DOI: 10.3969/j.issn.1673-5188.2017.03.004
http://kns.cnki.net/kcms/detail/34.1294.TN.20170918.1013.002.html, published online September 18, 2017

Evolutionary Algorithms in Software Defined Networks: Techniques, Applications, and Issues

LIAO Lingxia[1], Victor C. M. Leung[1], and LAI Chin-Feng[2]
(1. Department of Electrical and Computer Engineering, University of British Columbia, Vancouver BC V6T 1Z4, Canada;
2. Department of Engineering Science, National Cheng Kung University, Tainan, China)

Abstract

A software defined networking (SDN) system has a logically centralized control plane that maintains a global network view and enables network-wide management, optimization, and innovation. Network-wide management and optimization problems are typically very complex with a huge solution space, large number of variables, and multiple objectives. Heuristic algorithms can solve these problems in an acceptable time but are usually limited to some particular problem circumstances. On the other hand, evolutionary algorithms (EAs), which are general stochastic algorithms inspired by the natural biological evolution and/or social behavior of species, can theoretically be used to solve any complex optimization problems including those found in SDNs. This paper reviews four types of EAs that are widely applied in current SDNs: Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Simulated Annealing (SA) by discussing their techniques, summarizing their representative applications, and highlighting their issues and future works. To the best of our knowledge, our work is the first that compares the techniques and categorizes the applications of these four EAs in SDNs.

Keywords

SDN; evolutionary algorithms; Genetic Algorithms; Particle Swarm Optimization; Ant Colony Optimization; Simulated Annealing

1 Introduction

As various Internet-connected devices and advanced network applications gain popularity, the Internet traffic has become more and more complex in its data volume, data type, Quality of Service (QoS)/Quality of Experience (QoE) requirements, and security. This increased complexity creates a significant challenge in network management, which calls for flexible solutions through programmability of network devices. Conventional IP networks tightly couple the control logic on dedicated network devices, which are likely provided by diverse vendors, to configure, control, and monitor data flows. This makes it rather difficult for the network devices to cooperate and collect information on network dynamics that are changing in real time, to make decisions based on the network dynamics, and to enforce these decisions by automatically configuring or reconfiguring network devices. This motivates a new networking paradigm called software defined networking (SDN), which decouples network control from conventional network devices to form a logically centralized control plane, while a physically distributed data plane consisting of the network devices as data forwarders efficiently forwards the packets of individual data flows based on the rules generated by the control plane.

Software defined networks have a layered architecture consisting of a data layer, a control layer, and an application layer [1] (**Fig. 1**). The data layer forms a data plane that includes multiple simple network forwarders (switches) providing packet switching and forwarding, and also network statistics collec-

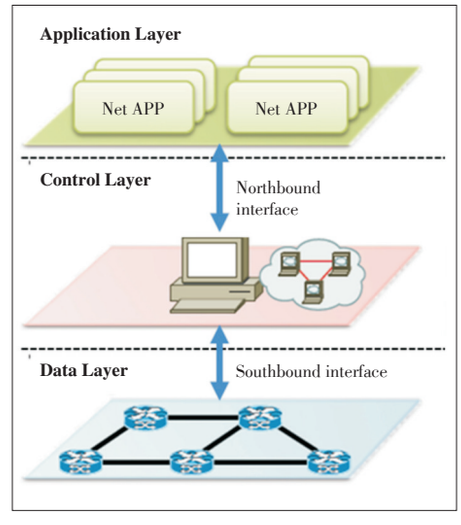


Figure 1. Layered architecture of an SDN system.

tion and reporting capabilities. The control layer is responsible for providing logically centralized control functionality for management of network nodes and flow forwarding. The application layer consists of end-user business applications that control switching devices by invoking the services in the control layer. The control layer and application layer form the SDN control plane. The SDN architecture provides an open standardized south-bound interface (e.g., OpenFlow protocol [2]) to manage the communications between the data and control layers. However, the north-bound interface between the control and application layers and the east-to-west-bound interfaces between the controllers inside the control layer are defined only functionally but not standardized. With these interfaces, a software defined network can maintain a logically centralized global network view in its control layer, allow each network application in the application layer to retrieve data from the global network view, and enforce network-wide management or security policies to optimize network performance, security, and resource usage. SDN makes a perfect architecture to enable network-wide optimization, artificial intelligence (AI), and machine learning mechanisms to form an open, customizable, programmable, and manageable network.

Network-wide optimization, AI, and machine learning problems typically are complex with huge search spaces, large numbers of variables, and multiple objectives. Many algorithms have been proposed to solve these problems in various user scenarios, among which evolution algorithms (EAs) are attractive candidates. EAs are stochastic algorithms inspired by the natural biological evolution and/or social behavior of species [3]. EAs typically have three major processes [4]: the initialization process, evaluation process, and new population generation process (**Fig. 2**). The initialization process randomly generates initial individuals, each of which represents a problem solution directly or indirectly as a string consisting of multiple elements, each element being a variable of the problem. In the evaluation process, each solution is evaluated for its fitness against the objectives so that the solutions with higher fitness value will be selected to feed into the new population generation process to generate a new population set for next iteration. Various EAs employ different ways to generate initial populations, evaluate fitness, and generate new populations.

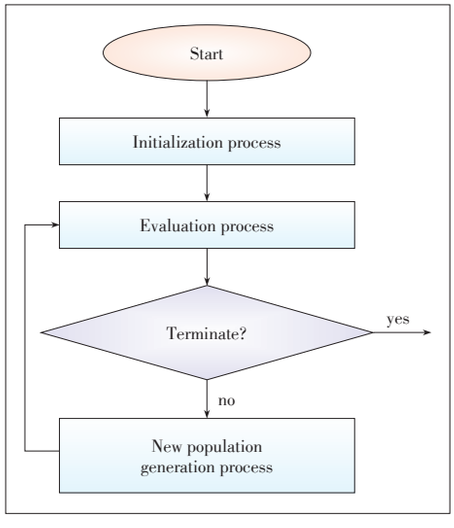


Figure 2. General flow chart of EAs.

As algorithms for solving complex optimization problems, EAs are general and can adapt to an unknown environment and autonomously decide the parameters of a dynamical optimization problem. By exploiting the diversity of solutions, an EA finds out the best solutions in a set of population with high fitness values and evolve to the next generation. Thus EAs can provide easily implementable scalability for solving a wide range of single- and multi-objective optimization problems [5]. These features motivate the growing interest in applying EAs in SDNs or many other complex systems.

This paper reviews four widely used types of EAs: Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Simulated Annealing (SA). We compare their formulation and characteristics and provide a brief survey of their application in SDNs. The goal of this paper is not to provide an exhaustive survey, but to highlight the features of these EAs, provide a representative sample of the important work, and point out the research issues and challenges in applying EAs in SDNs. Though there have been some surveys on EAs [5], [6] or on SDNs [7]–[9], to the best of our knowledge, our work is the first to summarize the features of these four EAs and review their application in SDNs.

The rest of this paper is organized as follows. Section 2 reviews the related literature. Section 3 introduces the four types of EAs. Section 4 summarizes the applications of these EAs in SDN networks. Section 5 highlights the potential issues and challenges. Conclusions are drawn in Section 6.

2 Review of Related Works

2.1 Surveys on EAs

EAs have been a hot research topic for many years and applied to solve complex problems in a large range of science and engineering fields. Many surveys on EAs have been published in the recent literature. Elbeltagi in [10] compares the formulation and performance of five EAs: GAs, memetic algorithms, PSO, ACO, and shuffled frog leaping. Crepinšek in [11] reviews nearly 100 existing papers since 2013 and summarizes how EAs do exploration and exploitation to achieve close-to-optimal solutions over a short convergence time. Von Lücken in [12], Mukhopadhyay in [13], and Cheshmehgaz in [14] review the application of EAs to solve multiple-objective optimization problems. EAs in data mining are reviewed in [13] and [15], distributed EAs are reviewed in [16], hybrid EAs are reviewed in [17], and approaches to optimize the performance of EAs are reviewed in [18]–[20]. Mehboob's recent survey on the applications of GAs in wireless networks [5] is similar to this paper

in summarizing the techniques and applications of EAs and pointing out the major issues and challenges in designing and applying EAs, but it focuses on GAs in wireless networks while this paper reviews the technologies and applications of PSO, ACO, SA as well as GAs for SDN networks.

2.2 Surveys on SDNs

Nunes in [6], Kreutz in [7], and Robertazzi in [8] and many more researchers have presented comprehensive surveys on SDN architecture, protocols, implementations, tools, and applications. Some works are focused on particular issues of SDNs; for instance, security issues have been reviewed by Yan [21], Scott-Hayward [22], and Farhady [23], network virtualization on SDN architecture has been reviewed by Blenk [24] and Jain [25], and flow table optimization has been reviewed by Jain [26]. Other works are focused on combining SDN with other networking technologies; for instance, the architecture, implementation, and applications of wireless SDNs have been summarized in [27], approaches enabling mobile SDNs are summarized in [28], application of SDN in wireless sensor networks is considered in [29], and application of SDN to cloud networks is studied in [30]. The most relevant work to our survey is [9], which reviews the applications of AIs in SDN networks, while our survey is focused on EAs, a subset of AI algorithms, and reviews the algorithms, the applications, and the issues of EAs in SDN networks.

3 Evolutionary Algorithms

EAs are the computational systems that mimic the efficient behavior of species and seek fast and robust solutions for complex optimization problems. They are the stochastic algorithms and can be used to find out the approximate optimal solutions for NP-hard optimization problems. GAs are the earliest EAs introduced by Holland in [31] in 1975. Later on other types of EAs are developed. As described in the introduction section, EAs typically have the same three processes: the initialization process, the evaluation process, and the new population generation process. In the rest of this section, we choose four major EAs widely used in current SDNs: GAs, PSO, ACS, and SA, and discuss how they can form their own initialization, evaluation, and new population generation processes.

3.1 GAs

GAs are inspired by biological genetic evolution that selects the individuals with best fitness values to generate offspring through crossover and mutation operations. As shown in **Fig. 3**, its new population generation process is split into selection, crossover, and mutation sub-processes. In the initialization process, GAs have to randomly generate a population set, each population represents a solution of the optimization problem. GAs often use a string (chromosome) consisting of a number of elements (genes) to represent a solution. The evaluation process defines a fitness function against the objectives to evaluate the fitness of a solution. The selection is used to form a parent set to feed into crossover and mutation functions. With the fitness value of solutions evaluated, the selection sub-process ranks the solutions and the ones with the higher fitness values form the parent set for offspring generation. The selection simulates the survivor of the fittest: the chromosomes with higher fitness are selected with higher probabilities to generate offspring. Crossover sub-process generates a child by mixing the genes of two parents and the mutation sub-process generates a child by randomly changing some of the genes in chromosomes. On one hand, crossover operation exploits the best traits of the current chromosomes, and strong chromosomes (with higher fitness) are more likely to be selected as parents, and hence there is a big chance that the new chromosomes may become similar after several generations, and the diversity of the population may decline and lead to population stagnation. On the other hand, mutation operation explores chromosomes to discover new traits. It injects diversity into the population and avoids the population stagnation. Crossover along with mutation provides the necessary "evolutionary mix of small steps and occasional wild gambles" to facilitate robust search in complex solution spaces [31]. Selected individuals are genetically modified to form the next generation of the population, the usage of crossover and mutation and stochastic selection allow a gradual improvement in the fitness of the solution and allow GAs to keep away from local optima.

