IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 25, 2020, accepted July 1, 2020, date of publication July 6, 2020, date of current version July 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3007542

An Optimal Searching Algorithm for the Equipment System-of-Systems Architecture Space With Uncertain Capabilities

TAO WANG, XIN ZHOU, WEIPING WANG, YIFAN ZHU, AND TIAN JING
College of Systems Engineering, National University of Defense Technology, Changsha 410073, China

Corresponding author: Xin Zhou (zhouxin09@nudt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China (No. 61273198) and the National Social Science Fund of China (No. 14BTQ034).

ABSTRACT Systematic operations make modern wars more and more complicated, thus it is a challenge to design, develop, and select an excellent System-of-Systems. In this paper, we propose an equipment System-of-Systems architecture (ESoSA) formalization framework based on the super network and an architecture space searching algorithm. First, the formal definition of ESoSA and the problem of ESoSA space searching problem framework are introduced, where the searching problem is transferred into a multi-agent dynamic programming problem. Each agent searches for the optimal architecture in its corresponding architecture space, and different architecture spaces may overlap with each other. Second, the objective of the problem is to choose several optimal architectures with maximized sampling rewards and minimized cumulative searching costs. Third, a sequential search algorithm based on decision indicators is proposed to solve the problem. Finally, the proposed algorithm is evaluated by simulation experiments, where the experimental results show that the proposed algorithm is with high quality, and outperforms other benchmark algorithms.

INDEX TERMS Super-network based architecture, multi-agent dynamic programming, sequential search.

I. INTRODUCTION

The interconnection between weapons has become more and more diverse currently [1]. Particularly, the widespread use of unmanned systems has led to major changes in the combat methods of modern warfare [2], [3], and it is a challenging issue to study warfare. The equipment system-of-systems (ESoS) [4] is a manifestation of the SoS[1] in the field of warfare. The ESoS is such complicated, and how to study the ESoS is a problem that needs to be solved urgently. Fortunately, the architecture provides an effective way to study this problem. The architecture is the combination of component systems in the ESoS and the interaction among component systems and with the external environment [7], [8]. In systems engineering, the ESoS architecture runs through the entire process, including requirement demonstration, model design, prototype development, and field testing [9]. Therefore, the ESoS is studied through its architecture[2] in this paper. Further a reasonable formal ESoS architecture is defined, and an multi-agent dynamic algorithm is proposed to achieve the optimal configuration of the core elements of the ESoS.

We believe that the ESoSA is the combination of equipment systems connected by command and control network, where these equipment systems have certain functions that are capable to complete some tasks [10]. Like building architecture to guide building construction, the ESoSA is used to guide the development of a specific ESoS. Assume that each ESoS has the potential capability of the ESoSA, and the real capability of ESoS is unknown before the ESoS being developed. Given this, an formalized architecture model, a space exploration problem framework and a dynamic programming[3] algorithm are proposed. In this paper, the following challenges need to be solved: First, the potential capabilities of the architecture is uncertain. In previous studies, the SoS

The associate editor coordinating the review of this manuscript and approving it for publication was Zonghua Gu.

[1] A system-of-systems (SoS) is the integration of a limited number of component systems, which are independent and operable, and are interconnected for a certain period of time to achieve a higher goal [5], [6].

[2] In general, the architecture is a conceptual blueprint defining the structure, operation of an organization, and evolution rule. In this paper, we give a description of the system-of-systems architecture in the field of equipment.

[3] In the dynamic programming problem, the complex problem is decomposed into a set of several sub-problems, where one sub-problem is solved each round and the system state has no aftereffect.

capabilities were determined after the architecture was given. However, the uncertainty of the potential capabilities of the architecture exists and reflected in the two aspects: the task uncertainty and the diversity of component combinations, and some minor factors that is not considered in the design process. Second, decision makers may have a variety of actions to develop the same architecture and acquire the actual capability. Therefore, decision makers should evaluate the expected rewards of these policies in order to make the best choice. Third, the decision makers may need several optimal architectures. The previous studies often selected only one architecture, and there was no research on the selection of multiple architectures in our knowledge. Given this, a novel architecture model and the ESoSA space searching problem are constructed, and a dynamic searching algorithm of architecture space is proposed.

Specifically, the work of this paper mainly includes the following three aspects:

- First, a super-network [9] based ESoSA framework is proposed. The ESoSA is constructed based on the equipment systems of the ESoS, that is, tasks, equipment systems, and command and control (C2) structures. Given this, the ESoSA search problem framework is proposed, where it is further transferred into a dynamic programming problem [11].
- Second, a multi-agent searching algorithm is proposed to find several optimal ESoSAs. Specifically, the decision indicator based sequential searching (DISS) algorithm is proposed, where its computational complexity is polynomial time and may be optimal.
- Third, the performance of DISS was tested and evaluated by experiments, where experimental results show that DISS is with high quality and outperforms other benchmark algorithms.

II. RELATED WORK

In this section, related work on the ESoSA modelling framework and solvers is reviewed.

A. THE EQUIPMENT SYSTEM-OF-SYSTEMs ARCHITECTURE

The capability is to complete several operation missions to achieve some desired operational effects under the premise of certain preparations and environmental conditions [12]. On the basis of this, the U.S. Department of Defense regards that the capabilities include doctrine, organization, training, education, personnel, facilities, and policy [13], [14]. In order to achieve the desired capability, equipment elements and non-equipment elements need to be combined organically. Presently, many scholars have put forward their understanding on the ESoSA, but the essential connotation is still unclear. We believe that the ESoSA is to integrate equipment systems with specific functions into a whole capable of accomplishing specific missions under constraints of the organizational structure. The ESoS capability is the comprehensive ability of the equipment system in the course of completing missions and tasks. The ESoS capability is hard to measure comprehensively and accurately, so we give a formal problem description of the ESoSA in this paper.

