electronics

MDPI

Article

Ship Network Traffic Engineering Based on Reinforcement Learning

Xinduoji Yang [1,†], Minghui Liu [2,†], Xinxin Wang [2], Bingyu Hu [2], Meng Liu [2] and Xiaomin Wang [2,*]

[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610000, China; xinduojiyang@std.uestc.edu.cn
[2] Yangtze Delta Region Institute (Quzhou), University of Electronic Science and Technology of China, Quzhou 324003, China; minghuiliu@csj.uestc.edu.cn (M.L.); xinxinwang@std.uestc.edu.cn (X.W.); bingyuhu@std.uestc.edu.cn (B.H.); mengliu@std.uestc.edu.cn (M.L.)
* Correspondence: xmwang@uestc.edu.cn
† These authors contributed equally to this work.

Abstract: This research addresses multiple challenges faced by ship networks, including limited bandwidth, unstable network connections, high latency, and command priority. To solve these problems, we used reinforcement learning-based methods to simulate traffic engineering in ship networks. We focused on three aspects—traffic balance, instruction priority, and complex network structure—to evaluate reinforcement learning performance in these scenarios. Performance: We developed a reinforcement learning framework for ship network traffic engineering that treats the routing policy as the state and the network state as the environment. The agent generates routing changes and uses actions to optimize traffic services. The experimental results show that reinforcement learning optimizes network traffic balance, reasonably arranges instruction priorities, and copes with complex network structures, greatly improving the network's quality of service (QoS). Through an in-depth analysis of the experimental data, we noticed that network consumption was reduced by 9.1% under reinforcement learning. Reinforcement learning effectively implemented priority routing of high-priority instructions while reducing the occupancy rate of the edge with the highest occupancy rate in the network by 18.53%.

Keywords: ship network; traffic engineering; reinforcement learning; quality of service (QoS)

Citation: Yang, X.; Liu, M.; Wang, X.; Hu, B.; Liu, M.; Wang, X. Ship Network Traffic Engineering Based on Reinforcement Learning. *Electronics* **2024**, *13*, 1710. https://doi.org/10.3390/electronics13091710

Academic Editor: Claus Pahl

Received: 1 April 2024
Revised: 22 April 2024
Accepted: 23 April 2024
Published: 29 April 2024

1. Introduction

As shown in Figure 1, the ship's network structure is divided into physical, access, and core layers. This design provides reliable communications [1], data transfer, and ship management.

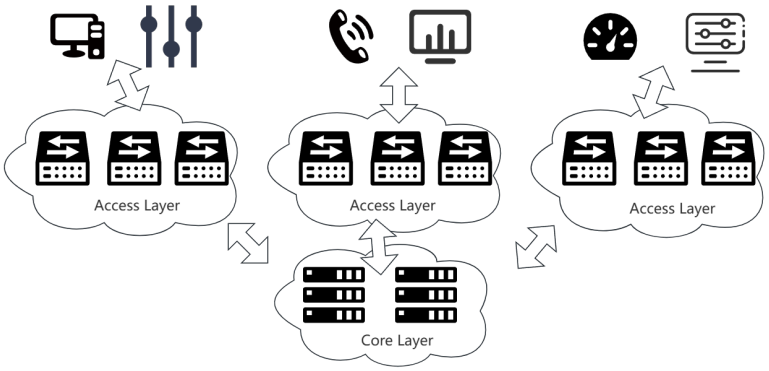

Figure 1. Network structure on the ship.

This layered structure helps optimize network performance, scalability, and management. In the ship industry, this network structure supports various applications, including

ship management systems, navigation systems, communication systems, etc. At the same time, it can also adapt to different connection needs, such as communication between the internal equipment of the ship, communication between the ship and shore-based systems, etc. Traffic engineering [2] is a method of network management and optimization that effectively controls and guides data traffic to improve network performance [3], availability, and efficiency. As shown in Figure 2 [4], if all tasks choose the shortest path, it will cause congestion in the shortest path and idle other paths. Optimal routes [5] are selected to reduce delay and improve network availability according to the network's topology and actual needs. Traffic engineering is particularly important in scenarios involving large networks, cloud computing environments, and Internet service providers, whose complex data traffic requires meticulous management and optimization to ensure efficient operation.

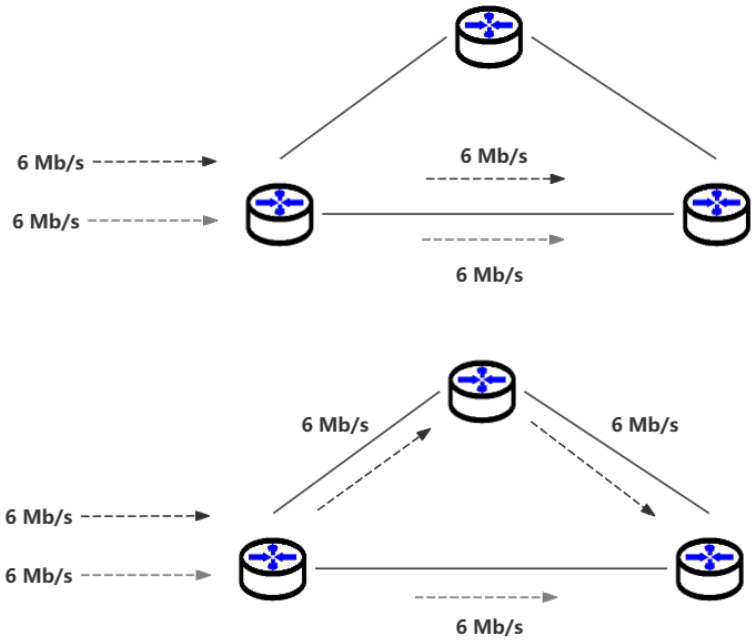

Figure 2. Schematic diagram of routing optimization.

