Journal of Artificial Intelligence Research 82 (2025) 2125-2166 Submitted 08/2024; published 04/2025

Solving Overlapping Coalition Structure Generation in Task-Based Settings

Guofu Zhang ZGF@HFUT.EDU.CN
Zhaopin Su SZP@HFUT.EDU.CN
(Corresponding Author)
Xiaoxiao Song 2019170919@MAIL.HFUT.EDU.CN
Zixuan Gao 2020171119@MAIL.HFUT.EDU.CN
School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, Anhui, China
Intelligent Interconnected Systems Laboratory of Anhui Province, Hefei, Anhui, China
Anhui Province Key Laboratory of Industry Safety and Emergency Technology, Hefei, Anhui, China

Miqing Li M.LI.8@BHAM.AC.UK
CERCIA, School of Computer Science, University of Birmingham, Birmingham, UK

Xin Yao XINYAO@LN.EDU.HK
School of Data Science, Lingnan University,Hong Kong SAR, China
CERCIA, School of Computer Science, University of Birmingham, Birmingham, UK

Abstract

The overlapping coalition structure generation problem (OCSGP) is a challenging computational problem in multi-agent systems. It focuses on selecting possibly overlapping coalitions from a set of agents to maximize the social welfare of all coalitions while containing all agents. However, in practical applications, coalitions may be formed to selectively respond to tasks from a pool of potential tasks assigned to agents. Consequently, this study considers OCSGP in a task-based setting, where each agent has finite resources and can only respond to tasks of interest, and each coalition can only take on mutually disjoint subsets of tasks. Specifically, we first present a model of the task-based OCSGP and investigate its computational complexity. Our theoretical results demonstrate that this specific OCSGP remains intractable even under restrictive assumptions. Subsequently, we develop a generic evolutionary algorithm framework (EAF) to find an approximately optimal overlapping coalition structure (OCS) in time quartic polynomial in the size of the instance. Particularly, we devise a specific solution-repair based heuristic of cubic time complexity to generate a feasible OCS. Finally, we compare the proposed EAF with a task-oriented heuristic and a hybrid algorithm for OCSGP, and examine its applicability in the pursuit-evasion problem. The experimental results reveal that the proposed EAF exhibits superior performance in finding feasible OCSs and demonstrates flexible adaptability to problem size and resource status.

1. Introduction

Coalition formation is the most basic and crucial cooperation form of in multi-agent systems (MAS) (Sandholm & Lesser, 1997; Shehory & Kraus, 1998). Forming a coalition can make incompetent agents capable enough to accomplish tasks by sharing their resources and working together. Usually, the coalition formation process consists of three main phases (Sandholm & Lesser, 1997):

- Coalition structure generation (CSG) (Greco & Guzzo, 2017; Rahwan et al., 2012, 2015; Service & Adams, 2011b). This phase aims to make each agent join a coalition and find an optimal set of coalitions (i.e., a coalition structure) to maximize social welfare.

- Coalition value calculation (Rahwan & Jennings, 2007; Riley et al., 2015). This phase involves calculating the utility value of each possible coalition and distributing this calculation among agents.

- Revenue distribution (Airiau & Sen, 2009; Anshelevich & Sekar, 2015; Deng & Papadimitriou, 1999; Ketchpel, 1994; Markakis & Saberi, 2005; Zick et al., 2012). This phase focuses on dividing the payoff of every formed coalition into a stable distribution, where no agent can object to its assigned payoff.

Obviously, CSG is the most important phase, the primary objectives of which are to determine the formation of coalitions and the allocation of tasks to each coalition. However, this process can be computationally challenging due to the exponential growth in the number of possible coalitions with the increase in the number of agents (Conitzer & Sandholm, 2006; Hoefer et al., 2018; Sandholm et al., 1999).

Over the past decade, a wide range of CSG model and algorithms have been developed in different situations, such as as characteristic function games (CFGs) (Greco & Guzzo, 2017; Rahwan et al., 2012, 2015; Service & Adams, 2011b), coalitional skill games (CSKGs) (Bachrach et al., 2010, 2013; Liu et al., 2016), coalitional resource games (CRGs) (Dunne et al., 2010; Su et al., 2020; Wooldridge & Dunne, 2006; Zhang et al., 2015), and overlapping coalition formation games (OCFGs) (Chalkiadakis et al., 2010; Mahdiraji et al., 2021; Zhang et al., 2020; Zick et al., 2019, 2014). More precisely, in CFGs, the coalition value is predefined by a characteristic function. CSKGs and CRGs associate the value of a coalition with its accomplishing tasks or achieving goals under the constraints of skills, resources, or capabilities. Additionally, they allow a coalition to simultaneously undertake multiple different tasks, implying that an agent can join diverse tasks in disguised forms. A harsh reality, however, is that both CSKGs and CRGs stipulate that an agent cannot but join only a coalition at any time, no matter how capable or resourceful it is. This property greatly reduces the enthusiasm of agents to respond to tasks, leads to extremely low resource utilization, and limits the cooperative flexibility of coalitions for task-solving.

In many practical application scenarios, decision makers always hope to maximize their benefits derived from a pool of potential or real-time tasks by using the existing resources of agents (Chen & Sun, 2012; Dang & Jennings, 2006). Consequently, it is possible for each agent to serve different tasks of interest (Su et al., 2020; Wooldridge & Dunne, 2006). For example, in computing power network (Yu et al., 2024; Di et al., 2023), if an agent is implemented on a sole available supercomputer, it should not be limited to joining only one coalition, as only it has enough processing power that can be shared across different coalitions with high demands. Additionally, in cooperative transportation (Guajardo et al., 2018), each truck prefers to accept the shipping orders nearby to reduce the transportation cost as much as possible. Similarly, in wireless sensor networks (Srinivasan & Murthy, 2020; Sun et al., 2016; Wang et al., 2014; Xiao et al., 2015), each sensor is inclined to monitor targets within its sensing region to maximize the network lifetime.

OCFGs are a formal model for cooperative games with overlapping coalition structures (OCSs), in which agents need to allocate their different types of resources to simultaneously serve different tasks as members of different coalitions (Zick et al., 2014, 2019). Clearly,

OCFGs can effectively model the above task-driven scenarios. Now, the primary concern in OCFGs is to find an optimal OCS to maximize the total coalition value. This is widely recognized as the OCS generation problem (OCSGP) and is also known for its high computational complexity (Zhang et al., 2020).

Against this background, in this work, we discuss OCSGP from the perspective of task-based settings and make the following contributions to the state-of-the-art in OCSGP.

- We propose a natural variation of the traditional OCSGP in a task-based setting. In the proposed model, each agent has finite resources and can only respond to a part of tasks, and each coalition can only take on mutually disjoint subsets of tasks.
- We analyze the size of the search space and the computational complexity of the proposed OCSGP. We show that checking whether a pending overlapping coalition structure (OCS) is feasible can be solved in polynomial time, while it is impractical to find the optimal OCS in time polynomial in the number of agents and tasks.
- We develop a generic evolutionary algorithm framework (EAF) to solve the proposed OCSGP efficiently. The proposed EAF can be combined with any binary evolutionary algorithms (EAs) (Poli & Langdon, 2006; Slowik & Kwasnicka, 2020; Yu et al., 2012). Particularly, we embed a specific solution-repair based heuristic of cubic time complexity to generate a feasible OCS. The embedded heuristic in EAF assigns a task only according to the residual resources of agents, which enables EAF to effectively prevent the potential resource conflicts over the rare, but highly demanded resources between the competitive overlapping coalitions. In this case, the proposed EAF can find an approximately optimal OCS in time quartic polynomial in the size of the instance.
- We compare EAF with a task-oriented heuristic (Zhang et al., 2020) and a hybrid algorithm (Zhan et al., 2012) for OCSGP, and examined its applicability in the pursuit-evasion problem (Lopez et al., 2020; Pan & Yuan, 2023). The experimental results demonstrate the high competitiveness of EAF.

The remainder of this work is structured as follows. First, in Section 2, an overview of the related work on OCFGs is provided. Section 3 presents the OCSGP in task-based settings. Sections 4 and 5 analyze the search space and the computational complexity of the proposed OCSGP, respectively. Section 6 proposes a generic EAF to solve OCSGP more efficiently, in which a heuristic is presented in detail to show how to generate a feasible OCS. In Section 7, the experimental methodology is introduced, and in Section 8 the results are reported. Finally, Section 9 concludes and recommends future work.

The remainder of the paper extensively employs various notations. For ease of reference, we have compiled a summary of the primary notations used and included it in Appendix A. Furthermore, to maintain a high level of readability in the main body of the text, we have opted to exclude detailed proofs. Instead, these comprehensive proofs can be found in Appendix B.

2. Related Work

Several state-of-the-art reviews of coalition structure generation and OCFGs has been carried out recently in (Mahdiraji et al., 2021; Rahwan et al., 2015). In this section, we only provide a general discussion regarding models and algorithms that we believe are closest

to task-related coalition structure generation. Additionally, similarities and differences between the related work will be examined. Moreover, the recent applications of OCFGs will be discussed.

2.1 Coalition Structure Models

The traditional coalition structure generation aims to find an optimal partition of the agent set to maximize the coalition structure value (Rahwan et al., 2015). Unfortunately, the value of each coalition is known in advance and is independent of the task. To address this problem, Deng and Papadimitriou (1999) considered a two-level hierarchy of agents to cooperate in solving a linear programming problem. Markakis and Saberi (2005) proposed a multicommodity flow game, where the payoff that a node receives depends on its capacity. Dang and Jennings (2006) proposed a model of coalition structure generation in task-based settings, in which alternative coalitions are responsible for mutually disjoint subsets of tasks and only one coalition can do nothing. Anshelevich and Sekar (2015) developed a transferable-utility coalition formation model and computed an approximately core stable solution. Präntare and Heintz (2020) considered the ordered coalition structure, in which each embedded coalition cannot but take on only one unrepeated task. Bachrach et al. (2013) presented a CSGs model for agents' coordination in uncertain environments. In CSGs, each agent has a set of skills that are hard to be quantified but can only be qualitatively expressed; a coalition can perform a task only if its members can cover the required skills of the task; and the coalition structure value is the maximum number of tasks that can be completed. Wooldridge and Dunne (2006) developed a CRGs model in terms of resource contention between goals. Differently, in CRGs, each agent has a personalized interest set; an agent will join a coalition only if it can find an interesting goal in the collective goal set; and a coalition structure is a set of cooperation structures, each of which contains the disjoint coalition, goal set, and contribution vector. Su et al. (2020) presented a natural extension of the traditional CRGs, in which each agent contributes its resources only to goals that are in its own interest set. Then, they proposed a flow-network-based exhaust algorithm and a heuristic hybrid algorithm to find the largest successful coalition, respectively. However, in the above models, the coalition structure generation problem stipulates that an agent cannot but join only a coalition at any time. That is, a coalition structure is a partition of the agent set. This harsh term will result in very low resource utilization and is extremely detrimental to the implementation of tasks, especially when only few agents have rare, but highly demanded resources.

To improve the flexibility of collaboration, Chalkiadakis et al. (2010) presented an OCFGs model on the basis of the previous work carried out by Shehory and Kraus (1998). In OCFGs, agents can simultaneously participate in completely different coalitions, as long as they have sufficient and required resources. In this case, an OCS is not a partition, but a cover of the agent set. Although the OCFGs model can maximize the resource utilization of agents, many computational questions surrounding it have become extremely complex and intractable. Specifically, Chalkiadakis et al. (2010) first explored the issue of core stability in OCFGs. Zick et al. (2011, 2012, 2014, 2019) presented an arbitration model of OCFGs and discussed the stability of the arbitration core. It has been revealed that the computational complexity of many decision problems in arbitration OCFGs is largely dependent on the resource amount owned by agents and the coalition size. Mamakos and Chalkiadakis (2017) presented an iterated OCFGs to deal with the situation that each agent has no knowledge of the amount of resources possessed by other agents, but knows well the potential resource investment of every other agent. However, they found that the optimal

response of an agent in the coalition-formation process is computationally intractable in polynomial time.

2.2 Algorithms for Coalition Structure Generation

Bachrach and Rosenchein (2010) proposed a fixed parameter tractable algorithm (FPTA) to search for the optimal coalition structure of CSKGs according to the tree decomposition in the hyper-graph. Liu et al. (2016) evaluated the usability of EAs, such as binary particle swarm optimization (BPSO) (Kennedy & Eberhart, 1997) and binary differential evolution (BDE) (Pampara et al., 2006), in solving CSKGs. Compared with FPTA, the combination of heuristic and EAs can significantly improve the exploration ability and search efficiency. Notice that although CSKGs still require that an agent cannot but join only a coalition, they allow a coalition to undertake diverse tasks, so that an agent can respond to different tasks in disguised forms. However, the qualitative characteristic of CSKGs assumes that each agent's resources are inexhaustible. Obviously, this assumption is too idealistic and is suitable only for some specific scenarios.

When it comes to OCFGs, to our best knowledge, the first attempt from the perspective of algorithms was made by Shehory and Kraus (1998). They proposed greedy algorithms to successively generate a coalition for a task in terms of the task priority, in which coalitions may partially overlap. However, they set a specific limitation on the coalition size, while how to set the coalition size is very difficult. To this end, Zhan et al. (2012) proposed a hybrid algorithm, henceforth called DPG, to solve the OCSGP in threshold task games (TTGs) (Chalkiadakis et al., 2010) on the basis of dynamic programming and greedy approach. TTGs is a specific case of OCFGs, in which each agent has only a type of resources. If a group of agents want to complete a task, they must serve the resource needs of the task. Consequently, DPG is very close to the solution in task-related settings. To efficiently solve OCSGP under multiple types of resources, Zhang et al. (2010, 2011, 2020) addressed the characteristic-function-based OCSGP in resource-constrained and subadditive task-oriented domains. Then, to resolve the potential resource conflicts between competitive coalitions, Zhang et al. (Zhang et al., 2020) developed a task-oriented heuristic (TOH) for solution repair, in which an agent will join a coalition only if it has the required resources and a task can be done only if a coalition can satisfy its resource needs. Particularly, TOH can be used in binary EAs to generate a feasible OCS more efficiently, especially in harsh environments with fierce competition over scarce resources.

2.3 Applications of OCFGs

Wang et al. (2014) proposed a distributed cooperative sensing algorithm in which the secondary users self-organize into a desirable network structure with overlapping coalitions. Zhang et al. (2014) formulated the cooperation problem between small cells as an OCFG to optimize their sumrate under the maximum transmit power constraints. Di et al. (2017) made smartphone users form overlapping coalitions for different sensing tasks to improve the quality of smartphone applications. Zhang et al. (2017) employed OCFGs to achieve effective opportunistic selection and cooperative localization for wireless networks. Hou et al. (2020) achieved the resource allocation for multi-cast transmission in wireless small cell networks by OCFGs. Qi et al. (2023) presented a sequential OCFG for resource allocation according to the task execution order in heterogeneous unmanned aerial vehicle networks.

Xiao et al. (2015) established a Bayesian non-transferable utility OCFG to model and analyze the spectrum sharing problem between a set of device-to-device (D2D) links in

cellular networks. Sun et al. (2016) proposed an overlapping coalition formation-based double auction to jointly consider the multi-demand of buyers, heterogeneous spectrum, and economical efficiency. Zhao et al. (2017) used OCFGs to allocate resource in the network coding aided cooperative D2D communications. Srinivasan and Murthy (2020) proposed spectrum slicing coalition formation protocol to address effective spectrum slicing. Chen et al. (2021) presented a generalized user grouping concept for non-orthogonal multiple access from an overlapping perspective.

Guajardo et al. (2018) addressed the collaborative transportation problem with overlapping coalition configuration to minimize the total cost. Shi et al. (2023) adopted OCFGs to select social vehicles to contribute data and improve the high definition map quality. Krausburg et al. (2021a, 2021b) proposed a sequential characteristic-function game to solve a disaster response management problem, in which discrete overlapping coalitions must respond to a disaster event.