In military applications, many SoSA frameworks are widely used. Common operational architecture modeling frameworks include: DoDAF citehandley2012incorporating, MoDAF [16], NAF [17]. In fact, these frameworks can be represented by the super-network model. The super network[4] refers to a network with huge scale, complicated connection and heterogeneous nodes, that means a super network is composed of several networks with different characteristics, such as multiple layers, multiple levels, multiple dimensions, multiple attributes, etc. Therefore, the super network is a general model that can be used to reflect the interaction and influence in the ESoSA. Presently, many researchers use super networks to study operational SoS problems. Yu *et al.* [19] proposed a super network includes perception nodes, C2 nodes and strike nodes, and the association between these nodes. Shi *et al.* [20] proposed a supernetwork composed of five heterogeneous nodes, including sensor node, information node, decision node, communication node, and engagement node. There are different types of interaction between nodes. Zhao *et al.* [21] established a weapon equipment system super network, and proposed a granular analysis to reduce the complexity of the weapon equipment system scheme generation based on the network-centric combat mode. The super network includes task sub-network, capability sub-network and system sub-network. Chen *et al.* [22] put forward a multi-layer command and control supernetwork model including sensing subnet, command subnet, and firepower subnet. Based on the capability generation mechanism, it needs to take capability requirements into consideration to construct the ESoS, where the ESoS should include equipment, functions, and command and control structure. Given this, a capability-oriented ESoSA is proposed in this paper.

B. ALGORITHMS FOR THE ESoSA SEARCHING PROBLEM

In this paper, each super-network is an architecture that has the capability to complete a specific mission, while multiple super-networks with their capabilities constitute the architecture space. The architecture space searching algorithm needs to be proposed to select several optimal architectures. The architecture space searching problem is modeled as an optimization problem, and the architecture space searching problem proposed in this paper can be converted into an optimal path search problem, that is similar to the traveling salesman problem. The traveling salesman problem is widely used in real problems, such as vehicle routes and cargo placement in warehouses [23]. The combinatorial optimization problem is to find the best combination of different discrete events that satisfy different constraints. The swarm intelligence algorithms are effective means to solve combinatorial

[4] A super network is a network of networks, that has multi-layer, multi-level, multi-dimensional, or multi-attribute feature [18].

optimization problems, such as genetic algorithm, particle swarm optimization algorithm, ant colony optimization algorithm, and firefly algorithm, bat algorithm, bacteria colony foraging algorithm, bee colony algorithm, gray wolf optimization, spider monkey optimization and other extended algorithms [24]. These swarm intelligence algorithms find the optimal value through a large number of simulation experiments, but it is difficult to guarantee to find the optimal value theoretically. However, the traveling salesman problem is little different from the problem proposed in this article. The objective of our problem is to find several optimal architectures, where it takes cost to explore the unknown architecture, so the exploring path is dynamically constructed; the reward of each architecture scheme is subject to a certain probability distribution before exploration, and the exact information is only known after exploring. Therefore, the swarm intelligent algorithms are difficult to apply to our problems directly.

Another literature that is closely related to our paper is the consider-then-choose policy [25], that is, the decision maker first determines the set of unknown schemes that will be explored, and chooses the most effective scheme after the uncertainty of these schemes been eliminated. In recent decades, researchers have put forward many workers on how decision makers explore unknown schemes to solve the uncertainty of the problem. Weitzman [26] studied the problem of sequential search in the case of uncertain utility values. In the paper, a searching rule and a stopping rule are proposed that use the reservation value as the threshold, where the reservation value depends on the trade-off between the actual utility value of the searched scheme, the search cost, and the evaluation of the utility value of the unsearched scheme. Similarly, Peter and Morgan [27] proposed rules based on fixed sample size and order, and sufficient conditions for the algorithm to reach its optimal state. Roberts and Lattin [28] studied the consideration set composition model of commodities and predicted the price of new commodities based on the calibration of the utility function of the commodities. Blanco *et al.* [29] developed a step-by-step learning model in which decision makers spent costs to learn information about unknown solutions and proposed the best rules for when to stop searching. Ke *et al.* [30] continued to search for information expanded to a variety of options, and showed that only in the case of a sufficiently high expected valuation should search or purchase goods. Similar to our problem framework, Chen *et al.* [31] proposed an optimal algorithm for the human-assisted robot search problem, where the sequential search method is used to help the robot choose an optimal solution from several solutions. Although the above work has different research backgrounds, they have similar research ideas and research methods, that is, a sequential search algorithm based on indicators is proposed to solve the optimal solution. But the disadvantage of these algorithms is that they can only provide one solution, not multiple solutions. Thus, further work need to be done to solve the problems proposed in the paper.

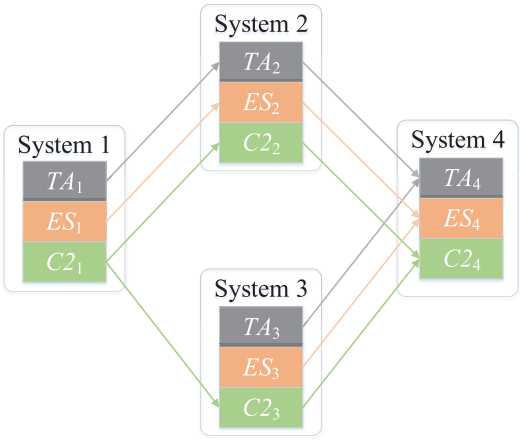


FIGURE 1. An example of the supernetwork-based ESoSA.

III. THE ESoSA SPACE SEARCHING PROBLEM

Here the definition of ESoSA is given, and the ESoSA space search problem is put forward. After that, the ESoSA space search problem is transformed into a multi-agent dynamic programming problem.

A. DEFINITIONS OF THE ESoSA

The ESoSA is formed by an organic combination of equipment components and non-equipment components, where the core elements are tasks, equipment systems and the C2 structure. The capability is used to measure the possibility that an ESoS accomplishes its mission. Specifically, an equipment system with certain functions can accomplish some tasks, and the ESoS based on these equipment systems with the C2 structure has the capability to complete a specific mission. In fact, the architecture is a model of the ESoS, and only takes core elements of the ESoS into considerations, so the ESoS developed based on the architecture has uncertainties. Here, developing an architecture means to build a ESoS based on the architecture. The definition of architectural capabilities is given as follows:

Definition 1 (Architecture Potential Capabiliy): The architecture potential capability refers to the capability of the ESoS to complete a specific mission, where the ESoS is developed according to the ESoSA, denoted as W.

The uncertainty of architecture potential capability can be represented by a probability distribution, that is, W follows a certain probability distribution. In the paper, the supernetwork is composed of three different types of networks: task network, equipment network and command network. Some definitions of the ESoSA is given as follows.

Definition 2 (Task Node): The task node is an activity process that can be performed by equipment, denoted as TA.

The mission can be represented as the task network, that is composed of several task nodes. We regard that when these tasks are completed, the mission is completed. The task network is denoted as $G^{TA} = (V^{TA}, E^{TA})$.

Definition 3 (Equipment Node): The equipment node refers to the equipment system that has specific functions and can independently complete specific tasks, denoted as ES.

