IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 1, 2019, accepted August 18, 2019, date of publication August 26, 2019, date of current version September 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2937321

An Optimal Approach for Combat System-of-Systems Architecture Search Under Uncertainty

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 under Grant 61273198.

ABSTRACT The systematic and network-centric warfare becomes complicated currently. It is a challenge problem to design and choose the optimal combat system-of-systems to complete missions. In order to solve this challenge, we propose a super-network model based combat system-of-systems architecture (CSoSA) and a scheme space exploration algorithm in this paper. First, a formal definition of CSoSA is given based on the operational capability generation elements. The framework for CSoSA scheme space exploration problem is then presented. The goal of the problem is to select the optimal architecture with maximized expected rewards and minimized cumulative costs. In addition, a novel and optimal algorithm for CSoSA dynamic exploration is proposed based on the greedy search algorithm. Finally, we evaluated our algorithm through simulation experiments, where the results showed that our algorithm is significantly better than several benchmark algorithms. In general, the proposed architecture framework and the exploration algorithm can assist commanders in making further decisions.

INDEX TERMS Combat system-of-systems, architecture modeling, super-network model, greedy search, dynamic planning.

I. INTRODUCTION

With the development of informatization and intelligence of weapon systems, the interconnection between weapons becomes frequent [1]. Especially, the widespread usage of unmanned systems makes the combat mode of modern warfare undergone significant changes. It is a challenging problem to study the warfare from the perspective of joint operations. Fortunately, the system-of-systems engineering gives an idea to solve the challenge. A system-of-systems is an ''Integration of a finite number of constituent systems which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal'' [2]. The combat system-of-systems is an application of the system-of-systems in the field of warfare. The architecture reflects the configuration of components in the system-of-systems and the interaction between components and the external environment [3], [4]. The architecture captures the physical entity, information structure and the system-of-systems capability, and is the core framework of the system-of-systems. The architecture runs through the process of design, requirements demonstration, prototyping, and application testing [5]. Therefore, we indirectly study the combat system-of-systems through architectures, that achieves best configuration of combat system-of-systems core elements by giving appropriate formal definitions and optimizing the combat system-of-systems architecture (CSoSA).

The CSoSA is a collection of equipment interconnected by a command and control network that provides multiple functions to support the completion of certain tasks [6]. The CSoSA is used to guide the establishment of a specific combat system-of-systems. In this paper, challenges are threefold. First, the uncertainty of the architecture capability needs to be taken into consideration. In the past, the operational capability is determined after the architecture is established [7]. In fact, there are still many uncertain factors in the specific combat system-of-systems established according to the CSoSA. These uncertainties may not be taken into consideration when designing the architecture. Second, it takes a price when developing the combat system-of-systems based on the CSoSA, so the commander needs to decide whether to continue searching for the undeveloped CSoSA, or stop searching and select the developed CSoSA.

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

Third, the commander has several choices to obtain the architecture capability, either by its institution, by other institutions, or by consulting with relevant institutions. Therefore, the commander should balance the value of different actions to make the best decisions.

To address these challenges, we focus on a class of agent search problems where different architecture capabilities are independent of each other, and we expect to find a polynomial time optimal solution. Although the independence assumption narrows the generality of our model, it still covers an important part of the agent search problem. In this paper, we advance the state-of-the-art technology in the following ways:

- First, we propose a super-network model based CSoSA framework. This framework is established based on the core elements of system-of-systems capability generation, i.e. tasks, equipment, and command and control structures. Given this, we propose a CSoSA search framework and cast it as a dynamic programming problem under the constraint of the uncertainty of the architecture capability.
- Second, an exploration algorithm is proposed to search the optimal CSoSA. Specifically, we designed a greedy search based dynamic planning (GSDP) algorithm, where the GSDP algorithm is polynomial time and is proved to be optimal under the assumption of reward independence.
- Third, we empirically evaluate the performance of GSDP through simulation experiments, that are divided into parameter sensitivity analysis experiments and scalability analysis experiments. Several benchmark algorithms are compared with GSDP in the CSoSA search framework. The experimental results show that GSDP is significantly better than other benchmark algorithms.

The remainder of this paper is organized as follows. Related work of the combat system-of-systems architecture, the super-network model, and exploration algorithms for the CSoSA searching problem are discussed in Section II. The CSoSA searching problem framework is defined in Section III. Our proposed algorithm and a theoretical analysis are presented in Section IV. An empirical analysis of our algorithm is shown in Sections V. Finally, the paper is concluded in Section VI.

II. RELATED WORK

In this section, the combat system-of-systems architecture, the super-network model, and exploration algorithms for the CSoSA searching problem are reviewed.

A. COMBAT SYSTEM-OF-SYSTEMS ARCHITECTURES

The CSoS is quite complex, that covers all aspects of warfare [8]. Currently, many scholars have proposed their own understanding on the CSoS, but only formed a unified qualitative understanding in some conceptions [9]. The CSoS needs the capability to accomplish missions. The definition of a capability given by RAND is to accomplish a series of combat missions in a combined manner in order to achieve certain specific operational effects, subject to prepared environmental conditions [10]. It means that the capability is mainly generated by combining tasks, equipment systems, and the command and control structure. Since the CSoS is quite hard to study comprehensively and accurately, an alternative method is to study the architecture.

There are many methods that are widely used for CSoSA modeling, such as DoD Architecture Framework [11], MOD Architecture Framework [12] and NATO Architecture Framework [13]. These frameworks are comprehensive, but weaken the influence of command and control elements. Another method is the agent-based model, which is popular in the military. Each intelligent entity can be modeled as an agent, and realistic relationships can be defined as interactions between agents. The behavior between agents reflects swarm intelligence, that may generates the emergency [14]. In this paper, we intend to use the agent-based model to study CSoSA by taking capability generating elements into consideration. In the next section, we focus on the super-network model based CSoSA.

B. THE SUPER-NETWORK MODEL

Generally, it is believed that the supernetwork refers to a network with a large scale, complex connections, and heterogeneous nodes. Nagurney was the first to propose a concept on the supernetwork [15], [16]. When dealing with the problem of interlacing logistics networks with information networks and capital networks, she refers to networks that are beyond existing networks as supernetworks. In other words, a super network is a network of networks. The supernetwork studied by Nagurney has one or several of the following features: multi-layer feature, multi-level feature, multi-dimensional feature, and multi-attribute feature. Thus, the super-network model can be used to represent interactions and influences between networks with different featuers. In addition, Berge [17] also proposed the similar concept called hypernetwork, which is a hypergraph-based network. The hypergraph is different from the undirected graph or the directed graph, where each edge in the hypergraph can connect more than two nodes. We believe that the supernetwork is more in line with the CSoSA. In this paper, the super-network model CSoSA is a complex network with specific goals composed of multiple types of networks.