Traffic engineering refers to a series of technologies and methods that improve network performance, increase bandwidth utilization, and ensure service quality by optimizing data transmission and distribution flows. In ship networks, traffic management involves ship communications and data transmission.

Ship networks often face limited bandwidth, unstable network connections, high latency, etc. To solve these challenges, traffic engineering aims to effectively manage the ship network's data flow and ensure various ship applications' (such as navigation, communication, monitoring, etc.) performance and reliability under limited resources. In ship networks, traffic engineering ensures effective utilization of bandwidth and meets the needs of different applications by dynamically allocating and adjusting bandwidth [6]. It also involves distributing data traffic across different network paths or communication channels to prevent one path from being overloaded, thereby improving overall network performance. Traffic engineering ensures that critical applications (such as navigation systems) obtain sufficient bandwidth and low latency, guaranteeing their normal operation. In response to network failures or interruptions, traffic engineering uses fault-tolerant mechanisms and backup paths to maintain data transmission continuity and reliability. It also involves real-time ship network monitoring and dynamically adjusting traffic engineering strategies based on traffic conditions and network status. Through effective traffic engineering, ship networks can better adapt to different environments and mission requirements, improving communication reliability and performance. These measures are critical for the proper functioning of various applications and systems on board ships.

Reinforcement learning (RL) [7] is a branch of machine learning that teaches agents to learn from environmental interaction and maximize accumulation (schematic diagram of reinforcement learning is shown in Figure 3). Reward signal: Reinforcement learning models involve decision-making in unknown environments and learning optimal strategies through trial and error. The core concepts of reinforcement learning include the following: agent, environment, state, action, reward, and policy. The reinforcement learning process usually involves a balance between exploration and exploitation [8]. The agent must use known strategies to maximize rewards while trying new strategies to gain more information.

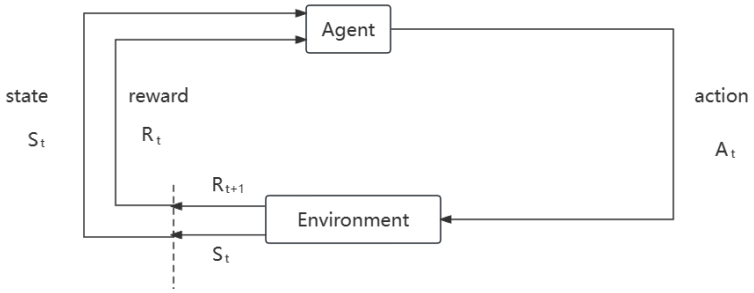

Figure 3. Schematic diagram of reinforcement learning.

Y. Shen et al. [9] used Q-learning to optimize the ship loading plan and verified its effectiveness in several actual application cases. Naval ship design is a complex and monumental task. Chou et al. [10] used AI to design modular naval ships, and the results were accurate, efficient, and interaction-friendly. Advances in computer vision may replace navigators. Erik Veitch and Ole Andreas Alsos [11] surveyed 42 studies on autonomous ship driving and proposed that before artificial intelligence can replace navigators, aspects such as flexibility, interactive design, and safety management must be considered. Zhao and Song [12] used YOLOv8 to detect and identify ships and solve problems caused by their irregular shapes and complex backgrounds. In the field of underwater acoustic recognition, machine learning methods rely on large amounts of data to achieve high accuracy, whereas the usual data amount in this field is small. Q. Yao et al. [13] used the residual network ResNet to conduct ship research for underwater acoustics. This method is also applicable to sound recognition in other fields. Thantharate and Anurag [14] presents a federated learning framework for collaborative cyber threat detection without compromising confidential data. Artificial intelligence has been widely used in the shipping industry but to a lesser extent in ship network design. In this study, we used reinforcement learning methods to conduct simulation research on traffic engineering for ship networks. We considered the many challenges faced by ship networks, including limited bandwidth, unstable network connections, high latency, and instruction priority. We focused on three aspects: traffic balancing [15], instruction priority, and complex network structure. We also used simulation experiments to explore reinforcement learning's effect on these scenarios. Our study results show that applying reinforcement learning to ship networks significantly improves the quality of service (QoS) [16]. Specifically, reinforcement learning optimized network traffic balance, prioritized instructions, and coped with complex network structures. By analyzing experimental data, we observed that network consumption was reduced by 9.1% under reinforcement learning. It achieved priority routing of high-importance instructions more effectively and reduced overall network utilization by about 17.55%.

This research provides a useful reference for optimizing ship network performance. It also highlights the potential advantages of reinforcement learning for environmental challenges in maritime communication. Our findings have important practical significance for improving the robustness and efficiency of ship communication systems.

2. Materials and Methods

2.1. Ship Network Structure

Ship networks carry a variety of data types crucial for vessel operations, safety, and communication. Understanding the characteristics and requirements of these data types is essential for effective network management and optimization. Control and command data are used for remote control of vessel systems, such as propulsion, navigation, and safety systems. Control and command data often require low-latency transmission to ensure timely response and control of vessel systems. While individual control commands may be small, frequent transmission and low-latency requirements necessitate moderate to high bandwidth allocation.

Ship network structure is a critical infrastructure designed to meet internal communication, data transmission, and management needs. Its design includes physical, access, and core layers. Each layer plays an important role in its design and helps provide reliable communications, data transfer, and ship management.