Applying GAs to solve an optimization problem has to firstly encode problem's solutions into chromosomes. There are many ways to encode a solution. **Table 1** lists almost all the popular encoding ways proposed in current research, and among them, binary encoding is the earliest encoding method and has been widely used in GAs, but it generates many chro-

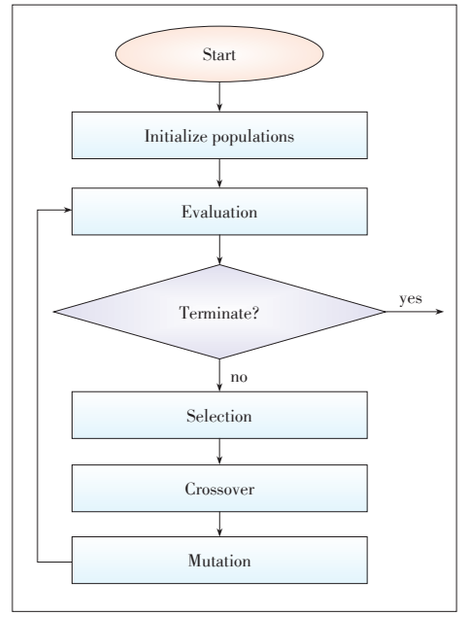


Figure 3. Flow chart of GAs.

mosomes even for a problem with small search space and may not suitable for some optimization problems. Permutation encoding is typically designed for optimizing ordering problems such as traveling salesman problems or job scheduling problems. Tree encoding is particularly used for expressions or evolving programming languages. Value encoding is useful in dealing with optimization problems when the variables are in real numbers or other complicated values, but the crossover and mutation functions may have to be updated to make the real numbers workable.

Fitness functions are corresponding to the objectives. They are typically the objective functions of optimization problems. A fitness function provides a mechanism to evaluate the solutions of a problem. Since the fitness function is utilized by each population to evaluate its fitness at each iteration, carefully designing the fitness function can reduce the convergence time of a GA and hence improve the performance of a GA.

3.1.1 Major Parameters of GAs

A GA has a number of parameters. It firstly needs to determine the population size and the maximum generations. The population size is the number of populations a GA has to maintain in a generation, while the maximum number of iterations is the maximum number of loops the algorithm can run. A GA also has to choose a selection function, crossover function, and mutation function.

A selection function often develops a particular way to select the members with higher fitness value to form a parent set for later crossover and mutation. Different selection approaches may require different methods to generate and assign probability to make sure the diversity and the improvement of the new populations. As listed in Table 1, the typical selection mechanisms used in current GAs include: 1) relative tournament evaluation; 2) roulette wheel selection; 3) relative pooling tournament evaluation; and 4) elitism. Relative tournament evaluation randomly chooses two members from the current population set, and the one with the higher fitness value is selected as a parent for next generation. Roulette wheel selection picks parents from individuals based on their fitness values, the higher the fitness value an individual has, the larger probability an individual has to be chosen as a parent. Relative pooling tournament evaluation throws members of the current population set in a competition, and the winner of the competition will be chosen as parents. Elitism maintains an archive that records all the populations that have been considered so far, and the individuals with the better fitness values in the archive are chosen as parents. Roulette wheel selection and relative pooling tournament evaluation are widely used.

As listed in Table 1, the typical crossover methods consist of: 1) one-point crossover; 2) two-point crossover; 3) uniform crossover; 4) cut and splice; and 5) ordered chromosome crossover. One-point crossover randomly generates a crossover point for two parents, swaps the genes before or after the crossover point to generate a new child chromosome. Two-point crossover randomly generates two crossover points for two parent, and the genes between these two points are swapped to produce new child chromosomes. Uniform crossover uses a fixed mixing ratio between two parents. Cut and splice crossover allows each parent to have its own choice in deciding crossover point. Ordered chromosome crossover consists of multiple crossover methods that change the chromosome by switching the position of genes, and is often used when a direct swap infeasible. Choosing crossover functions needs to make sure the new populations satisfy the constraints of the problems.

The typical mutation methods used in GAs consists of 1) bit-string mutation; 2) flip bit; 3) boundary; 4) uniform; and 5) Gaussian, as shown in Table 1. Bit-string mutation randomly flips the value of genes. Flip bit chooses genes from chromosomes to flip their values. Boundary replaces value of a gene with the upper or lower bound of the value. Uniform mutation replaces the score of the chosen chromosome with a uniform random value selected in the range of the user-specified bounds. Gaussian mutation adds a unit Gaussian distributed random value to the selected chromosome. The same as the crossover function, mutation function has to make sure the new populations satisfy the problem's constraints.

▼**Table 1. The major mechanisms in initialization, selection, crossover, and mutation of GAs**

Process	Mechanisms	Descriptions
Initialization	Binary encoding	A solution is a bit string with each element as 0 or 1
	Value encoding	A solution is a string with elements integers, real numbers, characters, or objects
	Permutation encoding	A solution is a sequence of number
	Tree encoding	A chromosome is a tree form of objects
Selection	Relative tournament	Two members are randomly chosen, and the parent is the one with higher fitness
	Routlette wheel	The probability an individual to be chosen as a parent is depended on their fitness
	Relative pooling tournament	Populations are thrown into a competition, and the winners are the parents
	Elitism	Populations with the higher fitness from all the populations generated so far from the parent set
Crossover	One-point	Randomly generate a crossover point
	Two-point	Randomly generate two crossover points
	Uniform	Use a fixed mixing ratio between two parents
	Cut and spice	Allow each parent to have its own choice in deciding crossover point
	Ordered chromosome	Switch the position of genes
Mutation	Bit-string	Randomly flip the value of genes
	Flip bit	Flip the value of selective genes
	Boundary	Replace the value of a gene with the upper or lower bound of the value
	Gaussian	Add a unit Gaussian distributed random value to the selected chromosome
	Uniform	Replace the value of a selective gene with a uniform random value within the user-specified bounds

3.1.2 Multi-Objective GAs

Multi-objective GAs (MOGAs) find out a Pareto optimal solution set for multi-objective optimization problems. An MOGA is also an

iteration procedure consisting of the same flow chart as a single objective GA, but uses different mechanisms in section, crossover, and mutation to find out the Pareto optimal solution set and maintain the diversity of the set. The first MOGA is introduced by Carlos M. Fonseca and Peter J. Fleming in [32]. It uses a section sub-process where each solution is assigned a rank, the dominated solutions have the rank of 1 and the non-dominated solutions have a higher rank based on the distance between them and a dominated solution. N. Srinivas and Kalyanmoy Debin in [33] propose a MOGA named the none-dominated sorting genetic algorithm (NSGA) based on several layered classification of the solutions. Deb in [34], [35] extends this NSGA to the NSGA-II that initializes a population set, ranks and sorts each population according to none-domination level, creates new pool of offspring by applying crossover and mutation operations, and then combines the parents and offspring before partitioning the new combined pool into fronts. The NSGA-II conducts niching by adding a crowding distance to each member. It uses this crowding distance in its selection sub-process to maintain the member diversity and make sure each member stays a crowding distance apart. This keeps the population diverse and helps the algorithm to explore the fi tness landscape. NSGA-II has been the most widely used MOGA in current research.

3.2 PSO

PSO is proposed by James Kennedy and Russell Eberhart in [36] in 1995 and since then applying PSO to solve different complex optimization problems has been a hot research topic. PSO is inspired by the migrating birds to reach unknown destination. When birds migrating, each bird looks in a specific direction and finds out the migration route through identifying the bird in the best position. When applying PSO to solve an optimization problem, each solution of the problem is a "bird" or referred to a "particle". The population of solutions is a swarm of particles. Each particle has a velocity corresponding to its current place. Once a particle reaches a new position, the best position of each particle (the local best position) and the best position of the whole swarm of particles (the global best position) are updated. The new velocity of a particle corresponding on the new global best position and the local best position can be calculated.

The basic flow chart of a classical PSO algorithm is shown in **Fig. 4**. It is similar as GAs to have an initialization process where initial swarm of particles is randomly generated. Each particle has its velocity and position. Then the evaluation process is to evaluate the fitness of each particle. Each time a particle gets evaluated, its fitness value is compared to the fitness of the global best position and the local best position. If the fitness value is better than the fitness of current global or/and local best positions, the current global or/and local best positions are updated accordingly. The velocity and position of the whole swarm are updated to generate a new swarm for the next iteration until the stopping criterion is met.

The velocity of a particle is calculated using (1), where V^c and V^l are the current and last velocity respectively, P^l is the last position of a particle, L_{best} and G_{best} are the local best position of a particle and the global best position of the whole swarm, w, c_1, c_2 are parameters, and R_1 and R_2 are random variables ranging from 0 to 1. The formula calculating velocity of a particle represents the process that involves social interaction and intelligence so that each particle can learn from their own experience (the local best position) and also from the experience of other particles (the global best position). In (1), w represents the inertia of a particle, c_1 and R_1 represent how much experience needs to be learned from the local best positions, and c_2 and R_2 represent the experience learn from the global best positions. Then the new position of a particle is the sum of the current velocity calculated by (1) and the last position of the particle using (2).

$$V^c = wV^l + c_1 r_1\left(L_{best} - P^l\right) + c_2 r_2\left(G_{best} - P^l\right), \quad (1)$$

$$P^c = P^l + V^c. \quad (2)$$

3.2.1 Major Parameters of PSO

A PSO algorithm often has the following parameters: the number of particles, the w, c_1 and R_1, and c_2 and R_2. A particle in PSO is analogous to a population in GAs. However, PSO fixes the number of particles and adjusts the movement of each particle toward the destination through social behavior, while GAs randomly initialize the populations and then generate new populations for next evolution iteration. Since PSO does not need to sort the fitness of particles in any process, and the movement of each particle in a PSO is guided by the local best position and the global best position, PSO algorithms of-

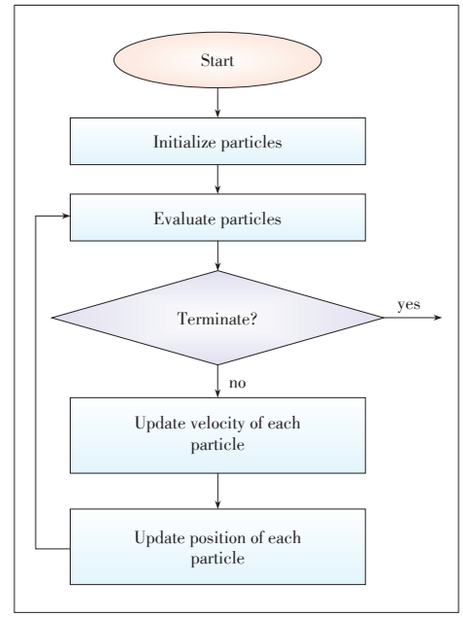


Figure 4. Flow chart of PSO.

ten have short convergence time.

