mathematics

MDPI

Article

Optimal Unmanned Combat System-of-Systems Reconstruction Strategy with Heterogeneous Cost via Deep Reinforcement Learning

Ruozhe Li, Hao Yuan, Bangbang Ren *, Xiaoxue Zhang, Tao Chen and Xueshan Luo

National Key Laboratory of Information Systems Engineering, National University of Defense Technology, Changsha 410073, China; liruozhe@nudt.edu.cn (R.L.); yuanhao@nudt.edu.cn (H.Y.); zxiaoxue@nudt.edu.cn (X.Z.); chentao@nudt.edu.cn (T.C.); xsluo@nudt.edu.cn (X.L.)
* Correspondence: renbangbang11@nudt.edu.cn; Tel.: +86-1937-412-6399

Abstract: The unmanned combat system-of-systems (UCSoS) in modern warfare is comprised of various interconnected entities that work together to support mission accomplishment. The soaring number of entities makes the UCSoS fragile and susceptible to triggering cascading effects when exposed to uncertain disturbances such as attacks or failures. Reconfiguring the UCSoS to restore its effectiveness in a self-coordinated and adaptive manner based on the battlefield situation and operational requirements has attracted increasing attention. In this paper, we focus on the UCSoS reconstruction with heterogeneous costs, where the collaboration nodes may have different reconstruction costs. Specifically, we adopt the heterogeneous network to capture the interdependencies among combat entities and propose a more representative metric to evaluate the UCSoS reconstruction effectiveness. Next, we model the combat network reconstruction problem with heterogeneous costs as a nonlinear optimization problem and prove its NP-hardness. Then, we propose an approach called SoS-Restorer, which is based on deep reinforcement learning (DRL), to address the UCSoS reconstruction problem. The results show that SoS-Restorer can quickly generate reconstruction strategies and improve the operational capabilities of the UCSoS by about 20∼60% compared to the baseline algorithm. Furthermore, even when the size of the UCSoS exceeds that of the training data, SoS-Restorer exhibits robust generalization capability and can efficiently produce satisfactory results in real time.

Keywords: unmanned combat system-of-systems; heterogeneous cost; optimal reconstruction strategy; deep reinforcement learning

MSC: 90-10

Citation: Li, R.; Yuan, H.; Ren, B.; Zhang, X.; Chen, T.; Luo, X. Optimal Unmanned Combat System-of-Systems Reconstruction Strategy with Heterogeneous Cost via Deep Reinforcement Learning. *Mathematics* **2024**, *12*, 1476. https://doi.org/10.3390/math12101476

Academic Editor: Andrea Scozzari

Received: 24 April 2024
Revised: 5 May 2024
Accepted: 8 May 2024
Published: 9 May 2024

1. Introduction

The rapid development of data links, satellite communications, and other technological innovations have changed the operational patterns of modern warfare. As a unique operational concept, mosaic warfare aims to realize the autonomous collaboration of various types of combat entities and construct an on-demand integrated, highly flexible, and adaptable system-of-systems (SoS) to meet the requirements of combat tasks [1,2]. The UCSoS concept refers to the integrated use of various homogeneous or heterogeneous unmanned combat platforms, including unmanned aircraft, unmanned vehicles, and unmanned underwater vehicles, to create a new form of distributed, networked, and collaborative combat force [3–5]. Under the mosaic warfare concept, the relationship between different combat entities in the UCSoS is no longer fixed but adaptively transformed based on combat tasks and evolving battlefield situations.

The UCSoS is susceptible to interference in hostile battlefields [6], e.g., being attacked, loss of communication between members, and malfunction [7], especially when critical

entities are destroyed, resulting in a significant decrease in the operational capability of the UCSoS. Due to the high development cost of combat entities, it is often difficult to produce more combat entities to replace the destroyed entities. Furthermore, repairing destroyed entities in a confrontational environment is difficult, making redundancy or repair strategies less likely to improve operational capability. To address these issues, researchers focus on the information flow between entities in the UCSoS and propose the dynamic reconstruction of the collaborative relationship between entities [8–11]. This method reconfigures the collaborative relationships of surviving entities to maintain reliable operational capabilities so that the UCSoS can regain the operational loop of observation, orientation, decision-making, and action (OODA) again and achieve efficient decision-centered operations. There are related studies that refer to the UCSoS reconstruction process as system-of-systems (SoS) resilience [12].

However, regarding the current research advancements in the reconstruction of the entity link relationship, most studies assume that the reconstruction costs of all links are identical. Hence, the total reconstruction cost is only limited by the number of added edges. In fact, the reconstruction cost corresponding to added edges may be heterogeneous. In general, the reconstruction cost of edges is mainly related to the operational capabilities of the entities connected by the edges. For instance, the cost of collaboration between two highly capable aircraft is often much higher than the cost of collaboration between two soldiers. In the case of the limited reconstructed resources and environment, it is both realistic and meaningful to consider the inclusion of heterogeneous costs in the strategy or model. In this paper, we focus on the problem of UCSoS reconstruction with heterogeneous costs. Our goal is to determine how to balance the reconstruction efficiency against the reconstruction cost, and to quickly find the optimal reconstruction strategy.

Nevertheless, it will face the following challenges when solving the UCSoS reconstruction problem with heterogeneous costs: *(i) How to evaluate the effect of UCSoS reconstruction?* There are often dependencies, connections, and interactions between entities in the UCSoS that show distinct functional and emergent characteristics at the SoS level. [13]. Thus, the method used to describe the characteristics of the UCSoS and the metric used to measure the effect of the reconstruction will directly determine the cost allocation strategy. *(ii) How to balance the reconstruction efficiency against the reconstruction cost?* Usually, reconstructing the edges between the nodes with higher capability can effectively restore the operational capability, but it will also consume higher costs. A reasonable reconstruction strategy should find the optimal edge or the edge with lower cost, and the reconstruction may improve the UCSoS performance more. However, the existing methods that focus on homogeneous cost make it difficult to find a low-cost and efficient set of key edges. *(iii) How to quickly generate reconstruction strategies in uncertain and large-scale scenes?* In real battlefield scenarios, due to the inability to know which nodes in the UCSoS will be attacked by the adversary, the obtained fragmented structure of the UCSoS is uncertain. It is challenging to readjust the connection between surviving combat entities in real time based on the current state of the UCSoS. In addition, the UCSoS reconstruction problem is essentially a variant of the portfolio decision problem [14,15], which means that the UCSoS reconstruction problem is a combinatorial optimization problem and also an NP-hard problem. Traditional algorithms, such as approximation algorithms and metaheuristic algorithms, are usually used to solve the portfolio decision problem, but it is difficult to achieve a satisfactory balance between solution time and solution quality in large-scale cases. In summary, designing a solution approach for UCSoS reconstruction that solves the above problems is both attractive and challenging.

Inspired by recent advances in solving large-scale optimization problems using deep reinforcement learning [16–18], we propose a novel method called DRL-Restorer to solve the UCSoS reconstruction problem with heterogeneous costs as quickly and accurately as possible. This method integrates deep learning and the performance recovery of UCSoS to make decisions more intelligent. In summary, the main achievements of this paper can be summarized as follows:

- We treat the UCSoS as a heterogeneous combat network (HCN) and incorporate the cost of edge reconstruction into the UCSoS reconstruction problem. The HCN reconstruction problem with heterogeneous costs is formulated as a nonlinear optimization problem.
- A task-oriented UCSoS operational capability metric is proposed to comprehensively consider the coverage and balance of striking enemy targets and the operational capability of the force. The metric can effectively describe the operational effectiveness of the UCSoS and provide an optimization goal for the reconstructed model.
- We propose an innovative approach to the UCSoS reconstruction problem called DRL-Restorer, which uses a deep neural network and the actor-critic algorithm to generate the optimal solution in real time. The proposed method is more suitable for dynamic combat scenarios that require real-time UCSoS reconstruction.
- With extensive experiments, the results demonstrate that DRL-Restorer exhibits significant advantages over the benchmark algorithm in terms of solution quality and scalability, while also finding optimal reconstruction strategies in a remarkably short time.

This remainder is organized as follows. Section 2 describes related work focusing on the UCSoS reconstruction and deep reinforcement learning. Section 3 presents the heterogeneous network model and the operational capability measure of the UCSoS. The problem of UCSoS reconstruction and the mathematical model are presented in Section 4. Then, the SoS-Restorer method is described in detail in Section 5. Experimental results are performed to verify the proposed approach in Section 6. Finally, the conclusion of this paper is given in Section 7.

2. Related Work

In this section, we first review the related work on UCSoS reconstruction. Then, we present the application of deep reinforcement learning to combinatorial optimization problems.

2.1. UCSoS Reconstruction

Recently, many scholars have studied the development of combat networks and UCSoS, but the results are mixed. Li et al. conducted a series of studies on heterogeneous combat networks, including HCN robustness [19], HCN operational capability disintegration [20], and temporal combat network operational capability evaluation [21]. The above studies consider the weakening of combat network capabilities in various attack scenarios and propose protection measures for critical entities to enhance the resilience of the UCSoS. However, the study fails to mention methods for restoring the operational capability of the UCSoS under disruptive conditions such as malfunctions or failures. Establishing system or functional redundancy, using backup nodes or parallel functional paths to replace failed nodes or subsystems, is a common strategy for restoring the efficiency of a failed SoS. A system reliability optimization model based on a hybrid redundancy strategy has been proposed by Zhang et al. [22]. Levitin et al. [23] optimized the number of components of an active redundancy system that performs a mission with multiple attempts. Peiravi et al. [24] proposed a redundancy allocation strategy that aims to increase system reliability by determining the most appropriate time for reconfiguration and activating an optimal number of components. However, in adversarial battlefields, the reasons for disrupting the UCSoS have become increasingly ambiguous and unpredictable [25]. In addition, the high cost of producing redundant combat units makes it difficult to counter interference through the implementation of redundancy strategies. A strategy is then proposed that leverages the collaborative relationships among surviving nodes to restore operational capability. In response to stochastic external disturbances and dynamic reconfiguration, Chen et al. [9] presented a mission reliability model for unmanned weapon SoS based on the operational loop. Sun et al. [10] proposed a method framework called UCSoS autonomous decision-making to enable SoS with reconfiguration capability amidst battlefield disturbances. Zhong et al. [26] aimed to restructure the collaborative relationships between

network nodes and optimize the selection of kill chains in the operational system, thereby enhancing its resilience to disruptions.

