

A Particle Swarm Optimization Algorithm for Controller Placement Problem in Software Defined Network

Chuangen Gao, Hua Wang^(✉), Fangjin Zhu,
Linbo Zhai, and Shanwen Yi

School of Computer Science and Technology,
Shandong University, Jinan 250101, China
gaochuangen@mail.sdu.edu.cn, wanghua@sdu.edu.cn

Abstract. Software defined network (SDN) decouples the control plane from packet processing device and introduces the controller placement problem. The previous methods only focus on propagation latency between controllers and switches but ignore either the latency from controllers to controllers or the capacities of controllers, both of which are critical factors in real networks. In this paper, we define a global latency controller placement problem with capacitated controllers, taking into consideration both the latency between controllers and the capacities of controllers. And this paper proposes a particle swarm optimization algorithm to solve the problem for the first time. Simulation results show that the algorithm has better performance in propagation latency, computation time, and convergence.

Keywords: Software defined network · Controller placement · Propagation latency · Particle swarm optimization

1 Introduction

Unlike traditional networks, where both control and forwarding planes are highly integrated on the same boxes, the Software Defined Network (SDN) architecture decouples control and forwarding planes. Such separation is realized by moving the network intelligence onto one or more external servers, called controllers, which make up a network-wide logically centralized control plane that oversees a set of dumb, and simply forwarding elements [1].

A particularly important task in SDN architectures is controller placement, i.e., the positioning of a limited number of controllers within a network to meet various requirements [2]. It is called the controller placement problem which is known to be NP-hard. These requirements range from latency constraints to failure tolerance and load balancing. We narrow our focus to latency constraints, because it places fundamental limits on availability and convergence time. Our goal is to find optimal minimum-latency placements, when the number of controllers is given. It has practical implications for software design, affecting whether controllers can respond to events in real-time, or whether they must push forwarding actions to forwarding elements in advance [3].

Previously, the solution only focused on propagation latency between controllers and switches but ignored the latency between controllers, which is a critical factor in real networks. If there is more than a single controller in a network, synchronization is necessary to maintain a consistent global state. Depending on the frequency of the inter-controller synchronization, the latency between the individual controllers plays an important role and thus should be considered during the controller placement [4]. Besides the propagation latency between controllers, the load of controllers is a critical factor in real networks too. Because of the constraints of processor, memory, access bandwidth and other resources, a commodity server only has the capacity to manage a limited number of routers [6]. Whenever the load of a controller reaches a threshold, the message processing latency on the controller will increase substantially. In summary, we make the following contributions:

- To our best knowledge, this is the first work that proposes the Particle Swarm Optimization algorithm to solve the controller placement problem which is known to be NP-hard in SDN. By simulations, we show that the algorithm achieves better solutions than close-to-optimal ones obtained by the Integer Linear Program (ILP).
- We define the Global Latency Control Placement Problem with Capacitated Controllers (CGLCPP) which takes into consideration both the latency from controllers to controllers and the capabilities of controllers for the first time.

The rest of this paper is organized as follows. The related work is reported in Sect. 2. The Mathematical Model is depicted in Sect. 3. Section 4 briefs the particle swarm optimization algorithm to solve the CGLCPP. The results of simulations and discussion of the performance are reported in Sect. 5. Finally, we conclude our work in Sect. 6.

2 Related Work

The controller placement problem in SDN was first proposed in [3]. The authors motivate the controller placement problem and present initial analysis of a fundamental design. This paper examines the impacts of placements on average and worst-case propagation latencies on real topologies. But its goal is not to find optimal minimum-latency placements and don't take into consideration the latency between controllers which is necessary in SDN with multiple controllers. This paper doesn't propose a useful algorithm to find the solution, and each optimal placement shown in this paper comes from directly measuring the metrics on all possible combinations of controllers. This method ensures accurate results, but at the cost of weeks of CPU time; the complexity is exponential for k , since brute force must enumerate every combination of controllers. The paper [4] discusses several aspects of the controller placement problem from the view of resilience and failure tolerance. The delay between controllers is only a small part of its multi-objective optimization considering and has not been studied in depth.

