ChinaRxiv

AI translation · View original & related papers at
chinarxiv.org/items/chinaxiv-201805.00359

Virtual Network Function Allocation and Scheduling Algorithms in Security Service Chains: A Postprint

Authors: Huang Rui, Zhang Hongqi

Date: 2018-05-18T00:00:00+00:00

Abstract

Virtual network functions (VNFs) in security service chains decouple traditional network security functions from hardware devices, enabling more dynamic and scalable deployment of service functions. However, the problem of rationally allocating VNFs to nodes and efficiently scheduling VNFs on nodes remains an urgent challenge. To this end, we propose an optimization algorithm-based solution in the software-defined networking (SDN) and network function virtualization (NFV) environment. First, we formulate the resource allocation and scheduling problem and formally define the optimization objective; second, we propose a greedy algorithm-based resource allocation scheme and a hybrid bee colony algorithm-based resource scheduling scheme to coordinately address the VNF resource allocation and scheduling problem in a unified manner. Finally, simulation experiments are designed to verify the time complexity of the proposed algorithms and their improvements in terms of total resource cost and total service revenue; meanwhile, by comparing the hybrid bee colony algorithm with the traditional bee colony algorithm, the results demonstrate that the former exhibits faster convergence speed.

Full Text

Preamble

URL: http://www.arocmag.com/article/02-2019-03-046.html
Journal: ChinaXiv Cooperative Journal - Application Research of Computers
Title: Research on Algorithm of VNF Allocation and Scheduling Problems in Security Service Chain

Authors: Huang Rui, Zhang Hongqi (PLA Information Engineering University, Zhengzhou 450001, China)

Abstract: The virtual network function (VNF) in the security service chain decouples traditional network security functions from hardware devices, making service function deployment more dynamic and scalable. However, the rational allocation of VNFs to nodes and the efficient scheduling of VNFs on nodes remain urgent problems to be solved. To address this, we propose an optimization algorithm-based solution in the software-defined networking (SDN) and network function virtualization (NFV) environment. First, we illustrate the resource allocation and scheduling problem and formally define the optimization objectives. Second, we propose a resource allocation scheme based on a greedy algorithm and a resource scheduling scheme based on a hybrid bee colony algorithm to coordinately solve the VNF resource allocation and scheduling problem. Finally, we design simulation experiments to verify the time complexity of the proposed algorithms and their improvements in total resource cost and total service revenue. Additionally, comparing the hybrid bee colony algorithm with the traditional bee colony algorithm shows that the former achieves faster convergence.

Keywords: security service chain; virtual network function; software defined network; network function virtualization; greedy algorithm; hybrid bee colony algorithm

Classification: TP393
DOI: 10.3969/j.issn.1001-3695.2017.09.0949

---

0 Introduction

Current network security services rely on network operators to provide dedicated hardware devices, which exacerbates the tight coupling between network security functions and hardware equipment. This results in a static and rigid network security service model that cannot meet the diverse demands of future network services [1,2]. Specifically, on one hand, the hardware implementation of existing security functions (firewall, IDS (intrusion detection system), IPS (intrusion protection system)) leads to closed, 固化, and inflexible security capabilities. On the other hand, the static deployment of security functions cannot adapt to dynamic changes in business requirements, causing waste of network resources. Consequently, dynamic composition mechanisms for security service chains under software-defined networking (SDN) [3,4] have emerged [5,6]. As a crucial component in the construction and deployment of security service chains, research on allocation and scheduling algorithms for virtual network functions (VNFs) [7] in service chains is imperative.

The rapid development of SDN has catalyzed the flourishing of network function virtualization (NFV) technology [8,9]. SDN decouples control functions from traditional distributed network devices and makes them directly programmable, achieving centralized network management and control. NFV transforms network functions deployed on proprietary hardware into VNFs running on general-purpose servers through virtualization, providing conditions for upper-layer net-

work function innovation. The combination of these two technologies offers strong support for exploring new network security service models, making rational allocation and optimized scheduling of VNFs in security service chains possible.