Currently, the supernetwork is use to model the combat system-of-systems. Fu-li *et al.* [18] proposed a military communication supernetwork structure in a network-centric environment. The network consists of five heterogeneous nodes: sensor nodes, information nodes, decision nodes, communication nodes, and effector nodes. Gao *et al.* [19] proposed a super-network model of command and control system, that includes observation nodes, command and control nodes, and effect nodes, and relationships between the three types of nodes. Zhao *et al.* [20] established a weapon equipment system super-network model based on the network-centric mode, and a granular analysis is proposed to reduce the

complexity of weapon system generation schemes based on the theory of quotient space. A multi-layer command and control super-network model is proposed based on attribute collaborative prioritization and hypergraph theory, including three levels and five types of networks [21]. The above super-network models are studied from the perspectives of equipment systems, functions and the command and control structure respectively. In fact, the CSoS is an organic whole, that needs to be guided by the capability. In this paper, a capability-oriented CSoSA is proposed based on the supernetwork model.

C. ALGORITHMS FOR THE CSOSA SEARCHING PROBLEM

In this paper, a CSoSA is established based on the super-network model. Each supernetwork is an architecture scheme with the capability to accomplish a specific mission. A set of supernetworks constitute an architecture scheme space. In order to select the optimal architecture scheme, the architecture scheme space exploration method is necessary. This problem can be modeled as an optimization problem. According to the number of objective functions, it can be divided into single-objective optimization and multi-objective optimization. Sometimes multiple objectives can be reorganized into a single objective for the convenience of analysis. For complex optimization problems, Pareto optimal methods such as NSGA-II [22], SPEA2 [23], and some extended algorithms [24] are commonly used. These algorithms apply to the problem that each input has a definite output. It is difficult to effectively solve the problem framework proposed in this paper, where each action may generate a large number of (possibly infinite) different value. Our problem framework can be transformed into a dynamic programming problem [25], [26], which is different from the current common dynamic programming problems, such as partially observable Markov decision problems [27]. Their objectives are to maximize the expected cumulative reward, while our objective is to select only the highest expected reward in the developed architecture scheme space. To solve the problem, a novel and optimal algorithm is then proposed.

III. THE CSOSA SEARCHING PROBLEM FRAMEWORK

In this section, we first give the definitions of the CSoSA. Then, the CSoSA searching problem is proposed. Third, the CSoSA searching problem is cast as a dynamic programming problem.

A. DEFINITIONS OF THE SUPERNETWORK BASED CSOSA

The capacity is generated by combining the equipment and non-equipment elements organically. When equipment systems with certain functions organized according to a command and control structure are able to complete a series of tasks, we believe that the CSoS has the ability to accomplish the mission. The performance of completing the task is measured by the operational effect, denoted as F. In fact, the operational effect has a great relationship with the equipment system, the command and control structure, and other

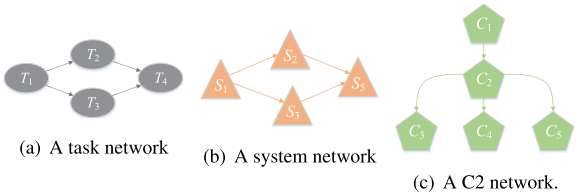


FIGURE 1. Examples of three types of networks. Fig. 1(a) shows the task network; Fig. 1(b) shows the system network; Fig. 1(c) shows the C2 network.

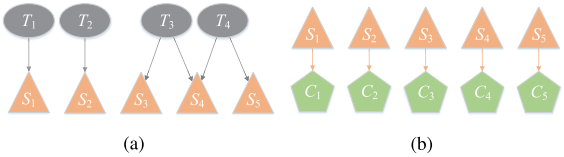


FIGURE 2. Examples of bipartite graphs. Fig. 2(a) shows the corresponding relationship between task nodes and system nodes; Fig. 2(b) shows the corresponding relationship between system nodes and C2 nodes.

factors that have not been considered. In addition, we believe that a supernetwork is a heterogeneous network that connects several types of nodes. Specifically, the supernetwork based CSoSA is composed of a task node network, a system node network, and a command and control node network. Here we give some definitions.

Definition 1 (Task Node): A task node is an atomic operational activity that can be executed by equipment, denoted as T.

In order to complete the mission, we need to decompose the mission into a series of task nodes, which we call the task link. A task link can be described as a directed graph. A mission can be decomposed into different task links, each mission link has a starting task node and an ending task node. Different task links may have different effects. There are mainly two logical relationships in a task link, that is sequential execution and parallel execution. Multiple missions correspond to multiple task links, which constitute the task network, denoted as $G^T = (V^T, E^T)$. Fig. 1(a) shows an example of a task network.

Definition 2 (System Node): A system node refers to an equipment that has the ability to accomplish certain tasks, denoted as S.

In the CSoSA, the connection of system nodes, such as drones, tanks, and machines, is influenced by task nodes. The system network is denoted as $G^S = (V^S, E^S)$. An example of the system network is shown in Fig. 1(b). Equipment with functions are used to complete tasks. A corresponding relationship between task nodes and system nodes is defined as a bipartite graph, denoted as $G^{TS} = (V^{TS}, E^{TS})$. An example of the bipartite graph is shown in Fig. 2(a).

Definition 3 (Command and Control Node): The command and control (C2) node is a logical node that is used to process information, denoted as C.

On one hand, the C2 node processes the command between the upper and lower levels, such as receiving task information from the superior C2 node, and releasing sub-task

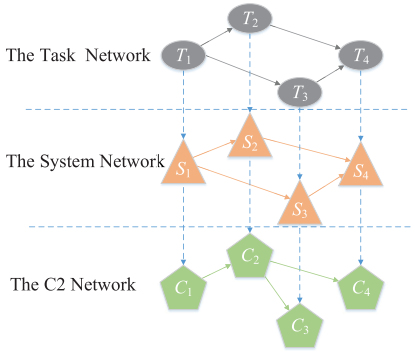


FIGURE 3. An example of the CSoSA scheme based on the super-network model.

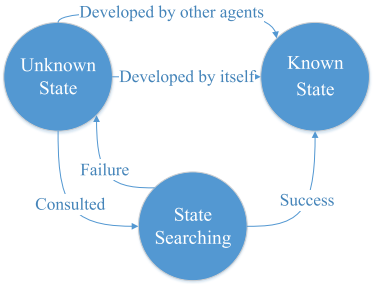


FIGURE 4. The state transition relationship. The unknown state indicates the scheme is undeveloped and its reward is unknown; the known state indicates the scheme is developed and its reward is known. The searching state indicates the scheme is being queried.

information to the subordinate C2 node. On the other hand, maintaining the information interaction with peer C2 nodes. Therefore, the relationship of all the C2 nodes in the CSoSA constitutes the organizational structure, denoted as $G^C = (V^C, E^C)$. An example of the C2 node network is shown in Fig. 1(c). Most of the organizational structure is modeled as but not limited to the tree graph. In addition, a corresponding relationship between system nodes and C2 nodes is defined as a bipartite graph, denoted as $G^{SC} = (V^{SC}, E^{SC})$. An example of the bipartite graph is shown in Fig. 2(b).