The physical layer forms the basis of the entire network structure and handles hardware devices and physical connections. In ship networks, the physical layer includes various transmission media, such as cables, optical cables, and satellite communications. This layer ensures that the physical connections between various parts of the ship are stable and reliable to support efficient data transfer. The access layer acts as a bridge connecting the physical layer and the core layer. It is responsible for data exchange, forwarding, and processing. Through the access layer, various ship equipment and sensors can connect to the network and perform data exchange and communication, including network switches, routers, and wireless access points connected to various devices. The core layer is the backbone of the entire network. It manages and schedules data traffic to ensure efficient data transmission from source to destination. In a ship network, the core layer may include high-performance switches and routers to ensure fast, reliable data transmission between different devices. This layer usually involves strategic design to ensure the network's high availability and fault tolerance. The core layer also provides connections to external networks, allowing the ship to communicate and exchange data with the outside world.

2.2. Q-Learning

Q-learning [17] is a reinforcement learning algorithm used to train agents in optimal environmental interaction strategies. Reinforcement learning is a branch of machine learning whose goal is to teach agents from trial and error and maximize accumulated rewards through environmental interaction [18].

Q-learning [18] was proposed by computer scientist Christopher Watkins in 1989. Its core idea is to represent the long-term cumulative reward of each state and action pair by learning a value function (Q-value function). The Q-value function is a measure of how good or bad it is to perform an action in a given state. The basic update rules of the algorithm are as follows:

$$Q(s,a) \leftarrow (1-\alpha) \cdot Q(s,a) + \alpha \cdot \left(r + \gamma \cdot \max_{a'} Q(s',a') \right)$$

where:

- $Q(s,a)$ is the Q-value of performing action a in state s.
- α (the learning rate) controls the extent to which newly acquired information overrides the old Q-value. A low value of α makes the agent more conservative and less likely to change its Q-values significantly based on new experiences, while a high value of α makes the agent more prone to quick changes in its Q-values.
- γ (the discount factor) determines the importance of future rewards. It represents the agent's preference for immediate rewards over future rewards. A value of γ close to 1 implies that future rewards are highly valued, while a value close to 0 implies that future rewards are discounted.

- s' is the new state the agent transitions to after performing action a.
- $\max_{a'} Q(s', a')$ represents the Q value corresponding to selecting the optimal action a' under the new state s'.

The term $R + \gamma \cdot \max_{a'} Q(s', a')$ represents the estimated total reward the agent expects to receive from the current state s after taking action a and transitioning to the next state s', considering the immediate reward R received and the maximum expected future reward $\gamma \cdot \max_{a'} Q(s', a')$ achievable from state s'. The learning rate α controls how much the new estimate of the Q-value (the right-hand side of the equation) updates the existing Q-value. It balances the trade-off between exploration and exploitation by determining the extent to which the agent relies on new information versus its previous knowledge. The discount factor γ discounts the value of future rewards, preventing the sum of rewards from becoming infinite in environments with infinite horizons or ensuring convergence in environments with finite horizons.

Q-learning trains the agent to learn optimal environmental interaction strategies and update the Q-value according to the reward signal, in other words, choose the action that maximizes the cumulative reward in each state. This strategy makes Q-learning excellent at solving a variety of reinforcement learning problems, including games, control problems, etc.

2.3. OSPF

Open Shortest Path First (OSPF) [19] is a dynamic routing protocol used for routing in computer networks. It is a type of Interior Gateway Protocol (IGP) used to exchange routing information between routers within a single autonomous system (AS). OSPF is an open standard whose protocol specifications are managed and maintained by the Internet Engineering Task Force (IETF). Among them, RFC 2328 [20] defines OSPFv2, which is an Internal Gateway Protocol (IGP) for IPv4 networks, used for dynamic routing selection between routers. It provides a detailed description of various aspects of the OSPFv2 protocol, including the data format of the protocol, protocol message exchange between routers, routing calculation, establishment and maintenance of neighbor relationships, etc. RFC 5340 [21] defines OSPFv3, which is an extension of the OSPF protocol for IPv6 networks. Similar to OSPFv2, OSPFv3 is used for routing selection in IPv6 networks. This system enables routers from different manufacturers to implement and support the same protocol standards. OSPF uses the shortest path first algorithm to determine paths for packets to travel through the network. In other words, the routing path it chooses is the shortest, i.e., has the lowest cost (usually the link bandwidth). OSPF is a link-state protocol that understands network topology by exchanging link-state databases (Link-State Database) [22]. Each router maintains a link-state database containing information about the status of all links in the network. OSPF runs within an autonomous system rather than between different autonomous systems. An autonomous system refers to a group of networks and routers controlled by a unified management entity. OSPF can divide the network into different areas to increase scalability and reduce the link-state database size. The router only knows the topology information within the area, not the overall topology of the autonomous system. OSPF helps connect disconnected areas through virtual links so that the entire autonomous system forms a logical connectivity structure.

OSPF is widely used in enterprise and Internet provider networks, providing a highly reliable, dynamically adaptable routing mechanism suitable for complex network environments.

3. Experiment

3.1. Traffic Balancing

Assume that there are two paths, 'left' and 'right', with the 'right' path being shorter than the 'left' path, and six instructions packaged into data packets, as shown in Figure 4. Firstly, for a task, only the left and right pathways are available for selection, forming the action space of reinforcement learning. Although individual actions are very easy

to determine, the continuous selection process of different tasks forms an exponential optimization space. In the traditional OSPF algorithm, all six instructions select the right path, which blocks it and reduces the instructions' QoS transmission efficiency.

