IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 21, 2019, accepted February 19, 2019, date of publication February 27, 2019, date of current version March 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2902043

Capability Construction of C4ISR Based on AI Planning

ZHIQIANG JIAO[1], PEIYANG YAO[2], JIEYONG ZHANG[2], LUJUN WAN[3], AND XUN WANG[1]

[1]Graduate College, Air Force Engineering University, Xi'an 710077, China
[2]Information and Navigation College, Air Force Engineering University, Xi'an 710077, China
[3]ACT Navigation, Air Force Engineering University, Xi'an 710077, China

Corresponding author: Zhiqiang Jiao (jzq_paper@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61573017 and Grant 61703425, in part by the Natural Science Foundation of Shaanxi Province under Grant 2016JQ6062 and Grant 2017JM6062, and in part by the Aeronautical Science Foundation of China under Grant 20175796014.

ABSTRACT This paper considers a capability construction problem of the C4ISR system under service-oriented architecture. A capability construction model is first established and described in the planning domain definition language as an artificial intelligence (AI) planning problem. To adapt the complex requirements of a C4ISR system and large scale of required services, an incremental macro-operation learning method based on n-gram analysis is proposed, and an enhanced domain is generated using a relaxation scheme. To improve the efficiency of the search algorithm, an ordered-hill-climbing (OHC) method is designed based on the length of the operations. With the above procedures, the AI planner, using macro-operation and the OHC, is presented for capability construction problems. The simulation results show that this method can effectively shorten the search time of capability construction.

INDEX TERMS Artificial intelligence planning, C4ISR, capability construction, service-oriented architecture.

I. INTRODUCTION

A new revolution under the promotion of information technology has produced a new form of war. To win the information war in the future, the C4ISR system needs to be strengthened. Developing and accelerating the formation of information-based system-combat capability, has become the inevitable choice of the army [1]. In recent years, with the rapid development of emerging technologies, such as service computing and cloud computing, the application of service-oriented architecture (SOA) in C4ISR has become imperative.

From the perspective of the entire system construction process, building a C4ISR system based on tasks under an SOA architecture mainly requires mapping from task to capability, capability to service, and service to resource [2]. The capability-to-service mapping is the main research content of this paper. It is the bridge between the system capability and the service, and plays a key role in the construction of the C4ISR. System capability describes the expectations of capabilities or performance of the C4ISR system from the user perspective, and the design, release, and deployment of services are generally implemented by IT technicians, such as software engineering developers. So, how to balance operational knowledge and IT technology when completing capability-to-service mapping is the problem to be solved in the construction of C4ISR system.

Traditionally, the solution to this problem is to use expert knowledge to build a capability-service template library and then use the template matching method to obtain the corresponding service combination. Clearly, this manual-based approach is time consuming and costly. At the same time, to generate new capability requirements that do not exist in template libraries, the above method will completely fail and must be re-analyzed by experts in the relevant field. The construction time of the system capability will be delayed under these circumstances. To solve this problem, a capability-service mapping method for the C4ISR system is proposed based on AI planning. This method uses IOPE (*Input*, *Output*, *Precondition* and *Effect*) to model system capabilities and services, then describe the problem as an AI planning problem in the planning-problem-description language (PDDL). For the characteristics of the problem, a planner based on macro-operation and ordered-hill-climbing (OHC) is designed.

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

The use of AI planning to complete capability-service mapping can effectively avoid the failure caused by new capability requirements in the template library method and greatly reduce manual participation to shorten the construction time of the system capability.

II. RELATED WORK

AI planning is an important branch of artificial intelligence research. Research on AI planning can be traced back to the logic theorist program designed by Newell and Simon in the 1960s. After years of development, AI planning technology continues to mature and has been successfully applied to cloud management [3], automatic control [4], task allocation [5], service composition [6], and other aspects.

The task of AI planning is to find a series of effective actions in a given planning domain, to ensure that the initial state in the planning problem can be successfully transferred to the goal state after applying the actions. Since the first international planning competition (IPC) was held, many high-performance planners have been proposed, such as FF [7], SHOP2 [8], and JLU-RLAO [9], and they all have good performance in their respective fields. It is worth mentioning that the FF planner, with outstanding performance in the classical planning domain, uses a heuristic search strategy to search the state and is combined with effective motion filtering technology to solve the classical programming problem, which provides a feasible paradigm for the subsequent planners. The Metric-FF [10] and Conformant-FF [11], which are based on FF, also performed well in competitions. On this basis, some scholars began to ameliorate the heuristic search-based planner to improve its planning efficiency. This involved two general ideas. One idea is to find a more efficient method for calculating heuristic values. Examples are the heuristic function based on delay partial reasoning in [12] and the landmark-based heuristic function in [13]. The other idea is to improve the efficiency of the search algorithm, such as the BDD-based search algorithm proposed in [14] and the OHC search algorithm proposed in [15].

However, as the number of operations in the planning domain continues to increase, the time for heuristic search methods will also grow exponentially, and some studies have begun to focus on using domain knowledge to speed up the planning process. Richard *et al.* [16] first proposed the concept of macro-operations in 1971. Macro-operations are composite operations composed of a series of atomic operations in the domain, which can be considered as domain knowledge. Afterwards, many studies began to focus on designing the macro-operation extraction method, which can be classified into two categories: one method is to directly analyze the atomic operations in the planning domain. Botea *et al.* [17] combined the atomic operations into a macro-operation by analyzing the interface dependencies between the atomic operations in the planning domain, and the implemented Macro-FF planner exhibits good planning performance. The other method is to extract effective macro-operations through learning methods based on existing planning solutions. Dulac *et al.* [18] used the N-grams analysis method to extract macro-operations from existing planning solutions to improve planning efficiency. With the help of a helpful action filter, only the macro-operations that are beneficial to the planning process can be selected into the enhance domain. In this way, the extra computational burden of excessive macro-operations is reduced. Jiang *et al.* [19] extracted the macro-operation by analyzing the relationship between each atomic operation in the existing planning solution and gives each macro-operation a certain heuristic value. By dynamically adjusting the heuristic value in the planning process, a better advantageous operation can be selected into the solution.

The above method is offline when extracting macro-operations, and the macro-operation library cannot be updated in real time during the planning process. Particularly for the second way of macro-operation extraction, if the planning solution samples used in the learning phase are insufficient or if the sample is not representative, the obtained macro-operation will not be able to play a role in the subsequent planning process, which could reducing the planning efficiency. (In [19], although the heuristic value of the macro-operation was dynamically adjusted during the planning process, the macro-operation library did not change). Therefore, much research on dynamic macro-operation generation has emerged in recent years. Reference [20] studied the online macro-operation generation method. By the analysis of the external relationship between operation and state, the macro-operation, which is beneficial to the planning process, is obtained, and the effectiveness of the generated macro-operation is improved. In [21], the VMSP algorithm, which is designed for data mining, is used to mine the sequence patterns with a high occurrence rate in existing planning solutions and merge them into macro-operations. Since VMSP is an incremental data mining algorithm, the planner has the ability to update macro-operation libraries in real time.

In general, the use of macro-operations to improve the efficiency of AI planners has become an important research direction. Many AI planners based on macro-operations have also been successfully applied to practical systems [23], [24]. As a result, this study involves the modeling of the mapping problem of the system capability and service in the C4ISR system as an AI planning problem. For the complex system capacity and large service scale, an AI planner based on macro-operation and OHC is proposed, with the simulation results showing the effectiveness of the planner.

III. PROBLEM STATEMENT

In the practice, it's hard to find a separate service that can satisfy the complex system capability requirement. As a result, the essence of the capability construction is to find a reasonable service composition which can satisfy the system capability requirements. The construction process is similar to the web service composition [25], but in the C4ISR system,

the service is different form the web service and the system capability requirement can be more complex than the web service request. In this paper, we assume that all services in C4ISR system are definite. That is to say, given the specific input and precondition, the output and effect of service are known to us. To illustrate the capability construction problem clearly, the following concepts need to be defined.