Fig. 3 shows an example of the CSoSA scheme based on the super-network model. A CSoSA scheme is built as follows: The task nodes network is first built. Specifically, the CSoS mission is decomposed into a number of feasible task networks. Second, the correspondence between task nodes and system nodes is given, and the system network is built according to the task network and the corresponding relationship. Third, the correspondence between system nodes and C2 nodes is given, and the C2 node is determined according to the relationship. However, the C2 network is not directly related to relationship between system nodes and task nodes, and it is constructed according to the feasible relationship between C2 nodes. Each supernetwork represents a scheme, and all supernetworks constitute the scheme space.

B. THE CSOSA SEARCHING PROBLEM

Assume that some drones are assigned to complete a reconnaissance mission in a complex environment. There are several schemes to complete the mission, but we can only choose one. It costs resources when developing a scheme, at the same time obtains a certain reward. In this section, we formalizes the problem from a more general perspective.

The CSoSA is a logical concept, and the actual operational capabilities is acquired through the development of the CSoS. Note that, the CSoS is likely to be affected by factors beyond the CSoSA, where these factors may result in emergency. Thus, the CSoS capability based on the CSoSA may not be unique, and each CSoSA scheme has an uncertain potential reward.

The commander needs to select an architecture scheme (scheme for short) in the scheme space to develop the CSoS. For simplicity, commanders, decision-making institutes, and consultancies are modeled as agents in this paper. The CSoS capability is measured by the sampled reward. The reward of each scheme x_k obeys the probability distribution $F_k(x_k)$, and the rewards of different schemes are independent of each other, where $k \in K, K = \{1, 2, \ldots, N\}$, and N is the number of schemes in the scheme space. The reward of each architecture is uncertain in advance, which is acquired by developing the CSoS, or by consulting other agents. The agent continually explores the scheme in the undeveloped scheme space and selects a developed scheme in the developed scheme space finally. The goal of the agent is to select an architecture with the highest expected reward and the least cumulative search costs.

Fig. 4 shows the state transitions. When exploring a scheme, the scheme has a potential reward. After the CSoS being developed, the scheme reward is known. Actions of the agent include: developing by itself, developing by other agents, and consulting the relevant agents. Specifically, after the CSoS being developed based on the scheme k with cost c_k^{self}, the unknown state transitions to a known state. In addition, the agent can request other agents to develop the CSoS with cost c_k^{other}. Furthermore, the agent can consult relevant agents, such as institutions or departments which may have done similar work. The cost of consultation process is denoted as c_k^{cons}.

CSoSASP

$$
\begin{aligned}
\max\ & E[\sum_{k\in K}(-d_k^{self}c_k^{self} - d_k^{other}c_k^{other} - n_k^{cons}c_k^{cons} \\
& + (d_k^{self}\beta_k^{self} + d_k^{other}\beta_k^{other} + d_k^{cons}\beta_k^{cons})s_k x_k)] \\
\text{s.t. } & d_k^{self} + d_k^{other} + d_k^{cons} \le 1, \quad k \in K \qquad (a) \\
& d_k^{self} + d_k^{other} + d_k^{cons} \ge s_k, \quad k \in K \qquad (b) \\
& \sum_{k\in K} s_k = 1, \qquad (c) \\
& d_k^{self}, d_k^{other}, d_k^{cons}, s_k \in \{0, 1\}, \quad k \in K \qquad (d) \\
& n_k^{cons} \in \mathbb{N}, \quad k \in K \qquad (e) \\
& \beta_k^{self}, \beta_k^{other}, \beta_k^{cons} \in [0, 1], \quad k \in K \qquad (f) \\
& c_k^{self}, c_k^{other}, c_k^{cons} \in \mathbb{R}^+, k \in K \qquad (g) \qquad (1)
\end{aligned}
$$

The objective function is for maximizing the reward of the developed architecture while minimizing the sum of

search costs. Specifically, the constraint (a) ensures that the scheme has been developed or not developed. The constraint (b) indicates that if a scheme is selected finally, then this scheme must have been developed. The constraint (c) indicates that only one scheme is finally selected. The constraint (d) represents the value spaces of the decision variables. The constraint (e) represents the number of times the agent requests relevant agents. The constraint (f) refers to the discount rate, which is the impact of development time on rewards. The constraint (g) indicates the cost of each action.

C. THE CSOSA DYNAMIC PROGRAMMING PROBLEM

In this section, we formalize the CSoSA searching problem into a dynamic programming problem and put forward an optimal algorithm. In the dynamic planning, we divide the scheme space K into two mutually exclusive sets: one is a growing set of developed architectures $D \in K$; the other is a decreasing set of undeveloped architectures $\bar{D} \in K$, $D \cup \bar{D} = K$. For each decision, the agent can choose whether to select and develop an unknown scheme from set $\bar{D}$ or stop searching and select one from set D. If the agent chooses to continue searching, then it has three type of actions, i.e. developing the CSoS by itself a_{self}, developing the CSoS by other agents a_{other}, and getting the reward by consulting the agents a_{cons}. If the agent stops searching, then it selects the developed scheme with the highest reward:

$$y = \max_{k \in K} x_k \tag{2}$$

The system state is defined as a statistic $(\bar{D}, y)$. In addition, the state evaluation function $\Psi(\bar{D}, y)$ is defined as the expected discount value by executing the optimal policy when the maximum reward is y and the undeveloped architecture set is $\bar{D}$. For each set $\bar{D}$ and the maximum reward, the state evaluation function needs to satisfy the following function.

$$\begin{aligned}
&\Psi(\bar{D}, y) \\
&\quad = \max\{y, \Psi^{self}(\bar{D}, y), \Psi^{other}(\bar{D}, y), \Psi^{cons}(\bar{D}, y)\} \\
&where \\
&\Psi^{self}(\bar{D}, y) \\
&\quad = \max_{k \in \bar{D}}\{-c_k^{self} + \beta_k^{self}[\Psi(\bar{D} - \{k\}, y) \int_{-\infty}^{y} dF_k(x_k) \\
&\qquad + \int_{y}^{\infty} \Psi(\bar{D} - \{k\}, x_k) dF_k(x_k)]\} \\
&\Psi^{other}(\bar{D}, y) \\
&\quad = \max_{k \in \bar{D}}\{-c_k^{other} + \beta_k^{other}[\Psi(\bar{D} - \{k\}, y) \int_{-\infty}^{y} dF_k(x_k) \\
&\qquad + \int_{y}^{\infty} \Psi(\bar{D} - \{k\}, x_k) dF_k(x_k)]\} \\
&\Psi^{cons}(\bar{D}, y) \\
&\quad = \max_{k \in \bar{D}}\{\frac{-c_k^{cons}}{p} + \beta_k^{cons}[\Psi(\bar{D} - \{k\}, y) \int_{-\infty}^{y} dF_k(x_k) \\
&\qquad + \int_{y}^{\infty} \Psi(\bar{D} - \{k\}, x_k) dF_k(x_k)]\}
\end{aligned} \tag{3}$$

