DeepRP: Bottleneck Theory Guided Relay Placement for 6G Mesh Backhaul Augmentation

Tianxin Wang and Xudong Wang, *Fellow, IEEE*

***Abstract*—Backhaul mesh networks are critical for ensuring coverage and connectivity of high-frequency 6G networks. To maintain high throughput, its architecture needs to be augmented by adding relays. However, how to place relays at appropriate sites poses two challenges: 1) there lacks a theory to capture the relationship between a certain change of network architecture and its throughput gain; 2) selecting the best sites for relays is a complicated combinatorial problem. To tackle the first challenge, this paper first establishes a clique-based bottleneck theory, through which a clique-based bottleneck structure of a given network architecture is constructed to determine the network throughput. Based on this bottleneck structure, clique gradients are then computed to quantify the impact of each clique on the overall network throughput. With the clique-based bottleneck theory, the second challenge is resolved by embedding clique gradients into a deep reinforcement learning (DRL) scheme. Specifically, the DRL actions are masked such that only the relay sites that match the highest clique gradients are selected. This DRL-based relay placement (DeepRP) scheme is evaluated via extensive simulations, and performance results show that it can boost network throughput by more than 50%, which is $10.4-32.1\%$ higher than those of baseline schemes.**

***Index Terms*—Backhaul mesh networks, relay placement, clique-based bottleneck theory, network throughput.**

I. Introduction

High-frequency networks that operate in millimeter-wave (mmWave) and Terahertz (THz) bands have a great potential to provide ultra-high data rates for future 6G applications [1]. However, high-frequency wireless links suffer from high path loss and severe blockage issues. To increase transmission distances and alleviate link blockage issues, wireless mesh networking is one of the most promising technologies for 6G network design with various deployment scenarios, such as backhaul mesh [2], sidelink mesh [3], and satellite mesh networks [4]. In 6G high-frequency wireless backhaul mesh networks, one of the most critical performance metrics is network throughput. Given all the source-destination pairs of end-to-end flows, the allocated rates for flows are continuously increased based on the max-min fairness [5] until the network reaches its saturation, such that no more rates can be offered. This network saturation throughput is the sum of all the flow saturation rates, which is considered as the network throughput in this paper.

It is challenging to achieve high network throughput in such a backhaul mesh. This is because network bottlenecks arise in various congestion regions of the network [6] that interact with each other. As shown in Fig. 1, two regions are overloaded with flow traffic such that they constitute the bottlenecks B_1 and B_2 for these flows. In this case, if one tries to resolve the bottleneck B_1, it is short-sighted to only focus on this single bottleneck without considering the interaction between two bottlenecks. If the formation of B_2 leads to B_1, there will be much more performance gain by resolving B_2 instead of B_1. Although some recent work [6] studies this interactions among bottlenecks in wired networks, there is no existing work studying the complicated relationship among bottlenecks in wireless networks.

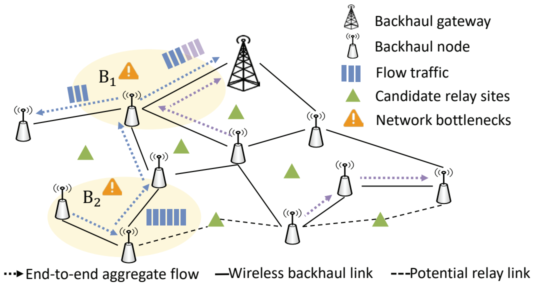


Fig. 1. A backhaul mesh network.

To boost network throughput of backhaul mesh, the interactions among bottlenecks in the wireless case need to be captured, such that bottlenecks can be appropriately alleviated. The conventional methods [7]–[11] for resolving bottlenecks in multi-hop wireless networks do not consider such interactions. In load-balancing approaches [7], [8], the flows are rerouted such that they can circumvent the heavily-loaded bottleneck links. In cross-layer scheduling approaches [9]–[11], joint source rate control and resource allocation is conducted to optimize resource utilization, where the flow source rates are reduced and radio resources are reallocated to avoid congestion. The above two types of work are aimed to enhance the utilization of network capacity under the given network architecture rather than to provide more network capacity.

In this paper, to fundamentally provide more network capacity, we propose to augment the 6G backhaul mesh network architecture with ***relays*** by considering the interactions among bottlenecks. The process of network architecture augmentation refers to ***relay placement*** in the rest of this paper, whose design principle perfectly aligns with the developing trends of 6G mesh networking, as there exist various candidate relays that support flexible networking, such as mobile base stations (BS), vehicle mounted relays, smart repeaters, and even intelligent reflective surfaces. The candidate location sites of relays are assumed to be known in this paper [12], [13], and a set of

The authors are with the University of Michigan–Shanghai Jiao Tong University (UM-SJTU) Joint Institute, Shanghai Jiao Tong University, Shanghai, China. Corresponding author: Xudong Wang; Email: wxudong@ieee.org.

relays are deployed at the selected candidate sites. In this way, the network throughput can be significantly boosted, and thus the full potential of 6G mesh networking can be unleashed.

To augment the network architecture for higher network throughput by adding relays, challenges lie in three aspects. First, the complicated interactions among network bottlenecks in 6G backhaul mesh must be captured, from which the relationship between the network architecture and the network throughput is determined. In wired networks, such interactions are captured via constructing a directed acyclic graph (DAG) named the bottleneck structure, and the quantitative theory of bottleneck structures (QTBS) is developed in [14], [15] to quantify the interactions among bottleneck links, and it is named the ***link-based bottleneck theory*** in the rest of the paper. However, this theory cannot be applied to the wireless case as it cannot capture various scenarios of link resource conflicts, such as inter-link interference, time sharing due to beam switching, and time division duplex (TDD) constraints [16]. How to develop the bottleneck theory for wireless networks remains an open problem. Second, the impact of each bottleneck on the network throughput needs to be accurately quantified. As the objective of relay placement is to boost network throughput with fewer relays, the bottlenecks with the highest impact on the overall network throughput must be identified, which enables the placement of relays to be specifically aimed at resolving these critical bottlenecks. Third, after identifying the network bottlenecks with a significant impact on network throughput, the relays need to be placed at the optimal sites that are selected from the candidate sites to resolve these bottlenecks. Since network throughput is derived based on bottleneck theory in this paper, the objective of throughput maximization cannot be expressed analytically as a differentiable function. Traditional optimization techniques cannot address this problem, and thus an efficient relay placement scheme is called for.

To tackle the above three challenges, the key designs are carried out as follows. First, to capture the interactions among bottlenecks and determine the relationship between network architecture and network throughput, the bottleneck theory named the ***clique-based bottleneck theory*** is developed for general wireless networks (with omni-directional or directional links). The theory applies the notion of clique from graph theory to capture link conflict among wireless links, and constructs a structure of bottlenecks in the unit of clique. With the clique-based bottleneck structure, the network throughput can be derived given a network architecture. Second, based on the constructed bottleneck structure, the clique gradient is derived to quantitatively determine how the perturbation of each bottleneck clique impacts the overall network throughput.

Third, the relay placement problem is decomposed into a multi-step, iterative process where one relay is placed at each step, until the number of relays reaches the given requirement. A deep reinforcement learning (DRL) based relay placement approach (***DeepRP***) is designed for this multi-step process of relay placement. DeepRP is guided by the clique-based bottleneck theory, and it is featured with three key aspects: 1) the graph convolutional network (GCN) is employed to embed the node features in the network topology; 2) the clique gradients are used in the action masking scheme to guide the agent to select the sites in proximity to the most critical bottlenecks; 3) the clique-based bottleneck structure is constructed after each decision step, based on which the reward is computed. As a result, DeepRP can achieve much higher throughputs than the baselines.

Contribution. The main contributions of this paper are summarized as follows:

- A clique-based bottleneck theory is developed for general wireless networks. In this theory, the clique-based bottleneck structures are constructed to identify the complicated interactions among bottlenecks and determine the relationship between the network architecture and the network throughput.
- A clique-gradient computation algorithm is designed based on the constructed bottleneck structure to quantify how the perturbation of each clique impacts the overall network throughput.
- A DRL-based relay placement approach is designed by incorporating the clique-based bottleneck theory, to augment the 6G backhaul architecture for higher network throughput. This approach significantly boosts network throughput as compared with the baselines.

The rest of the paper is organized as follows. In Section II, related work is presented, which is followed by the system model in Section III. The clique-based bottleneck theory is developed in Section IV, which is divided into two parts: the construction of clique-based bottleneck structures and the derivation of clique gradients. Based on the developed bottleneck theory, a DRL-based relay selection approach is presented in Section V. Performance evaluation is conducted in Section VI, and the paper is concluded in Section VII.

II. Related Work

The problem of designing network architectures of multi-hop wireless networks is closely related to two categories of work that attract much attention from researchers. The first category is topology management, which is focused on determining the topological connectivity and resource allocation among existing network nodes. The second category is relay placement, which puts emphasis on adding additional nodes to the existing network. In the following, these two categories of work are presented.

A. Topology Management for Multi-hop Networks

Considering the wireless backhaul networks, the paper [17] designs a multi-level hierarchical network architecture, and determines which level each node is at to form the hierarchical topology. It aims to enhance network connectivity. There also exists some work focusing on the high-frequency wireless backhaul networks, such as mmWave backhaul. In [18], a bayesian approach is designed to optimize the DAG structure of an integrated access and backhaul (IAB) network in mmWave bands, such that bursty traffic can be sustained with the highest probability. In [19], a DRL-based topology formation approach is designed for mmWave IAB networks to maximize the minimum link capacities on the paths from

IAB donor to IAB nodes. However, the above work does not consider the impact of inter-link interference, and their design objectives are not fundamentally providing more network capacity. In [20], a resource management framework is designed for the IAB network. Although the framework can optimize the end-to-end data rate, only one IAB node and one IAB donor is considered in [20], and thus only a single two-hop backhaul link is formed. Such a system set-up significantly simplifies the resource reuse scenario and directly eliminates the network bottleneck problem.

B. Relay Placement for Multi-hop Networks

Relay placement for wireless multi-hop backhaul networks is a critical problem that attracts much attention from researchers [21]–[23]. In [21], multiple relays are deployed between a pair of BSs in an mmWave backhaul network to overcome severe path loss and blockage issues. The placement of unmanned aerial vehicles (UAV) and reflective intelligent surfaces (RIS) for wireless backhauling are studied respectively in [22] and [23] to provide better alternative communication paths than the direct single-hop paths. However, the above work does not aim for boosting network-wide throughput. The optimal placement of multiple RISs in RIS-aided indoor THz communications are studied in [24]. This scheme aims to maximize the coverage area and only two-hop links (i.e., BS-RIS-user links) are considered. A practical heuristic relay placement scheme is designed for multi-hop wireless networks in [25]. It jointly optimizes relay node locations and traffic loads to minimize the overall number of packet retransmissions. By contrast, we aim to maximize the network saturation throughput. In [26], a multi-stage relay placement scheme is designed to maximize network connectivity, where the coarse relay sites are identified first and then the ideal sites are selected and refined. However, the scheme is more focused on inter-node reachability, and it does not consider the flow distribution and flow fairness regarding network throughput. The relay placement or relay selection algorithms designed in [27]–[29] aim to improve the throughput for relay-assisted cellular networks, but only two-hop transmission links between BS and users are taken into account.

Relay placement problem has also been studied in wireless sensor networks [30], [31], but their main objectives are to improve connectivity or reduce hop count for power-limited sensor nodes. Relay nodes are placed to maximize data throughput received at sink nodes in [32]. However, resource sharing and interference patterns are considered through the maximum number of flows that can circulate within a disk of a fixed radius centered at each node. This method cannot capture the interference pattern as accurately as the link conflict graph and cliques in our paper. Also, flow fairness is ignored in [32].