A. INPUT/OUTPUT PARAMETERS

The parameter set *Para* represents the set of all input/output parameters in the system. It is mainly composed of the situation information set *sInfo* and the command and control information set *cInfo*, namely, $Para = sInfo \cup cInfo$. A situation information $S \in sInfo$ can be defined as a seven-tuple:

$$S = \langle sname,\ sid,\ stime,\ slocation, starget,\ stype,\ sstate\rangle \quad (1)$$

where *sname*, *sid*, *stime*, *slocation*, and *starget* denote the name, id, time, location, and target of the situation information, respectively. The parameter *stype* indicates the type of the situation information, e.g., text, image, audio, and video. Parameter *sstate* represents the specific state of the target, in which the data type is determined by *stype*.

Similarly, a command and control (C2) information $C \in cInfo$ can be defined as a seven-tuple:

$$C = \langle cname,\ cid,\ ctime,\ chigh,\ csub,\ ctargrt,\ cdir\rangle \quad (2)$$

where *cname*, *cid*, and *ctime* denote the name, id, and time of the C2 information, respectively. The sender and receiver of the C2 information are indicated by *chigh* and *csub*, while *ctargrt* is the target involved in C, and *cdir* represents the specific command and control orders.

B. STATE SET

The state set *State* includes all possible states during the execution of the system and describes the situation of the combat units and the battlefield environment. A first-order language is used to describe the state, e.g., a state $p(name_1, name_2, \ldots, name_1) \in State$, where p is the predicate, and $name_i (i = 1, 2, \ldots n)$ are the variables.

In this paper, all the *precondition* and *effect* can be expressed by state set. When the concepts of input/output parameters and state set are defined, the system capability model and system service model can be described based on *input*, *output*, *precondition* and *effect*.

C. SYSTEM CAPABILITY MODEL

System capability is the large granularity function that C4ISR system must have to accomplish a task. It can be defined as a five-tuple:

$$capability = \langle CName, CInput, COutput, CPrecondition, Ceffect\rangle \quad (3)$$

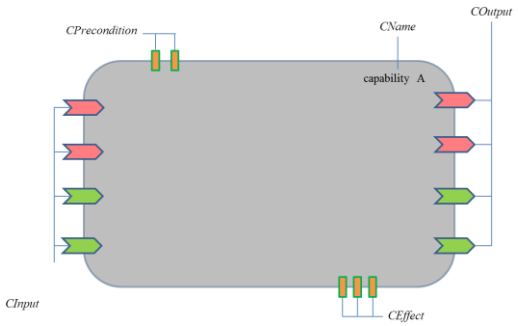


FIGURE 1. Schematic diagram of system capability model.

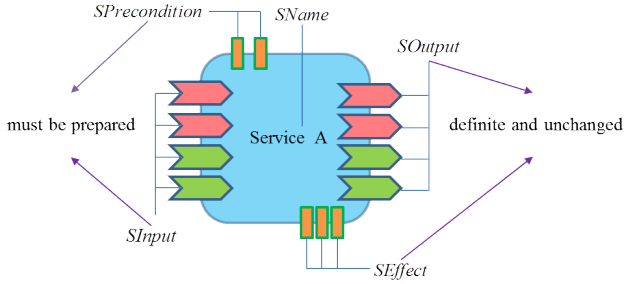


FIGURE 2. Schematic diagram of service model.

where *CName* is the name of the system capability. $CInput = \langle CisInfo, CicInfo\rangle$ is a two-tuple, which gives the information demanded by the system capability. $CisInfo \subset sInfo$ and $CicInfo \subset cInfo$ represent the situation information input set and C2 information input set, respectively. $COutput = \langle CosInfo, CocInfo\rangle$ is a two-tuple, which gives the information produced by the system capability. $CosInfo \in sInfo$ and $CocInfo \in cInfo$ represent the situation information output set and C2 information output set, respectively. $CPrecondition \subseteq State$ and $CEffect \subseteq State$ represent the precondition and effect of the system capability, respectively. A schematic diagram of the system capability model is shown in FIGURE 1.

D. SERVICE MODEL

The service can be defined as a five-tuple:

$$Service = \langle SName, SInput, SOutput, SPrecondition, SEffect\rangle \quad (4)$$

where *SName* is the name of the service. $SInput = \langle SisInfo, SicInfo\rangle$ is a two-tuple, which gives the information demanded by the service. $SisInfo \subset sInfo$ and $SicInfo \subset cInfo$ represent the situation information input set and C2 information input set, respectively. $SOutput = \langle SosInfo, SocInfo\rangle$ is a two-tuple, which gives the information produced by the service. $SosInfo \subset sInfo$ and $SocInfo \subset cInfo$ represent the situation information output set and C2 information output set, respectively. $SPrecondition \subseteq State$ and $SEffect \subseteq State$ are the initial and goal states of the service. A schematic diagram of the service model is shown in FIGURE 2.

Given a system capability requirement, if there does not exist one appropriate service in library, we need to merge

several atom services to satisfy the requirement. As we can see in the definition, when a service need to be called, the corresponding *input* and *precondition* should be prepared. That is to say, not all services can be merged together. To elaborate the composition constraints, the capability construction scheme is defined as follow.

E. CAPABILITY CONSTRUCTION SCHEME

The capability construction scheme can be defined as a six-tuple:

$$scheme^* = \langle SName^*, es, SInput^*, SOutput^*, Sin^*, Sfin^* \rangle \quad (5)$$

where $SName^*$ is the name of scheme. Sequence $es = (Service_1, Service_2, \ldots, Service_n)$ is the service sequence. $SInput^* = \langle SisInfo^*, SicInfo^* \rangle$ is a two-tuple, which gives the information demanded by the scheme. $SisInfo^* \subset sInfo$ and $SicInfo^* \subset cInfo$ represent the situation information input set and C2 information input set, respectively. $SOutput^* = \langle AosInfo^*, AocInfo^* \rangle$ is a two-tuple, which gives the information produced by the scheme. $SosInfo^* \in sInfo$ and $SocInfo^* \in cInfo$ represent the situation information output set and C2 information output set, respectively. $Sin^* \subseteq State$ and $Sfin^* \subseteq State$ are the initial and goal states of the scheme. To ensure that every service in scheme can be called successfully, the following constraints should be satisfied:

a) $Service_1.SInput \subseteq scheme^*.SInput^*$

b) $Service_1.SPrecondition \subseteq scheme^*.Sin^*$

c) $Service_{i+1}.SInput \subseteq scheme^*.SInput^* \cup (\bigcup_{k=1}^{i} Service_k.Soutput)$, $i = 1, 2, \ldots, n-1$

d) $Service_{i+1}.SPrecondition \subseteq scheme^*.Sin^* \cup (\bigcup_{k=1}^{i} Service_k.SEffect)$, $i = 1, 2, \ldots, n-1$

At the same time, the *output* and *effect* of the capability construction scheme cannot exceed the union of all the services' *output* and *effect* of in the scheme (constraint e) and f)).

e) $scheme^*.SOutput^* \subseteq \bigcup_{k=1}^{i} Service_k.SOutput$

f) $scheme^*.SFin^* \subseteq \bigcup_{k=1}^{i} Service_k.SEffect$

If one of the above constraints (constraints a) – f)) cannot be satisfied, the capability construction scheme is invalid.