However, despite existing research that considers the reconstruction of cooperation relationships between entities to restore reliable operational capability, these studies mostly ignore the heterogeneity of costs required to achieve synergies between different entities and their impact on the reconstruction problem. Furthermore, these studies are limited to addressing small-scale issues that improve the resilience of the SoS. However, the battlefields of the future will be composed of massive unmanned systems, which urgently require an algorithmic design that efficiently addresses the problem of UCSoS reconstruction in large-scale scenarios. The summary of the UCSoS reconstruction problem is shown in Table 1.

Table 1. Summary of the UCSoS reconstruction problem.

Reference	Research Content	Research Deficiencies
References [19–21]	Evaluate and disintegrate the operational capabilities of the UCSoS	No mention of how to restore the operational capabilities of the UCSoS
References [22–24]	Adopt a redundancy strategy to restore the efficiency of the failed SoS	Too expensive for UCSoS to build a redundancy strategy
References [9,10,26]	Reconstruct the collaborative relationship between survival combat entities to restore the operational capabilities of the UCSoS	Ignore the heterogeneity of collaboration costs between different entities

2.2. Deep Reinforcement Learning in Combinatorial Optimization Problems

Combinatorial optimization problems (COP) have wide applications in various fields, including national security, industrial manufacturing, and intelligent transportation [27]. These problems involve finding the best solution from a finite set of objects and are known to be NP-hard. Examples of common COP include the traveling salesman problem (TSP) [28], the knapsack problem [29], and the job-shop scheduling problem [30]. The traditional approach to solving COP mainly uses exact techniques [31] and approximate methods [32]. However, traditional approaches struggle to quickly generate optimal COP solutions as the complexity of real-world problems increases and the demand for real-time solutions grows. In addition, these traditional methods rely on iterative search algorithms and cannot learn from historical data. Consequently, any change in data for the same problem requires a new search and solution process, increasing computational complexity.

To tackle these challenges, researchers have proposed a new paradigm for solving COP [33] using end-to-end DRL: training a deep neural network to generate solutions to the given problem. The parameters of the neural network are typically optimized by using a set of problem instances that are in the same category as the problem.

First, Hopfield et al. [34] presented the application of neural networks in solving COP by performing experiments on a small TSP problem. However, in a fresh instance of the TSP problem, there is a need for re-training of the neural network parameters, which has no advantage over heuristic methods. To address this problem, Vinyals et al. proposed a sequence-to-sequence approach, i.e., the famous pointer network model [35], which achieved satisfactory results and triggered a wave of using deep neural networks to tackle COP. Due to the challenge of obtaining training labels for supervised learning methods, Bello et al. [36] used reinforcement learning to train the pointer network model and incorporated a critic network as a reference point to reduce training variance, which demonstrated its scalability in solving TSP and KnapSack problems. On the contrary, when it comes to tackling COP using a graph structure such as the minimum vertex cover problem (MVC), Dai first introduced a graph neural network called structure2vec to solve COP [37].

Furthermore, Li et al. [38] employed graph convolutional networks (GCN) and guided tree search techniques to tackle both the MVC and maximum independent set problems (MIS). This approach effectively handles scenarios where multiple optimal solutions exist. The summary of the DRL method for COP is shown in Table 2.

Table 2. Summary of the DRL method for COP.

Reference	Model	Training Method	Solving Problems
Reference [35]	Pointer network	Supervised training	TSP problem
Reference [36]	Pointer network	REINFORCE and critic baseline	TSP and KnapSack problems
Reference [37]	Structure2vec	DQN	MVC problem
Reference [38]	GCN	Guided tree search	MVC and MIS problems

Although the DRL method has been successfully applied to other combinatorial optimization problems, it faces specific challenges when dealing with the UCSoS reconstruction problem. Therefore, an appropriate design in addition to the basic methods of DRL is needed.

3. UCSoS Model

In this section, we review the heterogeneous network model of UCSoS and then propose the operational capability measurement method of UCSoS.

3.1. Heterogeneous Network Model of UCSoS

Definition 1 ([39] Heterogeneous Network)**.** *Given a network $G = (V, E)$, where V denotes the set of nodes and E denotes the set of edges. Define two types of mapping functions, (i) $\varphi : V \to \mathcal{A}$, where for each node $v \in V$ there exists a certain node type $\varphi(v) \in \mathcal{A}$, (ii) $\Psi : E \to \mathcal{R}$, where for each edge $e \in E$ there exists a specific link type $\Psi(e) \in \mathcal{R}$. If the number of node types $|\mathcal{A}| > 1$ or the number of link types $|\mathcal{R}| > 1$, then G is considered a heterogeneous network.*

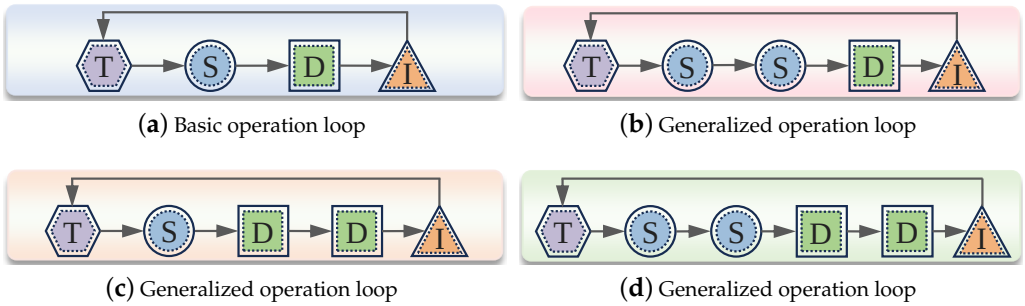
Definition 2 ([40] Meta-Path)**.** *For a heterogeneous network G, the meta-path $\mathcal{P}$ can be described as a path based on the network schema $\mathcal{T}_G = (\mathcal{A}, \mathcal{R})$, which represents a series of links between nodes A_1 and A_{l+1}: $A_1 \xrightarrow{R_1} \ldots \xrightarrow{R_l} A_{l+1}$, where node types $A_i \in \mathcal{A}$, $i \in \{1, 2, \ldots l+1\}$, and link types $R_i \in \mathcal{R}$, $i \in \{1, 2, \ldots l\}$.*

According to Cares's information age combat model [41], the UCSoS entities can be divided into four types based on their different functions on the battlefield:

- Sensor Entities (S): entities that carry out reconnaissance, detection, and early warning assignments.
- Decider entities (D): entities that carry out command and control missions.
- Influence entities (I): entities that carry out the precision strike, fire damage, and electronic interference functions.
- Target entities (T): enemy combat entities, including sensors, deciders, and influence entities, can all be considered targets on the battlefield.

A military engagement is a loop process in which the enemy target is detected by the sensor entities ($T \to S$) and the information from the enemy target is transmitted to the decider entities for information analysis ($S \to D$); the decider entities make operational decisions and give the order to fire at the influence entities ($D \to I$); then the influence entities execute the orders and attack the enemy targets ($I \to T$). Therefore, the concept of an operation loop is introduced based on the four entities of the UCSoS, and the operation loop can be divided into basic and generalized categories based on the number of entities. The basic operation loop contains only a sensor entity (S), a decider entity (D), an influence entity (I), and an enemy target (T) to fulfill the processes of observation, orientation,

decision-making, and action, which can be formed as $T \rightarrow S \rightarrow D \rightarrow I \rightarrow T$. In contrast, the generalized operational loop considers additional types of links, such as information sharing between sensor entities ($S \rightarrow S$) and between decider entities ($D \rightarrow D$). Figure 1 shows four types of operational loops.



(**a**) Basic operation loop (**b**) Generalized operation loop

(**c**) Generalized operation loop (**d**) Generalized operation loop

Figure 1. Four types of operational loops. Different types of entities are represented by graphics of different colors.

Based on the above, a UCSoS can be abstracted as a heterogeneous combat network $G = (V, E)$, where $V = S \cup D \cup I \cup T = \{v_1, v_2, \ldots, v_N\}$ is a node set representing entities of the UCSoS, and the edge set $E = E^{S \rightarrow S} \cup E^{S \rightarrow D} \cup E^{D \rightarrow D} \cup E^{D \rightarrow I} \cup E^{I \rightarrow T} \cup E^{T \rightarrow S} = \{e_{ij}\}$ is the flow of information between entity v_i and v_j. We consider different types of entities possessing distinct capabilities during the operation process: (i) sensor entities: reconnaissance capability; (ii) decider entities: decision-making capability; (iii) influence entities: strike capability; (iv) target entities: anti-reconnaissance capability.

3.2. Operational Capability Measurement of UCSoS

The operational loop depicts a full operational trajectory from reconnaissance to enemy target destruction, with each loop representing a mode of attack or kill chain against the target. The greater the number of operational loops constructed against the target, the more diverse the methods available to destroy the target entity [42]. Thus, we can measure the operational capability of the UCSoS by analyzing and synthesizing the capabilities of all the operational loops in the UCSoS.

For an operational loop consisting of multiple sensor (decider and influence) nodes, the reconnaissance (decision-making and strike) capability of the operational loop can be considered to be the sum of the capabilities of these sensor (decider and influence) nodes. Let l_j be an operational loop including a target node v_t, a sensor node set $S = \{v_s\}$, a decider node set $D = \{v_d\}$, and an influence node set $I = \{v_i\}$, then the operational capability of l_j can be expressed as follows according to Ref. [21]:

$$U(l_j) = \frac{1}{|l_j|} \times q(v_t) \times \sum_{v_s \in S} q(v_s) \times \sum_{v_d \in D} q(v_d) \times \sum_{v_i \in I} q(v_i) \tag{1}$$

where $|l_j|$ is the length of the operational loop. $q(v_s)$, $q(v_d)$, and $q(v_i)$ denote the capability values of our equipment, while $q(v_t)$ represents the anti-reconnaissance capability values of the target entities.

For an HCN G with a set of operational loops $L_G = \{l_j\}$, the usual approach to evaluating its operational capability $f(G)$ is by the weighted summation method, which is computed using Equation (2):

$$f(G) = \sum_{v_t \in T} \sum_{l_j \in L_G} w(v_t) \times U(l_j) \tag{2}$$

where $w(v_t)$ represents the weight of the target node in the l_j loop.