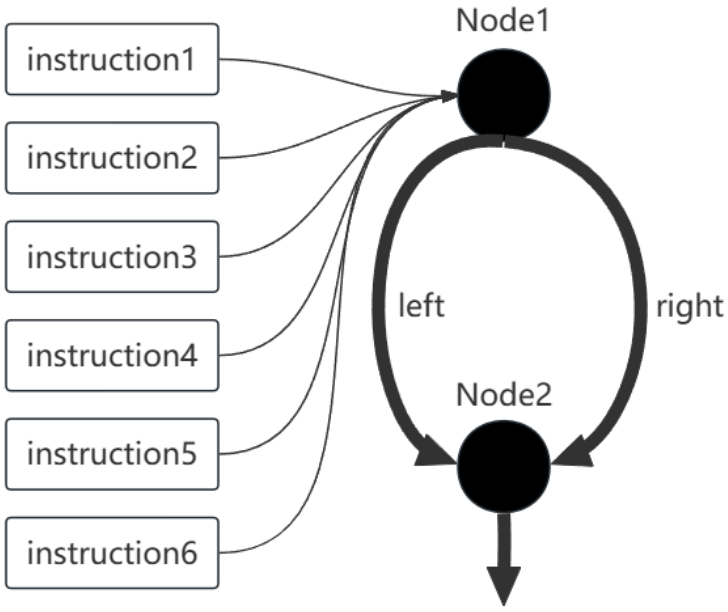

Figure 4. Schematic diagram of traffic balancing.

As mentioned in Section 2, reinforcement learning requires an environment, state, action, and reward. In traffic engineering, we can obtain the instruction id (0–5) and detect the busyness of the 'left' and 'right' paths. The states are combined into a state of reinforcement learning, as shown in the first column of the table, where '0' means the channel is idle and '1' means it is busy. When receiving an instruction, we must select an action from the left and right paths as the reinforcement learning action.

We designed the environment and rewards as follows: When the agent chooses a left (or right) action, we judge the left (or right) path in the current state. If the left (or right) channel is idle, a reward of '1' (or '2') is assigned because the right channel is shorter than the left. To achieve higher QoS, we generally hoped the instruction would pass through the right channel as much as possible without blocking. If the left (or right) path is busy, a reward of '0' is assigned.

Table 1 shows the Q table after 600 iterations. When the first instruction (id is '0') is obtained, the Q-target value for selecting the left path is 4.03361, which is lower than the right path at 7.15169. The Q-learning algorithm will select the right path with greater probability. Since the right channel is idle, the first instruction is transmitted through the right channel. The agent receives the second instruction (id is '1') but the right channel is occupied. Therefore, the agent reaches the status of the second row in the table. In the same way, when the second instruction is received (id is '1'), the Q-target value for selecting the left path is 5.72410, which is higher than the right path at 4.65601. The Q-learning algorithm will select the left path with greater probability. Since the left channel is idle, the second instruction will be transmitted through the left channel. When the agent receives the third instruction (id is '2'), the left channel is occupied. At the same time, the right channel completes the first instruction and returns to the idle state. The agent then reaches the status of the second row in the table. The agent only reaches the 'terminal state' when the six instructions are transmitted.

Table 1. Traffic balancing Q table.

State	Left	Right
0, 0, 0	4.03361	7.15169
1, 0, 1	5.72410	4.65601
2, 1, 0	4.38533	5.24900
3, 0, 1	3.61000	3.03399
4, 1, 0	2.53626	2.90000
5, 0, 1	1.00000	0.80855
Terminal	0.00000	0.00000
1, 1, 0	0.27307	4.47023
2, 0, 1	3.72987	0.14271
3, 1, 0	0.32789	3.81745
4, 0, 1	2.53300	0.48155
5, 1, 0	0.00000	1.94437

Due to the reward function settings in the Q table, the agent will prioritize the right channel when the left and right channels are both idle. Similarly, when the left (or right) channel is busy, the right (or left) channel will be selected. As shown in Figure 5, assuming that the transmission time of each instruction is 1 ms, the OSPF algorithm requires 6 ms to complete these instructions. By contrast, the reinforcement learning algorithm requires only 3 ms, indicating a 50% improvement in QoS compared to the OSPF algorithm. For more complex network topologies, defining the action space becomes challenging due to the increased number of decision points and potential actions at each point. In more complex scenarios, actions might be continuous variables representing parameters of a policy or control function. For instance, the agent could adjust the allocation of network resources along different paths in a continuous manner, such as by dynamically adjusting bandwidth allocation. However, this Q-learning algorithm only manages traffic balancing. Some ship system instructions are more important than others. The above algorithm may cause high-importance instructions to be transmitted through a longer path, reducing the ship command system's QoS. Therefore, we considered the priority transmission of high-importance instructions in our optimization goal and designed the following reinforcement learning algorithm.

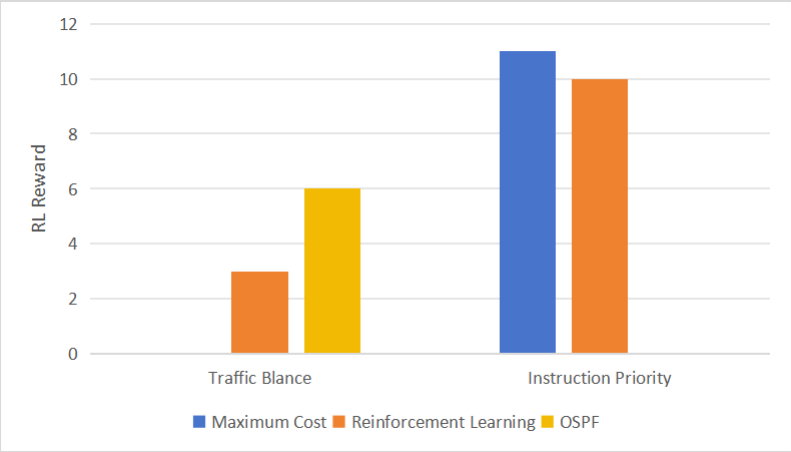

Figure 5. Traffic balancing and instruction priority experimental results.

