Maximizing Key Distribution Capability: An Application in Quantum Cryptography

Tu N. Nguyen*, Dung H. P. Nguyen†, Manh V. Nguyen*, Thinh V. Le*, Bing-Hong Liu†, and Thang N. Dinh‡
*Department of Computer Science, Kennesaw State University, Marietta, GA 30060, USA.
†Department of Electronic Engineering, National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan.
‡Department of Computer Science, Virginia Commonwealth University, VA 23284 USA.

***Abstract*—Quantum key distribution (QKD) integrated wavelength-division multiplexing (WDM) offers an information-theoretically secure solution to the key exchange problem. In this paper, we investigate the joint problem of how to efficiently schedule QKD to provision sufficient secret keys over WDM networks. Specifically, we propose and formulate the problem of maximizing key distribution capability (MKDC) by employing a mixed integer linear programming (MILP) model for the first time. We then suggest two near-optimal algorithms adopted to address larger-scale problems in polynomial time. One employs the linear programming relaxation technique combined with a rounding algorithm (LPR-RA) and the other is inspired by the fact that the application with a higher risk of disruption is prioritized to recharge secret keys, dubbed progressive serving algorithm (PSA). Simulation results show that both the LPR-RA and PSA can approach the best feasible solution or the upper-bound of the problem. In addition, by means of using NetSquid, an open simulator for quantum networks, we conduct experiments to get insight into the BB84, a protocol applied popularly in QKD networks nowadays.**

***Index Terms*—quantum key distribution (QKD), security applications, secret keys, key rate, consumption rate.**

I. Introduction

Quantum cryptography. The occurrence of quantum key distribution (QKD) [1]–[4] offers a promising technology for the distribution of information-theoretically secure keys. The difference distinguishing this technique from the traditional method is that the QKD employs the underlying principles of quantum mechanics, such as the quantum no-cloning theorem, instead of the computational complexity [5], [6]. For a simple QKD network involving two QKD nodes that seek to exchange secret keys, the process begins with connecting these nodes together over a dedicated quantum channel. This quantum channel is the lifeline of secure communication, ensuring that the transmission of quantum states remains isolated from potential eavesdroppers. They employee the conventional point-to-point QKD protocols, such as BB84 [7], COW [8], and GG02 [9], to securely share keys. One of the major operations of the above protocols is to transmit photons, fragile signals, across the quantum channel. Hence, one intends to use the low-noise dedicated fiber as the quantum channel to protect such signals from the interference of classical signals.

We thank the anonymous reviewers for their suggestions and feedback. This research was in part supported by US NSF under Grants: AMPS-2229073, AMPS-2229075, and CNS-2103405. Corresponding author: Tu N. Nguyen. The source code is released at: https://github.com/NextCNS/MaxMinProb.

QKD over wavelength-division multiplexing (WDM). Nevertheless, the dedicated fiber is costly, and deploying new networks for QKD only is expensive, complicated, and time-consuming; therefore, the deployment of new QKD networks is not feasible in practice. One potential approach is to take advantage of traditional networks to deploy QKD networks. The foundation of this approach is the wavelength-division multiplexing (WDM) technique [10], [11]. Accordingly, leveraging the capability of conveying multiple types of photons with various frequencies of optical fibers, the quantum and classical signals will be transmitted on the same optical fiber but use different wavelengths. Some studies [12], [13] have proposed techniques to separate quantum signals from classical signals and diminish the interference induced by the classical signals, which shows the feasibility of the approach. Eventually, this method is applied to deploy many QKD networks in reality, such as Madrid and Cambridge QKD networks [14], [15].

Fiber-based QKD networks and current challenges. Nonetheless, due to diverse applications running on optical fibers of traditional networks, the shortage of wavelength resources becomes one of the challenges that need to be bridged. Because a quantum channel cannot be shared with other data channels, we need to assign separate wavelengths to quantum channels [16]. However, we cannot allocate too many wavelengths to act as quantum channels of the QKD network. Doing so will diminish the performance of other applications stably running on the traditional network due to the reduction of the wavelengths used for data transmission [17]. Therefore, a limited number of wavelengths is assigned to form quantum channels of the QKD network, and the most important is how to utilize these links efficiently.

Motivation. Given limited wavelength resources in existing WDM networks – investigating the problem of how to efficiently schedule QKD to provision sufficient secret keys over WDM-based QKD networks – *is equally imperative.* Moreover, because end-to-end key distribution is essential for the quantum internet, a routing strategy plays an important role in enhancing the key distribution capability in quantum networks. Previous studies have considered the above issues in different contexts of the entanglement routing, however, none of these considers the nexus between the key generation and consumption. In the following, we review the state-of-the-art literature of QKD and associated routing algorithms.

- The most popular routing technique applied for QKD networks in practice is the Open Shortest Path First (OSPF) [18], [19], in which the Dijkstra-based algorithms are applied to find paths used for distributing secret keys. However, the network resources are not efficiently allocated for for QKD, particularly when either the number of requests for recharging keys or the network scale increases.
- To enhance the capability of the key distribution, some studies focus on improving the routing techniques, such as multiple paths [20] and random paths [21], [22]. However, none of these can maximize the key distribution capability.
- The authors in [23], [24] aim to maximize the key generation rate and minimize the consumed resources. Nevertheless, these studies do not take the key consumption of applications into account while the nexus between key generation and consumption is critical.

Our contribution. This work contributes to the development of QKD networks, specifically enhancing the key distribution capability by elaborating a novel routing perspective of the key distribution problem. We summarize key **innovation** and **contribution** of this work as follows:

- We first put forth a new architecture of QKD-over-WDM networks, in which quantum key pools (QKPs) are particularly designed to provision sufficient secret keys to the network applications.
- We deeply investigate the problem of maximizing key distribution capability (MKDC) through efficiently scheduling QKD over WDM networks.
- The MKDC problem is formulated with a mixed integer linear programming (MILP) model that can yield an optimal solution. In addition, two near-optimal algorithms, including the linear programming relaxation and rounding algorithm (LPR-RA) and progressive serving algorithm (PSA), are proposed to address the MKDC with larger instances.
- The proposed architecture and model are realized via NetSquid [25], a quantum network simulator, to validate the feasibility of QKD networks over the WDM. Furthermore, we conduct extensive simulations to evaluate the performance and efficacy of the proposed algorithms.

Organization. The rest of the paper is organized as follows. Section II presents the preliminaries to the research problem. The system model and research problem will be discussed in Section III. The proposed algorithms will be described in Section IV. Section V presents the implementation of the proposed architecture and model on NetSquid, a quantum network simulator, and discusses the results obtained from the realization. Section VI demonstrates simulation results. Finally, we conclude the paper with remarks in Section VII.

II. Preliminaries: QKD over WDM

Point-to-point QKD: QKD promises to offer security in communication using the laws of quantum mechanics. The protocol broadly employed in current QKD networks is BB84 [1]. This protocol is implemented to share secret keys between

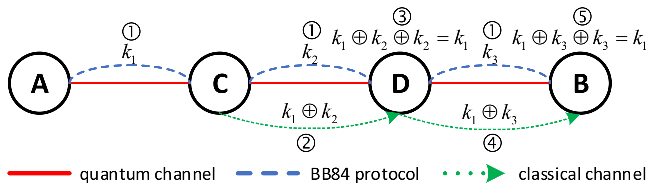

Fig. 1. An illustration of the key relay process with two intermediate nodes.

two nodes directly connected over a quantum channel, say Alice and Bob. Alice will randomly send Bob a bunch of photons with different polarization. Bob also randomly uses disparate bases to measure the photons to detect the polarization of the photons. They then exchange information over the classical channel to verify which bases are correctly used by Bob. Eventually, Alice agrees with Bob on the sequence of bits based on the verification result.