The paper [5] addresses the problem of placing controllers in SDN, so as to maximize the reliability of control networks. It presents a metric to characterize the reliability of SDN control networks, and further quantify the impact of controller

number on the reliability of control networks. It proposes two heuristic algorithms to solve the problem, which are the greedy algorithm and Simulated Annealing algorithm respectively. In this letter [6], it focuses on the load of controllers and defined a capacitated controller placement problem, taking into account the capabilities of controllers. But it hasn't dealt with the latency between controllers.

The papers [4, 7] address the controller placement problem according to multiple objective functions. They argue a controller placement should also fulfill certain resilience constraints especially for the control plane. The model simultaneously determines the optimal number, location, and type of controller as well as the inter-connections between all the network elements. The goal of the model is to minimize the cost of the network while considering different constraints.

3 Mathematical Model of the Global Latency Control Placement Problem

In SDNs, switches communicate with their controller via standard TLS or TCP connections. When multiple controllers are deployed, the latency between the individual controllers plays an important role, because communications between these controllers are also required to achieve global consistency of network state [8]. For example, the Google's B4 network provides connectivity among a modest number of data centers, e.g., for asynchronous data copies, index pushes for interactive serving systems, and end user data replication [9]. Thousands of internal application traffic runs across this network. And user data is the most latency sensitive, and is of the highest priority.

Besides the propagation latency between controllers, the load of controllers is a critical factor in real networks too. Because of the constraints of processor, memory, access bandwidth and other resources, a commodity server only has the capacity to manage a limited number of routers [6]. Whenever the load of a controller reaches a threshold, the message processing latency on the controller will increase substantially. Heavy-load controllers always have higher failure probability, because they have little resources to handle various errors and are more likely to be attacked.

The latency between the individual controllers plays an important role, and the load of controllers is a critical factor in real networks too. Thus both them should be considered during the controller placement. We define the Global Latency Control Placement Problem with Capacitated Controllers (CGLCPP), which consists of both the latency from controllers to switches and the latency from controllers to controllers, and takes into consideration the capabilities of controllers.

For a network graph (V, E) , where V is the set of nodes, E the set of links. Let n be the number of nodes. Let k denotes the number of controllers to be placed in the network, and n_c denotes the number of controller-to-controller paths. The edge weights represent propagation latencies, where $d(v, c)$ is the shortest path from node v to controller c , and $d(c_i, c_j)$ is the shortest path from controller c_i to controller c_j . Let $F(c)$ denote the forwarding switches controlled by the controller c . Each controller c has a capacity $L(c)$. The load of control attributed to switch v is denoted by $l(v)$.

In order to be applicable and universal, we compute the average delay. And the global average propagation latency for a placement of controllers S' is:

$$G(S') = \frac{1}{n} \sum_{v \in V} \min_{c \in S'} d(v, c) + \frac{1}{n_c} \sum_{c_i, c_j \in S'} d(c_i, c_j) \quad (1)$$

Definition:

Global Latency Control Placement Problem with Capacitated Controllers

$$\text{Min } G(S') \quad (2)$$

Subject to:

$$\sum_{v \in F(c)} l(v) \leq L(c) \quad \forall c \in S' \quad (3)$$

Given the number k of controllers to be deployed in SDN, our goal is to find the placement S' from the set of all possible controller placements S , such that $G(S')$ is minimum and $|S'| = k$.

4 Particle Swarm Optimization for GLCPP Algorithm

With the given input k , the number of controllers to place, the global latency control placement problem is an application example of the famous minimum k -median problem [10], which is proved to be NP-hard. To solve this problem, we propose particle swarm optimization algorithm that automate the controller placement decision for the first time, which has good performance in NP-hard problem optimization. Section 4.1 describes the PSO algorithm, and Sect. 4.2 explains PSO-CGLCPP algorithm.

4.1 PSO Algorithm

Particle swarm optimization algorithm (PSO) was proposed in 1995 by Eberhart and Kennedy [11]. It is a population-based stochastic optimization algorithm that originates from nature. PSO searches the optimum within a population called a swarm and benefits from two types of learning: cognitive learning based on an individual's history and social learning based on a swarm's history accumulated by sharing information among all particles in the swarm [12]. Successful applications of PSO have demonstrated that it is a promising and efficient optimization method.