Compared to topology management based on the existing nodes, adding new nodes can provide much more opportunities for resolving network bottlenecks. Therefore, this paper is focused on the approach of relay placement to boost network throughput, and our novelty lies in the following aspects: 1) we consider a general multi-hop and multi-flow wireless backhaul network, where the flow fairness and flexible flow distribution are considered; 2) based on this set-up, we aim to maximize the fairness-based network saturation throughput that is not considered thoroughly before; 3) to achieve the throughput-maximization goal, the problem of relay placement is studied by considering the interactions among network bottlenecks for the first time.

III. System Model

A. Network Model

As shown in Fig. 1, a TDD-based backhaul mesh network with one mesh gateway and multiple deployed mesh nodes is considered. There also exist multiple known candidate relay sites. The relays to be deployed are dedicated relays as in [21], which do not serve as access points for users. The mesh gateway can gather network-wide information and determine the sites for relay placement in a centralized way. As the network is assumed to operate in high-frequency bands (e.g., THz bands), directional links are used among all the network nodes. Let $\mathcal{N}$, $\mathcal{L}$, $\mathcal{F}$ be the set of network nodes, directed links, and flows, and n, l, f be the indices that refer to the entities, respectively. The capacity of link l is denoted as C_l. The resources allocated to each flow is represented in data rates (bps) instead of the physical time and frequency resources, which will be elaborated in Section IV. Note that the feasibility on enforcing data rates into physical resources with a certain MAC mechanism is outside the scope of this paper. This issue is addressed in [33]–[35].

There are multiple end-to-end aggregate flows in the backhaul network, either between the mesh node and the mesh gateway, or between two mesh nodes. The single-path routing scheme is adopted, and $\mathcal{D} = \{\mathcal{R}_1, ..., \mathcal{R}_F\}$ is the set of routes with $\mathcal{R}_f$ as flow f's routes. $\mathcal{D}$ is also referred to as *flow distribution*. A stable flow distribution is assumed to be known at the beginning of relay placement [33]. For backhaul networks, the source and destination of each aggregate flow change on a large timescale, which can be obtained by network operators via traffic monitoring [36]. During one procedure of relay placement, the source-destination pairs stay unchanged [33], while the routing paths can be flexibly adjusted.

A three-sector analog beamforming architecture is adopted, and beam switching is applied for each sector (only one link can be activated at one time in the same sector). The effective gain including antenna gain and beamforming gain within the main-lobe beam is assumed to be constant as in [2]. Let G_t and G_r be the transmit effective gain and receive effective gain, and perfect beam alignment is assumed. To model multi-path channel components, the Saleh-Valenzuela (S-V) model adopted [37], [38]. We only consider one ray within each cluster [38], as the finer details of each cluster's internal multi-path components are not critical in our problem. Therefore, the channel impulse response of the S-V model is expressed as

$$h(t) = \sqrt{L\left(f_c, d\right)} X \sum_{m=0}^{M} \beta_m e^{j\theta_m} \delta\left(t - T_m\right), \qquad (1)$$

where $L\left(f_c, d\right)$ is the path loss coefficient that will be elaborated later; X is a log-normal random variable with the standard deviation as σ_X for the shadowing effect; M is the

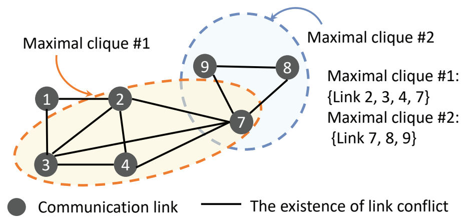

Maximal clique #1
Maximal clique #2
Maximal clique #1: {Link 2, 3, 4, 7}
Maximal clique #2: {Link 7, 8, 9}
1
2
3
4
7
8
9
● Communication link
The existence of link conflict

Fig. 2. A link conflict graph and maximal cliques.

number of non-negligible clusters within time duration T_s; θ_m is a random variable uniformly distributed over $[0, 2\pi)$; T_m is the arrival time of the m-th cluster modeled by a Poisson process with an average arrival rate of λ; β_m follows the Rayleigh distribution $f(\beta_m) = 2(\beta_m/\overline{\beta_m^2})e^{-\beta_m/\overline{\beta_m^2}}$ with $\overline{\beta_m^2}$ is the average power in the m -th cluster. The average power is given as $\overline{\beta_m^2} = \overline{\beta_0^2}e^{-T_m/\gamma}$ with γ as the cluster decay constant and $\overline{\beta_0^2} = 1$.

The path loss coefficient $L(f_c, d)$ includes the free-space spreading loss $L_s(f_c, d)$ and the molecular absorption loss $L_{abs}(f_c, d)$ in THz bands [1] [37], expressed as

$$\begin{aligned} L(f_c, d) &= L_s(f, d) L_{abs}(f, d) \\ &= G_r G_t \left(\frac{c}{4\pi f_c d_0}\right)^2 \left(\frac{d_0}{d}\right)^n e^{-k_{abs}(f_c)d} \end{aligned} \tag{2}$$

where c is the speed of light; f_c is the carrier frequency; d is the propagation distance; d_0 is the reference distance set as 1; $k_{abs}(f_c)$ is frequency-dependent absorption coefficient; n is the path loss exponent.

B. Link Conflict Graph and Maximal Clique

The undirected link conflict graph [39] $\mathcal{G}_c = \langle \mathcal{V}_c, \mathcal{E}_c \rangle$ captures resource conflicts among links, where the vertex set $\mathcal{V}_c$ consists of all the transmission links, and the edge set $\mathcal{E}_c$ describes the conflicting relationship of wireless resources between two transmission links. Note that the transmission link from node A to B is considered as a different link than the transmission link from node B to A. Let $e_{ij} \in \mathcal{E}_c$ be the edge between vertex i and vertex j. The edge exists if 1) there is cross-link interference between two links (i.e., the receiver of one link is within the beam interference range of another link), or 2) the two links cannot be active simultaneously due to the TDD constraint, or 3) beam switching is applied between the two links in the same sector. The two links cannot use the same resources if there exists an edge between them.

Based on the link conflict graph, the notion of ***clique*** is utilized as a critical tool for capturing the interference and resource sharing pattern. Each clique is a complete subgraph of the link conflict graph, and a ***maximal clique*** is the clique that is not contained in any other cliques. Via the existing algorithm [40] of solving maximal clique cover of a graph, a set of maximal cliques $\mathcal{K}$ is obtained over the link conflict graph $\mathcal{G}_c$. Only one link in a maximal clique can be activated in one resource unit.

IV. Clique-based Bottleneck Theory

The clique-based bottleneck theory is established. In this theory, the clique-based bottleneck structure is constructed to determine the relationship between the network architecture and the network throughput. Based on the constructed structure, the clique gradients are computed to quantify the impact of each bottleneck clique on the throughput.

A. Clique-based Bottleneck Structure

The bottleneck structures are constructed in the unit of maximal cliques to capture the link resource conflicts in wireless networks. To achieve this, the definition of a bottleneck clique is given, and then the algorithm of constructing the structure is developed based on clique fair shares.

1) Definition of Bottleneck Clique: The resource share of flow rate r_f on link l is defined as r_f/C_l. Since any two links within a maximal clique cannot be activated in the same resource unit, the sum of resource shares within each clique cannot exceed one. Therefore, the resource sharing pattern within each clique must satisfy the following feasibility constraint [41]: $\sum_{l\in\mathcal{L}_k}\sum_{f\in\mathcal{F}_l} r_f/C_l \leq 1, \forall k = 1, 2, ..., K$, where $\mathcal{L}_k$ is the set of links in clique k, and $\mathcal{F}_l$ is the set of flows traversing link $l \in \mathcal{L}_k$. When the above feasibility constraint is satisfied, how to further enforce the flow rates into physical resources by MAC mechanisms is another critical problem that has been addressed in [33], [41]. While the bottleneck clique can be defined under various fairness criteria [14], this paper is focused on the widely-used max-min fairness among end-to-end flow rates. The definition of bottleneck clique under max-min fairness extends the concept in [34], [41] for multi-hop flows, which is stated as follows.

Definition 1 (Bottleneck Clique**).** Clique k is a bottleneck clique for flow f if and only if the following conditions are satisfied: a) clique k must be a saturated clique that satisfies the feasibility constraint and has no remaining resource share for any flow, expressed as $\sum_{l\in\mathcal{L}_k}\sum_{f\in\mathcal{F}_l}\frac{r_f}{C_l} = 1$; b) the flow f's resource share under max-min fairness cannot be smaller than the fair shares of other flows on each link in clique k, expressed as $\frac{r_f}{C_l} \geq \frac{r_{f'}}{C_l}, \forall l \in \mathcal{L}_k, f' \in \mathcal{F}_l \setminus \{f\}$.

The above definition indicates that if the flow can only obtain a resource share smaller than the fair share obtained by other flows in clique k, this flow must be bottlenecked elsewhere other than clique k.

2) Algorithm of Constructing Clique-based Bottleneck Structures: From the above definition, multiple bottleneck cliques can be identified if the max-min fair flow rates $\{r_f | f \in \mathcal{F}\}$ are determined. However, there exist two questions: a) how to determine the max-min flow rate allocation for a backhaul mesh network based on maximal cliques; b) how the identified bottleneck cliques exert influence on each other. These two questions are merged and resolved by the construction of clique-based bottleneck structures. With the constructed bottleneck structure, the interactions among bottlenecks are captured, and the relationship between the network architecture and the network throughput is determined.

Before developing the algorithm for constructing a clique-based bottleneck structure, a critical metric named ***clique fair share*** s_k is introduced for clique k to specify the fair rate that each unconverged flow can be allocated in this clique. A flow

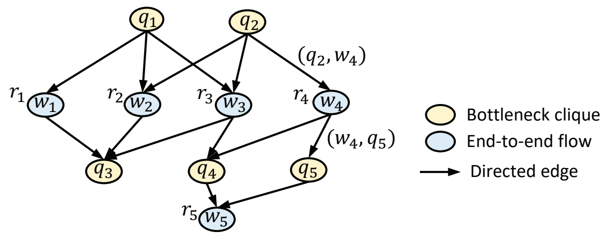


Fig. 3. A clique-based bottleneck structure.

is considered unconverged if it has not been allocated with the data rate of its fair share. Let $\mathcal{F}_l^{\mathrm{c}}$ and $\mathcal{F}_l^{\mathrm{u}}$ denote the set of converged flows on link l and the set of unconverged flows on link l respectively, and $\mathcal{F}_l^{\mathrm{u}} = \mathcal{F}_l \setminus \mathcal{F}_l^{\mathrm{c}}$. The definition of clique fair share is stated as follows.

Definition 2 (Clique Fair Share)**.** The clique fair share for clique k is obtained by equally allocating the remaining resource share among all the unconverged flows in this clique, expressed as

$$s_k = \frac{1 - \sum_{l \in \mathcal{L}_k} \sum_{f \in \mathcal{F}_l^{\mathrm{c}}} (r_f / C_l)}{\sum_{l \in \mathcal{L}_k} (|\mathcal{F}_l^{\mathrm{u}}| / C_l)}, \tag{3}$$

where the remaining resource share is obtained by deducting the converged flows' shares from 1 as shown in the nominator.