Obviously, most of the above practical applications are task driven: at each time, select the right coalition to undertake one task with the highest priority according to the priority of multiple tasks. However, these applications focus on finding feasible coalitions rather than an OCS. Differently, this work studies OCSGP in task-based settings. Specifically, in this work, a new model of the task-based OCSGP is proposed and analyzed in terms of the computational complexity. Note that this work focuses on how to search for the optimal OCS from the algorithmic perspective, rather than theoretical investigations on the stability of OCS. Therefore, a generic EAF for solving this specific OCSGP is developed to deal with the task assignment in a parallel manner. It is expected that the proposed generic model and algorithm can be used in the above real-world applications that have the specific need to efficiently and parallelly assign the right multiple tasks to agents.

3. Overlapping Coalition Structure Generation in Task-Based Settings

In numerous practical applications, coalitions are often formed to perform tasks from a collection of possible tasks that an agent system must undertake. However, the conventional characteristic-function-based OCSGP usually makes certain restrictive assumptions about coalition values. On the one hand, it frequently posits that a coalition's value is realized when its members combine their resources and tasks in some manner. On the other hand, it maintains that a coalition's value is unaffected by the actions of non-members. Clearly, the characteristic-function-based OCSGP does not address computational issues from the perspective of the connection between coalitions and tasks (Dang & Jennings, 2006). For OCSGP in a general task-based setting, the concept of tasks generally heightens the problem's complexity, as there are usually numerous ways to map coalitions to tasks, which will be demonstrated in greater detail in Section 4. Moreover, a coalition's value now hinges on the tasks it performs, and these tasks are accessible to all the agents that are interested in them. Therefore, the OCSGP in a task-based environment can be considered a more general case, making it easier to be applied in various scenarios.

In the realm of operations management and artificial intelligence, numerous scenarios can be interpreted as the OCSGP in task-based environments. Consider agents in the following cases.

> **Multiple virtual organizations:**
> In collaborative commercial orders, various agile enterprises team up to share resources in order to fulfill collective orders. However, each enterprise has finite resources and is likely to prioritize orders that align with their interests, such as those with friendly

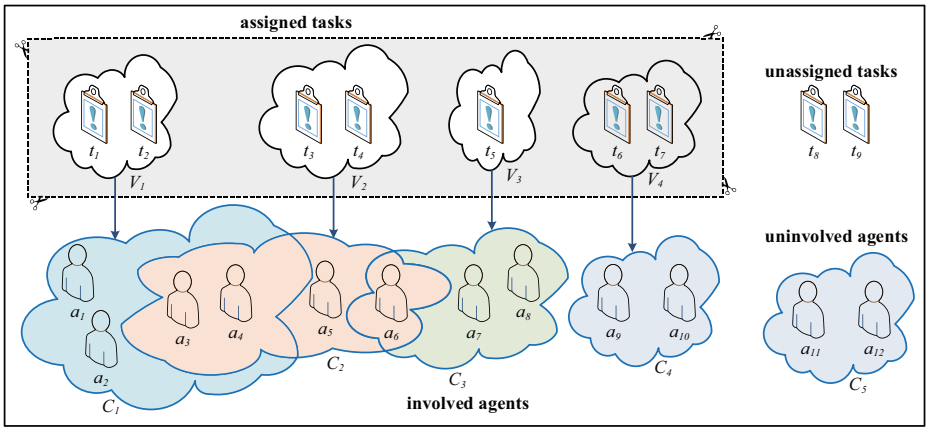


Figure 1: The illustration of the overlapping coalition structure in task-based settings.

costs, short lead times, or convenient coordination. Accordingly, the motivation is to create a network of overlapping virtual organizations that can fill the most profitable orders (Zhang et al., 2010).

Disaster response management:
In post-disaster emergency response, multiple supply locations possess different types of emergency relief supplies and multiple demand locations require services. In this scenario, the primary focus is on deploying emergency relief supplies efficiently and promptly (Krausburg et al., 2021b; Su et al., 2016). Since each supply location has limited resources, they may primarily respond to nearby or severely damaged demand locations. The objective, therefore, is to form a set of overlapping disaster response coalitions of supply locations to ensure that as many victims as possible can be covered.

We believe that the focus on the OCSGP in task-based settings is very natural, given the concerns of MAS and related disciplines: resource limitations, task preferences, and the need to efficiently assign the right tasks to agents. For a better understanding, Figure 1 describes the OCS in a task-based setting. Specifically, there is a set of agents, $A = \{a_1, \ldots, a_n\}$, $n \in \mathbb{N}$, confronting a set of simultaneous tasks, $T = \{t_1, \ldots, t_m\}$, $m \in \mathbb{N}$.

Each agent $a_j \in A$, $j \in \{1, \ldots, n\}$, has a r-dimensional resource vector, $\mathbf{B}_j = [b_1^j, \ldots, b_r^j]$, in which $b_k^j \in \mathbb{N}$ is the amount of resources of type k, $k \in \{1, \ldots, r\}$, $r \in \mathbb{N}$. Besides, each agent a_j has an exclusive interest set $T_j \subseteq T$, satisfying $\bigcup_{j=1}^{n} T_j = T$. Agent a_j will invest resources only in the tasks of interest in T_j.

Each task $t_i \in T$, $i \in \{1, \ldots, m\}$, has a demand vector, $\mathbf{D}_i = [d_1^i, \ldots, d_r^i]$, in which $d_k^i \in \mathbb{N}$ is the required amount of resources of type k of task t_i.

Once an agent a_j decides to serve a task t_i, a_j will have a contribution vector, $\mathbf{W}_{ji} = [w_1^{ji}, \ldots, w_r^{ji}]$, in which $w_k^{ji} \in \mathbb{N}$ is the invested amount of resources of type k of agent a_j for task t_i, satisfying $w_k^{ji} \leq \min\{d_k^i, b_k^j\}$. Note that if $t_i \ni T_j$, for each $k \in \{1, \ldots, r\}$, $w_k^{ji} = 0$.

Definition 1. *A task-based coalition (C, V) is a tuple comprising a non-empty agent subset $C \subseteq A$ and a collective task subset $V \subseteq T$ (Dang & Jennings, 2006).*

Definition 2. *A task-based overlapping coalition structure, $OCS = \{(C_1, V_1), \ldots, (C_s, V_s)\}$, is a set of task-based coalitions, as shown in Figure 1. Each C_l, $l \in \{1, \ldots, s\}$, will undertake a V_l, such as:*

- *For each $l \in \{1, \ldots, s\}$, $C_l \subseteq A$ and $C_l \neq \emptyset$;*

- $|\{(C_x, C_y)|x \neq y \wedge C_x \cap C_y \neq \emptyset\}| \geq 0$, *namely,* $C_1, \ldots, C_s$ *may be overlapped;*
- $\cup_{l=1}^{s} C_l = A$ *and* $(C_1, \ldots, C_s)$ *may be a cover of* A*;*
- *For each* $l \in \{1, \ldots, s\}$, $V_l \subseteq T$*;*
- $|\{(V_x, V_y)|x \neq y \wedge V_x \cap V_y = \emptyset\}| = 0$, *namely,* $V_1, \ldots, V_s$ *are disjoint;*
- $|\{V_l|V_l = \emptyset\}| \leq 1$. *That is, at most one* V_l *is empty, which can happen only if there exist some idle agents. In this case, all the idle agents will be merged into a* C_l *with* $V_l = \emptyset$*.*

In the above representations, a task can be completed only by at most one C_l, or no C_l can take on it due to unavailable agents or insufficient resources. If C_l has required and sufficient resources, it can simultaneously undertake multiple tasks of interest. Notice that in an OCS, if $C_1, \ldots, C_s$ are overlapped, $(C_1, \ldots, C_s)$ is often a cover rather than a partition of A, as $C_1, \ldots, C_s$ have overlapping members and their union covers all the members in A. On the other hand, $(V_1, \ldots, V_s)$ is a subset of a partition of T. This is because $V_1, \ldots, V_s$ are pairwise disjoint subsets of T and some tasks may not be served, namely, $\cup_{l=1}^{s} V_l \subseteq T$.

Definition 3. *A coalition* (C_l, V_l) *is an idle coalition if and only if* $C_l \neq \emptyset$ *and* $V_l = \emptyset$*.*

Example 1. *The OCS instance in Figure 1 is*

$$\begin{aligned}\{&(C_1 = \{a_1, a_2, a_3, a_4\}, V_1 = \{t_1, t_2\}),\\ &(C_2 = \{a_3, a_4, a_5, a_6\}, V_2 = \{t_3, t_4\}),\\ &(C_3 = \{a_6, a_7, a_8\}, V_3 = \{t_5\}),\\ &(C_4 = \{a_9, a_{10}\}, V_4 = \{t_6, t_7\}),\\ &(C_5 = \{a_{11}, a_{12}\}, V_5 = \emptyset)\}\end{aligned}$$

where C_1, C_2, C_3 *are overlapped;* C_4 *is disjoint;* V_1, V_2, V_3, V_4, V_5 *are disjoint; and* (C_5, V_5) *is idle, because it cannot afford any of* $\{t_8, t_9\}$*.*

Definition 4. *A coalition* (C_l, V_l) *in an* $OCS = \{(C_1, V_1), \ldots, (C_s, V_s)\}$ *is feasible if and only if* (C_l, V_l) *satisfies:*

- *For each agent* $a_j \in C_l$, $T_j \cap V_l \neq \emptyset$*, namely, each agent* a_j *in* C_l *can find at least a task of interest in* V_l*;*
- *For each task* $t_i \in V_l$, $\exists a_j \in C_l$ *satisfies* $t_i \in T_j$*, namely, each task* t_i *in* V_l *can find at least an agent* a_j *in* C_l *that is interested in it;*
- *For each task* $t_i \in V_l$*, and for each* $k \in \{1, \ldots, r\}$*,*

$$d_k^i = \sum\nolimits_{a_j \in C_l} w_k^{ji} \tag{1}$$

  That is, the resource requirements of each task t_i *in* V_l *can be met;*

- *For each agent* $a_j \in C_l$*, and for each* $k \in \{1, \ldots, r\}$*,*

$$\sum\nolimits_{x=1}^{s} \sum\nolimits_{t_i \in V_x} w_k^{ji} \leq b_k^j \tag{2}$$

  It means that the total amount of resources that each a_j *in* C_l *contributes to all the coalitions in the OCS cannot exceed the amount it has. In other words, there can be no resource conflict no matter within* (C_l, V_l) *or between* $(C_1, V_1), \ldots, (C_s, V_s)$*.*

Definition 5. *An OCS is feasible if and only if it includes at least a feasible coalition.*

It is worth mentioning that the proposed model is a natural variation of the traditional CSG in a task-based setting (Dang & Jennings, 2006). Differently, based on the above definitions, determining whether a coalition (C_l, V_l) in an OCS is feasible is not trivial. We need to consider not only each agent's interest set, but also the potential resource conflicts within and between coalitions. Notably, Constraint Sets (1) and (2) become much more strict than the feasibility conditions in (Dang & Jennings, 2006; Zhang et al., 2020). Additionally, the infeasible coalitions make no contribution to the value of an OCS. Therefore, all the infeasible coalitions can be merged into the idle coalition.

Traditionally, if (C_l, V_l) is feasible, its value can be computed by

$$v(C_l, V_l) = \sum\nolimits_{t_i \in V_l} \mu_i \tag{3}$$

where $\mu_i \in \mathbb{N}$ is a reward rendered by task t_i, if t_i can be completed by members in C_l. Note that $v(C_l \neq \emptyset, V_l = \emptyset)$=0 for the reason that no task is served by C_l. Then, the value of an OCS is just the total values of all the involved coalitions:

$$v(OCS) = \sum\nolimits_{l=1}^{s} v(C_l, V_l) \tag{4}$$

Given the above terms, the OCSGP is concentrated on finding an optimal OCS^* to maximize the social welfare:

$$OCS^* = \arg\max\nolimits_{OCS \in \prod_A^T} v(OCS) \tag{5}$$

where $\prod_A^T$ represents the set of all the possible OCSs derived from A and T.

4. Search Space of Proposed OCSGP

Is the proposed OCSGP still intractable under the newly added restrictive assumptions? To answer this question, in this section, we investigate the size of the search space of the proposed OCSGP. We aim to present a minimal search to establish a bound from the optimal. We have the following results.

Proposition 1. *The total number of possible* (C, V) *with* $V \neq \emptyset$ *is* $(2^n - 1) \cdot (2^m - 1)$.

Proof. It is well-known that there are in total $(2^n - 1)$ non-empty subset C of A (Rahwan et al., 2015). Similarly, there are $(2^m - 1)$ possible nonempty subset V of T. In a task-based setting, each C may be assigned a non-empty V. Consequently, there are at total $(2^n - 1) \cdot (2^m - 1)$ possible (C, V) with $V \neq \emptyset$. □

Proposition 2. *In an OCS, the number of agent occurrences is at most* $n \cdot 2^{n-1}$.

Proof. It is possible that an OCS contains all the different subsets of A. In this case, the number of agent occurrences is $1 \cdot \mathcal{C}_n^1 + 2 \cdot \mathcal{C}_n^2 + \cdots + n \cdot \mathcal{C}_n^n = \sum_{j=1}^{n} (j \cdot \mathcal{C}_n^j) = n \cdot 2^{n-1}$. □

Proposition 3. *For an* $OCS = \{(C_1, V_1), \ldots, (C_s, V_s)\}$, *we have* $s \leq m + 1$.

Proof. On one hand, each task is assigned at most one coalition, so for the m given tasks in T, there are at most m different, assigned coalitions in an OCS, each of which carries out a single, different task. On the other hand, some agents may not be selected to carry out the given tasks, so as well as the m coalitions, the OCS may also contain a coalition that has agent members but does nothing (i.e., the collective task subset is empty). Therefore, there are at most $m + 1$ overlapping coalitions in an OCS, namely, $s \leq m + 1$. □

Proposition 4. *For A, the total number of possible cover $(C_1, \ldots, C_s)$ with size s is $\frac{(2^n-1)!}{(2^n-1-s)!}$.*

Proof. See Appendix B. □

Proposition 5. *For T, the total number of possible partition subset $(V_1, \ldots, V_s)$ with size s is $\sum_{x=s}^{m+1} [\mathcal{C}_{m+1}^x \cdot S(x,s)]$.*

Proof. See Appendix B. □

Proposition 6. *The total number of possible OCSs with size s is*

$$\frac{(2^n-1)! \cdot \sum_{x=s}^{m+1} [\mathcal{C}_{m+1}^x \cdot S(x,s)]}{(2^n-1-s)!}$$

Proof. See Appendix B. □

Proposition 7. *The total number of possible OCSs is*

$$|\prod\nolimits_A^T| = \sum\nolimits_{s=1}^{\min\{2^n-1, m+1\}} \frac{(2^n-1)! \cdot \sum_{x=s}^{m+1} [\mathcal{C}_{m+1}^x \cdot S(x,s)]}{(2^n-1-s)!}$$

Proof. See Appendix B. □

Proposition 8. *The smallest number of OCSs that needs to be searched to establish a bound from the optimal OCS^* is at least $2^m - 1$.*

Proof. See Appendix B. □

Proposition 9. *The smallest number of OCSs that needs to be searched to establish a bound from the optimal OCS^* is $2^m - 1$.*

Proof. See Appendix B. □

5. Computational Complexity of Proposed OCSGP

The optimal OCS^* must be a feasible OCS. Usually, there are two types of coalitions in an OCS: one is the disjoint coalition with no overlapping member (e.g., C_4 in Figure 1); and the other is the overlapping coalition where some members are also contained by other coalitions (e.g., C_1, C_2, C_3 in Figure 1). Obviously, for a disjoint coalition, we only need to check whether a feasible resource allocation scheme can be found within the coalition. When it comes to overlapping coalitions, we not only need to check the resource allocation within each associated coalition, but also need to check whether there is a resource conflict between associated coalitions in terms of the overlapping members. In this section, we introduce and define a number of natural decision problems associated with the above cases, and then analyze their computational complexity. The results of this section are summarized in Table 1. Before proceeding, it should be noted that, if a decision problem is in P, we develop at least one algorithm of polynomial time complexity to solve that problem.

Problem	Comlexity	Reference
Feasible Disjoint Coalition	in P	Proposition 10
Feasible Overlapping Coalition	in P	Proposition 12
Feasible Overlapping Coalition Structure	in P	Proposition 14
Overlapping Coalition Structure Generation	NP-complete	Proposition 16

Table 1: Main complexity results relating to the proposed OCSGP.