Software-defined networking (SDN): SDN [26], [27] is a novel technology in which the control and data planes are separated. This characteristic allows to integrate new services and upgrade existing services rapidly. Moreover, this network architecture enables centralized management over a controller; hence, it also provides many chances for optimization in order to improve network performance. With such advantages, the SDN architecture is proposed to be the control layer of the current QKD networks. Accordingly, a controller is responsible for collecting requests for generating keys and allocating resources to satisfy the requests based on the global network information it is holding.

Key relay process: The point-to-point protocol presented above is applied to generate keys between two connected nodes by a quantum channel (also called adjacent nodes). For two remote nodes, we have to employ a so-called key relay process to share keys based on keys generated between adjacent nodes [28], [29]. Fig. 1 illustrates an example of this process in which nodes A and B need to share secret keys over intermediate nodes C and D. In the first step (①), keys will be generated between two adjacent nodes using the point-to-point protocol, that is, k_1 between A and C, k_2 between C and D, and k_3 between D and B. In the second step (②), C sends the XOR result between k_1 and k_2 ($k_1 \oplus k_2$) to D. After receiving $k_1 \oplus k_2$ from C, D will recover key k_1 by performing the XOR operation between the received result and k_2 (③), that is, $k_1 \oplus k_2 \oplus k_2 = k_1$. Node D and B then repeated steps ② and ③ in turn (steps ④ and ⑤), that is, D sends $k_1 \oplus k_3$ to B, and B performs $k_1 \oplus k_3 \oplus k_3$ to recover k_1, which is also the secret key that A wants to share with B. We summarize the critical characteristics of the key relay process as follows.

1) The relay paths connecting two nodes need to be predetermined. We can employ one or more paths to distribute one key depending on the requirement of the security level of the system [20].
2) As presented above, the secret key will be relayed at intermediate nodes. Therefore, if an eavesdropper can control only one intermediate node, the secret key will be revealed. To guarantee security, we suppose that all nodes are trusted once using the key relay process.

3) Because the key generation rates on different quantum channels are disparate depending on the length of quantum channels, we need to equip nodes with key storage to store generated keys in order to ensure that faster channels are not limited by slower channels.

Quantum key pool: There are two factors associated with the availability of security-hungry applications that we need to consider in QKD networks, which are key generation rate and consumption rate. The generation rate is required to be faster than the consumption rate to guarantee that there are always sufficient secret keys for the operation of the applications; otherwise, the applications will be interrupted. Unfortunately, coupled with the fact that generating keys is considerably affected by many different physical conditions, such as fiber loss and noise, the key generation rate is very low compared with the consumption rate. To bridge this challenge, one implements key stores (KSs) to store secret keys in advance [30]. Each QKD node possesses a corresponding key store. Nevertheless, one node is able to share keys with multiple nodes, it is difficult to synchronously manage keys for applications between two nodes; hence, another server, called quantum key pool (QKP), is realized to manage keys in the key stores of two nodes. In other words, there is a corresponding QKP implemented for each pair of nodes. A QKP monitors the remaining number of secret keys in the corresponding key stores and sends the SDN controller the request for recharging keys if necessary.

Key Distribution Capability: In this paper, we consider the key distribution capability of the QKD network in the relationship between the consumption rate and the remaining number of keys in key stores. We suppose that the consumption rate is kept constant within a specific interval; hence the remaining keys are sufficient to maintain the operation of applications within several time units. Specifically, if the remaining number of keys in servers is κ (keys) and the consumption rate is ρ (keys/time slot), the application can keep running within $\frac{\kappa}{\rho}$ time slots. The key distribution capability is to leverage the available resources to deliver as many keys as possible and maximize the minimum number of remaining time slots of requests after recharging keys.

III. System Model and Research Problem

A. QKD-Over-WDM Network Architecture

QKD over WDM. In this study, we consider the model of Key as a Service (KaaS) [31], in which secret keys are delivered over a QKD network deployed over the wavelength-division multiplexing (WDM) network infrastructure as illustrated in Fig. 2. This network is expressed by $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ and $\mathcal{L}$ are the sets of QKD nodes and QKD links, respectively. The BB84 protocol is employed for generating secret keys between two adjacent nodes, and two remote nodes share secret keys by the key relay process. Therefore, in addition to equipping components used for the BB84 protocol, each QKD node possesses a limited key storage employed to store its shared secret keys. The capacity of the key storage

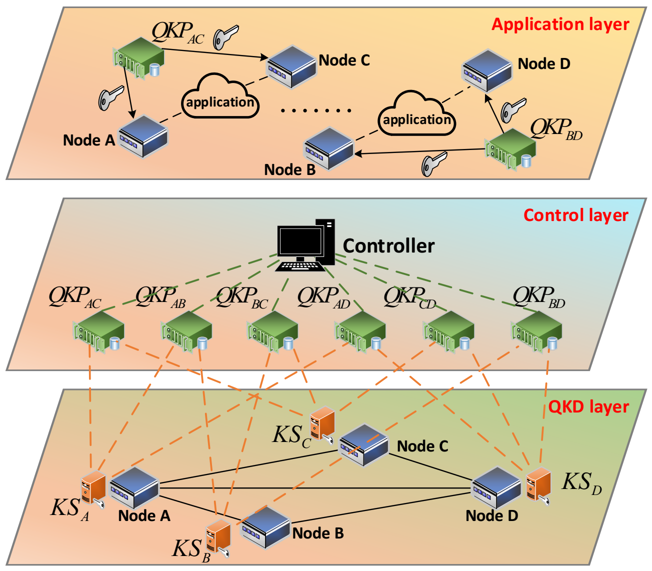

Fig. 2. An illustration of the system model.

of node $n \in \mathcal{N}$ is expressed over its c property, namely $n.c$. Because the wavelength is a limited resource of optical fibers used for many applications, each QKD link possesses a limited number of wavelengths, called quantum channels, employed for transmitting photons according to the BB84 protocol. The number of quantum channels per QKD link $(u, v) \in \mathcal{L}$, which connects QKD nodes u and v, is expressed by its c property, namely $(u, v).c$. In addition, because both the transmitting duration between two nodes and the survival capability of a photon are different depending on many factors, such as the fiber length and the photon loss along a fiber, quantum channels on different QKD links own disparate *key generation rates* (or key rates for short) measured by $kbps$. Moreover, we suppose that quantum channels on the same QKD link have the same key rate, also called the key rate of the QKD link. This is reasonable because such quantum channels have the same physical conditions. Moreover, we also suppose that secret keys used for security applications between two nodes in the network have the same length (measured by the number of bits). Thereby, the key rate of a QKD link can be expressed by the number of keys that can be generated per time slot. The key rate of QKD link (u, v) can be indicated over the k property of link (u, v), namely $(u, v).k$.

Key store (KS). Security applications need secret keys for their operation. The lack of secret keys will disrupt the applications. Nevertheless, coupled with the fact that the key generation rate is substantially lower than the data rate of applications, secret keys will be generated and stored in key stores (KSs) in advance to guarantee the continuity of the applications. For each QKD node, there is a corresponding KS used to store the produced keys. For each pair of nodes in the network, there is a corresponding quantum key pool (QKP), which manages secret keys in the KSs employed for security applications running between the two nodes. Eventually, because a QKP manages secret keys, it can know exactly the number of available keys remaining in the KSs and evaluate the current key *consumption rate*.

Controller. The network controller manages all the components of the network, including QKD nodes and QKPs. The controller is responsible for allocating network resources for generating secret keys using the BB84 protocol. As soon as the number of secret keys in KSs diminishes due to the consumption of security applications, QKPs will send requests for recharging secret keys to the controller. We also assume that the controller only receives the requests synchronously according to the time slot, that is, the controller will collect the requests at the beginning of a time slot, and then it will assign network resources to serve the requests.

The set of requests collected by the controller in a time slot is described by $\mathcal{R} = \{R_1, R_2, ..., R_k\}$. A request, in addition to an index to determine the QKP, includes the residual number of secret keys in the KSs and the current key consumption rate of applications. Thereby, request R_i can be described by $R_i = (s_i, d_i, \kappa_i, \rho_i)$, where s_i and d_i are the two nodes that demand to distribute secret keys, called the source and destination of R_i without loss of generality; κ_i and ρ_i are the residual number of keys in the KSs and the consumption rate of applications running between s_i and d_i, respectively. A key is distributed over one dedicated path connecting source-destination [20]. Based on the information from the requests, the controller can estimate how long security applications between two nodes can continue, counted as the number of time slots, with the number of residual keys in the KSs.