Tasks are completed by equipment, so the relationships between equipment are affected by tasks, and the equipment network is denoted as $G^{ES} = (V^{ES}, E^{ES})$.

Definition 4 (Command and Control Node): The command and control (C2) node is used to process information, manage organization, decision planning, control feedback, and the C2 network is denoted as $G^{c2} = (V^{C2}, E^{C2})$.

Given this, the architecture topology model, shown as 1, is a heterogeneous network G^A composed of three types of nodes and five types of relationships, denoted as $G^A = \langle V^{TA}, V^{ES}, V^{C2}, E^{TA}, E^{ES}, E^{C2}, E^{TE}, E^{EC} \rangle$, where E^{TS} represents the relationship between task nodes, and E^{SC} represents the relationship between equipment nodes and $C2$ nodes. The supernetwork-based architecture modeling method is as follows: First, the mission of the ESoS is decomposed into a task network that can be performed by equipment. Second, the correspondence between tasks and equipment is given, and the system network is constructed according to the task network. Third, the correspondence between the equipment node and the $C2$ node is given, and the $C2$ node is established. Note that each supernetwork means an architecture, and the architecture space is composed of all supernetworks. Forth, task nodes, equipment nodes, and $C2$ nodes in the supernetwork should be contained in the real system, and these systems make up an ESoSA.

It is worth noting that it takes a cost $c \in C$ to develop an architecture, and at the same time get a certain capability (reward) to complete the mission $w \in W$. Then, the formalization of architecture model is composed of the architecture topology model, cost and capability, which is denoted as $\langle G^A, C, W \rangle$. If there are more nodes and more complicated relationships in the supernetwork, then the cost of architecture development will be greater, and the potential capacity of the architecture may be greater. However, the cost of developing an architecture and the assessment of its capabilities are a complex subject that is oriented to specific fields and situations. In this paper, we regard that the probability distribution and cost of the architecture is known in advance.

B. THE ESoSA SEARCHING PROBLEM

Suppose there are N decision-makers, and each decision-maker needs to select a different architecture to develop a ESoS. There may be intersections between different architecture spaces. For simplicity, decision-makers are represented as agents, and we take the n-th agent as an example, denoted as m. The ESoS capability here is equal to the reward, a comprehensive measure of the cost of developing architecture and the benefits after executing the architecture.

The reward w of each architecture follows the probability distribution $W(w)$, and the reward of different architectures are independent of each other. Let $k_m \in K_m, K_m = \{1, 2, \ldots, |K_m|\}$, where K_m is the number of architectures of Agent m. In addition, there may be intersections between different architecture spaces, $K_i \cup K_j \neq \emptyset$, while there may be no intersections, $K_i \cup K_j = \emptyset$. The reward of each architecture is uncertain before developing, but can be obtained through different actions. Agent m can develop the architecture k_m by taking action $a_k^m \in A_k^m, A_k^m = \{1, 2, \ldots, |A_k^m|\}$. Agent m explores the architecture in the undeveloped architecture space, and finally chooses a best architecture as the final architecture among all the developed architecture spaces. The goal of each agent is to choose an architecture with the maximized sampling reward and the minimized exploring cost.

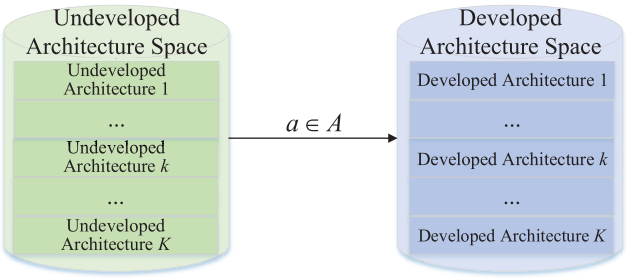


FIGURE 2. An example of undeveloped architecture space and developed architecture space.

Definition 5 (Architecture State): The architecture state refers to the form of the same architecture, including the undeveloped state and the developed state.

Fig. 2 describes an architectural state transition relationship, where the undeveloped architecture represents the architectural reward is unknown. After developing the architecture, the architectural reward is known and the architecture is transferred from undeveloped architecture space to the developed architecture space. Here, we define a formal description of the equipment SoSA space searching problem (ESoSASSP). Specifically, let us define two binary decision variables: $d_{a,k}^m$, when Agent m takes action a_k^m to develop architecture k_m, $d_{a,k}^n = 1$, otherwise $d_{a,k}^n = 0$; $s_{a,k}^m$, when Agent m finally chooses action a_k^m to develop the architecture k, $s_k^m = 1$, otherwise $s_k^m = 0$. Note that, the architecture k can only be chosen once, different agents should choose different architectures.

ESoSASSP

$$
\begin{aligned}
&\max\ E[\sum_{n\in M}\sum_{k\in K_m}\sum_{a\in A_k^m}(-d_{a,k}^m c_{a,k} + n_k s_{a,k}^m w_k)] \\
&\text{s.t. } d_{a,k}^m \le 1, m \in M, k \in K_m, a \in A_k^m \quad (a) \\
&\quad d_{a,k}^m \ge s_{a,k}^m, \quad m \in M, k \in K_m, a \in A_k^m \quad (b) \\
&\quad \sum_{k\in K_m}\sum_{a\in A_m} s_{a,k}^m = 1, m \in M \quad (c) \\
&\quad d_{a,k}^m \in \{0,1\}, \quad m \in M, k \in K_m,\ a \in A_k^m (d) \\
&\quad s_{a,k}^m \in \{0,1\}, \quad m \in M, k \in K_m,\ a \in A_k^m (e) \\
&\quad c_{a,k} \in \mathbb{R}^+, \quad k \in K_m,\ a \in A_k^m \quad (f) \qquad (1)
\end{aligned}
$$

The indicator is the sum of the maximum reward in developed architectures space and the cumulative developing cost,

where the objective function is to maximize the indicator, shown in Eq.(1). Specifically, constraint (a) ensures that for any agent's architecture space, the architecture is either developed or not. Constraint (b) means that if an agent finally chooses a solution, then the architecture needs have been developed. Constraint (c) means that each agent chooses only one architecture from the developed architecture space in the end. Constraint (d) and (e) represent the value space of two types of decision variables. Constraint (f) represents the value of the cost when taking actions.