Based on the definition of clique fair share, Algorithm 1 is developed to construct a clique-based bottleneck structure, i.e., a DAG $\mathcal{G}_{\mathrm{b}} = \langle \mathcal{V}_{\mathrm{b}}, \mathcal{E}_{\mathrm{b}} \rangle$. As shown in Fig. 3, the vertex set $\mathcal{V}_{\mathrm{b}}$ consists of all the bottleneck cliques $\hat{\mathcal{Q}}$ and flows $\mathcal{F}$ in the network (i.e., $\mathcal{V}_{\mathrm{b}} = \hat{\mathcal{Q}} \cup \mathcal{F}$), while the edge set $\mathcal{E}_{\mathrm{b}}$ comprises of the directed edges between bottleneck cliques and flows. Let $(\mathrm{q}_k, \mathrm{w}_f)$ and $(\mathrm{w}_f, \mathrm{q}_k)$ denote the directed edge from clique k to flow f, and the directed edge from flow f to clique k, respectively, where $k \in \hat{\mathcal{Q}}, f \in \mathcal{F}$. The main procedure of Algorithm 1 is described as follows. In each iteration, the cliques that are not bottlenecks with the minimum clique fair share value are chosen (Line 4-5), e.g., q_1 and q_2 are chosen and added into the bottleneck structure in Fig. 3. Next, the flows traversing the chosen cliques are allocated with the clique fair share as their data rates, and they are added into $\mathcal{G}_{\mathrm{b}}$ with the directed edges from the cliques (Line 8-10). For example, w_4 traverses w_2, so the edge $(\mathrm{q}_2, \mathrm{w}_4)$ is added. Furthermore, the edges are added from these flows to other bottleneck cliques they traverse (Line 11), such as $(\mathrm{w}_4, \mathrm{q}_5)$. After the edge addition, the clique fair share values are recomputed by excluding the converged flows (Line 4). The iteration continues until all the flows are converged, and thus $\mathcal{G}_{\mathrm{b}}$ is established.

The time complexity of Algorithm 1 is analyzed as follows. Only Line 5 runs in non-constant time while other lines of statements run in constant time. In Line 5, the sorting based on Min-Heap is adopted with the heap size set to K at most, where K is the number of cliques. Each heap update runs in $O(\log K)$. The number of such updates is bounded by $K\eta$, where η is the maximum number of flows traversing a clique. Therefore, the algorithm has the complexity of $O(\eta K \log K)$.

The clique-based bottleneck structure is a non-trivial extension of the original link-based bottleneck structure in [14] Similar to the original structure, the clique-based structure captures the interactions among bottlenecks and flows via directed edges. However, there exists one distinct feature in the clique-based case: the bottleneck cliques can be overlapped by including some common links. Therefore, the theorem in [14] saying that the root vertices in the bottleneck structure have a higher impact on the overall network than the leaf vertices cannot be applied to the clique-based case. To accurately quantify the impact of each bottleneck cliques on the overall network, clique gradients need to be derived, as shown in the following subsection.

Algorithm 1 Bottleneck structure construction

```
1:  Input: The set of maximal cliques $\mathcal{K}$, the set of flows $\mathcal{F}$;
2:  Initialization: $\mathcal{V}_{\mathrm{b}} = \mathcal{E}_{\mathrm{b}} = \mathcal{F}^{\mathrm{C}} = \emptyset$, $r_f = 0, \forall f \in \mathcal{F}$, $\hat{\mathcal{F}}_k$
    (the flow set traversing clique $k$), $\mathcal{Q}_f$(the set of cliques
    that are traversed by flow $f$), $\mathcal{F}^{\mathrm{C}}$(the converged flow set).
3:  repeat
4:      Compute clique fair shares $s_k, \forall k \notin \mathcal{V}_{\mathrm{b}}$;
5:      $m_k = \min\left\{ s_j | \hat{\mathcal{F}}_j \cap \hat{\mathcal{F}}_k \neq \{\emptyset\}, \forall j \notin \mathcal{V}_{\mathrm{b}} \right\}, \forall k \notin \mathcal{V}_{\mathrm{b}}$;
6:      for $k$ with $s_k = m_k$ do
7:          for $f \in \hat{\mathcal{F}}_k, f \notin \mathcal{F}^{\mathrm{C}}$ do
8:              $r_f = s_k$;
9:              $\mathcal{F}^{\mathrm{C}} = \mathcal{F}^{\mathrm{C}} \cup \{f\}$;
10:             $\mathcal{E}_{\mathrm{b}} = \mathcal{E}_{\mathrm{b}} \cup \{(\mathrm{q}_k, \mathrm{w}_f)\}$;
11:             $\mathcal{E}_{\mathrm{b}} = \mathcal{E}_{\mathrm{b}} \cup \{(\mathrm{w}_f, \mathrm{q}_j)\}, \forall j \in \mathcal{Q}_f \setminus \{k\}$;
12:         end for
13:         $\hat{\mathcal{Q}} = \hat{\mathcal{Q}} \cup \{k\}$;
14:         $\mathcal{V}_{\mathrm{b}} = \hat{\mathcal{Q}} \cup \mathcal{F}^{\mathrm{C}}$;
15:     end for
16: until $\mathcal{F}^{\mathrm{C}} = \mathcal{F}$
17: Obtain the network throughput $T = \sum_{f \in \mathcal{F}} r_f$;
18: Output: $\mathcal{G}_{\mathrm{b}} = \langle \mathcal{V}_{\mathrm{b}}, \mathcal{E}_{\mathrm{b}} \rangle, T$.
```

B. Clique Gradient

1) Definition of Clique Equivalent Capacity: To quantitatively characterize the capability of a clique, the clique equivalent capacity is defined via a set of virtual scheduling variables.

Definition 3 (Clique Equivalent Capacity)**.** The clique equivalent capacity R_k is defined as the weighted sum of all the link capacities in clique k, $R_k = \sum_{l \in \mathcal{L}_k} p_{k,l} C_l, \forall k \in \mathcal{K}$, where $\{p_{k,l} | l \in \mathcal{L}_k\}$ are the virtual scheduling variables that satisfy two constraints: $\sum_{l \in \mathcal{L}_k} p_{k,l} \leq 1, \forall k \in \mathcal{K}$; $p_{k,l} \geq 0, \forall l \in \mathcal{L}_k, k \in \mathcal{K}$.

With the virtual scheduling variables $\{p_{k,l} | l \in \mathcal{L}_k\}$, the capacity of a clique is characterized based on its link capacities and flow distribution. Analogous to how a link capacity is allocated among flows, the clique equivalent capacity is allocated among its traversing flows, from which the equivalent set of max-min flow rates as in Algorithm 1 can be obtained, as stated in the following theorem.

Theorem 1. *There exists a unique solution of $\{p_{k,l} | l \in \mathcal{L}_k\}$ for each clique in the bottleneck structure that achieves the same flow rate results as those obtained by Algorithm 1.*

Proof. First, consider one clique k on the first (root) level of bottleneck structure that has no predecessor flow. The flow rates in this clique are the same and can be computed by Eq. (3) in Algorithm 1, which is

$$s_k = r_f = \frac{1}{\sum_{l\in\mathcal{L}_k}\frac{b_l}{C_l}}, \forall f\in\mathcal{F}_l, \tag{4}$$

where $b_l = |\mathcal{F}_l^{\mathrm{u}}| = |\mathcal{F}_l|$. By using its equivalent capacity, these flow rates can also be computed by allocating the clique equivalent capacity among all the unconverged flows, which is

$$r_f = \frac{R_k}{\sum_{l\in\mathcal{L}_k} b_l} = \frac{\sum_{l\in\mathcal{L}_k} p_{k,l}C_l}{\sum_{l\in\mathcal{L}_k} b_l}, \tag{5}$$

where the clique equivalent throughput R_k is referred to Definition 3. Based on the max-min rate allocation in Algorithm 1, the unconverged flows are allocated with equal resource shares, that is,

$$\frac{p_{k,1}}{b_1} = \frac{p_{k,2}}{b_2} = ... = \frac{p_{k,l}}{b_l} = \beta, \tag{6}$$

where b_l is an positive integer and β is the positive ratio of resource share. The case of $b_l = 0$ is naturally excluded here, since $p_{k,l} = 0$ if $b_l = 0$. Considering Eq. (4), (5), and (6), the virtual scheduling variables can be derived as

$$p_{k,l} = \frac{b_l}{\sum_{l\in\mathcal{L}_k}\frac{b_l}{C_l}}\frac{\sum_{l\in\mathcal{L}_k} b_l}{\sum_{l\in\mathcal{L}_k} b_lC_l}, \forall l\in\mathcal{L}_k. \tag{7}$$

Specially, if all the links in clique k have the same capacity, then Eq. (7) is simplified as $p_{k,l} = \frac{b_l}{\sum_{l\in\mathcal{L}_k} b_l}$, and the clique equivalent capacity in this case is a convex combination of link capacities.

In the following, we prove that the uniquely derived $p_{k,l}$ satisfies the constraints in Definition 3. Obviously, $p_{k,l}$ is non-negative. Based on Eq. (6) and (7), the problem of proving $\sum_{l\in\mathcal{L}_k} p_{k,l} = \beta\sum_{l\in\mathcal{L}_k} b_l \le 1$ is transformed into proving the following inequality:

$$\frac{1}{\sum_{l\in\mathcal{L}_k}\frac{b_l}{C_l}}\frac{\sum_{l\in\mathcal{L}_k} b_l}{\sum_{l\in\mathcal{L}_k} b_lC_l} \le \frac{1}{\sum_{l\in\mathcal{L}_k} b_l}. \tag{8}$$

The above inequality is reformed as

$$\left(\sum_{l\in\mathcal{L}_k} b_l\right)^2 \le \sum_{l\in\mathcal{L}_k}\frac{b_l}{C_l}\sum_{l\in\mathcal{L}_k} b_lC_l, \tag{9}$$

and further as

$$\sum_{i\in\mathcal{L}_k}\sum_{j\in\mathcal{L}_k, j\ne i} 2b_ib_j \le \sum_{i\in\mathcal{L}_k}\sum_{j\in\mathcal{L}_k, j\ne i}(\frac{C_i}{C_j}+\frac{C_j}{C_i})b_ib_j. \tag{10}$$

Since we have $C_i/C_j + C_j/C_i \ge 2, \forall C_i, C_j \ge 0$, the inequality (10) always holds true and the equality is obtained with $C_i = C_j, \forall i,j\in\mathcal{L}_k$. Therefore, the uniquely derived virtual scheduling variables $\{p_{k,l}, \forall l\in\mathcal{L}_k\}$ satisfy the constraints in Definition 3. With this, the proof for the cliques on the root level is completed.