5.1 Disjoint Coalition

In a disjoint coalition (C, V), C is an isolated subset of A. In this case, we only need to focus on whether we can find a feasible resource allocation scheme for C to serve each task in V.

Proposition 10. *The problem of checking whether a disjoint coalition (C, V) in an OCS is feasible is in P.*

Proof. See Appendix B. □

To verify Proposition 10, Algorithms 1 and 2 show a deterministic algorithm for checking whether a disjoint coalition (C, V) in an OCS is feasible. We can explain the algorithm as follows:

- Check the validity of members in (C, V) based on Algorithm 2. As long as $\exists a_j \in C$ and $T_j \cap V = \emptyset$, or $\exists t_i \in V$ and $t_i \ni \bigcup_{a_j \in C} T_j$, or $\exists k \in \{1, \ldots, r\}, \exists t_i \in V$ and $\sum_{a_j \in C \wedge t_i \in T_j} b_k^j < d_k^i$, it can be immediately obtained that (C, V) is infeasible.
- If the resource requirements of each task in V can be met by the interested agents in C, further check whether there is a resource conflict between competitive tasks. For this purpose, build a flow network (Ahuja et al., 1993) based on Figure 2. For each $k \in \{1, \ldots, r\}$, update the capacity on each arc in the network with b_k^j and d_k^i, and use the Edmonds-Karp algorithm (Cormen et al., 1990) to compute the maximum flow $f_{\max}^k$ in the network. Based on the max-flow min-cut theorem and the integral flow theorem (Ahuja et al., 1993), if $f_{\max}^k < \sum_{t_i \in V} d_k^i$, there is no feasible flow in the network that can make each arc (t_i, E) saturated. This indicates that there exists a conflict over the resources of type k between competitive tasks. In this case, (C, V) is infeasible.
- If for each $k \in \{1, \ldots, r\}$, $f_{\max}^k \geq \sum_{t_i \in V} d_k^i$, there is no resource conflict between tasks. Thus, (C, V) is feasible.

Proposition 11. *The worst case complexity of Algorithm 1 is $O(n^6)$.*

Proof. See Appendix B. □

5.2 Overlapping Coalition

An overlapping coalition refers to a situation where it has some shared members with other coalitions. This scenario is rather intricate. For instance, assume that (C_1, V_1) and (C_2, V_2) have the same members, and (C_2, V_2) and (C_3, V_3) share the same members as well. It

Algorithm 1 Checking the feasibility of a disjoint (C, V).

Require: a disjoint (C, V)
check the validity of members based on Algorithm 2;
if (C, V) is invalid **then**
  return infeasible;
else
  build a flow network based on Figure 2;
  for each $k \in \{1, \ldots, r\}$ **do**
    update the capacity on each arc in the network with b_k^j and d_k^i;
    compute the maximum flow $f_{\max}^k$ in the network;
    if $f_{\max}^k < \sum_{t_i \in V} d_k^i$ **then**
      return infeasible;
    end if
  end for
end if
return feasible;

Algorithm 2 Checking the validity of agent and task members in (C, V).

Require: (C, V)
for each agent $a_j \in C$ **do**
  if $\exists T_j \cap V = \emptyset$ **then**
    return invalid;
  end if
end for
for each task $t_i \in V$ **do**
  if $\exists t_i \ni \bigcup_{a_j \in C} T_j$ **then**
    return invalid;
  end if
end for
for each task $t_i \in V$ **do**
  for each $k \in \{1, \ldots, r\}$ **do**
    if $\sum_{a_j \in C \wedge t_i \in T_j} b_k^j < d_k^i$ **then**
      return invalid;
    end if
  end for
end for
return valid;

is clear that even if (C_1, V_1) and (C_3, V_3) do not have overlapping members, the resource allocation strategy of (C_2, V_2) will inevitably impact the feasibility of both (C_1, V_1) and (C_3, V_3). In other words, (C_1, V_1) and (C_3, V_3) could potentially have a resource constraint relationship due to their shared connection through (C_2, V_2). Therefore, we must consider all the overlapping coalitions that have resource constraint relationships through their shared members to determine if a feasible overall resource allocation scheme exists to cater to the needs of all the involved coalitions. To this end, we provide the following definition.

Definition 6. *An associated coalition set (ACS) includes all the overlapping coalitions in which the relationship occurs and is passed through the joint agent members.*

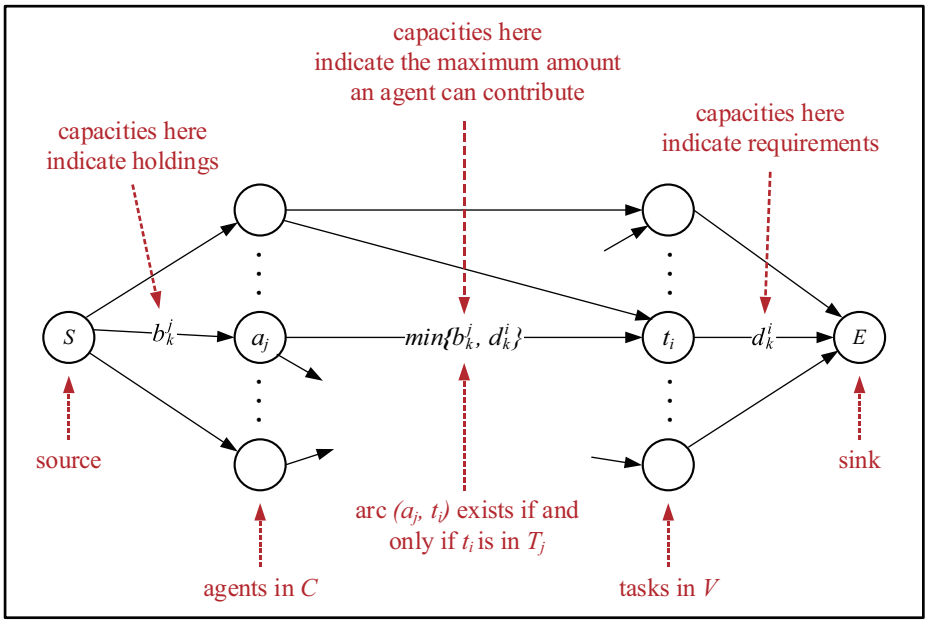


Figure 2: The overall structure of the flow network construction for the disjoint (C, V). Specifically, we add a source node S and a sink node E in a directed graph; each agent $a_j \in C$ is a supply node near S, and there is an arc between S and a_j; each task $t_i \in V$ is a demand node beside E, and there is also an arc between t_i and E; and for each agent $a_j \in C$ and each task $t_i \in V$, as long as $t_i \in T_j$, there exists an arc between a_j and t_i. Intuitively, the capacity for arc (S, a_j) indicates b_k^j; the capacity on arc (t_i, E) denotes d_k^i; and the capacity on arc (a_j, t_i) is $\min\{b_k^j, d_k^i\}$.

Example 2. *The ACS instance in Figure 1 is*

$$\{(C_1 = \{a_1, a_2, a_3, a_4\}, V_1 = \{t_1, t_2\}),$$
$$(C_2 = \{a_3, a_4, a_5, a_6\}, V_2 = \{t_3, t_4\}),$$
$$(C_3 = \{a_6, a_7, a_8\}, V_3 = \{t_5\})\}$$

Notice that although C_1 and C_3 are disjoint, they are connected through C_2. Clearly, $ACS \subseteq OCS$.

Proposition 12. *The problem of checking whether an overlapping coalition (C, V) in an OCS is feasible is in P.*

Proof. See Appendix B. □

To verify Proposition 12, Algorithm 3 lists a deterministic algorithm for checking whether an overlapping coalition (C, V) in an OCS is feasible. We can explain the algorithm as follows:

- Compute the ACS related to (C, V) and copy a backup ACS^* for the ACS.
- For each coalition $(C_l, V_l) \in ACS$, check the validity of members in (C_l, V_l) based on Algorithm 2. If (C_l, V_l) is invalid, (C_l, V_l) is infeasible, and thus execute $ACS \leftarrow ACS - \{(C_l, V_l)\}$ to drop (C_l, V_l) out of ACS.

Algorithm 3 Checking the feasibility of an overlapping (C, V).

Require: an overlapping (C, V)
calculate the ACS related to (C, V);
$ACS^* \leftarrow ACS$;
for each coalition $(C_l, V_l) \in ACS$ **do**
  check the validity of members based on Algorithm 2;
  if (C_l, V_l) is invalid **then**
    (C_l, V_l) is infeasible;
    $ACS \leftarrow ACS - \{(C_l, V_l)\}$;
  end if
end for
if $ACS = \emptyset$ **then**
  return all the coalitions in ACS^* are infeasible;
end if
if $ACS \neq \emptyset$ **then**
  build a flow network based on Figure 3;
  for each $k \in \{1, \ldots, r\}$ **do**
    update the capacity on each arc in the network with b_k^j and d_k^i;
    compute the maximum flow $f_{\max}^k$ in the network;
    if $f_{\max}^k < \sum_{t_i \in ACS} d_k^i$ **then**
      return all the coalitions in ACS^* are infeasible;
    end if
  end for
end if
return coalitions in ACS are feasible and coalitions in $ACS^* - ACS$ are infeasible;

- If $ACS = \emptyset$, all the overlapping coalitions in ACS^* are infeasible.
- If $ACS \neq \emptyset$, build a flow network based on Figure 3. For each $k \in \{1, \ldots, r\}$, update the capacity on each arc in the network with b_k^j and d_k^i, and compute the maximum flow $f_{\max}^k$ in the network. If $f_{\max}^k < \sum_{t_i \in ACS} d_k^i$, the residual coalitions in ACS are infeasible, implying that all the overlapping coalitions in ACS^* are infeasible.
- If for each $k \in \{1, \ldots, r\}$, $f_{\max}^k \geq \sum_{t_i \in ACS} d_k^i$, all the overlapping coalitions in ACS are feasible, while the coalitions in $ACS^* - ACS$ are infeasible.

Proposition 13. *The worst case complexity of Algorithm 3 is* $O(n^7)$.

Proof. See Appendix B. □

5.3 Overlapping Coalition Structure

Using the previously established theoretical and algorithmic findings, when given an OCS, we merely need to explore the OCS to identify all the disjoint coalitions and ACSs. Subsequently, we can sequentially evaluate the feasibility of each disjoint coalition and ACS. To facilitate understanding, we have developed a deterministic algorithm for identifying disjoint coalitions and ACSs for a given OCS, which is presented in Algorithm 4.

Proposition 14. *The problem of checking whether an OCS is feasible is in P.*

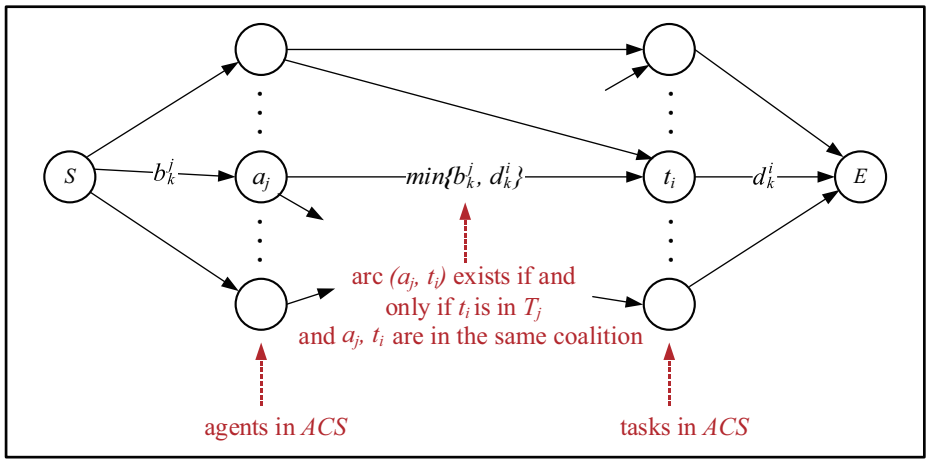


Figure 3: The overall structure of the flow network construction for the non-empty ACS. Similarly, each agent a_j in ACS is a supply node near S and each task t_i in ACS is a demand node beside E. Differently, there exists an arc between a_j and t_i if and only if $t_i \in T_j$, and at the same time, a_j and t_i belong to the same coalition (C_l, V_l) in ACS (i.e., $a_j \in C_l, t_i \in V_l$). That is, the arcs between agents and tasks in the network are divided according to each overlapping coalition in ACS. If t_i and a_j are not in the same coalition, an arc cannot be constructed even if a_j is interested in t_i, because no such collaboration exists in ACS.

Algorithm 4 Identifying disjoint coalitions and ACSs for a given OCS.

Require: an OCS
1: **while** $OCS \neq \emptyset$ **do**
2: **for** each coalition (C, V) in OCS **do**
3: **if** (C, V) has overlapping agents **then**
4: put all the connected coalitions in a ACS;
5: $OCS \leftarrow OCS - ACS$;
6: **else**
7: put (C, V) in a disjoint coalition set;
8: $OCS \leftarrow OCS - \{(C, V)\}$;
9: **end if**
10: **end for**
11: **end while**
12: **return** disjoint coalitions and ACSs;

Proof. See Appendix B. □

To verify Proposition 14, Algorithm 5 illustrates a deterministic algorithm for checking whether an OCS is feasible. We can explain the algorithm as follows:

- Traverse the pending OCS to delineate all the disjoint coalitions and all the ACSs based on Algorithm 4. Note that there may be more than one ACS in the OCS.
- For each disjoint coalition and each ACS, Algorithm 1 and Algorithm 3 can be called to discriminate their feasibility, respectively.

Algorithm 5 Checking the feasibility of an OCS.

Require: OCS

```
delineate all the disjoint coalitions and all the ACSs based on Algorithm 4;
for each disjoint coalition do
    check its feasibility based on Algorithm 1;
end for
for each ACS do
    check the feasibility of the contained overlapping coalitions based on Algorithm 3;
end for
for each feasible coalition do
    compute its value according to Equation (3);
end for
compute the value of OCS based on Equation (4);
return v(OCS);
```

- For all the feasible coalitions, their values can be calculated based on Equation (3). Then, the value of the OCS can be obtained according to Equation (4).

Proposition 15. *The worst case complexity of Algorithm 5 is* $O(n^7)$.

Proof. See Appendix B. □

Proposition 16. *The OCSGP is NP-complete.*

Proof. See Appendix B. □

6. EAF for Solving the Proposed OCSGP

It should be noted that Algorithms 1-5 can only be used to determine whether a pending OCS is feasible, but cannot find a satisfactory OCS in task-based settings. On the other hand, the theoretical results of complexity analysis show that the proposed OCSGP is intractable for the reason that the number of possible OCSs is exponential with n and m. It is impractical to find an OCS^* in polynomial time, especially when n and m is large. In many complex real-world decision-making problems, the pursuit of an optimal solution at significant computational cost is often not justified. In scenarios where high accuracy and real-time performance are desired, it is often more practical to find a near-optimal or satisfactory solution within an acceptable time.

EAs (Poli & Langdon, 2006; Slowik & Kwasnicka, 2020; Yu et al., 2012) are often highly effective in approximating near-optimal solutions for a wide range of problems and are widely used as optimization tools for complex real-world problems, especially in the context of searching for coalition structures (Sen & Dutta, 2000; Liu et al., 2016; Zhang et al., 2010, 2011, 2020) and evolving coalitions (Seo et al., 1999, 2000; Zhang et al., 2015). To solve the proposed OCSGP more efficiently, in this section, we adopt EAs to develop a generic EAF to find the approximately optimal OCS. However, although there have been a large amount of superior EAs for solving constrained optimization problems, these constrained EAs are problem-specific and unavailable for OCSGP. This is because the resource allocation scheme of an OCS directly determines its feasibility, while Constraint Sets (1) and (2) are too strict to be satisfied in the search course of EAs. Specifically, with the evolution of EAs, a population of candidate solutions also stay changing. The original EAs have no

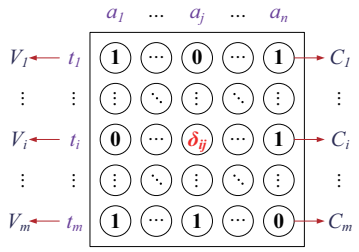


Figure 4: The 2D solution, representing a possible OCS.

problem-specific constraint handling technique and cannot guarantee that the generated solutions satisfy Constraint Sets (1) and (2). In such cases, EAs often fail in finding a feasible solution even after a considerable number of fitness evaluation. Consequently, the proposed EAF needs to uncover and resolve the potential resource conflicts over resources between the competitive overlapping coalitions. For this purpose, we embed a polynomial-time heuristic for solution repair in the proposed EAF. This customized heuristic assigns a task from the perspective of the current, remaining resources of agents, which can prevent resource conflicts and maximize the utilization of all the available resources. In contrast to the original EAs, the proposed EAF is enhanced by the embedded heuristic, always searches within the feasible region, and locates feasible OCSs more quickly, thereby a higher chance to find better OCSs.