The mathematical analysis of PSO is described as the following. There are n particles which represent potential solutions of the problem. The particle is defined as a d -dimensional vector. The current position of the particle in search space is $X_i = [x_{i1}, x_{i1}, \dots, x_{id}]$, $i = 1, 2, \dots, n$, and its current velocity vector is $V_i = [v_{i1}, v_{i2}, \dots, v_{id}]$. We use P_i to stand for individual best position of the particle, $P_i = [p_{i1}, p_{i2}, \dots, p_{id}]$.

And P_g is regarded as global best position vector which particle swarm have found, $P_g = [p_{g1}, p_{g2}, \dots, p_{gd}]$. So, the position and velocity vector of the particles should adjust according to the following equations:

$$V_{id}^{t+1} = \omega V_{id}^t + c_1 r_1 (P_{id} - X_{id}^t) + c_2 r_2 (P_{gd} - X_{id}^t) \quad (4)$$

$$X_{id}^{t+1} = X_{id}^t + V_{id}^{t+1} \quad (5)$$

where ω expresses the inertia weight, r_1 and r_2 are elements from random sequences in the range of (0, 1), which are mutually independent. The parameter c_1 controls the influence degree of a cognitive part of an individual, and c_2 determines the effect of a social part of the swarm.

The inertia weight ω lets the algorithm improve its performance in a series of applications. Paper [13] found that large inertia weight can help the global search, small inertia weight can improve local search ability. Therefore, adaptive adjustment of inertia weight is proposed. The inertia weight is not fixed value, but a function of linear reduction over time. The inertia weight function is shown as the following:

$$\omega = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{t_{\max}} \times t \quad (6)$$

ω_{\max} is set as the initial weight, ω_{\min} is set as the final weight. Variable t_{\max} represents the maximum number of iteration, t is the number of current iteration. Usually ω_{\max} is set as 0.9 and ω_{\min} is set as 0.1.

The individual best position vector of each particle is computed using the following expression:

$$P_i(t+1) = \begin{cases} P_i(t) & \text{if } f(X_i(t+1)) \geq f(P_i(t)) \\ X_i(t+1) & \text{if } f(X_i(t+1)) \leq f(P_i(t)) \end{cases} \quad (7)$$

where f represents the fitness function.

Then, the global best position vector is found by

$$P_g(t+1) = \arg \min_{P_i(t+1)} f(P_i(t+1)). \quad (8)$$

4.2 PSO-CGLCPP Algorithm

In PSO, the position vector represents a solution to the optimized problem. For the global latency control placement problem, each particle represents one kind of placement for controllers to be deployed. With the given input k , the number of controllers to place, the particle is defined as a k -dimensional vector, and each dimension represents one of the controllers. So the position permutation of a particle i is defined as $X_i = [x_{i1}, x_{i1}, \dots, x_{ik}]$. For we deploy each controller on one of the existing vertexes of the network, each dimension of position is a integer between 1 and n , i.e., $x_{ik} \in [1, n]$, where n is equal to the number of total vertexes in the network. Velocity works on the position sequence and

it is rather crucial. A good velocity gives the particle a guidance and determines whether the particle can reach its destination and by how fast it could [14].

When the position of the particle is established, namely the controllers are in place, we compute the adaptive value of particle according to formula (1). Because the node is assigned to its nearest controller, the delay from node to corresponding controller is generally measured in that link length value. Then we compute load of controller according to formula (3). If it exceeds the load limitation of controller, assign the switch to the next nearest switch controller. And the delay between controllers and controllers is defined as the shortest link between them in the same way.

In this section, we propose Particle Swarm Optimization for CGLCPP Algorithm (PSO-CGLCPP). The basic idea is, by iteration, multiple particles search the optimal solution in parallel, and minimum total delay controller placement is located. And the overall procedure is about: initialize n particles randomly, with each particle representing a kind of placement of k controllers. Then set the particle with the highest fitness to be the current best solution whose latency value is smallest. According to the PSO algorithm, use the PSO velocity formulas (4) and (5) to merge the controller placement and determine the new particle position until this swarm obtains its longest lifetime or it converges. If PSO-CGLCPP converges, then the best solution can be obtained. The PSO-CGLCPP algorithm for global latency controller placement problem is described as follows.