However, using Equation (2) to calculate the operational capability of UCSoS often poses a problem: most of our weapons will focus on the target nodes with higher weights,

while ignoring the coverage and balance of the destruction of enemy targets. As shown in Figure 2, the operational capability of the HCN constructed in the two scenarios is the same. However, compared to scenario 1, the operational loop constructed in scenario 2 can cover more enemy targets ($T1$ and $T2$), which means we can destroy more enemy combat forces and the chance of victory is greater. Therefore, when evaluating the operational capability of HCN, the range for attacking enemy targets should also be considered. Thus, we have

$$\Gamma(G) = f(G) \times \sum_{v_t \in T} \sigma(v_t) \tag{3}$$

where $\sigma(v_t)$ is a binary variable used to determine whether the target node v_t is inside the loop. If it is, then $\sigma(v_t) = 1$; otherwise it is equal to 0. Obviously, the operational capability of scenario 1 and scenario 2 in Figure 2 calculated by Equation (3) is 3.90 and 7.80, respectively. Higher values correspond to a more destructive operational effect.

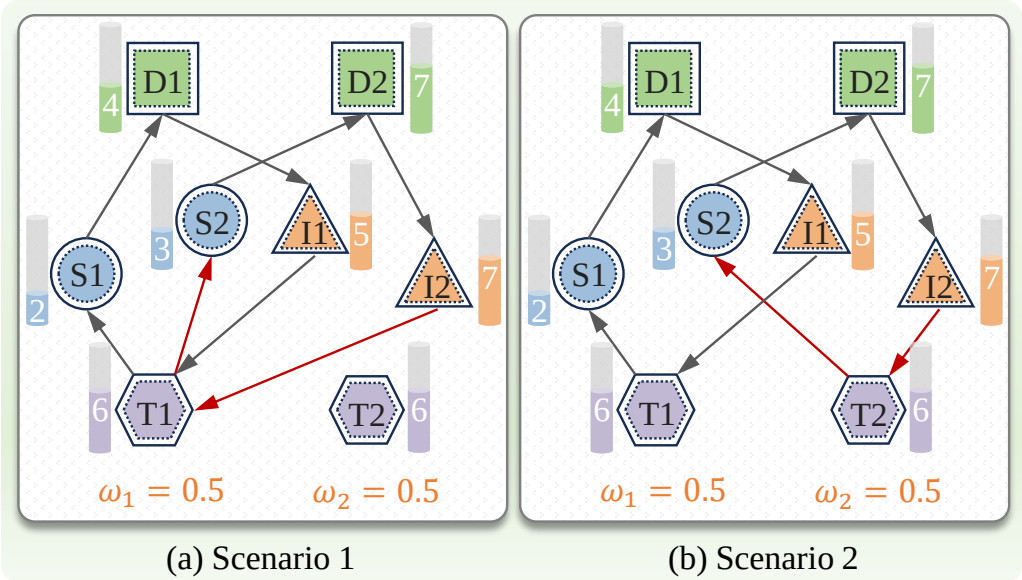


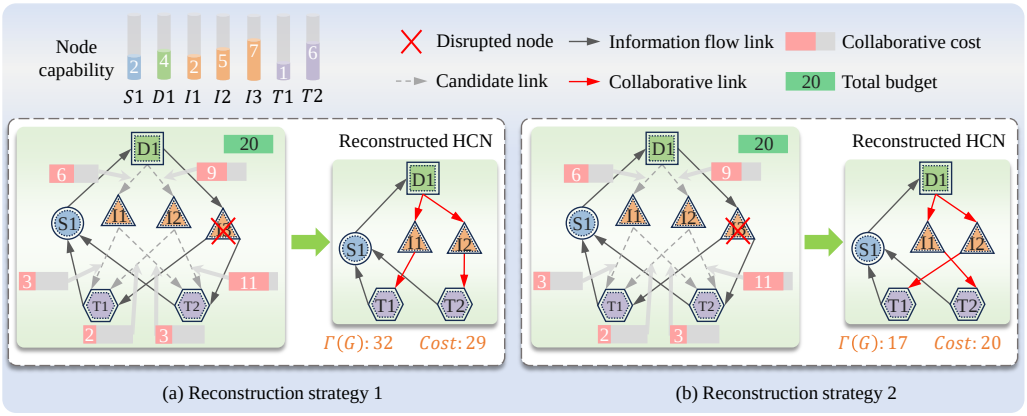
Figure 2. Two different scenarios for the construction of HCN. The color bar represents the capability value of the node. w_1 and w_2 represent the weights of the target nodes T_1 and T_2, respectively. (**a**) In scenario 1, two operational loops $T1 \to S1 \to D1 \to I1 \to T1$ and $T1 \to S2 \to D2 \to I2 \to T1$ are constructed for target $T1$. The operational capability of HCN calculated by Equation (2) is 3.90. (**b**) In scenario 2, the operational loops $T1 \to S1 \to D1 \to I1 \to T1$ and $T2 \to S2 \to D2 \to I2 \to T2$ are constructed for target $T1$ and $T2$, respectively. The operational capability of HCN calculated by Equation (2) is 3.90.

4. UCSoS Reconstruction Problem with Heterogeneous Costs

In this section, we present the mathematical model of the UCSoS reconstruction problem with heterogeneous costs, including the problem illustration, problem model, and complexity analysis.

4.1. Problem Illustration

Battlefield strategy requires the integrated and synchronized use of all relevant capabilities in response to changes in posture. As the disrupted entities on the battlefield are difficult to repair, the cooperative relationship of the surviving entities can be locally reorganized to enhance the combat effectiveness of the UCSoS. Figure 3 illustrates the diagram of the UCSoS reconstruction process.



(a) Reconstruction strategy 1 (b) Reconstruction strategy 2

Figure 3. Two different reconstruction strategies. (**a**) The link set $\{E^{D_1 \to I_1}, E^{D_1 \to I_2}, E^{I_1 \to T_1}, E^{I_2 \to T_2}\}$ is chosen to reconstruct the broken network. The cost of reconstruction and the network performance after reconstruction are 29 and 32, respectively. (**b**) The link set $\{E^{D_1 \to I_1}, E^{D_1 \to I_2}, E^{I_1 \to T_2}, E^{I_2 \to T_1}\}$ is chosen to reconstruct the broken network. The cost of reconstruction and the network performance after reconstruction are 20 and 17, respectively.

As shown in Figure 3, there is a UCSoS composed of a sensor node, a decider node, three influence nodes, and two target nodes. Each node in the UCSoS has different capability values. Without loss of generality, the capabilities of $\{S_1, D_1, I_1, I_2, I_3, T_1, T_2\}$ are $\{2, 4, 2, 5, 7, 1, 6\}$, respectively. Unfortunately, node I_3 is attacked by the enemy, resulting in the inability to form a complete operational loop against the targets T_1 and T_2. We assume that once an entity is attacked, the edges connected to the entity are deleted. At this point, we can add the cooperative relationship between the surviving entities to form the operational loop. Currently, there are six candidate links and the construction of each link requires different costs. The collaboration costs of the links $\{E^{D_1 \to I_1}, E^{D_1 \to I_2}, E^{I_1 \to T_1}, E^{I_1 \to T_2}, E^{I_2 \to T_1}, E^{I_2 \to T_2}\}$ are $\{6, 9, 3, 3, 2, 11\}$, respectively.

There are two different cost allocation strategies for reconstructing the UCSoS. In strategy 1, the links $\{E^{D_1 \to I_1}, E^{D_1 \to I_2}, E^{I_1 \to T_1}, E^{I_2 \to T_2}\}$ are added to the previous UCSoS with a total cost of 29. In strategy 2, the links $\{E^{D_1 \to I_1}, E^{D_1 \to I_2}, E^{I_1 \to T_2}, E^{I_2 \to T_1}\}$ are added to the previous UCSoS with a total cost of 20. With Equation (3), we can easily calculate that the operational capabilities of the UCSoS after executing strategies 1 and 2 are 32 and 17, respectively. However, despite the superior reconstruction effectiveness of strategy 1, it cannot ultimately be implemented because the reconstruction cost exceeds the budget $(29 \geq 20)$.

Through the above examples, the problem of UCSoS reconstruction can be formally defined.

Definition 3. *Given a broken heterogeneous UCSoS $G_0 = (V, E_0)$, where each node has a force type and a capability value. Suppose that the network of G_0 with all candidate edges added can be represented by $G_1 = (V, E_1)$. Therefore, the set of candidate links E_2 is equal to $E_1 - E_0$ and the construction of each link requires a cost. Then, the UCSoS reconstruction problem is to add appropriate links to G_0 from the set of candidate links E_2 at finite cost, so as to maximize the operational capability of the UCSoS.*

4.2. Problem Model

According to the above problem illustration, the optimization model of the UCSoS reconstruction problem can be established.

The broken UCSoS can be described as a heterogeneous combat network $G_0 = (V, E_0)$ with node set $V = \{v_1, v_2, \ldots, v_N\}$ and edge set $E_0 = \{e_{ij}\}, \forall i, j \in V$. Define the adjacency matrix of G_0 as $A(G_0) = (a_{ij})_{N \times N}$, where $a_{ij} = 1$ if v_j is connected to v_i. The state of the node v_i can be described by a tuple $v_i := \langle t_i, q_i, r_i^{in}, r_i^{out} \rangle$. Specifically, (i) t_i represents the type of force of node v_i, (ii) q_i denotes the capability of node v_i to correspond to the type of

force, and (iii) r_i^{in} and r_i^{out} represent the maximum amount of information the node v_i can receive and send, respectively. In general, the larger the operational capability of a node, the more information it can receive or send:

$$r_i^{in} = \alpha_i q_i, i = 1, \cdots, N, \alpha_i > 0 \tag{4}$$

$$r_i^{out} = \beta_i q_i, i = 1, \cdots, N, \beta_i > 0 \tag{5}$$

where α_i and β_i are random disturbance values, indicating that external factors such as electromagnetic interference affect the throughput of nodes.

In this paper, reconstruction methods only consider the edge increase strategy. Generally, the collaborative cost between nodes is related to the capability between nodes. The nodes with higher capability imply their higher importance on the battlefield, requiring more manpower and material resources for protection against enemy interference, which eventually leads to an increase in the cost of cooperation between nodes. Assuming that c_{ij} is the cost of collaboration between nodes v_i and v_j, then the calculation formula is as follows:

$$c_{ij} = \left(\frac{q_i + q_j}{2}\right)^p \tag{6}$$

where $p \geq 0$ is called the cost-sensitive parameter. The reconstruction cost c_{ij} of each edge is identical when $p = 0$. The larger p value implies that such a reconstruction cost is more sensitive.

