Received: 30 June 2019 | Revised: 26 August 2019 | Accepted: 16 October 2019
DOI: 10.1002/ett.3809

SPECIAL ISSUE ARTICLE

WILEY

Path-flow matching: Two-sided matching and multiobjective evolutionary algorithm for traffic scheduling in cloud data* center network

Lizhuang Tan[1] | Wei Su[1] | Shuai Gao[1] | Jingying Miao[1] | Yuan Cheng[1] | Peng Cheng[2]

[1]National Engineering Laboratory for Next Generation Internet Interconnection Devices, School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing, China

[2]College of Computer & Communication Engineering, China University of Petroleum, Qingdao, China

Correspondence
Lizhuang Tan, National Engineering Laboratory for Next Generation Internet Interconnection Devices, School of Electronics and Information Q1 Engineering, Beijing Jiaotong University, Beijing 100044, China.
Email: lzhtan@bjtu.edu.cn

Funding information
Fundamental Research Funds for the Central Universities of China, Grant/Award Number: 2018YJS002; CERNET Innovation Project, Grant/Award Number: NGII20180120; National Key R&D Program of China, Grant/Award Number: 2018YFB1800305; Jingmen Science and Technology Research and Development Plan Project, Grant/Award Number: 2018YFYB055

Abstract
Improving the operational efficiency of data center has always been an important direction for the development of ICT. In this paper, we apply two-sided matching decision-making process in game theory to traffic scheduling problem in data center network. From the perspective of matching between flow and path, the traffic scheduling is properly arranged. We first propose and model the path-flow matching problem, considering the preference ordering, then formulate the problem as a multiobjective optimization problem with the target to ensure the stability and satisfaction from the matching scheme, and design a preference-based path-flow ordering method Extended PIAS, and finally propose a lightweight scheduling algorithm LinkGame based on multiobjective evolutionary algorithm. Compared with the previous scheduling methods (ECMP, Hedera, and Fincher), experiment results demonstrate that LinkGame can simultaneously consider the stability and satisfaction of the matching results, with improved bandwidth utilization and flow completion time.

1 | INTRODUCTION

Data centers play a significant role in Big data, Internet of Things, and Cloud computing. In order to provide users with high-quality cloud services, some large Internet companies, eg, Microsoft, Google, Amazon, and Alibaba, have built a great many data centers around the world. According to Synergy Research,[1] there were 430 hyperscale data centers (with 50 000 to 100 000 servers) in the world in 2018. Within the data center, tens of thousands of servers are connected with high-bandwidth (10 to 100 Gbps) and low-latency (10 to 100 μs) data center network. Therefore, the data center network acts as a network infrastructure, carrying the most advanced information and communication technologies. In order to save costs, data centers prefer to use inexpensive switches to build networks.[2] How to make full use of abundant network resources in data center network has always been a problem for data center operators. Other than this, there are many latency-sensitive applications running in the data center, like e-commerce retail, web search, and social networking. User

*[Correction added on 8 January 2020, after first online publication: the article title has been corrected in this version].

requests for these latency-sensitive applications need to be responded as quickly as possible. In order to provide better services, data center traffic scheduling is extremely important and necessary.[3]

Currently, data centers use Equal-Cost MultiPath (ECMP)[4] as the basic load balance routing protocol. The traditional ECMP-based load balancing method ensures the equalization of different path utilization rates in the network by uniformly distributing the data flows to different transmission paths. ECMP does not consider the impact of long and short flows on network load balance.[5] Delay-sensitive short flows tend to be queued after long flows and cannot be effectively forwarded in time.[6] In addition, ECMP cannot detect network congestion. The mechanism that assigns flows to different paths by hashing or polling may result in congested links become more congested. In summary, the task of data center network traffic scheduling is to make appropriate arrangements for flows and paths. This arrangement can be very similar to the market matching problem in economics. Therefore, in this paper, we try to use the research method of economics to solve the traffic scheduling problem in data center network, for example, two-sided matching algorithm.

Two-sided matching decision is an important content in game theory.[7] It studies how to find the best matching suggestion for two-sided matching participants relying on the preference information provided. Two-sided matching participants are finite and disjoint. In the two-side matching decision process, one matching participant usually gives preference ordering information for the other relative participant. Common preference ordering information includes strong preference order (eg, better), weak preference order (eg, not inferior to), indifference preference order (eg, equivalent), and missing preference order (eg, not known). The main research of two-sided matching decision-making issues include marriage matching and employee matching.

The traffic scheduling problem in the cloud data center network meets the basic form of two-sided matching decision. Paths and flows can be thought of as the participants. The requirements of the flow (eg, flow completion time (FCT), and transmission rate) and the state of the path (eg, remaining bandwidth) are the basic information for the preference ordering. Network decision-maker (usually SDN controller or load balancer) designs appropriate traffic delivery schemes based on the preference ordering. We can refer to the problem of two-sided matching of paths and flows in a cloud data center network as a path-flow matching problem.

At present, most of the researches on two-sided matching solves the matching problem based on strong preference ordering or indifference preference ordering and pays attention to two-sided stability matching. However, the matching scheme usually is stable but not optimal. In a cloud data center, obtaining a stable traffic scheduling is not the only goal that operators expect. Operators prefer to get a scheduling that is optimal. Therefore, the goal of the path-flow matching problem can be described as seeking a matching solution that combines stability and satisfaction.

Motivated by the aforementioned analysis, the two-sided matching method in game theory is applied to the traffic scheduling of the data center network. In this paper, Section 2 summarizes the related work to data center network traffic scheduling and two-sided matching. Section 3 analyzes the matching relationship in data center network traffic scheduling problem and proves the feasibility of applying two-sided matching theory to data center network. Section 4 proposes and models the path-flow matching problem, considering the preference ordering for data center network, and designs a preference-based path and flow ordering scheme Extended PIAS. Section 5 proposes a matching model, solving algorithm LinkGame, which can simultaneously consider the stability and optimization of the path-flow matching results. Section 6 presents the simulation experimental results and discussion. Section 7 concludes the whole paper and points out the future work.

2 | RELATED WORK

Currently, data center network traffic scheduling can be divided into three categories: centralized control,[3,8-10] distributed global congestion-aware,[11-13] and distributed local stateless.[14,15] These papers take FCT or bandwidth utilization as the optimization goal and study the congestion sensing mechanism and deployment location. Designing efficient and scalable flow scheduling algorithms and deploying them are also the goal of these papers. These works remind us not only to pay attention to the implementation mechanism but also to study the theoretical feasibility of the scheduling algorithm.

Zhang et al[16] described subcarrier assignment problem for wireless networks as a many-to-many matching games, which reminds us that game theory, stable matching, and two-sided matching have been widely used to solve resource allocation and management problems in wireless networks. At the same time, there are some solutions to apply game theory to resource management and traffic scheduling in data center network. Therefore, the use of two-sided matching can better solve the data center network traffic scheduling problem.

Xu et al[17] proposed Anchor, which is the first work to use the stable matching for resource management in the Cloud. Anchor used the stable matching framework to decouple policies from mechanisms when mapping virtual machines to physical servers. Wang et al[18] used the stable matching theory for the SDN controller assignment in the data center network and formulated SDN controller assignment problem as a stable matching problem with transfers. Hosseini et al[19] introduced a developed model based on the game theory for identifying vulnerable data centers in cloud computing. In particular, they also presented a measure of the degree of vulnerability of data centers in cloud computing network. These three works used game theory for data center resource management, which reminds us whether we can use matching theory for traffic scheduling problem.

Zhang et al[20] used the stable matching for the data center network elephant flow scheduling and proposed the Fincher algorithm. Fincher transformed the elephant flow scheduling problem into a matching problem between flow and switch, creatively using the available storage space of the switch as a consideration for path allocation. However, the stable matching solution obtained by the Fincher algorithm is not necessarily optimal. This work prompted us to use the two-sided matching considering the preference ordering to find the optimal solution for traffic scheduling.

Abououf et al[21] proposed to use the Gale-Shapley matching game selection to allocate multiple workers to multiple tasks based on the preferences of both the tasks and workers. The matching result relying on the satisfaction from both parties is better than relying on stability. Although this paper is not a paper in the field of networking, it inspired us to consider the importance of satisfaction and stability for two-sided matching problem.

Liu et al[22] established three private link sets for three types of flows in data center network and proposed a mix-flow scheduling scheme DRL-Flow based on deep reinforcement learning. The three flow-scheduling strategies were priority-based allocation for mice flows, stable matching-based allocation for elephant flows with unknown sizes, and proportion-based allocation for elephant flows with known sizes. DRL-Flow used the idea of matching in different situations, which adds complexity to the actual deployment.

The two-sided matching problem can ultimately be regarded as a multiobjective optimization problem (MOP), which can be solved by heuristic algorithm or evolutionary algorithm.

Zeng et al[23] jointly considered and formulated switch activation and flow routing as an integer linear programming problem and proposed a heuristic algorithm to deal with its high computational complexity. Extensive simulation-based evaluations are conducted to validate the high efficiency of our algorithm.

Wang et al[18] proposed a hierarchically two-phase algorithm for dynamic SDN controller assignment in data center networks and integrated key concepts from both matching theory and coalitional games to solve it efficiently.

Bastam et al[24] used a linear programming to define the problem statement and then modeled the problem as a path based multicommodity flow. Then, a suitable iterative algorithm is utilized to quickly converge subproblem solutions leading to an answer to the problem. As a result, the proposed method serially yields an optimum solution, which takes significantly shorter time to reach than that of the generic (the LP model that is not decomposed) approach.

