the fronts generated, we manually generate 15 initial values of timeouts varying from 0.001×2^0 to 0.001×2^{13}s plus 10s. We apply the 9 timeout adjustment strategies and calculate the values of each objective. We add the results to the final fronts generated by both NSGA-II and MBO and choose the nondominated solutions to form the final frontiers.

5 Evaluation

This section evaluates the Pareto frontiers generated by NSGA-II and MBO together with the solutions with given initial timetous over the data set of TR1, which is the packet trace caught from the campus network of Guilin University of Aerospace Technology in December 2019. Three types of timeouts and 9 types of timeout adjustment strategies, as listed in Table 2, are considered. A threshold-based elephant model that identifies the flows with total byte counts greater than 6K forwarded to controllers as elephants is used to classify elephants.

This section firstly evaluates the Pareto frontiers of the proposed problems with two of the three objectives. We do not directly solve the three-objective optimization problems due to the difficulty in presenting the three-dimension Pareto frontiers. We consider to simultaneously optimize the elephant classification inaccuracy and control channel bandwidth usage, and the elephant classification time inefficiency and control channel bandwidth usage, respectively, because they conflict with each other. We do not simultaneously minimize the elephant classification time inefficiency and inaccuracy, because they are not conflicted. We also do the sensitivity analysis over the three objectives and discuss the further work.

5.1 Optimizing Classification Inaccuracy and Control Channel Bandwidth Usage

We firstly apply the three methods to solve the optimization problem that finds the best set of initial timeouts to minimize the elephant classification inaccuracy and control channel bandwidth usage. The approximation Pareto optimal fronts generated by the three methods are combined to form the final Pareto fronts. We consider 9 types of timeouts, and the final Pareto fronts are shown in Figure 1. Figure 1(a) to Figure 1(c) present three fronts, each of which combines the timeouts generated by the three methods. It is apparent that the fronts of timeout types 5, 6, and 9 have the better quality. It is noticed from Table 2 that the timeout types 5, 6, and 9 refer to the hard, idle, and hybrid timeouts with the value proportional increased after the flows are classified as elephants. As such types of adjustments keep the initial timeouts unchanged but proportional increase them with a given ratio (we let the ratio be 1.2) after the flows are identified as elephants, leading to a reduced control channel bandwidth usage and elephant classification inaccuracy. The other types of adjustment strategies do not able to effectively control the number of flow packets forwarded to controllers, and hence the control channel bandwidth usage.