Regarding VNF resource allocation and scheduling problems, existing research is limited and incomplete. Riera et al. [10,11] first formally defined the VNF resource scheduling problem, taking minimizing completion time as the optimization objective for a feasible solution, but did not provide specific solution methods. Mijumbi et al. [12] addressed the resource allocation/mapping problem in VNFs based on tabu search algorithms, yet did not propose a reasonable solution for resource scheduling. Drawing inspiration from these research ideas while addressing their limitations, this paper innovatively proposes a resource allocation scheme based on greedy algorithms and a resource scheduling scheme based on hybrid bee colony algorithms. We abstract the resource scheduling process as a flexible Job-Shop scheduling problem (FJSP) and employ a hybrid bee colony algorithm (HBC) to solve it, ensuring both the efficiency of the resource scheduling process and the coordination between resource allocation and scheduling.

---

1 Problem Description

The VNF physical resource configuration problem in security service chains can be divided into two categories: (1) Virtual Network Embedding (VNE). VNE refers to embedding virtual devices into physical devices, primarily involving the selection of physical nodes under different network states, calculation of available physical resources, and mapping between the two types of nodes [13,14]. Related research is already mature and thus beyond the scope of this paper. (2) Virtual Network Function Allocation and Scheduling (VNF-AS). VNF-AS refers to the rational allocation and scheduling of multiple VNFs on virtual nodes according to different security service requirements, optimizing execution order, reducing resource consumption, and striving to maximize energy efficiency. Related research is scarce and incomplete, making it the focus of this paper.

1.1 Resource Allocation and Scheduling Example

As shown in FIGURE:1, the VNF embedding and scheduling problem involves many possibilities for resource sharing. One scenario is where each VNF is embedded in a specific virtual device whose resources are exclusively used. For example, a security service chain consists of 5 different types of VNFs, meaning that for security service requests from different users, 5 specific virtual devices are required to provide services. This not only causes significant physical resource consumption but also makes it difficult to meet complex security service requests from large-scale network users. This paper adopts container technology to implement network function nodes. As shown in FIGURE:2, a high-volume

server (HVS) serves as the physical node providing security services, equipped with a container cluster that is uniformly orchestrated and managed by an orchestrator. Multiple VNFs can be activated in each container, enabling security service functions to share resources.

FIGURE:3–5 illustrate the resource allocation and scheduling scenarios for two different security service chains. FIGURE:3 shows a possible resource allocation when security service request S_1 arrives at time T_1. Node n_1 simultaneously provides resource support for virtual network functions f_8 and f_2. Similarly, assuming security service request S_2 arrives after some time, its resource allocation is shown in FIGURE:4. At this point, both security service chains are processed by virtual nodes simultaneously. From the figures, we can observe that both processing sequences of S_1 and S_2 contain VNFs f_6 and f_8. For S_1, they are processed by nodes n_2 and n_7 respectively without waiting, achieving fast and efficient processing. However, f_8 in S_2 accumulates on node n_1, queuing and waiting, causing unnecessary processing delays. This involves the VNF resource optimization allocation problem, for which we design a greedy algorithm-based solution strategy in subsequent sections.

FIGURE:5 presents the resource scheduling timeline for security service chains S_1 and S_2. Each function must be processed in sequence, and each node can only process one specific function at any given time. We define symbol $P_{f,ij}$ as the processing delay of function f_i on virtual node n_j, where $1 \leq i \leq m$ and $1 \leq j \leq n$, with m and n representing the number of VNFs and virtual nodes respectively. This paper adopts assumed processing times to represent the scheduling status of f_i on each node. At time T_1, security service request S_1 arrives and is immediately assigned to node n_1 for processing. Each subsequent function must wait until its predecessor is completed. At time T_4, f_2 finishes processing. Therefore, f_3 can only start working after waiting for the time interval $[T_4, T_6]$. This waiting interval not only increases the processing time of S_1 but may also reduce resource utilization due to long idle periods of other functions in S_2, which is a direct consequence of the previous resource allocation step. Assuming we map f_3 in S_2 to node n_4, we can effectively avoid additional time overhead in the resource scheduling process. Therefore, only by coordinating resource allocation and scheduling processes can we guarantee security services.

There are three possible cases for VNF resource scheduling: (a) Immediately process the next function after completing one (as shown in FIGURE:5 at time T_2, f_5 on n_2 is processed right after f_1); (b) A function must wait until the previous function is processed (as shown in FIGURE:5, S_2 arrives at T_6 but must wait until T_9 when n_1 becomes available); (c) A function must continuously wait until all functions in a chain are processed (as shown for node n_3 in FIGURE:5, which must wait for the time interval of S_1 to complete processing). In subsequent sections, we abstract the resource scheduling problem as FJSP and solve it based on the hybrid bee colony algorithm.