3 | MATCHING IN DATA CENTER NETWORK

3.1 | Equivalent link

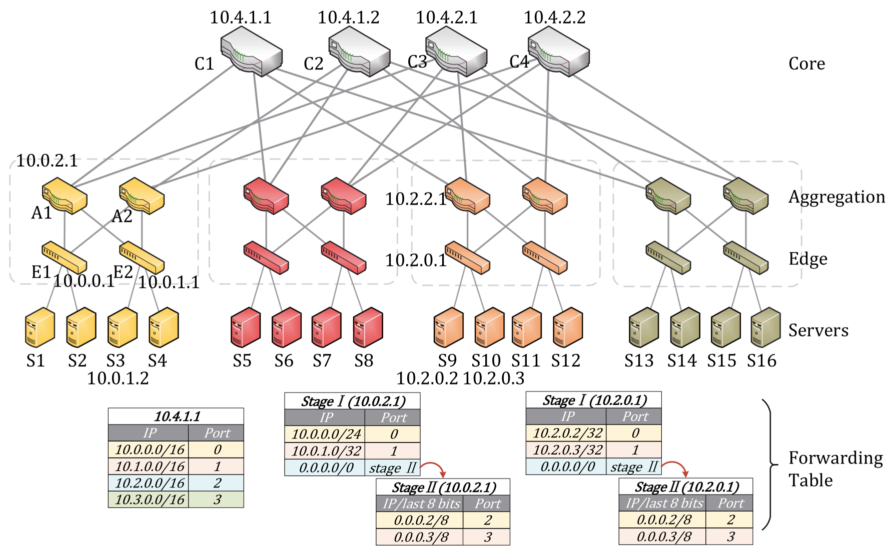
Data center network mostly adopts a layered architecture. Typical architectures include Fat-Tree,[25] Helios,[26] BCube,[27] and more. In these architectures, switches are divided into edge layer, aggregation layer, and core layer according to different functions and performances.

Figure 1 shows a typical Fat-Tree architecture. Sixteen servers are linked to the edge switches, which is the bottom layer of the network. The second layer is composed of the aggregation switches, and the top layer is 4 core switches.

In a fat-tree network, all switch IP addresses are related to network topology information.[28] All edge and aggregation layer switches have a two-stage forwarding table. All core switches only need to maintain four forwarding rules. For example, Figure 1 lists the forwarding tables of the core switch 10.4.1.1, the aggregation switch 10.0.2.1, and the edge switch 10.2.0.1. In Fat-tree architecture, the special numbering method and forward table for switches facilitate load balancing.

In data center network, there are large number of optional paths from a source server to a destination server. These optional paths generally have the same cost metric in the routing table of the switch. Therefore, switches typically use ECMP mechanisms (IP 5-tuple hashes) to spread traffic across different optional paths.[29]

The concept of an equivalent link is defined here: for two directed links $L1 \rightarrow M1$ and $L2 \rightarrow M2$ ($L1$ and $L2$ are switches at the beginning of the directed link, $M1$ and $M2$ are switches at the end of the directed link), if $L1$ and $L2$ are in the same layer, $M1$ and $M2$ are also on the same layer. For any one flow, both links can be used to carry the flow, or neither can be used to carry the flow. The two links are mutually equivalent. It should be noted that the equivalent links are not



10.4.1.1	
IP	Port
10.0.0.0/16	0
10.1.0.0/16	1
10.2.0.0/16	2
10.3.0.0/16	3

Stage I (10.0.2.1)	
IP	Port
10.0.0.0/24	0
10.0.1.0/32	1
0.0.0.0/0	stage II

Stage II (10.0.2.1)	
IP/last 8 bits	Port
0.0.0.2/8	2
0.0.0.3/8	3

Stage I (10.2.0.1)	
IP	Port
10.2.0.2/32	0
10.2.0.3/32	1
0.0.0.0/0	stage II

Stage II (10.2.0.1)	
IP/last 8 bits	Port
0.0.0.2/8	2
0.0.0.3/8	3

FIGURE 1 Fat-tree architecture and key switch forwarding table

necessarily identical. If the two links have different capacities but meet the aforementioned definitions, they should also be considered as equivalent links because the cost metrics of the path selections represented by different links in the switch routing table remains unchanged. For a flow from server $S3(10.0.1.2)$ to server $S9(10.2.0.2)$, $E2(10.0.1.1) \rightarrow A1(10.0.2.1)$, and $E2(10.0.1.1) \rightarrow A2(10.0.2.2)$ are equivalent link in the upward direction (the direction in which the lower layer switch forwards to the higher layer switch).

The key of data center network traffic scheduling is to distribute traffic to these equivalent links as evenly as possible, which is extremely similar to two-sided matching in game theory.

3.2 | Path-flow matching problem

Two-sided matching decision is extremely important in game theory. According to the number of matching objects, it can be divided into 1-1 matching, 1-n matching, and m-n matching. Traditional research issues include marriage matching,[30] medical residency matching,[31] partner matching,[32] and venture capital matching.[33] Scholars have proposed two-sided matching algorithms for solving different matching problems, such as Gale-Shapley algorithm,[34] Hospital-Resident algorithm,[35] and RANDBRK algorithm.[36]

The matching of paths and flows is also a two-sided matching process. For example, in a data center network, ECMP hash defects can cause a link overload, so traffic needs to be evenly distributed across all paths. Therefore, we can get the path-flow matching problem. The research content of path-flow matching problem is how to establish the matching relationship between flow and path to get the most satisfactory and stable routing scheme.

Unlike the traditional m-n matching problem, the path-flow matching does not have any prior knowledge about preference ordering in the matching process, for example, how many flows a path should match. Therefore, the path-flow matching problem is a dynamic special m-n matching problem. Path-flow matching not only pursues the stability of the matching scheme but also pays more attention to the satisfaction from the matching scheme. In addition, path-flow matching comes with a number of constraints, which makes the problem solving difficult and challenging.

4 | PATH-FLOW MATCHING MODEL

Assume that the set of paths is $P = \{p_1, p_2, p_3, \dots, p_m\}$, where p_i represents the ith path can carry traffic and $i = 1, 2, \dots, m$. The set of flows waiting to be scheduled is $F = \{f_1, f_2, f_3, \dots, f_n\}$, where f_j represents the jth waiting flow and $j = 1, 2, \dots, n$. Each path consists of several links. Each flow is identified by 5-tuple: source/destination IP, source/destination port number, and transport protocol.

The number of flows that path p_i expects to match is c_i, $\sum_{i=1}^{m} c_i = n$, and c_i is not greater than the maximum number of flows that the path p_i is allowed to carry. The number of paths that the flow f_j expects to match is 1. Preference ordering of path to flow is $R_i = f_{v_i(1)} \bigcirc f_{v_i(2)} \dots \bigcirc f_{v_i(n-1)} \bigcirc f_{v_i(n)}$, where $f_{v_i(1)}, f_{v_i(2)}, \dots, f_{v_i(n)} \in F$ and $v_i(1), v_i(2), \dots, v_i(n) \in [1, n]$. $f_{v_i(1)}$ means that flow $f_{v_i(1)}$ is in first. Preference ordering of flow to path is $O_j = p_{u_j(1)} \bigcirc p_{u_j(2)} \dots \bigcirc p_{u_j(m-1)} \bigcirc p_{u_j(m)}$, where

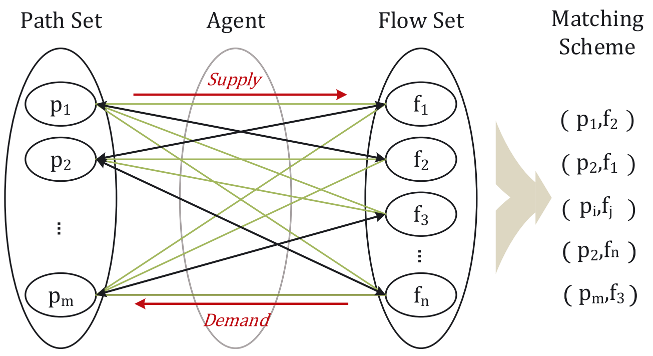


FIGURE 2 Path-flow matching considering the preference ordering in data center network

$p_{u_j(1)}, p_{u_j(2)}, \ldots, p_{u_j(m)} \in P$ and $u_j(1), u_j(2), \ldots, u_j(m) \in [1, m]$. $p_{u_j(1)}$ means that path $p_{u_j(1)}$ is in first. $\bigcirc$ represents the relationship between paths and flows. Preference ordering R_i and O_j should be a satisfactory to unsatisfactory ordering.

According to the actual situation of the network, the element relationship $\bigcirc$ in the preference ordering set R_i and O_j mainly has the following forms.

- $f_{v_i(1)} \succ f_{v_i(2)}$: For the path p_i, the flow $f_{v_i(1)}$ is superior to the flow $f_{v_i(2)}$. That is to say, compared to the flow $f_{v_i(2)}$, the path p_i is more suitable to carry the flow $f_{v_i(1)}$.
- $p_{u_j(1)} \succ p_{u_j(2)}$: For the flow f_j, the path $p_{u_j(1)}$ is superior to the path $p_{u_j(2)}$. That is to say, compared with $p_{u_j(2)}$, the path $p_{u_j(1)}$ can better satisfy the requirements of the flow f_j for bandwidth, delay, packet loss, etc.
- $f_{v_i(1)} \prec f_{v_i(2)}$: For the path p_i, the flow $f_{v_i(1)}$ is inferior to the flow $f_{v_i(2)}$.
- $p_{u_j(1)} \prec p_{u_j(2)}$: For the flow f_j, the path $p_{u_j(1)}$ is inferior to the path $p_{u_j(2)}$.
- $p_{u_j(2)} \leftrightarrow p_{u_j(3)}$: For the flow f_j, the path $p_{u_j(2)}$ is equivalent to the path $p_{u_j(3)}$.
- $f_{v_i(2)} \succcurlyeq f_{v_i(3)}$: For the path p_i, the flow $f_{v_i(2)}$ is better or equivalent to the flow $f_{v_i(3)}$.
- $p_{u_j(3)} \succcurlyeq p_{u_j(4)}$: For the flow f_j, the path $p_{u_j(3)}$ is better or equivalent to the path $p_{u_j(4)}$.
- $\ominus f_{v_i(4)}$: The path p_i cannot carry the flow $f_{v_i(4)}$.
- $\ominus p_{u_j(5)}$: For the flow f_j, the path $p_{u_j(5)}$ is unreachable or cannot meet its requirements for bandwidth, delay, packet loss, etc.