B. Research Problem

The number of secret keys remaining in KSs is apparently crucial to sustaining the operation of applications. Nevertheless, it is challenging to fully recharge keys for all requests due to the limitation of network resources. In this paper, we investigate the problem of maximizing key distribution capability (MKDC). Specifically, we aim to recharge secret keys such that maximizing the minimum number of time slots that applications can keep their availability. We formally state the research problem as follows.

INSTANCE: Given a QKD network described by $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ and a set of requests for recharging secret keys for KSs denoted by $\mathcal{R} = \{R_1, R_2, ..., R_k\}$. Each QKD node in $\mathcal{N}$ owns limited key storage. Each link in $\mathcal{L}$ possesses a limited key rate, counted as the number of keys that can be generated over the link by the BB84 protocol per time slot, and a specific number of quantum channels. The format of request R_i is $R_i = (s_i, d_i, \kappa_i, \rho_i)$, where $s_i, d_i \in \mathcal{N}$ are the two nodes that need to be distributed keys, κ_i is the number of residual keys in the corresponding KSs, and ρ_i is the consumption rate of applications running between the two QKD nodes.

TASK: Maximize the number of keys distributed over the network and the minimum number of time slots that KSs can supply secret keys to applications running on the network with the current consumption rate after recharging keys.

Motivating example: Consider an example with the QKD network shown in Fig. 3(a). For this example, we suppose that the number of memory units in each node is unlimited, and there is only one quantum channel on each link. The number located next to each link indicates the key rate of the corresponding QKD link. There are four requests for recharging keys, described by $R_1 = (A, C, 1, 1)$, $R_2 = (C, E, 1, 1)$, and $R_3 = (B, D, 1, 1)$. In the first solution presented in Fig. 3(b), requests R_1, R_2, and R_3 are recharged 7, 5, and 1 keys, respectively, over paths (A, B, C), (C, D, E), and (B, D, E). The total number of distributed keys is 13. Nevertheless, the application running between nodes B and B is at high risk of being interrupted because after recharging, the remaining keys are only enough to run within two time slots. A better solution is presented in Fig. 3(c). For this solution, request R_3 is recharged 3 keys, in which 1 key is recharged over path (B, D, E) and 2 keys are recharged over path (B, C, D, E). With the residual resources, requests R_1 and R_2 are recharged 5 and 3 keys, respectively. Though the total number of distributed keys is decreased compared with the first solution, the threat of application interruption is greatly reduced. After recharging, the applications with the highest risk of disruption can keep running within four time slots.

IV. Algorithms

In this section, we recast the MKDC problem with a mixed integer linear programming (MILP) model. The detail of the process is described in Section IV-A. Though we can use a standard LP solver to address the program, it takes a long solving time for larger instances. Therefore, we then propose two near-optimal algorithms to solve the MKDC. One employs the linear programming relaxation technique, called linear programming relaxation and rounding algorithm (LPR-RA), and the other is a greedy-based algorithm considering the resource utilization of requests, called progressive serving algorithm (PSA). The LPR-RA and PSA are presented in Section IV-B and Section IV-C, respectively.

A. Mixed Integer Linear Programming (MILP)

The first step for distributing secret keys between two nodes in the network is to determine paths connecting the two nodes. Keys will then be generated between two adjacent nodes along the path, and finally, the secret keys will be distributed by the key relay process. In this paper, we consider a key that can be delivered between the source and the destination as a flow over the path. The flow from the source to the destination of request R_i passing through directed link $\langle u, v\rangle$ is denoted by $f^i_{\langle u,v\rangle}$. Moreover, the total flow of request R_i is described by f^i, which is also the total number of keys distributed between s_i and d_i. The problem is formulated as follows.

$$\text{Maximize} \quad \beta \times \mu + (1-\beta) \times \sum_{R_i \in \mathcal{R}} f^i \tag{1}$$

subject to:

$$\sum_{v \in \mathcal{N}} f^i_{\langle s_i,v\rangle} - \sum_{v \in \mathcal{N}} f^i_{\langle v,s_i\rangle} = f^i \quad \forall R_i \in \mathcal{R} \tag{2}$$

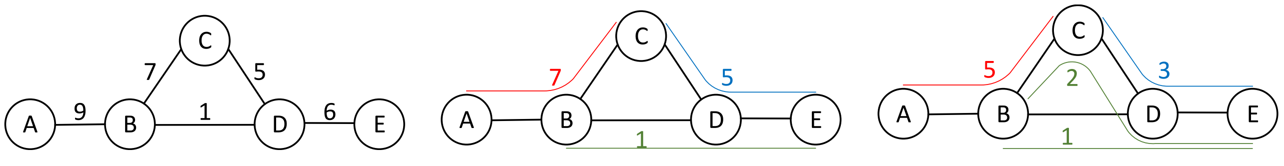

Fig. 3. A motivating example: a) The original QKD network, b) the first solution, and c) a better solution.

$$\sum_{v\in\mathcal{N}} f^i_{\langle u,v\rangle} - \sum_{v\in\mathcal{N}} f^i_{\langle v,u\rangle} = 0 \quad \forall R_i \in \mathcal{R} \text{ and } \forall u \in \mathcal{N} - \{s_i, d_i\} \tag{3}$$

$$\sum_{R_i\in\mathcal{R}} f^i_{\langle u,v\rangle} + \sum_{R_i\in\mathcal{R}} f^i_{\langle v,u\rangle} \le (u,v).c \times (u,v).k \quad \forall (u,v) \in \mathcal{L} \tag{4}$$

$$\sum_{R_i\in\mathcal{R}} f^i_{\langle u,v\rangle} + \sum_{R_i\in\mathcal{R}} f^i_{\langle v,u\rangle} \le u.c \quad \forall u \in \mathcal{N} \tag{5}$$

$$\frac{f^i + \kappa_i}{\rho_i} \ge \mu \quad \forall R_i \in \mathcal{R} \tag{6}$$

$$f^i_{\langle u,v\rangle}, f^i_{\langle v,u\rangle}, f^i \in \mathbb{Z}, f^i_{\langle u,v\rangle}, f^i_{\langle v,u\rangle}, f^i, \mu \ge 0 \quad \forall (u,v) \in \mathcal{L}, \forall R_i \in \mathcal{R} \tag{7}$$

The research problem is a joint optimization problem; hence, the objective function involves two goals normalized by β ($0 \le \beta \le 1$). With a larger β, the minimum number of remaining time slots (μ) is optimized with a higher priority. Constraint (2) indicates the total flow and also the total key that can be distributed between the source and the destination of a request. Constraint (3) shows the conservation of flows passing through an intermediate node, that is, the total inflow at a node equals to the total outflow. Inequality (4) is the constraint on the capacity of a link, that is, the total number of keys that can be generated on a link within a time slot must not exceed the link capacity. Constraint (5) implies that the key memory employed for the key relay process at a node must not surpass the node storage. In constraint (6), μ is the minimum number of time slots that a KS can provide secret keys to maintain the availability of security applications with the current consumption rate. Eventually, the number of keys generated over a link must be a positive integer, while μ can be a positive real number. By means of solving the above program using a standard solver, we can obtain the optimal solution to the problem with the expectation that both the minimum number of remaining time slots and the total key that can be distributed over the network achieve the maximum. Nevertheless, the running time will grow exponentially when the scale of the problem is increased. Therefore, we next propose two algorithms for addressing the problem in polynomial time. The first one is implemented on top of the linear programming relaxation (LPR), and the other is a greedy-based algorithm in which the request with the minimum number of remaining time slots will be served first.