However, the constraints a) – f) can only guarantee the effectiveness of scheme. We need the constraints g) – i) to ensure that the scheme can be considered to match the capability (record as $scheme^* \prec capability$).

g) $scheme^*.SInput^* \subseteq capability.AInput$

h) $scheme^*.Sin^* \subseteq capability.Ain$

h) $capability.Aoutput \subseteq scheme^*.Soutput^*$

i) $capability.AFin \subseteq scheme^*.SFin^*$

A schematic diagram of the capability construction scheme model is shown in FIGURE 3.

Now, by defining the above concepts, the construction of a system capability can be described as: for a given capability and service library, find a set of services in a service library, by which the corresponding capability construction scheme can match the capability.

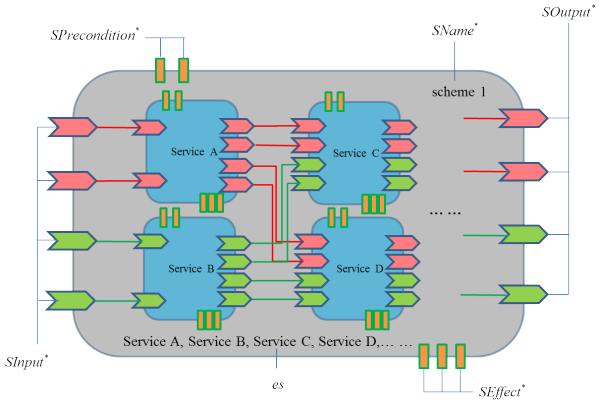


FIGURE 3. Schematic diagram of capability construction scheme model.

TABLE 1. Parameters and states of the problem.

Number	Name	Kind(○sInfo/△cInfo/□State)
1	Target Information	○
2	Power Information	○
3	Required Power Information	○
4	Missile Kind	○
5	Charge Number	○
6	Damage Effect	○
7	Strike Relationship	○
8	Safe Limit	○
9	Strike Time	○
10	Strike Distance	○
11	Strike Plan	△
12	Ground To Ground	□
13	Plan Approved	□
14	Plan Distributed	□

Taking the demand for fire allocation capability of ground-to-ground strikes in C4ISR systems as an example [26], the system is required to generate a strike plan and send it to each combat unit after approval by the superior. The parameters and states of the problem are shown in TABLE 1 TABLE 2 and TABLE 3 show the capability model and service model. Finally, a capability construction scheme that matches the demand can be seen in FIGURE 4.

IV. PDDL AND MODEL COVERT

Due to the similarity between the above capability construction and AI planning, this study considers converting the problem into an AI planning problem, and the PDDL is employed to describe the capability construction problem.

PDDL is a standardized planning-problem-description language widely used to describe AI problems. In PDDL, a planning problem consists of two files, a domain description file and a problem description file. The domain description file consists of predicates and parameterized actions, and the problem description file contains the object of the problem, the initial state, and the goal state. This way of separately

TABLE 2. Parameters and states of the problem.

CName	*CInput*		*COutput*		*CPrecondition*	*CEffect*
	SisInfo	*SicInfo*	*SosInfo*	*SocInfo*		
FD	(1)(2)	-	-	(11)	(12)	(13)

TABLE 3. Service model of the service library.

SName	*SInput*		*SOutput*		*SPrecondition*	*SEffect*
	SisInfo	*SicInfo*	*SosInfo*	*SocInfo*		
EvalPower	(1)(2)	-	(3)	-	(12)	-
EvalShell	(1)(3)	-	(4)		(12)	
EvalCharge	(1)(3)(4)	-	(5)	-	(12)	
CalcBestRn	(1)(3)(4)(5)	-	(6)		(12)	
TaskAssign	(1)(3)(6)	-	(7)	-	(12)	
CalcSafeLimit	(1)(3)(4)(5)(7)	-	(8)	-	(12)	
CalcTime	(1)(3)(4)(5)(7)	-	(9)	-	(12)	
CalcDistance	(1)(3)(4)(5)(7)	-	(10)	-	(12)	
PlanApproval	(1)(3)(4)(5)(7) (8)(9)(10)	-	-	(11)	(12)	(13)
PlanTranslated	-	(11)	-	-	(12) (13)	(14)

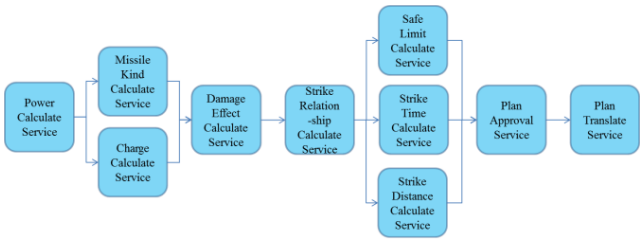


FIGURE 4. Capability construction scheme.

TABLE 4. Mapping rules between service/capability model and PDDL.

Service Model	PDDL	Capability Model	PDDL
Para	object	*Para*	object
p	predicate	*p*	predicate
SName	Action Name	*CName*	Problem Name
SInput *SPrecondition*	Precondition	*CInput* *Cin*	Init
SOutput *SEffect*	Effect	*COutput* *CEffect*	Goal

describing the problem and the domain causes the description of the planning problem to be more reasonable. That is, in the same domain of the planning problem, multiple problems can share a domain description, which saves storage space of the file.

The PDDL description of AI planning includes the object, predicates, initial state, goal state, and actions/operators. Considering the conversion method of OWL-S to PDDL in [20], we can use the mapping rules in TABLE 4 to perform the conversion.

Since there is no concept in PDDL that specifically refers to parameter input/output, we use the state to indicate whether the parameter is available (available Para, where available is a predicate, and Para is the corresponding input/output parameters). If the state available Para is true, it indicates that the parameter Para exists and can be directly used; if the state available Para is false, it indicates that the parameter Para cannot be used directly. The fire allocation capability construction problem in PDDL is described in FIGURE 5.

V. ENHANCED-DOMAIN AND ORDERED-HILL-CLIMBING-BASED PLANNER

Compared with the classic AI planning problem, the number of operations in the capability construct domain is much larger. Since the heuristic search planner needs to calculate the heuristic value for the candidate action in every iteration, the search efficiency of the planner is greatly decreased when the number of actions increase. Therefore, this study draws on the practice in [21] and [22] to analyze the existing planning solutions, extract the sequence of operations that often appear, and combine them into macro-operations to reduce planning time. The planner structure diagram proposed in this study is shown in FIGURE 6.

As can be seen from FIGURE 6, the planner can be divided into three parts: macro-operation learning and updating, enhanced domain generation, and planning problem solving. These three parts will be described in detail below.

A. MACRO-OPERATION LEARNING AND UPDATING

1) MACRO-OPERATION LEARNING

The purpose of macro-operation learning is to extract macro-operations from existing planning solutions to speed up the planning. This study uses the N-gram analysis method to learn macro-operations from existing solution sets. N-gram analysis is widely used for statistical natural language processing tasks such as replication detection [27] and machine translation [28]. For a piece of text, an N-gram is a set that

```
(define (domain FireAllocation)
(:requirements :typing)
(:types TargetInformation PowerInformation
RequiredPowerInformation MissileKind ChargeNumber
DamageEffect StrikeRelationship SafeLimit StrikeTime
StrikeDistance - sInfo
    StrikePlan - cInfo
    sInfo cInfo - Info
    GroundToGround PlanApproved PlanDistributed - State )

(:predicates (avaliable ?x - Info)
        (on ?x - State))

(:action EvalPower
:parameters (?x - TargetInformation ?y - PowerInformation ?z -
RequiredPowerInformation ?w - GroundToGround)
:precondition (and (avaliable ?x) (avaliable ?y) (on ?w))
:effect (avaliable ?z))

(:action EvalShell
:parameters (?x - TargetInformation ?y -
RequiredPowerInformation ?z - MissileKind ?w -
GroundToGround)
:precondition (and (avaliable ?x) (avaliable ?y) (on ?w))
:effect (avaliable ?z))

                .
                .
                .
)
```

```
(define (problem P1) (:domain FireAllocation)
(:objects
    Target_Information - TargetInformation
    Power_Information - PowerInformation
    Required_Power_Information - RequiredPowerInformation
    Missile_Kind - MissileKind
    Charge_Number - ChargeNumber
    Damage_Effect - DamageEffect
    Strike_Relationship - StrikeRelationship
    Safe_Limit - SafeLimit
    Strike_Time - StrikeTime
    Strike_Distance - StrikeDistance
            Strike_Plan - StrikePlan
            Ground_To_Ground - GroundToGround
    Plan_Approved - PlanApproved
    Plan_Distributed - PlanDistributed
            )
(:init
            (avaliable Target_Information)
            (avaliable Power_Information)
            (on Ground_To_Ground)
)

(:goal (
        and
            (avaliable Strike_Plan)
            (on Plan_Distributed)
    )
)
)
```

FIGURE 5. Fire allocation capability construction problem in PDDL.