3.2.2 Multi-Objective PSO

Classical PSO algorithms have no mechanisms to maintain the diversity of particles, they are often used to find the global optima for an optimization problem with single objective. However, great efforts have been made to extend classical PSO to solve multi-objective optimization problems. The typical mechanism used by PSO to maintain the diversity of solutions is to maintain two external archives: one for storing the leaders currently being used for performing the movement and the other for storing the final solutions. A crowding factor is used for the selection of non-dominance solutions from the solution archive to form a Pareto frontier for a multi-objective optimization problem [37]. Many more variable multi-objective PSO algorithms have been proposed [3], [38].

3.3 Ant Colony Optimization

ACO is a set of combinational optimization algorithms inspired by ants search for food and depositing pheromone on the route. When ants leave their nest to search for a food source and meet an obstacle, they randomly choose to the right or left directions to go forward. Then the ants in the direction with a shorter distance find a food source and carry the food back and deposit pheromone along their route. The following ants most likely choose the path with the larger amount of pheromone to find a food source. Since the more pheromone a route has, the more ants the route has been taken by, and the higher probability this route has shorter distance between the nest and a food source. Over time, this positive feedback process prompts all ants to choose the shorter path. The amount of pheromone on the ground influences the behavior of ants, the path with the largest amount of pheromone represents the shortest path between the nest and a source of food.

3.3.1 Ant System

The original ACO is the ant system (AS) proposed by Colorni, Dorigo, and Maniezzo in [39]–[41] and used to optimize the traveling salesman problem [42]. To apply AS to solve such a problem, one solution is represented as an ant, which is often a string. Each element of the string is a variable with a value within the given set. **Fig. 5** shows the flow chart of the AS.

AS often starts with generating m random ants and evaluate the fitness of each ant corresponding to an objective function, and then updates the pheromone concentration of each possible trail (variable value) using the following formula:

$$\tau_{ij}(t)=\rho\tau_{ij}(t-1)+\Delta\tau_{ij}, \tag{3}$$

where i is the variable of an ant; j is the option that the value of variable i choose; let l_{ij} be the value of variable i; T is the maximal number of iterations and t is a particular iteration; $\tau_{ij}(t)$ is the revised concentration of pheromone associated with l_{ij} at iteration t, $\tau_{ij}(t-1)$ is the concentration of pheromone at the previous iteration t-1; $\Delta\tau_{ij}$ is the change in pheromone concentration; and ρ is the pheromone evaporation coefficient with value ranging from 0 to 1 to avoid too strong influence of the old pheromone so that premature solution stagnation is incurred. $\Delta\tau_{ij}$ is the sum of the contributions of all ants associated with l_{ij} at iteration t, and can be calculated using (4):

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\tau_{ij}^{k}, \tag{4}$$

$$\Delta\tau_{ij}^{k}=\sum_{k=1}^{m}\begin{cases}\dfrac{R}{fitness^{k}} & if\ option\ l_{ij}\ is\ \text{chosen by ant}\ k\\ 0\ \text{otherwise}\end{cases}, \tag{5}$$

where m is the number of ants and $\Delta\tau_{ij}^{k}$ is the pheromone concentrate laid on value l_{ij} by ant k. $\Delta\tau_{ij}^{k}$ can be calculated by (5) with R being the pheromone reward factor and $fitness^{k}$ being the value of the objective function for ant k.

Once the pheromone is updated, each ant has to update its route respecting the pheromone concentration and also some heuristic preference. The ant k at iteration t will change the value for each variable according to the following probability:

$$P_{ij}(k,t)=(\tau_{ij}(t)^{\alpha}\times\eta_{ij}^{\beta})/(\sum_{l_{ij}}^{m}\tau_{ij}^{\alpha}\times\eta_{ij}^{\beta}), \tag{6}$$

where $P_{ij}(k,t)$ is the probability that option l_{ij} is chosen by ant k for variable i at iteration t; $\tau_{ij}(t)$ is the pheromone concentration associated with option l_{ij} at iteration t; η_{ij} is the heuristic factor for preferring among available options and is an indicator of how good it is for ant k to select option l_{ij}; and α and β are exponent parameters that specify the impact of trail and attractiveness, respectively, and take values greater than 0.

3.3.2 Ant Colony System

AS is the first ACO algorithm motivated by real ants and

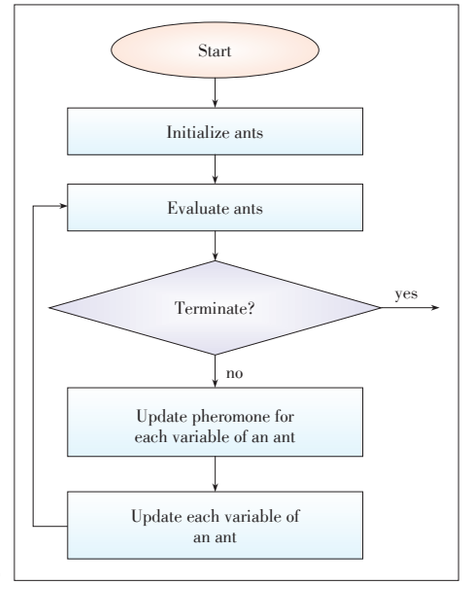


Figure 5. Flow chart of AS.

used to solve traveling salesman problem [42], but its performance cannot compete against the state-of-art heuristic algorithm. Later on, Gambardella and Dorego extend the AS algorithm to be Ant-Q algorithm [43] to improve its performance. The Ant-Q algorithm is further simplified as Ant Colony System (ACS) [44], which becomes the base of many ACO algorithms [45]–[47].

ACS has the same flow chart as AS. The major difference between them is that AS updates the pheromone of a trail using all the pheromone contributed by all the ants, and ACS updates the pheromone of a trail using the following formula,

$$\tau_{ij}(t)=\rho\tau_{ij}(t-1)+(1-\rho)\tau_0, \quad (7)$$

where ρ is still the pheromone evaporation coefficient with the value ranging from 0 to 1 (usually set to 0.9), and τ_0 is the initial pheromone value and can be defined as $\tau_0=(n\times L_{mn})^{-1}$, where L_{mn} is the tour length produced by execution of one ACS iteration without the pheromone concentrate.

The pheromone updating of AS is a globally updating and intends to increase the attractiveness of promising route, while the updating of ACS simplifies the search in a neighborhood of the best tour found so far of the algorithm, and it is local and more effective and can avoid long convergence time.

Once the route updating for each ant is done, each ant chooses its next move by choosing a random probability ρ (which is often fixed to 0.9). With the probability $(1-\rho)$ the next move is chosen randomly with a probability based on η_{ij}^{α} (α often equals to 1) and τ_{ij}^{β} (β often equals to 2), and with the probability ρ_0 the next move is chosen with a probability calculated by (5).

3.3.3 Major Parameters of ACO

The main parameters of a ACO algorithm include the number of ants m, number of iterations t, the attractiveness of a trail η_{ij}, the exponents α and β, the pheromone evaporation rate ρ, pheromone reward factor R, and the probability ρ_0.

3.3.4 Multi-Objective ACO

Angus and Woodward in [48] have reviewed a large collection of multi-objective ACO (MOACO) algorithms. The multi-objective ACO algorithms can be classified by how they make the choice of using single or multiple pheromones and pheromone update or decay, and by how these choices affect the performance of the algorithm as well. Some approaches implicitly or explicitly weight their multiple objectives in some kind of preferential order, and this approach can outperform the alternative pareto-based MOACO for some particular problems, but generating a Pareto optimal solution set so that a decision maker can make his choice based its own strategy is more general. The existing MOACO algorithms seeking the Pareto frontier may use single pheromone matrix or benefit from multiple pheromone matrix.

3.4 Simulated Annealing

Simulated annealing (SA) is a stochastic local search approach motivated by the behavior of physical systems in a heat bath. Local search is an approach that attempts to improve on a solution by a series of incremental and local changes. By given an initial solution, a local search algorithm defines a method that performs loops to find out the optimal solution in the neighborhood of the given initial solution, and expects that the local optima is the global optima. When a solid is put in a heat bath, it firstly raises the temperature of the solid to a point where its atoms can randomly move and then to lower the temperature to force the atoms to rearrange themselves into a crystallization state that minimizes energy of the system at a lower energy state. Carefully selecting the cooling schedule allows a solid to become a crystal that has the lowest energy instead of an amorphous state with higher energy. Since the solution of an optimization problem can be viewed as a solid in a heat bath, the cost of the objective function can be viewed as the energy of a solid, the optimal solution can be viewed as the ground energy state of a solid, moving a solution to a neighboring position can be viewed as the rapid quenching, and the control mechanism adopted by the search algorithm can be viewed as the cooling schedule. In this way, a simulated annealing algorithm can be developed to mimic the physical annealing process of physical material.

3.4.1 Simulated Annealing Algorithm

The SA algorithm is first proposed by Kirkpatrick, Gelatt, and Vecchi in [49]. The basic flow chart of the SA algorithm is shown in **Fig. 6**, where S is a given initial solution, S' is a randomly chosen neighbor of S in the initialization process. In the evaluation process, SA algorithms compare the difference of cost(S') and cost(S) (the cost function is similar as the fitness function in GAs, PSO, and ACO to evaluate how good a solution is in its solution space). In the new population generation process, the SA checks the cost difference between S' and S, generates new S, S', and T based on the cost difference between current S, S', and T.

3.4.2 Major Parameters of SA

The major parameters of a SA algorithm consist of the maximum number of iterations to apply the algorithm, the randomly generated neighbor solution, the temperature T, and the cooling ratio R ranging from 0 to 1.