Next, consider one clique k on the remaining levels of the bottleneck structure that has a set of predecessor flows with the same converged rates r. Let $a_l = |\mathcal{F}_l^{\mathrm{c}}|$, $d_l = b_l - a_l = |\mathcal{F}_l^{\mathrm{u}}|$, and the flow rates for the unconverged flows is derived by in Algorithm 1 as

$$s_k = \frac{1-\sum_{l\in\mathcal{L}_k}\frac{a_lr}{C_l}}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}}. \tag{11}$$

Similarly, they can also be computed by allocating the remaining clique equivalent capacity among the unconverged flows, which is

$$s_k = \frac{R_k - \sum_{l\in\mathcal{L}_k} p_{k,l}a_lr}{\sum_{l\in\mathcal{L}_k} d_l} = \frac{\sum_{l\in\mathcal{L}_k} p_{k,l}(C_l - a_lr)}{\sum_{l\in\mathcal{L}_k} d_l}. \tag{12}$$

Based on the max-min rate allocation in Algorithm 1, the unconverged flows are allocated with equal resource share, which follows

$$\frac{p_{k,1}}{d_1} = \frac{p_{k,2}}{d_2} = ... = \frac{p_{k,l}}{d_l} = \theta, \tag{13}$$

where d_l and θ are positive numbers. Considering Eq. (11), (12), and (13), the virtual scheduling variables can be derived as

$$p_{k,l} = \frac{d_l\left(1-\sum_{l\in\mathcal{L}_k}\frac{a_lr}{C_l}\right)}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}}\frac{\sum_{l\in\mathcal{L}_k} d_l}{\sum_{l\in\mathcal{L}_k} d_l(C_l - a_lr)} \tag{14}$$

Similar to the derivation for the first case, we prove that the uniquely derived $p_{k,l}$ satisfies the constraints in Definition 3. Obviously, $p_{k,l}$ is non-negative. The problem of proving $\sum_{l\in\mathcal{L}_k} p_{k,l} = \theta\sum_{l\in\mathcal{L}_k} d_l \le 1$ is transformed into proving the following inequality:

$$\begin{aligned}\theta &= \frac{1-\sum_{l\in\mathcal{L}_k}\frac{a_lr}{C_l}}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}}\frac{\sum_{l\in\mathcal{L}_k} d_l}{\sum_{l\in\mathcal{L}_k} d_l(C_l - a_lr)} \\ &\le \frac{1}{\sum_{l\in\mathcal{L}_k} d_l}.\end{aligned} \tag{15}$$

To prove the above inequality, the following transformation is conducted:

$$\begin{aligned}\theta &= \frac{\sum_{l\in\mathcal{L}_k} d_l[1 - r\sum_{l\in\mathcal{L}_k}\frac{a_l}{C_l}]}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_lC_l - r\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_la_l} \\ &= \frac{\sum_{l\in\mathcal{L}_k} d_l[1 - r\sum_{l\in\mathcal{L}_k}\frac{a_l}{C_l}]}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_lC_l[1 - \frac{r\sum_{l\in\mathcal{L}_k} d_la_l}{\sum_{l\in\mathcal{L}_k} d_lC_l}]} \\ &\le \frac{\sum_{l\in\mathcal{L}_k} d_l}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_lC_l} \\ &\le \frac{1}{\sum_{l\in\mathcal{L}_k} d_l}.\end{aligned} \tag{16}$$

With this, we prove that $\{p_{k,l}, \forall l\in\mathcal{L}_k\}$ satisfies the constraints in Definition 3.

Finally, consider one clique k that has a set of predecessor flows with various converged rates $\hat{r}_m, m = 1,2,...,M$ in the bottleneck structure. Let $a_{l,m} = \mathcal{F}_{l,m}^{\mathrm{c}}, d_l = |\mathcal{F}_l| - \sum_{m=1}^{M}|\mathcal{F}_{l,m}^{\mathrm{c}}|$. Following the similar methodology of combining two ways of flow rate computation, the virtual scheduling variables can be derived as

$$p_{k,l} = \frac{d_l\sum_{l\in\mathcal{L}_k} d_l[1-\sum_{m=1}^{M}\hat{r}_m\sum_{l\in\mathcal{L}_k}\frac{a_{l,m}}{C_l}]}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_lC_l[1-\sum_{m=1}^{M}\frac{\hat{r}_m\sum_{l\in\mathcal{L}_k} d_la_{l,m}}{\sum_{l\in\mathcal{L}_k} d_lC_l}]}, \tag{17}$$

and

$$\frac{p_{k,1}}{d_1} = \frac{p_{k,2}}{d_2} = ... = \frac{p_{k,l}}{d_l} = \theta, \tag{18}$$

where d_l and θ are positive. Similar to the previous case, to prove $\sum_{l\in\mathcal{L}_k} p_{k,l} = \theta \sum_{l\in\mathcal{L}_k} d_l \le 1$, the key inequality below is proved as follows:

$$\begin{aligned}\theta &= \frac{\sum_{l\in\mathcal{L}_k} d_l[1-\sum_{m=1}^{M}\hat{r}_m \sum_{l\in\mathcal{L}_k}\frac{a_{l,m}}{C_l}]}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_l C_l[1-\sum_{m=1}^{M}\frac{\hat{r}_m\sum_{l\in\mathcal{L}_k} d_l a_{l,m}}{\sum_{l\in\mathcal{L}_k} d_l C_l}]}\\ &\le \frac{\sum_{l\in\mathcal{L}_k} d_l}{\sum_{l\in\mathcal{L}_k}\frac{d_l}{C_l}\sum_{l\in\mathcal{L}_k} d_l C_l}\\ &\le \frac{1}{\sum_{l\in\mathcal{L}_k} d_l}.\end{aligned} \tag{19}$$

Therefore, the uniquely derived virtual scheduling variables satisfy the constraints in Definition 3, and thus Theorem 1 is proved to be true. □

2) Derivation of Clique Gradients: The clique gradient $\mathbf{g}(j)$ is defined as the impact of an infinitely small perturbation in clique j's equivalent capacity on the overall network throughput T, expressed by

$$\mathbf{g}(j) = \frac{\partial T}{\partial R_j} = \frac{\partial \sum_{f\in\mathcal{F}} r_f}{\partial R_j} = \sum_{f\in\mathcal{F}} \frac{\partial r_f}{\partial R_j}. \tag{20}$$

From the above expression, $\mathbf{g}(j)$ can be also viewed as the sum of the clique j's impacts on all the flow rates. The key question is how to efficiently compute $\mathbf{g}(j)$ for each bottleneck clique j based on the constructed bottleneck structure $\mathcal{G}_\mathrm{b}$. Assume that an infinitely small perturbation is imposed on clique j (viewed as the perturbation source), i.e., ΔR_j is exerted on R_j. For a bottleneck structure $\mathcal{G}_\mathrm{b}$, let Δs_k be the perturbation on the clique fair share of bottleneck clique k, and Δr_f be the perturbation on the rate of flow f, both of which are caused by the perturbation source ΔR_j. For each directed edge in $\mathcal{G}_\mathrm{b}$, the former(latter) vertex is called the predecessor(successor) of the latter(former) one. Note that Δs_k and Δr_f are normalized with ΔR_j to remove the physical dimension, and thus ΔR_j is not included in the propagation. To figure out how this perturbation ΔR_j propagates through the whole bottleneck structure and results in Δs_k and Δr_f, the propagation equations must be derived.

First, the propagation equation for Δs_k is determined as follows. The variation of the remaining resource in clique k consists of two parts: a) the variation of capacity caused by the perturbation source, expressed as the ratio of common virtual scheduling variables in two cliques k and j, i.e., $\left(\sum_{l\in\mathcal{L}_k\cap\mathcal{L}_j} p_{k,l}\right) / \left(\sum_{l\in\mathcal{L}_j} p_{j,l}\right)$; b) the variation of consumed resources caused by Δr_f from all the predecessor flows of clique k, expressed as $-\sum_{f\in\mathcal{P}_k}\sum_{l\in\mathcal{L}_k\cap\mathcal{R}_f}(p_{k,l}\Delta r_f)$. By dividing the variation of remaining resources among all the successor flows of this clique, Δs_k can be derived, as shown in the following propagation equation:

$$\Delta s_k = \frac{\frac{\sum_{l\in\mathcal{L}_k\cap\mathcal{L}_j} p_{k,l}}{\sum_{l\in\mathcal{L}_j} p_{j,l}} - \sum_{f\in\mathcal{P}_k}\sum_{l\in\mathcal{L}_k\cap\mathcal{R}_f}(p_{k,l}\Delta r_f)}{|\mathcal{B}_k|}, \tag{21}$$

where $\mathcal{P}_k$ and $\mathcal{B}_k$ represent the predecessor and the successor vertices of clique k in $\mathcal{G}_\mathrm{b}$.

Second, the propagation equation for Δr_f is obtained by taking the minimum of the clique fair share perturbations on flow f's predecessor cliques, as the flow is bottlenecked by the clique with the minimum fair share. Therefore, the perturbation on the flow f's rate is

$$\Delta r_f = \min_{k\in\hat{\mathcal{P}}_f} \Delta s_k, \tag{22}$$

where $\hat{\mathcal{P}}_f$ represents the predecessor vertices of flow f in $\mathcal{G}_\mathrm{b}$.

After two perturbation equations are derived, the algorithm of computing clique gradients named ***CliqueGrad*** can be developed: Starting from the perturbation source clique j, the two equations (21) and (22) are sequentially computed to obtain Δs_k for each clique and Δr_f for each flow in the bottleneck structure. The sequence of propagation is the sequence by which the clique or flow is added into $\mathcal{G}_\mathrm{b}$ in Algorithm 1. The time complexity of CliqueGrad is analyzed as follows. The two propagation equations, Eq. (21) and Eq. (22), run in constant time. To propagate the perturbation from the roots of the bottleneck structure to the leaves, a Min-Heap of size KF is built as in [14], where F is the number of flows. The operation of heap updates runs at most KF times [14], leading to $O(KF\log(KF))$.

It can be proved that the output of CliqueGrad by taking the sum of $\Delta r_f(\forall f\in\mathcal{F})$ is equal to the clique j's gradient, as stated in the following theorem.

Theorem 2 (The correctness of clique gradient computation)**.** *Assume that an infinitely small perturbation is imposed on clique j. By executing CliqueGrad, Δs_k and Δr_f can be achieved as*

$$\Delta s_k = \frac{\partial s_k}{\partial R_j}, \forall k\in\hat{\mathcal{Q}}, \tag{23}$$

$$\Delta r_f = \frac{\partial r_f}{\partial R_j}, \forall f\in\mathcal{F}. \tag{24}$$

As a result, the algorithm output $\hat{\mathbf{g}}(j)$ computed as $\sum_{f\in\mathcal{F}}\Delta r_f$ is equal to the clique gradient $\mathbf{g}(j)$.

Proof. The clique perturbation is obtained by imposing an infinitely small perturbation δ on each of clique j's link capacities, i.e., $\Delta R_j = \sum_{l\in\mathcal{L}_k} p_{k,l}\delta$. To prove Eq. (23) and (24), it is equivalent to prove

$$\tilde{s}_k = s_k + \Delta s_k(\delta \sum_{l\in\mathcal{L}_j} p_{j,l}), \forall k\in\hat{\mathcal{Q}}, \tag{25}$$

$$\tilde{r}_f = r_f + \Delta r_f(\delta \sum_{l\in\mathcal{L}_j} p_{j,l}), \forall f\in\mathcal{F}, \tag{26}$$

where $\tilde{s}_k$, $\tilde{r}_f$ are new values after perturbation.

An induction approach applied in the wired network [14] is followed to prove Eq. (26) and (25). Similar to Definition 5 in [14], clique x or flow f is said to be in the clique k's influence region, provided that the clique fair share s_x of clique x or the flow rate r_f can be impacted by the change of the clique equivalent throughput R_k of clique k. Let $\mathcal{I}(j)$ be the influence region set. Consider any vertex $y\in\mathcal{V}_b$ in the constructed bottleneck structure, where it can be a flow or a clique. If it is not in the influence region of clique j, i.e., $y \notin \mathcal{I}(j)$, it is evident that $\Delta r_y = 0, \forall y \in \mathcal{F}$ or $\Delta s_y = 0, \forall y \in \hat{\mathcal{Q}}$. Thus, Eq. (26) and (25) hold true. For any vertex $y \in \mathcal{I}(j)$, the following proof is given by induction.

To start with, consider vertex y as the the root node in $\mathcal{I}(j)$. Vertex y must be a clique and it has no predecessor vertex. Hence, we replace the notation y by k. The only impact on clique k's fair share must directly come from the perturbation source clique j. Let $\mathcal{M}_{k,j} = \mathcal{L}_k \cap \mathcal{L}_j$, representing the common links of two cliques. Then the new fair share value for clique k can be derived as follows:

$$\begin{aligned}\tilde{s}_k &= \frac{\tilde{R}_k}{|\mathcal{B}_k|} = \frac{R_k + \Delta R_k}{|\mathcal{B}_k|} \\ &= s_k + \frac{\delta \sum_{l \in \mathcal{M}_{k,j}} p_{k,l}}{|\mathcal{B}_k|} \\ &= s_k + \frac{\sum_{l \in \mathcal{M}_{k,j}} p_{k,l} / \sum_{l \in \mathcal{L}_j} p_{j,l}}{|\mathcal{B}_k|} \delta \sum_{l \in \mathcal{L}_j} p_{j,l} \\ &= s_k + \Delta s_k (\delta \sum_{l \in \mathcal{L}_j} p_{j,l}),\end{aligned} \tag{27}$$

where $\mathcal{B}_k$ represent the successor vertices of clique k in the bottleneck structure. The above shows Eq. (25) holds true.