Let $G_1 = (V, E_1)$ be a fully connected network formed by adding all candidate edges to G_0. We denote the set of added links by $\hat{E} \subseteq E_1 - E_0$, and the network after reconstruction by $\hat{G} = (V, E_0 + \hat{E})$. Let $n = \left|\hat{E}\right|$ denote the reconstruction strength. The reconstruction strategy is denoted by $X = (x_{ij})_{N \times N}$, where $x_{ij} = 1$ if $e_{ij} \in \hat{E}$; otherwise, $x_{ij} = 0$. Then, we have

$$n = \sum_{i=1}^{N} \sum_{j=1}^{N} x_{ij} \tag{7}$$

It is easy to see that the cost of the reconstruction strategy X should satisfy the following constraints:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} x_{ij} c_{ij} \leq C_{\max} \tag{8}$$

where $C_{\max}$ is the total budget for reconstruction costs.

To ensure that the number of received messages for our equipment after reconstruction does not exceed its maximum value, we have

$$\sum_{i=1}^{N} (a_{ij} + x_{ij}) \leq r_i^{in}, \forall j \tag{9}$$

To make sure that the number of messages sent by our equipment after reconstruction does not exceed its maximum value, we have

$$\sum_{j=1}^{N} (a_{ij} + x_{ij}) \leq r_i^{out}, \forall i \tag{10}$$

As introduced in Section 3.2, we define the effect of the reconstruction strategy as the increase of the operational capability after edge addition $\Phi(X) = \frac{\Gamma(\hat{G}) - \Gamma(G_0)}{\Gamma(G_1)}$. The objective of UCSoS reconstruction is to find a set of reconstruction strategies X^* that can maximize the recovery of the operational capability of UCSoS.

To sum up, the optimization model for the reconstruction problem defined in Definition 3 can be denoted as follows:

$$\begin{aligned} \max \quad & \Phi(X = (x_{ij})_{N\times N}) \\ \text{s.t.} \quad & (1) \sim (10) \end{aligned} \tag{11}$$

4.3. Complexity Analysis

To prove that the UCSoS reconstruction problem characterized by Model (11) is NP-hard, we first present the definition of a well-known NPC problem, i.e., the 0-1 backpack problem [43].

Definition 4. *The* 0–1 backpack problem *involves two sets of non-negative integers,* $C = \{c_1, c_2, ..., c_m\}$ *and* $U = \{u_1, u_2, ..., u_m\}$*, together with an integer* b*. The goal is to decide whether a subset* $s \subseteq \{1, 2, \ldots, m\}$ *exists such that* $\sum_{j\in s} c_j \leq b$ *and* $\sum_{j\in s} u_j$ *is maximized.*

Theorem 1. *The UCSoS reconstruction problem formulated in Model (11) is NP-hard.*

Proof. Assuming that there are three integer values $\{q_T, q_D, q_I\}$, then we can construct a special UCSoS problem case from a 0-1 backpack problem instance. As shown in Figure 4, the UCSoS G_0 is composed of a target entity T_1, a decider entity D_1, an influence entity I_1, $m+1$ sensor entities $\{S_0, S_1, S_2, ..., S_m\}$, and total budget C_{max}. Initially, there is only one meta-path in the UCSoS, i.e., $T_1 \to S_1 \to D_1 \to I_1$. $\{q_T, q_D, q_I\}$ represent the node capabilities of $\{T_1, D_1, I_1\}$, respectively. q_{S_i} represents the node capability of S_i. Then the operational capability of the UCSoS G_0 is

$$\Gamma(G_0) = f(G) = \frac{1}{3} \times q_T \times q_{S_0} \times q_D \times q_A \tag{12}$$

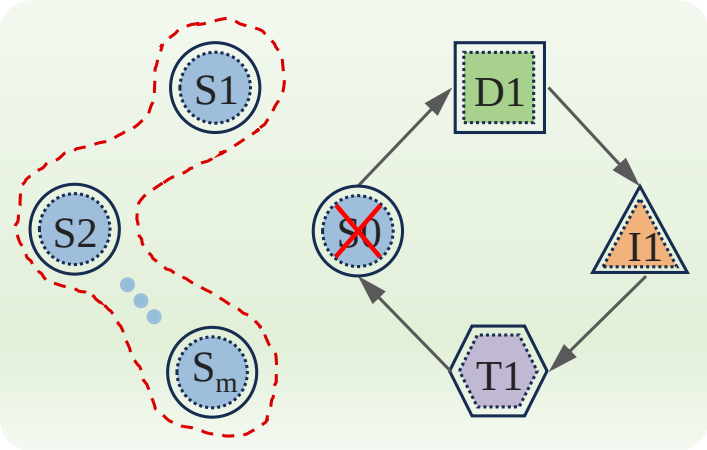


Figure 4. A simple example to show that the UCSoS reconstruction problem is NP-hard.

Next, assume that node S_0 in Figure 4 is destroyed and the cost-sensitive parameter $p = 1$. Then the cost of absorbing the nodes $\{S_1, S_2, ..., S_m\}$ into the UCSoS G_0 is $\{\frac{q_T+q_{S_1}}{2} + \frac{q_{S_1}+q_D}{2}, \frac{q_T+q_{S_2}}{2} + \frac{q_{S_2}+q_D}{2}, ..., \frac{q_T+q_{S_m}}{2} + \frac{q_{S_m}+q_D}{2}\}$.

The optimization objective of model 11 is to maximize $\Phi(X) = \frac{\Gamma(\hat{G})-\Gamma(G_0)}{\Gamma(G_1)}$, which equals to maximize $\Gamma(\hat{G})$, i.e.,

$$\Gamma(\hat{G}) = \frac{1}{3} \times q_T \times \sum_{S_i\in\hat{S}} q_{S_i} \times q_D \times q_A \tag{13}$$

where $\hat{S}$ represents the newly added sensor nodes. Furthermore, maximizing $\Gamma(\hat{G})$ equals to maximizing $\sum_{S_i\in\hat{S}} q_{S_i}$. In other words, the UCSoS problem instance G_0 can be reduced

to a 0-1 backpack problem defined in Definition 4, where $C = \{\frac{q_T+q_{S_1}}{2} + \frac{q_{S_1}+q_D}{2}, \frac{q_T+q_{S_2}}{2} + \frac{q_{S_2}+q_D}{2}, ..., \frac{q_T+q_{S_m}}{2} + \frac{q_{S_m}+q_D}{2}\}$, $U = \{q_{S_1}, q_{S_2}, ..., q_{S_m}\}$ and $b = C_{max}$. As the 0-1 backpack problem is NPC, the UCSoS reconstruction problem formulated in Model (11) is NP-hard. □

5. The Design of SoS-Restorer

In this section, we propose the deep learning-based approach SoS-Restorer to find the optimal solution for the UCSoS reconstruction problem. First, we present the overall framework of SoS-Restorer with severe-key procedures. Next, we describe the structure of the encoder-decoder neural network with an attention mechanism. Finally, we give the specific training procedure of SoS-Restorer.

5.1. General Overview

A complex UCSoS reconstruction problem can be viewed as a Markov decision process (MDP) [44]. At each iteration step, the agent selects a link from the optional link set as an action based on the current state and updates the state. In addition, the DRL method excels in executing sequential decision tasks without prior domain knowledge about the system [45], so it is appropriate to choose the DRL method to solve the problem of UCSoS reconstruction. In this paper, we use the famous actor-critic (AC) [46] method in DRL. In the AC method, actors are tasked with developing strategies to optimize cumulative returns. At the same time, the critic evaluates the actor-generated strategy and provides a value function to facilitate strategy updates. As illustrated in Figure 5, the SoS-Restorer solution process includes three stages: preparation, selection, and mapping.

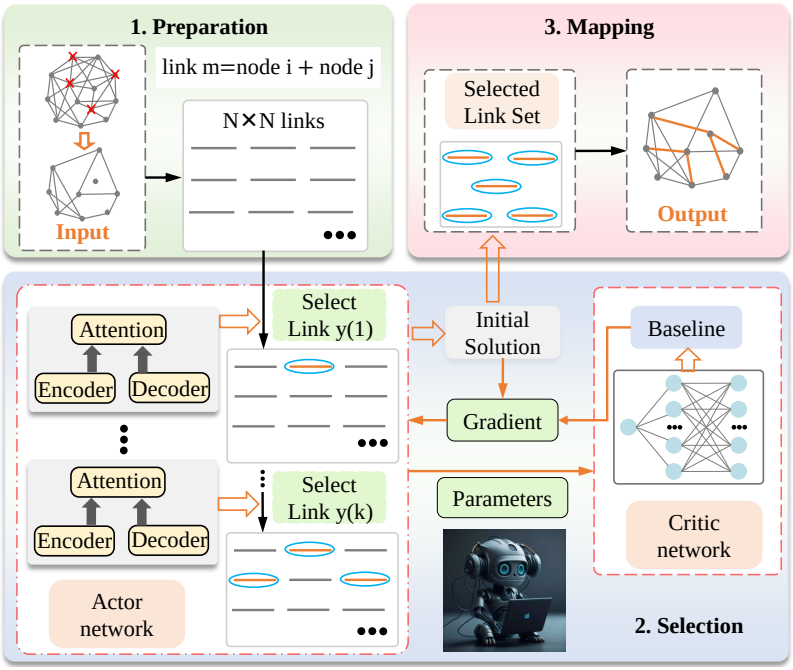


Figure 5. The general framework of SoS-Restorer. In the preparation stage, for a broken UCSoS with N nodes, any two nodes are connected to form $N \times N$ links. In the selection stage, we will select the solution to the link composition problem with the highest probability based on the deep neural network model until a complete solution is constructed. In the mapping phase, by mapping the selected link to the broken UCSoS, we can calculate the operational capability of the reconstructed UCSoS.

5.1.1. Stage I: Preparation

In the preparation stage, we pair the N surviving nodes currently distributed on the battlefield, resulting in the generation of $N \times N$ links. As shown in Figure 6, each link

consists of two nodes and can be viewed as an optional candidate action. In particular, the link $\mathcal{L}_k$ can be represented as $\mathcal{L}_k := \langle v_i, v_j \rangle$, which means that the node v_i collaborates with the node v_j.

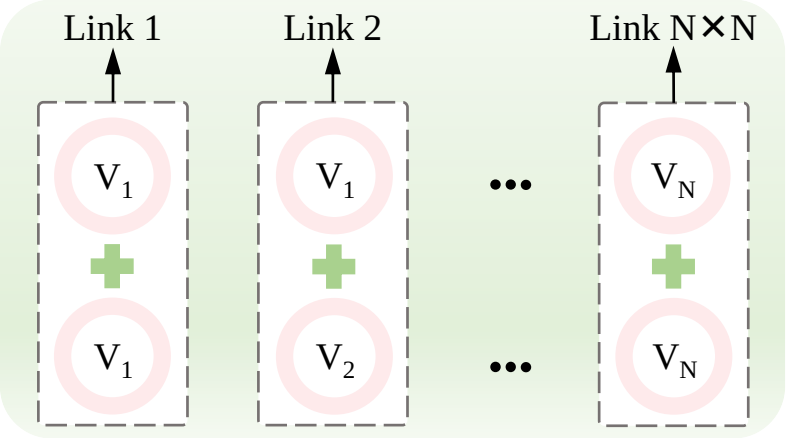


Figure 6. Input structure of the UCSoS reconstruction problem. The information contained in the input sequence link contains the input information of two nodes.