For example, $R_1 = f_5 \succ f_3 \succ f_1 \succcurlyeq f_2 \ominus f_4$ represents the preference ordering of f_1,f_2,f_3,f_4 and f_5 for path p_1. f_5 orders first, with the highest satisfaction from p_1. Meanwhile, p_1 cannot carry the flow f_4.

As shown in Figure 2, we call the two-sided matching between the path set P and the flow set F as a path-flow matching. The result of path-flow matching is to achieve the matching between path and flow.

According to the definition of two-sided matching, we can get the definition of path-flow matching.

Definition 1. Path-flow matching is defined as mapping μ: $P \cup F \rightarrow P \cup F$. For $\forall p_i \in P, \forall f_j \in F$, μ satisfies the following conditions.

1. $\mu(p_i) \in F \cup p_i$. If $\mu(p_i) = p_i$, the match fails, which means that the path p_i does not match the appropriate flow.
2. $\mu(f_j) \in P \cup f_j$. If $\mu(f_j) = f_j$, the match fails, which is not allowed in path-flow matching.
3. If $\mu(p_i) = f_j$ and $\mu(f_j) = p_i$, (p_i, f_j) is a matching pair of path-flow matching.
4. $\mu(p_i) \cap \mu(p_k) = \varnothing, i \neq k$.
5. $|\mu| = \max\{m, n\}$.
6. $\mu(f_j)$ is noninfective and surjective, so that all flows can be scheduled.

4.1 | Preference ordering

In order to realize the process of the game in traffic scheduling, path-flow matching model needs to order all flows and paths. The purpose of preference ordering is to obtain a preference matrix by describing all paths and flows.

For path p_i, all flows in the network are divided into two types: flows that can be carried and flows that cannot be carried. The flows that cannot be carried is mainly due to the unreachable route, which is represented by $\ominus f_{v_i(j)}$ in the preference ordering. For example, in Figure 1, the flow f_j (10.0.1.2, 10.2.0.2, 60546, 80, TCP) is a flow that can be carried by the path p_i (10.0.0.1 → 10.0.2.1 → 10.4.1.1 →10.2.2.1 →10.2.0.1).

The ordering R_i of individual preferences depends on the path p_i having a good understanding of the flows it can carry. Most previous proposals, such as PDQ,[37] pFabric,[38] and PASE,[39] assumed prior knowledge of accurate flow information, eg, flow sizes or deadlines. However, such prior knowledge of flow is difficult to obtain in real network. Filling the QoS

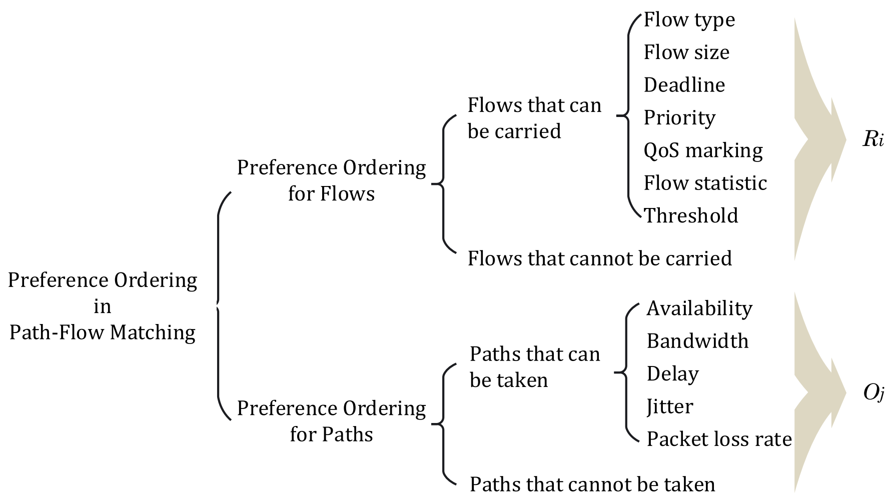

Preference Ordering in Path-Flow Matching
Preference Ordering for Flows
Flows that can be carried
Flow type
Flow size
Deadline
Priority
QoS marking
Flow statistic
Threshold
Flows that cannot be carried
Ri
Preference Ordering for Paths
Paths that can be taken
Availability
Bandwidth
Delay
Jitter
Packet loss rate
Paths that cannot be taken
Oj

FIGURE 3 Classification and factors of preference ordering in path-flow matching

marking and priority with the packet header extension field is the most ideal solution. The controller can determine the preference ordering based on the values of the extension fields. However, it requires the server or network proxy to have the corresponding encapsulation capabilities, which is not possible in some data center networks.

Preference ordering can also be achieved using port number or flow type. Different applications have different transmission requirements for FCT and bandwidth. Communication between different servers often uses sockets to mark different flows, so the port number can be used as the identity of the flows. Through the value of the port number field, the controller can complete the path-to-flow preference ordering. Since the port number of a certain service may change in practical applications, the accuracy of ordering by port number is not high.

Similar to R_i, for O_j, not all paths can carry flow f_j. A path that cannot carry flow f_j is represented by $\ominus p_{u_j(i)}$. We use the remaining bandwidth of the link to order the paths, which can carry flow f_j. In addition, the buffer queue length of the switch, which the flow passes through, is also recommended as the basis for ordering.[20] Although delay jitter is the most direct factor for FCT, it is not suitable as the preferred basis for preference ordering due to the difficulty of end-to-end measurement.

Figure 3 shows a summary of the influencing factors of R_i and O_j.

In conclusion, the flow-to-path preference ordering O_j is relatively easy to obtain, but the path-to-flow preference ordering R_i is more difficult to obtain. Section 4.2 introduces a path-to-flow preference ordering method.

4.2 | Extended PIAS preference ordering

The calculation of the path-to-flow preference ordering matrix R_i is more complicated. Inspired by PIAS,[40] we designed a flow preference ordering matrix R_i calculation scheme called Extended PIAS, using the flow table counter in the existing Openflow protocol. The Extended PIAS follows the Shortest Job First principle and does not require flow prior information of the flow and can balance the fairness of long and short flows. Compared with PIAS, the basic idea of Extended PIAS is to transfer the work done by the terminal or switch to the controller, which can minimize the impact on the terminal.

PIAS leverages multiple priority queues available in existing switches to implement multiple level feedback queue (MLFQ). Packets in different queues of MLFQ are scheduled with strict priority. However, the shortcoming of PIAS is that the ordering of the flow by the switch requires a large amount of threshold control message information, which increases the network load.

Extended PIAS transfers the flow ordering work, which should have been done on the switch to the controller. Controller periodically sends an OFPT_STATS_REQUEST message to the OpenFlow switch for obtaining statistic result about the flow table on the switch. By comparing the change values of the latest flow table matching statistical results, the controller classifies the corresponding flows. Extended PIAS just cares about the variation of flow table matches and does not care about the value of historical matches, thus achieving the fairness to long flow and short flow. The flow priority is positively correlated with the variation of the flow matching result. The more severe the flow change, the higher the flow priority.

As shown in Figure 4, Extended PIAS can order flows according to the changes of the flow table matching. Based on the results of statistics, the controller calculates the variation of the flows corresponding to the different flow table items.

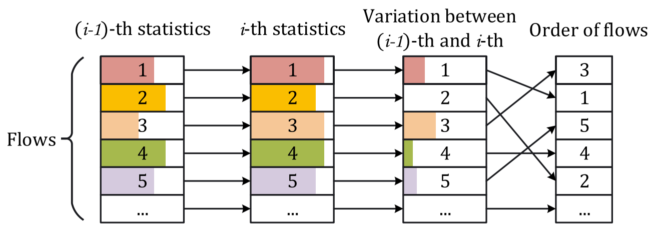


FIGURE 4 Extended PIAS preference ordering

No. 3 flow increases steeply, so it has the highest priority. During the same period, No. 2 flow has almost no new packets, so it has the lowest priority.

4.3 | Satisfactory two-sided matching

The traffic scheduling in the data center not only pursues the stability of the solution but also expects to obtain an optimal solution. We refer to the optimal stable scheduling scheme as a satisfactory two-side matching scheme.

According to preference ordering R_i and O_j, we can establish the preference ordering weight matrix $R = [r_{ij}]_{m\times n}$ and $O = [o_{ij}]_{m\times n}$. Since the operation of the weight matrix is easier than the operation of the preference ordering matrix in the calculation process, the path-flow matching model uses the weight matrix to solve the model in the latter calculation process. r_{ij} represents the satisfaction value of p_i for f_j. o_{ij} represents the satisfaction value of f_j for p_i. The higher the preference ordering, the larger the satisfaction value.

Definition 2. For path-flow matching μ, if $\exists a, b \in [1, n]; \exists c, d \in [1, m]$, where $\mu(f_a) = p_c, \mu(p_c) = f_a, \mu(f_b) = p_d, \mu(p_d) = f_b$ does not let

$$r_{ad} + r_{bc} \leq r_{ac} + r_{bd} \tag{1}$$

$$o_{ad} + o_{bc} \leq o_{ac} + o_{bd} \tag{2}$$

exist at the same time. μ is a satisfactory two-side matching.

Definition 3. If satisfactory two-side matching μ does not let

$$r_{ad} \leq r_{ac} \tag{3}$$

$$r_{bc} \leq r_{ac} \tag{4}$$

$$o_{ad} \leq o_{ac} \tag{5}$$

$$o_{bc} \leq o_{ac} \tag{6}$$

exist at the same time. μ is a stable two-sided matching.