C. THE ESoSA DYNAMIC PROGRAMMING PROBLEM

According to the description of ESoSASSP in the previous section, the problem can be transferred as a formal multi-agent dynamic programming problem. The total architecture space is denoted as $K = \bigcup_{m\in M} K_m$, where k_m is the architecture space of Agent m. Here, we take Agent m as an example. Let K_m be the space of developed architectures, and let $\bar{K}_m$ be the space of undeveloped architectures. At the beginning, the agent owns the undeveloped space, and the exact rewards of architectures are unknown. Suppose that the architecture space and the reward probabilities distributions of architectures are known in advance. For each decision, Agent m has the choice to explore an undeveloped architecture from the set $\bar{K}_m$, or stop to select a developed architecture from the set K_m. If Agent m continues to explore the undeveloped architecture space, then it can take the action $a_k^m \in A_k^m$, while if Agent m stops searching, it then chooses the architecture with the highest sampling reward in the set K_m.

Let $(\bar{K}, y)$ be the state of the problem, where $y = \max_{k\in K} w_k$ is the maximized sampling reward in the set K. The global state assessment function $\Psi(\bar{K}, y)$ is the expected reward obtained by performing the developing route under the condition that the maximized sampling reward is y and the undeveloped architecture space is $\bar{K}$, where the developing route is the exploration sequence and length of the unknown architecture. In this paper, each architecture reward is independent of each other, then the global state assessment function $\Psi(\bar{K}, y)$ can be divided into the sum of all local state assessment functions $\Psi(\bar{K}_m, y_m), m \in M$, where $y_m = \max_{k\in K_m} w_k$ is the maximized sampling reward in the set K_m. Thus, the objective of the problem is to find developing route that maximize the global state assessment function $\Psi(\bar{K}, y)$.

$$\Psi(\bar{K}, y) = \sum_{m\in M} \Psi(\bar{K}_m, y_m) \tag{2}$$

For the state $\Psi(\bar{K}_m, y_m)$, Agent m has several actions $a = \{a_1^m, a_2^m, \ldots, a_{|A|}^m\}$ to explore the unknown architectures, then it chooses the action based on the Eq.(3).

$$\begin{aligned}\Psi(\bar{K}_m, y_m) = \max\{&y_m, \Psi_1(\bar{K}_m, y_m), \ldots, \Psi_{|A|}(\bar{K}_m, y_m)\}\\ &where\\ \Psi_k(\bar{K}_m, y_m) = \max_{k\in\bar{K}_m}\{&-c_{a,k}^m\\ &+ \Psi(\bar{K}_m - \{k\}, y_m)\int_{-\infty}^{y_m} dW_k(w_k)\\ &+ \int_{y_m}^{\infty} \Psi(\bar{K}_m - \{k\}, y_k) dW_k(w_k)\}\end{aligned} \tag{3}$$

The variable $\Psi_k(\bar{K}_m, y_m)$ represent the expected state assessment function of Agent m after executing the action a_k^m at the state $(\bar{K}, y)$. Further, the Agent needs to compare the assessment function after taking different actions, and select the action with the largest expected reward. Taking the action a_k^m as an example, if the sampling reward of architecture k is less than the recorded maximized reward of Agent m, i.e. $w_k < y_m$, the current highest reward will not change and the assessment reward is $-c_{a,k}^m + \Psi_k(\bar{K}_m - \{k\}, y_m)$; if $w_k \le y_m$, then the current maximized sampling reward changes to w_k and the assessment reward is $-c_{a,k}^m + \Psi_k(\bar{K}_m - \{k\}, w_k)$.

The process of finding the optimal architecture can be regarded as an optimal route selection problem, that is, selecting the architecture with the largest sampling reward in the optimal route. However, the computational complexity is $O(n^2 2^n)$ [32], making it hard to be solved in a limited time.

IV. THE DYNAMIC PROGRAMMING ALGORITHM

We put forward a decision indicator based Sequential Searching (DISS) algorithm to solve the problem proposed in the previous section. We first define indicators $z_{a,k}$ for executing the action a_k^m for each architecture k, then give the theoretical analysis about the DISS algorithm.

A. THE DISS ALGORITHM

According to Eq.(3), the indicators $z_{a,k}$ for executing the action a_k^m for each architecture k are defined as follows:

$$z_{a,k} = -c_{a,k} + z_{a,k}\int_{-\infty}^{z_{a,k}} dW_k(w_k) + \int_{z_{a,k}}^{\infty} w_k dW_k(w_k) \tag{4}$$

The following equation can deduced based on Eq.(4):

$$c_{a,k} = \int_{z_{a,k}}^{\infty} (w_k - z_{a,k}) dW_k(w_k) \tag{5}$$

With reference to the Pandora's rule [26], a searching rule is designed under the conditions of $\Psi(\bar{K}_m, y_m)$ and $\{z_{a,k} | k \in K_m, a \in A_k^m\}$. The DISS algorithm is shown in Algorithm 1.

Without loss of generality, we take Agent m as an example. The single-agent searching algorithm can be divided into three stages: indicator ranking (lines 5-9), indicator judgment (lines 11-12), and architecture selection (lines 13-25). The indicator ranking refers to calculating all the decision indicators according to Eq.(5); the indicator judgment refers to that if Agent m wants to explore another unknown architectures, then it chooses an architecture and action with the largest indicator; the architecture selection means that when the collected maximum sampling reward is greater than the maximum indicator of all unknown architectures, Agent m stops exploring and chooses the architecture with the highest sampling reward.

Algorithm 1: The DISS Algorithm

procedure **DISS**($\bar{D}$)
begin
  $D, \pi \leftarrow \emptyset$;
  $c, y \leftarrow 0$;
  for $m \in M, k_m \in \bar{D}_m, a_k^m \in A_k^m$, **do**
    $z_{a,k} \leftarrow Solving(c_{a,k} = \int_{z_{a,k}}^{\infty}(w_k - z_{a,k})dW_k(w_k))$;
  for $m \in M$ **do**
    $\pi_m \leftarrow Sorting(z_{a,k}|k \in K_m, a \in A_k^m)$;
    $\pi \leftarrow \pi \cup \pi_m$
  for $m \in M$ **do**
    $k_m \leftarrow ParsingArchitecture(\pi_m(0))$;
    $a_k^m \leftarrow ParsingAction(\pi_m(0))$;
    if $y_m \leq z_{a,k}^m$ **then**
      $s_{a^*,k^*}^m \leftarrow 1$;
      continue;
    $s \sim W_k(w_k)$;
    $D_m \leftarrow D_m \cup \{k_m\}$;
    $\bar{D}_m \leftarrow \bar{D}_m \setminus \{k_m\}$;
    $\pi \leftarrow \pi \setminus \{z_{a,k}|a \in A_k^m\}$;
    $d_{a,k}^m \leftarrow 1$;
    $c \leftarrow c + c_{a,k}^m$;
    if $y_m < s$ **then**
      $a^* \leftarrow a_k^m$;
      $k^* \leftarrow k_m$;
      $y_m \leftarrow s$;
  return $(-c + \sum_{m \in M} y_m)$;

B. PERFORMANCE ANALYSIS

The cost and effectiveness of the DISS Algorithm are analysed in this section.