5.1.2. Stage II: Selection

In the selection stage, the neural network consisting of an encoder, decoder, and attention mechanism is used to output the optimal solution to the reconstruction problem. First, the encoder extracts the feature information of all the candidate links and generates their embedding vectors (i.e., high-dimensional vectors). These embedding vectors are then passed to the decoder as input to the decoding process. The decoding process aims to select the optimal subset of links from the candidate link set to form a solution to the problem. In each decoding step, we combine the decoder and the attention mechanism to compute the probability of each candidate link being selected, and then the agent selects the link with the largest probability as the action of that step. Once a link is selected, it is removed from the candidate link set. The above operation is repeated until the maximum reconstruction cost budget constraint is reached. The neural network architecture is described in detail in Section 5.2, while the training process of the neural network parameters is presented in Section 5.3.

5.1.3. Stage III: Mapping

After the selection phase, we can obtain the selected set of links. By mapping the selected links to the broken UCSoS, we can easily obtain the structure of the reconstructed UCSoS, and then calculate the reconstruction effect through the objective function Φ.

A key issue when using the DRL method is how to design the state, action, and reward functions. In our problem, we define the state S as the set of selected links. At time t, if the agent has already selected $t-1$ links, then the state of the environment can be represented as $s_t = \{\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_{t-1}\}$, $s_t \in S$. When designing the action space, we consider the selection of a link as an action, so the size of the action space is obviously $N \times N$. Furthermore, the definition of the reward function is identical to the objective function of the optimization problem, which is the operational capability value of the UCSoS after reconstruction.

Through the above design of the SoS-Restorer, we can easily obtain the solution to the UCSoS reconstruction problem. During the training phase, we use the reward value of the solution as feedback provided from the environment to perform backpropagation and adjust the network parameters. Once the loss value of the network parameters and the reconstruction solution stabilize or meet our expectations, we can obtain a well-trained network structure. During the application phase, we can use the trained network to quickly find the optimal solution to the reconstruction problem by inputting the broken UCSoS

and the reconstruction budget information. As shown in Figure 7, the main steps of the proposed SoS-Restorer method can be summarized as follows:

- Step 1: Initialize the trained parameters of the deep neural network, the fragmented UCSoS structure, and the total reconstruction budget.
- Step 2: The nodes in the fragmented UCSoS are paired in pairs to generate candidate links.
- Step 3: Termination condition: The iteration stops when the sum of costs for selected links exceeds the total reconstruction budget.
- Step 4: According to the deep neural network, calculate the probability of selecting each node.
- Step 5: Select the link with the highest probability using a greedy strategy.
- Step 6: If the termination condition is not met, go to step 3.

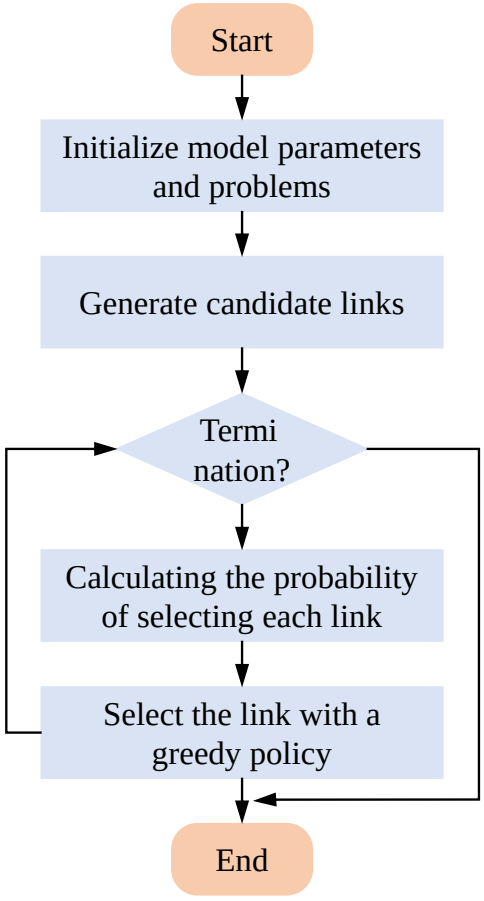


Figure 7. The flowchart of the proposed SoS-Restorer.

5.2. The Neural Network Architecture Model

Recently, in the field of natural language processing, Vinyals et al. proposed a pointer network architecture based on the sequence-to-sequence (Seq2Seq) model to solve COP tasks [47], which shows remarkable progress in terms of solution speed and quality. Both the encoder and the decoder of the pointer network use recurrent neural networks (RNNs). Inspired by the pointer network model, we propose a neural network architecture to address the UCSoS reconstruction problem. The neural network architecture in the AC framework consists of an encoder, a decoder, and attention components, as depicted in Figure 8.

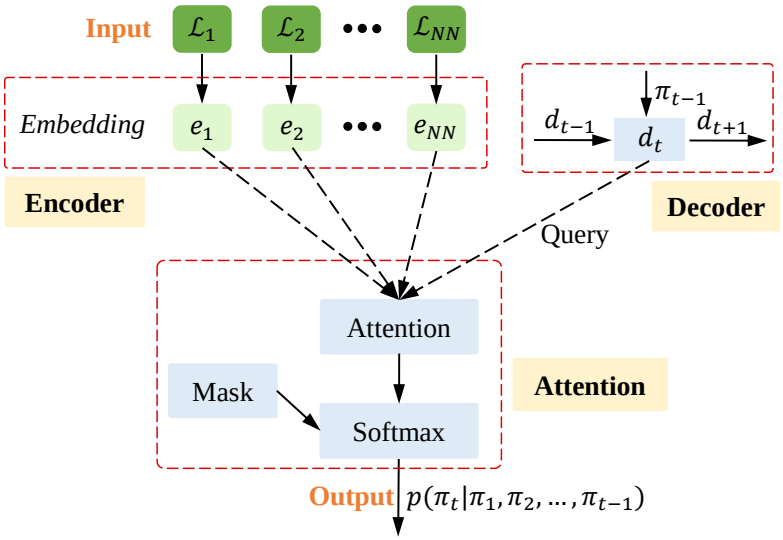


Figure 8. Our neural network architecture model. The encoder extracts features of input links through the CNN network, while the RNN decoder is responsible for storing information about the decoding sequence. The attention mechanism is used to generate the probability distribution for the next input based on the RNN hidden state and the embedded information.

5.2.1. The Encoder

The role of the encoder is to learn all the features of the input sequence, enabling the agent to comprehend the representation of each link. In pointer networks, the encoder adopts the RNN structure, which effectively captures the sequential information of the input sequence. However, for the UCSoS reconstruction problem, the input order of the links does not provide any valuable information since any random permutation contains the same information as the original input. Thus, to reduce computational complexity without compromising efficiency, one-dimensional convolutional neural networks (CNN) are used as the encoding neural networks. Specifically, each input link $\mathcal{L}_i$ is encoded and transformed into e_i, thereby generating a $(N * N) \times d_h$-dimensional matrix $E = \{e_1, e_2, \ldots, e_{NN}\}$, where $N * N$ and d_h represent the length of the input sequence and the dimensionality of the target vector, respectively.

5.2.2. The Decoder

The task of the decoder is to convert the high-dimensional vector generated by the encoder into the output sequence. Unlike the encoding stage, the decoding process needs to consider the sequential output order of the selected links. Therefore, it is appropriate to choose an RNN with a memory function as the decoding network. At each time step t, the decoding network uses its previous hidden state d_{t-1} and input π_{t-1} (i.e., the output sequence π_{t-1} at time step $t-1$) to generate the current hidden state d_t for querying by the attention layer.

5.2.3. The Attention

The traditional Seq2Seq model is only suitable for dealing with problems where the output dimensions are fixed, but it cannot deal with situations where the output dimensions change dynamically. Fortunately, Vinyals has successfully incorporated the attention mechanism into the Seq2Seq model, which effectively solves the problem of the uncertain output dimension [35]. For the defined UCSoS reconstruction problem, when the size of the UCSoS problem or the reconstruction budget is changed, the dimensions of the output links will also change accordingly. Therefore, we also introduce the attention mechanism into the neural network architecture.

As shown in Figure 8, the probability of selecting each link in the current step can be computed using the attention mechanism. Specifically, at step t, the weighted sum of the

hidden state d_t and the embedding vector e_j is computed, and then the result is passed through the tanh activation function as shown in Equation (14):

$$u_j^t = v^T tanh(W_a e_j + W_b d_t), j \in \{1, 2, \ldots, NN\} \tag{14}$$

where the parameters v, W_a and W_b can be obtained through training.

To reduce the action space and improve model efficiency, we introduced a masking mechanism during the problem-solving process of the neural network output. The mask is used to determine whether each link can be selected and takes a value of either 0 or 1, as follows:

$$mask_j^t = \begin{cases} 1 & \text{, if } o_j \text{ is valid at time } t \\ 0 & \text{, otherwise} \end{cases} \tag{15}$$

Finally, the final probability distribution of each link at time step t is calculated by:

$$p(\pi_t \mid \pi_0, \pi_1, \ldots, \pi_{t-1}, s_t) = softmax(u^t + mask^t) \tag{16}$$

During the training phase, to ensure that the link with the lower probability has a chance of being selected, we use importance sampling way to select the next action. However, during the testing phase, we adopt a policy of greedily selecting the link with the highest probability. The detailed algorithm of the proposed neural network architecture model is presented in Algorithm 1.

Algorithm 1: Processing procedure of the SoS-Restorer.