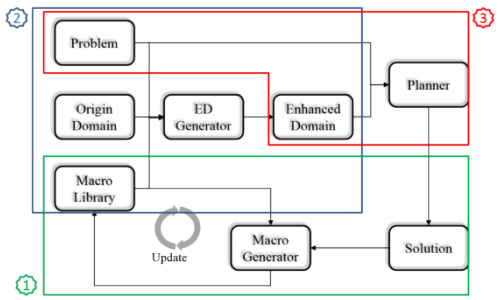


FIGURE 6. Planner structure diagram.

contains all combinations of consecutive N elements in the text. For example:

text: I like it !

2-gram (character): {"I_", "_l", "li", "ik", "ke", "e_", "_i", "it","t_" ,"_!"}

2-gram (word): {"I like", "like it", "it !"}

3-gram (character): {"I_l", "_li", "lik", "ike", "ke_", "e_i", "_it", "it_","t_!"}

3-gram (word): {"I like it", "like it !"} Similarly, for a planning solution, its N-gram can be expressed as follows (a simple example):

Solution:{EvalPower EvalShell EvalCharge CalcBestRn TaskAssign}

2-gram: {"EvalPower EvalShell", "EvalShell EvalCharge", "EvalCharge CalcBestRn", "CalcBestRn TaskAssign"}

3-gram: {"EvalPower EvalShell EvalCharge", "EvalShell EvalCharge CalcBestRn", "EvalCharge CalcBestRn TaskAssign"}

The N-gram extraction algorithm is given in Algorithm 1.

According to **Algorithm 1**, an N-gram set can be extracted from a solution. Assume the planning solution set is

Algorithm 1 N-Gram Extraction Algorithm

Input solution π

1: $G^n \leftarrow \phi$

2: For each i in $[1,\ |\pi| - n + 1]$ do

3: $\quad G^n \leftarrow G^n \cup \pi_{i,i+n-1}$

4: end

Output N-gram set G^n

$Solution = \{\pi_1, \pi_2, \ldots, \pi_{n_s}\}$ (where $n_s = |Solution|$), the N-gram set $NG^n = \{G_1^n, G_2^n, \ldots, G_{n_s}^n\}$ for the solution set can be achieved by calling **Algorithm 1** for every solution in *Solution*.

Unfortunately, it is not wise to treat all combinations of actions in the N-gram set as macro-operations. In the planning process, the macro-operation can play the role of accelerating the transition of the state, but if the number of macro-operations is too large, the number of candidate operations is invisibly increased, thereby reducing the planning efficiency. Therefore, it is necessary to analyze the action combinations in the n-gram set, select the action combinations that can assist in the planning process, and synthesize them into the planning domain.

For a combination of actions in an N-gram collection, the frequency of occurrence can be defined as

$$f_{ac^n} = \frac{card(\{G_j^n | ac^n \in G_j^n, j = 1, 2, \ldots, n_s\})}{n_s} \tag{6}$$

Certainly, a higher frequency of occurrence means that the action combination is more likely to advance the planning process. At the same time, it is also necessary to consider the effect of length n on the utility of action combination ac^n. A longer combination of actions is more likely to contain

some important steps in the planning process. Once accepted by the planner and added to the planning solution, it will greatly advance the planning process. For two combinations of actions that have the same frequency but different lengths, the combination with the longer length should have higher utility values. However, if the length of the combination is too long, the versatility of the action combination will be greatly reduced, which will bring additional computing burden. Taking the above situation into account, the utility value of the action combination is defined as follows:

$$U(ac^n) = f_{ac^n} + \alpha \frac{n}{n_{\max}} \tag{7}$$

where $n_{\max}$ represents the maximum length of the action combination and $\alpha \in [0, 1]$ is the weighting factor of length n. The set of action combination utility values can be defined as:

$$NGUS = \{NGU^i | i = 2, 3, \ldots, n_{\max}\} \tag{8}$$

$$NGU^i = \{(ac^i, U(ac^i_k)) | ac^i \in G^i_1 \cup G^i_2 \cup \ldots \cup G^i_{n_s}\} \tag{9}$$

After the calculation of the above utility values is performed, all combinations in the macro-operation candidate set are sorted according to utility value, and the action combinations with the top $g\%$ of utility values are selected to generate a macro-operation library by using **Algorithm 2**.

Algorithm 2 Macro Operation Generation Algorithm

Inputaction combination $ac^n = \langle a_1, a_2, \ldots, a_n \rangle$
1: $O_i.pre \leftarrow \emptyset O_i.addeff \leftarrow \emptyset O_i.deleff \leftarrow \emptyset$
2: For all a_i in ac^n do
3: For each precondition $P \in a_i.pre$
4: If $P \notin O_i.pre \wedge P \notin O_i.addeff$
5: $O_i.pre \leftarrow O_i.pre \cup \{P\}$
6: End
7: End
8: For each delete effect $D \in a_i.deleff$
9: If $D \in O_i.addeff$
10: $O_i.addeff \leftarrow O_i.addeff \setminus \{D\}$
11: Else
12: $O_i.deleff \leftarrow O_i.deleff \cup \{D\}$
13: End
14: End
15: For each add effect $A \in a_i.addeff$
16: If $A \in O_i.deleff$
17: $O_i.deleff \leftarrow O_i.deleff \setminus \{A\}$
18: End
19: $O_i.addeff \leftarrow O_i.addeff \cup \{A\}$
20: End
21: End
22: For each $LP \in O_i.pre \cup O_i.addeff \cup O_i.deleff$
23: Replace LP by the corresponding generalization parameter
24: End
25: Return O_i
Output Macro operation O_i

The aim of the **Algorithm 2** is to combine the actions into macro operation. In the algorithm, the *effect* consists of *addeff* and *deleff*. Literally, the *addeff*(*deleff*) means that it will be added to (deleted form) the state set after calling the corresponding service. At the same time, **Algorithm 2** can be divided into two parts, one is to merge the actions (line 2-21), and the other is to complete the generalization (line 22-24). In the merge part, the *precondition* and *effect* of macro operation O_i are adjusted when merging the actions. and in the generalization part, all the instance parameters will be generalized.

2) MACRO OPERATION LIBRARY UPDATE

The generation of macro-operations is based on the analysis of existing planning solutions. If a representative planning solution set cannot be collected before the macro-operation library is generated, the macro-operations are not strong enough to play a role in the subsequent planning process. Therefore, it is necessary to continuously analyze the newly obtained planning solution during the planning process to generate higher-quality macro-operations. To avoid repeated N-gram analysis of existing planning solutions, an incremental macro-operation library updating method is proposed here. The entire update process can be completed by performing an N-gram analysis on the newly added planning solution.