3.4.3 Multi-Objective SA Algorithms

The key issue in extending SA to solve multi-objective optimization problems is to determine how to calculate the probability of accepting a dominated solution [50]. The initial approach introduced by Serafini in [51] proposes a target-vector approach. Given solutions S and S' randomly generated in the

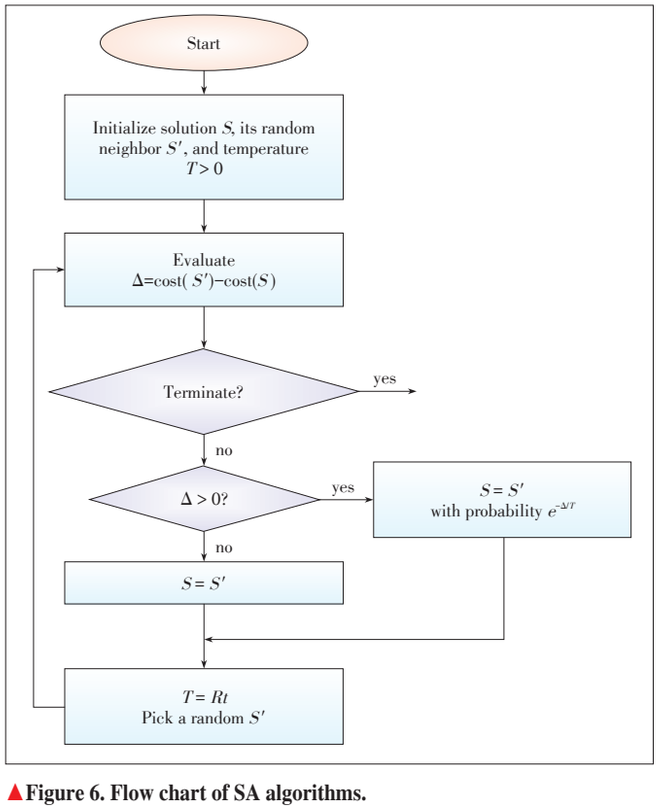
neighborhood of S, if S' is non-dominated, it is accepted as the next S and the non-dominated solution set is updated. Serafini's approach allows to maintain an archive of non-dominated solutions so that the generation of several members of the Pareto optimal set in a single run can be calculated, but only the local non-dominance is used to fill up the archive of solutions and a further filtering procedure is required to reduce the number of non-dominated solutions without sacrificing the diversity of the Pareto optimal solutions. Given S and the temperature T, $P(S',S,T)$ (the probability of accepting S' as a non-dominated solution) can be calculated as the following:

$$P(S',S,T)=Min\left(1,\ e^{Max_j}\left(\lambda_j\left(\cos t_j(S)-\cos t_j(S')\right)/T\right)\right), \quad (8)$$

where the weights λ_j are initialized to one and modified during the search process, and j is the particular iteration during the search process. Based on this approach, many more simulated annealing algorithms for multi-objective optimization problems are developed [52]–[54]. Simon in [3] and Solís in [55] give a comprehensive review on multi-objective SA algorithms.

3.5 Algorithm Summary

As a summary, the discussed 4 types of algorithms can be categorized into EAs with the same general components in Fig. 2 and the need to encode solutions and ensure the generated solutions satisfying the constraints. Each type of the algorithms has its way to initialize a population set. GAs, PSO, and ACO initialize their population set randomly, while AS has to choose an initial solution and then randomly generate another solution inside the neighborhood of the initial one. Each type of the algorithms needs to define a fitness function (cost function in SA) to evaluate the quality of the solutions. GAs evaluate the quality of each individual in the population set, rank and sort the populations according to their fitness number, choose the ones with the higher fitness to be parents for generating offspring; PSO and ACO simply evaluate the quality of each particle and ant, respectively; and SA evaluate the cost difference between the initial solution and its neighbor solution. To generate a new set of population, GA uses crossover and mutation operations; PSO calculates the velocity and updates the position of its particles; ACO updates the pheromone of each trail and computes the trail of each ant will take; and SA calculates the probability of accepting the neighbor solution as the new initial solution and updates new temperature for next iteration.



▲**Figure 6. Flow chart of SA algorithms.**

In general, GA is the only type of EAs that ranks the solutions, and hence the convergence time of GAs increases nonlinearly as the population size grows. PSO uses floating point arithmetic to compute velocity and position of a particle. It can generate any potential values of velocity and may lead to high density of members in population set. Since the best positions of a particle and the whole swarm of particles guide the movement of a particle, PSO has short convergence time but may incur premature convergence. ACO is efficient for traveling salesman or similar problems. It can be used in dynamic applications or adapts to the changing environment. However, ACO has uncertain convergence time and it is hard to get theoretical analysis. SA can be used in many optimization problems. SA outperforms all the existing approximation algorithms in graph partitioning problem but suffers poor performance in number partitioning problem. SA algorithms have long convergence time and are not suitable for online optimization problems [56]. **Table 2** shows the major features of each described EA.

4 Applications of EAs in SDN

EAs have been applied in SDN in a wide variety of contexts. We have searched the major applications published since 2013, and categorized them into routing, load balancing, controller placement, security, virtual network mapping, flow entry optimization, and hybrid network migrating. We list the major applications in **Table 3** and discuss the representative applications in each category in the rest of this section.

4.1 Routing

The current routing strategy that forwards a packet along the shortest path between its source and destination cannot always achieve the shortest network delay in a highly dynamic network, because this shortest path may suffer heavy work load or get congested. With a logically centralized control plane that frequently updates the dynamical global network states, SDNs

can develop many flexible routing strategies besides only using the shortest path. Since calculating a routing path is based on the huge volume of network statistics, finding out the optimal routing path in an acceptable time in a dynamical network environment is highly demanded. Therefore, an exhaustive algorithm is not time acceptable, and EAs are widely used in recently years to produce approximately optimal solutions in a short time. According to the objectives that a routing strategy wants to optimize, routing algorithms can be categorized to: 1) avoiding link congestion and balance link usage; 2) saving link power; and 3) maintaining QoS or QoE.

4.1.1 Avoidance of Link Congestion and Balancing of Link Usage

Liu in [57] presents a GA to solve the bandwidth-constrained multi-path optimization problem in a SDN. The proposed GA is implemented in a Floodlight [58] controller that manages a SDN network emulated by Mininet [59] for evaluation. The results indicate that the proposed GA can globally, flexibly, and effectively find out a routing path that minimizes the network delay under bandwidth constraint in a multi-path SDN network.

Ren in [60] develops a GA to solve a traffic scheduling problem in switch congestion control. It samples the link utilization ratio, and feeds it into the GA to optimize the switch traffic assignment to avoid link congestion. It also implements the GA in Floodlight controller over a Mininet emulated SDN with data flows generated by Iperf [61] for evaluation. The results show that the proposed GA is able to make the link arrangement utilization more balanced and reasonable.

Maniu in [62] applies GAs to compute the routing path for network flows so that the dynamic resource allocation can be optimized to enable a self-adaptive network with history extrapolation. The chromosome of the GAs is a sequence of nodes representing a routing path from source to destination. Each gene has a value representing the node ID in the network, and the length of the chromosome is the number of nodes in the routing path. Since the length of route is variable, the remaining locus in chromosome is completed with value 0. The Fitness function is a sum of link costs in a route. The evaluation shows the proposed GAs can provide an approximate optimal routing path to save the link cost efficiently.

Kikuta in [63] proposes an effective parallel GA to optimize explicit routing when using general purpose programming on graphic processing unit (GPU) in a SDN. The parallelism of GA consists of the search methods of GA itself, the calculation of each fitness function, and the evaluation of the network congestion ratio. It takes advantage of the multi-core processor for acceleration of graphic processing, and presents 10 times faster than the original parallel GA on GPU, and 9 times faster than the conventional CPU computation when enforcing explicit routing in a SDN.

Stefano in [64] introduces A4SDN for traffic engineering in software-defined networks. A4SDN is based on the alienated ant algorithm, a stochastic-based heuristic approach used to solve combinatorial and multi-constraint optimization problems. Based on artificial ant' behavior, the A4SDN forces the ants to distribute themselves over all the available paths rather than converge to a single one when searching for food. Using this strategy enables an autonomic dynamic routing and leads to a better exploitation of the network bandwidth for best effort data packets. A comparison between A4SDN with two Dijkstra-based shortest path routing solutions shows that A4SDN can guarantee a higher throughput together with a lower packet loss rate and network delay.

Wang in [65] adopts an ACO algorithm to route the network traffic to avoid the network congestion in traffic scheduling. The ACO algorithm is applied to dynamically adjust the calculation parameters of the routing algorithm. The proposed algorithm is compared to the traditional equal cost multi-path routing algorithm, and the results show that the ACO algorithm reduces link or node congestion and effectively improves the link utilization rate.

▼Table 2. Major features of GAs, PSO, ACO, and SA

	GA	PSO	ACO	SA
Motivation	Natural evolution	Bird migration	Ant search food	Solid in heat bath
Population size	Any size	Several	Any size	Typically two
Iterations	Yes	Yes	Yes	Yes
Single-objective	Yes	Yes	Yes	Yes
Multi-objective	Yes	Yes	Yes	Yes
Parallelism	Inherent parallelism	Extended	Inherent parallelism	Extended
Convergence	Slow	Fast	Uncertain	Slow
Solution quality	Near global optima	Near global optima	Near global optima	Near local optima
Applications	General	General	TSP, routing, dynamic and adaptive	General
Theoretical analysis	Hard	Hard	Hard	Hard

ACO: Ant Colony Optimization PSO: Particle Swarm Optimization TSP: Traveling Salesman Protocol
GA: Genetic Algorithm SA: Simulated Annealing

4.1.2 Saving Link Power

The routing application can be optimized for power efficiency by routing flows to minimize the number of links activated. Most of the energy saving strategies on current IP network only aggregate traffic into some links, which leads to imbalance link utilization and seriously impacts the QoS.

Zhu in [66] takes advantage of the centralized control and global vision of a SDN to achieve the network energy saving and load balancing by dynamically aggregating and balancing the traffic while ensuring QoS. It adds actual QoS constraints to the basic maximum concurrent

▼Table 3. Applications of EAs in SDNs

Application	Category	EA	Description
Liu in [57]	Routing	GA	Avoid link congestion
Ren in [60]		GA	Avoid switch congestion
Maniu in [62]		GA	Optimize link usage
Kikuta in [63]		Parallel GA	Optimize explicit routing in GPU
Stefano in [64]		ACO	Optimize network bandwidth usage
Wang in [65]		ACO	Avoid link congestion
Zhu in [67]		PSO	Energy saving
Subbiah in [68]		PSO	Finding the best node connector in switches for energy saving
Awad in [69]		PSO	Energy saving under the constraint of size-limited flow table
Dobrijevic in [70]		ACO	QoE aware routing
Tang in [71]		ACO	QoE aware routing
Blaguer in [72]		GA	QoE aware routing
Santl M in[73]		ACO	QoE aware routing
Kang in [74]	Load balancing	GA	Controller load balancing
Chou in [75]		GA	Controller load balancing
AMR in [77]		GA	ink load balancing
Sathyanarayana in [76]		ACO	Controller and link load balancing
Lin in [72]		ACO	Controller load balancing
Lange in [79]	Controller placement	SA	Minimize swi-to-con delay and controller load imbalance
Sanner in [80]		GA	Minimize swit-to-con delay
Jalili in [81]		GA	Minimize con-to-con delay and controller load imbalance
Ahmadi in [82]		GA	Minimize swi-to-con delay, con-to-con delay, controller load imbalance
Gao in [83]		PSO	Minimize swi-to-con delay considering controller capacity
Liu in [84]		PSO	Minimize swi-to-con delay and controller load imbalance
Li in [86]	Security	GA	Detect DDos attacks
Li in [87]		GA,PSO	Detect DDos attacks
Chen in [88]		ACO	Detect DDos attacks
Liu in [89]		ACO	Detect DDos attacks
Ojugo in [90]		GA	Security rule generation
Zhao in [91]		GA	Intrusion action detecting
Bouet in [92]		GA	Single security appliance placement
Famaluddine in [93]		GA	Multiple security appliances placement
Li in [94]	Virtual network mapping	PSO	Optimize network resource usage
Yao in [95]	Flow table optimization	PSO	Optimize flow table usage
Gao in [96]		ACO	Optimize flow table usage
Li in [97]		ACO	Optimize flow table usage
Guo in [98]	Hybrid SDN migration	GA	Migrate routers in hybrid network

ACO: Ant Colony Optimization
DDoS: distributed denial of service
EA: evolutionary algorithm
GA: Genetic Algorithm
GPU: graphic processing unit
PSO: Particle Swarm Optimization
QoE: Quality of Experience
SA: Simulated Annealing

flow problem to formulate a multi-objective mixed integer programming model and proposes a multi-objective PSO algorithm called MOPSO to solve this NP-hard problem.