Input: All link information $\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_{NN}\}$
Output: The final output sequence $\Pi = \{\pi(t)\}_1^T$

```
Calculate the embedding e_i for i ∈ {1, 2, ..., NN} by the Encoder
Start current position π(0) ← 0
for t = 1 : T do
    Update the hidden layer state d_t by the Decoder
    Calculate the value u_j^t for j ∈ {1, 2, ..., NN}
    Change the mask value mask_j^t for j ∈ {1, 2, ..., NN}
    Calculate the output probability of each link
      p(π_t | π_0, π_1, ..., π_{t-1}, s_t) = softmax(u^t + mask^t)
    π(t) = Greedy (p(π_t | π_0, π_1, ..., π_{t-1}, s_t)) if choose greedy mode
end
```

5.3. Training Procedure

The AC framework consists of an actor network and a critic network. The actor network, consisting of an encoder, a decoder, and an attention mechanism, which is a modified version of the pointer network proposed in this article, aims to generate the probability distribution of feasible actions at the current step. Meanwhile, the structure of the critic network is similar to that of the encoder in the actor network and aims to evaluate the state value of a given problem instance. Our goal is to obtain optimal network parameters during the training phase to output the best reconstruction strategy during the application phase.

In particular, if the actor network parameters are defined as θ and the input problem instance is given as s, then the training goal of the actor network parameters can be described as follows:

$$J(\theta \mid s) = \mathbb{E}_{\Pi \sim p_\theta(\cdot \mid s)} \Phi(\Pi \mid s) \tag{17}$$

where $\Phi(\Pi \mid s)$ represents the objective value of the optimal reconstructed link sequence Π for problem instance s.

We train the network parameters using the classical policy gradient method [48]:

$$\nabla_\theta J(\theta \mid s) = \mathbb{E}_{\Pi \sim p_\theta(\cdot \mid s)}[(\Phi(\Pi \mid s) - b(s)) \nabla_\theta \log(p_\theta(\Pi \mid s))] \tag{18}$$

where $b(s)$ is a benchmark independent of Π that is used to estimate the expected value of the problem instance s, with the goal of reducing training variance.

During training, the Monte Carlo technique is employed to sample problem instances $s_1, s_2, ..., s_D \sim S$, and then the average value of these samples is computed as a substitute for the expected value:

$$\nabla_\theta J(\theta) \approx \frac{1}{D} \sum_{i=1}^{D} (\Phi(\Pi_i \mid s_i) - b(s_i)) \nabla_\theta \log(p_\theta(\Pi_i \mid s_i)) \tag{19}$$

To enhance learning efficiency, it is appropriate to use a parameterized baseline to estimate the objective. Therefore, we introduce an auxiliary network called the critic network to learn the excepted objective value that is found by the current policy p_θ given a state. Let φ be the parameters of the critic network, and the loss of the critic network can be obtained as follows:

$$\mathcal{I}(\varphi) = \frac{1}{D} \sum_{i=1}^{D} \nabla_\varphi \left\| \Phi(\Pi_i \mid s_i) - b_\varphi(s_i) \right\|^2 \tag{20}$$

where $b_\varphi(s_i)$ is the output baseline of the critic.

In each iteration, we sequentially optimize the actor and critic parameters. The actor parameters θ are updated based on the output of the critic, ensuring that the agent learns appropriate strategies in a positive sense. The detailed iterative procedures are illustrated in Algorithm 2.

Algorithm 2: Training process in the AC framework

Input: Training set S, training epoch E, initialize actor network parameter θ and critic network parameter φ
Output: Optimal parameters θ^*, φ^* after training

for $epoch \leftarrow 1, 2, \ldots, E$ **do**
  Sample D instances from S
  for $i \leftarrow 1, 2, \ldots, D$ **do**
    Initialize $t \leftarrow 0$
    repeat
      Select π^i_{t+1} based on the given probability distribution $p(\pi^i_t \mid \pi^i_0, \pi^i_1, \ldots, \pi^i_t, s^i_t)$
      Update state $s^i_{t+1} \leftarrow (s^i_t, y^i_{t+1})$
      $t \leftarrow t + 1$
    until *satisfy iterative condition*;
    Calculate the objective value $\Phi(\Pi_i \mid s_i)$ and estimated value $b_\varphi(s_i)$
  end
  Update the parameters θ and φ:
  $d\theta \leftarrow \frac{1}{D} \sum_{i=1}^{D} (\Phi(\Pi_i \mid s_i) - b(s_i)) \nabla_\theta \log(p_\theta(\Pi_i \mid s_i))$
  $d\varphi \leftarrow \frac{1}{D} \sum_{i=1}^{D} \nabla_\varphi (\Phi(\Pi_i \mid s_i) - b_\varphi(s_i))^2$
end

6. Performance Evaluation

In this section, we first describe the details of our simulation setup and benchmark algorithms. Then, we conducted extensive experiments to evaluate the performance of the proposed SoS-Restorer model. Finally, we summarize the experimental results.

6.1. Simulation Setup

6.1.1. Dataset

As the UCSoS reconstruction problem formulated by Model (11) is a military problem, publicly available datasets are not currently available. Therefore, we generate the dataset independently.

Specifically, the HCN was randomly generated based on the operational rules, consisting of 40% sensor nodes, 30% decider nodes, 20% influence nodes, and 10% target nodes. Six types of edges are used to represent the information interaction between nodes in HCN. The capability values of the nodes are sampled from a uniform distribution $[1, 2]$. The random interference values α_i and β_i corresponding to node v_i are randomly selected from the interval $\left[0, \frac{N}{2}\right]$, where N is the number of nodes in the initially constructed network. Meanwhile, the cost-sensitive parameter p is set to 1 to indicate a linear relationship between the reconstruction cost of each edge and the capability of the corresponding node.

To validate the effectiveness of the SoS-Restorer method in restoring HCN capability, we considered the random attack pattern as the disturbance condition to obtain fragmented HCN. Since the complete removal of nodes of the same type would result in a reduction of HCN capabilities to zero and prevent the construction of a new operational loop, we set the number of randomly attacking HCN nodes as follows: removal of 40% of sensor nodes, 30% of decider nodes, and 20% of influence nodes.

6.1.2. Hyperparameter Setting

For the actor network, all experiments use a single-layer 1D convolutional network as the encoder and a single hidden-layer GRU recurrent neural network as the decoder. The number of hidden nodes in both the encoder and the decoder is 128. In addition, we use a multilayer 1D convolutional network as the critic network. The model parameters are trained using the Adam optimizer. The fixed learning rate is 10^{-4}.

Based on the above settings, we trained a SoS-Restoer model consisting of 20 HCN nodes. The training process utilized a dataset consisting of one million data points.

6.1.3. Device Configuration

The experiments are performed on a computer configured with CPU: Intel i9-12900K 3.2GHz; GPU: NVIDIA RTX3090; RAM: 64GB; OS: 64-bit Ubuntu 16.04. The software employed for development is Python 3.7.

6.2. Benchmarks

To accurately evaluate the performance of SoS-Restorer when faced with critical nodes in complex networks under attack, we selected three classical repair strategies as our benchmark algorithms [49,50].

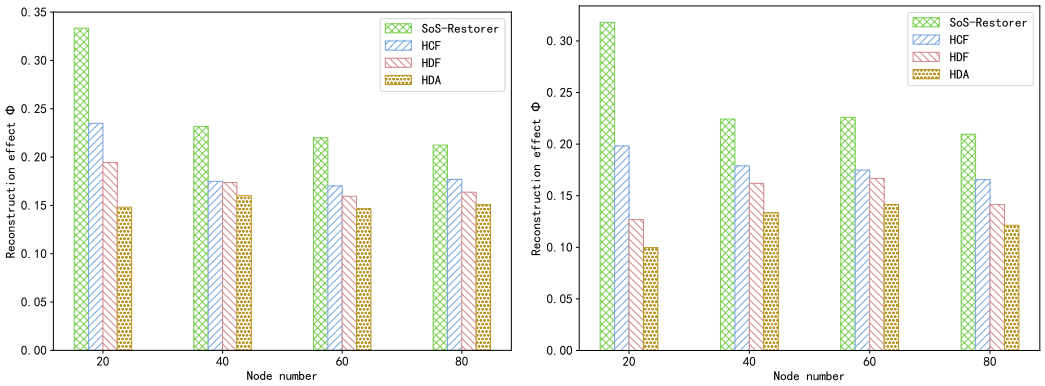
- **High Capability First (HCF):** Sort the nodes in descending order of capability and prioritize reconstructing the collaboration relationships between nodes with the highest capability.
- **High Degree First (HDF):** Sort the nodes in descending order of degree and prioritize reconstructing the collaboration relationships between nodes with the highest degree.
- **High Degree Adaptive (HDA):** Sort the nodes in descending order of degree, reconstruct the collaborative relationships between the nodes with the highest degrees, and then recalculate the degrees of each node. Repeat the above steps continuously.

6.3. Performance Results

6.3.1. Solving Quality and Speed

The performance of solutions generated by different methods was systematically compared through experiments conducted in two reconstruction cost scenarios: homogeneous cost ($p = 0$) and heterogeneous cost ($p = 1$). For each scenario, experiments were conducted on various network scales, including $\{20, 40, 60, 80\}$ network nodes. Ten problem instances

were randomly generated for each specific network scale, and the average reconstruction effect of each algorithm was calculated. The solving quality of different algorithms under the two reconstruction cost scenarios is illustrated in Figure 9, where a larger target value Φ indicates better reconstruction results.



(**a**) Homogeneous cost $p = 0$ (**b**) Heterogeneous cost $p = 1$

Figure 9. The average reconstruction effect in two reconstruction cost scenarios by different algorithms.

As depicted in Figure 9, we can see that the quality of the solutions generated by the SoS-Restorer is superior to that generated by the benchmark method in both the homogeneous cost and heterogeneous cost scenarios. Specifically, for the homogeneous cost scenario, regardless of the problem size, SoS-Restorer consistently outperforms the baseline algorithm in terms of solution quality, and the accuracy of the SoS-Restorer can be improved by about 20 $\sim$ 41% when compared to the well-performing HCF algorithm. For the heterogeneous cost scenario, the accuracy of the SoS-Restorer can also be improved by about 25 $\sim$ 60% compared to the HCF algorithm. Furthermore, regardless of the problem size, the quality of the solution generated by the HCF method is always superior to the HDF and HDA methods, indicating that reconstructing relationships between nodes with higher capability leads often restores higher operational capability. Figure 10 visualizes the solution to the HCN reconstruction problem; when faced with the same input instances, SoS-Restorer can output HCN reconstruction strategies that better align with the needs of the battlefield commander.

We plotted the box plots of solution time for SoS-Restorer in two different reconstruction cost scenarios, as shown in Figure 11. It can be observed that regardless of the scenario, the SoS-Restorer can provide problem-solving solutions within a reasonable time frame. As the size of the problem increases, SoS-Restorer adapts to the complexity by increasing its solution time but still manages to generate battle plans for the commander to decide within a few seconds.