Theorem 1. *If a satisfactory two-sided matching μ is stable, then μ may be the optimal two-sided matching.*[41]

Proof. The optimal two-sided matching scheme of path-flow matching problem must be satisfactory and stable. Satisfactory matching schemes are often the most basic solutions to the problem. If a satisfactory two-sided matching μ is stable, μ must be a solution in the optimal solution space. □

Definition 4. For satisfactory two-sided matching μ_1 and μ_2, if

1. $\forall f_i \in F, \mu_1(f_i) = p_j, \mu_2(f_i) = p_k, o_{ij} \succcurlyeq o_{ik}$, and $\exists f_l \in F, \mu_1(f_l) = p_q, \mu_2(f_l) = p_s, o_{lq} \succ o_{ls}$,
2. $\forall p_j \in P, \mu_1(p_j) = f_i, \mu_2(p_j) = f_k, o_{ij} \succcurlyeq o_{kj}$, and $\exists p_t \in P, \mu_1(p_t) = f_q, \mu_2(p_t) = f_s, o_{qt} \succ o_{st}$,

then μ_1 is Pareto dominance than μ_2.

Definition 4 means, compared to μ_1, μ_2 has no better path-flow matching pair.

Definition 5. If there is no other matching scheme Pareto dominance than the matching scheme μ, then μ is the Pareto effective matching scheme of path-flow matching problem, that is, the matching scheme μ is the optimal satisfactory two-sided matching.

5 | PATH-FLOW MATCHING PROBLEM SOLVING

Obviously, the path-flow matching problem is an MOP.[42] Both parties (paths and flows) involved in the matching process want to maximize their satisfaction without affecting the stability of the matching results. Therefore, this section gives the multiobjective optimization model of the path-flow matching problem and the solution algorithm LinkGame.

5.1 | Multiobjective optimization model

Path-flow matching is an MOP. We introduce the 0-1 variable x_{mn}, where $x_{ij} = 0$ indicates that p_i and f_j cannot form a matching relationship, and $x_{ij} = 1$ indicates that p_i and f_j can form a matching relationship, then (p_i,f_j) becomes a matching pair in the matching scheme μ.

According to Section 4, we can build the following multiobjective optimization model:

$$\max \quad z_1 = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \cdot o_{ij}, i = 1, 2, \ldots, m; j = 1, 2, \ldots, n \tag{7}$$

$$\max \quad z_2 = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \cdot r_{ij}, i = 1, 2, \ldots, m; j = 1, 2, \ldots, n \tag{8}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} x_{ij} \leq n, j = 1, 2, \ldots, n \tag{9}$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 1, 2, \ldots, m \tag{10}$$

$$x_{ij} + \sum_{k:o_{ik} \geqslant o_{ij}} x_{ik} + \sum_{k:r_{kj} \geqslant r_{ij}} x_{kj} \geq 1 \tag{11}$$

$$x_{ij} = 0, 1. \tag{12}$$

In the aforementioned equations, Equation (7) is the flow satisfaction objective function, and Equation (8) is the path satisfaction objective function. Equation (9) ensures that the number of flows that can be carried by each path cannot exceed the maximum number of flows, and Equation (10) makes each flow get only one matching from all paths. Equation (11) is the matching stable constraints. A complete x_{mn} can be called a feasible solution Z. The meaning of this multiobjective optimization model is to find an optimal matching scheme Z_{opt}.

Theorem 2. *For any one path-flow matching μ, there must be an optimal two-sided matching scheme.*

Proof. Path-flow matching μ is a multiobjective 0-1 integer programming problem with $m \times n$ variables, which can produce up to $2^{(m \times n)}$ feasible solutions. Since $\ominus p_j$ and $\ominus f_i$, the solution space is much smaller than $2^{(m \times n)}$. It means that the feasible solution of path-flow matching μ is finite. According to the previous research results,[43] the 1-1 two-sided matching must exist a stable matching solution. In other words, if the feasible domain composed of constraints is not empty and both Equation (7) and Equation (8) have an optimal solution, path-flow matching μ has an optimal solution. □

Based on Equations (7) and (8), we can construct the objective function of the MOP as follows:

$$\max \quad Z(x) = (z_1(x), z_2(x))^T, x \in \Omega, \tag{13}$$

where Ω is the feasible solution space.

Path-flow matching problem model is a multiobjective 0-1 integer programming model. We establish model evaluation indicators by membership function method. It is assumed that $z_1^{\max}$ and $z_2^{\max}$ are the single objective optimal values of Equation (7) and Equation (8) when only the targets z_1 and z_2 are considered, and $z_1^{\min}$ and $z_2^{\min}$ are the worst values of the single target, respectively. The membership functions ρ_{z_1} and ρ_{z_2} of two single targets can be expressed as follows:

$$\rho_{z_1} = \frac{z_1^{\max} - z_1}{z_1^{\max} - z_1^{\min}} \tag{14}$$

$$\rho_{z_2} = \frac{z_2^{\max} - z_2}{z_2^{\max} - z_2^{\min}}. \tag{15}$$

Let α and β are the weights of the targets ρ_{z_1} and ρ_{z_2}, respectively. $0 \leq \alpha, \beta \leq 1$, and $\alpha + \beta = 1$. Define the decision preference function V, ie,

$$V = \alpha \cdot \rho_{z_1} + \beta \cdot \rho_{z_2}. \tag{16}$$

The decision preference function V reflects the importance of satisfaction between the two parties in the matching process. We will detail how decision preferences are determined in Section 5.2.

5.2 | LinkGame algorithm

Evolutionary algorithm is a swarm intelligence search method for solving MOPs. Decomposition-based multiobjective evolutionary algorithm (MOEA/D) is a new decomposition algorithm that combines mathematical programming methods with evolutionary algorithms to solve MOPs.[44] Compared with other MOEAs, MOEA/D has obvious advantages in solving complex MOPs.

MOEA/D uses a decomposition strategy to transform an MOP into a number of single-objective optimization subproblems, and then uses evolutionary algorithms to simultaneously solve these single-object subproblems. The objective function of each subproblem is an aggregate function for each objective function, and the population of the algorithm consists of the current optimal solution of each subproblem. The optimization of each subproblem is done by an evolutionary operation between the adjacent subproblems. The neighbor relationship between subproblems is determined by the distance between the weight vectors of the subproblems. The two subproblems with similar weight vectors are similar.

In this paper, we take an MOEA/D based on uniform subproblem weight vector design for solving the path-flow matching model.[45] This algorithm is called LinkGame.

LinkGame uses a uniform design method to set the weight vector of each subproblem decomposed by MOEA/D,[46] so that the algorithm can search all regions evenly at the initial stage and improve the possibility of finding the Pareto optimal solution. For each Pareto optimal solution x^* of a multiobjective problem, there must be a weight vector λ such that x^* corresponds to an optimal solution to the single-objective problem. In addition, LinkGame ensures that the matching result is a stable matching scheme with the high satisfaction of both parties. Finally, the optimal solution in the satisfactory two-sided matching solution is output.

We assume that the population size in LinkGame is N, then $\lambda = \lambda^1, \lambda^2, \lambda^3, \ldots, \lambda^{N-1}, \lambda^N$ is a uniformly distributed set of subquestion weight vectors. N is the population size. As a result, we decompose multiobjective evolution problem into N single-objective optimization problems. λ^i can be expressed as follows:

$$\lambda^i = \left(\lambda_1^i, \lambda_2^i\right) = \left(\frac{i-1}{N}, \frac{N-i+1}{N}\right), i = 1, 2, \ldots, N. \tag{17}$$

Therefore, we can use the Tchebycheff method to split a multiobjective problem into N single-objective problems.[47] Equation (17) is equivalent to Equation (16). The ith single-objective optimization problem can be expressed as follows:

$$\min \quad g_i(x|\lambda^i, z^*) = \max\left\{\lambda_1^i \left|f_1(x) - z_1^*\right|, \lambda_2^i \left|f_2(x) - z_2^*\right|\right\}, \tag{18}$$

where the neighbor of the weight vector λ^i is the weight vector in λ closest to λ^i. LinkGame optimize these N subproblems simultaneously in an evolutionary process. Particularly, in Equation (18), each generation of population consists of the current optimal solution of each subproblem, and the optimization of each subquestion only requires its neighbor information. z^* is the reference point for the whole problem, which represents the estimate of the optimal solution. In path-flow matching problem, z^* can be expressed as follows:

$$z^* = (\max z_1, \max z_2). \tag{19}$$

$\max z_1$ is the maximum preference value of the flow and $\max z_2$ is the maximum preference value of the path. They can be calculated in the following way:

$$z^* = \left(\sum_{i=1}^{m} \max\{o_{i1}, o_{i2}, \ldots, o_{in}\}, \sum_{j=1}^{n} \max\{r_{1j}, r_{2j}, \ldots, r_{nj}\}\right), i = 1, 2, \ldots, m; j = 1, 2, \ldots, n. \tag{20}$$

In each iteration, the LinkGame needs to save the following three necessary information:

1. the current optimal solution for each subproblem: $x^1, x^2, \dots, x^N$;
2. the objective function value corresponding to each optimal solution: $FV^1, FV^2, \dots, FV^N$;
3. the optimal objective function solution currently found: $z = (z_1, z_2)^T$.

As shown in Figure 5, LinkGame algorithm consists of four steps, namely, initialization operation (IO), breeding operation (BO), correction operation (CO), and update operation (UO).

The IO mainly completes the random generation of the population $x^1, x^2, \dots, x^N$ and calculates the reference point z^* according to the two preference matrices R and O.