Assume that the set of action combination utility values, which were updated last time, is $NGUS_{last} = \{NGU^i_{Last} | i = 2, 3, \ldots, n_{\max}\}$. The new solution set is $Solution' = \{\pi'_1, \pi'_2, \ldots, \pi'_{n_s}\}$ (where $n_s = |Solution|$) and the corresponding set of operational combination utility values is $NGUS_{New} = \{NGU^i_{new} | i = 2, 3, \ldots, n_{\max}\}$.

Algorithm 3 Helps to merge the two sets.

In **Algorithm 3,** line 1-6 are used to initialize the merged set $NGUS_{Merge}$. the addition of new set lead to the increase of total solution number, as a result, the utility value of the action combination should be recalculated by line 4,

$$U(Moc^i_k) \leftarrow \left(U(Moc^i_k) - \alpha \frac{n}{n_{\max}}\right) \frac{n_s}{n'_s + n_s} + \alpha \frac{n}{n_{\max}}$$

where, Moc^i_k is an action combination in NGU^i_{Merge}, n_s represent the solution number in original solution set and n'_s represent the solution number in new solution set. It should be note that, every action combination in $NGUS_{merge}$ will be updated in this step. Line7-16 are the main process of merging. For each action combination Noc^i_k in NGU^i_{New}, if it also belongs to $ngu^i_{merge} \in ngus_{merge}$, its utility value should be updated again by line 10,

$$U(Moc^i_k) \leftarrow \frac{\left(U(Loc^i_k) - \alpha \frac{n}{n_{\max}}\right) n_s + \left(U(Noc^i_k) - \alpha \frac{n}{n_{\max}}\right) n'_s}{n_s + n'_s} + \alpha \frac{n}{n_{\max}}$$

where, $U(loc^i_k)$ represents the corresponding utility values in the $NGUS_{Last}$. Otherwise, its utility value should be

Algorithm 3 Utility Value Set Merging Algorithm

Input $NGUS_{Last} = \{NGU^i_{Last} | i = 2, 3, \ldots, n_{\max}\}$
$NGUS_{New} = \{NGU^i_{New} | i = 2, 3, \ldots, n_{\max}\}$
1: $NGUS_{Merge} \leftarrow NGUS_{Last}$
2: For each $NGU^i_{Merge} \in NGUS_{Merge}$
3: For each $(Moc^i_k, U(Moc^i_k)) \in NGU^i_{Merge}$
4: $U(Moc^i_k) \leftarrow \left(U(Moc^i_k) - \alpha \frac{n}{n_{\max}}\right) \frac{n_s}{n'_s + n_s} + \alpha \frac{n}{n_{\max}}$
5: End
6: End
7: For each $NGU^i_{New} \in NGUS_{New}$
8: For each $(Noc^i_k, U(Noc^i_k)) \in NGU^i_{New}$
9: If $\forall (Moc^i_k, U(Moc^i_k)) \in NGU^i_{Merge}, Noc^i_k = Moc^i_k$
10: $U(Moc^i_k) \leftarrow \frac{\left(U(Loc^i_k) - \alpha \frac{n}{n_{\max}}\right) n_s + \left(U(Noc^i_k) - \alpha \frac{n}{n_{\max}}\right) n'_s}{n_s + n'_s} + \alpha \frac{n}{n_{\max}}$
11: Else
12: $U(Noc^i_k) \leftarrow \left(U(Noc^i_k) - \alpha \frac{n}{n_{\max}}\right) \frac{n'_s}{n'_s + n_s} + \alpha \frac{n}{n_{\max}}$
13: $NGU^i_{Merge} \leftarrow NGU^i_{Merge} \cup \{(Noc^i, U(Noc^i_k))\}$
14: End
15: End
16: End
17: Return $NGUS_{Merge}$
Output $NGUS_{Merge} = \{NGU^i_{Merge} | i = 2, 3, \ldots, n_{\max}\}$

recalculated by line 12, and $(Noc^i, U(Noc^i_k))$ will be added to the NGU^i_{Merge} by line 13.

After obtaining the updated set of utility values, the action combinations with the top $g\%$ of utility values are then selected to generate the macro-operation library by using **Algorithm 2**.

B. ENHANCED DOMAIN GENERATION

The size of the generated macro-operation library should be large enough; otherwise, it may not be able to contain more effective macro-operations, resulting in less performance improvement of the planner. However, if all the macro-operations in the library are added to the planning domain for participation in the planning process, the search space will be too large to decrease the planning efficiency. Therefore, the macro-operations that are most likely to be used should be selected from the macro-operation library for the characteristics of each problem. Finally, the enhanced fields can be formed from the atomic operations and the selected operations to maximize planning efficiency.

Choosing a suitable set of macro-operations for different planning problems is essentially looking for a mapping relationship between problem conditions, effects, and macro-operations. Drawing on the idea of heuristic filtering in [18], this study uses the relaxation planning method to pre-plan the problem in the original planning domain and entire macro-operation library, and the enhanced domain is generated by the atomic operations in the original domain and the macro-operations that are used in the corresponding relaxation planning solution.

C. PLANNING PROBLEM SOLVING

1) HEURISTIC DESIGNING

Solving the planning problem is essentially the process of state search. The main idea is to calculate and compare the heuristic value of each state, and select the better state in the search process to get the final planning solution. In the path planning problem, the A* algorithm is a very classic heuristic algorithm. It determines the current action strategy by calculating the heuristic value of the location at the next moment in each search process. Its heuristic is defined as:

$$F(ST) = H(ST) + G(ST) \tag{10}$$

where $F(ST)$ is the heuristic of state ST, $H(ST)$ represents the cost of the state transition from the initial state to the current state, and $G(ST)$ represents the cost of the state transition from the current state to the goal state. In path planning, the state is the location of the target, and the initial and goal locations of the target are also known, so calculating $H(ST)$ and $G(ST)$ only requires a function that calculates the distance between the two locations. Due to the excellent performance of the A* algorithm, the heuristic has a wide range of applications in many path-planning scenarios.

Unlike the path-planning problem, the calculation of the state heuristic values in the AI planning problem is not as intuitive. Therefore, how to construct a reasonable heuristic to calculate the heuristic value of the current state is a key issue.

In response to this problem, many existing search methods ignore the delete effect of actions to relax the original planning problem and then estimate the distance from the current state to the goal state, e.g., [7], [10], [11], and [17]. The following is a brief introduction to the heuristic of the relaxation plan.

Given an AI planning problem $AIP = (P, A, I, G)$, where P is a finite set of logical propositions, A is a finite set of all actions, and I and G indicate the initial state and goal state of the problem, respectively, the relaxed planning problem can be expressed as $AIP' = (P, A', I, G)$, where A' is a set of actions in A, in which the delete effect is ignored. The relaxation planning graph of AIP' is a directed hierarchical graph with two types of nodes and three types of edges. The layers of the planning graph alternate between the propositional layer and the action layer, recorded as $PL_0, ACT_0, \cdots, PL_i, ACT_i, \cdots, ACT_{t-1}, PL_t$, and it is subject to:

$$PL_i = \begin{cases} I & i = 0 \\ PL_{i-1} \cup \{add(a) | \text{pre}(a) \subseteq PL_{i-1}\} & i > 0 \end{cases} \tag{11}$$

$$ACT_i = \{a \in A' | pre(a) \subseteq PL_i,\ i \geq 0\} \tag{12}$$

The expansion can be stopped when one of the following two conditions is met by the propositional layer PL_i: a) the fixed point is reached, $PL_i = PL_{i-1}$. b) the goal state is reached, $G \subseteq PL_i$. If the propositional layer of the graph can finally reach the goal state, then two sets can be obtained through the solution extraction process: a) the sub-goal set

extracted in each proposition layer $SG_i(1 \leq i \leq t)$. b) the action set extracted in each action layer that satisfies the relevant sub-goals $SA_i(0 \leq i \leq t-1)$. Finally, the relaxation planning solution $\langle SA_0, SA_1, \ldots, SA_{t-1} \rangle$ can be achieved. For a given state ST, the relaxation planning graph heuristic value can be defined as:

$$h(ST) = \sum_{i=0}^{t-1} |SA_i| \tag{13}$$

However, when the planning domain contains the macro-operations, using (13) to calculate the heuristic value of the state may result in the degradation of the solution quality. Since the macro-operation is obtained by combining multiple atomic operations, if its heuristic value is treated the same as an atomic operation, some atomic operations that are more favorable to the current state will lose their advantage. The planner will preferentially select the macro-operation for the current state, and overuse of macro-operations may increase the length of the planning solution. Therefore, it is necessary to adapt the calculation of the heuristic value after adding the macro-operations to avoid the increase in the length of the solution.