Theorem 1: The time complexity of DISS is a polynomial.

Proof: From the Algorithm 1, it shows that the time complexity of the DISS algorithm lies on the time complexity of the sorting rule, since each agent executes corresponding actions in order according to the architectural indicators, while this order remains the same for the entire exploration process. Thus, the time complexity of our proposed algorithm is equal to that of the sorting algorithm.

From this theory, we know that the computational complexity of DISS is due to the sorting algorithm. Specifically, the average time complexity of heap sorting, bubble sorting, Shell sorting is $Olog_2(n)$, $O(n^2)$, $O(n^{1.3})$ separately, and the space complexity of these sorting algorithms is $O(1)$. Therefore, our proposed algorithm is with reasonable cost.

Theorem 2: Each architecture selected by DISS is conditionally optimal.

Proof: Each agent acquires the architecture based on the sequential allocation method. In fact, the selection of each architecture in the ESoSASSP can be mapped to the classic Pandora problem. The Pandora problem is a type of search problem that studies economics. In the Pandora problem, the reward of each project is subject to a probability distribution. The actual reward of the project is unknown before running the project. In our problem, an architecture corresponds to a project, where it has a reward w_k following a distribution. Once the reward of the architecture k is sampled, the variables $z_{a,k}$ with different actions related to the architecture k are transferred into the set K. Since each architecture is generated one by one, DISS computes each architecture with regard to the exploration results of previous architectures. In paper [26], it proves that this exploration algorithm is able to solve the Pandora problem and acquire the optimal results.

In general, it is hard to prove that the DISS algorithm is optimal. However each architecture calculated by the DISS algorithm is conditionally optimal, the total architectures will get a good result. Thus the DISS algorithm is still a high quality solution.

V. EMPIRICAL EVALUATION

In the previous section, we theoretically analyze the DISS algorithm. Here, we analyze its performance based on simulation experiments.

A. EXPERIMENT SETUP

In order to complete a certain mission, åŞĹåĹ such as border patrol, continuous reconnaissance, and electromagnetic interference, it is necessary to send a drone swarm to the target area to perform the mission. For such a new mission, how to build a drone swarm is the first issues. Here we give an alternative method: first, the mission is decomposed into a task network; second, as a drone with certain functions can complete a specific task, it builds the relationships between drones and tasks; third, constructing an C2 network based on the drones; finally, a multi-agent system model is established, and each agent has a task list, specific functions, and C2 network. The multi-agent system is a kind of equipment architectures. In order to achieve the highest operational efficiency, we need to choose the best architecture.

Here we define some metrics to evaluate the performance:

- The reward is the difference between the maximized sampling reward and the cumulative cost of developed architectures.
- The number of development (NoD for short) is the number of developed architectures in a simulation.
- The runtime of a simulation records the running time of the program.

To compare the performance of the DISS algorithm, some benchmark algorithms for the ESoSASSP are given, including two general algorithms (exploring all architectures and randomly exploring architectures) and a specific algorithm (exploring architectures based on expectations):

- Random algorithm (RA for short), that is, each agent chooses an action at each moment randomly. Specifically, an architecture of Agent m, $k_m \in \bar{K}_m$ is randomly

selected. If $k_m \in K_m$, that is, the architecture has been developed, then the search is finished and the reward of this architecture is obtained; if $k_m \in \bar{K}_m$, that is, the architecture k_m has not been developed and its reward is unknown, then Agent m randomly chooses an action from its action space $a_k^m \in A_k^m$ until the search ends.

- Traversal Development Algorithm (TDA), that is, each agent will develop all the architectures. For an undeveloped architecture $k \in \bar{K}$, the agent chooses the action with the lowest cost, $a_k^* = \arg\min_{c_k}\{c_{a,k}|a_k^m \in A_k^m\}$. When agents have developed of all undeveloped architectures, the architecture with the highest sampling reward is chosen as the final solution.
- Expected Value Development Algorithm, (EVDA), is similar to the algorithm of proposed in this paper. The evaluation indicator of EVDA is the difference between the highest expected value and the cost, namely $\{\tilde{z}_{a,k}^m = \{E(w_k) - c_{a,k}^m|k \in K, a \in A_k^m\}$. When the highest sampling reward exceeds the indicator, the search is completed.

We design four cases to evaluate the scalability of the architecture space by setting parameters based on experience. The number of undeveloped architectures is $|\bar{K}| = \{20, 100, 1000, 10000\}$. There are three actions to develop each architecture, where the costs are subject to three uniform distributions $c_1 \sim U(1, 3), c_2 \sim U(0.5, 4), c_3 \sim U(1.5, 2.5)$. The reward of each architecture is subject to a probability distribution, $w_k \sim U(u_1, u_2)$ where $u_1 \sim U(50, 60)$, $u_2 \sim U(90, 100)$. Let the number of agents, i.e the number of selected architecture schemes (*NoS* for short), be 1 to 10.

- Case A: Exploring in the space with $|\bar{K}| = 20$ architectures.
- Case B: Exploring in the space with $|\bar{K}| = 100$ architectures.
- Case C: Exploring in the space with $|\bar{K}| = 1000$ architectures.
- Case D: Exploring in the space with $|\bar{K}| = 10000$ architectures.

B. EXPERIMENT RESULTS

(1) Case A

The average rewards of RA, TDA, EVDA, DISS in Case A are shown in Fig. 3. It reveals that as the number of selections NoS grows, these average rewards grows gradually, and the average reward of DISS is a little higher than that of other algorithms.

The average NoDs of RA, TDA, EVDA, DISS in Case A are shown in Fig. 4(a). It reveals that as the number of selections NoS increases, the number of development NoD of EVDA, RA, DISS increases gradually, and the NoD of TDA remains at 20. Fig. 4(b) reveals average runtimes in Case A. It shows that these average runtime are similar.