The BO picks out two parents from all the populations, denoted as r_1 and r_2, respectively. Both r_1 and r_2 belong to $x^1, x^2, \dots, x^N$. BO takes a crossover operator for r_1 and r_2 to generate child y, and then mutates y with the mutation

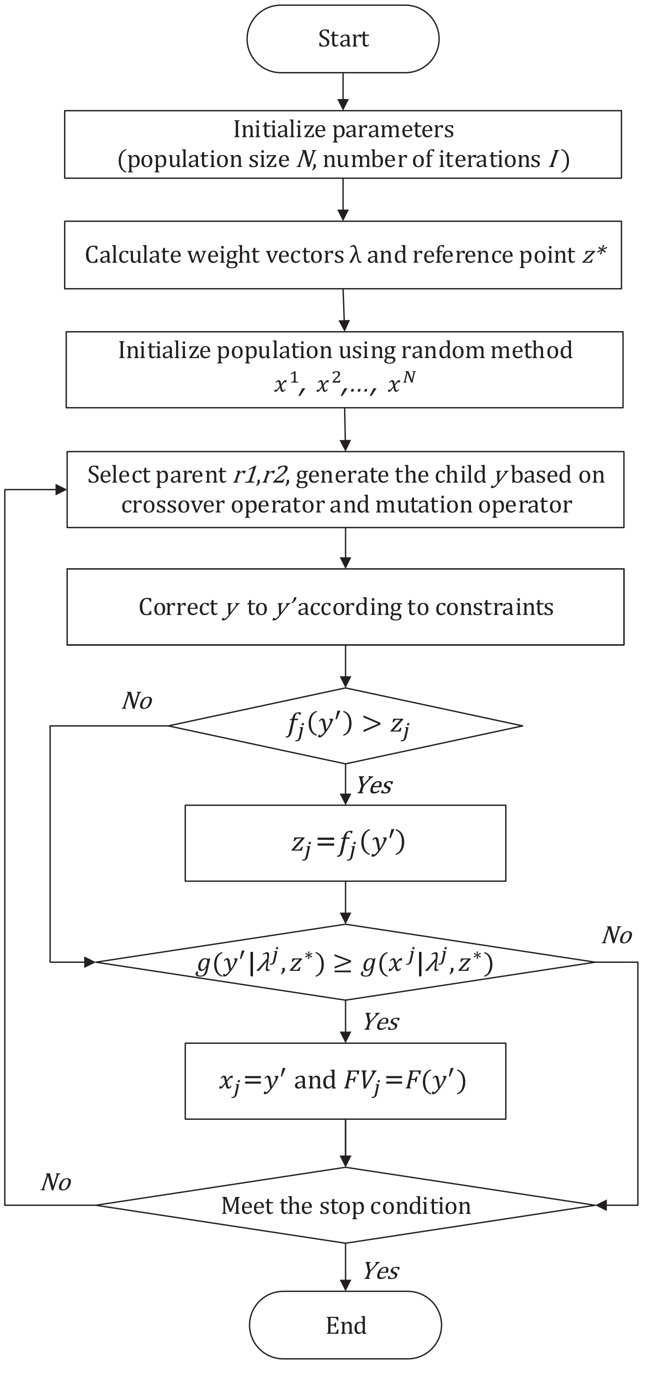

Start
Initialize parameters (population size N, number of iterations I)
Calculate weight vectors λ and reference point z*
Initialize population using random method x1, x2,…, xN
Select parent r1,r2, generate the child y based on crossover operator and mutation operator
Correct y to y' according to constraints
fj(y') > zj
No
Yes
zj=fj(y')
g(y'|λj,z*) ≥ g(xj|λj,z*)
No
Yes
xj=y' and FVj=F(y')
Meet the stop condition
No
Yes
End

FIGURE 5 Flowchart of LinkGame

probability p to produce a new y. BO uses the DE as a crossover operator.[48] Since the two parents r_1 and r_2 have some characteristics of the optimal solution, we cross the two parents and generate new individuals while retaining these characteristics.

Since the path-flow matching problem has some constraints, the CO judges whether the newly generated y is a feasible solution according to the constraint condition. If not, CO corrects y, so that y becomes a feasible solution.

The UO updates the necessary algorithm variables, such as the current optimal solution and neighbor state.

Parameters that affect the efficiency of the algorithm include population size N, mutation probability p, and iterations number I. In addition, the DE crossover operation has some parameters, but we think these parameters should use the default values, so we did not study the impact of these parameters on the performance of LinkGame.

Algorithm 1 LinkGame algorithm for path-flow matching

```
Input: number of flows $m$, number of paths $n$, preference ordering matrix $R$ and $O$, population size $N$, mutation probability $p$, number of iterations $I$.
Output: the optimal matching solution $Z_{opt}$.
  Step 1:Initialization.
  for n=1: N do
      $x^n = [x_{ij} = rand(0,1)]$;
      $FV^n = F(x^n)$;
  end for
  $z^* = (\sum_{i=1}^{m} \max\{o_{i1}, o_{i2}, \dots, o_{in}\}, \sum_{j=1}^{n} \max\{r_{1j}, r_{2j}, \dots, r_{nj}\})$;
  Step 2:Breeding.
  for i=1:N do
      $y_{tem} \leftarrow recombination(x^i, x^i - 1, x^i + 1)$;
      $y \leftarrow mutation(y_{tem})$;
      Step 3:correction.
      if $y \notin \Omega$ then
          $y \leftarrow repair(y)$;
      end if
  end for
  Step 4:Update.
  if $f_j(y) > f(x^n)$ then
      $z_j = f_j(y)$;
  end if
  if $g_i(y|\lambda^j, z^*) \geq g_i(x^j|\lambda^j, z^*)$ then
      $x_j = y$ and $FV^j = F(y)$;
  end if
  repeat step 2-4 until stop criteria are met;
```

5.3 | Analysis of LinkGame algorithm

The performance of LinkGame algorithm affects resource occupancy and efficiency, so we need to analyze the time complexity, space complexity, and performance of the LinkGame algorithm.

In terms of time complexity, LinkGame is *O(n)*. Therefore, as the scale of the problem increases, the execution efficiency of the LinkGame algorithm will not decrease much.

In terms of space complexity, the input of LinkGame is N matrices(population), and each matrix is *m*n*. LinkGame operates on these N matrices, so the input (population) of the algorithm is the most space-consuming part of the algorithm. In the execution process, the algorithm does not occupy a lot of space except for storing the optimal solution and a small amount of data temporarily occupied by the memory.

There are three main parameters that affect the performance of the algorithm, they are population size N, mutation probability p, and iterations number I. The more the population size N, the greater the ability of the algorithm to search the solution space, and the greater the probability of finding the optimal solution, but the larger the memory space. In our simulation process, because the machine performance is not very strong, we recommend population size $N = 500$. The effect of the number of iterations I is to force the algorithm to stop when the algorithm cannot satisfy the stop condition. Therefore, the number of iterations I is related to the running time of the algorithm. Since the controller has strict requirements on algorithm execution, such as scheduling every 10 seconds, we must set the number of iterations based on experience. In our experiment, the number of iterations I is 500, which can meet the requirements of scheduling every 10 seconds. The number of iterations I depends on the actual situation, such as scheduling period and machine performance. The mutation probability p reflects the ability of the algorithm to jump out of the current search space, typically 0.1.[49]

6 | DEPLOYMENT AND EVALUATION

In order to verify the performance of the proposed path-flow matching model and LinkGame algorithm in this paper, we deployed path-flow matching model in a simulated environment, and evaluated and compared LinkGame with ECMP, Hedera,[3] and Fincher.[20]

As shown in Figure 6, The deployment of the path-flow matching model requires four functional components, namely, the receiving module, path-flow ordering module, path-flow matching module, and flow rules generation module. The receiving module preprocesses the information collected by Floodlight. The path-flow ordering module orders the paths and flows according to link status and traffic changes. The path-flow matching module uses LinkGame algorithm to solve matching results. The flow rules generation module generates a traffic scheme based on the calculated optimal satisfaction matching result. Needless to say, four path-flow matching modules are based on the built-in modules of Floodlight. These modules are LinkDiscoveryManager module, FloodlightProvider module, TopologyService module, and DeviceManagerImpl module.

For traffic information collection, there are three methods of long flow detection. They are active monitoring, passive statistics, and sampling.[29] Considering performance, accuracy, and complexity, we chose OFPT_STATS_REQUEST protocol to collect traffic information, which is an active monitoring approach.

ECMP, Hedera, Fincher, and LinkGame are different in traffic scheduling. ECMP uses equal probability multipath forwarding for all flows. Hedera adopts ECMP for short flows and special scheduling for long flows. Fincher is similar to Hedera, but uses stable matching algorithm to schedule long flows. LinkGame distinguishes long and short flows according to the variation of flow and applies path-flow matching model according to the preference order. All three schemes except ECMP regenerate the traffic scheduling scheme every 10 seconds.[3] It should be specially stated that, in all experiments, the parameters of LinkGame are as follows: the number of iterations is 500, the population size is 500, and the mutation probability is 0.1. The reason for choosing these parameter settings is to meet the scheduling requirement of 10 seconds under the experimental hardware conditions.

Similar to previous research,[3,20] Mininet generates traffic using probabilistic model, random model, stag(p,q) model, and stride (i) model, respectively. All traffic models are implemented by using iperf in Mininet.

Under the probabilistic model, a server sends data (eg, initiates an UDP requests) to another server with a specified probability.

Under the random model, any server randomly sends data to other servers with equal probability.

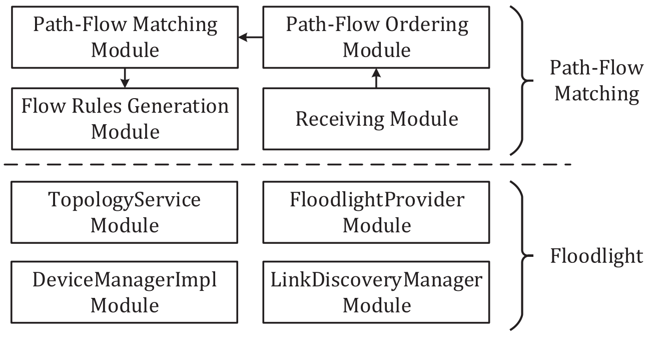

Path-Flow Matching Module
Path-Flow Ordering Module
Flow Rules Generation Module
Receiving Module
Path-Flow Matching
TopologyService Module
FloodlightProvider Module
DeviceManagerImpl Module
LinkDiscoveryManager Module
Floodlight

FIGURE 6 Path-flow matching architecture

Under the stag(p, q) model, a server sends data to another server under the same edge switch with the probability of p, and to another server at the same pod with the probability of q, and to other switches of the network with the probability of $(1 - p - q)$.