Algorithm 1. Overall procedure of proposed *PSO-CGLCPP*

- Input:**
- 1) Network topology graph (V, E) ;
 - 2) Objective function according to formula (1);
 - 3) The number of controllers k ;
 - 4) The maximal generation T ;

Output: The minimum global latency controller placement S^*

Initialization :

1. Randomly generate each particle's velocity and position;
2. Evaluate the fitness value of each particle;
3. Fill the P_i of each particle with its current position;
4. Fill the P_g with the optimal P_i ;

Optimization :

1. **repeat**
 2. **for** each particle
 3. Update particle's velocity according to (4);
 4. Update particle's position according to (5);
 5. Evaluate the fitness value of the particle;
 6. Update the P_i according to (7);
 7. Update the P_g according to (8);
 8. **end for**
 9. **until** stopping criterion is satisfied
-

Algorithm 2. Procedure of evaluating the fitness of the particle

1. **for** each particle
 2. **for** each node
 3. Find the shortest path to the each controller;
 4. Evaluate the longitude of the path;
 5. Assign the node to the controller nearest to it;
 6. Compute load of controller according to (3);
 7. **while** exceed load limitation of the controller **do**
 8. Assign the switch to the next nearest controller;
 9. **end for**
 10. **for** each controller
 11. Find the shortest path to other controllers;
 12. Evaluate the length of the path as controller-to-controller latency;
 13. **end for**
 14. Evaluate the global latency according to (1);
 15. **end for**
-

5 Simulation and Results Analysis

In this section, we generate network topologies randomly according to [15], which is almost closed to the real network. Controller's capability is a complex concept, including bandwidth, memory and computing resources and so on. In order to simplify the problem, we use a digital to denote it. The load of controller attributed to switch is processed in the same way. Our PSO-CGLCPP is written with C++ in VS2010, and runs on the machine equipped with Intel Core i5 4-Core processors and 4 GB RAM. We obtain the average solution by running the algorithm 100 times on every testing topology.

To evaluate the performance of our algorithm, we run the other two algorithms, along with the random placement algorithm (Random) for comparison purpose. The two algorithms are greedy algorithm (GL) proposed in [5] and integer linear programming algorithm (ILP) in [6] respectively. CPLEX is used to solve integer programming. For a NP-hard problem, it will be hard to find the solution based on the enumeration algorithm which ensures accurate results, but at the cost of weeks of CPU time. Linear programming algorithms are always used to solve the problems, and the solution is regarded as the optimal solution for a reference. So we mainly regard the ILP algorithm as a reference and compare with it. Since the random algorithm returns different results based on the locations selected, we execute random placements over 100 times, and select the placement that yields the best performance.

We evaluate the algorithms in two aspects, the first is the optimal latency solution, and the second aspect is computing time for algorithm to find the optimal solution.

And we further characterize both latency and computing time performance against the size of controllers. The results of the simulation are depicted in the following figures. PSO is abbreviated words of PSO-CGLCPP.

Figure 1 evaluates the optimal latency of four algorithms with increasing network nodes, given the same number of controllers. It shows that PSO and ILP get a better average delay. PSO’s performance is slightly better than the ILP algorithms. Greedy algorithm and Random algorithm perform relatively poor, and the worst is random algorithm. Furthermore the Random and GL increase quickly along with increasing nodes while PSO and ILP provide an optimal solution more reliably. The greedy algorithm picks the next vertex that best minimizes latency, which always produces the best location currently, but local optimum does not represent a global optimum. Because of diversity of particles, PSO can search the entire space and jump out of local optimum through exchanging information with each other. Finally the particle will flight to a better global optimum after several iterations.