2 Greedy Algorithm-Based Resource Allocation Scheme

2.1 Algorithm Motivation Analysis

In resource allocation and scheduling problems, there are many different optimization objectives. This paper selects processing time, resource cost, and service revenue as representative indicators for analysis.

1) Processing Time: Processing time refers to the time interval from when the last function of a security service request completes processing to when the service arrives (Equation (1), where t_t represents completion time and t_e represents arrival time). Processing time is a measure of two parameters: (a) Resource utilization (excessive processing time indicates that services occupy the network for long periods, causing high network load and resource consumption, i.e., low resource utilization); (b) Quality of service (shorter processing time means faster average processing speed and better service quality).

2) Resource Cost: Resource cost refers to the spatial storage and temporal processing overhead of physical network resources for a specific security service request. In Equation (2), δ_i represents the spatial storage resource consumed by all functions f_i in service chain S, while $(t_t - t_e)$ represents temporal processing overhead, which includes not only the actual processing time of nodes but also waiting delays. Constants α and β represent the weights of these two factors respectively.

3) Service Revenue: Service revenue refers to the comprehensive benefit that physical network resources provide for a specific security service request. In Equation (3), the Boolean variable γ_{ij} equals 1 if function f_i is allocated to node n_j and 0 otherwise. Variable ρ_{ij} represents the service revenue brought by allocating a specific function.

Considering that the three indicators have different optimization directions, we define a utility function as a comprehensive performance reflection, taking the function value of all performance parameters after utility function conversion. The utility value of the k-th solution is calculated as shown in Equation (4), where ω_k is a scaling coefficient used to adjust the influence factor of each performance parameter; $\kappa(r)$ is a conversion function; and r is the independent variable of conversion function $\kappa(r)$. The first parameter defines an increasing function $Y^+(r)$ about R, the second parameter defines a decreasing function $Y^-(r)$ about C, and for temporal performance parameters, we separately define a decreasing function $Y^-(r)$ about T. Here, we normalize the performance parameters to obtain normalized parameters r_k with values between $[0, 1]$ (Equation (5)). Referring to utility function theory, we provide the utility conversion function as shown in Equation (6).

This paper proposes a greedy algorithm for VNF resource allocation. According to different security service requests, we divide the greedy factor variable H in the algorithm into three types: (a) Greedy Least Processing Time (GLPT); (b) Greedy Fast Availability Time (GFAT); (c) Greedy Maximum Memory Space

(GMMS). GLPT refers to allocating VNFs to nodes that can provide the shortest processing time, which satisfies security service types billed by processing time. GFAT refers to allocating VNFs to nodes that can provide service the fastest with the shortest waiting queue, which meets urgent user security service demands. GMMS refers to allocating VNFs to nodes with the largest available storage space, which can balance actual network load and improve comprehensive network performance.

2.2 Algorithm Flow and Description

Algorithm 1 presents the specific process of the proposed algorithm, which consists of the following steps:

a) **Backup physical network state and initialization**: To reduce losses caused by errors during resource allocation, the algorithm first backs up the physical network state and initializes security service chain S, node set N, and greedy factor variable H.

b) **Scan nodes and allocate resources**: For any node n_j in N, calculate its processing end time using Equation (9). If both spatial storage and temporal processing constraints are satisfied (Equation (10)), add the node to the available node set N'; otherwise, scan the next node. After scanning, check set N'. If it remains empty, resource allocation fails, and the backed-up physical network state is reset.

c) **Sort by greedy factor**: For different greedy factors H, sort the nodes and sequentially select the top node for resource allocation, after which the algorithm terminates.

---

3 Resource Scheduling Scheme Based on Hybrid Bee Colony Algorithm

3.1 Algorithm Motivation Analysis

FJSP is an extension of the classic job shop scheduling problem and is a complex combinatorial optimization problem. FJSP can be described as: a processing system with n jobs processed on m machines, where each job contains o_i operations with predetermined processing sequences, and each operation can be processed on multiple machines with different processing times depending on machine performance. The scheduling objective is to allocate operations to machines and determine the processing sequence on each machine under machine capacity constraints and operation precedence constraints. Through comparative analysis, the VNF resource scheduling problem in this paper can be abstracted as FJSP. The mathematical model involves parameter definitions and constraints:

Parameters: n is the number of jobs; m is the number of machines; J_i is the i-th job; O_{ij} is the j-th operation of job i; M_{ijk} is the k-th machine that can process operation O_{ij}; S_{ijk} is the start time; E_{ijk} is the end time.