Under the stride(i) model, the mth server only sends data to the $(m + i)$th server.

The performance of entire network can be measured by two metrics: the FCT and throughput. The FCT is described as the interval between the start time stamp t_s when the first packet of a flow leaves the source server and the end time stamp t_e when the last packet of the flow arrives at the destination server.[50] The ideal FCT is the best achievable where only the service times of the intermediate nodes and the transmission time of the packets are considered.

We have evaluated and compared the performance of LinkGame, ECMP, Hedera, and Fincher on standardized average bisection bandwidth and FCT. The detail simulation parameters can be found in Table 1.

Figure 7 shows the overall standardized average bisection bandwidth performance of ECMP, Hedera, Fincher, and LinkGame. In most of traffic models, LinkGame has the best performance. Under the stag(1, 0) model, all three schemes except LinkGame have achieved almost the same results. The reason is that this traffic model does not generate traffic across core and aggregation switches. ECMP, Hedera, and Fincher use the ECMP method for traffic scheduling. In other words, Hedera and Fincher are equivalent to ECMP in stag(1, 0) model. LinkGame's traffic scheduling strategy is global and can achieve better bandwidth utilization in this model.

Long flow FCT is an important network performance indicator in data center network, which reflects the ability of the network to schedule long flow. In order to test the impact of different schemes on the long flow FCT, we set all long flow sizes in the network to 100 MB, and all short flows sizes to 50 KB. Figure 8 shows the long flow FCT of four schemes. LinkGame algorithm has achieved the best performance in all traffic models. In most scenarios, LinkGame can complete long flow transmission in about 22.3 seconds to 27.4 seconds. In the stag(1,0) model, LinkGame performance is degraded, but the long flow FCT is still shorter than the other three schemes.

Figure 9 demonstrates the improvement performance of standardized average bisection bandwidth of LinkGame. LinkGame improves bandwidth utilization by 8.3% to 36.9% compared to ECMP, 6.1% to 17.3% compared to Hedera, and 1.4% to 13% compared to Fincher.

TABLE 1 Simulation parameters

Parameters	Value
Number of core switches	4
Number of aggregation switches	8
Number of edge switches	8
Number of servers	16
Max packet size	1500 Bytes
Link bandwidth	10 Mbps
Long flow size	100 KB-2000 MB
Short flow size	1 KB-100 KB
Number of long flow: Number of short flow	1:9

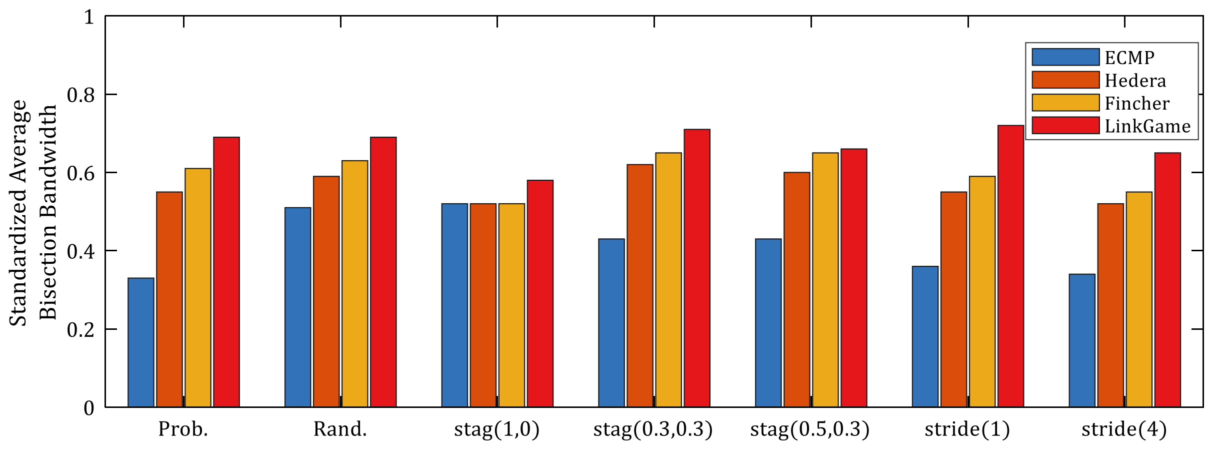


FIGURE 7 Performance of standardized average bisection bandwidth of different traffic scheduling schemes under different traffic models

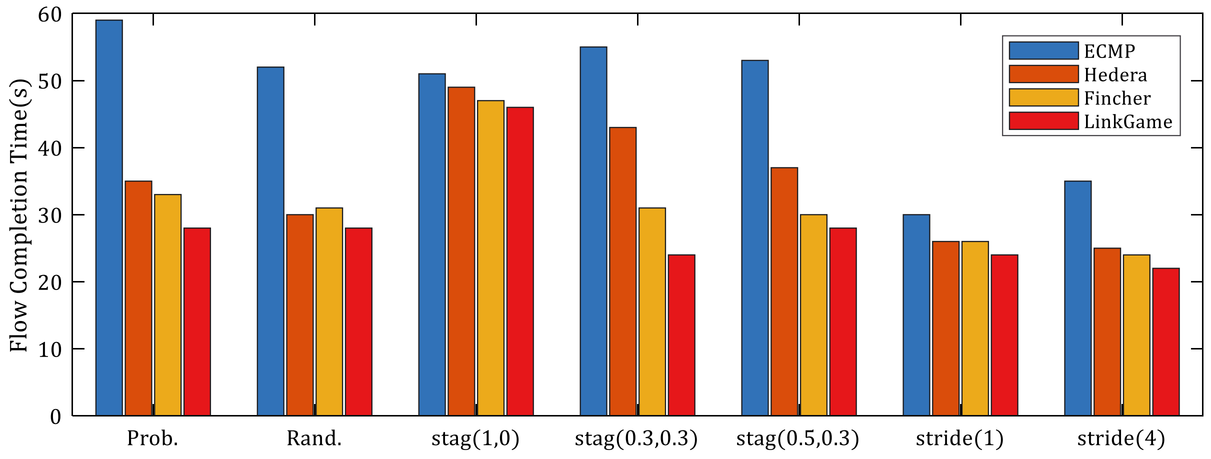


FIGURE 8 Performance of long flow FCT of different traffic scheduling schemes under different traffic models

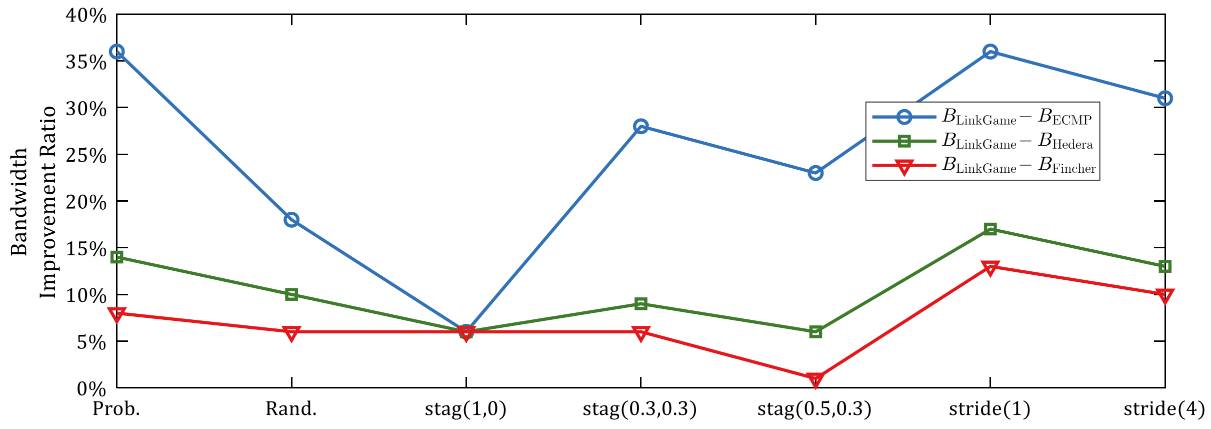


FIGURE 9 Improvement performance of standardized average bisection bandwidth of LinkGame compared to ECMP, Hedera and Fincher

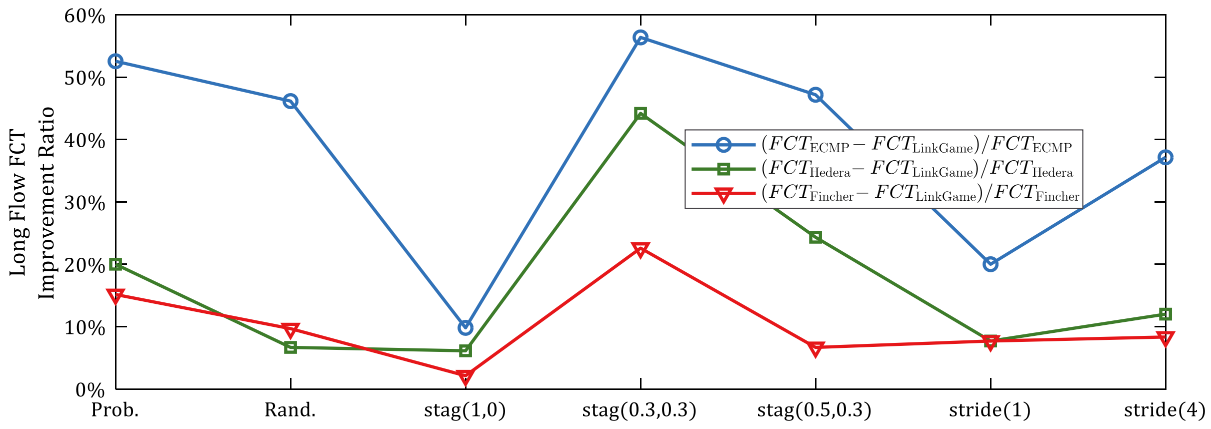


FIGURE 10 Improvement performance of long flow FCT of LinkGame compared to ECMP, Hedera, and Fincher

Figure 10 demonstrates the improvement performance of long flow FCT of LinkGame. LinkGame decreases FCT by 9.8%-56.3% compared to ECMP, 7.6%-44.1% compared to Hedera, and 2.1%-22.6% compared to Fincher.