Based on the above considerations, the state heuristic proposed in this study are defined as follows:

$$h(s) = \sum_{i=0}^{t-1} \sum_{j=1}^{|SA_i|} w(a_{ij}),\ a_{ij} \in SA_i \tag{14}$$

$$w(a) = \begin{cases} \delta a.length & a \text{ is macros} \\ 1 & others, \end{cases} \tag{15}$$

where $a.length$ is the length of macro-operation a and $\delta \in (0.5, 1)$ represents the macro-operation weight adjustment factor, which affects the planner's selection preference for macro-operations. The smaller the value is, the more the planner prefers to select the macro-operation into the planning solution.

2) STATE SEARCH ORDER

Enforced hill climbing (EHC) is a "radical" search strategy, which simplifies the search termination conditions of the current layer, based on the width-first search, and introduces the child nodes that do not satisfy the condition into the search queue to deepen the depth of the search (FIGURE 7). Due to the "greedy" feature of EHC, it is very useful in an AI planner for quickly finding a feasible solution from the initial state to the goal state.

When macro-operations are not introduced, the actions which are instantiated by atomic operations have the same priority. Therefore, EHC generally searches the candidate state randomly. However, after macro-operations are introduced, if the search is still performed randomly without considering the priority of the actions, the greedy characteristic of EHC may lead to the ignorance of the macro-action of a better candidate and the selection of an atomic action when it has a certain promotion effect, causing it to be selected

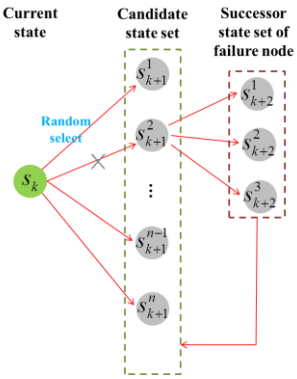


FIGURE 7. Enforced hill climbing.

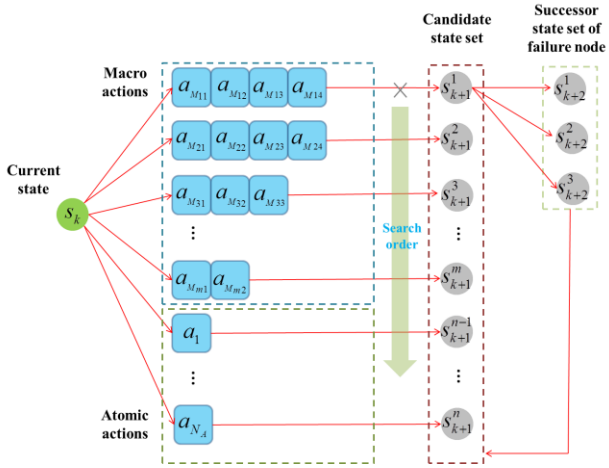


FIGURE 8. Ordered hill climbing.

before macro-actions. The occurrence of this situation will greatly affect the promotion of macrooperations in the planning process. It is only when macroactions are selected into the planning solution during the planning process can the entire planning process be accelerated. If no macro-action is used in a planning process, then a series of "extra" steps such as instantiating macrooperations and searching macro-actions will still be carried out, which are bound to extend the search time. In view of the above problems, the candidate state is sorted according to the length of the corresponding action, and the state is searched for in order (FIGURE 8).

VI. SIMULATION

In the simulation, the above methods are integrated into an enhanced-domain and ordered-hill-climbing-based planner (EDOHCP). To illustrate the superiority of the EDOHCP, the experiment compares EDOHCP with the other three planners:

a) Classic heuristic planner (CHP), which is based on FF and does not contain the macro-operations.

b) Macro-operation-based classic heuristic planner (MCHP), which adds the macro-operations into domain without the heuristic filtering.

c) Enhanced-domain-based classic heuristic planner (EDCHP), which performs the classic heuristic search in the enhanced domain.

For a few reasons, the real data in C4ISR system cannot be shown in this paper. To simulate the C4ISR system capability construction process and match the characteristics of its large number of operations, the travel and the education domains in OWLS-TC version 4.0 [29] are used in the simulation. A total of 1020 service requests are generated randomly in each domain, in which the first 1000 service requests are planned by the CHP, and the corresponding solutions are used in the macro-operation learning, and the last 20 service requests are used as test samples. The OWL-S2PDDL Converter in [20] is introduced to convert the services and requests to the PDDL. Here, we run the experiments in Eclipse (Version: 3.4.2) and the PC with Intel Core i3-4150 (3.50 GHz) and 4 GB memory.

The first set of experiments compared the planning time and number of searches of the four planners in the two domains. In the EDOHCP, we set the $n_{\max} = 6$, $\alpha = 0.9$, $\delta = 0.9$, $g = 30$, and the maximum number of macro-operations in the library $M_{\max} = 30$. For an impartial comparison between these planners, the average search time and number of searches of a Monte-Carlo simulation (with 40 runs) is provided in FIGURE 9- FIGURE 12.

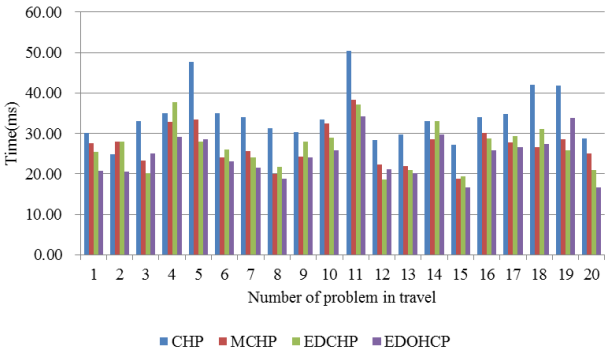


FIGURE 9. Search time in travel domain.

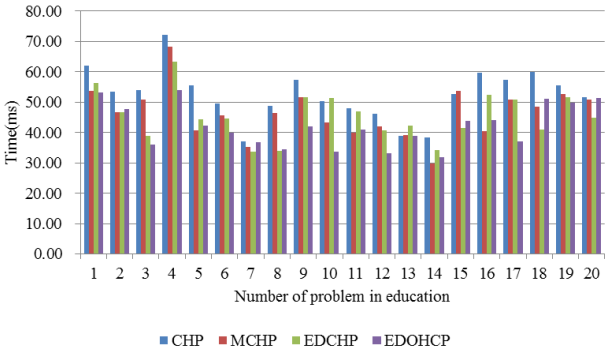


FIGURE 10. Search time in education domain.