3.2. Importance of Instructions

Control and command data play a pivotal role in remotely managing vital vessel systems, including propulsion, navigation, and safety mechanisms. Given the critical nature of these operations, certain instructions hold greater significance than others. To ensure swift response times and precise control over vessel systems, control and command data typically demand low-latency transmission. Although individual control commands

may be compact, the need for frequent transmission and adherence to low-latency standards necessitates allocating moderate to high bandwidth resources. Incorporating importance into the reward function involves assigning different weights or coefficients to rewards based on the importance level of instructions. Trade-off with network load balancing penalizing actions that lead to network congestion or imbalances in traffic distribution [23]. In Q-learning, Q(s1, a2) contains a maximum estimate of Q(s2). If the agent selects action a2 in state s1 and reaches state s2, no matter which action is selected in state s2, it will obtain the maximum estimated value in state s2. This estimated value is calculated from the maximum reward in state s2. In other words, only the reward in the desired final state must be set to 1, while the rewards in other states are set to 0. After 600 iterations, the reward in the final state will be continuously transferred with each calculation of Q(s, a), thereby updating the Q table.

We used the above principles to design a new Q-learning algorithm. In this algorithm, we assume that the third instruction is more important and must be transmitted from path 2. To balance the flow and ensure that high-importance instructions are transmitted preferentially, we set the reward of the final state [2, 1, 2, 1, 2, 1] to 1 and the rewards of the other states to 0. Among them is s_n is 2, indicating that the nth instruction was transferred through channel 2, and s_n is 1. In other words, the nth instruction is transferred through channel 1 and s_n is 0, suggesting that the nth instruction was not transferred. The Q-value calculated after 600 iterations is shown in Table 2.

Table 2. Instruction priority Q table.

State	Left	Right
0, 0, 0, 0, 0, 0	0.00000	0.48107
2, 0, 0, 0, 0, 0	0.55321	0.37440
2, 2, 1, 0, 0, 0	0.00000	0.00000
2, 2, 1, 2, 0, 0	0.00000	0.00000
2, 2, 1, 2, 2, 2	0.00000	0.00000
1, 0, 0, 0, 0, 0	0.00000	0.00000
1, 2, 0, 0, 0, 0	0.00000	0.00000
1, 2, 1, 0, 0, 0	0.00000	0.00000
1, 2, 1, 2, 0, 0	0.00000	0.00000
1, 2, 1, 2, 1, 0	0.00000	0.00000
2, 1, 0, 0, 0, 0	0.51743	0.63506
2, 1, 2, 0, 0, 0	0.77159	0.52929
2, 1, 2, 1, 0, 0	0.64075	0.88270
...	...	...
2, 1, 2, 1, 2, 0	1.00000	0.70565
...	...	...

When the agent is in state [2, 1, 2, 1, 2, 0] and the reward [2, 1, 2, 1, 2, 1] is 1, the Q-target with the left action is 1. Therefore, the sixth instruction is transmitted through path 1 and reaches the target state [2, 1, 2, 1, 2, 1]. If the agent chooses the right action in the state [2, 1, 2, 1, 0, 0], it will receive a maximum estimated value of 1 [2, 1, 2, 1, 2, 0]. If the agent chooses the left action, it will receive a maximum estimated value of 0 [2, 1, 2, 1, 1, 0]. Therefore, the calculated Q-target of the right path is 0.88270, which is higher than the left path at 0.64075. When the fifth instruction passed through path 2, it reached state [2, 1, 2, 1, 2, 0].

Since the third instruction has higher priority than the other instructions, its routing weight coefficient was set to 2, whereas the other instructions were set to 1. As mentioned in Section 3.1, the transmission times of the right and left channels were 1 ms and 2 ms, respectively. As shown in Figure 5, the maximum transmission consumption is in the state [1, 2, 1, 2, 1, 2]: 11. The state [2, 1, 2, 1, 2, 1] is the transmission consumption of reinforcement learning: 10, which is reduced by 9.1%. Based on the design of the above algorithm, ship network instructions achieve the optimized goals of traffic balance and preferentially

deliver high-importance instructions through the path at the least cost. However, the current network is still relatively simple. The ship's network has more than two paths, more than one starting node, and one receiving node. Therefore, we increased the network structure's complexity for better application to the ship network.

3.3. Complex Networks

The Abilene dataset is commonly used in traffic engineering. The network structure of this dataset is 12 nodes and 30 paths. The network structure diagram, weight coefficient, and bandwidth of each path are shown in Figure 6 and Table 3.

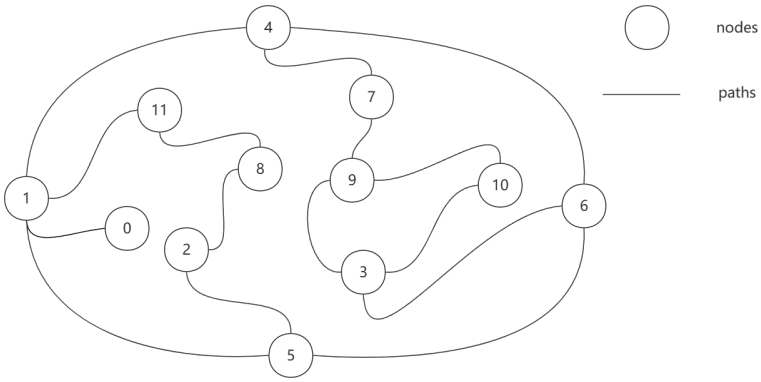

Figure 6. Abilene dataset network structure diagram.

Table 3. Abilene dataset weight coefficient and bandwidth.

Link_Index	Source	Destination	OSPF	Capacity (kbps)
0	0	1	1	9,920,000
1	1	0	1	9,920,000
2	1	4	1176	9,920,000
3	1	5	587	2,480,000
...	...	...	...	...
26	10	3	2095	9,920,000
27	10	9	861	9,920,000
28	11	1	846	9,920,000
29	11	8	233	9,920,000

According to the weight of each edge, the shortest path between each node can be calculated, as shown in Table 4.

