Int. J. Appl. Math. Comput. Sci., 2025, Vol. 35, No. 4, 667–675
DOI: 10.61822/amcs-2025-0047

OPTIMIZING ROUTING IN QUANTUM KEY DISTRIBUTION NETWORKS USING THE ARTIFICIAL FISH SWARM ALGORITHM

WOJCIECH SZCZEPANIK [a,*], MARCIN NIEMIEC [a]

[a] Institute of Telecommunications
AGH University of Krakow
Mickiewicza 30, 30-059 Kraków, Poland
e-mail: wojciech.szczepanik@agh.edu.pl

Quantum key distribution offers another way, alongside public key cryptography, of establishing cryptographic keys. The usage of quantum mechanics provides a high level of security at the cost of key establishment efficiency. To help with that, we propose a new link cost function used for establishing best paths in order to minimize key consumption. We optimize its parameters using the artificial fish swarm algorithm. The findings demonstrate that this approach enhances the performance of the quantum key distribution network.

Keywords: cybersecurity, routing, evolutionary algorithms, quantum key distribution, artificial fish swarm algorithm.

1. Introduction

One of the cybersecurity tasks is the provision of secure information transfer. In today's Internet, communication between any two users should be encrypted (Ashok and Gopikrishnan, 2023; Kurek *et al.*, 2015; El-Douh *et al.*, 2022). That is mainly achieved by public key cryptography. The connection is first established using asymmetric algorithms. Because symmetric algorithms are much faster, an asymmetric one is used only to exchange confidential keys, which are then used by both ends to encrypt and decrypt communication. However, there is a considerable risk that the methodologies we rely on today will require significant and substantial modifications to continue to ensure the security of connections over public channels, mainly because of the development of quantum computers, which can break the security of asymmetric cryptography.

1.1. Cryptography. Public key cryptography uses a pair of keys: a public one that is well-known and a private one known only to its owner (Yilmaz and Öztürk, 2025). This scheme uses asymmetric algorithms such as Rivest–Shamir–Adleman (RSA) or elliptic curves. These are vulnerable to Shor's algorithm (Shor, 1997; Ivanov and Stoianov, 2023), which could be run on a quantum computer. As of now, quantum computing remains in its infancy, making such attacks impractical. However, there is no assurance that this will not happen in the near future.

In order to address this risk, two types of solutions have been proposed. The first one involves the implementation of asymmetric algorithms that would not be vulnerable to quantum computer attacks. One of such examples is the module-lattice-based key encapsulation mechanism (ML-KEM) (NIST, 2024), which is accepted by the US National Institute of Standards and Technology (NIST) as part of the novel post-quantum cryptography standards.

The second potential approach involves the usage of quantum mechanics for the establishment of symmetric keys. Quantum key distribution (QKD), when correctly implemented, provides security based on quantum mechanics instead of computational complexity. By transmitting information using quantum states, it is nearly impossible for the eavesdropper to catch it without detection by both communicating parties. This solution was first proposed by Bennett and Brassard (2014) with the introduction of the BB84 algorithm. Numerous alternative quantum key distribution algorithms were later put forward.

1.2. Quantum key distribution networks. The ITU-T framework (ITUTSS, 2019) offers a detailed

*Corresponding author

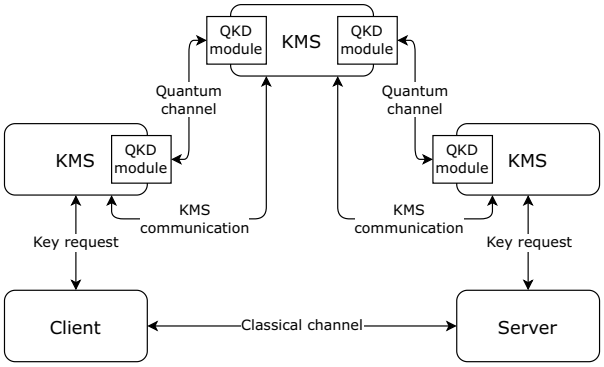

Fig. 1. Conceptual illustration of the QKD network architecture.

description of the architecture of networks that facilitate the distribution of quantum keys (Mehic *et al.*, 2020). Figure 1 shows the concept of a QKD network. This illustration represents only a segment of the complete network.

The key management system (KMS) is a trusted device capable of receiving encryption key requests from clients and servers using the classical channel. At present, available implementations demand that each of these devices be trusted because they store secret keys that are used for encryption. Each KMS is composed of one or more QKD modules. These are dedicated devices capable of sending and receiving quantum-encoded information. They set up a quantum channel which frequently takes the form of a fiber optic channel or uses free space transmission. Then a protocol such as BB84 (Bennett and Brassard, 2014) is used to generate identical binary sequences at both ends without the possibility of successful eavesdropping on this communication. The resulting bits are later utilized as encryption keys. Control communication between KMS nodes is performed over a separate communication channel.