Next, following the methodology of induction, we assume y is any vertex in $\mathcal{I}(j)$ except the root node, and each predecessor vertex of y in $\mathcal{I}(j)$ satisfies Eq. (26) if it is a flow, or Eq. (25) if it is a clique. In the following, it is proved that vertex y itself satisfies one of the two equations, which completes the induction.

Consider the first case that vertex y is a flow, i.e., $y \in \mathcal{F}$, and the index y is replaced by f. It is assumed by induction that the predecessor cliques of flow f satisfies Eq. (25). Combined with the propagation equation (22), it is shown that:

$$\begin{aligned}\tilde{r}_f &= \min_{k \in \hat{\mathcal{P}}_f} \tilde{s}_k \\ &= \min_{k \in \hat{\mathcal{P}}_f} \left(s_k + \Delta s_k \delta \sum_{l \in \mathcal{L}_j} p_{j,l} \right) \\ &= r_f + \delta \sum_{l \in \mathcal{L}_j} p_{j,l} \left(\min_{k \in \mathcal{P}_f^w} \Delta s_k \right) \\ &= r_f + \Delta r_f \left(\delta \sum_{l \in \mathcal{L}_j} p_{j,l} \right),\end{aligned} \tag{28}$$

where $\hat{\mathcal{P}}_f$ represents the predecessor vertices of flow f in the bottleneck structure. This states that Eq. (26) holds true.

Consider the second case that vertex y is a bottleneck clique, i.e., $y \in \hat{\mathcal{Q}}$, and thus the index y is replaced by k. Based on the max-min allocation of clique y's available throughput, we have

$$\tilde{s}_k = \frac{\tilde{R}_k - \sum_{f \in \mathcal{P}_k} \sum_{l \in \mathcal{L}_{k,f}} p_{k,l} \tilde{r}_f}{|\mathcal{B}_k|}, \tag{29}$$

where $\tilde{R}_k = R_k + \Delta R_k$, and $\mathcal{P}_k$ represents the predecessor flows of clique k in the bottleneck structure. Since $\tilde{r}_f$ satisfies Eq. (26) based on induction, we have $\tilde{r}_f = r_f + \Delta r_f (\delta \sum_{l \in \mathcal{L}_j} p_{j,l})$. Also, s_k is expressed as $s_k = \left(R_k - \sum_{f \in \mathcal{P}_k} \sum_{l \in \mathcal{L}_{k,f}} p_{k,l} r_f \right) / |\mathcal{B}_k|$. Therefore, Eq. (29) is further transformed into

$$\begin{aligned}\tilde{s}_k &= s_k + \frac{\Delta R_k - \sum_{f \in \mathcal{P}_k} \sum_{l \in \mathcal{L}_k \cap \mathcal{R}_f} p_{k,l} \Delta r_f \Delta R_j}{|\mathcal{B}_k|} \\ &= s_k + \frac{\Delta R_k / \Delta R_j - \sum_{f \in \mathcal{P}_k} \sum_{l \in \mathcal{L}_k \cap \mathcal{R}_f} p_{k,l} \Delta r_f}{|\mathcal{B}_k|} \Delta R_j \\ &= s_k + \Delta s_k (\delta \sum_{l \in \mathcal{L}_j} p_{j,l}),\end{aligned} \tag{30}$$

which states that Eq. (26) holds true. Note that based on the proof in [14], it suffices to use the original bottleneck structure with the imposed perturbation.

To this end, for any vertex in the bottleneck structure, either Eq. (23) or Eq. (24) is proved to hold true. The algorithm output of *CliqueGrad* is $\hat{\mathbf{g}}(j) = \sum_{f \in \mathcal{F}} \Delta r_f = \sum_{f \in \mathcal{F}} \frac{\partial r_f}{\partial R_j}$, which is exactly the clique gradient $\mathbf{g}(j)$. With that, the proof of Theorem 2 is completed. □

V. DRL-BASED RELAY PLACEMENT WITH CLIQUE-BASED BOTTLENECK THEORY

A. Relay Placement Problem

The problem of relay placement is placing a set of relay nodes $\hat{\mathcal{R}}$ at the selected candidate relay location sites to boost the network throughput. Let $\hat{\mathcal{R}}_{\text{can}}$ be the set of all the candidate sites, and N_{r} be the number of relays that can be deployed considering the total deployment cost. $\hat{\mathcal{R}}_{\text{can}}$ and N_{r} are considered as given information, and $\hat{\mathcal{R}} \in \hat{\mathcal{R}}_{\text{can}}, |\hat{\mathcal{R}}| = N_{\text{r}}$.

With the clique-based bottleneck theory developed in Section IV, two critical theoretical results have been drawn: 1) by constructing the bottleneck structure, the network throughput is determined given a network architecture as in Algorithm 1; 2) the impact of each bottleneck clique on the overall throughput, i.e., the clique gradient, is determined based on CliqueGrad. However, even with these theoretical tools, it is still challenging to solve the relay placement problem. This is because the objective of relay placement in this paper is unique: it aims to maximize the fairness-based network saturation throughput as stated in Section I. To derive such fairness-based network throughput, one must rely on the algorithm of bottleneck structure construction (Algorithm 1) developed in the bottleneck theory. However, the derivation of throughput via Algorithm 1 cannot be analytically expressed as a differentiable function to be optimized. Therefore, this problem does not fall into traditional integer programming [42], and thus the traditional optimization techniques such as branch-and-bound lose their effectiveness here [42]. This motivates us to develop a DRL-based method that does not require the objective function to be differentiable.

Instead of selecting multiple candidate sites at one time that incurs high complexity in DRL search space, we decompose the relay placement problem into ***a multi-step, iterative process*** where one relay is deployed at each step. In each step, a clique-based bottleneck structure is first constructed and then the clique gradients are computed based on the structure. The cliques with H highest gradients are named

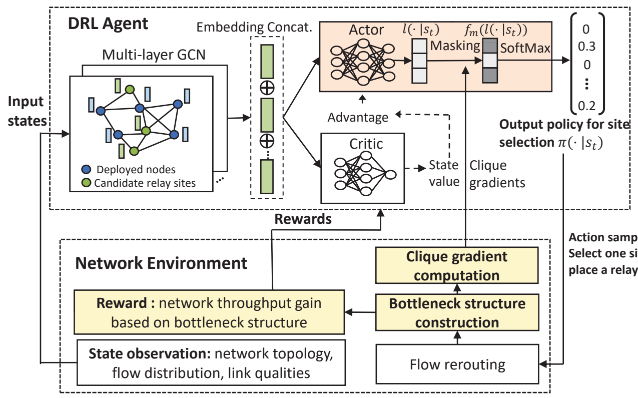

DRL Agent
Embedding Concat.
Multi-layer GCN
Input states
Deployed nodes
Candidate relay sites
Actor
$l(\cdot|s_t)$
$f_m(l(\cdot|s_t))$
Masking
SoftMax
0
0.3
0
0.2
Advantage
Output policy for site selection $\pi(\cdot|s_t)$
Critic
State value
Clique gradients
Rewards
Action sampling: Select one site to place a relay
Network Environment
Clique gradient computation
Reward : network throughput gain based on bottleneck structure
Bottleneck structure construction
State observation: network topology, flow distribution, link qualities
Flow rerouting

Fig. 4. DeepRP: DRL Design Framework

as H***-highest-gradient cliques***, which are identified as ***high-impact cliques*** for the network. The best candidate relay site that is in proximity of these cliques is selected. After deploying the new relay at the selected site by DRL, the flow distribution is changed accordingly by rerouting, and the new network architecture is obtained. The bottleneck structure can be then constructed again to start another iteration. The above DRL-based approach aims to maximize network throughput over multi-step decisions.

B. Key Designs of DeepRP

As shown in Fig. 4, the DRL framework named Deep Relay Placement (***DeepRP***) is designed based on an actor-critic architecture as elaborated in Section V-C. The mesh gateway is considered as the centralized DRL agent, which selects the action a_t based on the current state $\mathbf{s}_t$ at step t. Then, the agent collects the reward r_t from the environment and updates the learnable parameters of DRL agent. Before elaboration on key modules, the state, action, and reward are described as follows.

State representation. Since the information on candidate relay sites is known by the agent, a full virtual network topology $\mathcal{G}_{\rm w} = (\mathcal{V}_{\rm w}, \mathcal{E}_{\rm w})$ is constructed, where the node set $\mathcal{V}_{\rm w}$ includes all the deployed mesh nodes and candidate relay sites, and the edge set $\mathcal{E}_{\rm w}$ includes all the potential communication links among nodes. $\mathcal{G}_{\rm w}$ is considered as an undirected graph, and the edge weight is the estimated SNR of each undirected link normalized into the range of $(0, 1]$. Each node's feature has four elements: whether this node is deployed or not (represented by a binary value), the number of incoming flows, the number of outgoing flows, and the connectivity degree of this node in $\mathcal{G}_{\rm w}$. Let $\mathbf{F}$ denote the matrix of node features with the size $|\mathcal{V}_{\rm w}| \times 4$. Thus, the state is represented by $\mathbf{s}_t = \{\mathcal{G}_{\rm w}, \mathbf{F}\}$.

Action representation. The action a_t is selecting one site among all the candidate sites to place a new relay. As a policy-based DRL method is adopted [43], a_t is sampled based on a probability distribution vector generated by the DRL actor network (i.e., a *DRL policy* $\pi(\cdot|s_t)$). Specifically, the policy vector has the dimension of the number of candidate sites $|\hat{\mathcal{R}}_{\text{can}}|$, and each element of this vector is the probability of choosing a specific site for relay deployment.

Reward function. The reward assigned to step t is the network throughput gain after deploying the new relay, i.e., $r(\mathbf{s}_t, a_t) = T_{t+1} - T_t$, where T_{t+1} is obtained under the new network architecture after action a_t is taken. By computing losses from rewards, the following learnable networks are updated: GCNs $\theta_{\rm g}$, actor network $\theta_{\rm a}$ that generates policy $\pi(\cdot|s_t))$, and critic network $\theta_{\rm v}$ that outputs a scalar $(V^{\pi}(\mathbf{s}))$ based on the current state. The details of loss computation and updating process are in Section V-C.

As shown in Fig. 4, three key modules are designed for DeepRP to efficiently resolve the relay placement problem: 1) GCN-based graph embedding; 2) clique-gradient-based action masking; 3) bottleneck-structure-based reward computation. In the following, these module designs are elaborated.