B. Linear Programming Relaxation and Rounding Algorithm

In the following, we describe the key steps of the Linear Programming Relaxation and Rounding Algorithm (LPR-RA).

- **Relaxing the mixed integer linear programming.** We first relax the mixed integer linear program by replacing constraint (7) with constraint (8). Accordingly, the original MILP is transformed into a linear program (LP), which can be addressed within polynomial time. Though the solutions obtained from the linear program are infeasible, they supply a helpful reference to be able to achieve a good feasible solution. In this paper, we propose rounding a fractional solution down to an integer to make it feasible in terms of the integrality of the flow variables; hence we call this method the linear programming relaxation and rounding algorithm (LPR-RA). However, how to round and choose integer solutions to guarantee the constraints of the problem? Algorithm 1 describes this technique in more detail.

$$f^i_{\langle u,v\rangle}, f^i_{\langle v,u\rangle}, f^i, \mu \ge 0 \quad \forall (u,v) \in \mathcal{L}, \forall R_i \in \mathcal{R} \tag{8}$$

- **Solving the linear program and infeasible links.** The major procedures of the algorithm are solving the relaxed program and rounding the solutions. These procedures are repeated continuously until the result can not be improved. Accordingly, we first solve the LP derived from relaxing the MILP (Line 4). Because the solutions obtained from the LP still satisfy the constraints of a flow network, there are fractional flows from the sources to the corresponding destinations of the requests after solving the LP. We then consider the flows whose values are greater than or equal to 1 for each request because rounding down a less than 1 value returns 0, which inherently is not meaningful. It is worth noting that after this step, the remaining links form a flow network derived from the source and terminated at the destination, where all flows are greater than or equal to 1. Because there may be multiple flows traversing a link and almost flows are fractional, we cannot know the total flow passing the network. Therefore, we separate the flows by determining all possible paths connecting the source and the destination. The flow passing through each path is determined by rounding down the minimum flow traversing the links along the path to the nearest integer.
- **Determining feasible paths and corresponding flows.** After ascertaining a path, the flows crossing over the links along the path will be updated by reducing by the path flow (Lines 8-12). The total flow is finally aggregated

Algorithm 1: Linear programming relaxation and rounding algorithm (LPR-RA)

```
Input: QKD network $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, set of requests
       $\mathcal{R} = \{R_1, R_2, ..., R_k\}$
Output: The maximum of the minimum number of
        remaining time slots
1  $\mu = min\{\kappa_i/\rho_i | R_i \in \mathcal{R}\}$;
2  $total\_keys = 0$;
3  while True do
4      Solve the LP constructed based on the current
        status of the network and requests, say the
        solutions are $\overline{f^i_{\langle u,v\rangle}}$;
5      foreach $R_i \in \mathcal{R}$ do
6          $f^i = 0$;
7          Remove links with flows $\overline{f^i_{\langle u,v\rangle}} < 1$ from the
            solution of $R_i$;
8          while there exists a path $P_i$ connecting $s_i$ to $d_i$
            do
9              $temp = min\{\overline{f^i_{\langle u,v\rangle}} | \langle u,v\rangle \in P_i\}$;
10             foreach $\langle u,v\rangle \in P_i$ do
11                 $\overline{f^i_{\langle u,v\rangle}} = \overline{f^i_{\langle u,v\rangle}} - \lfloor temp \rfloor$;
12                 Remove $\langle u,v\rangle$ if $\overline{f^i_{\langle u,v\rangle}} < 1$;
13             $f^i = f^i + \lfloor temp \rfloor$;
14         $\kappa_i = \kappa_i + f^i$;
15         $\mu_i = \frac{\kappa_i}{\rho_i}$;
16         $total\_keys = total\_keys + f^i$;
17     $min\_\mu = min\{\mu_i | R_i \in \mathcal{R}\}$;
18     if $min\_\mu > \mu$ then
19         $\mu = min\_\mu$;
20     if $\mu$ and $total\_keys$ can not be improved then
21         break;
22     Update the capacity of links and nodes;
23     Update the number of remaining keys of requests;
24 return $\mu$ and $total\_keys$
```

Algorithm 2: progressive serving Algorithm (PSA)

```
Input: QKD network $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, set of requests
       $\mathcal{R} = \{R_1, R_2, ..., R_k\}$
Output: The maximum of the minimum number of
        remaining time slots
1  $flag = False$;
2  $total\_keys = 0$;
3  $\mathcal{R}' \leftarrow \emptyset$;
4  while $|\mathcal{R}| > 0$ do
5      $min\_\mu = min\{\kappa_i/\rho_i | R_i \in \mathcal{R}\}$;
6      $\mathcal{P} \leftarrow \emptyset$;
7      Determine the set of requests with the $min\_\mu$
        remaining time slots, say $\mathcal{R}_m$;
8      foreach $R_i \in \mathcal{R}_m$ do
9          if the residual resources of the network cannot
            serve $R_i$ then
10             $\mathcal{R} \leftarrow \mathcal{R} \backslash R_i$;
11             $\mathcal{R}' \leftarrow \mathcal{R}' \cup R_i$;
12             continue;
13         $P_i \leftarrow shortest\_path(R_i)$;
14         if $\mathcal{P} = \emptyset$ or $|P_i| < |\mathcal{P}|$ then
15             $\mathcal{P} \leftarrow P_i$;
16     Update $total\_keys$ if necessary;
17     Update the node and link resources along path $\mathcal{P}$;
18     Update the number of remaining keys for the
        corresponding request with path $\mathcal{P}$;
19 $\mu = min\{\kappa_i/\rho_i | R_i \in \mathcal{R}'\}$;
20 return $\mu$ and $total\_keys$
```

from the flows determined above (Line 13). Afterward, the remaining time slots of the current request (μ_i) will be identified according to the total flow, the current number of the remaining keys, and the consumption rate (Line 15). The total key distributed over the network is also updated based on the total flow of the current request (Line 16). If the minimum number of remaining time slots and the total distributed key, which is determined after considering all the requests, can be improved compared to the previous ones, we will update the capacity of links and nodes, and the number of remaining keys of the requests (Lines 22-23) and repeat the above steps. Otherwise, we terminate the loop and take the current number of remaining time slots and the current total distributed key ($total\_keys$) as the final results.

C. Progressive Serving Algorithm (PSA)

In the following, we put forth an efficient heuristic algorithm inspired by the fact that the request with the minimum number of remaining time slots needs to be satisfied first in order to achieve the objective of the problem. The detail of the algorithm is described in Algorithm 2.

In Algorithm 2, we first determine set ($\mathcal{R}_m$) of requests with the same number of remaining time slots (i.e., the ratio κ_i/ρ_i for request $R_i \in \mathcal{R}$). We next consider whether we can serve each request in $\mathcal{R}_m$ with the available resources of the network. A request is served if we can discover a path connecting its source and destination on which there are enough available resources. A node is available when it owns at least two available memory units if it is an intermediate node and one memory unit if it is a source or a destination. Similarly, a link is available if it possesses at least one available quantum channel. We then select the first request with the shortest path connecting its source and destination as the request that needs to be served. The rationale behind this step is derived from the fact that serving a request with the shortest path will consume the least network resources, hence increasing the chance to serve other requests. The total distributed keys will then be updated by increasing 1 unit.

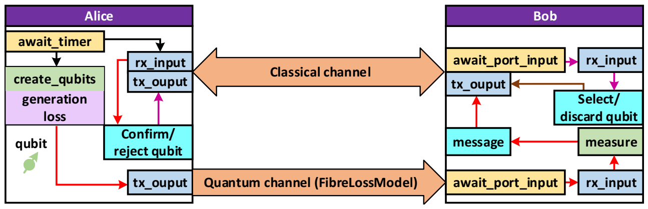

Fig. 4. NetSquid implementation.