Constraints: a) **Mutual exclusion constraint**: $\sum_{i=1}^{n}\sum_{j=1}^{o_i} X_{ijk} \leq 1$ indicates that a node can only process one service function at a time. b) **Continuity constraint**: $E_{ijk} = S_{ijk} + p_{ijk} \times X_{ijk}$ indicates that the processing of service functions cannot be interrupted. c) **Precedence constraint**: $S_{ij'k'} - E_{ijk} \geq 0$ if $e_g = k$ and O_{ij} and $O_{i'j'}$ are processed by the same node and O_{ij} is processed first, indicating that service functions in the same security service chain have sequence constraints, while there are no sequence constraints between different service chains.

Traditional bee colony algorithms applied to production scheduling problems mostly form complete scheduling schemes through point-by-point traversal and perform full-cycle operations [16,17]. This method involves large computational load, is severely affected by population size, and point-by-point optimization easily leads to premature convergence and local optima. Therefore, this paper adopts bee colony optimization (BCO) [18] combined with random methods to generate initial populations for employed bees and onlooker bees respectively, which improves search efficiency while enhancing global search capability.

3.2 Algorithm Architecture

The bee colony behavior model of HBC originates from the minimal model of bee foraging behavior established by Karaboga [15], including four components: food sources, employed bees, onlooker bees, and scout bees. In the model, food sources represent different solutions to the problem, and the profitability of food sources (amount of nectar) represents solution quality. Employed bees are responsible for continuously visiting a specific food source and its nearby area for a certain period; onlooker bees collect information about food sources from employed bees, accept each other with a certain probability, and mutually evaluate to retain excellent individuals; scout bees are responsible for randomly searching for new food sources. The three types of bees achieve optimization through collaboration.

FIGURE:6 shows the main framework of the hybrid bee colony algorithm. The main algorithm consists of four parts: population initialization, employed bee algorithm, onlooker bee algorithm, and scout bee algorithm, including four parameters: population size SN, iteration number $mcycle$, adjustment parameter rf, and neighborhood scale $nscale$. Population size SN includes both employed bees and onlooker bees, each accounting for half of the population; iteration number $mcycle$ controls the algorithm termination condition; adjustment parameter rf coordinates the local and global search capabilities of the algorithm, starting with a small value that gradually increases with iterations to ensure convergence; neighborhood scale $nscale$ controls the generation range of neighborhood solutions, starting with a large value to improve the solution space

traversal capability of the bee colony, and gradually decreasing with iterations to narrow the search range and perform fine-grained search of better solutions.

3.3.1 Population Initialization

1) Bee Colony Optimization Algorithm: The BCO algorithm mainly includes forward and backward operations. The forward operation updates the encoding of each employed bee, while the backward operation discards some employed bees according to probability and selects other bees to replace them. FIGURE:7 shows the BCO algorithm flow, where P_c is the current discarding probability of employed bees, calculated as shown in Equation (11), where fit_c is the fitness of the current employed bee and fit_{best} is the best fitness of the current population. Finally, an employed bee colony $\{X_i\}(i = 1, 2, ..., SN/2)$ is generated, and $array_i$ is initialized to record the number of generations without update.

2) Random Method: The random method directly generates entire encodings through random arrays and random number combinations: (a) For each onlooker bee, generate a random array with length equal to the number of service functions as the service function segment encoding; (b) For each service function above, randomly select processing nodes to form the node segment encoding; (c) Traverse onlooker bees to establish population $\{OL_i\}(i = 1, 2, ..., SN/2)$, and initialize array $array_i$ to record the number of generations without update.

3.3.2 Employed Bee Foraging Algorithm

Employed bees generate new solutions through foraging in the neighborhood. For the FJSP problem, this paper adopts an improved neighborhood solution generation method to update the population and establishes the employed bee foraging algorithm $Foraging(i), (i = 1, 2, ..., SN/2)$:

Generate neighborhood solution $Neighbor(X_i)_j$; compare the fitness values of X_i and X_j, and perform greedy selection. If the adjustment parameter rf is satisfied, execute the scout bee algorithm.

Neighborhood Solution Update Algorithm: According to neighborhood scale $nscale$, select two pairs of service function code positions on X_i for exchange, reassign nodes for the swapped service functions, and repeat the above process.