The search time of each planner in the travel domain and education domain is shown in FIGURE 9 and FIGURE 10. In general, the CHP takes the longest time. The MCHP is significantly less time-consuming than the CHP, and the

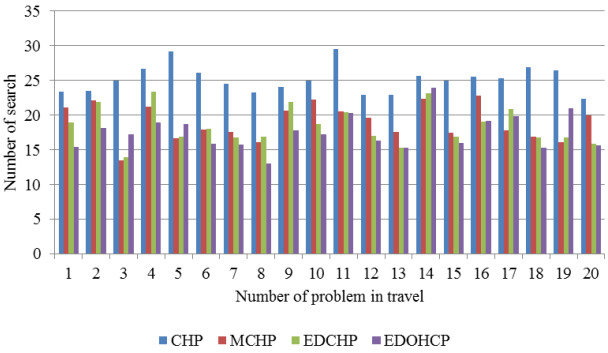


FIGURE 11. Number of searches in travel domain.

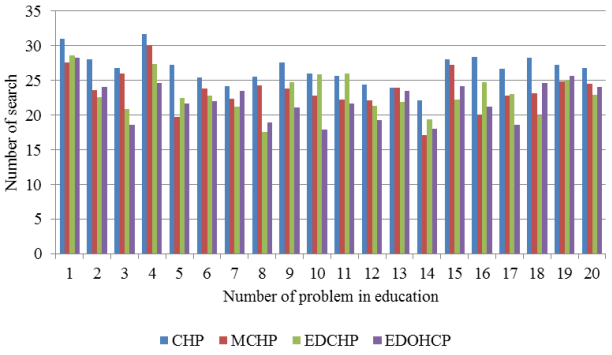


FIGURE 12. Number of searches in education domain.

EDCHP takes slightly less time than the MCHP. Although the number of macro-operations in the enhanced domain is less than the number in the entire macro-operation library, and the search branch of EDCHP is relatively less than the MCHP in the subsequent search process, there will still be a certain amount of time overhead when generating the enhanced domain. If the shortened time in the search process cannot compensate for the generation time of the enhanced domain, the time overhead of EDCHP will exceed that of the MCHP (see groups 4, 6, 8, and 9 in FIGURE 9 and groups 1, 5, 10, and 11 in FIGURE 10). The EDOHCP proposed in this study uses the OHC method to search the state, which speeds up the progress of the planner in the search phase. Therefore, in most cases, the planning solution can be obtained in the shortest time.

FIGURE 11 and FIGURE 12 show the search times of the four planners in the travel domain and the education domain. It can be seen that the results of the search times have the same regularity as the results of the planning time in FIGURE 9 and FIGURE 10.

TABLE 5 shows the average of the total planning time and total number of searches for the 40 Monte Carlo simulations of the 20 test samples in the travel and education domains. It can be seen that, compared with the CHP, MCHP, and EDCHP, the planning time of EDOHCP proposed in this study is shortened by 28.52%, 9.18%, and 8.20%, and the number of searches are reduced by 30.28%, 7.74%, and 5.11% in the travel domain. At the same time, the planning time was shortened by 19.73%, 9.50%, and 7.52%, and the

TABLE 5. Total search time and numbers of each planner.

	CHP	MCHP	EDCHP	EDOHCP
travel (time)	685.68	539.75	533.93	490.15
education (time)	1048.85	930.30	910.38	841.90
travel (iterations)	503.175	380.25	369.7	350.825
education (iterations)	535.18	472.30	460.75	441.45

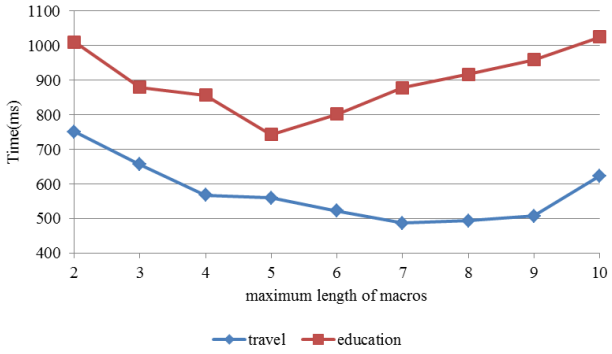


FIGURE 13. The total search time of EDOHCP with the different maximum length of macros.

number of searches are reduced by 17.51%, 6.53%, and 4.19% in the education domain.

The second set of experiments compared the total planning time of EDOHCP with the different maximum length of macro-operations, and the other parameters remained unchanged from the first set of experiments. FIGURE 13 shows that as the maximum length of macro-operations increases, the planning time continues to decrease and then increase. In the travel domain, the EDOHCP has the shortest planning time when $n_{\max} = 5$, and in the education domain, the EDOHCP has the shortest planning time when $n_{\max} = 7$. In essence, the above phenomenon is caused by the utilization of macro-operations. The reason why the macro-based planner can reduce the planning time is mainly because the macro-operation accelerates the search progress and reduces the number of searches in the planning process. Clearly, the longer the length of the macro-operations is, the more favorable the advancement of the search process is, but at the same time, the probability of its usage is inevitably reduced. If the macro-operation is not successfully utilized during the planning process, then extra processing steps will extend the planning time.

The third set of experiments compared the total planned time of EDOHCP in the two domains under different learning times, while maintaining constant the other parameters from the first set of experiments. FIGURE 14 shows that as the learning time increases, the planning time of EDOHCP also generally decreases. It is worth noting that the decrease here is probabilistic. That is to say, the relationship between the learning times and the performance is not absolute. Statistically speaking, the greater the learning time is, the more

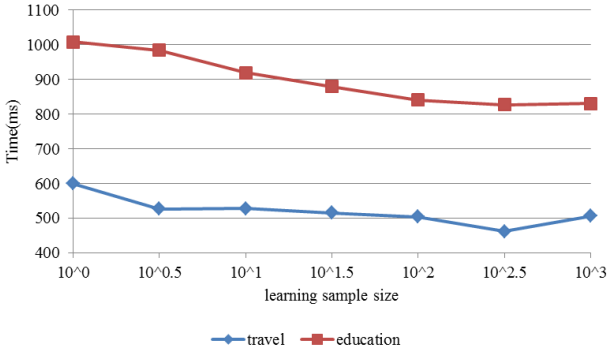


FIGURE 14. The total search time of EDOHCP with the different learning sample size.

likely that macro-operations can be used in future planning problems (In the travel domain, the planning time is lowest when the learning sample size is10^2.5 instead of 10^3).

VII. CONCLUSION

To solve the capability construction problem in a C4ISR system and quickly map the system capability to the service, this study models the system capability and service first, then describes the system capability construction problem as an AI planning problem and designs the macro-operations learning algorithm and heuristic search algorithm from the characteristics of the problem. From the simulation experiment results, the proposed planner can effectively shorten the solution time and find a feasible solution that satisfies the condition in fewer search times. Meanwhile, some parameters in the planner, such as the maximum length of the macro-operation and the weight factor will have a significant impact on the planning results. In essence, the impact of the above parameters on the planner is reflected in the calculation of the action combination utility value, so the next step can focus on how to calculate the utility value of the action combination according to the characteristics of the domain.

ACKNOWLEDGMENT

The authors would like to thank Dr. Tengyao Li and Dr. Tute Shang in the AFEU CPS Security Lab, for providing great technical support for this study.

REFERENCES

[1] L. Cao *et al.*, *C4ISR System*, 2nd ed. Beijing, China: National Defence of Industry Press, 2016, pp. 8–9.

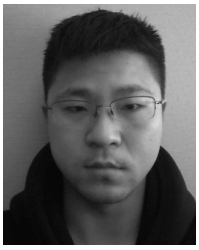
[2] T. Zhu, W. T. Liang, and S. H. Huang, "Task-oriented architectural modeling method for network information system of systems," *Command Inf. Syst. Technol.*, vol. 9, no. 4, pp. 17–22, Aug. 2018.