GCN-based graph embedding. The DRL agent conducts node embedding over the full virtual network topology $\mathcal{G}_{\rm w}$. Let A be the adjacency weight matrix of $\mathcal{G}_{\rm w}$ with self-connections and D is a diagonal matrix whose diagonal elements are $D_{ii} = \sum_j A_{ij}$. Considering a multi-layer GCN, the layer-wise propagation rule [1] for layer ℓ to obtain the $(\ell+1)$-th layer embedding matrix $X^{(\ell+1)}$ is

$$X^{(\ell+1)} = \sigma\left(D^{-1/2} A D^{-1/2} X^{(\ell)} \theta_{\rm g}^{(\ell)}\right), \tag{31}$$

where $X^{(0)} = \mathbf{F}$ is the input node feature matrix, and $\theta_{\rm g}^{(\ell)}$ denotes the parameters of ℓ-th layer GCN, and $\sigma(\cdot)$ is the ReLU activation function. A single-layer GCN aggregates one-hop neighbor nodes' states for each node, and thus each node can obtain a node-level embedding vector that encodes multi-hop node states after multi-layer GCN embedding. As shown in Fig. 4, a graph-level embedding is then obtained by concatenating all the embeddings of candidate relay nodes, and its dimension is $|\hat{\mathcal{R}}_{\text{can}}| d_x$ with $d_{\rm x}$ as the dimension of one node embedding. Note that $|\hat{\mathcal{R}}_{\text{can}}|$ is the number of candidate sites. This concatenated embedding serves as the input for both the actor and critic networks subsequently.

Clique-gradient-based action masking. To generate a DRL policy that enables efficient learning, the logit-based action masking [44] is designed based on clique gradients, which includes three steps. First, a set of valid actions $\mathcal{A}_{\text{valid}}$ is defined: the action of selecting an undeployed candidate site that is in proximity to H-highest-gradient cliques is considered valid. More specifically, a site is in proximity of one clique if it is in one-hop communication range of at least two nodes in the clique. In this way, the range of site selection is significantly narrowed down to be aimed at resolving the high-impact bottlenecks. In addition, if a candidate site is already selected in previous steps, it cannot be selected again, and it is excluded from the valid action set. Second, a masking function $f_m(\cdot)$ is designed, and it is applied on the logits, $l\left(\cdot|s_t\right)$, generated by the last layer of actor network $\theta_{\rm a}$. Let $M = |\hat{\mathcal{R}}_{\text{can}}|$ for ease of notation, and let a^i denote action i is taken based on s_t. The masking function is

$$f_m\left(l\left(a^i|s\right)\right) = \begin{cases} l\left(a^i|s\right), & \text{if } a^i \in A_{\text{valid}} \\ \eta, & \text{if } a^i \notin A_{\text{valid}} \end{cases}, \tag{32}$$

where $i = 1, \ldots, M$, and the invalid actions are masked out by setting their logits to a large negative number $\eta = -1 \times$

10^8 [44]. Third, Softmax activation is applied on the masked logits to generate the final probability distribution over all the actions, i.e., DRL policy $\pi(\cdot|s)$, expressed by

$$\begin{aligned}\pi\left(\cdot|s_t\right) &= \left[\pi\left(a^1|s_t\right),\ldots,\pi\left(a^M|s_t\right)\right]\\ &= \mathrm{softmax}\left(f_m\left(l\left(a^1|s_t\right)\right),\ldots,f_m\left(l\left(a^M|s_t\right)\right)\right)\end{aligned}$$

where $\pi\left(a^i|s_t\right)$ is the probability of selecting action i. As a result, the probability of selecting non-critical relay sites is reset to zero.

Bottleneck-structure-based reward computation. In this paper, the network saturation throughput is theoretically acquired via the construction of bottleneck structures. Specifically, a bottleneck structure is rebuilt via the algorithm of bottleneck structure construction after taking one step of relay placement a_t to obtain throughput T_{t+1}. The throughput gain $(T_{t+1} - T_t)$ is then taken as the step reward r_t.

Based on the design of the second and third key modules, the contributions of bottleneck theory for the DRL design are summarized as follows. First, the clique gradients are used in the action masking scheme to guide the agent to select the sites in proximity to the most critical bottlenecks. Second, the clique-based bottleneck structure is constructed after each decision step, based on which the reward is computed.

C. DRL Training and Inference

1) Offline training process: DeepRP is trained through the actor-critic algorithm Proximal Policy Optimization-Clip (PPO-Clip) [43]. PPO-Clip has been guaranteed with convergence to the local minimum in recent research [45], [46]. Especially, it can attain global optimality with over-parameterized neural networks [45]. The relay placement process of adding N_r relays is viewed as one DRL episode.

In each step t of an episode, as shown in Fig. 4, the DRL agent embeds the input states $\mathbf{s}_t$ through a multi-layer GCN to obtain a graph-level embedding. This embedding goes through the actor and critic network respectively. The actor network outputs the logits for each potential action, and then the clique-gradient-based action masking is applied to obtain a new probability distribution vector over all the actions. Based on the new distribution, action a_t is sampled, i.e., a valid relay site is determined and a new relay is added to the backhaul mesh. Next, a heuristic flow rerouting algorithm named Fair-Share-based (FS-based) rerouting is designed as follows. A new type of link cost c_l for link l is defined as $c_l = \sum_{k\in\hat{\mathcal{Q}}_l} 1/s_k$, where $\hat{\mathcal{Q}}_l$ is the set of cliques containing link l, and s_k is the clique k's fair share with no flow converged. High-impact flows are identified as the flows that traverse the high-impact cliques. Each of these high-impact flows are rerouted by the Dijkstra's algorithm to minimize the total link cost.

After flow rerouting, the clique-based bottleneck structure is constructed to obtain the reward of this step as in Section V-B.When one episode finishes, the episode reward is computed as $r_\mathrm{ep} = \sum_{t=1}^{N_\mathrm{r}}(T_{t+1} - T_t) = T_{N_\mathrm{r}}$, which is the network throughput under the augmented architecture with N_r relays. After collecting a set of episodes, the neural network parameters are updated by stochastic gradient descent. Two loss functions are specified. First, the actor loss is the standard PPO-Clip objective referred to equation (7) in [43], where the advantage is estimated by time difference residual, i.e., $A_t^\pi = r_t + V^\pi(\mathbf{s}_{t+1}) - V^\pi(\mathbf{s}_t)$. The gradient with respect to the actor loss is used to update $\theta_\mathrm{g}, \theta_\mathrm{a}$. Second, the critic loss is the mean-square error between the reward-to-go and state values across this set of episodes as in [43]. The gradient with respect to the critic loss is used to update $\theta_\mathrm{g}, \theta_\mathrm{v}$. The DRL agent is continuously trained over episodes until convergence.

2) Online inference process: For online inference, network designers first need to determine a fixed set of candidate relay sites for the network area of interest. Based on this given information, DeepRP is then run online for N_r steps with the trained DRL agent, and the selected relays are physically deployed in the backhaul network.

The online time complexity of DeepRP is analyzed as follows. The complexity of bottleneck structure construction and clique gradient computation is analyzed in Section IV-A and IV-B respectively. For the neural network inference, the time complexity depends on the number of weight multiplications. Through GCNs, the number of multiplications is $O(c_1|\mathcal{V}_\mathrm{w}|)$; through the actor network, the number of multiplications is $O(c_2|\mathcal{R}_\mathrm{can}|)$, where c_1, c_2 are NN-related constants. In addition, the re-routing algorithm has the same time complexity as Dijkstra's algorithm, $O(L\log N)$, where N, L are the number of deployed nodes and links respectively. As DeepRP needs to be run online for N_r times, the overall time complexity is $O(N_\mathrm{r}KF\log(KF) + N_\mathrm{r}L\log N + c_1N_\mathrm{r}|\mathcal{V}_\mathrm{w}| + c_2N_\mathrm{r}|\mathcal{R}_\mathrm{can}|)$.

VI. Performance Evaluation

A. Experiment Settings

1) Simulation set-up: In this paper, relay placement is conducted for each backhaul mesh topology within the 2D range 1km$\times$1km. To generate these typologies, a real-world dataset is used in which 46050 BSs are densely deployed in an urban environment [47]. 1000 different topologies each of 1km$\times$1km are sampled from this dataset, and they are further divided into the training (70%) and test set (30%). In each sampled network topology, the candidate relay sites are uniformly distributed over the area. For each topology, a THz backhaul mesh network as in Section III is set up with the bandwidth as 1 GHz and carrier frequency as 0.1 THz. All the network nodes including the relays have the total transmit power of 36 dBm. The noise power spectrum density is -168 dBm/Hz. The SNR threshold for communication links is set to 20 dB. The interference range is characterized by the interfering distance and interfering angle, fixed at 300 m and $\pi/12$ for each side of the main lobe. The end-to-end aggregate flows are generated as follows: there exist one aggregate flow from each mesh node to the mesh gateway, and multiple flows from each mesh node to a few other reachable mesh nodes. For each backhaul topology, 100 different patterns of flow distribution are generated given a certain number of aggregate flows. Before relay placement, the single-path routing is conducted for each flow.

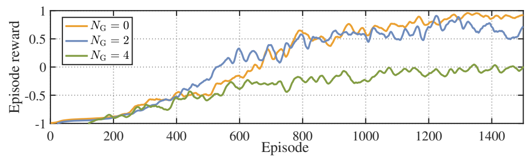
2) DRL Training set-up: The graph embedding network has three layers with 4, 64, 4 neurons in each layer respectively. With the number of candidate relay sites fixed as 50, the actor network has five layers with 200, 128, 128, 128, 50 neurons in each layer respectively. The critic network has four layers with 200, 256, 256, 1 neurons in each layer respectively. Note that in the last layer of the actor network, Softmax activation is used; except that, in each layer of the above three networks, ReLU activation is used. In summary, the total number of parameters is reported as 183,287. The learning rates for actor and critic networks are set to 0.0001 and 0.003 respectively, and the discount factor is set to 0.99.

3) Baseline Approaches: Various baselines of relay placement are explained as follows. First, the random relay placement scheme (**RandomRP**) uniformly samples N_r sites from all the candidate sites in the topology to place relays. Second, the heuristic relay placement scheme without clique gradients (**HeurRP-wo-grad**) [30] is an iterative approach which greedily chooses one relay site with the maximum connectivity degree to the deployed nodes at each iteration, until N_r relays are placed. If two relay sites have the same connectivity degree, this scheme selects the relay site with better average link quality. Third, the heuristic relay placement scheme with clique gradients (**HeurRP-with-grad**) follows the similar greedy approach as the second one, but it narrows down the site selection range to the proximity of the H-highest-gradient cliques. Based on relay placement, several flow rerouting schemes are compared as follows. These schemes can be viewed as the variants of Dijkstra's algorithm with different definitions of link costs: 1) in shortest-path rerouting, the link cost is constant as one; 2) in traffic-load-based (TL-based) rerouting [48], the link cost is the number of traversed flows on this link; 3) in link-capacity-based (LC-based) rerouting, the link cost is taken as the reciprocal of the link capacity [8]. For the above schemes, the rerouted paths should not be more than K hops, which is exerted as the maximum-hop constraint for Dijkstra's algorithm [48]. K is set to the original path length plus two hops in this paper.

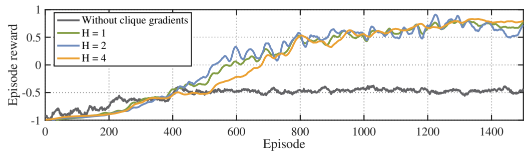
Key simulation parameters are set as follows unless otherwise stated: $f_c = 0.1\text{THz}, G_r = G_t = 18\text{dB}$, absorption coefficient $k_{\text{abs}}(f_c) = 0.0033\ \mathrm{m}^{-1}$, time duration $T_s = 50\text{ns}$, shadowing standard deviation $\sigma_X = 7\text{dB}$, cluster arrival rate $\lambda = 0.13\ \mathrm{ns}^{-1}$, cluster decay factor $\Gamma = 3.12\text{ns}$, path loss exponent $n = 2$; the number of relays to be deployed $N_\mathrm{r} = 10$; the uniform distribution density of candidate relay sites (also termed as ***candidate site density***) is 0.0001 per m^2; the number of end-to-end flows is 200. The results are averaged over the test topologies with the deployed node density around 0.5×10^{-4} per m^2, and each test topology is averaged over 100 uniform relay distribution. Note that the episode rewards are normalized between -1 and 1 during training.