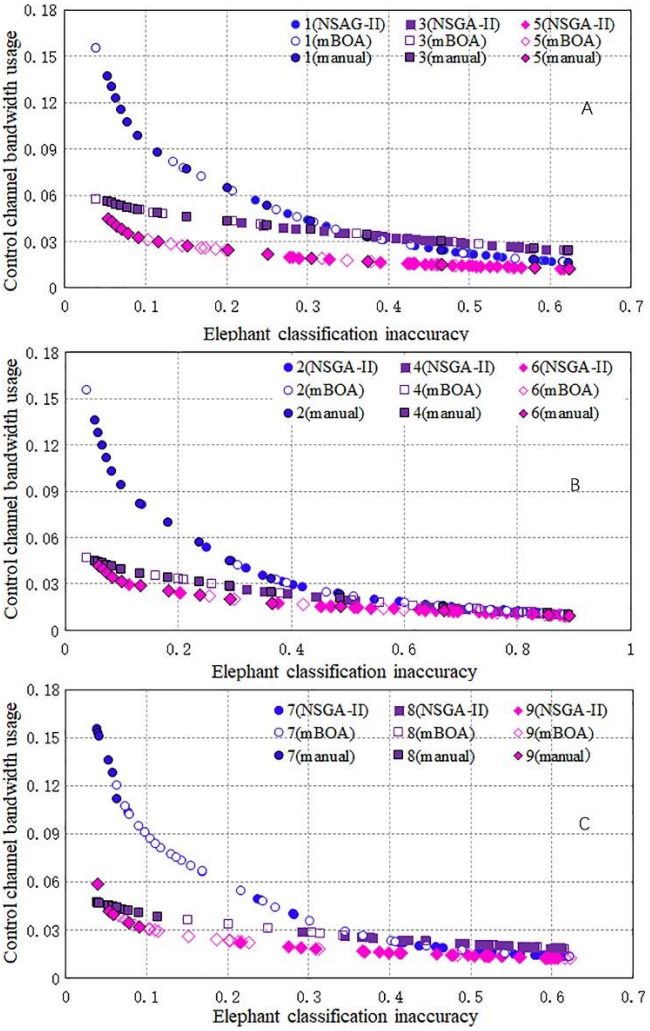

Figure 1. Pareto frontiers for the objectives of control channel bandwidth usage and elephant classification inaccuracy

It is also noticed that each Pareto front consists of the solutions generated by NSGA-II, MBO, and the given timeouts, represented by the solid, hollow, and solid with black frame icons in Figure 1, respectively. This indicates that all three methods cannot generate approximation Pareto fronts outperforming each other, although we have applied them several times and take the best solutions. It seems that MBO generates more solutions with lower elephant classification inaccuracy in the fronts than NSGA-II. However, we cannot determine the major reasons as two algorithms initial the population randomly, and more research should be done in our future research.

5.2 Optimizing Elephant Classification Time Inefficiency and Control Channel Bandwidth Usage

As similar as the fronts for elephant classification inaccuracy and control channel bandwidth usage, the fronts for elephant classification time inefficiency and control channel bandwidth usage also consist of the solutions generated by all NSGA-II, MBO, and the given timeouts, and each method cannot out perform the others in generating high quality approximation Pareto fronts. As shown in the Figure 2, the solid and hollow ones in the fronts represent the solutions

generated by NSGA-II and MBO, respectively, and the solid one with black frame represent the solutions generated by given timeouts. The types of adjustment strategies 5, 6, and 9 also outperform others, because increasing the value of timeouts after flows classified elephants effectively reduce the control channel bandwidth usage.

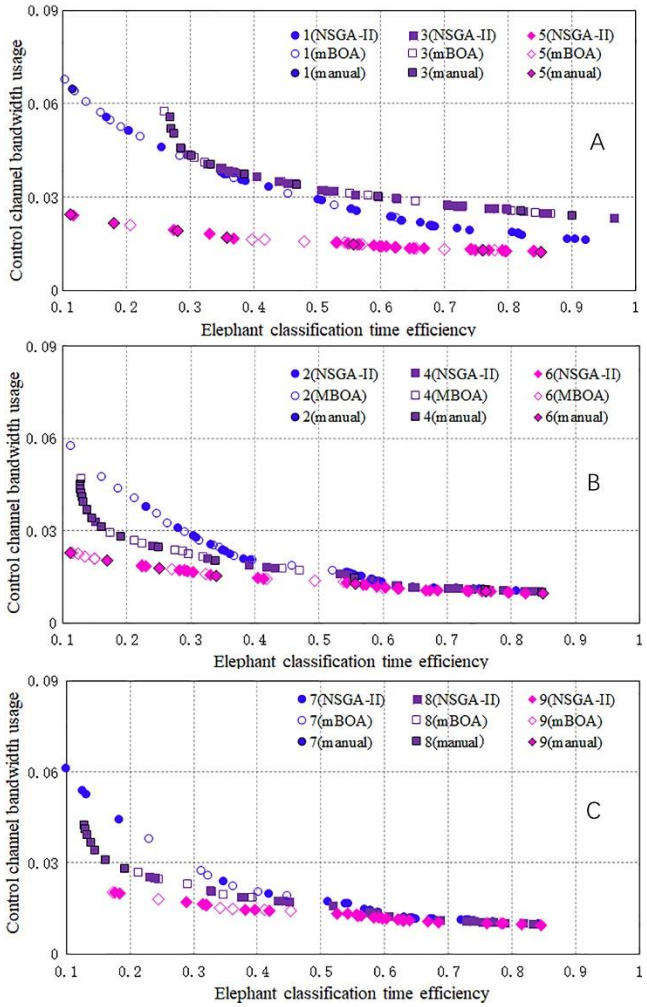

Figure 2. Pareto frontiers for objectives of control channel bandwidth usage and classification time inefficiency