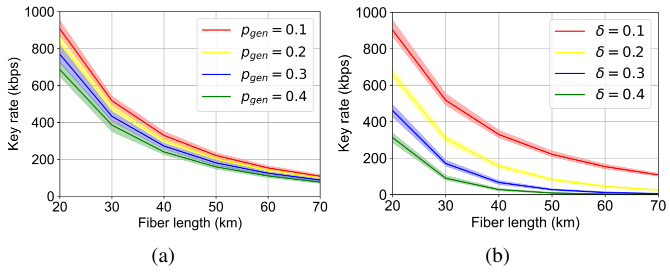

Fig. 5. Dependence of the key generation rate on the fiber length (a) the probability that the photons are lost immediately after generation (b) fiber attenuation.

Moreover, the resources of nodes and links along the selected path will then be updated; that is, the total capacity of each link will be decreased by 1, and the number of available memory units will be reduced by 2 for the intermediate nodes and decreased by 1 for the source and destination nodes. If a request cannot be served with the residual resources of the network, it will be moved from the original set ($\mathcal{R}$) to the final set ($\mathcal{R}'$) containing requests that cannot be served anymore. The algorithm will iterate until the residual resources are insufficient to serve any requests.

V. Key rate benchmark on NetSquid

In this section, we conduct experiments on the BB84 protocol using NetSquid [25], an open simulator for quantum networks. We realize the protocol by the model illustrated in Fig. 4. In the model, Alice and Bob are two adjacent QKD nodes that need to share secret keys. These nodes are connected directly over two separate channels with the same length, classical and quantum. Qubits, expressed by photons, will propagate from Alice to Bob over the quantum channel while the information is exchanged over the classical channel. In addition, qubits conveyed over the quantum channel must suffer a loss with probability p_{loss} determined as follows [32].

$$p_{loss} = 1 - (1 - p_{gen}) \times 10^{-\delta\ell/10} \qquad (9)$$

where p_{gen} is the probability that the photons are lost immediately after generation because the apparatus used in experiments in reality is not ideal, δ (dB/km) is the attenuation parameter of the optical fiber employed as the quantum channel, and ℓ is the fiber length. Both the channels introduce a propagation delay determined as $\Delta = \frac{\ell}{c_f}$, where c_f is the speed of light in the fiber, which is set to 2×10^5 (km/s) in the experiments. Moreover, we also suppose that there is no error on the classical channel. In all the experiments, we evaluate the key generation rate, counted as $kbps$, according to the fiber length, which is varied from 20 to 70 (km). The experiment results are shown in Fig. 5, in which data employed to draw curves (bold color) is the average of 100 trials. The band covering a curve reflects the variation in the results of the corresponding case.

In the first experiment, we alter p_{gen} between the values in set $\{0.1, 0.2, 0.3, 0.4\}$ and fix δ at 0.1. In the second experiment, the fiber attenuation (δ) is changed from 0.1 to 0.4 while the initial probability (p_{gen}) is kept constant at 0.1. The results of these experiments are displayed in Fig. 5(a) and Fig. 5(b), respectively. It is worth noticing that the red curves in both figures are nearly analogous with the other because the parameters used for these cases are the same, that is, $p_{gen} = 0.1$ and $\delta = 0.1$. The main observation from the figures is that the key rate is quite low and substantially decreases according to the fiber length, particularly in the second experiment. In addition, the larger the initial probability or the fiber attenuation, the lower the key rate. There are three significant reasons engendering this result. The first reason is associated with the delay of photons and exchanged information between Alice and Bob once propagating over the classical and quantum channels. The longer the distance, the longer the latency, which decreases the number of bits generated within a time unit. Secondly, some photons will be lost during generation and transmission over the fiber. In particular, the loss will be more serious with the larger values of p_{gen}, δ, and ℓ according to (9). For instance, with the case that $p_{gen} = 0.1$, $\delta = 0.4$, and $\ell = 50$ in the second experiment, the probability of photon loss once traversing over the quantum channel is $p_{loss} = 0.991$, which means all the generated photons is nearly lost during transmission. This induces the green curve in Fig. 5(b) to approach 0 when the fiber length exceeds 50 (km). Moreover, also according to (9), δ and ℓ offer more influence on p_{loss} than p_{gen}, which causes the key rate to reduce strongly once increasing δ and ℓ. This is the reason why each curve in Fig. 5(b) is separated from the others, different from Fig. 5(a), where all the curves approach closely. Eventually, because Bob randomly employs one out of two bases (i.e., rectilinear and diagonal) to measure photons, half of the photons that can reach Bob will be ignored due to the mismatch between the bases used by Alice and Bob.

VI. Performance Evaluation

In this section, we conduct extensive simulations in order to evaluate the performance of the proposed algorithms, i.e., LPR-RA and PSA, as well as factors that affect the objective of the problem.

A. Simulation Setting

In this subsection, we present the default values of parameters employed in simulations; that is, if these parameters are not set in simulation scenari os, they will get the corresponding default values. Firstly, the networks used in the simulations are randomly generated employing the Erdos-Renyi model [33],

TABLE I
SUMMARY OF PARAMETERS IN THE SIMULATION

Parameter	Description
Network	Erdos-Renyi model
Number of nodes	100
Probability for link creation (α)	0.05
The number of quantum channels on links	$[1, 10)$ (channels/link)
The key rate of quantum channels	$[1, 5)$ (keys/time slot)
The number of memory units of nodes	$[10, 60)$ (units)
The number of requests	20
β	0.99

in which the number of nodes and the probability for link creation (α) are set to 100 and 0.05, respectively. The number of quantum channels on network links is distributed between the values in the half-open interval $[1, 10)$. The key generation rate of quantum channels on the same link, counted as the number of keys that can be generated per time slot, is identical and distributed between the values in the half-open interval $[1, 5)$. Each node hosts a specific number of memory units for key storage, distributed between the values in the half-open interval $[10, 60)$. The number of quantum channels, the key rate of quantum channels, and the number of memory units are all randomly set according to the discrete uniform distribution. Secondly, the set of requests includes 20 elements by default and is also randomly generated. In addition, we intend to optimize μ with a higher priority; thus, we set β to 0.99. To mitigate errors engendered by randomly generating parameters, each point employed to draw the curves in figures is the average of 100 trials[1]. We summarize the parameters used in the simulation in Table I.

Because the problem in the paper is first proposed, we employ two baselines for performance evaluation. The baseline used for the small-scale problem is the solution obtained from solving the mixed integer linear programming (MILP) model directly. Because the MKDC is a joint optimization problem involving two objectives, this baseline may not be the optimal solution, but it is a good feasible solution for comparison. Particularly, in the case of assessing the application sustaining capacity, it provides the best solution amongst the proposed methods. For the large-scale problem and other scenarios, we use the solution obtained by solving the relaxed version of the MILP model (LPR) as the baseline. Although this approach cannot yield a feasible solution due to the violation of the integrality constraint, it can be used as an upper-bound to evaluate the performance of the LPR-RA and PSA.

The performance metric used to evaluate the proposed algorithms in simulation scenarios is the minimum number of time slots that security applications running on the network can be maintained. An application is maintained if the number of secret keys in the corresponding KSs is enough to keep operating within a specific number of time slots with the current consumption rate. We call the maximum of the minimum number of time slots that an application can be maintained the

[1]The source code is released at: https://github.com/NextCNS/MaxMinProb.

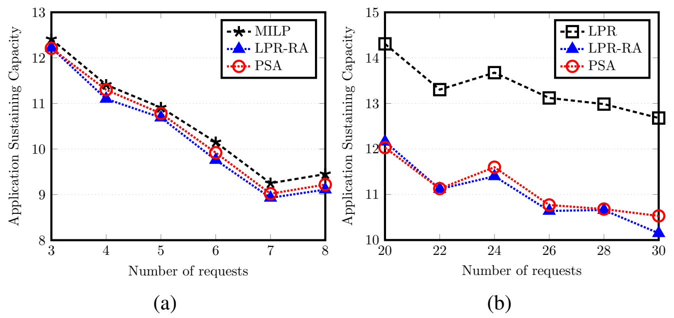

Fig. 6. Dependence of the application sustaining capacity on the number of requests (a) small-scale problem and (b) large-scale problem.

application sustaining capacity of the network. In addition, we employ the Jain index [34] to evaluate the fairness in terms the number of remaining time slots that applications can keep running in. Finally, we also assess the performance of the proposed algorithm over the total secret keys that can be distributed over the network.