6.3.2. Generalization Ability

A robust generalization capability allows the model to effectively adapt to novel situations and to outperform its performance based on the training data alone. During the training phase, SoS-Restorer is trained using instances that have a fixed range of information throughput and operational capability for the HCN node. However, in real-world combat scenarios, variations in the external environment, such as enemy electromagnetic jamming and changes in terrain conditions, can have a significant impact on the physical attributes of nodes. When there is a change in the attribute of the node in the SoS, it is necessary to adjust the reconstruction strategy accordingly. From the above experiments on solving quality and speed, we can conclude that regardless of network size, SoS-Restorer can quickly produce high-quality solutions without the need for retraining. Therefore, SoS-Restorer shows strong generalization ability when faced with networks of different sizes. Next, we will adjust node attributes, such as node capacity and throughput range, to make

them inconsistent with the node attribute range of the training phase, so that evaluate the generalization performance of SoS-Restorer when network node attributes change.

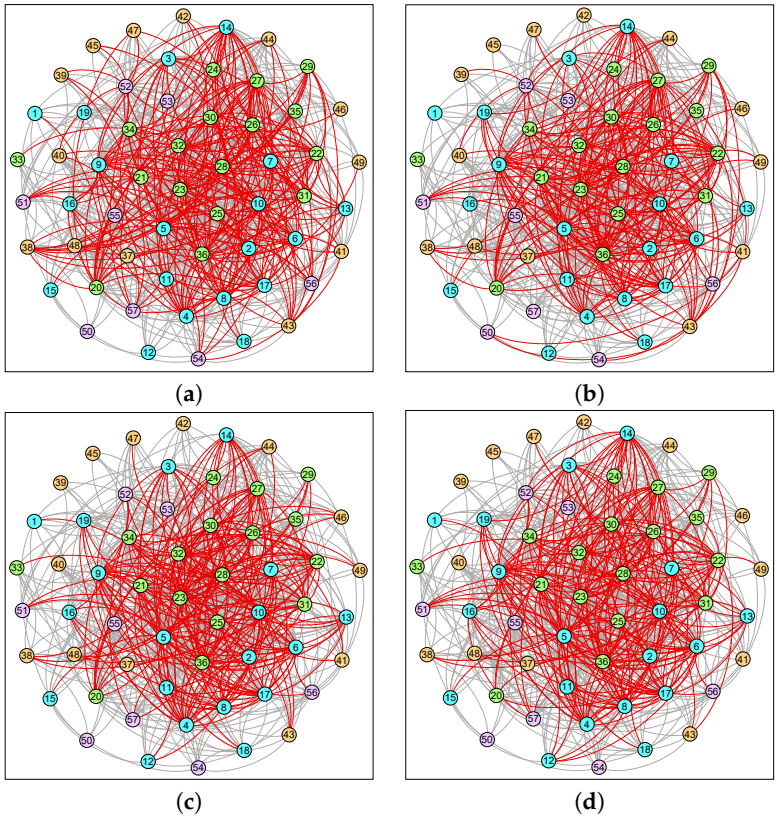


Figure 10. Visualization of the reconstruction results for different algorithms. Bright blue nodes represent sensor entities, vivid green nodes represent decider entities, light yellow nodes indicate influence entities, and pale purple nodes denote target entities. Light gray border represents existing edges, light red border represents edges added during reconstruction. (**a**) Solution to the HCN reconstruction generated by SoS-Restorer. (**b**) Solution to the HCN reconstruction generated by HCF. (**c**) Solution to the HCN reconstruction generated by HDF. (**d**) Solution to the HCN reconstruction generated by HDA.

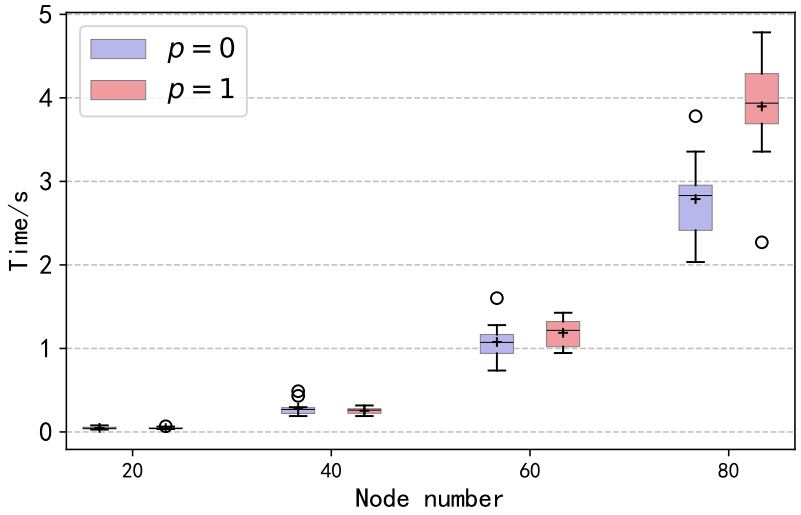


Figure 11. The solution time of SoS-Restorer for two reconstruction cost scenarios at the same scale.

Specifically, we modified the range of operational capability and throughput for SoS nodes and then compared the average result of each algorithm on 10 problem instances

when the cost-sensitive parameter $p = 1$. The results are shown in Table 3. We can see that regardless of adjusting the range of node attributes, the solution quality of SoS-Restorer is always better than the traditional methods. At the same time, we also drew a box plot of the solving time for SoS-Restorer in specific scale problems as shown in Figure 12. It is clear that SoS-Restorer can also find optimal solutions quickly.

Table 3. The average solution effect of each algorithm at different scales.

Algorithm	Node20	Node40	Node60	Node80
SoS-Restorer	0.2988	0.1910	0.1814	0.2103
HCF	0.1539	0.1483	0.1620	0.1793
HDF	0.1579	0.1186	0.1183	0.1246
HDA	0.1068	0.0737	0.1032	0.1141

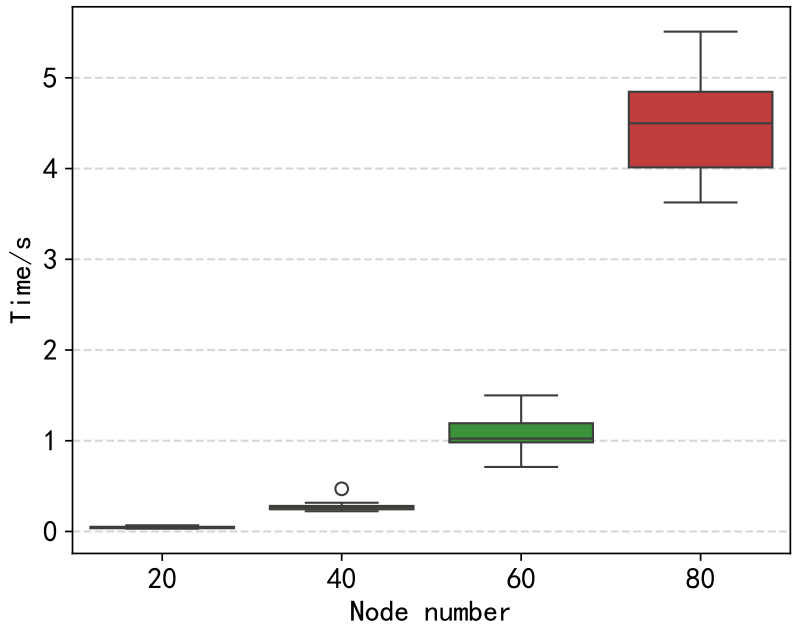


Figure 12. The solution time of SoS-Restorer when cost-sensitive parameter $p = 1$. Box plots of different colors represent the average solution time of SoS-Restorer at different problem sizes.

6.4. Summary of the Results

From the experimental results, we can conclude that SoS-Restorer can solve the UCSoS reconstruction problem efficiently and effectively. Compared to the baseline algorithm, SoS-Restorer shows some encouraging advantages, such as demonstrating strong generalization ability and finding high-quality solutions in a reasonable time frame, which can be summarized as follows:

- Strong generalization ability: Once the network parameters of the proposed model have been trained, it can be applied to new problems without needing re-training. In our experiments, our method can accurately find the optimal reconstruction strategy regardless of whether the reconstruction cost is homogeneous ($p = 0$) or heterogeneous ($p = 1$) and regardless of the size of the problem instances.
- Achieving an optimal balance between solution speed and solution quality: Another advantage of SoS-Restorer is that the optimization solution can be obtained directly by simple forward propagation of deep neural networks. Therefore, the reconstructed solution can always be found in a reasonable time while ensuring the quality of the solutions.

7. Conclusions

With the rapid iteration of advanced technology, the UCSoS has become increasingly fragile and inevitably subject to interference. Thus, how to quickly reorganize existing simple-functioning operational units into new UCSoS is a challenging task. In this paper, in order to facilitate the modeling and optimization of the UCSoS reconstruction problem with heterogeneous costs, we first abstract it as an HCN that includes different types of entities and links and improve the measurement method for evaluating the effectiveness of UCSoS reconstruction. Then, inspired by recent advances in the field of deep reinforcement learning, we propose a DRL-based method called SoS-Restorer to quickly find optimal reconstruction strategies. The personalized design of SoS-Restorer ensures its outstanding performance in terms of its problem-solving quality, solution speed, and generalization ability. Evaluation results show that SoS-Restorer can improve the solution quality by about 20∼60% compared to traditional algorithms. In addition, SoS-Restorer can provide reconstructed problem solutions within a few seconds to meet the demands of real-time reconstruction in dynamic combat scenarios.

Author Contributions: Conceptualization, R.L. and H.Y.; methodology, R.L., H.Y. and X.Z.; software, R.L. and H.Y.; validation, R.L., H.Y. and X.Z.; formal analysis, R.L., H.Y., X.Z. and B.R.; investigation, R.L., H.Y., X.Z., B.R., T.C. and X.L.; resources, X.Z. and B.R.; writing—original draft preparation, R.L., H.Y. and B.R.; writing—review and editing, R.L., H.Y., X.Z., B.R., T.C. and X.L.; supervision, X.Z., B.R., T.C. and X.L.; funding acquisition, X.Z. and X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the National Natural Science Foundation of China (No. 62302510). The authors are thankful for the support of the COSTA: complex system optimization team of the College of System Engineering at NUDT.

Data Availability Statement: Data will be made available on request.