Table 4. Shortest path in Abilene dataset.

Source -> Destination	Shortest Path
0 -> 1	0, 1
0 -> 2	0, 1, 5, 2
0 -> 3	0, 1, 5, 6, 3
0 -> 4	0, 1, 4
0 -> 5	0, 1, 5
0 -> 6	0, 1, 5, 6
...	...
11 -> 8	11, 8
11 -> 9	11, 8, 2, 5, 6, 3, 9
11 -> 10	11, 8, 2, 5, 6, 3, 10

As mentioned in Section 3.1, transmitting all flows through the shortest path causes partial path congestion and reduces the network's overall performance. Therefore, we employed reinforcement learning algorithms to reroute flows in the network and improve its overall performance. First, we calculated the reachable path between each node, as shown in Table 5.

Table 5. Abilene dataset reachable paths.

Source -> Destination	Shortest Path
0 -> 1	'0', '1'
0 -> 2	'0', '1', '4', '6', '5', '2' '0', '1', '4', '7', '9', '3', '6', '5', '2' '0', '1', '4', '7', '9', '10', '3', '6', '5', '2' '0', '1', '5', '2' '0', '1', '11', '8', '2'
...	...
11 -> 10	'11', '1', '4', '6', '3', '9', '10' '11', '1', '4', '7', '9', '3', '10' '11', '1', '5', '6', '3', '9', '10' '11', '1', '5', '6', '3', '10' '11', '1', '5', '6', '4', '7', '9', '3', '10' '11', '1', '5', '6', '4', '7', '9', '10' '11', '8', '2', '5', '1', '4', '6', '3', '9', '10' '11', '8', '2', '5', '1', '4', '6', '3', '10' '11', '8', '2', '5', '1', '4', '7', '9', '3', '10' '11', '8', '2', '5', '1', '4', '7', '9', '10' '11', '8', '2', '5', '6', '3', '9', '10' '11', '8', '2', '5', '6', '3', '10' '11', '8', '2', '5', '6', '4', '7', '9', '3', '10' '11', '8', '2', '5', '6', '4', '7', '9', '10'

The Abilene dataset collects the traffic matrix from the network path every 5 min, generating 288 pieces of data daily. Using datasets from 1 March 2004 to 4 March 2004, we simulated the transmission of 2016 pieces of data in the network. We calculated the utilization of each path and used average utilization as an evaluation indicator of the network's overall efficiency. For ease of presentation, only the experimental results of the three-node network are shown below. In the experiment, three nodes codenamed '1', '2', and '3' sent messages to each other. In reinforcement learning, there are six actions, '1 -> 2', '1 -> 3', '2 -> 1', '2 -> 3', '3 -> 1', '3 -> 2', that represent the packets to be rerouted. There are two paths between each node, as shown in Table 6.

Table 6. Reachable paths between nodes.

Source -> Destination	Path
1 -> 2	'1', '2' '1', '3', '2'
1 -> 3	'1', '3' '1', '2', '3'
2 -> 1	'2', '1' '2', '3', '1'
2 -> 3	'2', '3' '2', '1', '3'
3 -> 1	'3', '1' '3', '2', '1'
3 -> 2	'3', '2' '3', '1', '2'

Reinforcement learning has a total of [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 1] . . . [1, 1, 1, 1, 1, 1] 64 states using '0' and '1' to represent whether the messages between nodes are routed through the first or second path, respectively. The partial Q table after 600,000 iterations is shown in Table 7.

Table 7. Complex network Q table.

State	1 -> 2	1 -> 3	2 -> 1	2 -> 2	3 -> 1	3 -> 2
0, 0, 0, 0, 0, 0	8.00188	8.04001	8.02912	7.88910	7.82145	8.05650
. . .	. . .	. . .	. . .	. . .	. . .	. . .
0, 0, 0, 0, 0, 1	9.43658	9.43658	9.43658	9.43658	9.43658	9.43658
0, 0, 0, 1, 0, 1	10.99794	10.99794	10.99794	10.99794	10.99794	10.99794
. . .	. . .	. . .	. . .	. . .	. . .	. . .
1, 1, 1, 1, 1, 1	0.12408	0.12317	0.11300	0.00000	0.00000	0.00000
. . .	. . .	. . .	. . .	. . .	. . .	. . .

As seen in the Q table, 63 states are covered from state [0, 0, 0, 0, 0, 0] to state [1, 1, 1, 1, 1, 1], indicating that training was relatively sufficient. The occupancy rate of the edge with the highest occupancy rate in the network is 80.57% in the state [0, 0, 0, 0, 0, 0]. In the Q table, the 3 -> 2 packets rerouted from the '1' channel are transferred from the '2' channel. After rerouting, the network status becomes [0, 0, 0, 0, 0, 1] and the average utilization rate is 62.04%, which helps reduce network congestion and perform traffic engineering on complex networks.

The greedy algorithm is an algorithm that adopts a series of local optimal choices when solving problems. At each step, the greedy algorithm will choose the optimal solution in the current situation, without considering the possible consequences of that choice. In our problem, the greedy algorithm selects a path from the available paths for each packet it receives to route it, resulting in the lowest occupancy rate of the edge with the highest occupancy rate in the network.

The occupancy rate of the edge with the highest occupancy rate in the network is:

$$\text{Occupancy Rate (\%)} = \frac{\sum_{i=1}^{n} Weight_i}{\text{Maximum Capacity}} \times 100$$

The Max Occupancy rate of the edge with the highest occupancy rate in the network is:

$$\text{Max Occupancy Rate (\%)} = \max_{i=1}^{n} \text{Occupancy Rate}_i$$

Although greedy algorithms cannot guarantee obtaining the global optimal solution, they can usually find a solution close to the optimal solution in a more efficient way.