B. Simulation Results

The main observation form the results: the performance of the LPR-RA and PSA nearly approaches to the other in Fig. 6. Both algorithms show good performance in the small-scale scenario when their solutions are close to the best solution obtained by directly solving the MILP. In Fig. 7 and Fig. 8, the LPR-RA shows a better performance once increasing the network scale (i.e., the number of QKD nodes, links, the number of channels per QKD link, and the channel capacity).

1) Impact of the scale of the problem: For this set of simulations, we assess the impact of the number of requests in set $\mathcal{R}$ on the application sustaining capacity. We divide the simulation into two scenarios, equivalent to the small-scale and large-scale of the problem. In the small-scale problem, the number of requests varies from 3 to 8, and the number of QKD nodes is fixed to 30. In the large-scale problem, we evaluate the application sustaining capacity when the number of requests alters between the values in set $\{20, 22, 24, 26, 28, 30\}$, and the number of QKD nodes is set to 100. As we observe in Fig. 6, when the number of requests increases, the primary trend is steadily down, but the drop pace is lower in the large-scale problem. This is because the network resources, i.e., quantum channels and key memories, are consumed more to be able to increase the minimum number of remaining time slots for more requests. This reduces the chance of improving the number of remaining time slots of other requests due to the limitation of the network resources, causing a decrease in the application sustaining capacity.

2) Impact of the network: In the second simulation, we evaluate the dependence of the application sustaining capacity on the network components, that is, nodes and links. In the first scenario, we vary the number of QKD nodes from 50 to 300 while keeping the values of other parameters at the default. The result of the simulation is recorded and presented in Fig. 7(a). In the figure, we can observe that when the

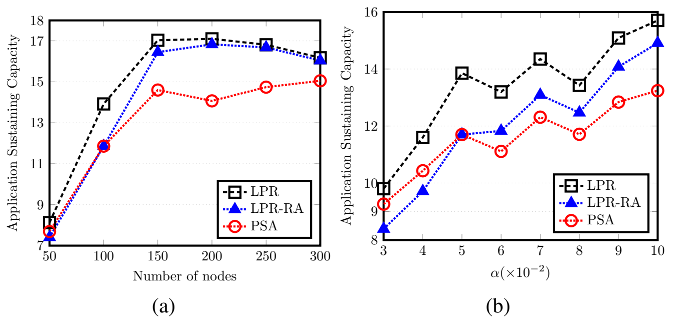

Fig. 7. Dependence of the application sustaining capacity on the network components (a) network nodes (b) network links.

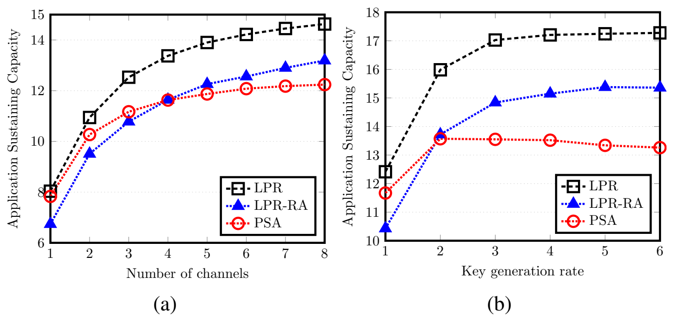

Fig. 8. Dependence of the application sustaining capacity on quantum channels (a) the number of quantum channels on each link (b) the key generation of quantum channels.

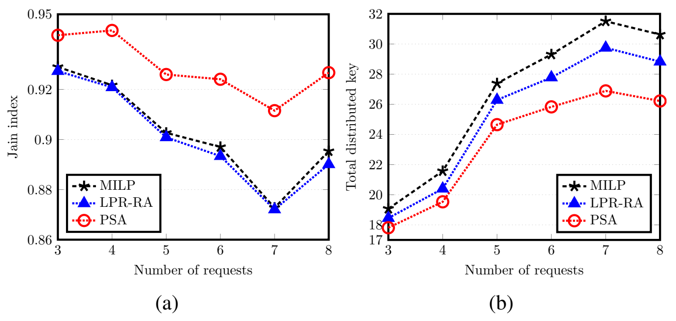

Fig. 9. Dependence of the a) fairness index and b) total distributed key on the number of requests.

number of nodes is increased from 50 to 150, the application sustaining capacity is also improved significantly. However, when the number of nodes exceeds 150, the curves tend to remain unchanged or decrease slightly. This is because there are many ways to establish paths connecting the sources and destinations of the same request used for the QKD relay when increasing the number of nodes. This leads to an increase in the possibility of serving requests. Nevertheless, one of the factors confining the capacity of the network is the density of links (specified by α), which is kept constant in this scenario. It is worth noticing that as long as we cannot determine a path to satisfy one request among the requests that need to be served, i.e., the requests with the same minimum number of remaining time slots, the application sustaining capacity cannot be improved. That is the reason the result cannot be enhanced though the number of nodes is increased.

In the second scenario, we alter the network scale by changing the probability of generating links (α) from 0.03 to 0.1 while keeping the number of nodes. As shown in Fig. 7(b), though the curves fluctuate unpredictably, the major trend for all algorithms is upward. Similarly to the above scenario, the increase in the number of links will render increasing in the potential for satisfying requests. Nonetheless, the bottleneck confining the application sustaining capacity in this scenario is the number of memory units hosted by network nodes. When the number of memory units is not enough, we also cannot establish paths for the key relay process though there are still many redundant links.

3) Impact of quantum channels: In order to evaluate the impact of quantum channels, we conduct simulations in two scenarios. In the first scenario, the number of quantum channels on all links is identical and changes from 1 to 8. The simulation result is illustrated in Fig. 8(a). All algorithms yield the same results in terms of the tendency of the curves. Generally, the application sustaining capacity increases continuously according to the number of quantum channels. Nevertheless, the growth pace gradually decreases when increasing the number of channels. This is because the increase in the number of quantum channels will lead to an increase in the possibility of key generation on each link, but the key storage of nodes is the bottleneck in this scenario that limits the number of generated keys on links. Therefore, it is hard to improve the application sustaining capacity even when the number of quantum channels is set to a relatively large value.

In the second scenario, we explore the influence of quantum channels on the application sustaining capacity by changing the channel capacity, counted as the number of keys a channel can generate per time slot, from 1 to 6. The result is shown in Fig. 8(b). Similarly to the above scenario, the increase in channel capacity will result in an increase in link capacity, but due to the limitation of key storage of nodes, we cannot improve the application sustaining capacity anymore when the channel capacity gains a threshold. As observed from Fig. 8(b), the thresholds are 3 and 2 for the LPR-RA and PSA, respectively.

4) The fairness and total distributed key: We employ the same configuration as the small-scale scenario for this set of simulations. From the results shown in Fig. 9, we can observe that there is a trade-off between fairness and the total key that can be distributed over the network. Specifically, the PSA provides the best fairness but distributes the fewest keys. This correctly reflects the essence of the algorithms, that is, the PSA, by means of using the progressive technique for all requests, always attempts to maintain the best fairness, while the MILP and LPR-RA, after maximizing the minimum number of remaining time slots among the requests, tend to optimize the total number of distributed keys. From Fig. 9(a), the fairness index tends to decrease once increasing the number of requests due to two reasons. The first one is the limitation of the network resources. The second is that resource contention increases with the increase in the number of requests. In addition, from Fig. 9(b), the total key distributed over the network increases according to the number of requests. Nevertheless, if there are more chances to improve the minimum number of remaining time slots, the total distributed key tends to decrease. This is clearly shown in Fig. 6(a) and Fig. 9 when the number of requests varies from 7 to 8. While the total distributed key decreases, the application sustaining capacity and the Jain index increase.