For the purpose of illustration, we start by proposing the solution representation and show how to decode a solution. Then, we show how to generate a feasible OCS by solution repair. Finally, we detail the procedure of EAF and investigate its computational complexity.

6.1 Solution Representation, Initialization, and Decoding

To characterize an OCS, we design a specific 2D binary solution representation, as shown in Figure 4. Row i represents task t_i, and column j stands for agent a_j. If bit $\delta_{ij} = 1$, a_j will take on t_i; otherwise, a_j has nothing to do with t_i.

To initialize a solution, we generate a random number $rand \in (0, 1)$ with a normal distribution for each δ_{ij}. If $rand \geq 0.5$, $\delta_{ij} \leftarrow 1$; otherwise, $\delta_{ij} \leftarrow 0$.

To decode a solution, we present a deterministic algorithm, as shown in Algorithm 6. Specifically, we first go through each row. It is clear that each row represents a possible coalition $(C_i, V_i = \{t_i\})$. For each row with no bit "1", $C_i \leftarrow \emptyset$. For those rows with exactly the same agent members, merge the corresponding C_i and V_i, respectively. Then, we check each column. For columns with no bit "1", we put these agents in a subset C', whose collective task subset is empty. Finally, all the non-empty C_i with their V_i and the non-empty C' can form an OCS.

Example 3. *Consider the following solution.*

	a_1	a_2	a_3	a_4	a_5	a_6
t_1	0	1	0	1	1	0
t_2	1	0	1	0	1	0
t_3	0	1	0	1	1	0
t_4	0	0	0	0	0	0

Algorithm 6 The algorithm for decoding a solution.

Require: a solution

for each $i \in \{1, \ldots, m\}$ **do**
  $\mathbf{C}_i \leftarrow \emptyset$;
  $\mathbf{V}_i \leftarrow \emptyset$;
  for each $j \in \{1, \ldots, n\}$ **do**
    if $\delta_{ij} = 1$ **then**
      $C_i \leftarrow C_i + \{a_j\}$;
    end if
  end for
  if $C_i \neq \emptyset$ **then**
    $V_i \leftarrow V_i + \{t_i\}$;
  end if
end for
$C' \leftarrow \emptyset$
for each $j \in \{1, \ldots, n\}$ **do**
  if there is no bit "1" in column j **then**
    $C' \leftarrow C' + \{a_j\}$
  end if
end for
Merge all the C_i and V_i with exactly the same agent members;
$OCS \leftarrow \emptyset$;
for each coalition (C_i, V_i) **do**
  if $C_i \neq \emptyset$ **then**
    $OCS \leftarrow OCS + \{(C_i, V_i)\}$;
  end if
end for
if $C' \neq \emptyset$ **then**
  $OCS \leftarrow OCS + \{(C', \emptyset)\}$;
end if
return an OCS;

It can be observed that $(C_1 = \{a_2, a_4, a_5\}, V_1 = \{t_1\})$, $(C_2 = \{a_1, a_3, a_5\}, V_2 = \{t_2\})$, $(C_3 = \{a_2, a_4, a_5\}, V_3 = \{t_3\})$, $C_4 = \emptyset$, *and* $C' = \{a_6\}$. *Since* C_1 *and* C_3 *have the same agent members, the two coalitions can be merged into* $(C_1 = \{a_2, a_4, a_5\}, V_1 = \{t_1, t_3\})$. *Taking* C_1, C_2, *and* C' *together, we obtain an OCS:*

$$\{(\{a_2, a_4, a_5\}, \{t_1, t_3\}), (\{a_1, a_3, a_5\}, \{t_2\}), (\{a_6\}, \emptyset)\}$$

Note that although we can directly decode an OCS from the solution, we do not know whether this OCS is feasible or not, so that we cannot calculate its value. Accordingly, the next major difficulty is: how to check, repair, and evaluate the decoded OCS?

6.2 The Heuristic for Solution Repair

According to Definition 4, we know that the core issue of coalition feasibility is determining whether there are possible resource conflicts within and between overlapping coalitions in an OCS. Under the strict constraints of resources and interest sets, when we maximize the

total benefits, some tasks may be facilitated, but others hindered. To analyze the solution and characterize the possible resource conflicts, we give the following definitions.

Definition 7. *Each agent $a_j \in A$ has a residual vector $\mathbf{P}_j = [p_1^j, \ldots, p_r^j]$, where $p_k^j \in \mathbb{N}$ is the current residual amount of resources of type k of agent a_j.*

Definition 8. *Each agent subset $C_i \subseteq A$ has an investment vector $\mathbf{E}_i = [e_1^i, \ldots, e_r^i]$, where $e_k^i \in \mathbb{N}$ is the amount of resourced of type k invested by agent members for C_i, satisfying for each $k \in \{1, \ldots, r\}$, $e_k^i = \sum_{a_j \in C_i} w_k^{ji}$.*

To pursue the feasibility of an OCS, in this work, we put forward a heuristic for solution repair. The core idea is to deal with a pending solution from the perspective of the $\mathbf{D}_i$ of each task and the $\mathbf{P}_j$ of each agent. First, we check each row. For each $i \in \{1, \ldots, m\}$, we examine whether t_i can be afforded by agents that have interest in it. If the $\mathbf{P}_j$ of the active agents cannot achieve $\mathbf{D}_i$, t_i should be excluded from the OCS. If $\mathbf{D}_i$ is achievable, we make $(C_i, V_i = \{t_i\})$ feasible by reversing bits in row i on the basis of the constraints of resources and interest sets. Then, we determine each member's $\mathbf{W}_{ji}$ and update their $\mathbf{P}_j$ to ensure that $\mathbf{E}_i = \mathbf{D}_i$. Similarly, other rows can also be checked according to agents' $\mathbf{P}_j$ and tasks' $\mathbf{D}_i$. After all the rows have been inspected, we decode the solution to generate an OCS according to Algorithm 6 and compute its value on the basis of Equations (3) and (4).

The basic process of the proposed heuristic for solution repair is shown in Algorithm 7. We can explain the algorithm as follows:

- Initialize $\mathbf{E}_i = 0$ and $\mathbf{P}_j = \mathbf{B}_j$.
- Examine the feasibility of (C_i, V_i) by repeating the following procedure until all the rows are travelled.
  - Pick randomly an unchecked row i from a uniform distribution.
  - If $\exists k \in \{1, \ldots, r\}$, $\sum_{j=1 \wedge t_i \in T_j}^{n} p_k^j < d_k^i$, task t_i cannot be completed due to insufficient resources of agents. Let agents quit the competition for t_i by reversing all the bits "1".
  - If for each $k \in \{1, \ldots, r\}$, $\sum_{j=1 \wedge t_i \in T_j}^{n} p_k^j \geq d_k^i$, t_i can be satisfied.
    - Check the availability of agents for C_i according to the constraints of resources and interest sets.
      - For each bit "1", if it has no interest in t_i or has no required resource, kick this agent out of C_i by reversing the bit.
      - For each bit "0", if it is interested in t_i and has available resources, let this agent join C_i by reversing the bit.
    - Create a resource allocation plan for C_i by repeating the following operations until all the bits "1" are examined.
      - Choose randomly an unexamined bit "1" in row i from a uniform distribution.
      - If t_i has been satisfied, this agent is dispensable for C_i. Thus, drop this agent out of C_i by reversing the bit and give it more opportunities to associate itself with other tasks.

Algorithm 7 The heuristic for solution repair.

Require: a solution

for each $i \in \{1, \ldots, m\}$ **do**
  $\mathbf{E}_i \leftarrow 0$;
end for
for each $j \in \{1, \ldots, n\}$ **do**
  $\mathbf{P}_j \leftarrow \mathbf{B}_j$;
end for
while not all the rows are checked **do**
  pick randomly an unexamined row i;
  if $\exists k \in \{1, \ldots, r\}, \sum_{j=1 \wedge t_i \in T_j}^{n} p_k^j < d_k^i$ **then**
    for each $j \in \{1, \ldots, n\}$ **do**
      if $\delta_{ij} = 1$ **then**
        $\delta_{ij} \leftarrow 0$;
      end if
    end for
  else
    for each $j \in \{1, \ldots, n\}$ **do**
      if $\delta_{ij} = 1$ but $t_i \ni T_j$ or for each $k \in \{1, \ldots, r\}$, $\min\{d_k^i, p_k^j\} = 0$ **then**
        $\delta_{ij} \leftarrow 0$;
      end if
      if $\delta_{ij} = 0$ but $t_i \in T_j$ and $\exists k \in \{1, \ldots, r\}, \min\{d_k^i, p_k^j\} > 0$ **then**
        $\delta_{ij} \leftarrow 1$;
      end if
    end for
    while not all the bits "1" are inspected **do**
      choose randomly an unchecked bit "1";
      if each $k \in \{1, \ldots, r\}$, $\min\{d_k^i - e_k^i, p_k^j\} = 0$ **then**
        $\delta_{ij} \leftarrow 0$;
      else
        for each $k \in \{1, \ldots, r\}$ **do**
          $w_k^{ji} \leftarrow \min\{d_k^i - e_k^i, p_k^j\}$;
          $p_k^j \leftarrow p_k^j - w_k^{ji}$;
          $e_k^i \leftarrow e_k^i + w_k^{ji}$;
        end for
      end if
    end while
  end if
end while
decode the solution to generate an OCS according to Algorithm 6;
return a feasible OCS;

· If t_i is not achieved, this agent is necessary for t_i. Allocate its resources to C_i according to the principle of maximum individual contribution: for each $k \in \{1, \ldots, r\}$,

$$w_k^{ji} \leftarrow \min\{d_k^i - e_k^i, p_k^j\} \tag{6}$$

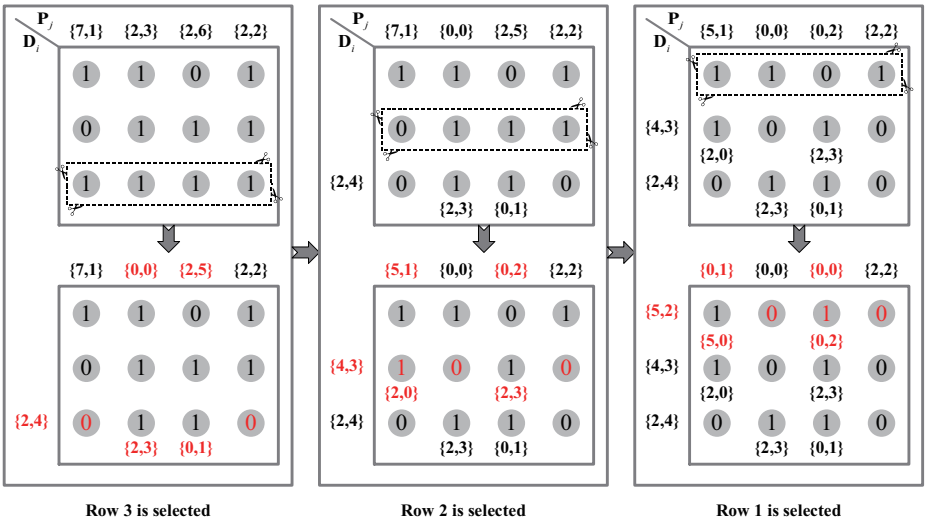


Figure 5: The corresponding revision procedure for a 2D solution of the given instance in Example 4.

Then, update $\mathbf{P}_j$ of this agent and $\mathbf{E}_i$ of C_i: for each $k \in \{1, \ldots, r\}$,

$$p_k^j \leftarrow p_k^j - w_k^{ji} \tag{7}$$

$$e_k^i \leftarrow e_k^i + w_k^{ji} \tag{8}$$

- Generate a feasible OCS according to Algorithm 6.

Example 4. *Consider the following instance.*

$$T = \{t_1, t_2, t_3\}, A = \{a_1, a_2, a_3, a_4\}$$
$$T_1 = \{t_1\}, T_2 = \{t_1, t_2, t_3\}, T_3 = \{t_2, t_3\}, T_4 = \{t_1\}$$
$$\mathbf{B}_1 = [3, 2], \mathbf{B}_2 = [6, 3], \mathbf{B}_3 = [2, 6], \mathbf{B}_4 = [2, 2]$$
$$\mathbf{D}_1 = [5, 2], \mathbf{D}_2 = [4, 3], \mathbf{D}_3 = [2, 4]$$
$$\mu_1 = 7, \mu_2 = 7, \mu_3 = 6$$

For a visual understanding how the proposed heuristic works, in Figure 5 we show the detailed revision procedure for a 2D solution derived from the above instance. Specifically, row 3 is first selected randomly. t_3 *can be satisfied by the residual resources of* a_2 *and* a_3*. In row 3,* a_1 *is not interested in* t_3*, so* $\delta_{31} \leftarrow 0$*. Then, to achieve* t_3*,* a_2 *is first selected to contribute* $\mathbf{W}_{23} = [2, 3]$ *and* a_3 *is successively selected to contribute* $\mathbf{W}_{33} = [0, 1]$*. Afterwards, row 2 is randomly chosen.* t_2 *can be satisfied by the residual resources of* a_2 *and* a_3*. Note that* a_4 *is not interested in* t_2 *and thus* $\delta_{24} \leftarrow 0$*. In row 2, to accomplish* t_2*,* a_3 *is first selected to contribute* $\mathbf{W}_{32} = [2, 3]$ *and* a_2 *is successively selected to contribute* $\mathbf{W}_{22} = [2, 0]$*. Finally, only row 1 needs to be checked.* t_1 *can be satisfied by the residual resources of* a_1*,* a_2*, and* a_4*. However,* a_3 *is not interested in* t_1*, so* $\delta_{13} \leftarrow 0$*.* a_2 *has available resources for* t_1*, and thus* $\delta_{12} \leftarrow 1$*. In row 1, to complete* t_1*,* a_2 *is first selected to contribute* $\mathbf{W}_{21} = [2, 0]$ *and* a_1 *is then selected to contribute* $\mathbf{W}_{11} = [3, 2]$*. Since* t_1 *has been satisfied,* a_4 *is unnecessary for* t_1 *and* $\delta_{14} \leftarrow 0$*. After the*

Algorithm 8 The proposed EAF for solving the OCSGP in task-based settings.

Require: an OCSGP instance and the employed EA
create randomly a group of initial solutions;
check and repair each solution based on Algorithms 6 and 7;
evaluate the fitness of each amended solution according to Equations (3) and (4);
while stopping criterion is not met **do**
  create new solutions by exploitation and exploration steps of the employed EA
  check and repair each new solution with Algorithms 6 and 7;
  evaluate the fitness of each new solution according to Equations (3) and (4);
  replace the worst solutions by the best new ones;
end while
return the best found OCS^*;

repair, we travel each row and each column and can obtain that $(C_1 = \{a_1, a_2\}, V_1 = \{t_1\})$, $(C_2 = \{a_2, a_3\}, V_2 = \{t_2\})$, $(C_3 = \{a_2, a_3\}, V_3 = \{t_3\})$, *and* $C' = \{a_4\}$. *Therefore, the generated* $OCS = \{(\{a_1, a_2\}, \{t_1\}), (\{a_2, a_3\}, \{t_2, t_3\}), (\{a_4\}, \emptyset)\}$, *in which the former two overlapping coalitions are feasible. The value of* OCS *is* $7 + 7 + 6 = 13$. *Clearly, the generated* OCS *is an optimal* OCS^*.