5.3 Sensitivity Analysis

As the Pareto frontiers of the proposed optimization problems consist of multiple solutions and present the tradeoffs of objectives, this subsection evaluates how the initial value of timeouts affects the optimization of the objectives. As shown in Figure 3, the frontiers of the hybrid timeouts demonstrated a better quality than hard and idle timeouts increasing proportionally for objectives of control channel bandwidth usage and elephant classification inaccuracy, although the frontiers of types 9 and 6 have the similar quality for objectives of control channel bandwidth usage and elephant classification time inefficiency. We choose types 7, 8, and 9 for the sensitivity analysis. The timeout adjustment strategies 7, 8, and 9 refer to the hybrid type of timeout with value unchanged, updated based the mean variance of packet inter-arrival time of flows, and proportional increased after flows classified as elephants.

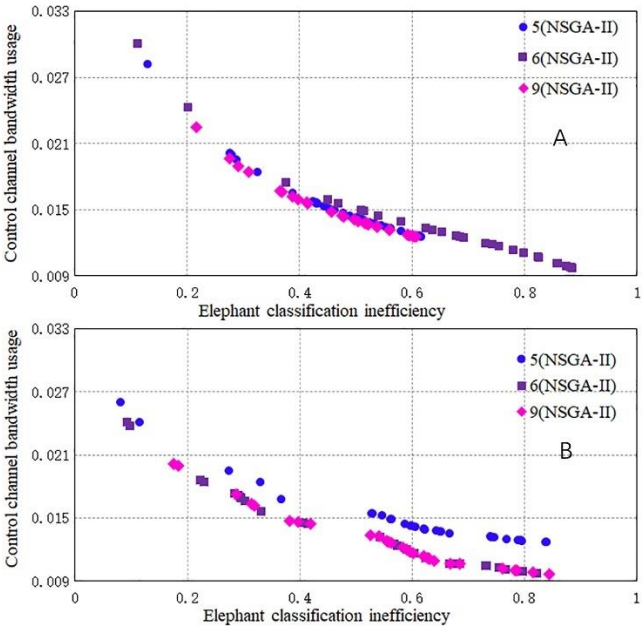

Figure 3. Pareto frontiers for the value of timeouts proportionally increased

Particularly, we let the initial values of timeout be 0.00001s, 0.00004s, 0.001s, 0.004s, 0.1s, 0.4s, 1s, 4s. We calculate the values of three objectives when the timeouts adjusted using strategies 7, 8, and 9. We use the values of objectives under strategy 7 with initial timeout value of 0.00001s as the baseline and calculate the ratio of other initial timeouts for each objective. The results are shown in Figure 4. Figure 4(b) has logarithmic scale for horizontal ax and is able to present the objective values under smaller initial values of timeouts more clearly.