[3] S. Satyal, I. Weber, L. Bass, and M. Fu, "Rollback mechanisms for cloud management APIs using AI planning," *IEEE Trans. Dependable Secure Comput.*, to be published.

[4] J. Liu, F. Yang, and J. Li, *Research of AI Planning for Space Flight Control Based on PDDL* (Lecture Notes in Electrical Engineering), vol. 323. Berlin, Germany: Springer, 2015, pp. 359–369.

[5] L. Machado, R. Prikladnicki, F. Meneguzzi, C. R. de Souza, and E. Carmel, "Task allocation for crowdsourcing using AI planning," in *Proc. IEEE/ACM Int. Workshop Crowdsourcing Softw. Eng.*, May 2016, pp. 36–40.

[6] E. Kaldeli, A. Lazovik, and M. Aiello, "Domain-independent planning for services in uncertain and dynamic environments," *Artif. Intell.*, vol. 236, no. 3, pp. 30–64, Jul. 2016.
[7] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *J. Artif. Intell. Res.*, vol. 14, no. 1, pp. 253–302, May 2001.
[8] D. Nau *et al.*, "SHOP2: An HTN planning system," *J. Artif. Intell. Res.*, vol. 20, no. 1, pp. 379–404, Dec. 2003.
[9] J. G. Sun, M. H. Yin, and S. Lü, "JLU-RLAO and JLU-QLAO: Two non-deterministic planners," *J. Comput. Res. Develop.*, vol. 46, no. 4, pp. 667–675, Apr. 2009.
[10] J. Hoffmann, "The metric-FF planning system: Translating 'ignoring delete lists' to numeric state variables," *AI Access Found.*, vol. 20, no. 2003, pp. 291–341, Dec. 2003.
[11] J. Hoffmann and R. I. Brafman, "Conformant planning via heuristic forward search: A new approach," *Artif. Intell.*, vol. 170, pp. 507–541, May 2006.
[12] D. P. Cai *et al.*, "Fast forward planning system based on delayed partly reasoning," *Chin. J. Comput.*, vol. 31, no. 5, pp. 793–802, May 2008.
[13] L. Zhang, C. J. Wang, and J. Y. Xie, "Cost optimal planning with LP-based multi-valued landmark heuristic," in *Proc. Int. Conf. Auton. Agents Multi-Agent Syst., Int. Found. Auton. Agents Multiagent Syst.*, Jan. 2014, pp. 509–516.
[14] Y. Y. Xu and W. Y. Yue, "BDD-Based incremental heuristic search," *J. Softw.*, vol. 20, no. 9, pp. 2352–2365, Sep. 2009.
[15] R. S. Liang, Y. F. Jiang, and H. Z. Yang, "Forward heuristic search planning based on ordered hill climbing algorithm," *J. Univ. Electron. Sci. Technol. China*, vol. 42, no. 3, pp. 464–468, Mar. 2013.
[16] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Comput. Intell., Amer. Assoc. Artif. Intell.*, vol. 2, nos. 3–4, pp. 189–208, Sep. 1995.
[17] A. Botea, M. Enzenberger, M. Mueller, and J. Schaeffer, "Macro-FF: Improving AI planning with automatically learned macro-operators," *J. Artif. Intell. Res.*, vol. 24, no. 1, pp. 581–621, Jan. 2005.
[18] A. Dulac, D. Pellier, H. Fiorino, and D. Janiszek, "Learning useful macro-actions for planning with N-Grams," in *Proc. IEEE 25th Int. Conf. Tools Artif. Intell.*, Nov. 2013, pp. 803–810.
[19] Z. Jiang, J. Wen, J. Zeng, Y. Zhang, X. Wang, and S. Hirokawa, "Dynamic macro-based heuristic planning through action relationship analysis," *IEICE Trans. Inf. Syst.*, vol. E98-D, no. 2, pp. 363–371, Feb. 2015.
[20] H.-S. Kim and I.-C. Kim, "Mapping semantic Web service descriptions to planning domain knowledge," in *Proc. World Congr. Med. Phys. Biomed. Eng.* Berlin, Germany: Springer, 2007, pp. 388–391.
[21] L. Chrpa, M. Vallati, and T. L. Mccluskey, "On the online generation of effective macro-operators," in *Proc. 24th Int. Joint Conf. Artif. Intell. (IJCAI)*, Jul. 2015, pp. 1544–1550.
[22] S. Castellanos-Paez *et al.*, "Mining useful macro-actions in planning," in *Proc. IEEE Int. Conf. Artif. Intell. Pattern Recognit.*, Sep. 2016, pp. 1–6.
[23] C. Marinagi, T. Panayiotopoulos, and C. Spyropoulos, "PTTPS: A model for timeline-based planning and scheduling of patient tests in hospitals," *Global J. Technol.*, vol. 3, no. 2013, pp. 1766–1771, Feb. 2013.
[24] D. Perez *et al.*, "Solving the physical traveling salesman problem: Tree search and macro actions," *IIEEE Trans. Comput. Intell. AI Games*, vol. 6, no. 1, pp. 31–45, Mar. 2014.
[25] G. B. Zou, Y. Gan, Y. Chen, and B. Zhang, "Dynamic composition of Web services using efficient planners in large-scale service repository," *Knowl.-Based Syst.*, vol. 62, pp. 98–112, Mar. 2014.
[26] X. Z. Zhou, *Aided Decision-Making Techniques in Command and Control System*. Beijing, China: National Defense Industry, 2012.
[27] M. A. Khan, A. Aleem, A. Wahab, and M. N. Khan, "Copy detection in urdu language documents using N-Grams model," in *Proc. IEEE Int. Conf. Comput. Netw. Inf. Technol.*, Jul. 2011, pp. 263–266.
[28] R. Wang, M. Utiyama, I. Goto, E. Sumita, H. Zhao, and B.-L. Lu, "Converting continuous-space language models into N-Gram language models with efficient bilingual pruning for statistical machine translation," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 15, no. 3, pp. 1–26, Mar. 2016.
[29] *OWL-S Service Retrieval Test Collection*. Accessed: Jul. 9, 2018. [Online]. Available: http://www.semwebcentral.org/projects/owls-tc/



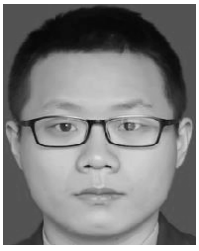
ZHIQIANG JIAO was born in 1992. He received the B.S. degree in information and communication engineering and the M.S. degree in information fusion from Air Force Engineering University, in 2014 and 2017, respectively, where he is currently pursuing the Ph.D. degree.

From 2014 to 2016, he has focused on the maneuvering target tracking and multi-target tracking technology. Since 2017, he has been starting researching on the C4ISR system and C2theory. His research interests include information fusion, command information systems, and mission planning.

PEIYANG YAO was born in 1960. He received the B.S. and M. S. degrees from Xidian University in 1991 and 1982, respectively. He is currently a Professor with the Information and Navigation College, Air Force Engineering University. His research interests include command and control theory, and command automation systems.

JIEYONG ZHANG was born in 1983. He received the B.S., M.S., and Ph.D. degrees from Air Force Engineering University, in 2006, 2008, and 2012, respectively, where he is currently a Lecturer with the Information and Navigation College. His research interests include mission planning technique and military organizational analysis.

LUJUN WAN was born in 1986. He received the B.S., M.S., and the Ph.D. degrees from Air Force Engineering University, in 2007, 2009, and 2013, respectively, where he is currently a Lecturer with the ATC Navigation College. His research interests include mission planning technique and military organizational analysis.



XUN WANG was born in 1990. He received the B.S. degree in communication engineering from Shandong University, in 2013, and the M.S. degree in command information systems from Air Force Engineering University, in 2016, where he is currently pursuing the Ph.D. degree. His research interests include command information systems and mission planning.