(2) Case B

The average rewards of RA, TDA, EVDA, DISS in Case A are shown in Figure 5. The value of TDA at $NoS = 1$ is less than 0, it is not shown in this figure. As the variable *NoS* grows, average rewards of these algorithms increase. In summary, DISS performs a little betther than that of other algorithms.

The average *NoD*s of the four algorithms in Case B are shown in Fig. 6(a). Since TDA develops all architectures, the *NoD* is equal to the number of undeveloped architectures. The average *NoD*s of DISS are a little larger than that of EVDA and less than that of RA in these simulations. The average runtimes of the four algorithms in Case B are shown in Fig. 6(b). It demonstrates that the average runtime of TDA is highest in this Case.

(3) Case C

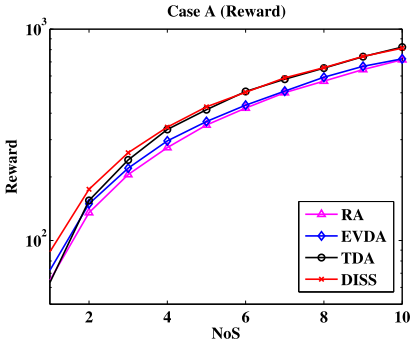
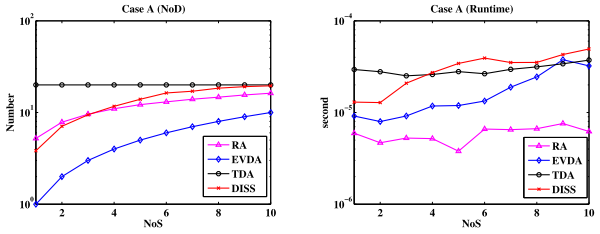


FIGURE 3. Average rewards in Case A.



(a) (b)

FIGURE 4. NoDs and runtimes of RA, TDA, EVDA, DISS in Case A, where Fig. 4(a) shows the average NoDs in Case A, and Fig. 4(b) shows the average runtimes in Case A.

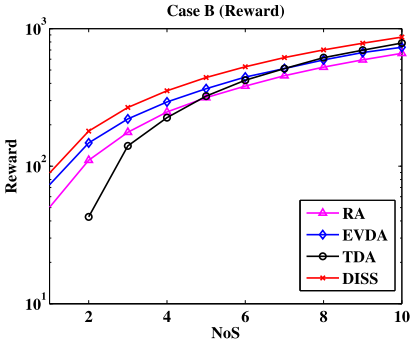
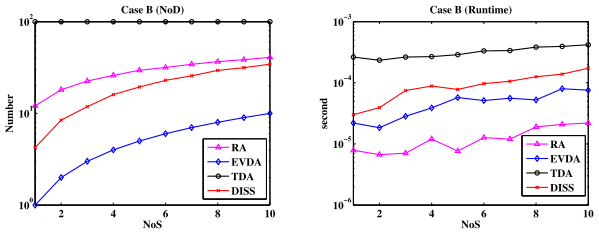


FIGURE 5. Average rewards in Case B.

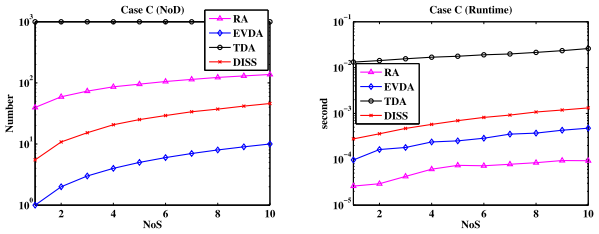


(a) (b)

FIGURE 6. ***NoD*s and *runtime*s of RA, TDA, EVDA, DISS in Case B, where Fig. 6(a) shows the average *NoD*s in Case B, and Fig. 6(b) shows the average runtimes in Case B.**

TABLE 1. **Average rewards in Case C.**

NoS	1	2	3	4	5
TDA	-1397.4	-1276.3	-1206.2	-1108.1	-1049.4
EVDA	75.2	149.2	222.5	296.9	369.5
RA	-6.8	26.9	72.7	120.3	178.4
DISS	92.0	183.8	273.5	365.7	456.4
NoS	6	7	8	9	10
TDA	-911.57	-804.1	-706.0	-612.2	-484.4
EVDA	445.9	519.7	590.2	669.4	742.2
RA	230.8	288.7	344.9	405.7	465.1
DISS	546.1	637.1	725.0	813.9	908.1



(a) (b)

FIGURE 7. ***NoD*s and *runtime*s of RA, TDA, EVDA, DISS in Case C, where Fig. 7(a) shows the average *NoD*s in Case C, and Fig. 7(b) shows the average runtimes in Case C.**

The average rewards of RA, TDA, EVDA, DISS in Case C are tabulated in Tab. 1. The average reward of DISS is much higher than that of other algorithms in these cases.

Fig. 7(a) shows the average *NoD*s of RA, TDA, EVDA, DISS in Case C. It reveals that the average *NoD* of DISS is less than that of RA and TDA, and the average *NoD* of TDA is always 1000. Fig. 7(b) shows the average runtimes of RA, TDA, EVDA, DISS in Case C. The runtime of TDA is highest in these simulations, and the runtime of DISS is between RA and EVDA.

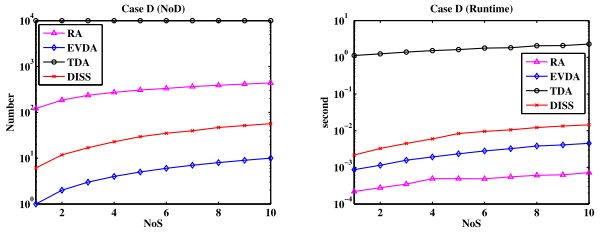
(4) Case D

The average rewards of RA, TDA, EVDA, DISS in Case D are tabulated in Tab. 2. Similar to the results in Case C, DISS performs much better than RA, TDA, EVDA in all simulations.