It can be observed that in Algorithm 7, each task t_i is checked and assigned separately. Whether t_i is achievable depends directly on whether the current constraints of resources and interest sets can be obeyed. Once the remaining, available resources are insufficient, t_i will be abandoned. Additionally, if t_i can be satisfied, only the agents that are interested in t_i and have the available resources can be selected to join C_i. The above mechanism can effectively prevent the potential resource conflicts within and between C_i and make each non-empty (C_i, V_i) certainly feasible. Moreover, each selected agent a_j does whatever it can to contribute its residual resources to t_i. Once t_i is satisfied, those unemployed agents will be excluded from C_i to compete for other tasks that they have interest in. The purpose is to make other tasks as achievable as possible under the premise of ensuring the feasibility of (C_i, V_i) to maximize utilization of the available resources as well as the social welfare. Obviously, Algorithm 7 looks like a local improvement strategy and can make the generated OCS as good as possible. This is because in each generated OCS, there is no unassigned task that can be achieved by the residual resources of agents.

To further verify the above distinguishing features, we now theoretically analyze the efficiency and serviceability of Algorithm 7. We have the following results.

Proposition 17. *The worst case complexity of Algorithm 7 is* $O(n^3)$.

Proof. See Appendix B. □

Proposition 18. *As long as* T *contains at least a task that can be achieved by agents in* A, *the pending solution will be repaired to generate a feasible OCS by Algorithm 7.*

Proof. See Appendix B. □

6.3 The Procedure of EAF

The proposed EAF for solving the OCSGP in task-based settings is listed in Algorithm 8, where Algorithms 6 and 7 serve as crucial guarantees for the EAF to identify feasible OCSs.

Specifically, EAF starts with the random initialization of a group of solutions. Next, check and amend each solution to make them as feasible as possible according to Algorithms 6 and 7. Then, compute the fitness of solutions according to Equations (3) and (4). In each generation, the original solutions are utilized to create a group of new solutions by the exploitation and exploration steps of the employed EA; these newly-obtained solutions require feasibility repair based on Algorithms 6 and 7, followed by evaluation in accordance with Equations (3) and (4); afterwards, replace the worst solutions by the best new ones. Finally, the best found solution is returned as the final output.

Proposition 19. *The worst case complexity of Algorithm 8 is* $O(n^4)$.

Proof. See Appendix B. □

It should be pointed out that EAF cannot guarantee to find an optimal solution in quartic polynomial time, as EAF is a kind of metaheuristic optimization framework.

7. Experimental Methodology

In this section, we introduce the experimental methodology for assessing the performance of our EAF (i.e., Algorithms 6-8).

7.1 Research Questions

Our experiments aim to answer the following four research questions:

- Research Question 1 (RQ1): Is it possible for Algorithm 7 to enhance the search performance of basic EAs when applied to OCSGP?
- Research Question 2 (RQ2): Can our EAF surpass the performance of the current state-of-the-art hybrid algorithm for OCSGP?
- Research Question 3 (RQ3): Is our EAF's performance influenced by the size of the instance?
- Research Question 4 (RQ4): Can our EAF be effectively utilized in other coalition-based application problems?

We use RQ1 to show whether Algorithm 7 can generate a feasible OCS and greatly enhance the efficiency of EAs on OCSGP. We ask RQ2 to determine whether our EAF can provide better results than the existing hybrid algorithm for OCSGP. We investigate RQ3 to verify whether our EAF is sensitive to the size of the instance. We address RQ4 by examining the applicability of our EAF in the pursuit-evasion domain (Lopez et al., 2020; Pan & Yuan, 2023).

7.2 Compared Methods

To our best knowledge, most of the existing works on OCFGs are concentrating on analyzing and forming overlapping coalitions rather than solving OCSGP. To comprehensively assess the performance of the proposed EAF, we follow the previous practice in (Su et al., 2020; Zhang et al., 2020) and adopt three very basic binary EAs, namely, genetic algorithm (GA) (Goldberg, 1989), BPSO (Kennedy & Eberhart, 1997), and BDE (Pampara et al., 2006). The reason for that is 1) we want to show that even with very basic EAs, the proposed

framework performs better than its competitors, and 2) the improvement obtained does not come from the EA part (as very basic EAs are used) but indeed the proposed framework.

To address RQ1, we integrate Algorithm 7 and TOH (Zhang et al., 2020) with GA, BPSO, and BDE, respectively. As mentioned earlier, TOH is a task-oriented heuristic combined with EAs to effectively find the approximately optimal OCS for the given tasks. We want to check whether Algorithm 7 can adapt to the changes of EAs. For ease of description, TOH indicates a family of TOH+GA, TOH+BPSO, and TOH+BDE; EAs represent a family of GA, BPSO, and BDE; and EAF denotes a family of EAF+GA, EAF+BPSO, and EAF+BDE.

To address RQ2, we compare EAF with a state-of-the-art hybrid algorithm DPG (Zhan et al., 2012), which is a combination of dynamic programming and greedy approach and is effective to solve the task-related OCSGP in TTGs. We use DPG to highlight the effectiveness of EAF.

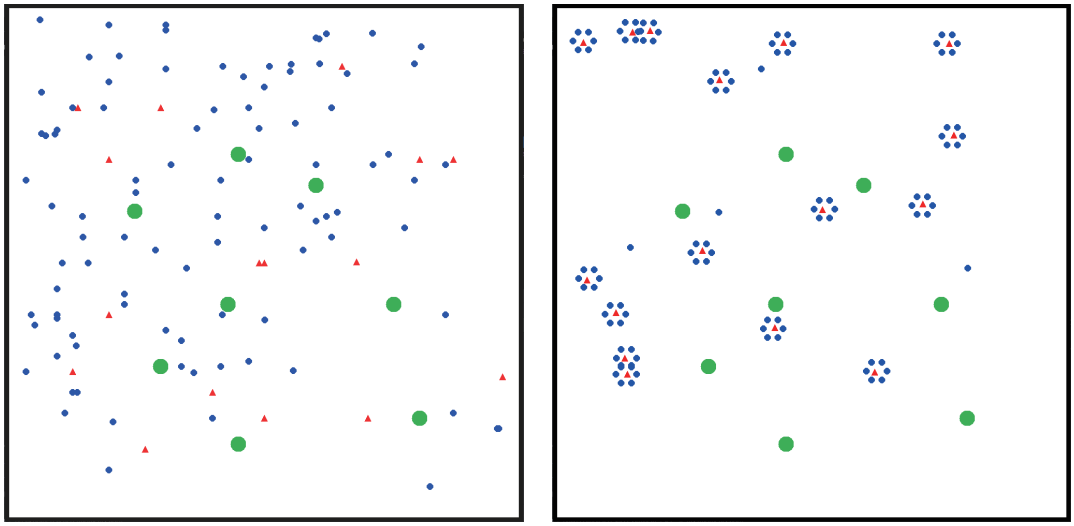
To address RQ3, we examine EAF on various instances with different numbers of agents, tasks, and resources. We use those instances to comprehensively evaluate the stability of EAF.

To address RQ4, we introduce the multi-agent pursuit-evasion problem, which is a widely studied benchmark in artificial intelligence and swarm robotics (Lopez et al., 2020; Pan & Yuan, 2023). In this domain, agents can represent autonomous mobile robots, unmanned air vehicles, spacecraft, wireless sensors, and more. As illustrated in Figure 6, the core issue in this domain is the development of coordination mechanisms that enable multiple pursuers to cooperatively capture multiple evaders in a typically discrete grid world with obstacles. Specifically, the primary challenge is to assign pursuers to evaders in order to minimize the total number of steps or the total time required to capture all the evaders successfully. This scenario is clearly task-driven, with each pursuer representing an agent and each evader being regarded as a task. The key concerns are, therefore, the formation of coalitions and the allocation of tasks to each coalition. Therefore, we employ our EAF+GA to explore the optimal task allocation scheme (i.e., the optimal coalition structure), which minimizes the longest distance between pursuers and evaders. Additionally, we compare EAF+GA with the state-of-the-art shortest distance first (SDF) algorithm (Lopez et al., 2020; Pan & Yuan, 2023; Zhou & Chen, 2024) and shortest capture time first (SCTF) algorithm (Li et al., 2024; Zhou et al., 2018) for forming pursuit coalitions. In SDF, each agent selects the nearest evader to pursue. SCTF estimates the capture time of all evaders using the fast marching method (Adalsteinsson & Sethian, 1995; Wu & Liu, 2010) and assigns agents with the shorter arrival time to sequentially track the evader with the shortest capture time. Clearly, SDF and SPTF adopt different perspectives, focusing on agents and tasks, respectively.

All the codes of the compared methods were written in C++ and run on a computer with Intel Xeon E5 2.20 GHz CPU, 32.0 GB of RAM, and Windows Server 2012.

7.3 Parameter Settings

We adopted the recommended parameter settings in (Liu et al., 2016; Su et al., 2020; Zhang et al., 2020). Specifically, the maximum number of fitness evaluation is 1,500. In GA, the crossover rate is 0.9 and the mutation rate is 0.1. For BPSO, both of the two learning factors are set to 2.0 and the maximum velocity is restricted to 5.0. As for BDE, the crossover probability is 0.25 and the scaling factor is 1.0.



(a) (b)

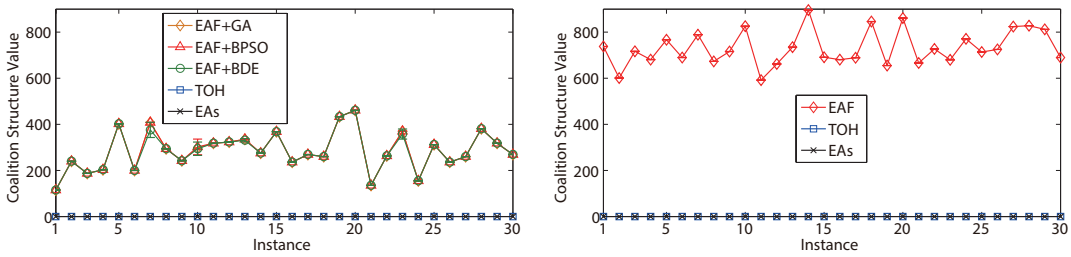
Figure 6: The schematic diagram of the multi-agent pursuit-evasion problem, in which the red triangle denotes an evader, the small blue circle represents a pursuer, and the large green circle indicates an obstacle. (a). The original state. (b). The final pursuit result.

7.4 Test Instances

To address RQ1, RQ2, and RQ3, we do all the tests in two specific task-based settings. One is the well-resourced setting: agents in A have enough resources to easily accomplish tasks in T. The other is the poorly resourced setting: agents' resources are insufficient and conflicts may often occur over the rare, but highly demanded resources between competitive tasks. It is worth remarking that the CSG is a fundamental algorithmic problem within coalitional games, rather than a specific application-based problem. Consequently, there is a scarcity of real-world data available. Therefore, we generated test instances via simulations of some class of OCSGP. Following the existing work, we generated at total 150 different instances randomly from a normal distribution in the above two settings on the basis of $n \in [10, 100]$, $m \in [6, 24]$, $r \in [1, 24]$, $b_k^j \in [0, 300]$, $d_k^i \in [0, 450]$, and $\mu_i \in [50, 100]$. It is expected that we can simulate as many resource conflicts over overlapping coalitions as possible, so as to more fully verify the effectiveness of our method. This is also the most prevalent approach for generating test instances in studying algorithms for resource-constrained coalition structure generation (Dang & Jennings, 2006; Dunne et al., 2010; Shehory & Kraus, 1998; Service & Adams, 2011a; Zhan et al., 2012; Zhang et al., 2011, 2020). Note that each instance was tested on each compared method repeatedly for 30 independent runs under diverse random seeds to enable statistical analysis.

To address RQ4, we analyze the algorithmic performance across different numbers of evaders. For each number of evaders, we randomly create 10 instances, each corresponding to a distinct initial location. Moreover, each instance was repeatedly tested on each compared algorithm for 30 independent runs.

Admittedly, the utilized instances do not encompass all the possible OCSGPs in task-based environments. It is essential to highlight that our objective in this work is not to thoroughly confirm our hypothesis in every possible scenario, but rather to examine whether



(a) (b)

Figure 7: The coalition structure value (mean and standard deviation) attained by different EAs combined with different heuristics under different instances. (a). In the poorly resourced setting. (b). In the well-resourced setting.

it holds true for the representative cases of OCSGP. We hope that our research can act as a launching pad to stimulate discussions within the coalition structure generation community regarding this crucial issue. Building on this foundation, future studies can expand upon our work to cover additional OCSGPs that we have not yet addressed.

8. Results and Analysis

In this section, we present the experimental results and address the research questions raised in Section 7.1. To ensure verifiability, we provide the source codes of the compared methods and the raw instances at the following link: `https://github.com/zgfhfut/OCSG`.

It should be noted that in this study, the number of potential OCSs grows exponentially with the number of agents and tasks. In the poorly resourced setting, there are intense resource conflicts within coalitions. It is impractical to employ a brute-force method (e.g., exhaustive search) to discover the optimal solution. Fortunately, in well-resourced settings, each task can be easily fulfilled by agents. Therefore, we can ascertain whether the obtained OCS is an optimal solution by examining the value of the OCS, as the maximum OCS value is merely the sum of the rewards for all the tasks.

8.1 RQ1: Comparing Different Heuristics

Algorithm 7 in the proposed EAF determines whether the generated OCS is feasible and plays a key role in the search course of EAF. Thus, in the first experiment, we compared EAF with TOH (Zhang et al., 2020) and the basic GA, BPSO, and BDE without heuristic. We randomly generated 60 different test instances in the two settings according to $n = 20$, $m = 10$, and $r = 5$. Notice that in this setting, there are nearly 2^{200} possible OCSs for each instance.

Figure 7 depicts the coalition structure value (mean and standard deviation) attained by different EAs combined with different heuristics under different instances of the two settings. It can be observed that both TOH and EAs failed in finding a feasible OCS on all the 60 instances in the two settings. This is because EAs have no heuristic to handle Constraint Sets (1) and (2); although TOH can handle the constraints of resources, it can do nothing about the constraints of the interest set. The above results imply that no matter how powerful TOH and EAs are, it is difficult for them to find a feasible OCS without leveraging the problem's knowledge. Conversely, EAF found feasible OCSs on each instance

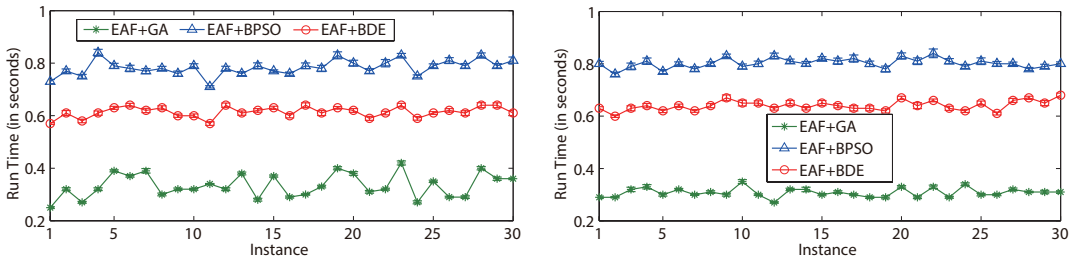


Figure 8: The run time (mean and standard error, in seconds) consumed by EAF under different instances. (a). In the poorly resourced setting. (b). In the well-resourced setting.

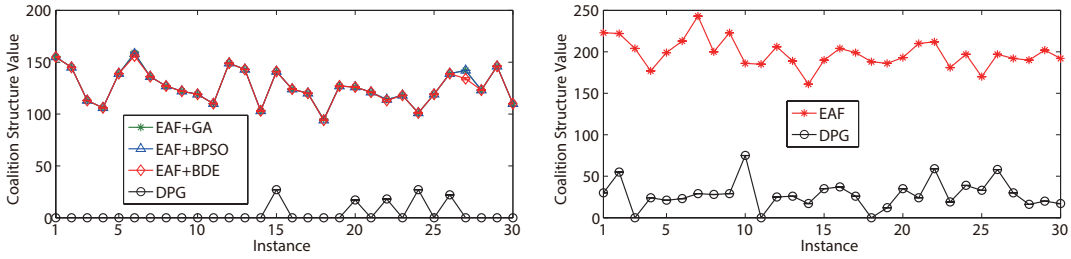
in each setting with very low standard deviation. Additionally, as shown in Figure 7(a), there is minimal difference between EAF+GA, EAF+BPSO, and EAF+BDE. This suggests that our framework is compatible with various basic binary EAs, and the enhancement is primarily due to our framework rather than the used EAs. Surprisingly, EAF found the optimal solution in the well-resourced setting, because all the 10 tasks were performed according to the value of the final OCS. This is why there is no standard deviation in Figure 7(b). The above observations also indicate that EAF has good stability and robustness. An explanation is that EAF not only can make each generated OCS feasible by Algorithm 7, but also can make $v(OCS)$ as maximal as possible for the reason that Algorithm 7 is able to maximize the utilization of agents' resources.