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

References

1. Sapaty, P.S. Mosaic warfare: From philosophy to model to solutions. *Int. J. Robot. Autom* **2019**, *2019*, 157–166. [CrossRef]
2. Clark, B.; Patt, D.; Schramm, H. *Mosaic Warfare Exploiting Artificial Intelligence and Autonomous Systems to Implement Decision-Centric Operations*; Center for Strategic and Budgetary Assessments (CSBA): Washington, DC, USA, 2020.
3. Zhang, Y.; Guo, Q.-S.; Fan, Y.-P. Research on Operational Effectiveness Test Evaluation Method of Ground Unmanned Combat System Based on Capability. *Fire Control Command. Control* **2021**, *1633*, 182–187.
4. Zhong, Y.; Yao, P.; Zhang, J.; Wan, L. Formation and adjustment of manned/unmanned combat aerial vehicle cooperative engagement system. *J. Syst. Eng. Electron.* **2018**, *29*, 756–767.
5. Wang, Z.; Guo, Y.; Li, N.; Yuan, H.; Hu, S.; Lei, B.; Wei, J. Autonomous confrontation strategy learning evolution mechanism of unmanned system group under actual combat in the loop. *Comput. Commun.* **2023**, *209*, 283–301. [CrossRef]
6. Zhu, X.; Zhu, X.; Yan, R.; Peng, R. Optimal routing, aborting and hitting strategies of UAVs executing hitting the targets considering the defense range of targets. *Reliab. Eng. Syst. Saf.* **2021**, *215*, 107811. [CrossRef]
7. Madni, A.M.; Sievers, M.; Erwin, D. Formal and Probabilistic Modeling in Design of Resilient Systems and System-of-Systems. In Proceedings of the AIAA Scitech 2019 Forum, San Diego, CA, USA, 7–11 January 2019.
8. Fan, D.; Sun, B.; Dui, H.; Zhong, J.; Wang, Z.; Ren, Y.; Wang, Z. A modified connectivity link addition strategy to improve the resilience of multiplex networks against attacks. *Reliab. Eng. Syst. Saf.* **2022**, *221*, 108294. [CrossRef]
9. Chen, Z.; Zhou, Z.; Zhang, L.; Cui, C.; Zhong, J. Mission reliability modeling and evaluation for reconfigurable unmanned weapon system-of-systems based on effective operation loop. *J. Syst. Eng. Electron.* **2023**, *34*, 588–597. [CrossRef]
10. Sun, Y.; Zhang, T. Research on Autonomous Reconstruction Method for Dependent Combat Networks. *IEEE Syst. J.* **2023**, *17*, 1–10. [CrossRef]
11. Sun, Q.; Li, H.; Zhong, Y.; Ren, K.; Zhang, Y. Deep reinforcement learning-based resilience enhancement strategy of unmanned weapon system-of-systems under inevitable interferences. *Reliab. Eng. Syst. Saf.* **2023**, *242*, 109749. [CrossRef]
12. Sun, Q.; Li, H.; Wang, Y.; Zhang, Y. Multi-swarm-based cooperative reconfiguration model for resilient unmanned weapon system-of-systems. *Reliab. Eng. Syst. Saf.* **2022**, *222*, 108426. [CrossRef]
13. Raman, R.A.r.; D'Souza, M.A. Decision learning framework for architecture design decisions of complex systems and system-of-systems. *Syst. Eng.* **2019**, 538–560. [CrossRef]
14. Fang, Z. System-of-Systems Architecture Selection: A Survey of Issues, Methods, and Opportunities. *IEEE Syst. J.* **2022**, *16*, 4768–4779. [CrossRef]

15. Davendralingam, N.; Delaurentis, D.A. A Robust Portfolio Optimization Approach to System of System Architectures. *Syst. Eng.* **2015**, *18*, 269–283. [CrossRef]
16. Lin, M.; Chen, T.; Chen, H.; Ren, B.; Zhang, M. When architecture meets AI: A deep reinforcement learning approach for system of systems design. *Adv. Eng. Inform.* **2023**, *56*, 101965. [CrossRef]
17. Wang, Q.; Lai, K.H.; Tang, C. Solving combinatorial optimization problems over graphs with BERT-Based Deep Reinforcement Learning. *Inf. Sci.* **2023**, *619*, 930–946. [CrossRef]
18. Yu, J.J.Q.; Yu, W.; Gu, J. Online Vehicle Routing With Neural Combinatorial Optimization and Deep Reinforcement Learning. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3806–3817. [CrossRef]
19. Li, J.; Jiang, J.; Yang, K.; Chen, Y. Research on Functional Robustness of Heterogeneous Combat Networks. *IEEE Syst. J.* **2018**, *13*, 1487–1495. [CrossRef]
20. Li, J.; Zhao, D.; Ge, B.; Jiang, J.; Yang, K. Disintegration of Operational Capability of Heterogeneous Combat Networks Under Incomplete Information. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *50*, 5172–5179. [CrossRef]
21. Li, J.; Zhao, D.; Jiang, J.; Yang, K.; Chen, Y. Capability Oriented Equipment Contribution Analysis in Temporal Combat Networks. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 696–704. [CrossRef]
22. Zhang, J.; Lv, H.; Hou, J. A novel general model for RAP and RRAP optimization of k-out-of-n:G systems with mixed redundancy strategy. *Reliab. Eng. Syst. Saf.* **2023**, *229*, 108843. [CrossRef]
23. Levitin, G.; Xing, L.; Dai, Y. Optimizing partial component activation policy in multi-attempt missions. *Reliab. Eng. Syst. Saf.* **2023**, *235*, 109251. [CrossRef]
24. Peiravi, A.; Nourelfath, M.; Zanjani, M.K. Universal redundancy strategy for system reliability optimization. *Reliab. Eng. Syst. Saf.* **2022**, *225*, 108576. [CrossRef]
25. Ordoukhanian, E.; Madni, A. Model-Based Approach to Engineering Resilience in Multi-UAV Systems. *Systems* **2019**, *7*, 11. [CrossRef]
26. Zhong, Y.; Li, H.; Sun, Q.; Huang, Z.; Zhang, Y. A kill chain optimization method for improving the resilience of unmanned combat system-of-systems. *Chaos Solitons Fractals* **2024**, *181*, 114685. [CrossRef]
27. Papadimitriou, C.H.; Steiglitz, K. *Combinatorial Optimization: Algorithms and Complexity*; Dover Publications, Inc.: Mineola, NY, USA, 1998.
28. Dorigo, M.; Gambardella, L. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* **1997**, *1*, 53–66. [CrossRef]
29. Horowitz, E.; Sahni, S. *Computing Partitions with Applications to the Knapsack Problem*; Cornell University: Ithaca, NY, USA, 1972.
30. Yuan, E.; Wang, L.; Cheng, S.; Song, S.; Fan, W.; Li, Y. Solving flexible job shop scheduling problems via deep reinforcement learning. *Expert Syst. Appl.* **2024**, *245*, 123019. [CrossRef]
31. Marinescu, R.; Dechter, R. AND/OR Branch-and-Bound search for combinatorial optimization in graphical models. *Artif. Intell.* **2009**, *173*, 1457–1491. [CrossRef]
32. Rabiner, L. Combinatorial optimization:Algorithms and complexity. *IEEE Trans. Acoust. Speech Signal Process.* **2003**, *32*, 1258–1259. [CrossRef]
33. Li, K.; Zhang, T.; Wang, R.; Wang, Y.; Han, Y.; Wang, L. Deep Reinforcement Learning for Combinatorial Optimization: Covering Salesman Problems. *IEEE Trans. Cybern.* **2022**, *52*, 13142–13155. [CrossRef]
34. Hopfield, J.J.; Tank, D.W. Neural computation of decisions in optimization problems. *Biol. Cybern.* **1985**, *52*, 141–152. [CrossRef]
35. Vinyals, O.; Fortunato, M.; Jaitly, N. Pointer networks. In Proceedings of the International Conference on Neural Information Processing Systems, Montreal, QC, USA, 7–12 December 2015.
36. Bello, I.; Pham, H.; Le, Q.V.; Norouzi, M.; Bengio, S. Neural combinatorial optimization with reinforcement learning. *arXiv* **2016**, arXiv:1611.09940.
37. Dai, H.; Khalil, E.B.; Zhang, Y.; Dilkina, B.; Song, L. Learning Combinatorial Optimization Algorithms over Graphs. *Statistics* **2017**, *52*, 6348–6358.
38. Li, Z.; Chen, Q.; Koltun, V. Combinatorial optimization with graph convolutional networks and guided tree search. In Proceedings of the NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, New Orleans, LA, USA, 2–8 December 2018.
39. Chen, W.; Li, J.; Jiang, J. Heterogeneous Combat Network Link Prediction Based on Representation Learning. *IEEE Syst. J.* **2021**, *15*, 4069–4077. [CrossRef]
40. Sun, Y.; Han, J.; Yan, X.; Yu, P.S. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proc. Vldb Endow.* **2011**, *4*, 992–1003. [CrossRef]
41. Cares, J.R. *An Information Age Combat Model*; Technical Report; Produced for the United States Office of the Secretary of Defense: Arlington, VA, USA, 2004.
42. Pan, X.; Wang, H.; Yang, Y.; Zhang, G. Resilience based importance measure analysis for SoS. *J. Syst. Eng. Electron.* **2019**, *30*, 920–930.
43. Agnetis, A.; Mirchandani, P.B.; Pacifici, P.A. Scheduling Problems with Two Competing Agents. *Oper. Res.* **2004**, *52*, 229–242. [CrossRef]
44. Singh, R.; Gupta, A.; Shroff, N.B. Learning in Constrained Markov Decision Processes. *IEEE Trans. Control. Netw. Syst.* **2023**, *10*, 441–453. [CrossRef]

45. Zhan, W.; Luo, C.; Wang, J. Deep-Reinforcement-Learning-Based Offloading Scheduling for Vehicular Edge Computing. *IEEE Internet Things J.* **2020**, *7*, 5449–5465. [CrossRef]
46. Bahdanau, D.; Brakel, P.; Xu, K.; Goyal, A.; Lowe, R.; Pineau, J.; Courville, A.; Bengio, Y. An Actor-Critic Algorithm for Sequence Prediction. *arXiv* **2016**, arXiv:1607.07086.
47. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2, Montreal, QC, USA, 8–13 December 2014.
48. Williams, R.J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **1992**, *8*, 229–256. [CrossRef]
49. Hu, B.; Li, F.; Zhou, H. Robustness of Complex Networks under Attack and Repair. *Chin. Phys. Lett.* **2009**, *26*, 128901.
50. Bin, H.; Fang, L. Repair strategies of scale-free networks under multifold attack strategies. *Syst. Eng. Electron.* **2010**, *32*, 43–47.