To initiate a secure connection with a remote server, clients communicate with the local KMS and request keys. One of possible ways to do this is through the proposed REST API standard (ETSI, 2019). The client provides information about the quantity of needed keys and the location of the destination server. Subsequently, this information is relayed to the global controller or directly to the target KMS to provide the same key to the client and the server. The way in which that communication operates is beyond the scope of this article.

Because of the no-cloning theorem (Wootters and Zurek, 1982; Gisin *et al.*, 2002), it is impossible to replicate quantum states to extend the signal range. Currently implementable quantum key distribution networks rely on the existence of trusted nodes. If the communication has to pass through multiple nodes, key material is consumed on each link on the path. The end-to-end key could be transported using the XOR operation on keys on KMSs on the path, which results in an identical key on both endpoints.

The achieved key generation rates are several orders of magnitude smaller than the throughput using the classical channel (Toshiba, 2024). Longer keys in symmetric algorithms increase security because they make brute-force attacks exponentially harder. Because of that, an efficient routing solution is necessary. It would enable an increase either in the number of provided symmetric cryptographic keys or in their length as the QKD network would utilize generated keys more efficiently. This mirrors the routing problem in classical networks with one major distinction. The keys generated in the key buffers on each link could be accumulated in the quantum key pool (Cao *et al.*, 2017a). This permits a temporary key consumption rate greater than the key generation one.

1.3. Related research. Multiple studies have explored routing within QKD networks (QKDNs). The approach described by Dianati *et al.* (2008) utilizes a hop-by-hop key relay system that incorporates a revised version of the Open Shortest Path First (OSPF) protocol. Cao *et al.* (2017b) introduced an integer linear programming (ILP) model and a heuristic approach to address the challenge of routing, wavelength, and time slot assignment (RWTA) in a QKDN. The study by Yang *et al.* (2017) presented an innovative link cost function that utilizes the residual link key stored in each link. The paths were determined using the Dijkstra algorithm, incorporating a penalty for longer paths in order to reduce the total number of keys consumed in the network. Yang *et al.* (2018) proposed a similar Dijkstra-based solution. Using the average key generation and key depletion index within the link cost equation, this approach achieved an increase in the key exchange success rate compared to the reference study. The study by Johann *et al.* (2023) introduced a machine learning approach using long short-term memory (LSTM) networks, applying them to each link to predict future request rates. Traffic requests were aggregated and processed every second. The proposed solution closely approximated the perfect estimate. In the work of Lee *et al.* (2023), the newly proposed algorithm was compared with the estimated perfect ILP solution. Although the minimum and average available keys in the buffers were slightly reduced, the solution computation was significantly faster.

1.4. Main contributions. The present study is focused on efficient key management in QKDNs. The article adds to the body of research through

- development of a QKDN simulator operating on

single packet requests,

- introducing an innovative approach to determining link costs for per-packet routing,
- using artificial fish colony optimization to finding factors for the link cost equation.

The remainder of the paper is structured as follows. Section 2 provides a detailed formulation of the problem alongside the proposed solution, which employs the artificial fish swarm optimization algorithm. In Section 3 the results of the experimental research are shown. Lastly, Section 4 concludes the article and outlines potential future research directions related to the problem.

2. Problem statement and the proposed solution

To address routing challenges within a QKD network, we propose a novel parametrized link cost function designed to facilitate the calculation of routing paths. The parameters of this function are fine-tuned by employing the artificial fish swarm optimization algorithm (AFSA). The evaluation of network routing performance through a mathematical model is theoretically plausible (Dubey *et al.*, 2021), yet it often proves to be a challenging endeavor. Consequently, simulation methods are typically employed, as they tend to provide more precise results (Borylo *et al.*, 2017). Therefore, we developed a simulator tailored specifically for this purpose.

2.1. Problem formulation. The purpose of the routing mechanism within the QKD network is to determine the most efficient relay path for a successful key exchange. Such a path needs to have a sufficient amount of key material in each link which it contains, otherwise it will not be feasible to generate an end-to-end key with it. The optimal path would not only be the one with the least number of links in it, but would also minimize the number of future denied requests by utilizing links with more key material in buffers.

In situations where the quantity of requests surpasses the system's capacity to recalculate a new route for each request, a heuristic approach must be implemented. This method may be employed by using the same routes for a period of time and recalculating them. It would still retain some of the advantages of dynamic routing while being more scalable and introducing less latency due to the usage of precomputed paths.

This approach might utilize link features such as the amount of residual key in buffers and the number of recently consumed key bits. The weights of the link cost function could be optimized with the use of the artificial fish swarm algorithm.

2.2. Artificial fish swarm algorithm. The artificial fish swarm algorithm (AFSA) is a bio-inspired optimization algorithm proposed in 2002 (Li *et al.*, 2002). It uses the concept of the artificial fish (AF) as a candidate solution. Each AF is indicated as a vector $\vec{x} = (x_1, x_2, \ldots, x_d)$, where d denotes the dimension of the vector. Initially, the AFSA creates n AFs randomly scattered throughout the search space, with n representing the size of the population. Then each AF searches for the position with the superior fitness function value using three behaviors: preying, swarming, and following. The search continues until a termination criterion is met, such as reaching the maximum number of epochs.