The variables $\Psi^{self}(\bar{D}, y)$, $\Psi^{other}(\bar{D}, y)$, $\Psi^{cons}(\bar{D}, y)$ represent the state evaluation function after executing the action a_{self}, a_{other}, a_{cons} separately at the state $(\bar{D}, y)$. Furthermore, the variable p is the probability that the agent can obtain the required information. Specifically, the agent needs to compare the expected discount values generated by different type of actions:

- For the action a_{self}, if the sampled reward $x_k \leq y$, then the current maximum sampled reward remains at y, and the current expected state evaluation value is $-c_k^{self} + \beta_k^{self}\Psi(\bar{D} - \{k\}, y)$; If $x_k > y$, then the current maximum sampled reward will be updated to x_k, and the current expected state evaluation value is $-c_k^{self} + \beta_k \Psi(\bar{D} - \{k\}, x_k)$.
- For the action a_{other}, the current expected state evaluation value is $-c_k^{other} + \beta_k^{other}\Psi(\bar{D} - \{k\}, y)$ for $x_k \leq y$; the current expected state evaluation value is $-c_k^{other} + \beta_k \Psi(\bar{D} - \{k\}, x_k)$ for $x_k > y$.
- For the action a_{cons}, the current expected state evaluation value is $-\frac{c_k^{cons}}{p} + \beta_k^{cons}\Psi(\bar{D} - \{k\}, y)$ for $x_k \leq y$; the current expected state evaluation value is $-\frac{c_k^{cons}}{p} + \beta_k \Psi(\bar{D} - \{k\}, x_k)$ for $x_k > y$.

The CSoSA searching problem is converted to a dynamic planning problem based on Eq. 3. However, it is quite hard to directly solve the recursive state evaluation function in this dynamic programming problem. In the next section, we propose an algorithm for optimally solving problems in polynomial time.

IV. THE DYNAMIC PLANNING ALGORITHM

In this section, we propose a greedy search based dynamic planning (GSDP) algorithm to solve the CSoSA dynamic programming problem, where it makes decisions by judging defined indicators.

A. DECISION INDICATORS

In this section, we define indicators for executing the action a_{self}, a_{other}, and a_{cons} for each scheme k, where the indicators are z^{self}, z^{other}, and z^{cons} separately. The indicators z^{self}, z^{other}, and z^{cons} are similar to the concept of expected benefits, which are as follows:

$$\begin{aligned}
z_k^{self} &= -c_k^{self} + \beta_k^{self}[z_k^{self} \int_{-\infty}^{z_k^{self}} dF_k(x_k) + \int_{z_k^{self}}^{\infty} x_k dF_k(x_k)] \\
z_k^{other} &= -c_k^{other} + \beta_k^{other}[z_k^{other} \int_{-\infty}^{z_k^{other}} dF_k(x_k) \\
&\quad + \int_{z_k^{other}}^{\infty} x_k dF_k(x_k)] \\
z_k^{cons} &= -\frac{c_k^{cons}}{p} + \beta_k^{cons}[z_k^{cons} \int_{-\infty}^{z_k^{cons}} dF_k(x_k) \\
&\quad + \int_{z_k^{cons}}^{\infty} x_k dF_k(x_k)]
\end{aligned} \tag{4}$$

Algorithm 1: The GSDP Algorithm

procedure **Searching**($\bar{D}$)
begin
  for $k \in \bar{D}$ **do**
    $z_k^{self} \leftarrow Solving(c_k^{self} = \beta_k^{self} \int_{z_k^{self}}^{\infty}(x_k - z_k)dF_k(x_k) - (1-\beta_k^{self})z_k^{self})$;
    $z_k^{other} \leftarrow Solving(c_k^{other} = \beta_k^{other} \int_{z_k^{other}}^{\infty}(x_k - z_k)dF_k(x_k) - (1-\beta_k^{other})z_k^{other})$;
    $z_k^{cons} \leftarrow Solving(-\frac{c_k^{cons}}{p} = \beta_k^{cons} \int_{z_k^{cons}}^{\infty}(x_k - z_k)dF_k(x_k) - (1-\beta_k^{cons})z_k^{cons})$;
  $\pi \leftarrow Sorting(z_k^{self}, z_k^{other}, z_k^{cons} | k \in \bar{D})$;
  return SequrenceSearching($\pi, \bar{D}$)

Given the state $\Psi(\bar{D}, y)$ and the set of indicators c_k^{self}, c_k^{other}, c_k^{cons}, we can design a simple but optimal search rule that is divided into judgment rules and selection rules. The judgment rule means that if the agent is going to further explore a scheme with an unknown reward, the plan with the highest indicator is selected. Then, the agent executes the action corresponding to the highest indicator. The selection rule refers to that the agent stops searching and selects a developed scheme with the largest sampled reward.

B. THE GSDP ALGORITHM

The calculation of each indicator is independent and is not affected by the reward probability distribution of other schemes. The relationship of these three algorithms is that Algorithm 1 calls Algorithm 2, while Algorithm 2 calls Algorithm 3. Specifically, Algorithm 1 denotes the GSDP algorithm. Specifically, action indicators for all the schemes are first calculated according to the Eq. 4. Second, the indicators are sorted based on sorting methods, such as the heap sorting method [28], and the sorting result is saved to the vector π. Third, the *SequenceSearching* procedure is executed to get the optimal scheme.

Algorithm 2 denotes the *SequenceSearching* procedure. The best scheme can be selected after no more than N iterations. In each iteration, the current maximum sample reward y is compared with the maximum indicator $\pi(0)$. If $y \geq \pi(0)$, then the agent stops searching and the scheme with the maximum sampled reward y is final selected. Otherwise, the agent continues to search the scheme k by using the action a that corresponds to $\pi(0)$. If the agent gets the reward for scheme k, then the variables $D, \bar{D}, \pi, y$ are updated, where $\bar{D} \setminus k$ is a set of schemes that removes k from $\bar{D}$.

Algorithm 3 denotes the *Developing* procedure. If the action is a_{cons}, then there is a judgement as to whether the agent being consulted can query the reward of the scheme k. The symbol $\sim$ means a sampling, and $y_k \sim F_k(x_k)$ means a sampling of the probability distribution $F_k(x_k)$.

C. PERFORMANCE ANALYSIS

In this section, the complexity and optimality of the GSDP algorithm are analyzed below.

Algorithm 2: The Sequence Searching Algorithm

procedure **SequenceSearching**($\pi, \bar{D}$)
begin
  $y \leftarrow 0$;
  $D \leftarrow \emptyset$;
  for $i = 1 \rightarrow K$ **do**
    $k \leftarrow ParsingIndex(\pi(0))$;
    $a \leftarrow ParsingAction(\pi(0))$;
    if $y > \pi(0)$ **then**
      return y;
    else
      $(IsUpdated, y_k) \leftarrow Developing(a, k)$;
  if $IsUpdated = TRUE$ **then**
    $D \leftarrow D \cup k$;
    $\bar{D} \leftarrow \bar{D} \setminus k$;
    $\pi \leftarrow \pi \setminus \{z_k^{self}, z_k^{other}, z_k^{cons}\}$;
    if $y < y_k$ **then**
      $y \leftarrow y_k$;
  else
    $k \leftarrow k - 1$;

Algorithm 3: The Developing Algorithm

procedure **Developing**($\pi, \bar{D}$)
begin
  if $a = Counsultation$ **then**
    if $Available = TRUE$ **then**
      $IsUpdated \leftarrow TRUE$;
      $y_k \sim F_k(x_k)$;
    else
      $IsUpdated \leftarrow FALSE$;
      $y_k \leftarrow 0$;
  else
    $IsUpdated \leftarrow TRUE$;
    $y_k \sim F_k(x_k)$;

Theorem 1: GSDP is a polynomial time optimal algorithm for the CSoSA searching problem.