B. Validation of DeepRP

1) Node embedding module: Three different cases are considered, where the numbers of GCN layers are 0, 2, and 4, respectively. The node embedding module is removed in the first case. As shown in Fig. 5(a), the training of DRL agent can reach convergence under all three cases, with the episode rewards stable after 1200 training episodes. However, the converged rewards without node embedding is significantly lower than the other two. This difference is also reflected in the network throughput obtained by three schemes, as shown in 6(a). Under various relay densities, DeepRP with GCN-based node embedding achieves $7.88 - 8.29\%$ higher throughput than that without it, which validates the necessity of node embedding. As 2-layer and 4-layer GCNs can achieve similar performance, the 2-layer GCN is selected for DeepRP to reduce computation complexity. This design also matches the physical interference pattern, where the nodes within two hops may interfere with each other even with beamforming.



(a) Various numbers of GCN layers

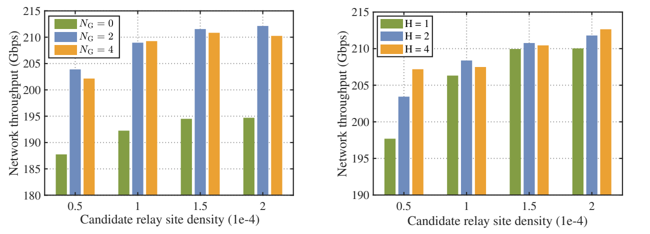


(b) Various numbers of high-impact cliques

Fig. 5. Episode rewards of DRL training.

2) H-highest-gradient cliques: As described in Section V-C, H-highest-gradient cliques are identified and used in action masking. Different values of H are considered, i.e., H = 1, 2, 4. The case without clique gradients is also considered, where an action can be selected out of all the candidate sites. However, as shown in Fig. 5(b), the training without clique gradients cannot converge within 1400 episodes, which validates the necessity of embedding clique gradients into DeepRP. With clique gradients, the agent is guided to place relays targeted at the most critical bottlenecks. Thus, more useful actions can be generated during training, resulting in an efficient convergence. The remaining cases are compared in Fig. 6(b). As the candidate site density increases, the performance with H = 1 increases by 7.14%, while the case with H = 4 does not see much gain. This can be interpreted as follows: when the candidate site density is low, there may not be enough relays near the highest-gradient clique, and in this case setting H to a larger value brings the opportunity of exploring more relays; when the candidate site density is high enough, setting a large H cannot bring much gain. Therefore, the value of H should be adjusted with the candidate site density in practice. In the following evaluation, H is set to 2 for the site density of 0.0001 per m^2 or above, and set to 4 otherwise. It should be noted that the trained agent is capable of adapting to new network scenarios without retraining, as the DRL agent is trained with diverse backhaul topologies and flow distributions.

3) Flow rerouting scheme: Four flow rerouting schemes are compared with various numbers of flows in the network. As the number of flows increases from 100 to 500, RS-based rerouting can achieve the throughput at most 5.53% higher than TL-based rerouting, and 16.82% higher than the other two schemes. Therefore, RS-based rerouting is adopted in the following evaluation.



(a) Various numbers of GCN layers (b) Various numbers of high-impact cliques

Fig. 6. Network throughput achieved by DeepRP.

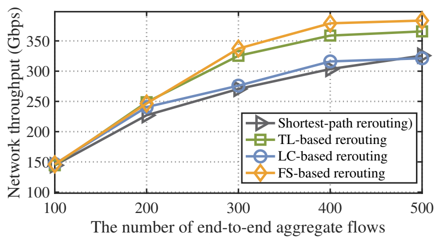


Fig. 7. Network throughput with different flow rerouting methods.

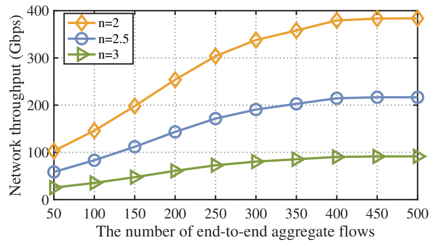


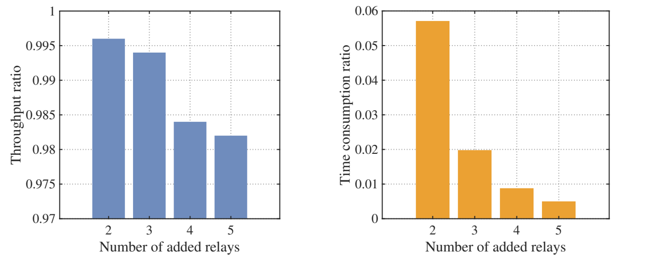
Fig. 8. Impact of various path loss exponent values.

4) Various path loss exponent: The impact of various path loss exponent n is studied, specifically $n = 2, 2.5, 3$, where $n = 2$ means the free space propagation and a larger n means a more lossy environment. The added simulations are shown below. The number of added relays is fixed at 30. As the flow density in the network increases, the network throughput reaches a stable level. As shown in Fig. 8. The throughputs when $n = 2.5$ and $n = 3$ are approximately 56.4% and 23.8% of that when $n = 2$. The reason for throughput decrease is straightforward: as n increases, the path losses become larger, and thus the receive SNRs and link rates become lower.

C. Comparisons with Baseline Approaches

1) Global optimality for small-scale problems: To empirically evaluate the optimality of DeepRP, we compare it with the exhaustive search scheme for small-scale problems (15 candidate sites in total and the number of added relays varying from 2 to 5). As shown in Fig. 9, DeepRP achieves the throughput that is 98.2%-99.6% of the optimal throughput obtained by exhaustive search. The inference time consumption of DeepRP is below 1% of the time consumed by exhaustive search when the number of added relays is 5. For large-scale problems, the time consumption of exhaustive search is not tractable.

2) Network throughput: First, the network throughput with different numbers of added relays is compared as shown in Fig. 10. As the number of added relays increases from 0 to 20, there exists a rapid increase in network throughput with



(a) Throughput ratio (DeepRP vs. Exhaustive) (b) Time consumption ratio (DeepRP vs. Exhaustive)

Fig. 9. Comparisons with the exhaustive search in small-scale problems.

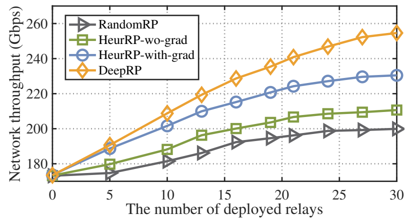


Fig. 10. Network throughput with various numbers of deployed relays.

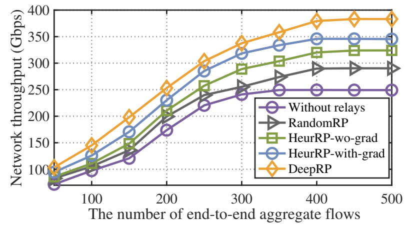


Fig. 11. Network throughput with various numbers of flows.

DeepRP, but the upward trend becomes much slower with the number of added relays above 20. This is due to the fact that the relays at critical locations have been exploited. Furthermore, DeepRP outperforms HeurRP-with-grad by a margin of $5.76 - 10.37\%$ in network throughput when the number of added relay is larger than 10. Interestingly, HeurRP-with-grad can achieve $8.67 - 10.11\%$ higher throughput than HeurRP-wo-grad, which indicates the effectiveness of clique gradients in guiding relay placement. Second, the network throughput with various numbers of flows is evaluated as in Fig. 11. In this case, the number of added relays for four schemes is fixed at 30. As the flow density in the network increases, the network throughput reaches a stable level, where the throughput obtained by DeepRP is 10.42% higher than HeurRP-with-grad, and 21.51% and 32.11% higher than HeurRP-wo-grad and RandomRP on average, respectively. Third, the throughput gain over the original network without relays is evaluated under various relay densities, as shown in Fig. 12. In each topology, 20 relays are deployed. As the sites become denser especially over 0.0001 per m^2, the throughput gain achieved by DeepRP becomes more significant, which is $40.65 - 50.84\%$. This is because a dense candidate site distribution can increase the chance of finding relays near the most critical bottleneck. Furthermore, when the candidate site density is above 0.0001 per m^2, DeepRP can achieve a 13.36% larger gain than HeurRP-with-grad, and a 25.38% larger gain than HeurRP-wo-grad.

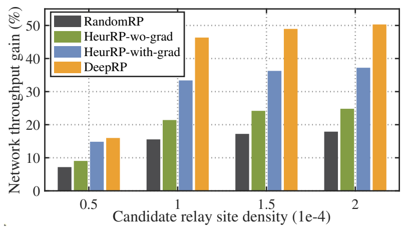


Fig. 12. Network throughput gain over the original network under various candidate relay site densities.

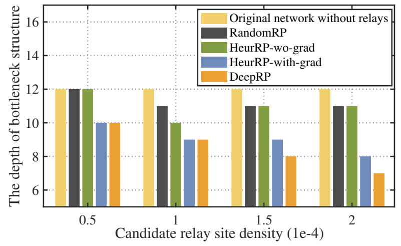


Fig. 13. The depth of clique-based bottleneck structures.

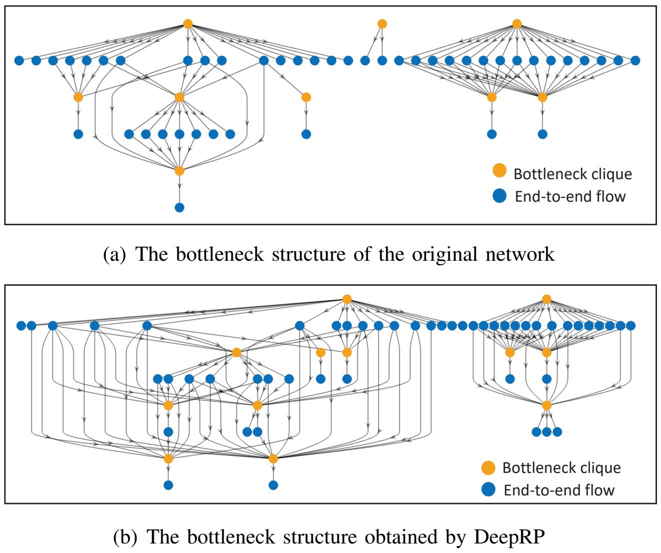

Bottleneck clique
End-to-end flow
(a) The bottleneck structure of the original network
Bottleneck clique
End-to-end flow
(b) The bottleneck structure obtained by DeepRP

Fig. 14. The comparison of clique-based bottleneck structures.

3) Depth of Bottleneck structure: As proved in the original theory [6], the smaller the depth of bottleneck structure is (i.e., the longest length of a directed path), the better the network congestion performance is, which also fits the clique-based case. With 30 deployed relays and 200 flows, the comparison between the average depths of clique-based bottleneck structures is shown in Fig. 13. As the site density increases, the gap between the depth of the original network's structure and the depth of augmented network's structure by DeepRP becomes larger. In all the cases, DeepRP achieves the shallowest structure, and it shrinks the structure depth by 41.67% than the original one when the density reaches 0.0002 per m^2.

4) Clique-based Bottleneck Structure: The bottleneck structures of the original network and the augmented network by DeepRP are compared in Fig. 14. To ease illustration, the number of flows is set to 50, and the number of added relays is 10. The original bottleneck structure has 7 levels with the longest directed path length as 7. After relay placement by DeepRP, the interactions among bottlenecks and flows are much simplified such that a 5-level structure with fewer directed paths. DeepRP can achieve more simplified structures than the baseline methods, but such results are omitted here due to space limit.