The preying is modeled on the current position of an AF and its surrounding area on the basis of its field of view v. The visual length v is calculated using the Euclidean distance. The i-th AF conducts a random search within its visual field by

$$\vec{x}_v = \vec{x}_i^{(t)} + \left(v \cdot U(0,1) \cdot \frac{\vec{u}}{\|\vec{u}\|} \right), \tag{1}$$

where $\vec{x}_i^{(t)}$ represents the current position of the i-th AF, $\vec{x}_v$ is a random position inside the field of vision of the i-th AF, $U(0,1)$ denotes a random number drawn from a uniform distribution over the interval $(0,1)$, vector $\vec{u}$ consists of uniformly distributed numbers over the interval $(-1,1)$, $\|\vec{u}\|$ is the $L2$ norm (or Euclidean length) of $\vec{u}$, and $\frac{\vec{u}}{\|\vec{u}\|}$ results in a unit vector.

If the fitness score of $\vec{x}_v$ is better, the i-th AF progresses in its direction by using the following formula:

$$\vec{x}_{p,i} = \vec{x}_i^{(t)} + \left(s \cdot U(0,1) \cdot \frac{\vec{x}_v - \vec{x}_i^{(t)}}{\|\vec{x}_v - \vec{x}_i^{(t)}\|} \right), \tag{2}$$

where $\vec{x}_{p,i}$ represents the position of the i-th AF after performing the praying behavior, with s denoting the *step size*. This position is located along the line $\vec{x}_v - \vec{x}_i^{(t)}$. The distance $\|\vec{x}_{p,i} - \vec{x}_i^{(t)}\|$ does not exceed the step length s. Otherwise, an alternative position $\vec{x}_v$ is randomly selected as described in (1).

If the i-th AF is unable to find an improved position after exploring t potential candidates, referred to as the *max-try-number*, it will take a random step within a radius v:

$$\vec{x}_{p,i} = \vec{x}_i^{(t)} + \left(v \cdot U(0,1) \cdot \frac{\vec{u}}{\|\vec{u}\|} \right). \tag{3}$$

Swarming is a behavioral pattern that enables fish to gather in a group and simultaneously avoid overly congested areas. Let $\vec{x}_c$ denote the central position of the swarm, which is calculated as the average position of all AFs:

$$\vec{x}_c = \frac{1}{n} \sum_{i=1}^{n} \vec{x}_i. \tag{4}$$

The i-th AF moves towards $\vec{x}_c$ as it provides a better fitness value and the area around it is not overcrowded. The determination of overcrowding around the position $\vec{x}_c$ is determined by

$$C(\vec{x}_c) = \begin{cases} 1, & \text{if } \dfrac{N(\vec{x}_c)}{n} > \delta, \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where $C(\vec{x}) = 1$ indicates that the vicinity of $\vec{x}$ is overcrowded, $N(\vec{x})$ refers to the number of AFs within a Euclidean distance of less than v from the i-th AF, and $\delta \in [0, 1]$ is defined as the crowding factor. The AFSA aims to preserve the diversity of the swarms by avoiding movements toward congested places to maintain the swarm diversity, and by avoiding premature convergence. If $C(\vec{x}_c) = 0$ and $f(\vec{x}_c) \leq f(\vec{x}_i)$, the i-th AF advances a step toward the center position:

$$\vec{x}_{s,i} = \vec{x}_i^{(t)} + \left(s \cdot U(0,1) \cdot \frac{\vec{x}_c - \vec{x}_i^{(t)}}{\|\vec{x}_c - \vec{x}_i^{(t)}\|} \right); \tag{6}$$

otherwise, the i-th AF will execute the preying behavior.

Following is a behavior that enables the AF to approach others with superior solutions. Specifically, for the i-th AF, if there exists any other AF (denoted by j) within its neighborhood v exhibiting a better fitness value, the i-th AF then advances towards the position of the j-th AF by

$$\vec{x}_{f,i} = \vec{x}_i^{(t)} + \left(s \cdot U(0,1) \cdot \frac{\vec{x}_j - \vec{x}_i^{(t)}}{\|\vec{x}_j - \vec{x}_i^{(t)}\|} \right); \tag{7}$$

otherwise, it will execute the preying behavior.

After initialization of the fish swarm, each AF engages in swarming and following behavior at each optimization step t. In addition to that, after each iteration, the value of v is updated with the equation

$$v^{(t+1)} = q \cdot v^{(t)}, \tag{8}$$

where $q \in [0, 1]$ is the attenuation factor.

The AFSA is used only during link cost function parameter tweaking. Its substantial overhead does not impact the performance during usual QKD network operations.