It is apparent that under the three strategies the elephant classification inaccuracy and flow table size increased as the initial values of timeouts grew, but the control channel bandwidth usage decreased as the initial value of timeouts increased. This is because larger timeouts forwarded less packets of flows to controllers, leading to the reduced byte counts forwarded, the reduced control channel bandwidth usage, and the increased elephant classification inaccuracy. Timeouts with larger value increase the life time of flow entries in flow tables, leading to a larger flow table size. For each initial value of timeouts, the adjustment strategies 7, 8, and 9 demonstrated the same elephant classification inaccuracy, because they applied the same initial value to timeouts and maintained the value unchanged till the flows classified as elephants. After flows are classified as elephants, adjustment strategies 7, 8, 9 switched the hard timeout to idle timeout, however, strategy 7 remained the value of timeout unchanged, strategy 8 adjusted the value of timeout based on the mean and variance of packet inter-arrival time of flows, and strategy 9 increased the value of timeout with the ratio of 1.2. Therefore, strategy 8 costs the lest flow table size, strategy 7 has the higher control channel bandwidth than strategy 9 when the initial values of timeouts are small (less than 0.1s, as shown in Figure 4). Strategies 8 and 9 cost lower control channel bandwidth than strategy 7, because they forwarded less number of packets of flows to controllers. However, as the value of initial timeouts increased to larger than 1s, the control channel bandwidth usage under three strategies tends to be the same, because most of the elephants have a relatively small mean packet inter-arrival time, increasing the value of idle

timeouts when the value has been greater than the packet inter-arrival time does not really reduce the number of packets forwarded to controllers and the control channel bandwidth usage.

In general, all objectives are sensitive to short timeouts of flow entries. The objectives of elephant classification inaccuracy and flow table size are more sensitive than the objective of control channel bandwidth usage given the proposed timeout adjustment strategy. However, other adjustment strategies such as adjusting the value of timeouts before flows classified as elephants may perform very differently. More research should be done in our future research.