In Figure 7, The maximum occupancy rate of the edge with the highest occupancy rate in the network is 142.41%, the greedy algorithm reduced the occupancy rate of the edge with the highest occupancy rate in the network to 88.54%. The occupancy rate of the edge with the highest occupancy rate in the network using OSPF is 80.57%. After traffic engineering of the network through reinforcement learning, the occupancy rate of the edge with the highest occupancy rate in the network was reduced by 18.53% to 62.04%.

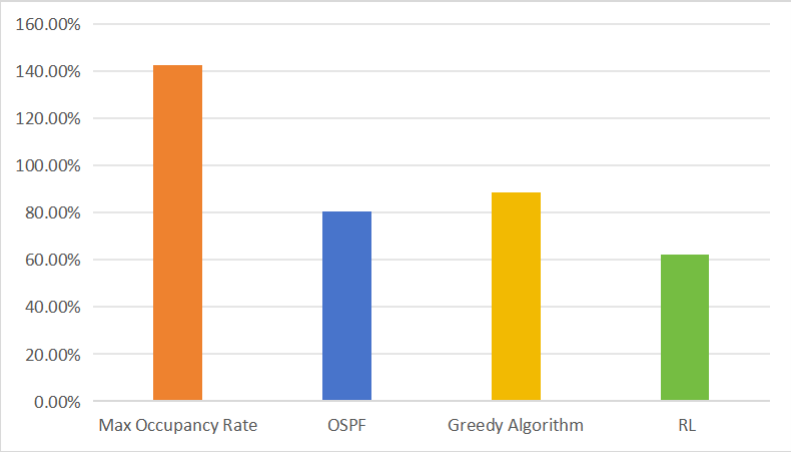

Figure 7. Occupancy rate: Max Occupancy Rate, OSPF, greedy algorithm, RL.

4. Discussion and Conclusions

In this study, we address the multiple challenges faced by ship networks, including limited bandwidth, unstable network connections, high latency, and command priority. To solve these problems, we chose the reinforcement learning method and conducted a simulation study of traffic engineering on the ship network. We focused on three aspects, namely, traffic balance, instruction priority, and complex network structure, to evaluate the reinforcement learning performance in these scenarios. The proposed algorithm for ship network traffic engineering aims to optimize network performance by balancing traffic load, prioritizing critical instructions, and adapting to dynamic network conditions. The algorithm seeks to mitigate congestion by optimizing traffic distribution and load balancing across network segments. As a result, the occurrence and severity of congestion-related issues, such as packet drops, increased latency, and degraded throughput, are reduced, leading to smoother and more efficient network operation. The algorithm's ability to prioritize high-importance instructions and optimize routing decisions contributes to minimizing latency in the network.

The experimental results show that applying reinforcement learning to ship networks greatly improves the network's quality of service (QoS). Specifically, reinforcement learning optimized network traffic balance, reasonably arranged instruction priorities, and coped with complex network structures. Through an in-depth analysis of the experimental data, we noticed network consumption was reduced by 9.1% under reinforcement learning, which implemented priority routing of high-priority instructions more effectively while reducing the occupancy rate of the edge with the highest occupancy rate in the network by 18.53%. Although the simulation experiment results demonstrated the positive effects of reinforcement learning, it is important to consider whether the experimental environment fully considers the complexity of the real ship network. Future work should consider adding validation with real maritime communication data to fully evaluate the applicability of reinforcement learning in real-world scenarios. Parameter selection in reinforcement learning algorithms can significantly impact the results. To ensure the robustness and reliability of the research results, future research can explore the selection of algorithm parameters and generalized performance under different ship network configurations.

It is important to acknowledge that RL is not a novel concept. However, its application to the specific context of ship network optimization represents a novel contribution. While the experimental setup may provide insights into the potential benefits of RL, it is crucial to recognize the limitations of the proposed structure and dataset. Real-world ship networks may exhibit complexities and dynamics not fully captured in the experimental setup. Therefore, while the study provides valuable insights into the feasibility of using RL for ship network optimization, further validation in real-world maritime environments is necessary to confirm its effectiveness and legitimacy. Additionally, future research should explore more diverse and realistic scenarios to ensure the generalizability of the findings.

Author Contributions: Conceptualization, X.Y. and M.L. (Minghui Liu); methodology, X.W. (Xinin Wang); software, M.L. (Meng Liu); validation, B.H.; funding acquisition, X.W. (Xiaomin Wang). All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key Research and Development Program under Grant 2022YFB3104600; the China Postdoctoral Science Foundation Funded Project (2023M740519); Medico-Engineering Cooperation Funds from University of Electronic Science and Technology of China (No. ZYGX2021YGLH213, No. ZYGX2022YGRH016); Interdisciplinary Crossing and Integration of Medicine and Engineering for Talent Training Fund, West China Hospital, Sichuan University under Grant No. HXDZ22010; the Municipal Government of Quzhou (Grant 2022D018, Grant 2022D029, Grant 2023D007, Grant 2023D015, Grant 2023D033, Grant 2023D034, Grant 2023D035); Zhejiang Provincial Natural Science Foundation of China under Grant No. LGF22G010009; and Guiding project of Quzhou Science and Technology Bureau (subject No. 2022005, 2022K50, 2023K013, 2023K016).

Data Availability Statement: The data that support the findings of this study are available from the corresponding author, Xiaomin Wang, upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

Nomenclature

The following terms are used in this manuscript:

Ship Network	A ship network refers to the communication and data transmission infrastructure deployed on a vessel.
Traffic Engineering	Traffic engineering focuses on optimizing the performance, reliability, and efficiency of communication networks.
Reinforcement Learning (RL)	Reinforcement learning enabling agents to adapt to complex and dynamic environments and achieve sophisticated goals autonomously.
Quality of Service (QoS)	Quality of Service (QoS) refers to the set of metrics and mechanisms used to measure and ensure the performance, reliability, and availability of network services.