2.3. Link cost algorithm. The network topology of a QKDN can be represented as a graph $G = (V, E)$, where nodes V represent KMSs with QKD devices, and edges E indicate quantum channel links with buffers dedicated for storing securely exchanged keys. The Dijkstra algorithm (Dijkstra, 1959) can be utilized to determine the shortest path between two nodes. This algorithm addresses the single-source shortest-path issue in graphs with nonnegative edge weights by producing a shortest-path tree. Once computed, these paths remain unchanged until there is an update to the edge weights. Consequently, recalculating the path for every new key request becomes redundant.

In considering the dynamic weight of links, it is essential to account for both the present residual key within the buffers and the recent key usage rates. Identifying an appropriate function that can effectively minimize the likelihood of request denials is a complex challenge. Our proposed approach addresses this challenge by introducing a weight function composed of seven distinct parameters, each of which is optimized through the AFSA.

Let us define the AF as $\vec{x} = (x_1, x_2, \ldots, x_7)$, which is a vector of parameters to optimize. To make the final equation more straightforward, we introduce the auxiliary buffer emptiness function:

$$B_{i,j}(t) = \frac{S_{\max} - S_{i,j}(t)}{S_{\max}}, \tag{9}$$

where i and j are KMS indexes indicating a link between them, $S_{\max}$ denotes the maximum buffer capacity, and $S_{i,j}(t)$ refers to the remaining key in the buffer at the time t. This function reaches a maximum of 1 when the buffer is completely empty and a minimum of 0 when the buffer is entirely full.

The average scaled key consumption during the interval τ is given by

$$U_{i,j}^{(\tau)}(t) = \frac{C_{i,j}^{(\tau)}(t)}{\tau \cdot G_{i,j}^{(\tau)}(t)}, \tag{10}$$

where $C_{i,j}^{(\tau)}(t)$ represents the quantity of key consumed on the link during the interval τ and $G_{i,j}^{(\tau)}(t)$ is the amount generated. During experiments, the values of τ are 5 and 30 seconds—the former reflecting the shortest path recomputation interval and the latter providing a more stable historical statistic.

Moreover, we incorporate a penalty mechanism for links where the current residual key falls below the specified threshold. This strategy aims to minimize the utilization of links with a scarce amount of buffered bits, restricting their usage to scenarios where all alternative paths are similarly deficient in key resources. Such links are expected to have a lower key availability than what is needed within the time frame up to the subsequent recalculation. Consequently, this would lead to the denial of requests due to insufficient key resources. The penalty is given by the following equation:

$$P_{i,j}(t) = \begin{cases} 0, & \frac{S_{i,j}(t)}{S_{\max}} \geq x_6, \\ 1000, & \text{otherwise,} \end{cases} \tag{11}$$

where x_6 is one of the parameters of the artificial fish vector.

The final component of the link cost function is the inverse of the amount of key present in a buffer:

$$I_{i,j}(t) = \min\left(\frac{S_{\max}}{S_{i,j}(t)}, 100\right). \tag{12}$$

The function has an upper bound of 100 in order to not exceed the value of activated $P_{i,j}(t)$.

The total weight of the link can be derived using the formula

$$\begin{aligned} W_{i,j}(t) = {} & x_1 \cdot B_{i,j}(t) + x_2 \cdot e^{(1+x_3)\cdot B_{i,j}(t)} \\ & + x_4 \cdot U_{i,j}^{(5)}(t) + x_5 \cdot U_{i,j}^{(30)}(t) \\ & + P_{i,j}(t) + x_7 \cdot I_{i,j}(t). \end{aligned} \tag{13}$$

Dijkstra's algorithm functions only on graphs that have cycles with nonnegative edge weights. To preserve this requirement, any negative edge weight $W_{i,j}(t)$ is transformed to zero. This modification is carried out within the simulator, allowing the AFSA to continue operating without facing any restrictions despite altering these values.

The proposed link cost function is efficient, needing minimal calculations and causing little overhead. Each weight is computed just once by at least one node at the end of every quantum link, minimizing the computation load. Relying solely on link cost also boosts security—there is no need to share the link's exact status with other nodes in the network.

2.4. Simulation environment. Because the proposed approach is novel and no appropriate simulator could be identified, a new one was developed. For packet request routing, discrete event simulation was utilized to ensure that the results were consistent. This approach ensures that each event occurs at a specific time point, indicating a shift in the system's state. Using the same seed and parameters guarantees identical outcomes.

The simulator was developed in Python using the NetworkX library (Hagberg *et al.*, 2008). The entire network topology was represented as a graph. Each node within this graph symbolized a KMS. Key requests were generated directly on the KMS nodes themselves, rather than from their associated clients. The specifics of communication between clients and the KMS are outside the scope of this work and were not simulated.

QKD links were simulated as part of the links in the graph. Both buffers at the end of each link contained the same amount of key material as generated by the corresponding KMS. The key generation rate depends on the length of the link. In our study, we assume the presence of trusted relays on longer links which counteract this effect. As our research deals with key management in QKD networks, we did not simulate underlying hardware layer and key generation algorithms. With this assumptions, we set the key generation rate on all links to 1 Kbit/s with an increment in buffers occurring every 5 seconds. All key buffer sizes $S_{\max}$ were set to 1 Mbit, and they were half filled at the start of the simulation.