In order to observe the bandwidth utilization in more detail, we measured the bandwidth utilization of some links between core layer switches and aggregation layer switches by adjusting Iperf rate in Mininet. Figure 11 shows the results of link utilization in different traffic modes, where the abscissa represents the link number between core switch and aggregation switch. For example, A1-C1 represents the link from the core switch A1 to the aggregation switch C1. LinkGame

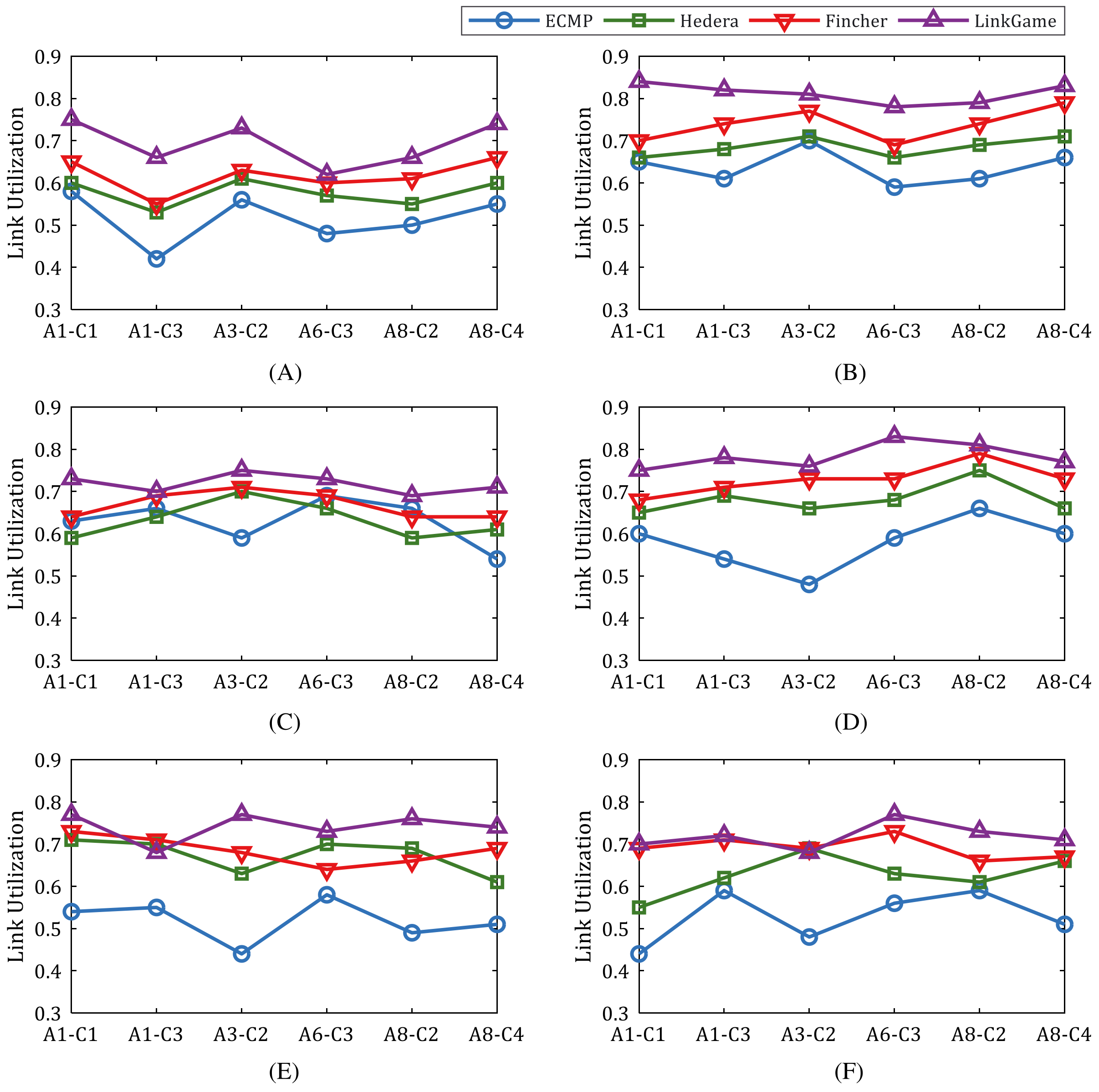


FIGURE 11 Link utilization of different traffic scheduling schemes under different traffic models. A, Prob.; B, Rand.; C, Stag(1,0); D, stag(0.3,0.3); E, stride(1); F, stride(4)

algorithm has better link utilization than ECMP, Hedera, and Fincher. ECMP selects paths in random, and multiple long flows collide on the same link to form a hotspot path. Therefore, the link utilization of ECMP is low. Hedera and Fincher also adopted the ECMP scheme for short flow, so the utilization rate is not much improved compared with ECMP. LinkGame adopts matching to process long flow and short flow and uses traffic change rate to indicate the importance of flow. Therefore, there is no hotspot path problem like ECMP in LinkGame scheme.

Figure 12 shows the packet loss rate results of four schemes. In order to be convincing, we set the link loss rate in all schemes to 1% and use iperf to record the network packet loss rate. Simulation results show that LinkGame has the lowest packet loss rate. ECMP lacks congestion awareness, which may increase network congestion and further lead to network packet loss. Moreover, ECMP cannot avoid packet loss due to link congestion and switch congestion. Both Hedera and Fincher schedule on the long flow, which can avoid the long-term occupation of network resources caused by long flow, and reduce FCT while reducing packet loss rate. Hedera has packet loss due to switch congestion. However, we use a single queue cache for the switch in Mininet to simulate a shared memory switch for Fincher. Therefore, Fincher has the lower packet loss rate than Hedera. It is worth noting both LinkGame and Fincher schemes perform well under the prob., rand., stage(0.3,0.3), and stride(4) models.

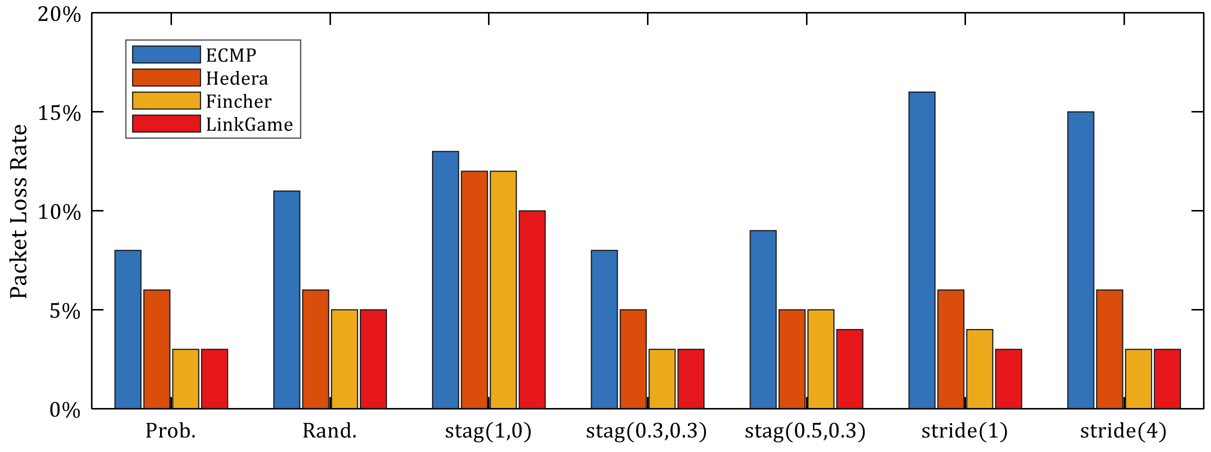


FIGURE 12 Performance of packet loss rate of LinkGame compared to ECMP, Hedera, and Fincher

7 | CONCLUSIONS

In this paper, we analyzed in detail the process of applying the two-sided matching model, considering the preference ordering to data center traffic scheduling. We first propose and model the path-flow matching problem considering the preference ordering by giving the formulation description of the path-flow matching problem, then design a preference-based path-flow ordering scheme Extended PIAS, and finally propose the algorithm LinkGame. LinkGame is an MOEA, which can simultaneously consider the stability and optimization of the matching results. Compared with the ECMP, Hedera, and Fincher, LinkGame performs extremely well in both bandwidth utilization and long flow FCT.

The key component of the path-flow matching model is preference ordering. In addition to the Extended PIAS mentioned in this paper, the various ordering methods summarized in Figure 3 of this paper deserve to be studied. Therefore, designing a more efficient ordering method is the research direction to improve the matching model effect, which is our future work.

ACKNOWLEDGMENTS

This work was supported by the Fundamental Research Funds for the Central Universities of China under Grant 2018YJS002, the CERNET Innovation Project under Grant NGII20180120, the National Key R&D Program of China (2018YFB1800305), and the Jingmen Science and Technology Research and Development Plan Project under Grant 2018YFYB055.

CONFLICT OF INTEREST

The authors declare no potential conflict of interests.

AUTHOR CONTRIBUTIONS

All authors have made substantial contributions to this manuscript. Lizhuang Tan proposed path-flow matching model and drafted the manuscript. Wei Su contributed significantly to analysis and manuscript preparation. Shuai Gao and Jingying Miao helped perform the analysis with constructive discussions and revised this manuscript. Yuan Cheng designed the LinkGame algorithm. Peng Cheng designed and performed the experiment and evaluation. In addition, we would like to thank Kun Yu and Man Li. They participated in the discussion and writing of this manuscript.

