Decentralized Coalition Structure Formation for Interdependent Tasks Allocation

Douae Ahmadoun
LIPADE, University of Paris
Thales Research and Technology
Paris, France
douae.ahmadoun@etu.u-paris.fr

Elise Bonzon
LIPADE, University of Paris
Paris, France
elise.bonzon@u-paris.fr

Cédric Buron
Thales Research and Technology
Palaiseau, France
cedric.buron@thalesgroup.com

Pavlos Moraitis
LIPADE, University of Paris
Argument Theory Paris, France
pavlos.moraitis@u-paris.fr

Pierre Savéant
Thales Research and Technology
Palaiseau, France
pierre.saveant@thalesgroup.com

Onn Shehory
Bar Ilan University
Ramat-Gan, Israel
onn.shehory@biu.ac.il

***Abstract*—This paper addresses the problem of task allocation among multiple autonomous agents that must accomplish a complex global task. Solutions to the problem have real-world applications in defense, space, disaster management, etc. We solve this problem via agent coalition formation. Multiple coalition formation mechanisms were introduced in prior art, seldom accounting for interdependent tasks. We address this challenge. We introduce an anytime decentralized coalition formation mechanism that enables agents with complementary capabilities to form, autonomously and dynamically, feasible coalition structures that accomplish a global, composite task. The formed structures are incrementally improved via agent replacements to optimize a global utility. We analyze the complexity and show that, although the general problem is NP-hard, our mechanism provides a solution within acceptable time. We present extensive experimental results that illustrate the added value of our approach.**

***Index Terms*—coalition formation, decentralization, agents, task allocation, task interdependence, constraint solving**

I. Introduction

With the rise of low-cost robotics and drones, multi-agent coordination (MAC) has proven very effective for robotic teamwork (e.g., [20] [11]). Many MAC problems require multiple heterogeneous agents to concurrently perform a joint task, comprised of sub-tasks. E.g., in search and rescue problems [2], robots with complementary capabilities perform a set of tasks that jointly address a global task. Yet, the vast majority of such solutions assume task independence. In this study we assume *task interdependence*.

Such MAC problems are commonly solved via agent coalition formation [19]. Thus, the global task is accomplished by a set of coalitions comprising a coalition structure [16]. Optimal coalition formation and coalition structure generation are exponentially complex. Recent progress lead to complexity reduction in specific domains, however optimal solutions remain exponential. Task interdependence further increases complexity as the formation of a coalition and its utility may depend on other coalitions. Distributed solutions that attempt to ease complexity, e.g. [12], opt for an anytime approach, where quality improves as the formation process progresses.

In this paper we present a novel decentralized, anytime coalitions formation and task allocation mechanism, that diverges from the art in several ways. Specifically, we address coalition formation where sub-tasks and coalition utilities are interdependent, thus affecting the global utility of the coalition structure. To address this, our mechanism simultaneously considers local and global task requirements, accounting for interrelations thereof. Such interrelations are seldom considered in prior art. Additionally, our approach explicitly represents both qualitative and quantitative information on agent characteristics and task requirements.

Our coalition formation and task allocation mechanism is fully decentralized, thus preventing a single point of failure. Initially, agents only know their own characteristics, the global task and its sub-tasks and their respective requirements, and the set of the available agents. Gradually, agents may accumulate information on the characteristics of other agents and on potential coalitions and coalition structures. Throughout the process, each agent matches task requirements against its characteristics (and characteristics of other agents it learned about) and accordingly decides which coalition it should join to maximize global utility.

Our mechanism comprises 2 stages. Stage I finds a feasible coalition structure if one exists. Denote the method of stage I as *Feasible Interdependent Coalition Structure Anytime Method (FICSAM)*. If a solution is found, in stage II agents incrementally improve it in a decentralized manner via replacements of single agents in the coalition structure, while maintaining feasibility (i.e. no single or global task requirements are violated). Thus, we guarantee at anytime, the generation of a solution that systematically increases the global utility. Denote the method of stage II as *Improved Feasible Interdependent Coalition Structure Anytime Method (IFICSAM)*.

Extensive experiments show promising performance: the global utility with up to 100 agents and up to 20 tasks is close to the optimum and obtained within a reasonable runtime.

II. Problem Formulation

Given a global task $T = \{t_1, t_2, .., t_{|T|}\}$ and a set of agents $A = \{a_1, a_2, .., a_{|A|}\}$, we solve the problem of decentralized allocation of tasks $t_j \in T$ to agents $a_i \in A$, to accomplish T.