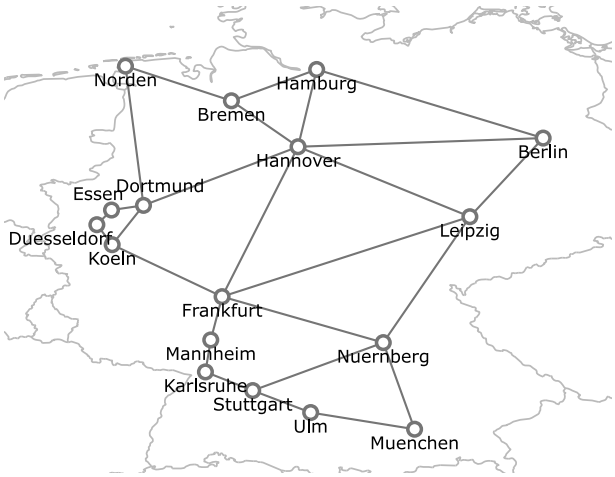

Fig. 2. Topology *nobel-germany* from SNDlib.

Currently, there is a lack of data on real-world implementations of the complex QKDN topology. Evaluating the efficiency of routing protocols on a nearly-linear graph is unreliable. Because of that, information on the structure of the network was taken from SNDlib (Orlowski *et al.*, 2010), which contains large-scale real-life networks. For training, *nobel-germany* was used and it contained 17 nodes and 26 links. The larger *germany50* topology with 50 nodes and 88 links was employed for evaluation. Figure 2 provides a visual representation of *nobel-germany*.

One of the main reasons for selecting these topologies were corresponding demand matrices featuring a time frame of one day, one month, and one year. Each pair of nodes contains information on the average traffic between them at specified intervals. For the one-day horizon that was used, they come with a granularity of 5 minutes. The value of demands was treated as the average amount of bits/s requested and was used to generate intervals between each 256 bit key request using exponential distribution. They were multiplied by a factor of 1.5 to force the situation where the keys are depleted in the buffers during the simulation.

3. Experimental results

In this section, we present the results of experiments performed to verify the proposed solution. We initially discuss the environment in which the parameter search was conducted. Subsequently, we assess the

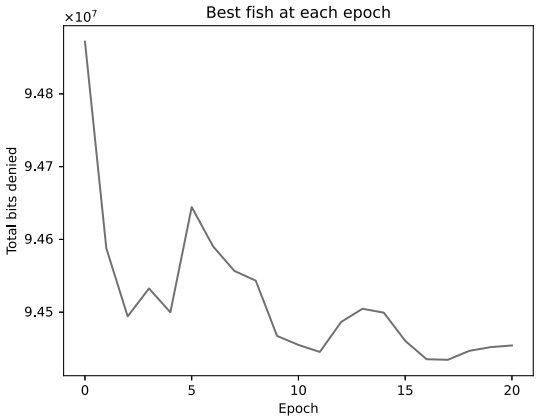

Fig. 3. Best fish with the lowest amount of denied bits.

Table 1. Value of parameters of the best fish (*nobel-germany*).

Fish parameter	Value
x_1	0.36427
x_2	0.53421
x_3	0.93294
x_4	0.78970
x_5	0.09121
x_6	0.99019
x_7	0.18507

generalization ability of the proposed method.

3.1. Parameter search. Finding the best parameters for minimizing total bits denied was done using the *nobel-germany* topology. The *population size* of the artificial fish swarm was established at $n = 10$, with the *visual length* set at $v = 1$. The *step size* was configured to $s = 0.8$, and an attenuation factor of $q = 0.9$ was used. Furthermore, *max-try-number* was limited to five attempts. The search spanned an *epoch* of 20. Initial values for all $\vec{x}$ were uniformly distributed within the range $(0, 1)$. Each simulation run used the same seed to make the results directly comparable.

The values of the total bits denied for the best position of the fish in each epoch are presented in Fig. 3. The progression was neither linear nor substantial, with only a slight reduction of a few percent in the bits denied. This modest improvement suggests that the proposed method might not be capable of uncovering a significantly better solution.

The highest amount of bits accepted was achieved by the fish in the 17th epoch. These values, presented in Table 1, were used for all remaining evaluations. The evolution of the parameters at all examined positions is presented in Fig. 4.

The first parameter x_1 was responsible for the linear correlation between the cost of the link and the fulfillment of the buffer. The decrease in it over time might indicate that non-linear parts of the function provide greater benefit. The exponential non-linearity factors, x_2 and x_3, increase progressively, particularly the exponential term x_3.

The consumption observed in the last five seconds (x_4) appeared to have more significance for determining the link cost compared to the data from the preceding 30-second interval (x_5).