Figure 8 pictures the run time (mean and standard error, in seconds) consumed by EAF under different instances of the two settings. It can be seen that EAF consumed less than 1s on each instance, with low standard error in both the two settings. This observation suggests that EAF is very efficient even in the case of a huge solution space.

8.2 RQ2: Comparing with Hybrid Algorithms

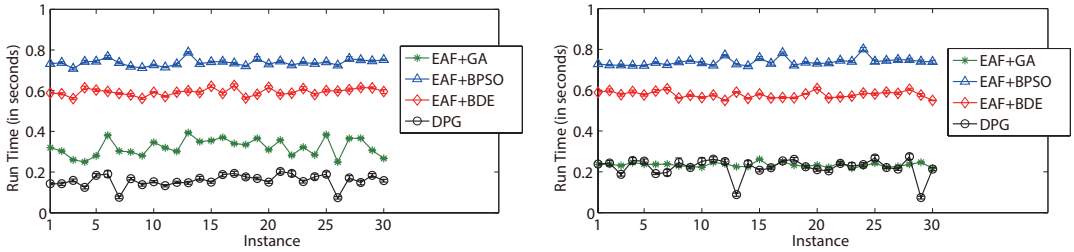
In this experiment, we compared EAF with the existing DPG (Zhan et al., 2012) for solving OCSGP in TTGs, where there is only a type of resources. We randomly generated 60 different test instances of the two settings in a case of $n = 20$, $m = 10$, and $r = 1$.

Figure 9 illustrates the coalition structure value (mean and standard deviation) obtained by EAF and DPG under different instances in the two settings. As can be seen, DPG often failed in finding a feasible OCS even in the well-resourced setting, implying that DPG is unreliable. On the contrary, EAF found feasible OCSs on each instance with significantly better coalition structure values. A possible reason is that DPG does not care about the interest set constraint, and thus the selected agents may not be interested in the assigned task. On the other side, Algorithm 7 in EAF is able to help EAF solve all the potential constraint violations, which greatly improves the search performance of EAF. It can also be observed that, much like Figure 7(a), in Figure 9(a), the disparity between EAF+GA, EAF+BPSO, and EAF+BDE is barely noticeable, suggesting that the enhancement is not due to the employed EAs, but rather stems from the proposed framework. Note that in Figure 9(b), EAF found the optimal solution on each instance for the reason that all the 10 tasks have been completed in terms of the value of the attained OCS.



(a) (b)

Figure 9: The coalition structure value (mean and standard deviation) obtained by EAF and DPG under different instances. (a). In the poorly resourced setting. (b). In the well-resourced setting.



(a) (b)

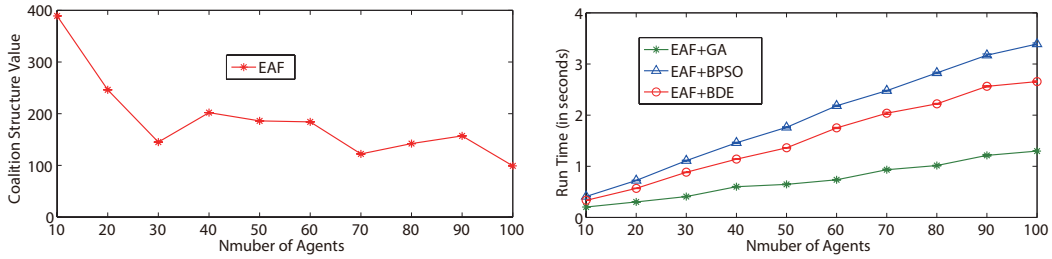
Figure 10: The run time (mean and standard error, in seconds) consumed by EAF and DPG under different instances. (a). In the poorly resourced setting. (b). In the well-resourced setting.

Figure 10 depicts the run time (mean and standard error, in seconds) consumed by EAF and DPG under different instances of the two settings. It can be observed that both EAF and DPG consumed less than 1s and are efficient for solving OCSGP. This is because there is only a type of resources in each instance.

8.3 RQ3: Impact of the Problem Size

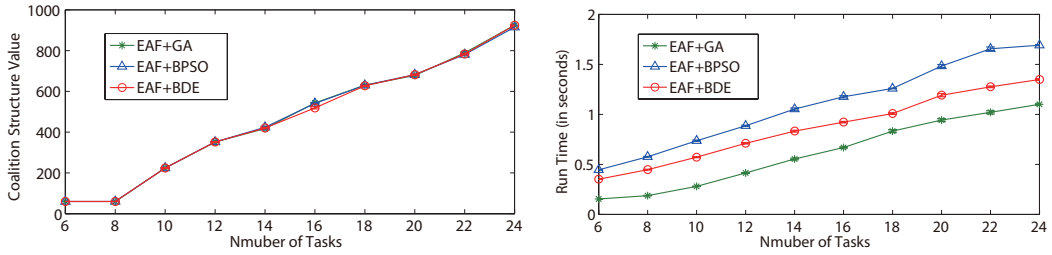
The results of the previous two experiments demonstrate that EAF can find the optimal solution efficiently in the well-resourced setting. Therefore, to further examine the adaptability of EAF to the problem size, in the third experiment, we compared EAF+GA, EAF+BPSO, and EAF+BDE in the poorly resourced setting under different n, m, and r. We want to know whether EAF is still able to work under the tight resource constraints and the huge solution space.

Figure 11 shows the results obtained by EAF in the poorly resourced setting under different n, $m = 10$, and $r = 5$. Notice that there are about 2^{100} and 2^{1000} OCSs when $n = 10$ and $n = 100$, respectively. As can be observed in Figure 11(a), with the increase of n, the coalition structure values obtained by EAF showed a decline. An explanation is that each $\mathbf{B}_j$ decreases with the increase of n to obey the poorly resourced setting, and thus it is increasingly difficult for agents to meet the needs of tasks. However, surprisingly, EAF could still find a feasible OCS even when $n = 100$. On the other hand, in Figure 11(b),



(a) (b)

Figure 11: The results obtained by EAF in the poorly resourced setting under different n. (a). The coalition structure value (mean and standard deviation). (b). The run time (mean and standard error, in seconds).



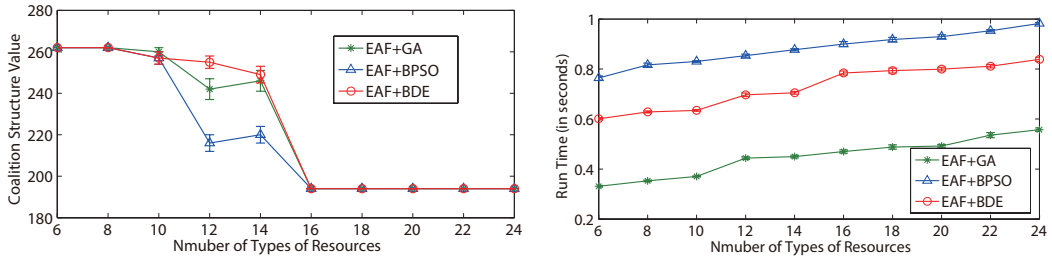
(a) (b)

Figure 12: The results obtained by EAF in the poorly resourced setting under different m. (a). The coalition structure value (mean and standard deviation). (b). The run time (mean and standard error, in seconds).

the time consumption of EAF rose slowly with the increase of n. Note that when $n = 100$, EAF consumed less than 4 s, which is perfectly acceptable from the perspective of the huge solution space. Additionally, it can also be seen that the standard deviation of coalition structure value and the standard error of run time are very low. This observation implies that EAF is stable with the increase of n.

Figure 12 pictures the results obtained by EAF in the poorly resourced setting under different m, $n = 20$, and $r = 5$. In this setting, there are about 2^{120} and 2^{480} OCSs when $m = 6$ and $m = 24$, respectively. As can be seen from Figure 12(a), with the increase of m, the coalition structure values obtained by EAF rose sharply. The reason is that each $\mathbf{D}_i$ decreases with the increase of m in the poorly resourced setting, and thus it is increasingly easy for tasks to be satisfied by agents. Note that EAF could find a feasible OCS even when $m = 24$. In Figure 12(b), the run time of EAF increased slowly with the growth of m. When $m = 24$, EAF consumed less than 2s. In addition, it can also be found that the standard deviation of coalition structure value and the standard error of run time are very low. The above results indicate that EAF still keeps stable with the increase of m.

Figure 13 illustrates the results obtained by EAF in the poorly resourced setting under different r, $n = 20$, and $m = 10$. In this setting, the number of possible OCSs is 2^{200}. It can be observed in Figure 13(a), with the increase of r, the coalition structure values obtained by EAF showed a downward trend on the whole. This is because the growth of r increases the possibility of potential resource conflicts in the poorly resourced setting,



(a) (b)

Figure 13: The results obtained by EAF in the poorly resourced setting under different r. (a). The coalition structure value (mean and standard deviation). (b). The run time (mean and standard error, in seconds).

and it is increasingly difficult for tasks to be satisfied by agents with their finite resources. Note that EAF could still find a feasible OCS even when $r = 24$. In Figure 13(b), the time consumption of EAF rose slowly with the increase of r. When $r = 24$, EAF consumed less than 1s. Additionally, it can also be found that the standard deviation of coalition structure value and the standard error of run time are low. The above results demonstrate that EAF is also stable with the increase of r.

In brief, EAF exhibits high effectiveness, efficiency, and stability from the empirical results on different n, m, and r in the poorly resourced setting. This suggests that EAF is not sensitive to the problem size, and verifies the theoretical results in Propositions 17 and 19.

8.4 RQ4: Application to the Pursuit-Evasion Problem

Figure 14 depicts the longest distance (mean and standard deviation) between pursuers and evaders obtained by the three algorithms under various numbers of evaders. It can be observed that for different numbers of evaders, the EAF+GA algorithm achieves the shortest distance compared to SDF and SCTF, indicating that EAF+GA can significantly reduce the movement distance of pursuers to their corresponding evaders. Specifically, when there are a small number of evaders, the difference in performance between the three algorithms is not pronounced. This is because with a small number of evaders, there are relatively few possible combinations of task allocations, and the allocation strategies produced by each algorithm do not demonstrate significant differences in effectiveness. However, when there are a large number of evaders, the possibility of task allocations increases progressively, and the superiority of our EAF+GA becomes increasingly evident. Furthermore, the size of the standard deviation gradually decreases as the number of evaders increases. This is because as the number of evaders grows, the distribution of pursuers in the confined grid environment becomes denser, and the number of pursuers in each area approaches equilibrium. Accordingly, the longest distance achieved by each algorithm on different instances tends to converge.

Figure 15 illustrates the number of total steps (mean and standard deviation) taken by the three algorithms for capturing all the evaders under different instances on the cases of 8 and 16 evaders, respectively. It can be observed that EAF+GA took significantly less steps than SDF and SCTF on each instance. Specifically, EAF+GA saves at least 22.2% and 17.5% steps on average under 8 and 16 evaders, respectively. The above findings

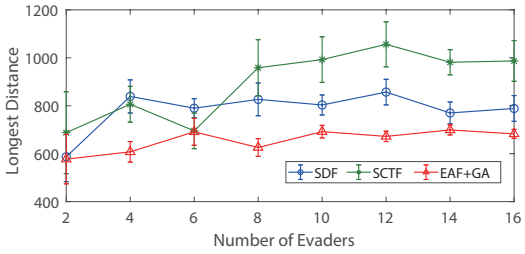


Figure 14: The longest distance (mean and standard deviation) between pursuers and evaders obtained by the three algorithms under different numbers of evaders.

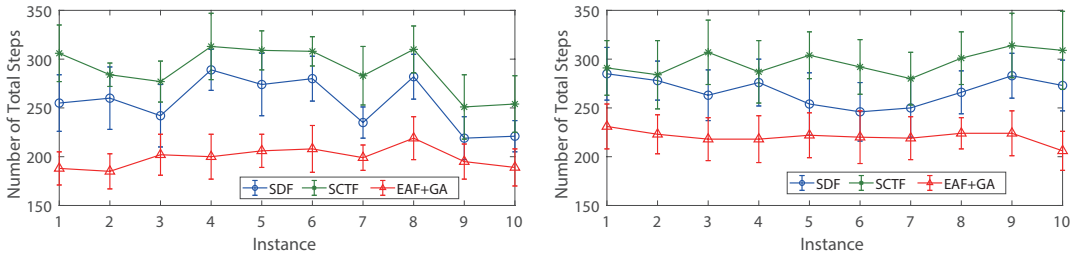


Figure 15: The number of total steps (mean and standard deviation) taken by the three algorithms for capturing all the evaders under different instances. (a). 8 evaders. (b). 16 evaders.

suggest that our EAF+GA is both efficient and effective in task-oriented scenarios, capable of assigning the appropriate tasks to the relevant agents. The reason for this lies in the EAF+GA's comprehensive evaluation and selection of potential combinations of coalitions and tasks through coalition structure generation, along with a task allocation strategy that takes into account the coordination among coalition members, thereby enhancing capture efficiency. In contrast, both SDF and SCTF are fundamentally greedy algorithms that assign tasks by minimizing the distance or travel time between pursuers and evaders. The mechanism of greedy behavior results in choices aimed at maximizing individual interests, without considering team collaboration. Consequently, the formed distribution strategy is highly uneven, leading to some pursuer coalitions quickly reaching the vicinity of their evaders, while others need to cover a longer distance to reach the vicinity of the evaders they are pursuing. This lack of coordination among team members cannot guarantee that the structure of the pursuer coalitions will maximize their benefits, ultimately leading to a decline in overall capture efficiency.

8.5 Discussion

In this subsection, we discuss the implications of the experimental results on the four research questions and explore the possible reasons for the effectiveness.

Regarding the first research question, which examines the benefits of Algorithm 7 to EAs for addressing OCSGP, the results reveal a surprising outcome. With the assistance of Algorithm 7, EAs demonstrate more stable and efficient performance on the 60 instances

in comparison to TOH. Furthermore, each basic EA is able to find the optimal solution in the well-resourced setting.

The second research question investigates whether EAF surpasses the existing DPG in solving OCSGP. The results are encouraging, as EAF demonstrates superiority in solving OCSGP and is able to find the optimal solution in the well-resourced setting. In contrast, despite having only one type of resource in the 60 instances, DPG struggles to find a feasible OCS even in the well-resourced setting.

The third research question addresses whether EAF is susceptible to the size of the instance in the poorly resourced setting. As expected, when the number of agents, tasks, and resources increases, EAF remains stable in exploring feasible OCSs, with a minimal increase in time consumption.

The fourth research question concerns the applicability of EAF to the multi-agent pursuit-evasion problem. Unlike the compared SDF and SCTF, EAF concurrently considers tasks and agents and pursues the overall reward of the coalition structure. This enables EAF to identify better pursuer coalitions with higher capture efficiency.

The mentioned enhancements are due to the fact that our EAF can directly employ the polynomial-time Algorithm 7 for repairing solutions to address resource conflicts between competing coalitions. Consequently, EAF is able to consistently explore the feasible region and generate feasible OCSs more efficiently, increasing the likelihood of discovering OCSs with higher values.

9. Conclusion

This work discusses the overlapping colition structure generation problem (OCSGP) from the perspective of task-based settings. This approach is novel as it considers agents with finite resources that can only respond to parts of tasks, and coalitions that take on mutually disjoint subsets of tasks. We have analyzed the size of the search space and the computational complexity of the proposed OCSGP. Specifically, we show that checking whether a pending overlapping coalition structure (OCS) is feasible can be solved in polynomial time. However, finding the optimal OCS remains impractical within polynomial time concerning the number of agents and tasks. To tackle the intractability, we developed a generic evolutionary algorithm framework (EAF) designed to solve the proposed OCSGP efficiently. This EAF can be combined with any binary evolutionary algorithms. A key feature of our EAF is an embedded solution-repair heuristic with cubic time complexity, which ensures feasible OCS generation by assigning tasks based on the residual resources of agents. This mechanism effectively prevents resource conflicts between competitive overlapping coalitions. We compared our EAF with a task-oriented heuristic and a hybrid algorithm for OCSGP. Additionally, we examined its applicability in a pursuit-evasion problem. The experimental results demonstrate the high competitiveness and efficiency of our proposed EAF, showing its capability to find approximately optimal solutions in quartic polynomial time relative to the instance size.