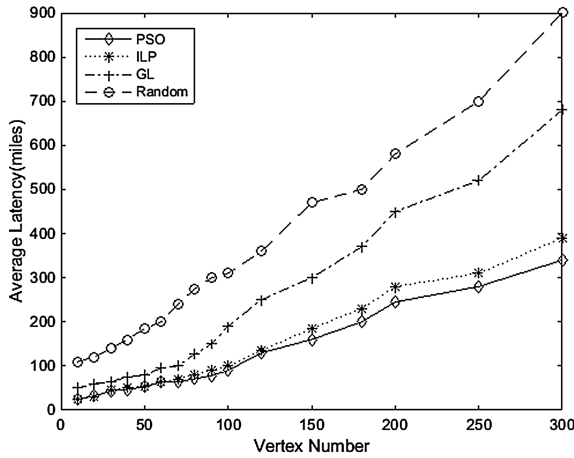


Fig. 1. Optimal latency solution of the algorithms

Figure 2 analyzes number of the controllers’ impact on average latency under the same network topology. In the experiments, we gradually increase the number of controllers for each placement strategy. With the increase of controllers placed in, the average latency decreases. The average delay gained by PSO algorithm is always relatively stable, less volatile, compared with the greedy algorithm. This is because the particle swarm optimization through mutual learning and iteration between the particles can always find a better controllers’ position to achieve better results. Furthermore, the diminishing returns that level off around 4–8 controllers, suggested by the literature [3], have been verified.

As can be seen from the Fig. 3, greedy algorithm consumes the shortest time, but the solution is far away from the optimal one, which is unacceptable in most environments. Processing time of both PSO and ILP are much more than GL algorithm,

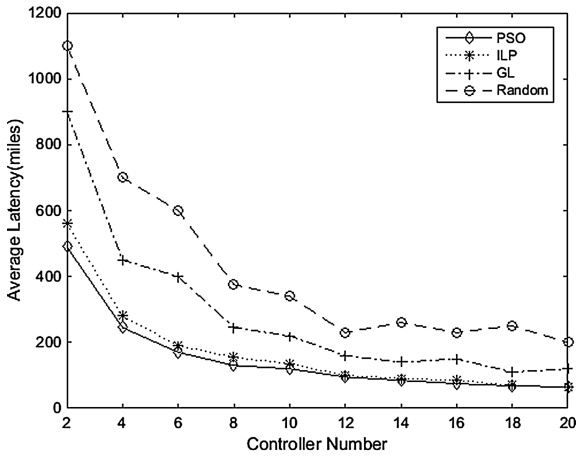


Fig. 2. Controller number’s impact on average latency

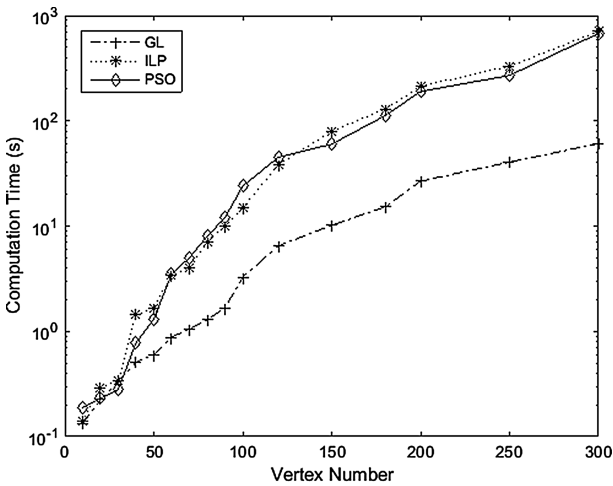


Fig. 3. Computing time of the algorithms

but produce a better solution. The time taken by the PSO and ILP algorithm are in almost close level. When network size is larger, PSO algorithm converges to the optimal solution slightly faster than ILP. PSO convergence must take advantages of the swarm intelligence, and achieve a more satisfactory solution within an acceptable timeframe, which can match with commercially available strategies. Although PSO takes much more time in the procedure of exploring the optimal placement than GL, the time is acceptable in lots real application scenarios in which the task has no real-time demand. Figure 4 evaluates calculation time with increasing controller nodes to be deployed, when the network size is the same. As can be seen, when the number of controllers deployed becomes bigger, the convergence of time becomes larger and