The penalty threshold indicated by x_6 appeared to behave differently from our expectations. Instead of removing only links with a small amount of stored keys in buffers for shortest path selection by increasing its weight, it put the penalty for all links except those with buffers that were fully or nearly fully filled. This strategy promotes the usage of links where more key bits could not be stored, and thus the oldest bits would be discarded. This increases overall network throughput. At the same time, it comes with the downside of turning the cost of most links similar, at a large level, which makes the algorithm act similarly to hop count routing, where all links are treated equally.

The last parameter x_7 represented the proportion of inverse buffer fulfillment, and its value did not converge for each fish. Nonetheless, it might have played an important role in the link cost function.

To assess the efficacy of the proposed function, it was evaluated against the baseline models. The first one used a constant value for each cost of the link (*const*). The second one utilized the inverse of the normalized amount of key in the link buffers following the formula

$$C_{i,j}^{\text{inverse}}(t) = \frac{S_{\max}}{S_{i,j}(t)}, \tag{14}$$

referred to as *inverse*. Table 2 presents a comparison of the primary bits needed and delivered across the entire network during the simulation on *nobel-germany*, with the demands increased by a factor of 1.5.

As expected, assigning a constant cost for all links yields the poorest performance. Both *afsa* and *inverse* manage to reduce the number of denied bits by nearly half. Moreover, *afsa* outperforms *inverse* by roughly one percentage point, suggesting that utilizing multiple functions as the link cost could be a more effective approach.

3.2. Generalization. After assessing the proposed function and determining that it outperformed the baselines on the topology used for parameter search, we proceeded to test it on a different network. For this purpose, another SNDlib topology named *germany50* was selected. It came with its own demand matrices, which, in a similar manner, were used as the average number of requested key bits/s. The comparison was conducted using the same seed for each simulation, both for the

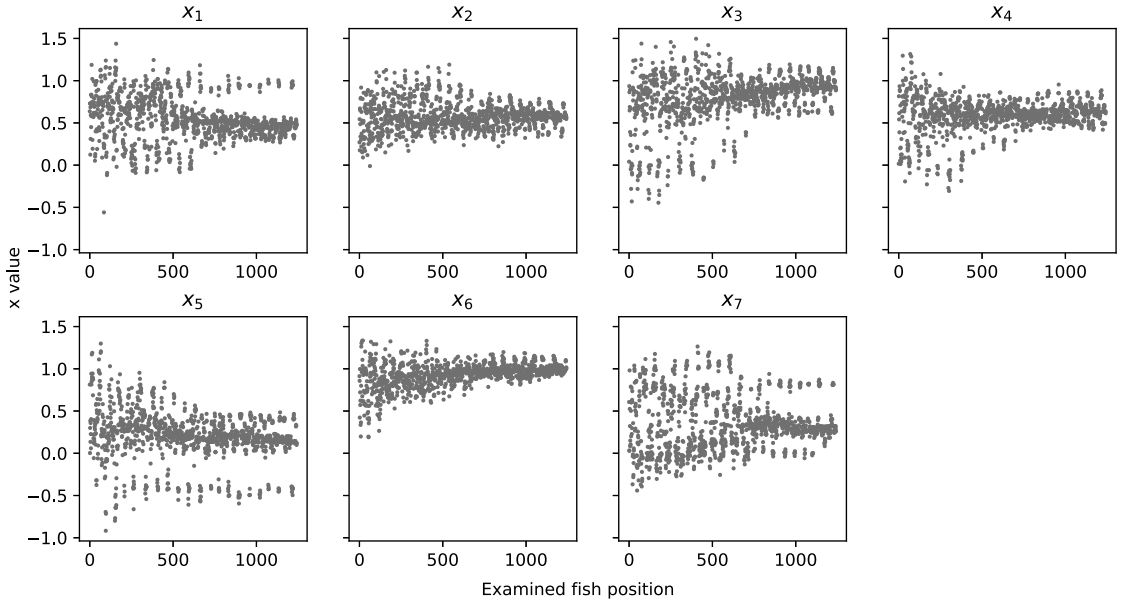

Fig. 4. Change of parameter values during the search using the AFSA.

Table 2. Comparison of algorithms evaluated on the *nobel-germany* topology.

Algorithm	Requested bits	Denied bits	Provided bits	Percentage of denied bits
const	592390400	177488896	414901504	0.2996
inverse	592390400	99293696	493096704	0.1676
afsa	592390400	94434560	497955840	0.1594

afsa and the baseline methods. The results obtained for different values of demand multipliers are presented in Table 3.

The difference in performance between the *inverse* and the AFSA optimized function is smaller than on the *nobel-germany* network. This implies possible overfitting, which results in diminished performance on other network topologies. In the case of the AFSA, overcoming this with the usage of different topologies or seeds for each fish is not straightforward. In such a case, the values of the fitness function returned by the simulator would not be fully comparable as the environment would differ.

The primary aim of analyzing the unchanged parameters of the link cost function on a different topology was to assess their transferability. Additional fine-tuning, similar to the search on *nobel-germany*, is expected to boost network performance.