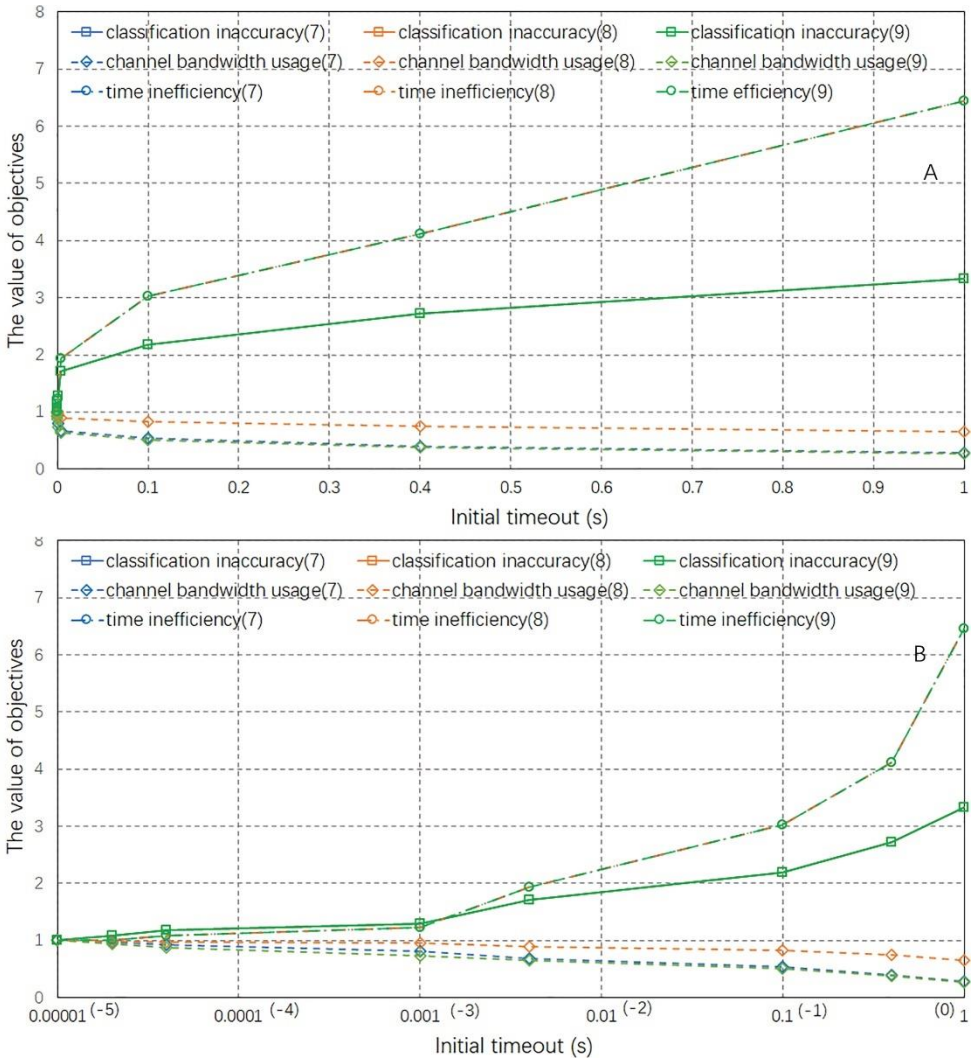

Figure 4. Sensitivity analysis, the value of objectives via the initial flow entry timeout

It is apparent that under the three strategies the elephant classification inaccuracy and flow table size increased as the initial values of timeouts grew, but the control channel bandwidth usage decreased as the initial value of timeouts increased. This is because larger timeouts forwarded less packets of flows to controllers, leading to the reduced byte counts forwarded, the reduced control channel bandwidth usage, and the increased elephant classification inaccuracy. Timeouts with larger value increase the life time of flow entries in flow tables, leading to a larger flow table size. For each initial value of timeouts, the adjustment strategies 7, 8, and 9 demonstrated the same elephant classification inaccuracy, because they applied the same initial value to timeouts and maintained the value unchanged till the flows classified as elephants. After flows are classified as elephants, adjustment strategies 7, 8, 9 switched the hard timeout to idle timeout, however, strategy 7 remained the value of timeout unchanged, strategy 8 adjusted the value of timeout based on the mean and variance of packet inter-arrival time of flows, and strategy 9 increased the value of timeout with the ratio of 1.2. Therefore, strategy 8 costs the lest flow table size, strategy 7 has the higher control channel bandwidth than strategy 9 when the initial values of timeouts are small (less than 0.1s, as shown in Figure 4). Strategies 8 and 9 cost lower control channel bandwidth than strategy 7, because they forwarded less number of packets of flows to controllers. However, as the value of initial timeouts increased to larger than 1s, the control channel bandwidth usage under three strategies tends to be the same, because most of the elephants have a relatively small mean packet inter-arrival time, increasing the value of idle timeouts when the value has been greater than the packet inter-arrival time does not really reduce the number of packets forwarded to controllers and the control channel bandwidth usage.

In general, all objectives are sensitive to short timeouts of flow entries. The objectives of elephant classification inaccuracy and flow table size are more sensitive than the objective of control channel bandwidth usage given the proposed timeout adjustment strategy. However, other adjustment strategies such as adjusting the value of timeouts before flows classified as elephants may perform very differently. More research should be done in our future research.

6 Conclusions

This paper formulated a five-objective optimization problem that found the best flow entry timeouts to jointly minimize the elephant classification inaccuracy, time inefficiency, control channel bandwidth usage, network latency, and flow table size. As optimization of control channel bandwidth usage is to optimize network latency and optimization of elephant classification inaccuracy is to optimize flow table size, the number of objectives was reduced to three to simplify the problem. As flow entries may have hard, idle, and hybrid types of timeouts, while the value of timeouts can be static and dynamically adjusted, nine types of adjustment strategies were proposed. Under such strategies, the simplified optimization problem was converted to a three-objective optimization problem that found the best initial value of flow entry timeouts to jointly minimize the elephant classification inaccuracy and time inefficiency and control channel bandwidth usage. To provide a better quality of fronts, NSGA-II and MBO methods were applied to solve the problem. The fronts generated were combined, the nondomination solutions are chosen for the final fronts. The quality of fronts is further improved by combining the solutions with initial value of timeouts manually generated. A threshold-based elephant model was chosen for elephant classification. The solutions are evaluated over a real traffic trace. The results shew that the hybrid of hard and idle timeouts with the value proportionally increased can achieve fronts with better quality. All objectives are very sensitive to the short timeouts, especially the classification accuracy, time efficiency, and flow table size. More adjustment strategies should be investigated to jointly minimize the three objectives in our future research.