In our future research, we are committed to further refining our theoretical foundations by focusing on deriving tighter bounds on the performance of our EAF, particularly in terms of solution quality and convergence rates. We will analyze how different parameters and configurations impact the efficiency and effectiveness of the EAF and extend our complexity analysis to cover more general cases, exploring how varying the number of agents, tasks, and resource constraints affects computational feasibility. Additionally, we will investigate the robustness of our approach in dynamic environments where agent capabilities and task

requirements may change over time. To extend the empirical validation of our method, we will explore its applicability in diverse domains such as disaster response, logistics optimization, and healthcare settings. Comprehensive comparative studies will evaluate our EAF against a wider array of existing methods and benchmarks to better understand its strengths and limitations. Furthermore, we plan to apply our method to real-world datasets and case studies to assess its practical utility and identify areas for improvement.

Acknowledgments

The authors wish to thank the referees for their valuable comments and constructive suggestions which lead to the significant improvement of this paper. This work was supported in part by the MOE (Ministry of Education in China) Project of Humanities and Social Sciences under Grant 24YJA870011, in part by the Anhui Provincial Natural Science Foundation under Grant 2208085MF166, and in part by the Fundamental Research Funds for the Central Universities under Grants PA2023IISL0097 and PA2023GDSK0049.

Appendix A. Notational Conventions

A	The set of agents. Members of the set are denoted by a_j and subsets of A are denoted by C, C_i, etc.
T	The set of tasks. Members are denoted by t_i and subsets of T are denoted by V, V_l, etc.
u_i	The reward rendered by task t_i.
T_j	The interest set of agent a_j.
$\mathbf{B}_j$	The resource vector of agent a_j. Members of the vector are denoted by b_k^j.
$\mathbf{D}_i$	The demand vector of task t_i. Members of the vector are denoted by d_k^i.
$\mathbf{W}_{ji}$	The contribution vector of agent a_j for task t_i. Members of the vector are denoted by w_k^{ji}.
$\mathbf{P}_j$	The residual vector of agent a_j. Members of the vector are denoted by p_k^j.
$\mathbf{E}_i$	The investment vector of agents in C_i. Members of the vector are denoted by e_k^i.
(C, V)	The task-based coalition.
OCS	The task-based overlapping coalition structure.
$v(C_l, V_l)$	The value of (C_l, V_l).
$v(OCS)$	The value of OCS.

Appendix B. Proofs

To maintain readability in the main text, we have excluded some detailed proofs. We present these omitted proofs below for completeness.

Proposition 4. *For A, the total number of possible cover $(C_1, \ldots, C_s)$ with size s is $\frac{(2^n-1)!}{(2^n-1-s)!}$.*

Proof. It is known that there are $2^n - 1$ non-empty subsets of A. In OCSGP, each agent may join these $2^n - 1$ subsets at the same time. For a cover $\{C_1, \ldots, C_s\}$ of A, without loss of generality, we assume that the first subset C_1 has $2^n - 1$ possibilities. Then, there are only $2^n - 1 - 1$ possibilities for the next subset C_2. The rest may be deduced by analogy and induction. Particularly, the last subset C_s has $2^n - 1 - (s-1)$ possibilities. To sum up,

the total number of possible cover $(C_1, \ldots, C_s)$ with size s is $(2^n - 1) \cdot (2^n - 1 - 1) \cdots (2^n - 1 - (s-1)) = \frac{(2^n-1)!}{(2^n-1-s)!}$. □

Proposition 5. *For T, the total number of possible partition subset $(V_1, \ldots, V_s)$ with size s is $\sum_{x=s}^{m+1} [\mathcal{C}_{m+1}^x \cdot S(x, s)]$.*

Proof. T includes m different tasks. Assume that $\emptyset$ is the $(m+1)$-th virtual task in T. We first select $x \geq s$ tasks from T, and the possible combinations are $\mathcal{C}_{m+1}^x$. Then, for each combination, we need to divide these x tasks into s parts disorderly, whose possibilities are known as the Stirling numbers of the Second Kind and denoted as $S(x, s)$. Therefore, the number of possible partition subset $(V_1, \ldots, V_s)$ for each x is $\mathcal{C}_{m+1}^x \cdot S(x, s)$. Note that $s \leq x \leq m + 1$. To sum up, the total number of possible partition subset $(V_1, \ldots, V_s)$ with size s is $\sum_{x=s}^{m+1} [\mathcal{C}_{m+1}^x \cdot S(x, s)]$. □

Proposition 6. *The total number of possible OCSs with size s is*

$$\frac{(2^n - 1)! \cdot \sum_{x=s}^{m+1} [\mathcal{C}_{m+1}^x \cdot S(x, s)]}{(2^n - 1 - s)!}$$

Proof. For an $OCS = \{(C_1, V_1), \ldots, (C_s, V_s)\}$, we know that $\{C_1, \ldots, C_s\}$ has $\frac{(2^n-1)!}{(2^n-1-s)!}$ possibilities (see Proposition 4), and $(V_1, \ldots, V_s)$ has

$$\sum_{x=s}^{m+1} [\mathcal{C}_{m+1}^x \cdot S(x, s)]$$

possibilities (see Proposition 5). Accordingly, $\{(C_1, V_1), \ldots, (C_s, V_s)\}$ has at total

$$\frac{(2^n - 1)! \cdot \sum_{x=s}^{m+1} [\mathcal{C}_{m+1}^x \cdot S(x, s)]}{(2^n - 1 - s)!}$$

possibilities. □

Proposition 7. *The total number of possible OCSs is*

$$|\prod\nolimits_A^T| = \sum\nolimits_{s=1}^{\min\{2^n-1, m+1\}} \frac{(2^n - 1)! \cdot \sum_{x=s}^{m+1} [\mathcal{C}_{m+1}^x \cdot S(x, s)]}{(2^n - 1 - s)!}$$

Proof. There are at most $2^n - 1$ nonempty subsets of A. It is clear that the collection of all the possible subsets of A is also the largest cover of A. Therefore, for each OCS with size s, we have $s \leq 2^n - 1$. From Proposition 3, we have $s \leq m + 1$. Consequently, we can obtain that $s \leq \min\{2^n - 1, m + 1\}$. Based on Proposition 6, the number of OCSs with size s is

$$\frac{(2^n - 1)! \cdot \sum_{x=s}^{m+1} [\mathcal{C}_{m+1}^x \cdot S(x, s)]}{(2^n - 1 - s)!}$$

Therefore, the total number of possible OCSs is

$$|\prod\nolimits_A^T| = \sum\nolimits_{s=1}^{\min\{2^n-1, m+1\}} \frac{(2^n - 1)! \cdot \sum_{x=s}^{m+1} [\mathcal{C}_{m+1}^x \cdot S(x, s)]}{(2^n - 1 - s)!}$$

□

Proposition 8. *The smallest number of OCSs that needs to be searched to establish a bound from the optimal OCS^* is at least $2^m - 1$.*

Proof. From Proposition 1, we obtain that the total number of possible (C_l, V_l) with $V_l \neq \emptyset$ is $(2^n - 1) \cdot (2^m - 1)$. Obviously, to determine the smallest number of OCSs to be searched to establish a bound from OCS^*, all these $(2^n-1)\cdot(2^m-1)$ coalitions should be considered. For each $y \in \{1, \ldots, n\}$, there are $\mathcal{C}_n^y$ subset C_l of size y of A and (2^m-1) non-empty subset V_l of T, so there are $\mathcal{C}_n^y \cdot (2^m-1)$ different (C_l, V_l) with $|C_l| = y$. Then, the number of agent occurrences in all these coalitions is $\sum_{y=1}^{n} [\mathcal{C}_n^y \cdot (2^m - 1) \cdot y] = (2^m - 1) \cdot \sum_{y=1}^{n} (y \cdot \mathcal{C}_n^y) = (2^m - 1) \cdot n \cdot 2^{n-1}$. Additionally, from Proposition 2, we know that the number of agent occurrences in each OCS is at most $n \cdot 2^{n-1}$. Therefore, we need to search through at least $\frac{(2^m-1)\cdot n \cdot 2^{n-1}}{n \cdot 2^{n-1}} = 2^m - 1$ different OCSs to cover all the possible (C_l, V_l). That is, the smallest number of OCSs that needs to be searched through to establish a bound from OCS^* is at least $2^m - 1$. □

Proposition 9. *The smallest number of OCSs that needs to be searched to establish a bound from the optimal OCS^* is $2^m - 1$.*

Proof. In Proposition 8, we demonstrated that searching at least $2^m - 1$ OCSs is necessary to establish a bound from the optimal OCS^*. Our next step is to construct a set of $2^m - 1$ OCSs such that searching through this set will yield a bound from the optimal. We will now outline the construction of such a set.

Consider the following set of coalition structures:

$$G = \{\{(A, V)\} \mid V \subseteq T, V \neq \emptyset\}$$

In other words, G consists of all the possible coalition structures $\{(A, V)\}$, where V is a non-empty subset of the task set T.

First, we will demonstrate that the number of coalition structures in G is precisely $2^m - 1$. Given that there are $2^m - 1$ non-empty subsets of T, G comprises $2^m - 1$ coalition structures.

Next, we will establish that after examining all the coalition structures in G, a bound from the optimal can be determined. Let

$$(C^*, V^*) = \arg\max_{C \in A, V \in T} v(C, V)$$

That is, coalition (C^*, V^*) possesses the maximum rewards among all the possible coalitions. Since $C^* \subseteq A$, in task-based settings, for a given V^*, we have $v(A, V^*) = v(C^*, V^*)$. Consider a coalition structure $\vec{OCS} = \{(A, V^*)\}$, we obtain $v(\vec{OCS}) = v(C^*, V^*)$.

Now, for any coalition structure $OCS = \{(C_1, V_1), \ldots, (C_s, V_s)\}$, we have

$$v(OCS) = \sum\nolimits_{l=1, V_l \neq \emptyset}^{s} v(C_l, V_l)$$

Keep in mind that given there are at most $2^n - 1$ subsets of A and m tasks in T, there are at most only $\min\{2^n - 1, m\}$ overlapping coalitions with a non-zero value in any coalition structure. Therefore, we can deduce that

$$v(OCS) \leq \min\{2^n - 1, m\} \cdot v(C^*, V^*) = \min\{2^n - 1, m\} \cdot v(\vec{OCS})$$

In turn, we obtain

$$v(OCS^*) \leq \min\{2^n - 1, m\} \cdot v(\vec{OCS})$$

Considering $\vec{OCS} \in G$, upon examining all the coalition structures in G, a bound $\min\{2^n - 1, m\}$ from the optimal OCS^* can be established. □

Proposition 10. *The problem of checking whether a disjoint coalition (C, V) in an OCS is feasible is in P.*

Proof. Since (C, V) is a disjoint coalition, we first check whether V is an empty set. If V is non-empty, we further check whether each agent in C is interested in V and each task in V is of interest to an agent in C. Obviously, the above tests can be done in polynomial time, because both $|C|$ and $|V|$ are bounded. Then, we need to decide whether a feasible resource allocation scheme $\mathbf{W}_{ji}$ can be found to satisfy Constraint Sets (1) and (2). Since $|C|$, $|V|$, and r are bounded, these tests can also be performed in polynomial time. In brief, the problem of checking whether a disjoint coalition (C, V) in an OCS is feasible can be solved in polynomial time. □

Proposition 11. *The worst case complexity of Algorithm 1 is $O(n^6)$.*

Proof. See Algorithm 1, it is clear that the computational complexity of the checking algorithm depends on the complexity of the used maximum flow algorithm. In Fig. 2, it can be observed that there are at most $(2 + n + m)$ nodes and $(n + n \cdot m + m)$ arcs in the network. Therefore, the computational complexity of the Edmonds-Karp algorithm is $O((2+n+m)(n+n\cdot m+m)^2)$ (Cormen et al., 1990). In addition, all the k types of resources should be examined for uncovering the potential resource conflicts. Therefore, the worst case complexity of Algorithm 1 is $O(k \cdot (2 + n + m)(n + n \cdot m + m)^2) = O(n^6)$, which is of polynomial complexity and is consistent with the result in Proposition 10. □

Proposition 12. *The problem of checking whether an overlapping coalition (C, V) in an OCS is feasible is in P.*

Proof. Since (C, V) is an overlapping coalition, we need to consider (C, V) and all the other overlapping coalitions that are connected to (C, V) together. That is, we should examine the ACS derived from (C, V). For each coalition in the ACS, we first check whether every member in it is interested in the collective task set and each task in the collective task set is of interest to a member in it. Then, we need to decide whether there is a feasible resource allocation scheme that can satisfy Constraint Sets (1) and (2). From Proposition 3, we know that the number of the connected coalitions is bounded, and thus the above tests can be completed in polynomial time. Therefore, the problem of checking whether an overlapping coalition (C, V) in an OCS is feasible can be solved in polynomial time. □

Proposition 13. *The worst case complexity of Algorithm 3 is $O(n^7)$.*

Proof. From Proposition 11, the main time-consumption of Algorithm 3 is the maximum flow computation, whose time complexity is at most $O(n^6)$. Additionally, by Proposition 3, there are at most m overlapping coalitions with non-empty collective task subsets in an ASC. Thus, the worst case complexity of Algorithm 3 is $O(m \cdot n^6) = O(n^7)$, which is of polynomial complexity and is consistent with the result in Proposition 12. □

Proposition 14. *The problem of checking whether an OCS is feasible is in P.*

Proof. From Proposition 3, we know that the number of coalitions in an OCS is bounded. Additionally, based on Definition 5, we have as long as the OCS contains a feasible coalition (C, V), the OCS is just feasible. Moreover, from Propositions 10 and 12, we can obtain that checking whether a coalition (C, V) in the OCS is feasible is polynomial-time decidable. Accordingly, checking whether an OCS contains a feasible coalition (C, V) is also polynomial-time decidable. It means that the problem of checking whether an OCS is feasible can be solved in polynomial time. □

Proposition 15. *The worst case complexity of Algorithm 5 is $O(n^7)$.*

Proof. In Algorithm 5, checking disjoint coalitions and ACSs takes up the major computational overhead. According to Propositions 11 and 13, the time complexity of the above operations is at most $O(n^7)$. Therefore, the worst case complexity of Algorithm 5 is also $O(n^7)$, which is of polynomial complexity and is consistent with the result in Proposition 14. □

Proposition 16. *The OCSGP is NP-complete.*

Proof. For membership of NP, we need to guess a feasible OCS. Specifically, it involves computing the value of each coalition in the OCS, and there are at most $m+1$ overlapping coalitions in any OCS (see Proposition 3). Additionally, to determine the value of a coalition, it needs to check feasibility according to Definition 4. Both steps can be done in polynomial time due to the size of the OCS (see Proposition 14), and so the problem is in NP.

To prove OCSGP is NP-hard, we give a reduction from the problem of finding an optimal OCS in TTGs (Chalkiadakis et al., 2010) to our problem. TTGs is a specific case of OCFGs and is given by: A set of agents $A = \{a_1, \ldots, a_n\}$, in which each agent a_j has a weight $b^j \in R^+$; a set of tasks $T = \{t_1, \ldots, t_m\}$, where each task t_i has a threshold $d^i \geq 0$ and a utility $u_i \geq 0$. The problem of finding an optimal OCS in TTGs is as follows: given A and T, whether is there an OCS whose utility is at least K? Chalkiadakis et al. (2010) and Zhan et al. (2012) showed that this problem is NP-complete. We will now construct an instance of our problem as follows. Let $r = 1$ (i.e., there is only one type of resources) and each $j \in \{1, \ldots, n\}$, $T_j = T$, namely, each agent is interested in all the tasks in T. Clearly, this instance is just a TTG. Consequently, based on Proposition 8 of (Chalkiadakis et al., 2010), we can obtain that our problem is NP-hard.