VII. Conclusion and Future Work

The problem of augmenting 6G backhaul architecture with relays to boost network throughput was studied in this paper. Via developing the clique-based bottleneck theory, the interactions among bottlenecks in the wireless case was captured, from which the relationship between network architecture and network throughput was obtained. The clique gradients were then defined and computed based on the constructed structure, to identify the critical bottlenecks in the network. By integrating the clique-based theory, a DRL-based relay placement approach DeepRP was designed to accomplish relay placement. It is the first work that establishes the bottleneck theory for wireless networks and performs relay placement based on this theory.

There exist two interesting topics that need further investigation. First, the performance guarantee of DeepRP is dependent on the clique-based bottleneck theory. More theoretical analysis on the connection between the developed theory and DeepRP will be provided in future work. Second, for a general wireless mesh network rather than backhaul mesh, a deterministic flow state may be hard to capture, and thus the relay placement need to be targeted at statistical flow states with random source-destination pairs. How to place relays to achieve higher throughput in such scenarios is subject to future study.

References

[1] Z. Xiao, L. Zhu, Y. Liu, P. Yi, R. Zhang, X.-G. Xia, and R. Schober, "A survey on millimeter-wave beamforming enabled UAV communications and networking," *IEEE Commun. Surv. Tutor.*, vol. 24, no. 1, pp. 557–610, 2022.

[2] C. Saha and H. S. Dhillon, "Millimeter wave integrated access and backhaul in 5G: Performance analysis and design insights," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 12, pp. 2669–2684, 2019.

[3] N. Chukhno, A. Orsino, J. Torsner, A. Iera, and G. Araniti, "5G NR sidelink multi-hop transmission in public safety and factory automation scenarios," *IEEE Netw.*, pp. 1–7, 2023.

[4] X. Lin, S. Cioni, G. Charbit, N. Chuberre, S. Hellsten, and J.-F. Boutillon, "On the path to 6G: Embracing the next wave of low earth orbit satellite access," *IEEE Commun. Mag.*, vol. 59, no. 12, pp. 36–42, 2021.

[5] D. P. Bertsekas and R. G. Gallager, *Data networks*, 2nd ed. Prentice Hall, Dec 1992.

[6] J. Ros-Giralt, A. Bohara, S. Yellamraju, M. H. Langston, R. Lethin, Y. Jiang, L. Tassiulas, J. Li, Y. Tan, and M. Veeraraghavan, "On the bottleneck structure of congestion-controlled networks," in *Proc. ACM SIGMETRICS*, Dec. 2019, pp. 1–31.

[7] Y. Yan, Q. Hu, and D. M. Blough, "Load-balanced routing for hybrid fiber/wireless backhaul networks," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2021, pp. 1–6.

[8] C. K. Toh, A.-n. Le, and Y.-z. Cho, "Load balanced routing protocols for ad hoc mobile wireless networks," *IEEE Commun. Mag.*, vol. 47, no. 8, pp. 78–84, 2009.

[9] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1452–1463, 2006.

[10] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proc. of the IEEE*, vol. 95, no. 1, pp. 255–312, 2007.

[11] L. Hai, Q. Gao, J. Wang, H. Zhuang, and P. Wang, "Delay-optimal back-pressure routing algorithm for multihop wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2617–2630, 2018.

[12] Y. Yan, Q. Hu, and D. M. Blough, "Feasibility of multipath construction in mmwave backhaul," in *Proc. IEEE WoWMoM*, 2021, pp. 81–90.

[13] J. Li, Y. Niu, H. Wu, B. Ai, R. He, N. Wang, and S. Chen, "Joint optimization of relay selection and transmission scheduling for UAV-aided mmwave vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 5, pp. 6322–6334, 2023.

[14] J. Ros-Giralt, N. Amsel, S. Yellamraju, J. Ezick, R. Lethin, Y. Jiang, A. Feng, and L. Tassiulas, "A quantitative theory of bottleneck structures for data networks," *arXiv preprint arVix:2210.03534*, Oct. 2022.

[15] J. Ros-Giralt, N. Amsel, S. Yellamraju, J. Ezick, R. Lethin, Y. Jiang, A. Feng, L. Tassiulas, Z. Wu, M. Y. Teh, and K. Bergman, "Designing data center networks using bottleneck structures," in *Proc. ACM SIGCOMM*, Aug. 2021, pp. 319–348.

[16] T. Wang and X. Wang, "Boosting capacity for 6G terahertz mesh networks based on bottleneck structures," in *Proc. IEEE GLOBECOM*, 2023.

[17] A. Abdelmoaty, D. Naboulsi, G. Dahman, G. Poitau, and F. Gagnon, "Resilient topology design for wireless backhaul: A deep reinforcement learning approach," *IEEE Wirel. Commun. Lett.*, vol. 11, no. 12, pp. 2532–2536, 2022.

[18] C. Huang and X. Wang, "A Bayesian approach to the design of backhauling topology for 5G IAB networks," *IEEE Trans. Mob. Comput.*, vol. 22, no. 4, pp. 1867–1879, 2023.

[19] M. Simsek, O. Orhan, M. Nassar, O. Elibol, and H. Nikopour, "IAB topology design: A graph embedding and deep reinforcement learning approach," *IEEE Commun. Lett.*, vol. 25, no. 2, pp. 489–493, 2021.

[20] M. Diamanti, P. Charatsaris, E. E. Tsiropoulou, and S. Papavassiliou, "The prospect of reconfigurable intelligent surfaces in integrated access and backhaul networks," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 2, pp. 859–872, 2021.

[21] Q. Hu and D. M. Blough, "Relay selection and scheduling for millimeter wave backhaul in urban environments," in *Proc IEEE Int. Conf. on Mob. Ad Hoc and Sens. Sys. (MASS)*, 2017, pp. 206–214.

[22] J. Chen, U. Mitra, and D. Gesbert, "3D urban UAV relay placement: Linear complexity algorithm and analysis," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 5243–5257, 2021.

[23] K. Ntontin, D. Selimis, A.-A. A. Boulogeorgos, A. Alexandridis, A. Tsolis, V. Vlachodimitropoulos, and F. Lazarakis, "Optimal reconfigurable intelligent surface placement in millimeter-wave communications," in *Proc Euro. Conf. on Ant. and Prop. (EuCAP)*, 2021, pp. 1–5.

[24] N. U. Saqib, S. Hou, S. H. Chae, and S.-W. Jeon, "Reconfigurable intelligent surface aided hybrid beamforming: Optimal placement and beamforming design," *IEEE Trans. Wirel. Commun.*, 2024.

[25] M. Nikolov and Z. J. Haas, "Relay placement in wireless networks: Minimizing communication cost," *IEEE Trans. Wirel. Commun.*, vol. 15, no. 5, pp. 3587–3602, 2016.

[26] M. Minelli, M. Ma, M. Coupechoux, J.-M. Kelif, M. Sigelle, and P. Godlewski, "Optimal relay placement in cellular networks," *IEEE Trans. Wirel. Commun.*, vol. 13, no. 2, pp. 998–1009, 2014.

[27] R. Magán-Carrión, R. A. Rodríguez-Gómez, J. Camacho, and P. García-Teodoro, "Optimal relay placement in multi-hop wireless networks," *Ad Hoc Netw.*, vol. 46, pp. 23–36, 2016.

[28] C.-Y. Chang, C.-T. Chang, T.-C. Wang, and M.-H. Li, "Throughput-enhanced relay placement mechanism in wimax 802.16 j multihop relay networks," *IEEE Syst. J.*, vol. 9, no. 3, pp. 728–742, 2014.

[29] Z. Yang, Q. Zhang, and Z. Niu, "Throughput improvement by joint relay selection and link scheduling in relay-assisted cellular networks," *IEEE Trans. Veh. Technol.*, vol. 61, no. 6, pp. 2824–2835, 2012.

[30] R. Magán-Carrión, R. A. Rodríguez-Gómez, J. Camacho, and P. García-Teodoro, "Optimal relay placement in multi-hop wireless networks," *Ad Hoc Netw.*, vol. 46, pp. 23–36, 2016.

[31] W. Liang, C. Ma, M. Zheng, and L. Luo, "Relay node placement in wireless sensor networks: From theory to practice," *IEEE Trans. Mob. Comput.*, vol. 20, no. 4, pp. 1602–1613, 2021.

[32] E. F. Flushing and G. A. D. Caro, "A flow-based optimization model for throughput-oriented relay node placement in wireless sensor networks," in *Proc. Annu. ACM Symp. Appl. Comput.*, March 2013, pp. 632–639.

[33] R. Li and P. Patras, "Max-min fair resource allocation in millimetre-wave backhauls," *IEEE Trans. Mob. Comput.*, vol. 19, no. 8, pp. 1879–1895, 2020.

[34] X. L. Huang and B. Bensaou, "On max-min fairness and scheduling in wireless ad-hoc networks: Analytical framework and implementation," in *Proc. ACM Mobihoc*, 2001, p. 221–231.

[35] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *Proc. ACM Mobicom*, 2003, pp. 66–80.

[36] I.-F. Chao and W.-S. Hsu, "A MAC protocol design for maximizing end-to-end throughput and fairness guarantee in chain-based multi-hop wireless backhaul networks," *IEEE Trans. Mob. Comput.*, vol. 22, no. 7, pp. 3743–3756, 2023.

[37] C. Lin and G. Y. Li, "Indoor terahertz communications: How many antenna arrays are needed?" *IEEE Trans. Wirel. Commun.*, vol. 14, no. 6, pp. 3097–3107, 2015.

[38] Y. S. Cho, J. Kim, W. Y. Yang, and C. G. Kang, *MIMO-OFDM Wireless Communications with MATLAB*. John Wiley and Sons, 2010.

[39] T. Wang, S. Chen, Y. Zhu, A. Tang, and X. Wang, "Linkslice: Fine-grained network slice enforcement based on deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 8, pp. 2378–2394, 2022.

[40] J. Cheng, Y. Ke, A. W.-C. Fu, J. X. Yu, and L. Zhu, "Finding maximal cliques in massive networks," *ACM Trans. Database Syst.*, vol. 36, no. 4, pp. 1–34, 2011.

[41] P. Wang, H. Jiang, W. Zhuang, and H. V. Poor, "Redefinition of max-min fairness in multi-hop wireless networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 12, pp. 4786–4791, 2008.

[42] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, "Mixed-integer nonlinear optimization," *Acta Numerica*, vol. 22, pp. 1–131, 2013.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[44] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *arXiv preprint arXiv:2006.14171*, 2020.

[45] B. Liu, Q. Cai, Z. Yang, and Z. Wang, "Neural trust region/proximal policy optimization attains globally optimal policy," in *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[46] N. Huang, P.-C. Hsieh, K.-H. Ho, and I.-C. Wu, "Ppo-clip attains global optimality: Towards deeper understandings of clipping," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 11, 2024, pp. 12 600–12 607.

[47] J. Zhang, S. Chen, X. Wang, and Y. Zhu, "DeepReserve: Dynamic edge server reservation for connected vehicles with deep reinforcement learning," in *Proc. IEEE Conf. Compt. Commun. (INFOCOM)*, 2021, pp. 1–10.

[48] J. Gao and L. Zhang, "Load-balanced short-path routing in wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 4, pp. 377–388, 2006.

Tianxin Wang received the B.S. degree in information science and engineering from Southeast University, Nanjing, China, in 2020. She is currently pursuing the Ph.D. degree with the Wireless Networking and Artificial Intelligence Laboratory, Shanghai Jiao Tong University, Shanghai, China. From 2023 to present, she is visiting Intelligent Transmission and Processing Lab in Imperial College London. Her research interests include wireless networking and machine learning.