Acknowledgements

This work was supported by the National Natural Science Foundation of China through the Grants 61962016 and Guilin University of Aerospace Technology, China, through the Grants XJ22KT20.

References

[1] U. Deshpande, N. Rajesh, S. Dhananjaya, Survey of SDN Traffic Flow Classification Approaches, *INFOCOMP Journal of Computer Science*, Vol. 20, No. 1, pp. 1-16, June, 2021.

[2] L. X. Liao, H.-C. Chao, M.-Y. Chen, Intelligently Modeling, Detecting, and Scheduling Elephant Flows in Software Defined Energy Cloud: A Survey, *Journal of Parallel and Distributed Computing*, Vol. 146, pp. 64-78, December, 2020.

[3] K. Kirkpatrick, Software-Defined Networking, *Communications of the ACM*, Vol. 56, No. 9, pp. 16-19, September, 2013.

[4] L. X. Liao, X. Ma, Z. Li, H. C. Chao, Dynamic Flow Entry Timeouts Based Packet Nonuniform Sampling for Elephant Flow Detection, *International Symposium on Computer Technology and Information Science*, Guilin, China, 2021, pp. 431-439.

[5] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, *IEEE transactions on evolutionary computation*, Vol. 6, No. 2, pp. 182-197, April, 2002.

[6] P. Galuzio, E. H. V. Segundo, L. S. Coelho, V. C. Mariani, Mobopt — Multi-Objective Bayesian Optimization, *SoftwareX*, Vol. 12, Article No. 100520, December, 2020.

[7] Y. Liu, B. Tang, D. Yuan, J. Ran, H. Hu, A Dynamic Adaptive Timeout Approach for SDN Switch, *2nd IEEE International Conference on Computer and Communications*, Chengdu, China, 2016, pp. 2577-2582.

[8] B. Isyaku, M. B. Kamat, K. B. A. Bakar, M. S. M. Zahid, F. A. Ghaleb, Ihta: Dynamic Idle-Hard Timeout Allocation Algorithm Based Openflow Switch, *IEEE 10th Symposium on Computer Applications & Industrial Electronics*, Peneng, Malaysia, 2020, pp. 170-175.

[9] R. Challa, Y. Lee, H. Choo, Intelligent Eviction Strategy for Efficient Flow Table Management in Openflow Switches, *IEEE NetSoft Conference and Workshops*, Seoul, Korea, 2016, pp. 312-318.

[10] Z. Guo, R. Liu, Y. Xu, A. Gushchin, A. Walid, H. J. Chao, Star: Preventing Flow-table Overflow in Software-Defined Networks, *Computer Networks*, Vol. 125, pp. 15-25, October, 2017.

[11] C. Wang, H. Y. Youn, Entry Aggregation and Early Match Using Hidden Markov Model of Flow Table in SDN, *Sensors*, Vol. 19, No. 10, Article No. 2341, May, 2019.

[12] B. Leng, L. Huang, C. Qiao, H. Xu, X. Wang, Ftrs: A Mechanism for Reducing Flow Table Entries in Software Defined Networks, *Computer Networks*, Vol. 122, pp. 1-15, July, 2017.

[13] X.-N. Nguyen, D. Saucez, C. Barakat, T. Turletti, Officer: A General Optimization Framework for Openflow Rule Allocation and Endpoint Policy Enforcement, *IEEE Conference on Computer Communications (INFOCOM)*, Hong Kong, China, 2015, pp. 478-486.

[14] J.-F. Huang, G.-Y. Chang, C.-F. Wang, C.-H. Lin, Heterogeneous Flow Table Distribution in Software-Defined Networks, *IEEE Transactions on Emerging Topics in Computing*, Vol. 4, No. 2, pp. 252-261, April-June, 2016.