To sum up, the OCSGP is NP-complete, which is consistent with the result in Propositions 8 and 9. □

Proposition 17. *The worst case complexity of Algorithm 7 is $O(n^3)$.*

Proof. It can be observed that the most time-consuming step of Algorithm 7 is the feasibility examination for rows. There are at total m rows in a solution. For each selected row i, we first need to evaluate all the n agents and all the r types of resources to determine whether t_i can be achieved. The number of operations required for this check is $n \cdot r$. If t_i is achievable, at most n agents will be selected to create a resource allocation plan. Obviously, the number of operations needed for resource allocation is also $n \cdot r$. In summary, the total number of operations performed by row checks of Algorithm 7 is $m \cdot (n \cdot r + n \cdot r) = O(m \cdot n \cdot r) = O(n^3)$. □

Proposition 18. *As long as T contains at least a task that can be achieved by agents in A, the pending solution will be repaired to generate a feasible OCS by Algorithm 7.*

Proof. Without loss of generality, we assume that only a $t_i \in T$ can be satisfied and other tasks in T are unachievable for agents in A. According to Algorithm 7, after the repair, all the rows except row i contain only bits "0". When t_i is selected for checks, the remaining resources of available agents can make sure that $(C_i, \{t_i\})$ is feasible. Then, at least we can obtain a feasible $OCS = \{(C_i, \{t_i\}), (\{A - C_i\}, \emptyset)\}$ if $A - C_i \neq \emptyset$ or $OCS = \{(C_i, \{t_i\})\}$ if $A - C_i = \emptyset$. □

Proposition 19. *The worst case complexity of Algorithm 8 is $O(n^4)$.*

Proof. As shown in Algorithm 8, the run time of EAF is mainly spent in the employed EA (e.g., BPSO) and the embedded heuristic (i.e., Algorithm 7). The cost of Algorithm 7 is $O(n^3)$ (see Proposition 17). Suppose that in the employed EA, the maximum number of fitness evaluation is FE. Note that for each candidate solution, EAF needs to call Algorithm 7 to render it feasible before its fitness evaluation. As a consequence, the time complexity of EAF is $O(FE \cdot n^3) = O(n^4)$, which is of quartic polynomial complexity. □

References

Adalsteinsson, D., & Sethian, J. A. (1995). A fast level set method for propagating interfaces. *Journal of Computational Physics*, *118*(2), 269–277.

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network Flows: Theory, Algorithms and Application*. Prentice Hall, New Jersey, USA.

Airiau, S., & Sen, S. (2009). A fair payoff distribution for myopic rational agents. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1162–1163, Budapest, Hungary.

Anshelevich, E., & Sekar, S. (2015). Computing stable coalitions: Approximation algorithms for reward sharing. In *Proceedings of the 11th International Conference on Web and Internet Economics*, pp. 31–45, Amsterdam, Netherlands.

Bachrach, Y., Meir, R., Jung, K., & Kohiit, P. (2010). Coalitional structure generation in skill games. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pp. 703–708, Atlanta, Georgia, USA.

Bachrach, Y., Parkes, D. C., & Rosenschein, J. S. (2013). Computing cooperative solution concepts in coalitional skill games. *Artificial Intelligence*, *204*, 1–21.

Chalkiadakis, G., Elkind, E., Markakis, E., Polukarov, M., & Jennings, N. R. (2010). Cooperative games with overlapping coalitions. *Journal of Artificial Intelligence Research*, *39*, 179–216.

Chen, J., & Sun, D. (2012). Coalition-based approach to task allocation of multiple robots with resource constraints. *IEEE Transactions on Automation Science and Engineering*, *9*(3), 516–528.

Chen, W., Zhao, S., Zhang, R., & Yang, L. (2021). Generalized user grouping in noma based on overlapping coalition formation game. *IEEE Journal on Selected Areas in Communications*, *39*(4), 969–981.

Conitzer, V., & Sandholm, T. (2006). Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence*, *170*(6-7), 607–619.

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1990). *Introduction to Algorithms*. MIT Press, Cambridge, MA, USA.

Dang, V. D., & Jennings, N. R. (2006). Coalition structure generation in task-based settings. In *Proceedings of the 17th European Conference on Artificial Intelligence*, pp. 210–214, Garda, Italy.

Deng, X., & Papadimitriou, C. (1999). Decision-making by hierarchies of discordant agents. *Mathematical Programming*, *86*(2), 417–431.

Di, B., Wang, T., Song, L., & Han, Z. (2017). Collaborative smartphone sensing using overlapping coalition formation games. *IEEE Transactions on Mobile Computing*, *16*(1), 30–43.

Di, Z., Luo, T., Qiu, C., Zhang, C., Liu, Z., Wang, X., & Jiang, J. (2023). In-network pooling: Contribution-aware allocation optimization for computing power network in b5g/6g era. *IEEE Transactions on Network Science and Engineering*, *10*(3), 1190–1202.

Dunne, P. E., Kraus, S., Manisterski, E., & Wooldridge, M. (2010). Solving coalitional resource games. *Artificial Intelligence*, *174*(1), 20–50.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Boston, MA, USA.

Greco, G., & Guzzo, A. (2017). Constrained coalition formation on valuation structures: Formal framework, applications, and islands of tractability. *Artificial Intelligence*, *249*, 19–46.

Guajardo, M., Ronnqvist, M., Flisberg, P., & Frisk, M. (2018). Collaborative transportation with overlapping coalitions. *European Journal of Operational Research*, *271*(1), 238–249.

Hoefer, M., Vaz, D., & Wagner, L. (2018). Dynamics in matching and coalition formation games with structural constraints. *Artificial Intelligence*, *262*, 222–247.

Hou, R., Huang, K., & Lui, K. (2020). An overlapping coalition formation game based multicast scheme in backhaul-limited small cell networks. *IEEE Journal on Selected Areas in Communications*, *66*(3), 647–655.

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 4104–4108, Orlando, FL, USA.

Ketchpel, S. (1994). Forming coalitions in the face of uncertain rewards. In *Proceedings of the 12th AAAI Conference on Artificial Intelligence*, pp. 414–419, Seattle, WA, USA.

Krausburg, T., Dix, J., & Bordini, R. H. (2021a). Computing sequences of coalition structures. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pp. 1–7, Orlando, FL, USA.

Krausburg, T., Dix, J., & Bordini, R. H. (2021b). Feasible coalition sequences. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 719–727, London, United Kingdom.

Li, S., Wang, C., & Xie, G. (2024). Optimal strategies for pursuit-evasion differential games of players with damped double integrator dynamics. *IEEE Transactions on Automatic Control*, *69*(8), 5278–5293.

Liu, Y., Zhang, G., Su, Z., Yue, F., & Jiang, J. (2016). Using computational intelligence algorithms to solve the coalition structure generation problem in coalitional skill games. *Journal of Computer Science and Technology*, *31*(6), 1136–1150.

Lopez, V. G., Lewis, F. L., Wan, Y., Sanchez, E. N., & Fan, L. (2020). Solutions for multiagent pursuit-evasion games on communication graphs: Finite-time capture and asymptotic behaviors. *IEEE Transactions on Automatic Control*, *65*(5), 1911–1923.

Mahdiraji, H. A., Razghandi, E., & Hatami-Marbini, A. (2021). Overlapping coalition formation in game theory: A state-of-the-art review. *Expert Systems with Applications*, *174*.

Mamakos, M., & Chalkiadakis, G. (2017). Probability bounds for overlapping coalition formation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 331–337, Melbourne, Australia.

Markakis, E., & Saberi, A. (2005). On the core of the multicommodity flow game. *Decision Support Systems*, *39*(1), 3–10.

Pampara, G., Engelbrecht, A. P., & Franken, N. (2006). Binary differential evolution. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 1873–1879, Vancouver, BC, Canada.

Pan, T., & Yuan, Y. (2023). A region-based relay pursuit scheme for a pursuitcevasion game with a single evader and multiple pursuers. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *53*(3), 1958–1969.

Poli, R., & Langdon, W. B. (2006). Backward-chaining evolutionary algorithms. *Artificial Intelligence*, *170*(11), 953–982.

Prantare, F., & Heintz, F. (2020). An anytime algorithm for optimal simultaneous coalition structure generation and assignment. *Autonomous Agents and Multi-Agent Systems*, *34*(1).

Qi, N., Huang, Z., Zhou, F., Shi, Q., Wu, Q., & Xiao, M. (2023). A task-driven sequential overlapping coalition formation game for resource allocation in heterogeneous uav networks. *IEEE Transactions on Mobile Computing*, *22*(8), 4439–4455.

Rahwan, T., & Jennings, N. R. (2007). An algorithm for distributing coalitional value calculations among cooperating agents. *Artificial Intelligence*, *171*(8–9), 535–567.

Rahwan, T., Michalak, T., Wooldridge, M., & Jennings, N. R. (2012). Anytime coalition structure generation in multi-agent systems with positive or negative externalities. *Artificial Intelligence*, *186*, 95–122.

Rahwan, T., Michalak, T. P., Wooldridge, M., & Jennings, N. R. (2015). Coalition structure generation: A survey. *Artificial Intelligence*, *229*, 139–174.

Riley, L., Atkinson, K., Dunne, P. E., & Payne, T. R. (2015). Distributing coalition value calculations to coalition members. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pp. 2117–2123, Austin, Texas, USA.

Sandholm, T., Larson, K., Andersson, M., Shehory, O., & Tohme, F. (1999). Coalition structure generation with worst case guarantees. *Artificial Intelligence*, *111*(1-2), 209–238.

Sandholm, T. W., & Lesser, V. R. T. (1997). Coalitions among computationally bounded agents. *Artificial Intelligence*, *94*(1-2), 99–137.

Sen, S., & Dutta, P. S. (2000). Searching for optimal coalition structures. In *Proceedings of the 4th International Conference on MultiAgent Systems*, pp. 287–292, Boston, MA, USA.

Seo, Y. G., Cho, S. B., & Yao, X. (1999). Emergence of cooperative coalition in nipd game with localization of interaction and learning. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 877–884, Washington, DC, USA.

Seo, Y. G., Cho, S. B., & Yao, X. (2000). Exploiting coalition in co-evolutionary learning. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1268–1275, La Jolla, CA, USA.

Service, T. C., & Adams, J. A. (2011a). Coalition formation for task allocation: theory and algorithms. *Autonomous Agents and Multi-Agent Systems*, *22*(2), 225–248.

Service, T. C., & Adams, J. A. (2011b). Randomized coalition structure generation. *Artificial Intelligence*, *175*(16-17), 2061–2074.

Shehory, O., & Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence*, *101*(1-2), 165–200.

Shi, S., Hu, C., Wang, D., Zhu, Y., & Han, Z. (2023). Federated hd map updating through overlapping coalition formation game. *IEEE Transactions on Mobile Computing*, *23*(2), 1641–1654.

Slowik, A., & Kwasnicka, H. (2020). Evolutionary algorithms and their applications to engineering problems. *Neural Computing & Applications*, *32*(16), 12363–12379.

Srinivasan, M., & Murthy, C. S. R. (2020). Efficient spectrum slicing in 5g networks: An overlapping coalition formation approach. *IEEE Transactions on Mobile Computing*, *19*(6), 1299–1316.

Su, Z., Zhang, G., Liu, Y., Yue, F., & Jiang, J. (2016). Multiple emergency resource allocation for concurrent incidents in natural disaster. *International Journal of Disaster Risk Reduction*, *17*, 199–212.

Su, Z., Zhang, G., Yue, F., He, J., Li, M., Li, B., & Yao, X. (2020). Finding the largest successful coalition under the strict goal preferences of agents. *ACM Transactions on Autonomous and Adaptive Systems*, *14*(4).

Sun, Y., Wu, Q., Wang, J., Xu, Y., & Anpalagan, A. (2016). Veracity: Overlapping coalition formation-based double auction for heterogeneous demand and spectrum reusability. *IEEE Journal on Selected Areas in Communications*, *34*(10), 2690–2705.

Wang, T., Song, L., Han, Z., & Saad, W. (2014). Distributed cooperative sensing in cognitive radio networks: An overlapping coalition formation approach. *IEEE Transactions on Communications*, *62*(9), 3144–3160.

Wooldridge, M., & Dunne, P. E. (2006). On the computational complexity of coalitional resource games. *Artificial Intelligence*, *170*(10), 853–871.

Wu, C., & Liu, J. (2010). Level set and fat fast marching method for normal and dynamic path planning of pursuit-evasion problem. In *Proceedings of the IEEE International Conference on Automation and Logistics*, pp. 267–272, Hong Kong, China.

Xiao, Y., Chen, K., Yuen, C., Han, Z., & DaSilvan, L. A. (2015). A bayesian overlapping coalition formation game for device-to-device spectrum sharing in cellular networks. *IEEE Transactions on Wireless Communications*, *14*(7), 4034–4051.

Yu, T., Yang, H., Yao, Q., Yu, A., Zhao, Y., Liu, S., Li, Y., Zhang, J., & Cheriet, M. (2024). Multi-visual-gru-based survivable computing power scheduling in metro optical networks. *IEEE Transactions on Network and Service Management*, *21*(1), 1302–1315.

Yu, Y., Yao, X., & Zhou, Z. (2012). On the approximation ability of evolutionary optimization with application to minimum set cover. *Artificial Intelligence*, *180-181*(11), 20–33.

Zhan, Y., Wu, J., Wang, C., & Xie, J. (2012). On the complexity and algorithms of coalition structure generation in overlapping coalition formation games. In *Proceedings of the IEEE 24th International Conference on Tools with Artificial Intelligence*, pp. 868–873, Athens, Greece.

Zhang, G., Jiang, J., Lu, C., Su, Z., Fang, H., & Liu, Y. (2011). A revision algorithm for invalid encodings in concurrent formation of overlapping coalitions. *Applied Soft Computing*, *11*(2), 2164–2172.

Zhang, G., Jiang, J., Su, Z., Qi, M., & Fang, H. (2010). Searching for overlapping coalitions in multiple virtual organizations. *Information Sciences*, *180*(17), 3140–3156.

Zhang, G., Su, Z., Li, M., Qi, M., Jiang, J., & Yao, X. (2020). A task-oriented heuristic for repairing infeasible solutions to overlapping coalition structure generation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *50*(3), 785–801.

Zhang, G., Yang, R., Su, Z., Yue, F., Fan, Y., Qi, M., & Jiang, J. (2015). Using binary particle swarm optimization to search for maximal successful coalition. *Applied Intelligence*, *42*(2), 195–209.

Zhang, R., Zhao, Z., Cheng, X., & Yang, L. (2017). Overlapping coalition formation game based opportunistic cooperative localization scheme for wireless networks. *IEEE Transactions on Communications*, *65*(8), 3629–3642.

Zhang, Z., Song, L., Han, Z., & Saad, W. (2014). Coalitional games with overlapping coalitions for interference management in small cell networks. *IEEE Transactions on Wireless Communications*, *13*(5), 2659–2669.

Zhao, Y., Li, Y., Wu, D., & Ge, N. (2017). Overlapping coalition formation game for resource allocation in network coding aided d2d communications. *IEEE Transactions on Mobile Computing*, *16*(12), 3459–3472.

Zhou, P., & Chen, B. M. (2024). Distributed optimal solutions for multiagent pursuit-evasion games for capture and formation control. *IEEE Transactions on Industrial Electronics*, *71*(5), 5224–5234.

Zhou, Z., Ding, J., Huang, H., Takei, R., & Tomlin, C. (2018). Efficient path planning algorithms in reach-avoid problems. *Automatica*, *89*, 28–36.

Zick, Y., Chalkiadakis, G., & Elkind, E. (2019). Cooperative games with overlapping coalitions: Charting the tractability frontier. *Artificial Intelligence*, *271*, 74–97.

Zick, Y., & Elkind, E. (2011). Arbitrators in overlapping coalition formation games. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pp. 55–62, Taipei, China.

Zick, Y., Markakis, E., & Elkind, E. (2012). Stability via convexity and lp duality in ocf games. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 1506–1512, Toronto, Canada.

Zick, Y., Markakis, E., & Elkind, E. (2014). Arbitration and stability in cooperative games with overlapping coalitions. *Journal of Artificial Intelligence Research*, *50*, 847–884.