Fig. 8(a) depicts the average *NoD*s of RA, TDA, EVDA, DISS in Case D. It shows the average *NoD*s of DISS ranges from 6 to 60 and the average *NoD*s of EVDA developments is from 1 to 10 in these simulations. Fig. 8(b) depicts the

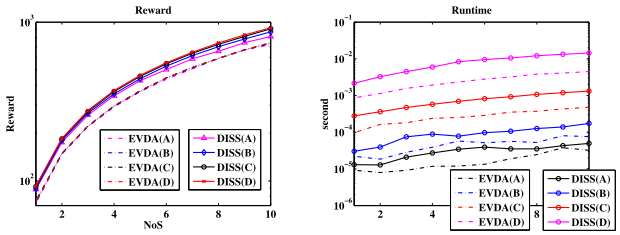
TABLE 2. **Average rewards in Case D.**

NoS	1	2	3	4	5
TDA	-14855.9	-14737.0	-14509.5	-14542.6	-14402.3
EVDA	73.9	149.5	221.7	297.8	370.7
RA	-178.3	-233.9	-264.1	-267.5	-265.1
DISS	92.9	185.5	278.2	371.4	463.0
NoS	6	7	8	9	10
TDA	-14277.9	-14262.6	-14123.1	-14014.5	-13840.0
EVDA	518.9	519.7	592.8	669.1	743.8
RA	-242.4	-232.7	-211.9	-190.0	-163.8
DISS	555.2	646.4	740.1	830.6	922.4



(a) (b)

FIGURE 8. ***NoD*s and *Runtime*s of RA, TDA, EVDA, DISS in Case A, where Fig. 8(a) shows the *NoD*s in Case D, and Fig. 8(b) shows the *Runtime*s in Case D.**



(a) (b)

FIGURE 9. **Average rewards and runtimes of DISS and EVDA in four cases.**

average runtimes of RA, TDA, EVDA, DISS in Case D. The runtime of TDA is highest, while the average runtime of DISS is between RA and EVDA.

(5) Comparison of DISS and EVDA

Finally, the average rewards and average runtimes of DISS and EVDA in Case A, Case B, Case C, and Case D, are comprehensively compared. The average rewards are shown in Fig. 9(a). It demonstrates that the quality of DISS is better than that of EVDA in these simulations, especially the average reward of DISS increases as the architecture space increases. The average runtime of DISS and EVDA in these simulations are depicted in Fig. 9(b). It obviously show that as the number of architecture space increases, the average runtime of DISS increase polynomially, and the runtime of EVDA is slightly lower than that of DISS in these simulations.

C. EXPERIMENT ANALYSIS

Based on the above simulation results, some interesting phenomena could be found for the DISS algorithm. First,

since the reward of each architecture is greater than zero, the average reward of DISS will increase with the increase of NoS. Second, the DISS algorithm is a kind of sequential allocation methods, which means that each agent's architecture is determined in turn. As the variable NoS increases, the variable NoD will also increase. Third, the increasing speed of runtime may decrease with the increase of NoS. On one hand, a new architecture is selected each round, and the total runtime will increase gradually; on the other hand, the developed architectures without been selected could be used as the condition for subsequent sampling, hence the runtime may decrease. Forth, the experiment results show that the performance of DISS is better than RA, TDA, and EVDA. The reason is that DISS selects the architecture greedily through the index-based method, which can find the optimal development path and select best architecture. In the DISS algorithm, it develops a suitable number of architectures, and stops to select the architectures with the highest sampling reward. The indicator is calculated based on the specific problem characteristics. Intuitively, the indicator is the trade-off of the maximum sample value, the searching cost, and state evaluation value, which seams to the expected reward of the problem.

VI. CONCLUSION

In this paper, we propose a supernetwork-based ESoSA model with some prior knowledge and a multi-agent architecture space searching algorithm. Specifically, the objective of the problem is to choose several optimal architectures in each agent's own space. The architecture space search process is modeled as a multi-agent dynamic programming problem. In the paper, we assume that the probability distribution of the potential capabilities of each architecture is known before exploration, but the specific capability of architecture is unknown until an agent develops it. In order to solve such problem, a sequential search algorithm is proposed based on decision indicators. Finally, the simulation experiments show that the time complexity of the proposed algorithm is polynomial and the reward is much higher than other benchmark algorithms

The research work in this paper has some academic value and practical value. First, we expand the problem framework from the single-agent dynamic programming to the multi-agent dynamic programming, and propose an algorithm analyzed from both aspects of effectiveness and cost theoretically. Specially, the algorithm may be optimal through theoretical analysis, which is able to compute the best exploration path and architecture scheme. Second, our work has a wide range of applications. Instead of only recommending one architecture in the previous research, our proposed algorithm could recommend any number of architectures within limited time and resources.

The proposed algorithm can be applied in the field of exhibiting the property of sequential search. Since we give many of assumptions of the formulation, such as the independent reward distribution, sequential searching, our algorithm has many limitations. It is very hard to model well and puts forward an optimal algorithm to solve the problems, that is, dependent reward distribution, parallel search, and pay-as-you-go research. Future work will focus on extending our algorithm to settings with these problems.

ACKNOWLEDGMENT

(Tao Wang and Xin Zhou are co-first authors.)