Subbiah in [67] proposes a PSO based energy aware routing algorithm at open virtual switches. It finds out the best node connectors in the switches from source host to destination host to reduce the path energy consumption. It improves the networking performance compared to the conventional one.

Awad in [68] considers a routing optimization problem for energy saving with a set of practical constraints consisting of the size-limited flow table and discrete link rate. It develops a low-complexity PSO based and power efficient routing heuristic algorithm to solve this problem. Performance evaluation results indicate that the proposed algorithm achieves more than 90% of the optimal network power consumption while requiring only 0.0045% to 0.9% of the computation time in real network topologies.

4.1.3 QoS and QoE

Dobrijevic in [69] applies an ACO approach to flow routing in SDN environments. Based on the global network view and the flexible configuration provided by a SDN, the approach estimates the QoE and seeks to optimize the user QoE for multimedia services. As different service has its QoE affected by network metrics such as packet loss and network delay differently, it proposes an ACO-based heuristic algorithm, based on the service type and its typical integral media flows, to calculate the best routing paths that aware QoE and conform to network limitations and traffic demands. The algorithm is integrated into an OpenDaylight controller, and the evaluation results indicate the proposed approach has promising QoE improvements and lower running time over shortest path routing.

Tang in [70] applies an improved ACO algorithm to the calculation of routes, meeting QoS requirements through obtaining the network topology, resources usage, and network statistics from the controller. If the bandwidth of a single path cannot meet the bandwidth requirement of an application, it can aggregate multiple paths and distribute the work load to these multiple paths to maintain the QoS of the application.

Blaguer in [71] applies a GA to find a routing path for a stream from a source to a target node in a SDN to maximize the concurrent streams without degrading the QoE. Since multimedia streams have to satisfy a certain maximum latency requirements and a minimal bandwidth requirement, finding the optimal routing path that satisfies the constraints costs time, especially in a large scale network. The GA offers an acceptable approximate optimal solution with short convergence

time for a large scale of network.

Santl M in [72] applies ACO for QoE-centric flow routing in SDNs. It views a SDN as a weighted graph with a QoE measure as the ultimate metric. QoE depends on QoS and is expressed in terms of delay, jitter, and packet loss rate. Achieving good QoE often has to satisfy combination of multiple QoS metrics. It associates weights to the graph nodes based on values of delay and packet loss rate for each network device. The delay sums up delay of each node on a path, and a packet loss rate is calculated by $1-\prod \in pa\,\mathrm{th}\left(1-lossRate\left(node\right)\right)$. When applying ACO, it specifies an ant type for a flow type, and multiple ants of the same type are sent from the flow source to the destination in iterations, tracking estimated QoE and seeking to maximize the final result. The evaluation is conducted in a SDN emulated by Mininet with Floodlight controller, the proposed ACO indicates promising QoE improvements over shortest path routing as well as low convergence time.

4.2 Load Balancing

In SDN, load balancing can be categorized into two types: controller load balancing and link load balancing. Controller load balancing is a dynamical optimization problem and balances the load of controllers based on the real time changed network state. Link load balancing is often achieved by using ACO algorithms.

Kang in [73] and Chou in [74] propose similar load balancing strategies that use a way to monitor the load of each controller, once controller load imbalance is detected, a GA is applied to generate new switching assignment to balance the controller load.

AMR in [75] also applies a GA to balance the load of each controller corresponding to a set of workload. The performance of the proposed GA outperforms the random and round robin methods.

Sathyanarayana in [76] applies an ACO algorithm to select the best path to reach a controller with the least load to balance the load of both controllers and the paths leading to the controllers. The proposed approach is implemented as a load balancing module in the SDN controller. This module uses resource usage of the controller and the network statistics collected by the controller to find both the best server and the best path for network flows.

Lin in [77] proposes a dynamic load balancing approach based on an ACO algorithm with combined job classification in a layered control plane consisting of a root controller and multiple lower layer controllers. The root controller monitors the state of the whole network, and each lower layer controller manages a subset of network. Every job in the network is firstly sent to the root controller to decide which subset of network and which lower layer controller should take the job based on the job demand for CPU performance, then the job is sent to the corresponding lower layer controller to run the ACO algorithm to calculate the best routing path that minimizes the link load based on the dynamic network load provide by the root controller.

4.3 Controller Placement

Each new flow in a SDN suffers a flow setup delay since each switch in the data plane has to involve the control plane to setup new flows for them. This flow setup delay affects how fast a SDN can forward a new flow and how many new flows the control plane can set per second, and hence limits the network performance and scalability. Therefore, a large scale network or a geographically wide area network may need a logically centralized but physically distributed control plane with multiple controllers to provide a short flow setup delay anticipated by the network. This creates a controller placement problem that is firstly introduced by Heller in [78] to optimize the location of controllers so that the switch-to-controller delay can be minimized in a wide area of SDN. Later on the controller placement problem is extended to a multi-objective optimization problem to minimize the controller-to-controller delay, controller load imbalance, and many other objectives for both wide area of SDNs or data center SDNs.

An exhaustive algorithm can be used to find the global optimal solution in a small scale network with very limited number of controllers, but may not be time acceptable in a large scale network or as the number of controllers in the network increases. Heuristic algorithms have been proposed to solve a particular problem in a particular scenario, but EAs are more general and can solve general optimization problems with approximately optimal solutions in an acceptable computation time.

Lange in [79] proposes a SA algorithm to provide Pareto optimal controller placements to minimize switch-to-controller delay and controller load imbalance under a given maximal controller-to-controller delay. This application is the most representative sample that applies SA in SDN networks.

Sanner in [80] uses a GA to find out the controller placement so that the switch-to-controller latency can be minimized. It encodes a controller placement as a chromosome, but the encoding method is not clearly provided. It only optimizes one objective, and the performance of proposed GA is good comparing to Integer Linear Programmer.

Jalili in [81] applies a multi-object GA to find out the Pareto optimized controller placement so that the controller-to-controller delay and controller load imbalance can be optimized. In particular, it applies NSGA-II with each chromosome representing a controller placement and each gene of the chromosome representing the ID of node in the network. The evaluation is done on finding appropriate placements with 6 controllers for the Internet2 OS3E network topology.

Ahmadi in [82] develops a hybrid NSGA-II to solve the controller placement problem that minimizes the switch-to-controller delay, the controller-to-controller delay, and the controller load imbalance. Comparing to a typical NSGA-II, this hybrid NSGA-II improves the population initialization process by add-

ing an improved local search to generate better solutions in its population set. This hybrid NSGA-II also develops a hybrid crossover function that enforces path-relink strategy and cross-controllers-operator.

Gao in [83] proposes a PSO to find out the optimal controller placement that minimizes the switch-to-controller delay with consideration of the controller capacity. It uses multiple particles, and each particle represents a controller placement consisting of k variables; each variable identifies the position of a controller in the network. The velocity and position of a particle are randomly generated and keep improved using the local best position of the particle and the global best position of the whole swarm of particles. The performance of the proposed PSO is compared to a greedy algorithm and an integer linear programming algorithm, and the results show the proposed PSO performs rapidly and effectively.

Liu in [84] implements a network clustering PSO algorithm to find the best controller placement to minimize the multi-objective optimization problem in SDNs. The proposed PSO takes consideration of the controller capacity, switch-to-controller delay, and the controller load balancing, and formulates an optimization problem that finds out the controller placement to maximize the utilization ratio of each controller and to minimize the switch-to-controller delay under the constraint of a maximal controller load imbalance. Each particle is encoded as a string consisting of n elements representing n switches in a network. The value of an element of the string indicates the position of a controller in the network. The proposed PSO combines a clustering mechanism to calculate the velocity and position of a particle. The evaluation is based on real topology and work load and shows the algorithm´s effectiveness.

4.4 Security

Internet service providers and equipment vendors are subject to many security threats. One of the most prevalent security threats is the distributed denial of service (DDoS) attack, where the attack traffic and attacker's IP address are respectively difficult to detect and trace, because attack traffic is similar to regular traffic and the attack is executed by multiple attackers. An intrusion detection system is a type of security software designed to automatically alert administrators when someone or something is to compromise information system through malicious activities or through security policy violations. Applying EAs in the security of SDN networks can be categorized into 1) detecting attack and 2) optimizing security appliance placement.

4.4.1 Detecting Distributed Denial-of-Service Attacks

EAs can be used in an intrusion detection system to detect the attack traffic. Based on the past behavior, a profile of normal behavior consisting of multiple attributes such as service types, flags, and logs can be created [85]. EAs can detect the unseen patterns and find out the malicious traffic based on this profile, but detecting a new attack is difficult.

Li in [86] proposes a cross validation-GA to enable a support vector machine classification with optimized punish parameter c and the kernel function parameter λ in DDoS attack detection. The proposed algorithm performs better than a typical support vector machine, a clustering model, and a BP neural network model.

Li in [87] combines BP, PSO, and GA to develop a particle swarm BP neural network algorithm for DDoS attack detection. The evaluation shows the proposed algorithm can achieve high detection rate with low miss report rate and convergence time.

Chen in [88] proposes a novel distributed DDoS attack detection and identification framework using an ACO based metaheuristic approach for low-rate DDoS attacks. The proposed framework consists of three stages: a multi-agent algorithm, an information heuristic rule, and a search method. The proposed framework's time and space complexity is compared to the PSO and probabilistic packet marking. The evaluation shows the proposed framework can solve the problems in using other algorithms and demonstrates better performance than existing methods.

Liu in [89] proposes a random routing mutation based on an improved ACO algorithm to change the data transmission routing dynamically to avoid DDoS attack and improve network security. The ACO algorithm is used to find out the optimal routing path that has a minimal number of overlapping nodes compared to the recently used routing paths and meet the load balancing needs of the entire network.

Ojugo in [90] applies a GA to classify network audit data so that a better signatures for rule based intrusion detect system can be created. A chromosome is a rule consisting of 7-feats, each of which is a fixed length vector including one or more genes of different types. The goodness of chromosomes is evaluated. If the chromosome correctly classifies an attack, it is considered good; otherwise it is bad and not selected for crossover to produce offspring. Therefore, the more attacks a chromosome detects, the higher fitness value. Applying GA to generate the security classification rule reduces security experts from rule creation. The rule generation can speed up and counter new attacks.

Zhao in [91] applies a clustering GA to solve the intrusion detection problem. The proposed clustering GA algorithm consists of a clustering step and a genetic optimization step. It does not only automatically cluster cases, but also can detect unknown intrusion actions. Its overall accuracy can reach up to 95% with a very low false alarm rate.