3.3.3 Onlooker Bee Algorithm

The onlooker bee algorithm $Observer(i), (i = 1, 2, ..., SN/2)$ performs information exchange between two bee colonies:

a) For OL_i, select X_i through roulette wheel selection based on fitness.
b) If X_i is better, replace OL_i with X_i.

c) If OL_i is better, decide whether to retain OL_i based on the probability calculated by Equation (11), where $cfit$ is the fitness of X_i and fit is the fitness of OL_i.
d) If the condition is satisfied, perform scout bee operation on OL_i.
e) Execute the neighborhood update algorithm $Neighbor(OL_i)$ on OL_i to generate neighborhood solution OL_i^*.
f) Evaluate the neighborhood solution and determine whether to update OL_i.

3.3.4 Scout Bee Algorithm

The scout bee algorithm $Scout$ is triggered by adjustment parameter rf. When the $array_i$ value corresponding to employed bee X_i or onlooker bee OL_i equals $SN/2$, a random solution is generated to replace the original solution. The random solution is generated using the random population initialization method.

---

4 Experimental Evaluation

4.1 Experimental Environment

To verify the algorithm effectiveness, we developed an independent event simulator using Java and evaluated the proposed algorithms through MATLAB software. Each resource allocation and scheduling event follows a Poisson distribution. We defined 10 different network functions numbered 1-10, with each security service request composed of one or more functions. A security service arrives every 5 time units, and each service function exclusively occupies a node' s resources until processing is completed. Unless otherwise specified, experimental parameters are randomly selected between maximum and minimum values under uniform distribution conditions (TABLE:1). Each independent experiment involves three steps: security service chain arrival, resource allocation and scheduling, and service departure after processing completion.

TABLE:1 Experimental Parameter Ranges | Parameter (Unit) | Minimum | Maximum | |————|——|——| | Number of nodes | | | | Node storage space (MB) | | | | Number of functions per node | | | | Processing time per function (ms) | | | | Storage space per function (MB) | | | | Security service chain length | | | | Security service processing time (ms) | | |

4.2 Algorithm Testing and Results Analysis

Considering that VNF resource allocation and scheduling is a continuous process where allocation results directly affect scheduling performance, we conduct combined tests of greedy algorithms (GLPT, GFAT, and GMMS) and the hybrid bee colony algorithm (HBC). To verify the performance improvement of the proposed population initialization method, we compare it with the traditional bee colony algorithm. Experimental results are shown in FIGURE:8–12.

1) Analysis of Average Waiting Time and Average Processing Time: FIGURE:8 and FIGURE:9 show the average waiting time and average processing time for 1,000 security service chains generated by the independent event simulator. Waiting time refers to the interval between when a service function arrives and when it begins processing. Average waiting time is the mean value of waiting times for all service functions of a security service. Completion time is determined by both processing time and waiting time. FIGURE:8 shows that GLPT-HBC has longer average waiting time compared to other algorithms because GLPT tends to allocate service functions to nodes with the shortest processing time, causing overload on these nodes and resulting in queueing and high average waiting time. GMMS-HBC tends to allocate service functions to nodes with the lowest load, reducing service execution and waiting time. GFAT-HBC demonstrates the best time performance because its optimization objective is to minimize completion time, and waiting time is part of processing time. Therefore, results in FIGURE:9 are consistent with those in FIGURE:8.

2) Analysis of Total Resource Cost and Total Service Revenue: FIGURE:10 and FIGURE:11 show the cumulative total resource cost and total service revenue of resource allocation and scheduling. The values are cumulative because each successful allocation and scheduling calculates resource cost (Equation (2)) and service revenue (Equation (3)), which are added to previous values. FIGURE:10 shows GLPT-HBC has the highest resource cost, GFAT-HBC has the lowest, and all three algorithms have similar growth rates. Minor differences are caused by varying average waiting times. FIGURE:11 shows that as security service requests increase, GFAT-HBC' s service revenue continuously grows and is about 40% higher than the other two algorithms, while GMMS-HBC and GLPT-HBC have similar growth rates. In the long term, GFAT-HBC can provide greater revenue for service providers.

3) Comparative Analysis of Bee Colony Algorithms: FIGURE:12 compares the traditional bee colony algorithm using only random methods for initial population generation with the hybrid bee colony algorithm using the proposed population initialization method. Both algorithms solve the problem under population size 50, iteration number 50, and other consistent parameters, both finding the optimal solution value of 7. Comparative analysis shows that the proposed hybrid bee colony algorithm improves initial solution quality while accelerating convergence speed.