VII. Conclusion

This paper considers QKD deployed over the WDM networks. We have investigated the *first-of-its-kind* maximizing key distribution capability (MKDC) problem considering the nexus between key generation rate and consumption rate. Specifically, our goal is to sustain the operation of the applications as long as possible by provisioning sufficient secret keys and simultaneously maximize the number of keys that can be distributed over the network within a time slot. We have formulated the problem with a mixed integer linear programming model. We then proposed two near-optimal algorithms, including LPR-RA and PSA to address the MKDC in polynomial time. To demonstrate the feasibility of the proposed model and algorithms, we have realized the system on a quantum network simulator i.e. NetSquid considering different topologies and traffic matrices while testing the key rate and how the quantum loss affects the key generation rate, particularly the fiber loss. In addition, comprehensive simulations have been conducted to validate the efficacy of LPR-RA and PSA. The results indicate that both algorithms can approach the best results in all the simulation scenarios.

Appendix

In this section, we show that the LPR-RA can satisfy all the constraints of the MKDC problem. In addition, we also analyze the time complexity of the algorithms. In the theorems and proofs, $|.|$ indicates the cardinality of the set, and lg denotes the decimal logarithm of a number.

Theorem 1. *Solutions obtained from the LPR-RA satisfy all the constraints of the MKDC problem.*

Proof: The constraints of the MKDC problem include two groups. The first group is related to the flow network, used to determine paths connecting two nodes that need to share secret keys. The second group is used to restrict the path in the limitation of the network resources (i.e., the key storage of nodes and the quantum channels on links). The solution obtained from the relaxed version of the mixed integer linear programming model, though fractional, always forms a flow network for each request. The LPR-RA determines the shared key paths based on such networks; hence satisfy the flow network constraints. For each path, the flow is determined by rounding the minimum among flows on links along the path down to the nearest integer. This guarantees that the resources used for distributing keys along the paths do not exceed the available resources, thus satisfying the second group of constraints. This completes the proof. ■

Theorem 2. *The LPR-RA can terminate in polynomial time.*

Proof: The LPR-RA includes two portions, solving the linear program and rounding the results. The first one can be achieved in polynomial time [35]; therefore, it suffices to show that the second part can run in polynomial time and the **while** loop between Lines 3 and 23 in Algorithm 1 can terminate. After addressing the relaxed version of the mixed integer linear programming model, we consider each request in set $\mathcal{R}$ in turn; thus, the loop between Lines 5 and 16 iterates at most $O(|\mathcal{R}|)$ times. We next remove the links whose flow is less than 1 (Line 7), which takes $O(|\mathcal{L}|)$ time. The **while** loop between Lines 8 and 13 repeats at most $O(\theta)$ times, where $\theta = \max\left\{\left\lfloor \overline{f^i} \right\rfloor |\forall R_i \in \mathcal{R}\right\}$, $\overline{f^i}$ is a solution of the linear program. In the LPR-RA, we employ the Dijkstra algorithm to find paths distributing keys, which requires at most $O(|\mathcal{N}|\lg|\mathcal{N}| + |\mathcal{L}|)$ [35]. The **for** loop between Lines 10 and 12 used to determine the flow of a path takes at most $O(|\mathcal{L}|)$ time. Hence, the **for** loop between Lines 5 and 16 takes $O\left(|\mathcal{R}|\left(|\mathcal{L}| + \theta\left(|\mathcal{N}|\lg|\mathcal{N}| + |\mathcal{L}| + |\mathcal{L}|\right)\right)\right) = O\left(|\mathcal{R}|\,\theta\left(|\mathcal{N}|\lg|\mathcal{N}| + |\mathcal{L}|\right)\right)$ time. In addition, it takes $O(|\mathcal{R}|\lg|\mathcal{R}|)$ time to determine the request with the minimum number of remaining time slots (Line 17). Updating the capacity of links and nodes (Line 22) needs $O(|\mathcal{N}|+|\mathcal{L}|)$ time. Updating the requests (Line 23) takes $O(|\mathcal{R}|)$ time. Therefore, the total running of the rounding step of the LPR-RA is $O\left(|\mathcal{R}|\,\theta\left(|\mathcal{N}|\lg|\mathcal{N}| + |\mathcal{L}|\right) + |\mathcal{R}|\lg|\mathcal{R}| + |\mathcal{L}| + |\mathcal{N}| + |\mathcal{R}|\right) = O\left(|\mathcal{R}|\left(\theta\left(|\mathcal{N}|\lg|\mathcal{N}| + |\mathcal{L}|\right) + \lg|\mathcal{R}|\right)\right)$. This indicates that the rounding stage can run in polynomial time. Moreover, the network resources are gradually depleted after each loop, which renders the flows in the linear program approach 0. Finally, the outer **while** loop is terminated once the condition in Line 20 is satisfied. This completes the proof. ■

Theorem 3. *The time complexity of the PSA is bounded in* $O(|\mathcal{R}|^2(\lg|\mathcal{R}| + |\mathcal{N}|\lg|\mathcal{N}| + |\mathcal{L}|))$.

Proof: In Algorithm 2, the loop between Lines 4 and 18 iterates at most $O(|\mathcal{R}|)$ times. For determining the minimum number of remaining time slots among the requests (Line 5), the time complexity requires $O(|\mathcal{R}|\lg|\mathcal{R}|)$. Moreover, it takes $O(|\mathcal{R}|)$ time to determine the set of requests with the minimum number of remaining time slots (Line 7). The **for** loop between Lines 8 and 15 repeats at most $O(|\mathcal{R}|)$ times. In addition, we employ the Dijkstra algorithm to determine the shortest path; therefore, the time complexity of Line 13 is $O(|\mathcal{N}|\lg|\mathcal{N}| + |\mathcal{L}|)$ [35]. Thus, the loop between Lines 8 and 15 takes $O(|\mathcal{R}|(|\mathcal{N}|\lg|\mathcal{N}| + |\mathcal{L}|))$ time. The updating operation at Line 17 needs to consider all the nodes and links in the worst case, hence taking $O(|\mathcal{N}| + |\mathcal{L}|)$ time. Eventually, the total running time of Algorithm 2 is $O(|\mathcal{R}|(|\mathcal{R}|\lg|\mathcal{R}| + |\mathcal{R}| + |\mathcal{R}|(|\mathcal{N}|\lg|\mathcal{N}| + |\mathcal{L}|) + |\mathcal{N}| + |\mathcal{L}|)) = O(|\mathcal{R}|^2(\lg|\mathcal{R}| + |\mathcal{N}|\lg|\mathcal{N}| + |\mathcal{L}|))$. This completes the proof. ■

References

[1] M. Mehic, M. Niemiec, S. Rass, J. Ma, M. Peev, A. Aguado, V. Martin, S. Schauer, A. Poppe, C. Pacher, and M. Voznak, "Quantum key distribution: A networking perspective," *ACM Comput. Surv.*, vol. 53, no. 5, sep 2020.

[2] D. Nadlinger, P. Drmota, B. Nichol, G. Araneda, D. Main, R. Srinivas, D. Lucas, C. Ballance, K. Ivanov, E.-Z. Tan *et al.*, "Experimental quantum key distribution certified by Bell's theorem," *Nature*, vol. 607, no. 7920, pp. 682–686, 2022.

[3] M. Minder, M. Pittaluga, G. L. Roberts, M. Lucamarini, J. Dynes, Z. Yuan, and A. J. Shields, "Experimental quantum key distribution beyond the repeaterless secret key capacity," *Nature Photonics*, vol. 13, no. 5, pp. 334–338, 2019.

[4] E. Diamanti, H.-K. Lo, B. Qi, and Z. Yuan, "Practical challenges in quantum key distribution," *npj Quantum Information*, vol. 2, no. 1, pp. 1–12, 2016.