Proof 1: First, the time complexity of SGDP depends on the time complexity of the sorting algorithm. In the algorithm, the agent performs the actions sequentially based on the order of indicators, where the order does not change during the search process. Therefore, the complexity of SGDP is equal to the complexity of the sorting method. For example, the time complexity of heap sorting is $O(n \log n)$ [29], and the average time complexity of bubble sorting is $O(n^2)$ [30]. Second, SGDP is an optimal algorithm for the CSoSA dynamic programming problem. The CSoSA dynamic programming problem can be mapped to the classic Pandora

problem, that is used to study economics problem [31]. In the CSoSA dynamic programming problem, each developed scheme is seen as a project k with a sample reward r_k. A undeveloped scheme k can be seen as three projects k^{self}, k^{other}, k^{self}, each of which has a cost c_k^{self}, c_k^{other}, $\frac{c_k^{cons}}{p}$ separately, and a reward probability distribution $F_k(x_k)$. Once the sampled reward for scheme k is obtained, the three projects are moved into the explored set D. DSGP uses an index-based search strategy, that is, if the agent is to explore a new scheme, then it chooses the scheme with the highest indicator, otherwise it chooses a developed scheme with the largest sample reward. It proves that this kind of searching policy can effectively solve the Pandora problem and get the best expected reward [31]. Thus, Theorem 1 holds.

Further, an indicator function H is defined as follows:

$$H = \beta \int_z^{\infty} (x - z) dF(x) - (1 - \beta)z \tag{5}$$

Proposition 1: If the indicator functions of all schemes are the same and the relationships of $\{c_k^{self}, c_k^{other}, c_k^{cons} | k \in K\}$ are known, then the relationships of $z_k^{self}, z_k^{other}, z_k^{cons} | k \in K$ are determined.

Proof 2: The derivative function of the indicator function $H(z)$ for the variable z is as follows:

$$\dot{H}(z) = -\beta(x - z) - (1 - \beta) \tag{6}$$

Since $x \in [z, \infty]$, $\beta \in [0, 1]$, we know that $\dot{H}(z) \leq 0$. Based on the Eq. 4, it can be derived that $c = H(z)$. Thus the variable z is negatively correlated with c. Thus, Proposition 1 holds. This property facilitates preliminary judgment and may simplify the calculation process.

Proposition 2: The probability p is independent of indicators $\{z_k^{self}, z_k^{other} | k \in K\}$; and if the costs $\{c_k^{inst} | k \in K\}$ are fixed, then the probability p is positively correlated with $\{z_k^{cons} | k \in K\}$.

Proof 3: As mentioned in Eq. 4, each indicator is only related to its corresponding scheme, and not related to other schemes. Thus the probability p is independent of indicators $\{z_k^{self}, z_k^{other} | k \in K\}$. In addition, we know that $c_k^{cons} = pH(z_k^{cons})$, and the indicator function $H(z)$ decreases monotonically as the variable z increases according to Proposition 1. Then, when the cost c_k^{cons} is fixed, the probability p increases as z increases. This property indicates that the greater the probability p, the greater the expected reward of the scheme. Thus, Proposition 2 holds.

V. EMPIRICAL EVALUATION

In this section, the performance of the GSDP algorithm is analyzed by comparing with some benchmark algorithms.

A. EXPERIMENT SETUP

In the experiment, an assumed battle scenario is given, that a new CSoS is formed to complete some missions in a complex environment. In this paper, the CSoS is developed based on the CSoSA, where each CSoSA has an uncertain reward. Since it is uneconomical to search all CSoSA schemes, it is a challenging problem to select a best CSoSA with a maximum reward and a minimum total cost.

In order to evaluate the performance of the GSDP algorithm, we define the following statistical indicators:

- The average reward is the objective function shown in the Eq. 1, where the sampled reward in a simulation is the reward for the selected scheme minus the costs of all the developed schemes.
- The average number of consultations (*NoC*) is the average number of the agent that consult other agents.
- The average number of developed schemes (*NoD*) is the average number of schemes that has been developed.
- The average runtime time (*Time*) is the real time each algorithm runs.

The average reward and the runtime time are used to evaluate the performance of the algorithm, the number of consultations *NoC* and number of developed schemes *NoD* are used to analyze the searching process. In addition, the GSDP algorithm is compared with three benchmark algorithms under the CSoSA searching framework.

- Random algorithm (RA), is that the agent randomly selects an action in each decision. First, a scheme k is randomly selected in the set K. If the scheme has been developed $k \in D$, then the search is ended and the scheme is selected as the final scheme; if the scheme k is undeveloped $k \in \bar{D}$, then the agent randomly executes an action, such as a_{self}, a_{other}, a_{cons}, until the end of the search.
- Global Exploration Algorithm (GEA), is that the agent develops all the schemes and get the sampled reward of each scheme. Specifically, for an undeveloped scheme k, the agent executes the action with the least cost $a_k = \arg\min_{c_k} \{c_k^{self}, c_k^{other}, \frac{c_k^{cons}}{p}\}$. When the agent completes the development of all the schemes, the scheme with the highest reward is selected.
- Local Exploration Algorithm (LEA), is that the agent develops a scheme and chooses an action by judging an indicator. The searching process of the LEA is similar to that in GSDP, while the difference between them is that is the calculation of the indicator, where the indicators in LEA $\{\tilde{z}_k^{self}, \tilde{z}_k^{other}, \tilde{z}_k^{cons} | k \in K\}$ are $\{\beta E(x_k) - c_k^{self}, \beta E(x_k) - c_k^{other}, \beta E(x_k) - \frac{c_k^{cons}}{p} | k \in K\}$ separately. When the maximum reward in the developed schemes exceeds this indicator, the agent stops searching and selects the scheme with the highest reward.

These experiments run on a computer with a 2.6 GHz dual-core CPU, and 8 GB of RAM.