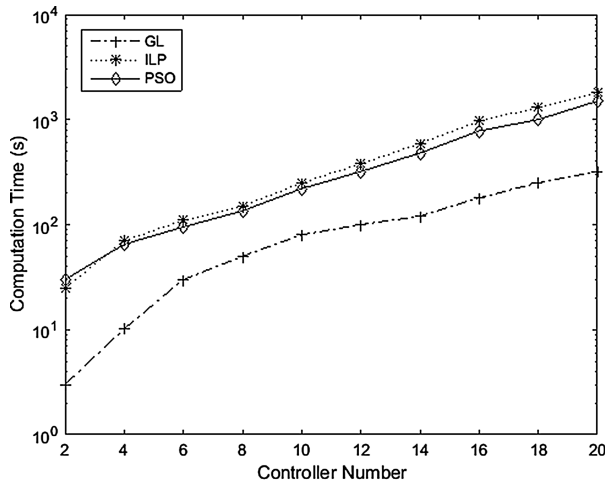


Fig. 4. Controller number's impact on computation time

larger. And the growth rate of computation time becomes bigger, because faster growth of number of links between controllers.

6 Conclusion

In this paper we investigate the controller placement problem in software defined network, and define the Global Latency Control Placement Problem with Capacitated Controllers (CGLCPP), taking into consideration both the latency between controllers and the capabilities of controllers. We then propose a PSO-CGLCPP algorithm based on particle swarm optimization to solve the problem for the first time. Our method could find an optimized solution for the given controllers while conforming to the constraints of controllers. Experimental results show that the algorithm performs rapidly and effectively.

Acknowledgments. The study is supported by the Natural Science Foundation of Shandong Province (Grant No. ZR2015FM008; ZR2013FM029), the Science and Technology Development Program of Jinan (Grant No. 201303010), the National Natural Science Foundation of China (NSFC No. 60773101), and the Fundamental Research Funds of Shandong University (Grant No. 2014JC037).

References

1. McKeown, N., et al.: Open flow: enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* **38**(2), 69–74 (2008)
2. Lange, S., et al.: Heuristic approaches to the controller placement problem in large scale SDN networks. *IEEE Trans. Netw. Serv. Manage.* **12**(1), 4–17 (2015)

3. Heller, B., Sherwood, R., McKeown, N.: The controller placement problem. In: Proceedings of ACM SIGGCOM HotSDN, pp. 7–12 (2012)
4. Hock, D., et al.: POCO-framework for Pareto-optimal resilient controller placement in SDN-based core network. In: IEEE NOMS (2014)
5. Hu, Y., Wang, W., et al.: Reliability-aware controller placement for software-defined networks. In: IEEE International Symposium on Integrated Network Management (2013)
6. Yao, G., Bi, J.: On the capacitated controller placement problem in software defined networks. *IEEE Commun. Lett.* **18**(8), 1339–1342 (2014)
7. Sallahi, A., St-Hilaire, M.: Optimal model for the controller placement problem in software defined networks. *IEEE Commun. Lett.* **19**(1), 30–33 (2015)
8. Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., Shenker, S.: Onix: a distributed control platform for large-scale production networks. In: Proceedings of OSDI (2010)
9. Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., et al.: B4: experience with a globally-deployed software defined wan. *ACM SIGCOMM Comput. Commun. Rev.* **43**, 3–14 (2013)
10. Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., Pandit, V.: Local search heuristics for k-median and facility location problems. *SIAM J. Comput.* **33**(3), 544–562 (2004)
11. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium on Micromachine and Human Science, pp. 39–43 (1995)
12. Pehlivanoglu, Y.V.: A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks. *IEEE Trans. Evol. Comput.* **17**(3), 436–452 (2013)
13. Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. In: IEEE International Conference of Evolutionary Computation, Piscataway, vol. 8, no. 3, pp. 240–255 (1998)
14. Gong, M., Cai, Q., Chen, X., Ma, L.: Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition. *IEEE Trans. Evol. Comput.* **18**(1), 82–97 (2014)
15. Naldi, M.: Connectivity of Waxman topology models. *Comput. Commun.* **29**, 24–31 (2005)