[5] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dušek, N. Lütkenhaus, and M. Peev, "The security of practical quantum key distribution," *Rev. Mod. Phys.*, vol. 81, pp. 1301–1350, Sep 2009. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.81.1301

[6] F. Xu, X. Ma, Q. Zhang, H.-K. Lo, and J.-W. Pan, "Secure quantum key distribution with realistic devices," *Rev. Mod. Phys.*, vol. 92, p. 025002, May 2020. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.92.025002

[7] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical Computer Science*, vol. 560, pp. 7–11, dec 2014.

[8] D. Stucki, N. Brunner, N. Gisin, V. Scarani, and H. Zbinden, "Fast and simple one-way quantum key distribution," *Applied Physics Letters*, vol. 87, no. 19, 11 2005, 194108. [Online]. Available: https://doi.org/10.1063/1.2126792

[9] F. Grosshans and P. Grangier, "Continuous variable quantum cryptography using coherent states," *Phys. Rev. Lett.*, vol. 88, p. 057902, Jan 2002. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.88.057902

[10] B. Qi, W. Zhu, L. Qian, and H.-K. Lo, "Feasibility of quantum key distribution through a dense wavelength division multiplexing network," *New Journal of Physics*, vol. 12, no. 10, p. 103042, oct 2010. [Online]. Available: https://dx.doi.org/10.1088/1367-2630/12/10/103042

[11] K. A. Patel, J. F. Dynes, M. Lucamarini, I. Choi, A. W. Sharpe, Z. L. Yuan, R. V. Penty, and A. J. Shields, "Quantum key distribution for 10 Gb/s dense wavelength division multiplexing networks," *Applied Physics Letters*, vol. 104, no. 5, 02 2014, 051123.

[12] L.-J. Wang, L.-K. Chen, L. Ju, M.-L. Xu, Y. Zhao, K. Chen, Z.-B. Chen, T.-Y. Chen, and J.-W. Pan, "Experimental multiplexing of quantum key distribution with classical optical communication," *Applied Physics Letters*, vol. 106, no. 8, 02 2015, 081108.

[13] N. A. Peters, P. Toliver, T. E. Chapuran, R. J. Runser, S. R. McNown, C. G. Peterson, D. Rosenberg, N. Dallmann, R. J. Hughes, K. P. McCabe, J. E. Nordholt, and K. T. Tyagi, "Dense wavelength multiplexing of 1550nm QKD with strong classical channels in reconfigurable networking environments," *New Journal of Physics*, vol. 11, no. 4, p. 045012, apr 2009.

[14] M. I. García Cid, L. Ortiz Martín, and V. Martín Ayuso, "Madrid quantum network: A first step to quantum internet," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, ser. ARES 21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3465481.3470056

[15] J. Dynes, A. Wonfor, W.-S. Tam, A. Sharpe, R. Takahashi, M. Lucamarini, A. Plews, Z. Yuan, A. Dixon, J. Cho *et al.*, "Cambridge quantum network," *npj Quantum Information*, vol. 5, no. 1, p. 101, 2019.

[16] Y. Cao, Y. Zhao, Q. Wang, J. Zhang, S. X. Ng, and L. Hanzo, "The evolution of quantum key distribution networks: On the road to the qinternet," *IEEE Communications Surveys Tutorials*, vol. 24, no. 2, pp. 839–894, 2022.

[17] Y. Cao, Y. Zhao, Y. Wu, X. Yu, and J. Zhang, "Time-scheduled quantum key distribution (QKD) over WDM networks," *Journal of Lightwave Technology*, vol. 36, no. 16, pp. 3382–3395, 2018.

[18] M. Dianati, R. Alléaume, M. Gagnaire, and X. S. Shen, "Architecture and protocols of the future european quantum key distribution network," *Security and Communication Networks*, vol. 1, no. 1, pp. 57–74, 2008. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.13

[19] M. Peev, C. Pacher, R. Alléaume, C. Barreiro, J. Bouda, W. Boxleitner, T. Debuisschert, E. Diamanti, M. Dianati, J. F. Dynes, S. Fasel, S. Fossier, M. Fürst, J.-D. Gautier, O. Gay, N. Gisin, P. Grangier, A. Happe, Y. Hasani, M. Hentschel, H. Hübel, G. Humer, T. Länger, M. Legré, R. Lieger, J. Lodewyck, T. Lorünser, N. Lütkenhaus, A. Marhold, T. Matyus, O. Maurhart, L. Monat, S. Nauerth, J.-B. Page, A. Poppe, E. Querasser, G. Ribordy, S. Robyr, L. Salvail, A. W. Sharpe, A. J. Shields, D. Stucki, M. Suda, C. Tamas, T. Themel, R. T. Thew, Y. Thoma, A. Treiber, P. Trinkler, R. Tualle-Brouri, F. Vannel, N. Walenta, H. Weier, H. Weinfurter, I. Wimberger, Z. L. Yuan, H. Zbinden, and A. Zeilinger, "The SECOQC quantum key distribution network in Vienna," *New Journal of Physics*, vol. 11, no. 7, p. 075001, jul 2009.

[20] H. Zhou, K. Lv, L. Huang, and X. Ma, "Quantum network: Security assessment and key management," *IEEE/ACM Transactions on Networking*, vol. 30, no. 3, pp. 1328–1339, 2022.

[21] Q.-C. Le, P. Bellot, and A. Demaille, "Towards the world-wide quantum network," in *Information Security Practice and Experience*, L. Chen, Y. Mu, and W. Susilo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 218–232.

[22] C. le Quoc, P. Bellot, and A. Demaille, "Stochastic routing in large grid-shaped quantum networks," in *2007 IEEE International Conference on Research, Innovation and Vision for the Future*, 2007, pp. 166–174.

[23] E. E. Moghaddam, H. Beyranvand, and J. A. Salehi, "Resource allocation in space division multiplexed elastic optical networks secured with quantum key distribution," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 9, pp. 2688–2700, 2021.

[24] X. Yu, S. Li, Y. Zhao, Y. Cao, A. Nag, and J. Zhang, "Routing, core and wavelength allocation in multi-core-fiber-based quantum-key-distribution-enabled optical networks," *IEEE Access*, vol. 9, pp. 99 842–99 852, 2021.

[25] T. Coopmans, R. Knegjens, A. Dahlberg, D. Maier, L. Nijsten, J. de Oliveira Filho, M. Papendrecht, J. Rabbie, F. Rozpedek, M. Skrzypczyk *et al.*, "Netsquid, a network simulator for quantum information using discrete events," *Communications Physics*, vol. 4, no. 1, p. 164, 2021.

[26] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[27] A. Aguado, V. Lopez, D. Lopez, M. Peev, A. Poppe, A. Pastor, J. Folgueira, and V. Martin, "The engineering of software-defined quantum key distribution networks," *IEEE Communications Magazine*, vol. 57, no. 7, pp. 20–26, 2019.

[28] L. Salvail, M. Peev, E. Diamanti, R. Alléaume, N. Lütkenhaus, and T. Länger, "Security of trusted repeater quantum key distribution networks," *Journal of Computer Security*, vol. 18, no. 1, pp. 61–87, 2010.

[29] C. Yang, H. Zhang, and J. Su, "The QKD network: model and routing scheme," *Journal of Modern Optics*, vol. 64, no. 21, pp. 2350–2362, 2017.

[30] R. Alléaume, F. Roueff, E. Diamanti, and N. Lütkenhaus, "Topological optimization of quantum key distribution networks," *New Journal of Physics*, vol. 11, no. 7, p. 075002, jul 2009. [Online]. Available: https://dx.doi.org/10.1088/1367-2630/11/7/075002

[31] Y. Cao, Y. Zhao, J. Wang, X. Yu, Z. Ma, and J. Zhang, "Kaas: Key as a service over quantum key distribution integrated optical networks," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 152–159, 2019.

[32] C. Cicconetti, M. Conti, and A. Passarella, "Request scheduling in quantum networks," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 2–17, 2021.

[33] P. Erdős, A. Rényi *et al.*, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[34] R. K. Jain, D.-M. W. Chiu, W. R. Hawe *et al.*, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, vol. 21, 1984.

[35] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.