B. PARAMETER SENSITIVITY EXPERIMENTS

In this section, the sensitivity of the parameters is analyzed. As the example shown in Section III-A, the agent selects the best scheme in the scheme space with $K = 4$. In this experiment, let $\beta_k^{self} = \beta_k^{other} = \beta_k^{cons}, k \in K$, and let β denote the discount rate. Some of the common parameters are as follows:

TABLE 1. Parameter values.

variable	value	variable	value
$F1(x1)$	$U(5,15)$	$F2(x2)$	$U(8,20)$
$F3(x3)$	$U(7,18)$	$F4(x4)$	$U(3,17)$
c_1^{self}	0.5	c_2^{self}	2
c_3^{self}	1	c_4^{self}	1.5
c_1^{other}	2	c_2^{other}	5
c_3^{other}	4	c_4^{other}	3
c_1^{cons}	0.01	c_2^{cons}	0.05
c_3^{cons}	0.02	c_4^{cons}	0.03

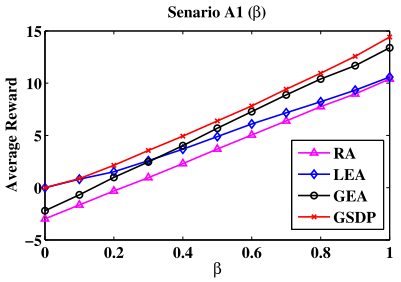


FIGURE 5. The average rewards of four algorithms in Scenario A1.

- Scenario A1: Let the probability p be 0.05, and the impact of the discount rate $\beta = \{0, 0.1, \ldots, 1\}$ is evaluated. Each algorithm runs 1000 simulations in each scenario.
- Scenario A2: Let the discount rate β be 0.95, and the impact of the probability $p = \{0, 0.1, \ldots, 1\}$ is evaluated. Each algorithm runs 1000 simulations in each scenario.

Fig. 5 shows the average rewards of four algorithms in Scenario A1. It shows that as the discount rate increases, the average reward gradually increases. The experimental results are in line with the actual situation, that is, the smaller the discount rate, the smaller the benefit, and vice versa. In addition, the reward of GSDP is slightly higher than the reward of other algorithms in all scenarios.

Fig. 6(a) shows the average *NoC* of the four algorithms in Scenario A1. It shows that the average *NoC* of GEA is significantly higher than other algorithms, and the average *NoC* of GSDP is similar to that of LEA. Fig. 6(b) shows the average *NoD* of the four algorithms in Scenario A1. Since GEA develops all the schemes, the average *NoD* is equal to the number of schemes. Note that the average *NoD* of GSDP is between 1-2, which is slightly higher than that of LEA and lower than that of RA and GEA.

Fig.7 shows the average rewards of four algorithms in scenario A2. It shows that as the probability p increases, the average reward gradually increases. The experiment result shows that the cost of consulting is often less than the cost of development in many scenarios. In addition, the average reward of GSDP is slightly higher than the reward of GEA, and it is significantly higher than the reward of RA and LEA in all scenarios.

Fig. 8(a) shows the average *NoC* of the four algorithms in Scenario A2. It shows that as the probability p increases,

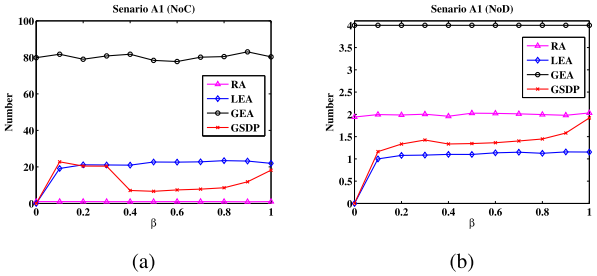


FIGURE 6. *NoC* and *NoD* of four algorithms in Scenario A1. Fig. 6(a) shows the *NoC* in Scenario A1; Fig. 6(b) shows the *NoD* in Scenario A1.

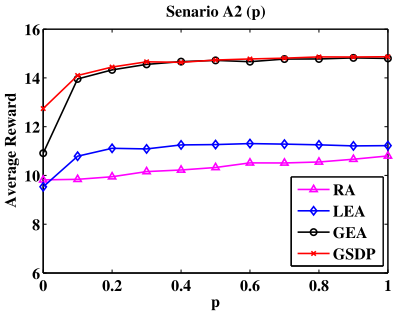


FIGURE 7. The average rewards of four algorithms in Scenario A2.

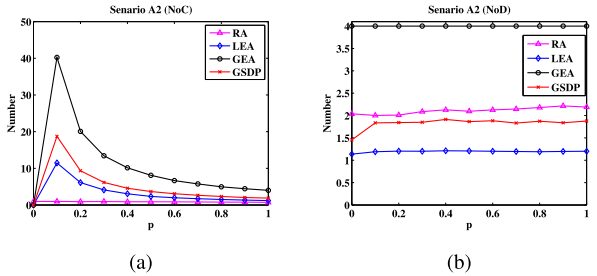


FIGURE 8. *NoC* and *NoD* of four algorithms in Scenario A2. Fig. 8(a) shows the *NoC* in Scenario A2; Fig. 8(b) shows the *NoD* in Scenario A2.

the *NoC* decreases gradually. Similar to the results in Scenario A1, the average NoC of GEA is s higher than other algorithms, and the average *NoC* of GSDP is similar to that of LEA. Fig. 8(b) shows the average *NoD* of the four algorithms in Scenario A2. Since GEA develops all the schemes, the average *NoD* is equal to the number of schemes. Note that the average *NoD* of GSDP is between 1.5-2, which is slightly higher than that of LEA and lower than that of RA and GEA.

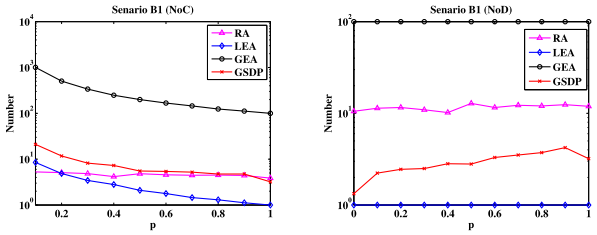
These results clearly illustrate that higher the β or p, the higher the average reward, that helps to improve the performance of these algorithms. As the discount rate β increases, the sampled reward will increase, while the probability p increases, the cost will decrease.

C. THE SCALABILITY EXPERIMENT

In this section, three scenarios are construct to evaluate the scalability of the scheme space on these algorithms. In general, the time and cost of the action a_{cons} is lest, the time of a_{other} is less than that of a_{self} and the cost of a_{other} is high than that of a_{self}, i.e. $c_k^{cons} < c_k^{self} < c_k^{other}$, $\beta_k^{cons} < \beta_k^{other} < \beta_k^{self}$. Some of the common parameters are as follows.