REFERENCES

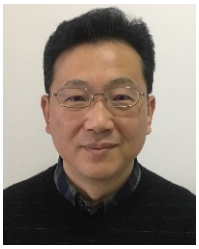
[1] J. S. Thurnher, "Feasible precautions in attack and autonomous weapons," in *Dehumanization of Warfare*. Cham, Switzerland: Springer, 2018, pp. 99–117.
[2] X. Zhou, W. Wang, T. Wang, M. Li, and F. Zhong, "Online planning for multiagent situational information gathering in the Markov environment," *IEEE Syst. J.*, vol. 14, no. 2, pp. 1798–1809, Jun. 2020.
[3] X. Zhou, W. Wang, T. Wang, Y. Lei, and F. Zhong, "Bayesian reinforcement learning for multi-robot decentralized patrolling in uncertain environments," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 11691–11703, Dec. 2019.
[4] H. He, W. Wang, Y. Zhu, X. Li, and T. Wang, "An operation planning generation and optimization method for the new intelligent combat SoS," *IEEE Access*, vol. 7, pp. 156834–156847, 2019.
[5] C. Kerr, R. Jaradat, and N. U. Ibne Hossain, "Battlefield mapping by an unmanned aerial vehicle swarm: Applied systems engineering processes and architectural considerations from system of systems," *IEEE Access*, vol. 8, pp. 20892–20903, 2020.
[6] M. Jamshidi, "System of systems–Innovations for 21st century," in *Proc. IEEE Region 10 3rd Int. Conf. Ind. Inf. Syst.*, Dec. 2008, pp. 6–7.
[7] M. Li, M. Li, K. Yang, B. Xia, and C. Wan, "A network-based portfolio optimization approach for military system of systems architecting," *IEEE Access*, vol. 6, pp. 53452–53472, 2018.
[8] B. Zhao, X. Wang, D. Lin, M. M. Calvin, J. C. Morgan, R. Qin, and C. Wang, "Energy management of multiple microgrids based on a system of systems architecture," *IEEE Trans. Power Syst.*, vol. 33, no. 6, pp. 6410–6421, Nov. 2018.
[9] N. Jia, Y. You, Y. Lu, Y. Guo, and K. Yang, "Research on the search and rescue system-of-systems capability evaluation index system construction method based on weighted supernetwork," *IEEE Access*, vol. 7, pp. 97401–97425, 2019.
[10] T. Wang, X. Zhou, W. Wang, Y. Zhu, and T. Jing, "An optimal approach for combat system-of-systems architecture search under uncertainty," *IEEE Access*, vol. 7, pp. 119140–119150, 2019.
[11] X. Zhou, W. Wang, Y. Zhu, T. Wang, and B. Zhang, "Centralized patrolling with weakly-coupled agents using Monte Carlo tree search," *IEEE Access*, vol. 7, pp. 157293–157302, 2019.
[12] T. Patten, R. Fitch, and S. Sukkarieh, "Large-scale near-optimal decentralised information gathering with multiple mobile robots," in *Proc. ACRA*, 2013, pp. 1–15.
[13] D. P. Bertsekas, *Abstract Dynamic Programming*. Cambridge, MA, USA: Massachusetts Institute of Technology, Athena Scientific, 2018.
[14] M. Schwager *et al.*, "A multi-robot control policy for information gathering in the presence of unknown hazards," in *Robotics Research*. Cham, Switzerland: Springer, 2017, pp. 455–472.
[15] H. A. H. Handley,"Incorporating the NATO human view in the DoDAF 2.0 meta model," *Syst. Eng.*, vol. 15, no. 1, pp. 108–117, Mar. 2012.
[16] M. Hause, "The unified profile for DoDAF/MODAF (UPDM) enabling systems of systems on many levels," in *Proc. IEEE Int. Syst. Conf.*, Apr. 2010, pp. 426–431.
[17] J. Stirna, "A comparative analysis of concepts for capability design used in capability driven development and the nato architecture framework," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.*, 2017, pp. 27–38.
[18] A. Nagurney and J. Dong, *Supernetworks: Decision-Making for the Information Age*. Broadheath, U.K.: Elgar, Edward Publishing, Incorporated, 2002.
[19] X. Gao, H. Yu, Y. Wang, and B. Chen, "A modeling method for command and control supemetworks based on hyperedge generation strategies," in *Proc. IEEE 22nd Int. Conf. Comput. Supported Cooperat. Work Design ((CSCWD))*, May 2018, pp. 128–132.

[20] S. Fu-li, L. Yong-lin, and Z. Yi-fan, "A military communication supernetwork structure model for netcentric environment," in *Proc. Int. Conf. Comput. Inf. Sci.*, Dec. 2010, pp. 33–36.
[21] Q. Zhao, S. Li, Y. Dou, X. Wang, and K. Yang, "An approach for weapon System-of-Systems scheme generation based on a supernetwork granular analysis," *IEEE Syst. J.*, vol. 11, no. 4, pp. 1971–1982, Dec. 2017.
[22] B. Chen, H. Yu, Y. Wang, X. Gao, and Y. Xu, "Multilevel command and control supernetwork modeling based on attribute synergy prioritization," *IEEE Access*, vol. 7, pp. 32693–32702, 2019.
[23] R. Matai, S. P. Singh, and M. L. Mittal, "Traveling salesman problem: An overview of applications, formulations, and solution approaches," in *Traveling Salesman Problem, Theory and Applications*, vol. 1. 2010.
[24] M. A. H. Akhand, S. I. Ayon, S. A. Shahriyar, N. Siddique, and H. Adeli, "Discrete spider monkey optimization for travelling salesman problem," *Appl. Soft Comput.*, vol. 86, Jan. 2020, Art. no. 105887.
[25] R. Wang and O. Sahin, "The impact of consumer search cost on assortment planning and pricing," *Manage. Sci.*, vol. 64, no. 8, pp. 3649–3666, Sep. 2018.
[26] M. L. Weitzman, "Optimal search for the best alternative," *Econometrica*, vol. 47, no. 3, pp. 641–654, 1979.
[27] M. Peter and M. Richard, "Optimal search," *Econometrica*, vol. 53, no. 4, pp. 923–944, 1985.
[28] J. H. Roberts and J. M. Lattin, "Development and testing of a model of consideration set composition," *J. Marketing Res.*, vol. 28, no. 4, pp. 429–440, Nov. 1991.
[29] B. Fernando, S. Monic, and J. M. Villasboas, "Optimal search for product information," *Manage. Sci.*, vol. 58, no. 11, pp. 2037–2056, 2013.
[30] T. T. Ke, Z.-J.-M. Shen, and J. M. Villas-Boas, "Search for information on multiple products," *Manage. Sci.*, vol. 62, no. 12, pp. 3576–3603, Dec. 2016.
[31] S. Chen *et al.*, "A polynomial time optimal algorithm for robot-human search under uncertainty," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 819–825.
[32] R. Bellman, "Dynamic programming treatment of the travelling salesman problem," *J. ACM*, vol. 9, no. 1, pp. 61–63, Jan. 1962.

TAO WANG received the Ph.D. degree in software engineering from the National University of Defense Technology (NUDT), Changsha, China, in 2007. He is currently an Associate Professor with NUDT. His research interests focus on systems engineering and simulation, multi-agent decision making under uncertainty, and data mining.

XIN ZHOU received the Ph.D. degree in management science and engineering from the National University of Defense Technology (NUDT), Changsha, China, in 2019. He is currently a Lecturer with the College of Systems Engineering, NUDT. His research interests focus on systems engineering and simulation, multi-agent decision making under uncertainty and reinforcement learning.

WEIPING WANG received the Ph.D. degree in systems engineering from the National University of Defense Technology (NUDT), Changsha, China, in 1998. He is currently a Professor with NUDT. His research interest focuses on systems engineering and simulation.



YIFAN ZHU received the Ph.D. degree in systems engineering from the National University of Defense Technology (NUDT), Changsha, China. He is currently a Professor with NUDT. His research interest focuses on systems engineering and simulation.

TIAN JING received the M.S. degree in management science and engineering from the College of Systems Engineering, National University of Defense Technology (NUDT), Changsha, China, in 2018. He is currently an Assistant with NUDT. His current research interests include swarm control and system of systems engineering.