4.4.2 Security Appliance Placement

Current IP networks need large-scale adoption of security appliances to improve the whole network security and to protect the information privacy. Security appliances can be virtualized and dynamically deployed as pieces of software on commodity hardware. Deploying such software security appliances

is costly in terms of license fees and power consumption. Designing cost effective security appliance deployment strategies that meet the security operational constraints is thus mandatory for the adoption of this approach.

Bouet in [92] introduces a security appliance deployment problem that minimizes the number of security appliance and the network load under the management constraints such as the maximal number of security appliance engines, the maximal usable link bandwidth, and the maximal unallocated flows. This optimization problem is a multi-objective optimization with conflicted objectives. Reducing the number of security appliances tends to potentially increase the distance of the paths between the source of a flow and a security appliance, and the usage of link bandwidth along the paths as well. Minimizing the link bandwidth usage increases the number of security appliances to be deployed. Bouet only deploys one security appliance and develops a GA to solve this problem, while Famaluddine in [93] extends this problem to multiple security appliances and solve the same security appliance deployment problem that minimizes the network load and the number of appliances using GAs.

4.5 Virtual Network Mapping

Multi-tenant cloud network needs to map a tenant virtual network onto the physical network substrate to provide resource and information isolation among tenants. To maximize the resource usage without degrading the service level of a tenant virtual network, a virtual network mapping problem can be formulated to maximize the network resource usage under the constraints of tenant virtual network service requirements, network resource limitation, management limitation, and more. Li in [94] proposes a virtual network mapping model based on a PSO algorithm. It proposes a virtual network framework, where each tenant virtual network is controlled by tenant's own controller. The proposed PSO algorithm has better performance than a shortest path algorithm in improving the utilization of network bandwidth.

4.6 Flow Table Optimization

SDNs need to involve their control plane to generate forwarding rules at switches according to the management policy. Network-wide optimization policy often has to be enforced by injecting many local forwarding rules at corresponding switches. However, the tenary content addressable memory (TCAM) used to store the forwarding rules in SDN switches is limited resources. This creates a problem of optimizing the flow table usage without affecting the network-wide management.

Yao in [95] investigates this problem and formulates it as a bounded forwarding-rules maximum flow (BFR-MF) problem, and solves it by applying an improved PSO algorithm. This improved PSO keeps updating the position of particles to maximize the overall feasible traffic. The fairness among flows is maintained to guarantee the QoS requirements of flows. Extensive simulations show that the improved PSO algorithm performs well in optimizing network utilization.

Gao in [96] formulates this problem as an mixed integer linear programing problem that optimizes the TCAM resources under the QoS constraint of flows in a multiple uni-cast session SDN. This problem is actually a routing rule space occupation problem that finds out the best switches to store routing rule without sacrificing the QoS of flows. An ACO algorithm is applied to solve this problem and demonstrates an expected performance in evaluation.

Li in [97] formulates this problem as a BFR-MF problem as similar as Yao in [95], but applies an improved ACO algorithm to optimize the flow table usage with the performance and the level of QoS of flows guaranteed. The simulation indicates that the proposed ACO performs better network utilization under the constraints of QoS in data center networks.

4.7 Hybrid SDN Migration

Though some companies have moved their inter-connecting data centers to fully SDN-enabled networks, many more companies have to look for an incremental deployment of SDN devices in its network due to the existing economical, technical, and organizational challenges. This implies that over a long period of time, a hybrid network consisting of the conventional IP network devices and Openflow enabled network devices is existed. Since the same network protocols can run over a conventional network and a SDN, there is no big technical problem in forming and running such a hybrid network. However, it creates an optimization problem that migrates legacy routers to SDN-enabled routers to maximize the network usage with minimized investment.

Guo in [98] investigates this problem and formulates it as an optimization problem that finds out the optimal migration sequence of routers under the consideration of the traffic engineering performance and the investment. It tries to utilize the potential of network resources in traffic engineering to reduce routers that need to be migrated, and to avoid investing more budgets to migrate more routers that may make little gain for traffic engineering. It applies GA to calculate the migration sequence that minimizes the link usage. The GA uses the permutation of routers as its chromosome, and the fitness function is the sum of the link usage. Once a router is migrated it remains unchanged in the following migration iterations for network stability. The GA approach obtains a better performance compared to a static algorithm or a greedy algorithm in migration.

5 Issues and Challenges

5.1 Requirements of Applying EAs

EAs can theoretically solve any optimization problems with two requirements to be met: 1) encoding candidate solutions and 2) generating fitness function to evaluate solutions. EAs re-

quire the solutions of problems to be represented as some kind of expression, for example, the chromosome of a GA, the particle of a PSO, the ant of an ant colony optimization, or the solid of a simulated annealing. These encodings are often represented as a string, each element of which represents a variable with bounded value set for the given problem. The number of the elements in the string represents the number of variables in the given problem and the bound value set of each variable represents the constraints for each variable. Applying EAs also needs a way to evaluate solutions—the fitness function. There is no way to evaluate the rightness of a fitness function. However, the fitness score is necessary for indicating how good a candidate solution is. As long as the encoding scheme and fitness function are developed, there is really no restriction on the types of problems that EAs can solve.

5.2 Tasks That Fit EAs

EAs can be used to solve combinatorial optimization problems and have been used to solve them in a big range of fields, for instance, traveling salesman problems, job shop scheduling problems, vehicle routing problems, multi-commodity distribution network design problems, multi-mode resource constrained project scheduling problems, warehouse design problems, and many other more. GAs and SA algorithms are very general, and can be used in a distributed or paralleled manner to solve complex optimization problems with large search space and multiple conflicted objectives. PSO is mainly used to solve unconstrained, single-objective optimization problems though many mechanisms have been developed to allow a PSO algorithm to support constrained problems or maintain the diversity of solutions for multi-objective optimization problems. Unlike GAs and SA algorithms can optimize problems with variables representing nodes and links of a graph, ACO is typically used for problems that optimize links of a graph, such as travelling salesman problem and network routing problems, and solve them in a dynamical manner. ACO algorithms often run continuously and can adapt to the real time changed environments. The improved ACO can also tackle a problem with unknown features, which makes ACO can be used in data mining, data analysis, and classifying malicious flows and detecting attacks in security problems.

5.3 Implementation Issues

The discussed four EAs have the same three major processes in general as shown in Fig. 2. Therefore, when applying these EAs to solve optimization problems, the first important task is to determine how to represent the solution of the problem. Since GA provides a large flexibility to encode its chromosome, it is important to choose the encoding with less number of genes to reduce search space and easy to enforce constraints. The second task in the implementation of EAs is to maintain the diversity of the populations so that the solution space can be explored globally and near globally optimized solutions can be found. Lack of diversity in the populations often incurs a permature converged solution and results in a local optimal solution rather than a global optimal one. The re-initialization strategy that adds new randomly generated populations to the current population set can avoid this premature convergence process that results in a local optimal solution and is highly recommended in PSO as well as GAs. The third task in implementing EAs is to improve the solution accuracy. Since EAs are approximately algorithms, we expect the solutions generated by them as close optimal as possible. Typically, increasing the size of population set of an EA can allow it to search wider solution space, but it is only suitable for GAs with a flexible population size that can be adjusted according to the solution space of a problem. However, since the density of the populations in the solution space is very little, it is often found that EAs cannot produce high quality solutions with high accuracy to the real global optima. Hybrid EAs can be a fix.

5.4 Open Issues and Future Work

EAs are very difficult to do theoretical analysis. Though EAs can be applied in many optimization problems, and have been shown that their solutions perform better than some other heuristic algorithms in particular circumstance, we cannot expect the same EAs can perform better than the heuristic algorithms on some problems outside the circumstance. One set of populations may bring solutions better than another set of populations, but we do not know why and how. An enhanced theoretical understanding of EAs are needed to increase user's trust in further widely applying EAs in various fields.

Most of EAs are used to tackle a problem with a solution string with up to tens of genes, and we have never seen an EA used to solve an optimization problem with very long solution string in practice. However, as SDN technology used in wireless sensor network, Internet of thing, and 5G mobile networks, the scale of a SDN based network becomes larger and larger. The number of network devices in such networks can increase to thousands or millions level. Calculating the routing path or optimizing the placement of some NFVs over the whole network or a subset of network may have to generate a solution string with hundreds of genes or even more. Solving such EAs with long solution strings and huge solution space creates many practical problems in algorithm designing, distributing and parallelism, and performance tuning.

Since different EAs have their own advantages and disadvantages, a single EA can hardly obtain a good enough solution. PSO is easy to be implemented and has a short convergence time, but suffers from partial optimism incurred by the lack of regulation in particle speed and direction. ACO is efficient for some particular types of optimization problems such as routing and link usage optimization, but suffers from uncertain convergence time and is really hard to get theoretical analysis. GA is inherent parallelism and good at maintaining diversity, but suffers from low accuracy and uncertain convergence time. Many

efforts have been put on combining two or three of EAs together to 1) reduce the convergence time, 2) improve the quality of solutions , and 3) incorporate EAs as part of a larger system.

6 Conclusions

EAs are stochastic algorithms inspired by the natural biological evolution and social behavior of species. They typically have the same flow charts consisting of three major components: population initialization, fitness evaluation, and new population generation; but adopt various mechanisms in each component. Although each discussed EA has its own advantage and disadvantage, EAs are general and versatile, and can be used for complex combinatorial problems in a wide variety of circumstances.

With the logically centralized global network vision, SDN's control plane makes a perfect place to develop and deploy network-wide management and optimization solutions, such as optimizing a routing path to avoid network convergence and balance link load, to maintain QoS and QoE, or to reduce link energy consumption; optimising the controller placement for distributed controllers; detecting the DDoS attacks or optimizing the security appliance placement; and optimizing flow tables usage, virtual network mapping, and router migration in a hybrid SDN. Such network-wide management and optimization problems are often complex. EAs can be applied to find out the near optimal solutions for them in an acceptable time.

There are basically two requirements that determine the feasibility of applying an EA to solve an optimization problem: 1) encoding a solution and 2) evaluating the quality of a solution. Any problems that meet these requirements can be solved by EAs. Many practical issues and open issues have been raised when applying EAs to slove such problems, especially regarding their applications in SDNs: theoretical analysis, dealing with the optimization problem for huge scale of SDN networks, and developing hybrid EAs to further improve the solution quality and convergence time.