The fitness score of the optimized function shows some improvement, especially under typical scenarios where congestion is lower. Reducing the demands by multiplying their number by 0.9 led to complete elimination of denied requests, making it unnecessary to explore scenarios with fewer keys requested than that. The most significant improvement occurred when the demands remained unchanged, meaning the multiplier was set to 1.0. As the number of key requests increased incrementally during the simulations, the benefit diminished, while still being visible at a multiplier of 2.0. When denied requests go above 0.25 of the total, the results are no longer better than the *inverse* baseline. This may be due to the parameter search being conducted in conditions of lower congestion, with approximately 0.16 of requests rejected. While performance aligns with the baseline at higher denial rates, such scenarios are less typical. Furthermore, in real cases, the machine learning algorithm must search for optimal parameters only for a selected QKDN topology. Thus, the stability topology of QKD networks minimizes the significance of this issue.

4. Summary

In this paper, we provided an overview of quantum key distribution networks with an explanation of their working principles and their importance for future secure information transmissions. We proposed a new function that facilitates the exchange of encryption keys between endpoints. The parameters of the proposed solution were optimized using an artificial fish swarm optimization algorithm.

The findings demonstrate that this approach enhances the performance of the QKD network. The improvement is not substantial, but remains

Table 3. Comparison of algorithms run on *germany50* for different amounts of requests with the resulting percentage of denied requests.

Demand multiplier	*const*	*inverse*	*afsa*	*afsa-inverse*
0.9	0.1412	0.0009	0.0000	−0.0009
1.0	0.1755	0.0260	0.0227	−0.0033
1.1	0.2068	0.0513	0.0487	−0.0027
1.2	0.2389	0.0792	0.0768	−0.0025
1.3	0.2673	0.1078	0.1057	−0.0020
1.5	0.3163	0.1595	0.1578	−0.0017
1.7	0.3596	0.2035	0.2017	−0.0019
2.0	0.4126	0.2569	0.2556	−0.0012
2.5	0.4769	0.3337	0.3341	0.0004
3.0	0.5218	0.3891	0.3887	−0.0004
4.0	0.5854	0.4663	0.4655	−0.0008
5.0	0.6305	0.5198	0.5202	0.0004

meaningful. Further evaluation indicated a tendency towards overfitting due to worse performance in a different environment. Despite this, our composed link cost function still outperformed baseline models.

For future work, overfitting reduction methods might be discussed. Additionally, parameter search algorithms other than the AFSA may also be studied. Examining methods that use graph structures, such as graph neural networks, is another possibility.

Acknowledgment

The research was partly supported by the program *Excellence Initiative—Research University* for the AGH University of Krakow and by the EU Horizon Europe Framework Program under the grant agreement no. 101119547 *(PQ-REACT)*.

References

Ashok, K. and Gopikrishnan, S. (2023). Improving security performance of healthcare data in the Internet of medical things using a hybrid metaheuristic model, *International Journal of Applied Mathematics and Computer Science* **33**(4): 623–636, DOI: 10.34768/amcs-2023-0044.

Bennett, C.H. and Brassard, G. (2014). Quantum cryptography: Public key distribution and coin tossing, *Theoretical Computer Science* **560**(Part 1): 7–11, DOI: 10.1016/j.tcs.2014.05.025.

Borylo, P., Cholda, P., Domzal, J., Jaglarz, P., Jurkiewicz, P., Lason, A., Niemiec, M., Rzepka, M., Rzym, G. and Wojcik, R. (2017). SDNroute: Integrated system supporting routing in software defined networks, *2017 19th International Conference on Transparent Optical Networks (ICTON), Girona, Spain,* pp. 1–4.

Cao, Y., Zhao, Y., Colman-Meixner, C., Yu, X. and Zhang, J. (2017a). Key on demand (KoD) for software-defined optical networks secured by quantum key distribution (QKD), *Optics Express* **25**(22): 26453–26467.

Cao, Y., Zhao, Y., Yu, X. and Wu, Y. (2017b). Resource assignment strategy in optical networks integrated with quantum key distribution, *Journal of Optical Communications and Networking* **9**(11): 995–1004.

Dianati, M., Alléaume, R., Gagnaire, M. and Shen, X.S. (2008). Architecture and protocols of the future european quantum key distribution network, *Security and Communication Networks* **1**(1): 57–74.

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs, *Numerische Mathematik* **1**(1): 269–271, DOI: 10.1007/BF01386390.

Dubey, S.P., Kedar, G.D. and Ghate, S.H. (2021). A communication network routing problem: Modeling and optimization using non-cooperative game theory, *International Journal of Applied Mathematics and Computer Science* **31**(1): 155–164, DOI: 10.34768/amcs-2021-0011.

El-Douh, A.A.-R., Lu, S.F., Elkouny, A.A. and Amein, A. (2022). Hybrid cryptography with a one time stamp to secure contact tracing for COVID-19 infection, *International Journal of Applied Mathematics and Computer Science* **32**(1): 139–146, DOI: 10.34768/amcs-2022-0011.