References

1. Bhatti, J.; Humphreys, T.E. Covert control of surface vessels via counterfeit civil GPS signals. *J. Inst. Navig.* 2014, *unpublished*.
2. Tabatabaee, V.; Bhattacharjee, B.; La, R.J.; Shayman, M.A. Differentiated traffic engineering for QoS provisioning. In Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami, FL, USA, 13–17 March 2005; IEEE: Piscataway, NJ, USA, 2005; Volume 4, pp. 2349–2359.
3. Tangmunarunkit, H.; Govindan, R.; Shenker, S.; Estrin, D. The impact of routing policy on internet paths. In Proceedings of the IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213), Anchorage, AK, USA, 22–26 April 2001; IEEE: Piscataway, NJ, USA, 2001; Volume 2, pp. 736–742.
4. Wang, N.; Ho, K.H.; Pavlou, G.; Howarth, M. An overview of routing optimization for internet traffic engineering. *IEEE Commun. Surv. Tutorials* **2008**, *10*, 36–56. [CrossRef]
5. Sarkar, D.; Choudhury, S.; Majumder, A. Enhanced-Ant-AODV for optimal route selection in mobile ad-hoc network. *J. King Saud Univ. Comput. Inf. Sci.* **2021**, *33*, 1186–1201. [CrossRef]
6. Esmailpour, A.; Nasser, N. Dynamic QoS-based bandwidth allocation framework for broadband wireless networks. *IEEE Trans. Veh. Technol.* **2011**, *60*, 2690–2700. [CrossRef]
7. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep reinforcement learning: A brief survey. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38. [CrossRef]
8. Chen, S.; Liu, M.; Deng, P.; Deng, J.; Yuan, Y.; Cheng, X.; Xie, T.; Xie, L.; Zhang, W.; Gong, H.; et al. Reinforcement learning based diagnosis and prediction for COVID-19 by optimizing a mixed cost function from CT images. *IEEE J. Biomed. Health Inform.* **2022**, *26*, 5344–5354. [CrossRef] [PubMed]
9. Shen, Y.; Zhao, N.; Xia, M.; Du, X. A deep q-learning network for ship stowage planning problem. *Pol. Marit. Res.* **2017**, *24*, 102–109. [CrossRef]
10. Chou, Y.C.; Benjamin, C.O. An AI-based Decision Support System for Naval Ship Design. *Nav. Eng. J.* **1992**, *104*, 156–165. [CrossRef]
11. Veitch, E.; Alsos, O.A. A systematic review of human-AI interaction in autonomous ship systems. *Saf. Sci.* **2022**, *152*, 105778. [CrossRef]

12. Zhao, X.; Song, Y. Improved Ship Detection with YOLOv8 Enhanced with MobileViT and GSConv. *Electronics* **2023**, *12*, 4666. [CrossRef]
13. Yao, Q.; Wang, Y.; Yang, Y. Underwater acoustic target recognition based on data augmentation and residual CNN. *Electronics* **2023**, *12*, 1206. [CrossRef]
14. Thantharate, P.; Anurag, T. CYBRIA-Pioneering Federated Learning for Privacy-Aware Cybersecurity with Brilliance. In Proceedings of the 2023 IEEE 20th International Conference on Smart Communities: Improving Quality of Life Using AI, Robotics and IoT (HONET), Boca Raton, FL, USA, 4–6 December 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 56–61.
15. Ghomi, E.J.; Rahmani, A.M.; Qader, N.N. Load-balancing algorithms in cloud computing: A survey. *J. Netw. Comput. Appl.* **2017**, *88*, 50–71. [CrossRef]
16. Campbell, A.; Coulson, G.; Hutchison, D. A quality of service architecture. *ACM SIGCOMM Comput. Commun. Rev.* **1994**, *24*, 6–27. [CrossRef]
17. Clifton, J.; Laber, E. Q-learning: Theory and applications. *Annu. Rev. Stat. Its Appl.* **2020**, *7*, 279–301. [CrossRef]
18. Xie, T.; Wei, Y.; Xu, L.; Li, Q.; Che, F.; Xu, Q.; Cheng, X.; Liu, M.; Yang, M.; Wang, X.; et al. Self-supervised contrastive learning using CT images for PD-1/PD-L1 expression prediction in hepatocellular carcinoma. *Front. Oncol.* **2023**, *13*, 1103521. [CrossRef] [PubMed]
19. Pióro, M.; Szentesi, Á.; Harmatos, J.; Jüttner, A.; Gajowniczek, P.; Kozdrowski, S. On open shortest path first related network optimisation problems. *Perform. Eval.* **2002**, *48*, 201–223. [CrossRef]
20. Moy, J. Ospf Version 2. RFC2328, April 1998. Available online: http://tools.ietf.org/html/rfc2328 (accessed on 1 April 2024).
21. Coltun, R.; Ferguson, D.; Moy, J.; Lindem, A. Rfc 5340: Ospf for ipv6. 2008. Available online: https://dl.acm.org/doi/abs/10.17487/rfc5340 (accessed on 1 April 2024).
22. Adjih, C.; Baccelli, E.; Jacquet, P. Link state routing in wireless ad-hoc networks. In Proceedings of the IEEE Military Communications Conference, MILCOM 2003, Boston, MA, USA, 13–16 October 2003; IEEE: Piscataway, NJ, USA, 2003; Volume 2, pp. 1274–1279.
23. Zhu, Z.; Chen, M.; Zhu, C.; Zhu, Y. Effective defense strategies in network security using improved double dueling deep Q-network. *Comput. Secur.* **2024**, *136*, 103578. [CrossRef]