---

5 Conclusion

This paper proposes solutions for VNF resource allocation and scheduling problems in security service chains based on greedy algorithms and hybrid bee colony algorithms. We design three greedy algorithms for different application scenarios to meet diverse user security service demands. Simulation results verify the different processing capabilities of the three algorithms. We innovatively pro-

pose a hybrid bee colony algorithm based on population initialization methods to solve VNF scheduling problems. Experimental results show that this algorithm achieves faster convergence than traditional bee colony algorithms and is suitable for large-scale network demands. Currently, research on virtual network functions in security service chains is continuously emerging. Future work will further improve this research and conduct in-depth experimental verification of service chains.

References

[1] Huang Tao, Liu Jiang, Huo Ru, et al. Survey on future network architecture research [J]. Journal on Communications, 2014, 35(8): 184-197.
[2] Lan Julong, Cheng Dongnian, Hu Yuxiang. Research on reconfigurable information communication infrastructure network architecture [J]. Journal on Communications, 2014(1): 128-139.
[3] Dave T. OpenFlow: enabling innovation in campus networks [J]. ACM Sigcomm Computer Communication Review, 2008, 38(2): 69-74.
[4] Zuo Qingyun, Chen Ming, Zhao Guangsong, et al. Research on SDN technology based on OpenFlow [J]. Journal of Software, 2013(5): 1078-1097.
[5] Xiong Gang, Hu Yuxiang, Duan Tong, et al. A dynamic composition mechanism for security service chains in software-defined networking [J]. Journal of Electronics & Information Technology, 2016, 38(5): 1234-1241.
[6] Liu J, Li Y, Wang H, et al. Leveraging software-defined networking for security policy enforcement [J]. Information Sciences: An International Journal, 2016, 327(C): 288-299.
[7] Clayman S, Maini E, Galis A, et al. The dynamic placement of virtual network functions [C]// Proc of IEEE Network Operations and Management Symposium. 2014: 1-9.
[8] Chiosi M, Clarke D, Willis P, et al. Network functions virtualization—introductory white paper [C]// Proc of SDN and OpenFlow World Congress. 2012: 1-5.
[9] Herrera J G, Botero J F. Resource allocation in NFV: a comprehensive survey [J]. IEEE Transactions on Network & Service Management, 2017, 13(3): 1-5.
[10] Riera J F, Hesselbach X, Escalona E, et al. On the complex scheduling formulation of virtual network functions over optical networks [C]// Proc of IEEE International Conference on Transparent Optical Networks. 2014: 1-5.
[11] Ferrer Riera J, Escalona E, Batalle J, et al. Virtual network function scheduling: concept and challenges [C]// Proc of IEEE International Conference on Smart Communications in Network Technologies. 2014: 1-5.
[12] Mijumbi R, Serrat J, Gorricho J L, et al. Design and evaluation of algorithms for mapping and scheduling of virtual network functions [C]// Proc of IEEE Network Softwarization. 2015: 1-9.
[13] Fischer A, Botero J F, Beck M T, et al. Virtual network embedding: a survey [J]. IEEE Communications Surveys & Tutorials, 2013, 15(4): 1888-1906.

[14] Rabbani M G, Pereira E R, Podlesny M, et al. On tackling virtual data center embedding problem [C]// Proc of IFIP/IEEE International Symposium on Integrated Network Management. 2013: 177-184.
[15] Karaboga D. An idea based on honey bee swarm for numerical optimization, Technical Report-TR06 [R]. 2005.
[16] Chong C S, Low M Y H, Sivakumar A I, et al. Using a Bee colony algorithm for neighborhood search in Job-Shop scheduling problems [C]// Proc of European Conference on Modelling & Simulation: Simulations in United Europe. 2008.
[17] Wong L P, Chi Y P, Low M Y H, et al. Bee colony optimization algorithm with big valley landscape exploitation for Job-Shop scheduling problems [C]// Proc of Conference on Winter Simulation. Winter Simulation Conference. 2008: 2050-2058.
[18] Teodorovic D, Lucic P, Markovic G, et al. Bee colony optimization: principles and applications [C]// Proc of IEEE Seminar on Neural Network Applications in Electrical Engineering. 2007: 151-156.

Note: Figure translations are in progress. See original paper for figures.

Source: ChinaXiv —Machine translation. Verify with original.