ETSI (2019). Quantum key distribution (QKD); Protocol and data format of REST-based key delivery API, *Technical report*, European Telecommunications Standards Institute, Sophia Antipoli, `https://www.etsi.org/deliver/etsi_gs/QKD/001_099/014/01.01.01_60/gs_qkd014v010101p.pdf`.

Gisin, N., Ribordy, G., Tittel, W. and Zbinden, H. (2002). Quantum cryptography, *Reviews of Modern Physics* **74**(1): 145.

Hagberg, A.A., Schult, D.A. and Swart, P.J. (2008). Exploring network structure, dynamics, and function using NetworkX, *Proceedings of the 7th Python in Science Conference, Pasadena, USA,* pp. 11–15, DOI: 10.25080/TCWV9851.

ITUTSS (2019). Recommendation y.3800 (10/19): Overview on networks supporting quantum key distribution, *Technical report*, International Telecommunication Union

Telecommunication Standardization Sector, Geneva, https://www.itu.int/rec/T-REC-Y.3800-201910-I/en.

Ivanov, A. and Stoianov, N. (2023). Implications of the arithmetic ratio of prime numbers for RSA security, *International Journal of Applied Mathematics and Computer Science* **33**(1): 57–70, DOI: 10.34768/amcs-2023-0005.

Johann, T., Giemsa, D., Kuehl, S., Dochhan, A. and Pachnicke, S. (2023). Routing optimization of QKD-networks using machine-learning based prediction, *Photonic Networks: 24th ITG-Symposium, Berlin, Germany*, pp. 1–5.

Kurek, T., Niemiec, M. and Lason, A. (2015). Taking back control of privacy: A novel framework for preserving cloud-based firewall policy confidentiality, *International Journal of Information Security* **15**(3): 235–250.

Lee, C., Kim, Y., Shim, K. and Lee, W. (2023). Key-count differential-based proactive key relay algorithm for scalable quantum-secured networking, *Journal of Optical Communications and Networking* **15**(5): 282–293.

Li, X., Shao, Z. and Qian, J. (2002). Optimizing method based on autonomous animats: Fish-swarm algorithm, *System Engineering Theory and Practice* **22**(11): 32–38 (in Chinese).

Mehic, M., Niemiec, M., Rass, S., Ma, J., Peev, M., Aguado, A., Martin, V., Schauer, S., Poppe, A., Pacher, C. and Voznak, M. (2020). Quantum key distribution: A networking perspective, *ACM Computing Surveys (CSUR)* **53**(5): 1–41, DOI: 10.1145/3402192.

NIST (2024). Module-lattice-based key-encapsulation mechanism standard, *Technical report*, Information Technology Laboratory, National Institute of Standars and Technology, Gaithersburg.

Orlowski, S., Wessäly, R., Pióro, M. and Tomaszewski, A. (2010). SNDLIB 1.0—Survivable network design library, *Networks* **55**(3): 276–286.

Shor, P.W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM Journal on Computing* **26**(5): 1484–1509, DOI: 10.1137/S0097539795293172.

Toshiba (2024). Quantum key distribution products, Toshiba, Tokyo, https://www.global.toshiba/ww/products-solutions/security-ict/qkd/products.html.

Wootters, W.K. and Zurek, W.H. (1982). A single quantum cannot be cloned, *Nature* **299**(5886): 802–803.

Yang, C., Zhang, H. and Su, J. (2017). The QKD network: Model and routing scheme, *Journal of Modern Optics* **64**(21): 2350–2362.

Yang, C., Zhang, H. and Su, J. (2018). Quantum key distribution network: Optimal secret-key-aware routing method for trust relaying, *China Communications* **15**(2): 33–45, DOI: 10.1109/CC.2018.8300270.

Yilmaz, M.B. and Öztürk, K. (2025). Vanilla convolutional neural network is all you need for online and offline signature verification, *International Journal of Applied Mathematics and Computer Science* **35**(2): 357–370, DOI: 10.61822/amcs-2025-0025.

Wojciech Szczepanik received his MS degree from the AGH University of Krakow in 2021. He is currently pursuing a PhD on the use of machine learning for cybersecurity tasks. He is participating in the EU Horizon Europe project *PQ-REACT*.

Marcin Niemiec is a university professor at the Institute of Telecommunications, AGH University of Krakow. His research interests focus on cybersecurity, especially cryptography, network security, and quantum cryptography. He has actively participated in the 6th, 7th, H2020, and Horizon Europe Framework Programmes (*e-Photon/ONE+, BONE, SmoothIT, INDECT, SCISSOR, ECHO, PQ-REACT*), Eureka-Celtic (*DESYME*), as well as many national research projects. He has co-authored over 100 publications.

Received: 28 February 2025
Revised: 24 June 2025
Re-revised: 12 August 2025
Accepted: 17 August 2025