ORCID

Lizhuang Tan https://orcid.org/0000-0001-6826-4596

REFERENCES

1. Reno N. Hyperscale capex jumped 59% in the second quarter, maintaining record start to year. https://www.srgresearch.com/articles/hyperscale-capex-jumped-59-second-quarter. Accessed August 27, 2018.
2. Vahdat A, Al-Fares M, Farrington N, Mysore RN, Porter G, Radhakrishnan S. Scale-out networking in the data center. *IEEE Micro*. 2010;30(4):29-41.
3. Al-Fares M, Radhakrishnan S, Raghavan B, Huang N, Vahdat A. Hedera: dynamic flow scheduling for data center networks. Paper presented at: 7th USENIX Symposium on Networked Systems Design and Implementation; 2010; San Jose, CA.
4. Wang Y-C, Lin Y-D, Chang G-Y. SDN-based dynamic multipath forwarding for inter–data center networking. *Int J Commun Syst*. 2019;32(1):e3843.
5. Alizadeh M, Greenberg A, Maltz DA, et al. Data center TCP (DCTCP). *ACM SIGCOMM Comput Commun Rev*. 2011;41(4):63-74.
6. Zhang J, Ren F, Yue X, Shu R, Lin C. Sharing bandwidth by allocating switch buffer in data center networks. *IEEE J Sel Areas Commun*. 2014;32(1):39-51.
7. Roth AE, Sotomayor M. Two-sided matching. *Handb Game Theory Econ Appl*. 1992;1:485-541.
8. Benson T, Anand A, Akella A, Zhang M. MicroTE: fine grained traffic engineering for data centers. In: Proceedings of the 7th Conference on Emerging Networking Experiments and Technologies; 2011; Tokyo, Japan.
9. Perry J, Ousterhout A, Balakrishnan H, Shah D, Fugal H. Fastpass: a centralized “zero-queue” datacenter network. *ACM SIGCOMM Comput Commun Rev*. 2015;44(4):307-318.
10. Chen L, Lingys J, Chen K, Liu F. AuTO: scaling deep reinforcement learning for datacenter-scale automatic traffic optimization. In: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication; 2018; Budapest, Hungary.
11. Alizadeh M, Edsall T, Dharmapurikar S, et al. CONGA: distributed congestion-aware load balancing for datacenters. *ACM SIGCOMM Comput Commun Rev*. 2014;44(4):503-514.
12. Kabbani A, Vamanan B, Hasan J, Duchene F. Flowbender: flow-level adaptive routing for improved latency and throughput in datacenter networks. In: Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies; 2014; Sydney, Australia.
13. Zhang H, Zhang J, Bai W, Chen K, Chowdhury M. Resilient datacenter load balancing in the wild. In: Proceedings of the Conference of the ACM Special Interest Group on Data Communication; 2017; Los Angeles, CA.
14. Vanini E, Pan R, Alizadeh M, Taheri P, Edsall T. Let it flow: resilient asymmetric load balancing with flowlet switching. In: Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation; 2017; Boston, MA.
15. Jin R, Li J, Tuo X, Wang W, Li X. A congestion control method of sdn data center based on reinforcement learning. *Int J Commun Syst*. 2018;31(17):e3802.
16. Zhang H, Yang N, Long K, Pan M, Karagiannidis GK, Leung VC. Secure communications in NOMA system: subcarrier assignment and power allocation. *IEEE J Sel Areas Commun*. 2018;36(7):1441-1452.
17. Xu H, Li B. Anchor: a versatile and efficient framework for resource management in the cloud. *IEEE Trans Parallel Distrib Syst*. 2013;24(6):1066-1076.
18. Wang T, Liu F, Xu H. An efficient online algorithm for dynamic SDN controller assignment in data center networks. *IEEE/ACM Trans Netw*. 2017;25(5):2788-2801.
19. Hosseini S, Vakili R. Game theory approach for detecting vulnerable data centers in cloud computing network. *Int J Commun Syst*. 2019;32(8):e3938.
20. Zhang Y, Cui L, Zhang Y. A stable matching based elephant flow scheduling algorithm in data center networks. *Computer Networks*. 2017;120:186-197.
21. Abououf M, Singh S, Otrok H, Mizouni R, Ouali A. Gale-Shapley matching game selection—a framework for user satisfaction. *IEEE Access*. 2019;7:3694-3703.
22. Liu Wai-xi, Cai J, Wang Y, Chen QC, Tang D. Mix-flow scheduling using deep reinforcement learning for software-defined data-center networks. *Internet Technol Lett*. 2019;2(3):e99.
23. Zeng D, Yang G, Gu L, Guo S, Yao H. Joint optimization on switch activation and flow routing towards energy efficient software defined data center networks. Paper presented at: 2016 IEEE International Conference on Communications (ICC); 2016; Kuala Lumpur, Malaysia.
24. Bastam M, Sabaei M, Yousefpour R. A scalable traffic engineering technique in an SDN-based data center network. *Tran Emerg Telecommun Technol*. 2018;29(2):e3268.
25. Leiserson CE. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Trans Comput*. 1985;100(10):892-901.
26. Farrington N, Porter G, Radhakrishnan S, et al. Helios: a hybrid electrical/optical switch architecture for modular data centers. *ACM SIGCOMM Comput Commun Rev*. 2011;41(4):339-350.
27. Guo C, Lu G, Li D, et al. BCube: a high performance, server-centric network architecture for modular data centers. *ACM SIGCOMM Comput Commun Rev*. 2009;39(4):63-74.
28. Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. *ACM SIGCOMM Comput Commun Rev*. 2008;38(4):63-74.
29. Tan L, Su W, Gao S, Cheng P. L4S: low-speed software synergetic sampling and detecting long flow for data center network. Paper presented at: 2018 International Conference on Networking and Network Applications (NaNA); 2018; Xi'An, China.
30. Chiappori P-A, Oreffice S, Quintana-Domeque C. Fatter attraction: anthropometric and socioeconomic matching on the marriage market. *J Political Econ*. 2012;120(4):659-695.

31. Agarwal N. An empirical model of the medical match. *Am Econ Rev*. 2015;105(7):1939-1978.
32. Mindruta D, Moeen M, Agarwal R. A two-sided matching approach for partner selection and assessing complementarities in partners' attributes in inter-firm alliances. *Strateg Manag J*. 2016;37(1):206-231.
33. Fu H, Yang J, An Y. Made for each other: perfect matching in venture capital markets. *J Bank Finance*. 2019;100:346-358.
34. Gale D, Shapley LS. College admissions and the stability of marriage. *Am Math Mon*. 1962;69(1):9-15.
35. Roth AE. New physicians: a natural experiment in market organization. *Science*. 1990;250(4987):1524-1528.
36. Halldórsson M, Iwama K, Miyazaki S, Yanagisawa H. Randomized approximation of the stable marriage problem. *Theor Comput Sci*. 2004;325(3):439-465.
37. Hong C-Y, Caesar M, Godfrey P. Finishing flows quickly with preemptive scheduling. In: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication; 2012; Helsinki, Finland.
38. Alizadeh M, Yang S, Sharif M, et al. pfabric: minimal near-optimal datacenter transport. *ACM SIGCOMM Comput Commun Rev*. 2013;43(4):435-446.
39. Munir A, Baig G, Irteza SM, Qazi IA, Liu AX, Dogar FR. Friends, not foes: synthesizing existing transport strategies for data center networks. *ACM SIGCOMM Comput Commun Rev*. 2015;44(4):491-502.
40. Bai W, Chen L, Chen K, Han D, Tian C, Sun W. PIAS: practical information-agnostic flow scheduling for data center networks. In: Proceedings of the 13th ACM Workshop on Hot Topics in Networks; 2014; Los Angeles, CA.
41. Azevedo EM, Leshno JD. A supply and demand framework for two-sided matching markets. *J Political Econ*. 2016;124(5):1235-1268.
42. Lin Y, Wang Y-M, Chen S-Q. Hesitant fuzzy multiattribute matching decision making based on regret theory with uncertain weights. *Int J Fuzzy Syst*. 2017;19(4):955-966.
43. Echenique F, Galichon A. Ordinal and cardinal solution concepts for two-sided matching. *Games Econ Behav*. 2017;101:63-77.
44. Zhang Q, Li H. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans Evol Comput*. 2007;11(6):712-731.
45. Tan Y-Y, Jiao Y-C, Li H, Wang X-K. MOEA/D+ uniform design: A new version of MOEA/D for optimization problems with many objectives. *Comput Oper Res*. 2013;40(6):1648-1660.
46. Tan Y-Y, Jiao Y-C, Li H, Wang X-K. A modification to MOEA/D-DE for multiobjective optimization problems with complicated Pareto sets. *Information Sciences*. 2012;213:14-38.
47. Miettinen K. *Nonlinear Multiobjective Optimization*. New York: Springer Science & Business Media; 2012.
48. Price K, Storn RM, Lampinen JA. *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany: Springer; 2006.
49. Tan Y-Y, Jiao Y-C, Li H, Wang X-K. MOEA/D-SQA: a multi-objective memetic algorithm based on decomposition. *Engineering Optimization*. 2012;44(9):1095-1115.
50. Carpio F, Engelmann A, Jukan A. DiffFlow: differentiating short and long flows for load balancing in data center networks. Paper presented at: 2016 IEEE Global Communications Conference (GLOBECOM); 2016; Washington, DC.

How to cite this article: Tan L, Su W, Gao S, Miao J, Cheng Y, Cheng P. Path-flow matching: Two-sided matching and multiobjective evolutionary algorithm for traffic scheduling in cloud data center network. *Trans Emerging Tel Tech*. 2019;e3809. https://doi.org/10.1002/ett.3809