[15] B. Yan, Y. Xu, H. J. Chao, Adaptive Wildcard Rule Cache Management for Software-Defined Networks, *IEEE/ACM Transactions on Networking*, Vol. 26, No. 2, pp. 962-975, April, 2018.

[16] G. Grigoryan, Y. Liu, M. Kwon, PFCA: A Programmable FIB Caching Architecture, *IEEE/ACM Transactions on Networking*, Vol. 28, No. 4, pp. 1872-1884, August, 2020.

[17] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue,

T. Hama, S. Shenker, Onix: A Distributed Control Platform for Large-Scale Production Networks, *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*, Vancouver, BC, Canada, 2010, pp. 351-364.

[18] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, G. Parulkar, Onos: Towards an Open, Distributed SDN OS, *3rd workshop on Hot topics in software defined networking*, Chicago, Illinois, USA, 2014, pp. 1-6.

[19] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, S. Banerjee, Devoflow: Scaling Flow Management for High-Performance Networks, *ACM SIGCOMM 2011 Conference*, Toronto, Ontario, Canada, 2011, pp. 254-265.

[20] M. Malboubi, L. Wang, C.-N. Chuah, P. Sharma, Intelligent SDN Based Traffic (de)Aggregation and Measurement Paradigm (iSTAMP), *IEEE Conference on Computer Communications*, Toronto, ON, Canada, 2014, pp. 934-942.

[21] Y. Afek, A. Bremler-Barr, S. L. Feibish, L. Schiff, Sampling and Large Flow Detection in SDN, *2015 ACM Conference on Special Interest Group on Data Communication*, London, UK, 2015, pp. 345-346.

[22] R. Cohen, E. Moroshko, Sampling-on-Demand in SDN, *IEEE/ACM Transactions on Networking*, Vol. 26, No. 6, pp. 2612-2622, December, 2018.

[23] J. Suh, T. T. Kwon, C. Dixon, W. Felter, J. Carter, Opensample: A Low-Latency, Sampling-based Measurement Platform for Commodity SDN, *IEEE 34th International Conference on Distributed Computing Systems*, Madrid, Spain, 2014, pp. 228-237.

[24] D. Tammaro, S. Valenti, D. Rossi, A. Pescape, Exploiting Packet-Sampling Measurements for Traffic Characterization and Classification, *International Journal of Network Management*, Vol. 22, No. 6, pp. 451-476, November/December, 2012.

[25] C. MadhusudhanaRao, M. M. Naidu, Flow Sampling for Network Intrusion Detection –an Acceptance Sampling Approach, *International Journal of Applied Engineering Research*, Vol. 13, No. 13, pp. 11030-11034, 2018.

[26] R. M. A. Ujjan, Z. Pervez, K. Dahal, A. K. Bashir, R. Mumtaz, J. Gonzalez, Towards Sflow and Adaptive Polling Sampling for Deep Learning Based DDoS Detection in SDN, *Future Generation Computer Systems*, Vol. 111, pp. 763-779, October, 2020.

[27] J. Zhao, N. Liu, Semi-Supervised Classification Based Mixed Sampling for Imbalanced Data, *Open Physics*, Vol. 17, No. 1, pp. 975-983, December, 2019.

[28] S. Shirali-Shahreza, Y. Ganjali, Delayed Installation and Expedited Eviction: An Alternative Approach to Reduce Flow Table Occupancy in SDN Switches, *IEEE/ACM Transactions on Networking*, Vol. 26, No. 4, pp. 1547-1561, August, 2018.

[29] G. Zhao, H. Xu, S. Chen, L. Huang, P. Wang, Joint Optimization of Flow Table and Group Table for Default Paths in SDNs, *IEEE/ACM Transactions on Networking*, Vol. 26, No. 4, pp. 1837-1850, August, 2018.

[30] S. Q. Zhang, Q. Zhang, A. Tizghadam, B. Park, H. Bannazadeh, R. Boutaba, A. Leon-Garcia, TCAM Space-Efficient Routing in a Software Defined Network, *Computer Networks*, Vol. 125, pp. 26-40, October, 2017.