TABLE 2. Average rewards in Scenario B1.

p	0	0.1	0.2	0.3	0.4	0.5
GSDP	13.56	16.56	17.40	17.99	17.85	17.90
LEA	11.53	14.43	14.80	13.98	14.38	14.48
RA	-7.24	-7.04	-6.13	-4.41	-2.04	-3.07
GEA	-93.86	-33.47	-8.73	0.19	5.35	7.89
p	0.6	0.7	0.8	0.9	1	
GSDP	18.28	18.37	18.44	17.88	18.53	
LEA	14.20	14.21	14.72	14.90	13.33	
RA	-2.31	-3.68	-1.85	-2.44	-0.36	
GEA	9.44	10.5	11.47	12.55	12.86	

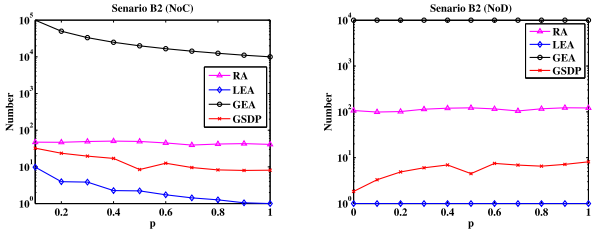


(a) (b)

FIGURE 9. *NoC* and *NoD* of four algorithms in Scenario B1. Fig. 9(a) shows the *NoC* in Scenario B1; Fig. 9(b) shows the *NoD* in Scenario B1.

TABLE 3. Average rewards in Scenario B2.

p	0	0.1	0.2	0.3	0.4	0.5
GSDP	14.60	18.48	18.81	18.89	19.00	19.12
LEA	12.67	14.45	14.61	15.18	14.95	15.63
RA(10^2)	-2.06	-1.79	-1.74	-1.91	-1.89	-1.83
GEA(10^3)	-10.72	-5.52	-2.72	-1.82	-1.35	-1.08
p	0.6	0.7	0.8	0.9	1	
GSDP	19.07	19.16	19.13	19.12	19.10	
LEA	14.54	15.20	14.72	15.28	14.46	
RA(10^2)	-1.69	-1.44	-1.54	-1.54	-1.49	
GEA(10^3)	-0.90	-0.77	-0.66	-0.59	-0.53	



(a) (b)

FIGURE 10. *NoC* and *NoD* of four algorithms in Scenario B2. Fig. 10(a) shows the *NoC* in Scenario B2; Fig. 10(b) shows the *NoD* in Scenario B2.

Let all the reward distributions $F_k(x_k)$ follow the uniform distribution, i.e. $F_k(x_k) \sim U(a_k, b_k)$, where $a_k \sim U(10, 15)$, $a_k \sim U(15, 20)$. In addition, let the costs be $c_k^{self} \sim U(0.2, 2)$, $c_k^{other} \sim U(1, 5)$, $c_k^{cons} \sim U(0.01, 0.1)$, and the discount rate $\beta_k^{self} \sim U(0.75, 0.85)$, $\beta_k^{other} \sim U(0.85, 0.95)$, $\beta_k^{self} \sim U(0.95, 1)$, $k \in K$. Let the probability be $p = \{0, 0.1, \ldots, 1\}$, and each algorithm runs 1000 simulations in each scenario.

- Scenario B1: The agent searches the best scheme in the scheme space with $K = 10^2$.
- Scenario B2: The agent searches the best scheme in the scheme space with $K = 10^4$.
- Scenario B3: The agent searches the best scheme in the scheme space with $K = 10^6$.

The average rewards of four algorithms in scenario B1 are tabulated in Table 2. It shows that as the probability p increases, the average reward increases. The reward of GSDP is much higher than that of other algorithms in all scenarios.

Fig. 9(a) shows the average *NoC* of the four algorithms in Scenario B1. It shows that as the probability p increases, the *NoC* decreases gradually. The *NoC* of GEA is significantly higher than that of other algorithms. The RA, LEA and GSDP have similar *NoC*. Fig. 9(b) shows the average *NoD* of the four algorithms in Scenario B1. It shows that the average *NoD* of GSDP is between 1-4 and the *NoD* of LEA developments is 1 in these scenarios.

The average rewards of four algorithms in scenario B2 are tabulated in Table. 3. It shows that as the probability p increases, the average reward of GSDP gradually increases and converges. GSDP has a higher average reward than that of other algorithms in all scenarios.

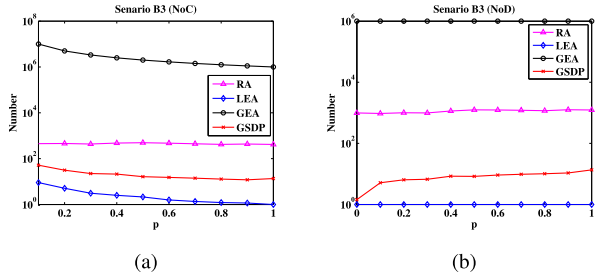
TABLE 4. Average rewards in Scenario B2.

p	0	0.1	0.2	0.3	0.4	0.5
GSDP	15.55	18.72	19.06	19.28	19.28	19.38
LEA	12.82	14.20	15.19	14.98	14.76	15.01
RA(10^3)	-2.04	-1.87	-1.88	-1.78	-1.96	-2.05
GEA(10^5)	-10.77	-5.50	-2.75	-1.83	-1.38	-1.10
p	0.6	0.7	0.8	0.9	1	
GSDP	19.40	19.45	19.42	19.52	19.50	
LEA	15.05	14.73	14.81	14.90	14.59	
RA(10^3)	-1.96	-1.85	-1.72	-1.80	-1.69	
GEA(10^5)	-0.92	-0.79	-0.69	-0.61	-0.55	

Fig. 10(a) shows the average *NoC* of the four algorithms in Scenario B2. The *NoC* of GEA is significantly higher than that of other algorithms. The *NoC* of GSDP is between RA and LEA. Fig. 10(b) shows the average *NoD* of the four algorithms in Scenario B2. It shows the average *NoD* of GSDP is less than 10 and the average *NoD* of LEA developments is always 1 in these scenarios.

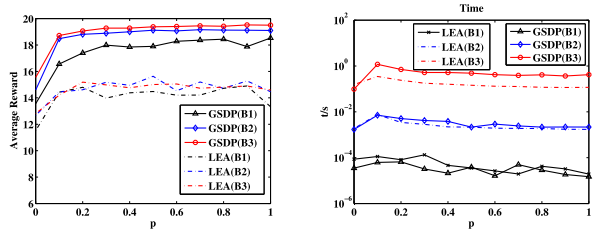
The average rewards of four algorithms in Scenario B3 are tabulated in Table. 4. Similar to the results in Scenario B2, as the probability p increases, the average reward of GSDP gradually increases and converges. GSDP has a higher average reward than that of other algorithms in all scenarios.

Fig. 11(a) shows the average *NoC* of the four algorithms in Scenario B3. The *NoC* of GEA is significantly higher than that of other algorithms. The *NoC* of GSDP is between RA and LEA. Fig. 11(b) shows the average *NoD* of the four algorithms in Scenario B3. It shows the average *NoD* of GSDP is about 20 and the average *NoD* of LEA developments is always 1 in these scenarios.