References

[1] ONF, "Software-defined networking: the new norm for networks," Open Networking Foundation white paper, Apr. 13, 2012, retrieved Aug. 22, 2014.
[2] OpenFlow Switch Consortium. (2009). *OpenFlow Switch Specification Version 1.0. 0* [Online]. Available: http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf
[3] Dan Simon, *Evolutionary Optimization Algorithms*. Hoboken, USA: John Wiley & Sons, 2013.
[4] Daniel W. Dyer, Evolutionary computation in java, a practical guide to the watchmaker framework [Online]. Available: http://watchmaker.uncommons.org/manual/index.html
[5] U. Mehboob, J. Qadir, S. Ali, and A. Vasilakos, "Genetic algorithms in wireless networking: techniques, applications, and issues," *Soft Computing*, vol. 20, no. 6, pp. 2467–2501, 2016.
[6] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014. doi: 10.1109/SURV.2014.012214.00180.
[7] D. Kreutz, F. M. Ramos, P. E. Verissimo, et al., "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015. doi: 10.1109/JPROC.2014.2371999.
[8] T. G Robertazzi, "Software-defined networking," in *Introduction to Computer Networking*. New York City, USA: Springer, 2017, pp. 81–87.
[9] M. Latah and L.Toker, "Application of artificial intelligence to software defined networking: a survey," *Indian Journal of Science and Technology*, vol. 9, no. 44, 2016. doi: 10.17485/ijst/2016/v9i44/89812.
[10] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Advanced Engineering Informatics*, vol. 19, no. 1, pp. 43–53, 2005. doi: 10.1016/j.aei.2005.01.004.
[11] M. Crepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: a survey," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, article no. 35, 2013. doi: 10.1145/2480741.2480752.
[12] C. v̌on Lücken, B. Barán, and C. Brizuela, "A survey on multi-objective evolutionary algorithms for many-objective problems," *Computational Optimization and Applications*, vol. 58, no. 3, pp. 707–756, 2014. doi: 10.1007/s10589-014-9644-1.
[13] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, et al., "A survey of multiobjective evolutionary algorithms for data mining: part I," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 4–19, 2014. doi: 10.1109/TEVC.2013.2290086.
[14] H. R. Cheshmehgaz, H. Haron, and A. Sharifi, "The review of multiple evolutionary searches and multi-objective evolutionary algorithms," *Artificial Intelligence Review*, vol. 43, no. 3, pp. 311–343, 2015. doi: 10.1007/s10462-012-9378-3.
[15] P. Larrañaga, H. Karshenas, C. Bielza, et al., "A review on evolutionary algorithms in Bayesian network learning and inference tasks," *Information Sciences*, vol. 233, pp. 109–125, 2013. doi: 10.1016/j.ins.2012.12.051.
[16] Y. J. Gong, W. N. Chen, Z. H. Zhan, el al., "Distributed evolutionary algorithms and their models: a survey of the state-of-the-art," *Applied Soft Computing*, vol. 34 , pp. 286–300, 2015. doi: 10.1016/j.asoc.2015.04.061.
[17] R. C. Purshouse, K. Deb, M. M. Mansor, et al., "A review of hybrid evolutionary multiple criteria decision making methods," in *IEEE Congress on Evolutionary Computation (CEC)*, Beijing, China, 2014. doi: 10.1109/CEC.2014.6900368.
[18] D. J. Munk, G. A. Vio, and G. P. Steven, "Topology and shape optimization methods using evolutionary algorithms: a review," *Structural and Multidisciplinary Optimization*, vol. 52, no. 3, pp. 613–631, 2015. doi: 10.1007/s00158-015-1261-9.
[19] B. Kazimipour, X. Li, and A. K. Qin, "A review of population initialization techniques for evolutionary algorithms," in *IEEE Congress on Evolutionary Computation (CEC)*, Beijing, China, 2014. doi: 10.1109/CEC.2014.6900618.
[20] G. Karafotias, M. Hoogendoorn, and Á. E. Eiben, "Parameter control in evolutionary algorithms: trends and challenges," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 167– 187, 2015. doi: 10.1109/TEVC.2014.2308294.
[21] Q. Yan, F. R. Yu, Q. Gong, et al., "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 602– 622, 2016. doi: 10.1109/COMST.2015.2487361.
[22] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys & Tutorials*,18.1, pp. 623-654, 2016. doi: 10.1109/COMST.2015.2453114.
[23] H. Farhady, H. Y. Lee, and A. Nakao, "Software-defined networking: a survey," *Computer Networks*, vol. 81, pp.79– 95, 2015. doi: 10.1016/j.comnet.2015.02.014.
[24] A. Blenk, A. Basta, M. Reisslein, et al., "Survey on network virtualization hypervisors for software defined networking," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 655– 685, 2016. doi: 10.1109/COMST.2015.2489183.
[25] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, 2013. doi: 10.1109/COMST.2015.2489183.
[26] X. N. Nguyen, D. Saucez, C. Barakat, et al., "Rules placement problem in openflow networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp.1273–1286, 2016. doi: 10.1109/COMST.2015.2506984.
[27] N. A. Jagadeesan and B. Krishnamachari, "Software-defined networking paradigms in wireless networks: a survey," *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, article no. 27, 2015. doi: 10.1145/2655690.
[28] T. Chen, M. Matinmikko, X. Chen, et al., "Software defined mobile networks: concept, survey, and research directions," *IEEE Communications Magazine* ,

vol. 53, no. 11, pp.126–133, 2015. doi: 10.1109/MCOM.2015.7321981.
[29] D. Gante, A, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," in *27th Biennial Symposium on Communications (QBSC)*, Kingston, Canada, 2014. doi: 10.1109/QBSC.2014.6841187.
[30] Y. Jararweh, M. Al-Ayyoub, E. Benkhelifa, et al., "Software defined cloud: survey, system and evaluation," *Future Generation Computer Systems*, vol. 58, pp. 56–74, 2016. doi: 10.1016/j.future.2015.10.015.
[31] J. H. Holland, *Adaptation in Natural and A rtificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Cambridge, USA: MIT Press, 1992.
[32] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: formulation, discussion and generalization," in *Proc. Fifth International Conference on Genetic Algorithms*, San Mateo, California, pp.416–423, 1993.
[33] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.
[34] K. Deb, S. Agrawal, A. Pratab, et al., "A fast elitist nondominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Proc. Parallel Problem Solving from Nature VI Conference*, Paris, France, pp. 849–858, 2000.
[35] K. Deb, A. Pratap, S. Agarwal, et al., "A fast and elitist multiobjective genetic algorithm: NSGA–II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002. doi: 10.1109/4235.996017.
[36] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, Perth, Australia, 1995, pp, 1942–1948.
[37] M. R. Sierra and C. A. Coello Coello, "Improving PSO-Based multiobjective optimization using crowding, mutation and ∈-dominance," in *Third International Conference on Evolutionary Multi-Criterion Optimization.*, Guanajuato, Mexico, 2005, pp. 505–519.
[38] S. Lalwani, S. Singhal, R. Kumar, et al., "A comprehensive survey: applications of multi-objective particle swarm optimization (MOPSO) algorithm," *Transactions on Combinatorics*, vol. 2, no. 1, pp.39–101, 2013.
[39] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*. Cambridge, USA: MIT Press, 1992.
[40] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: an autocatalytic optimizing process," Politecnico di Milano, Tech. Rep. 91-016 revised, 1991.
[41] M. Dorigo, "Optimization, learning and natural algorithms," Ph. D. Thesis, Politecnico di Milano, Italy, 1992.
[42] L. M. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," in *Proc. IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 1996. doi: 10.1109/ICEC.1996.542672.
[43] M. Dorigo and L. M. Gambardella, "Ant-Q: a reinforcement learning approach to the traveling salesman problem," in *Proc. 12th International Conference on Machine Learning*, Morgan Kaufmann, USA, 2016, pp. 252–260.
[44] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997. doi: 10.1109/4235.585892.
[45] B. Bullnheimer, R. F. Hart, and C. Strauss, "A new rank based version of the ant system—a computational study," *Central European Journal for Operations Research and Economics*, vol. 7, no. 1, pp. 25–38,1997.
[46] T. Stutzle and H. H. Hoos, "Max-Min ant system," *Future Generation Computer System*, vol. 16, pp. 889–914, 2000
[47] M. Dorigo and T. Stutzle, *Ant Colony Optimization*. Cambridge, USA: MIT Press, 2004
[48] D. Angus and C. Woodward, "Multiple objective ant colony optimization," *Swarm Intelligence*, vol. 3, no. 1, pp. 69–85, 2009. doi: 10.1007/s11721-008-0022-4.
[49] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[50] P. Czyzzak and A. Jaszkiewicz, "Pareto simulated annealing - A metaheuristic technique for multiple-objective combinatorial optimization," *Journal of Multi-Criteria Decision Analysis*, vol. 7, no. 1, pp. 34-47, 1998
[51] P. Serafini, "Simulated annealing for multiple objective optimization problems," in *Proc. 10th International Conference on Multiple Criteria Decision Making: Expand and Enrich the Domains of Thinking and Application*, Berlin, Germany, vol. 1, pp. 283–292, 1994.
[52] M. P. Hansen, "Generating a diversity of good solutions to a practical combinatorial problem using vectorized simulated annealing," Institute of Mathematical Modelling, Technical University of Denmark, Tech. Rep., 1997.
[53] T. Ray, R. Gokarn, and O. Sha, "A global optimization model for ship design," *Computers in Industry*, vol. 26, no. 2, pp. 175–192, 1995. doi: 10.1016/0166-3615(95)00003-M.
[54] A. J. Ruiz-Torres, E. E. Enscore, and R. R. Barton, "Simulated annealing heuristics for the average flow-time and the number of tardy jobs bicriteria identical parallel machine problem," *Computers and Industrial Engineering*, vol. 33, no. 1–2, pp. 257–260, 1997. doi: 10.1016/S0360-8352(97)00087-9.
[55] J. F. Solís, H. J. Fraire, J. C. Soto-Monterrubio, et al., "Multi-objective simulated annealing algorithms for general problems," in *Handbook of Research on Military, Aeronautical, and Maritime Logistics and Operations*. Hershey, USA: IGI Global, 2016, pp. 280–292.
[56] E. Aarts, J. Korst, and W. Michiels, "Simulated annealing," in *Search Methodologies*, E. K. Burke and J. Korst, Ed. Boston, USA: Springer, 2014, pp. 265–285.
[57] Y. Liu, Y. Pan, M. Wang, et al., "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.
[58] Project Floodlight. (2017). *Floodlight* [Online]. Available: http://floodlight.openflowhub.org
[59] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proc. 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Monterey, USA, 2010, Article No. 19. doi: 10.1145/1868447.1868466.
[60] H. Ren, X. Li, J. Geng, et al., "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.
[61] J. Dugan. (2017). *IPerf—The Network Bandwidth Measurement Tool* [Online]. Available: https://iperf. fr
[62] R. Maniu and L. A. Dumitru, "Exploring the possibilities of a self-regulating SDN controller," *Scientific Bulletin "Mircea cel Batran" Naval Academy*, vol. 18, no. 1 , pp. 58–61, 2015.
[63] K. Kikuta, E. Oki, N. Yamanaka, et al., "Effective parallel algorithm for GPGPU-accelerated explicit routing optimization," in *IEEE Global Communications Conference (GLOBECOM)*, San Diego, USA, 2015, pp. 1–6.
[64] A, D. Stefano, G. Cammarata, G. Morana, et al., "A4SDN-adaptive alienated ant algorithm for software-defined networking," in *10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, Krakow, Poland, 2015.
[65] Y. L. Wang, K. J. Yuan, W. Fang, et al., "Research of a SDN traffic scheduling technology based on ant colony algorithm," in *2016 International Conference on Information Engineering and Communications Technology*. Lancaster, USA: DEStech Publications, Inc., 2016. doi: 10.12783/dtetr/iect2016/3754.
[66] R. Zhu, H. Wang, Y. Gao, et al., "Energy saving and load balancing for SDN based on multi-objective particle swarm optimization," in *15th International Conference on Algorithms and Architectures for Parallel Processing*, Zhangjiajie, China, 2015, pp. 176–189. doi: 10.1007/978-3-319-27137-8_14.
[67] S. Subbiah and V. Perumal, "Energy-aware network resource allocation in SDN," in *IEEE International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Chennai, India, 2016. doi: 10.1109/WiSPNET.2016.7566506.
[68] M. K. Awad, M. El-Shafei, T. Dimitriou, et al., "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, 2017. doi: 10.1002/nem.1972.
[69] O. Dobrijevic, M. Santl, and M. Matijasevic, "Ant colony optimization for QoE-centric flow routing in software-defined networks," in *11th IEEE International Conference on Network and Service Management (CNSM)*, Barcelona, Spain, 2015. doi: 10.1109/CNSM.2015.7367371.
[70] Y. Tang, W.Z. Wang, S. N. Dong, et al., "An Improved Ant Colony Algorithm for Routing in Software Defined Networking," *Journal of Computational and Theoretical Nanoscience*, vol. 13, no. 1, pp. 438–442, 2016. doi: 10.1166/jctn.2016.4824.
[71] R. Balaguer. (2017). *Flow embedding algorithms for software defined audio networks* [Online]. Available: http://ftp.tik.ee.ethz.ch/pub/.../MA-2014-14.pdf
[72] O. Dobrijevic, M. Santl, and M. Matijasevic, "Ant colony optimization for QoE-centric flow routing in software-defined networks," in *11th International Conference on Network and Service Management (CNSM)*, Barcelona, Spain, 2015, pp. 274–278.
[73] S. B. Kang and G. I. Kwon, "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.
[74] L. D. Chou, Y. T. Yang, Y. M. Hong, et al., "A genetic-based load balancing algorithm in openflow network," in *Advanced Technologies, Embedded and Multi-*