[31] Z. Guo, Y. Xu, R. Liu, A. Gushchin, K.-Y. Chen, A. Walid, H. J. Chao, Balancing Flow Table Occupancy and Link Utilization in Software-Defined Networks, *Future Generation Computer Systems*, Vol. 89, pp. 213-223, December, 2018.

[32] M. K. Awad, M. El-Shafei, T. Dimitriou, Y. Rafique, M. Baidas, A. Alhusaini, Power-Efficient Routing for SDN with Discrete Link Rates and Size-Limited Flow Tables: A Tree-Based Particle Swarm Optimization Approach, *International Journal of Network Management*, Vol. 27, No. 5, Article No. e1972, September/October, 2017.

[33] J. Galan-Jimenez, J. Berrocal, J. L. Herrera, M. Polverini, Multi-Objective Genetic Algorithm for the Joint Optimization of Energy Efficiency and Rule Reduction in Software-Defined Networks, *11th International Conference on Network of the Future*, Bordeaux, France, 2020, pp. 33-37.

[34] V. Pereira, M. Rocha, P. Sousa, Hybrid IP/SDN Routing for Inter-Data Center Communications, *IEEE 8th International Conference on Cloud Networking*, Coimbra, Portugal, 2019, pp. 1-3.

[35] G. A. Ajaeiya, N. Adalian, I. H. Elhajj, A. Kayssi, A. Chehab, Flow-Based Intrusion Detection System for SDN, *IEEE Symposium on Computers and Communications (ISCC)*, Heraklion, Greece, 2017, pp. 787-793.

[36] Q. Xiang, J. Zhang, K. Gao, Y. Lim, F. Le, G. Li, Y. Yang, Toward Optimal Software-Defined Interdomain Routing, *IEEE Conference on Computer Communications (INFOCOM)*, Toronto, ON, Canada, 2020, pp. 1529-1538.

[37] P. Mercati, B. Li, M. Ergin, C. Tai, M. Kishinevsky, B. Serafimov, S. Ravisundar, E. Walsh, T. Long, Mobo-nfv: Automated Tuning of a Network Function Virtualization System Using Multi-Objective Bayesian Optimization, *IFIP/IEEE International Symposium on Integrated Network Management*, Bordeaux, France, 2021, pp. 90-98.

[38] A. S. D. Silva, C. C. Machado, R. V. Bisol, L. Z. Granville, A. Schaeffer-Filho, Identification and Selection of Flow Features for Accurate Traffic Classification in SDN, *IEEE 14th International Symposium on Network Computing and Applications*, Cambridge, MA, USA, 2015, pp.134-141.

[39]]N. Khan, D. E. Goldberg, M. Pelikan, Multi-Objective Bayesian Optimization Algorithm, *4th Annual Conference on Genetic and Evolutionary Computation*, New York City, USA, 2002, pp. 684-684.

Biographies

Changqing Zhao, senior lecturer with the School of Electronic Information and Automation of Guilin University of Aerospace Technology, China. His research interests include digital image/speech processing and network management and optimization.

Ling Xia Liao, professor with the School of Electronic Information and Automation, Guilin University of Aerospace Technology, China. Her research interests include intelligent network management and optimization, distributed systems, and edge computing.

Han-Chieh Chao, professor and chair of the Department of Electrical Engineering, National Dong Hwa University, Taiwan. His research interests include high speed networks, wireless networks, IPv6 based networks and digital divide.

Roy Xiaorong Lai, Co-Founder and Chairman of Confederal Networks Inc., Seattle, WA, USA. He is a SM of IEEE. His research interests includes wireless network, artificial intelligence algorithms and applications, and blackchain.

Miao Zhang, Professor with Quanzhou University of Information Engineering, China. His research interests include Internet of Things, distributed networks, blockchain, and artificial intelligence algorithms and applications.