(a) (b)

FIGURE 11. ***NoC*** **and** ***NoD*** **of four algorithms in Scenario B3. Fig. 11(a) shows the** ***NoC*** **in Scenario B3; Fig. 11(b) shows the** ***NoD*** **in Scenario B3.**



(a) (b)

FIGURE 12. **Average rewards and average running times of GSDP and LEA in Scenarios B1, B2, B3. Fig. 12(a) shows the average rewards; Fig. 12(b) shows the average running times.**

Further, the average rewards and average running times in Scenario B1, Scenario B2, and Scenario B3, are comprehensively analyzed. The average rewards of GSDP and LEA in these scenarios are shown in Fig. 12(a). It shows that the average reward of GSDP is higher than that of LEA in these scenarios, and as the increasing of the scheme space, the average reward of GSDP also increases. The running time of GSDP and LEA in these scenarios are shown in Fig. 12(b). As the increasing of the scheme space, the average runtime of GSPD shows a polynomial increase. In Scenario B1 and Scenario B2, the running time of GSPD and LEA is similar. In Scenario B3, the running time of GSDP is slightly higher than that of LEA.

In general, the experiment results show that GSDP is significantly better than the benchmarking algorithms. In addition, rewards can be significantly improved through good consulting and high discount rates. It is because that GSDP only develops a small number of schemes and stop to choose the scheme with the highest reward, and GSDP greedily selects schemes by using an indicator-based algorithm, where these indicators can assist in searching the optimal scheme.

VI. CONCLUSION

The CSoSA optimization method is an important attempt of the system-of-systems engineering in the network-centric warfare, that provides an idea for guiding the development of the CSoS. In this paper, we first propose a super-network model for the CSoSA, which covers the elements of the CSoS capability generating. Second, a novel model is proposed to formalize the architecture searching problem under the environment with uncertain knowledge and assists of other agents. Specifically, we consider the scenario that an agent explores several architectures and selects the optimal architecture, where the performance of these architectures to be explored is unknown until the agent explores or other agents query. A polynomial time optimal scheme exploration method is proposed to solve the formulation, where our algorithm is much better than other benchmarking algorithms. In addition, our algorithm can be extended to some more general problems, such as maximizing a general function with the collection of all explored architectures. Future work will focus on the problem that the agent is going to select a given number of optimal architectures from the architectural space. Further, we hope to study the problem of parallel collaborative search of multiple agents.

REFERENCES

[1] X. Zhou, W. Wang, T. Wang, X. Li, and Z. Li, "A research framework on mission planning of the UAV swarm," in *Proc. 12th Syst. Syst. Eng. Conf. (SoSE)*, Jun. 2017, pp. 1–6.

[2] 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.

[3] 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.

[4] B. Zhao, X. Wang, D. Lin, M. Calvin, J. 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.

[5] 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.

[6] Z. Shu, W. Wang, and R. Wang, "Design of an optimized architecture for manned and unmanned combat system-of-systems: Formulation and coevolutionary optimization," *IEEE Access*, vol. 6, pp. 52725–52740, 2018.

[7] Y. Dou, L. Li, Q. Zhao, and Y. Chen, "Research on capability requirements generation of Weapon System-of-systems based on CRTAM model," in *Proc. 7th Int. Conf. Syst. Syst. Eng. (SoSE)*, Jul. 2012, pp. 185–190.

[8] D. Konur and C. H. Dagli, "Military system of systems architecting with individual system contracts," *Optim. Lett.*, vol. 9, no. 8, pp. 1749–1767, Dec. 2015.

[9] S. Sommerer, M. D. Quevara, M. A. Landis, J. M. Rizzuto, J. M. Sheppard, and C. J. Qrant, "Systems-of-systems engineering in air and missile defense," *Johns Hopkins APL Tech. Dig.*, vol. 31, no. 1, pp. 5–20, 2012.

[10] Chairman of the Joint Chiefs of Staff (CJCS), "Operation of the joint capabilities integration and development system," The Pentagon, Washington, DC, USA, Tech. Rep. CJCS Manual3170.01C, May 2007.

[11] DoD Architecture Framework Working Group, "DoD architecture framework version 1.0," Dept. Defense, 2003.

[12] I. Bailey, "Brief introduction to MODAF with v1.2 updates," in *Proc. IET Seminar Enterprise Archit. Frameworks*, 2008, pp. 1–18.

[13] H. A. H. Handley and R. J. Smillie, "Architecture framework human view: The NATO approach," *Syst. Eng.*, vol. 11, no. 2, pp. 156–164, Jun. 2008.

[14] P. Uday, R. Chandrahasa, and K. Marais, "System importance measures: Definitions and application to system-of-systems analysis," *Rel. Eng. Syst. Saf.*, vol. 191, Nov. 2019, Art. no. 106582.

[15] A. Nagurney and J. Dong, *Supernetworks: Decision-Making for the Information Age*. Cheltenham, U.K.: Edward Elgar Publishing, 2002.

[16] A. Nagurney and T. Wakolbinger, "Supernetworks: An introduction to the concept and its applications with a specific focus on knowledge supernetworks," *Int. J. Knowl., Culture Change Manage.*, vol. 4, 2005, pp. 1–16.

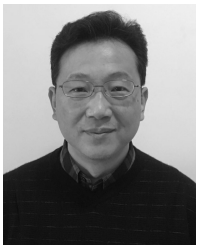
[17] C. Berge, *Graphs and Hypergraphs*. Amsterdam, The Netherlands: North Holland, 1973.

[18] 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.
[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] 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.
[21] 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.
[22] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[23] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proc. Evol. Methods Design Optim. Control Appl. Ind. Problems*, K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou, and T. Fogarty, Eds. Athens, Greece, 2001, pp. 95–100.
[24] Z. Shu and W. Wang, "Preference-inspired co-evolutionary algorithms with local PCA oriented goal vectors for many-objective optimization," *IEEE Access*, vol. 6, pp. 68701–68715, 2018.
[25] X. Zhou, W. Wang, Y. Zhu, T. Wanga, and B. Zhang, "Centralized patrolling with weakly-coupled agents using Monte Carlo tree search," *IEEE Access*, to be published.
[26] D. P. Bertsekas, *Abstract Dynamic Programming*. Belmont, MA, USA: Athena Scientific, 2018.
[27] 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.*, to be published.
[28] S. Rustagi and T. Yadav, "Algorithms 'optimised sorting process,"' *Bhagwan Parshuram Inst. Technol.*, vol. 4, no. 1, p. 52, 2018.
[29] C. J. H. McDiarmid and B. A. Reed, "Building heaps fast," *J. Algorithms*, vol. 10, no. 3, pp. 352–365, 1989.
[30] W. Min, "Analysis on bubble sort algorithm optimization," in *Proc. Int. Forum Inf. Technol. Appl.*, vol. 1, 2010, pp. 208–211.
[31] M. L. Weitzman, "Optimal search for the best alternative," *Econometrica, J. Econ. Soc.*, vol. 47, no. 3, pp. 641–654, 1979.

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

XIN ZHOU received the master's degree in systems simulation from the National University of Defense Technology (NUDT), Changsha, China, where he is currently pursuing the Ph.D. degree in systems engineering. His research interests include 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, where he is currently a Professor. His research interests include systems engineering and simulation.



YIFAN ZHU received the Ph.D. degree in systems engineering from the National University of Defense Technology (NUDT), Changsha, China, where he is currently a Professor. His research interests include systems engineering and simulation.

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