media for Human-Centric Computing, Y.-M. Huang, H.-C. Chao, D.-J. Deng, and J. H. Park Ed. Dordrecht, Netherlands: Springer, 2014, pp. 411–417.
[75] A. M. R. Ruelas and C. E. Rothenberg. (2015). *Implementation of neural switch using OpenFlow as load balancing method in data center* [Online]. Available: https://www.fee.unicamp.br/sites/default/files/departamentos/dca/eadca/eadca-viii/artigos/sessao8/ruelas_rothenberg.pdf
[76] S. Sathyanarayana and M. Moh, "Joint route-server load balancing in software defined networks using ant colony optimization," in *IEEE International Conference on High Performance Computing & Simulation (HPCS)*, Innsbruck, Austria, 2016. doi: 10.1109/HPCSim.2016.7568330.
[77] W. Lin and L. Zhang, "The load balancing research of SDN based on ant colony algorithm with job classification," in *2nd Workshop on Advanced Research and Technology in Industry Applications (WARTIA 2016)*, Dalian, China, 2016.
[78] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. 1st Workshop on Hot Topics in Software Defined Networks*, Helsinki, Finland, 2012, pp. 7–12. doi: 10.1145/2342441.2342444.
[79] S. Lange, S. Gebert, T. Zinner, et al., "Heuristic approaches to the controller placement problem in large scale SDN networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015. doi: 10.1109/TNSM.2015.2402432.
[80] J. M. Sanner, M. Ouzzif, Y. Hadjadj-Aoul, et al. (2016, Nov.). *Evolutionary algorithms for optimized SDN controllers & NVFs' placement in SDN networks* [Online]. Available: https://hal.inria.fr/hal-01421799
[81] A. Jalili, V. Ahmadi, M. Keshtgari, et al., "Controller placement in software-defined WAN using multi objective genetic algorithm," in *2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, Tehran, Iran, 2015. doi: 10.1109/KBEI.2015.7436121.
[82] V. Ahmadi, A. Jalili, S. M. Khorramizadeh, et al., "A hybrid NSGA-II for solving multiobjective controller placement in SDN," in *2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, Tehran, Iran, 2015. doi: 10.1109/KBEI.2015.7436122.
[83] C. Gao, H. Wang, F. Zhu, et al., "A particle swarm optimization algorithm for controller placement problem in software defined network," in *15th International Conference on Algorithms and Architectures for Parallel Processing*, Zhangjiajie, China, 2015, pp. 44–54. doi: https://doi.org/10.1007/978-3-319-27137-8_4.
[84] S. Liu, H. Wang, S. Yi, et al., "NCPSO: a solution of the controller placement problem in software defined networks," in *15th International Conference on Algorithms and Architectures for Parallel Processing*, Zhangjiajie, China, 2015, pp. 213–225. doi: https://doi.org/10.1007/978-3-319-27137-8_17.
[85] J. Ashraf and S. Latif, "Handling intrusion and DDoS attacks in software defined networks using machine learning techniques," in *National Software Engineering Conference (NSEC)*, Rawalpindi, Pakistan, 2014. doi: 10.1109/NSEC.2014.6998241.
[86] X. Li, D. Yuan, H. Hu, et al., "DDoS detection in SDN switches using support vector machine classifier," in *2015 Joint International Mechanical, Electronic and Information Technology Conference (JIMET-15)*, Chongqing, China, 2015.
[87] F. Li, "Application of particle swarm BP neural network in DDoS attack detection," *Microcomputer & Its Applications*, no. 3, 2014. doi: 10.3969/j.issn.1674-7720.2014.03.016.
[88] H. H. Chen and S. K. Huang, "LDDoS attack detection by using ant colony optimization algorithms," *Journal of Information Science & Engineering*, vol. 32, no. 4, pp. 995–1020, 2016.
[89] J. Liu, H. Zhang, and Z. Guo, "A defense mechanism of random routing mutation in SDN," *IEICE Transactions on Information and Systems*, vol. 100, no. 5, pp. 1046–1054, 2017. doi: 10.1587/transinf.2016EDP7377.
[90] A. A. Ojugo, A.O. Eboka, O.E. Okonta, et al, "Genetic algorithm rule-based intrusion detection system (GAIDS)," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 8, pp. 1182–1194, Aug. 2012
[91] J. L. Zhao, J. F. Zhao, and J. J. Li, "Intrusion detection based on clustering genetic algorithm," in *International Conference on Machine Learning and Cybernetics*, Guangzhou, China, pp. 3911– 3914, 2005. doi: 10.1109/ICMLC.2005.1527621.
[92] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized deep packet inspection functions in SDN," in *IEEE Military Communications Conference (MILCOM 2013)*, San Diego, USA, 2013. doi: 10.1109/MILCOM.2013.172.
[93] P. Murukan, D. Jamaluddine, S. Kolhapure, et al. (2016, Feb.). *A cost-based placement algorithm for multiple virtual security appliances in cloud using SDN: MO-UFLP (multi-ordered uncapacitated facility location problem)* [Online]. Available: https://arxiv.org/abs/1602.08155
[94] K. Li, J. Bao, Z. Lu, et al., "A PSO-based virtual SDN customization for multi-tenant cloud services," in *Proc. 11th International Conference on Ubiquitous Information Management and Communication*, Beppu, Japan, Article No. 91, 2017. doi: 10.1145/3022227.3022317.
[95] X. Yao, H. Wang, C. Gao, et al., "Maximizing network utilization for SDN based on particle swarm optimization," in *28th International Conference on Tools with Artificial Intelligence (ICTAI)*, San Jose, USA, 2016. doi: 10.1109/ICTAI.2016.0142.
[96] C. Gao, H. Wang, L. Zhai, et al., "Optimizing routing rules space through traffic engineering based on ant colony algorithm in software defined network," in *28th International Conference on Tools with Artificial Intelligence (ICTAI)*, San Jose, USA, 2016. doi: 10.1109/ICTAI.2016.0026.
[97] S. Yi, H. Wang, X. Yao, et al., "Maximizing network utilization for SDN based on WiseAnt colony optimization," in *IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Sydney, Australia, 2016. doi: 10.1109/HPCC-SmartCity-DSS.2016.0137.
[98] Y. Guo, Z. Wang, X. Yin, et al., "Incremental deployment for traffic engineering in hybrid SDN network," *34th International Performance Computing and Communications Conference (IPCCC)*, Nanjing, China, 2015. doi: 10.1109/PCCC.2015.7410320.

Manuscript received: 2017-06-08

Biographies

LIAO Lingxia (liaolx@ece.ubc.ca) is currently a Ph.D. candidate of Department of Electrical and Computer Engineering of the University of British Columbia (UBC), Canada. She had a bachelor degree from Tsinghua University, China and a master degree from UBC. She was a science researcher in Computer Science Department of UBC, and the R&D director and the general manager of high performance computing of Inspur Group, China. She had over ten years' experience working in computer and network industry. Her current research interests are E-hearthcare, cloud computing, software defined networking, network function virtualization, network monitoring and optimization, and next generation network. She has contributed multiple technical papers and book chapters in these areas.

Victor C.M. Leung (vleung@ece.ubc.ca) is a professor of Electrical and Computer Engineering and holder of the TELUS Mobility Research Chair at the University of British Columbia (UBC). He has contributed some 1000 technical papers, 37 book chapters and 12 book titles in the areas of wireless networks and mobile systems. He was a Distinguished Lecturer of the IEEE Communications Society. He is serving/has served on the editorial boards of the *IEEE Journal on Selected Areas in Communications*, *IEEE Access*, IEEE Transactions on Wireless Communications, IEEE Transactions on Computers, IEEE Transactions on Vehicular Technology, *IEEE Wireless Communications Letters* and several other journals, and has contributed to the organizing and technical program committees of numerous conferences. Dr. Leung was a winner of the 2011 UBC Killam Research Prize, the IEEE Vancouver Section Centennial Award, the 2017 Canadian Award for Telecommunications Research. He co-authored a paper that won the 2017 IEEE Communications Society Fred W. Ellersick Prize.

LAI Chin-Feng (cinfon@ieee.org) is an associate professor at Department of Engineering Science, National Cheng Kung University, China and Department of Computer Science and Information Engineering, National Chung Cheng University, China since 2016. He received the Ph.D. degree in Department of Engineering Science from National Cheng Kung University, China in 2008. He received Best Paper Awards from IEEE 17th CCSE, 2014 International Conference on Cloud Computing, IEEE 10th EUC, and IEEE 12th CIT. He has more than 100 paper publications and 4 papers selected to TOP 1% most cited articles by Essential Science Indicators (ESI). He serves as an associate editor-in-chief, editor, or associate editor for many journals and is TPC Co-Chair for many conferences during 2012–2017. His research focuses on Internet of Things, body sensor networks, E-healthcare, mobile cloud computing, cloud-assisted multimedia network, embedded systems, etc. He has been an IEEE senior member since 2014.