Definition 1 (Agent characteristics)**.** *$a_i \in A$ is described by a set of attributes $X = \{x_1, .., x_{|X|}\}$, $x_k(a_i)$ is the value of a_i under attribute k. Without loss of generality (w.l.o.g.) consider each attribute as a function $x_k : A \to D_k$, D_k the domain of values for k. We call these attributes* agent characteristics*.*

The agents have to perform tasks as defined below:

Definition 2 (Single task requirements)**.** *$t_j \in T$ is described by a set of attributes Γ_{t_j}=$\{\gamma_1, .., \gamma_m\}$, $\gamma_l(t_j)$ is the value of task t_j under attribute l. W.l.o.g. we consider each attribute as a function $\gamma_l : T \to K_l$, K_l the domain of values for attribute l. We call the attributes Γ_{t_j}* single task requirements*.*

Definition 3 (Task combinations requirements)**.** *Consider a set of attributes Λ_{cbt_j}=$\{\lambda_1, .., \lambda_n\}$ concerning task combinations s.t. $\lambda_k(cbt_j)$ the value of task combination $cbt_j \in T_{cbt}$, $T_{cbt} \subseteq 2^T$, under attribute k. W.l.o.g. consider each attribute as a function $\lambda_i : T_{cbt} \to L_k$, L_k the domain of values for k. We call the attributes Λ_{cbt_j}* task combinations requirements*.*

Single task requirements can be seen as constraints that have to be satisfied for the tasks to be accomplished. Task combinations requirements can be seen as constraints that have to be satisfied to address interdependence among tasks (e.g. temporal constraints imposing accomplishment order).

Example 1. *Consider $A = \{a_1, a_2, a_3, a_4\}$ scattered on a $10m \times 10m$ grid, that must perform $T = \{t_1, t_2\}$. Agent characteristics are location x_1, energy x_2 and payload x_3 (normal camera C or thermal one R). See Table I.*

x_k	$x_k(a_1)$	$x_k(a_2)$	$x_k(a_3)$	$x_k(a_4)$
x_1	$(0,0)$	$(3,3)$	$(1,2)$	$(5,3)$
x_2	10	5	7	8
x_3	C	R	C	R

TABLE I: Example of characteristics of agents

Each task has 4 requirements. γ_1: task location and the maximal allowed distance of an agent from that location. γ_2: minimum energy needed to perform the task. γ_3 minimum required payload for the task (brought by all the agents). γ_4: minimum number of agents needed to accomplish the task. Requirements of $\{t_1, t_2\}$ are presented in Table II.

γ_l	$\gamma_l(t_1)$	$\gamma_l(t_2)$
γ_1	$\langle(3,0), \leq 4\rangle$	$\langle(4,4), \leq 10\rangle$
γ_2	≥ 3	≥ 7
γ_3	$\supseteq \{(C,1)\}$	$\supseteq \{(C,1),(R,1)\}$
γ_4	≥ 1	≥ 2

TABLE II: Example of requirements of tasks

We also define a requirement on combinations of tasks over the maximum number of agents allocated to the tasks. Here $\Lambda_{cbt} = \{\lambda_1\}$, where $\lambda_1(\{t_1, t_2\}) \leq 3$.

Combined characteristics of a set of agents may allow to fulfil task requirements, or conflict with such requirements.

Definition 4 (Fulfillment relation)**.** *Let $s = \{a_1, .., a_{|s|}\}$ a set of agents, $X_s = \{X_{a_1}, .., X_{a_{|s|}}\}$ their characteristics, $t_j \in T$ a task and $\bowtie$ denoting the satisfaction (w.r.t. a mathematical operator e.g. $=, \leq, \geq$,... according to the case) of the assignment of a value to a requirement $\gamma_l(t_j)$ by the assignment of a value to some characteristic $x_k(a_i)$. We say that s can fulfil a requirement $\gamma_l \in \Gamma_{t_j}$ denoted $s_{cc} \circlearrowleft \gamma_l(t_j)$ if there exists a combination of characteristics $cc = \{x_k, .., x_r\} \subseteq X_{a_1} \cup, .., \cup X_{a_{|s|}}$ such that $\sum_{i=1}^{|s|} x_k(a_i) \bowtie \gamma_l(t_j)$ for some $x_k \in cc$ or $x_r(a_i) \bowtie \gamma_l(t_j)$ for some $x_r \in cc$ with $r \neq k$, saying (slightly abusing the notation) that $cc \bowtie \gamma_l$. In contrast, we say that s_{cc} has a conflict with the requirement $\gamma_l \in \Gamma_{t_j}$, if $\exists x_k \in cc$ s.t. x_k is in conflict with this γ_l denoted as $x_k \not\bowtie \gamma_l$.*

Example 1. ***Continued.*** *To assess requirement fulfilment, we first compute agent distances from task locations (see Table III). For simplicity, w.l.o.g., we use Manhattan distances.*

$d(a_i, t_j)$	a_1	a_2	a_3	a_4	$d(a_i, t_j)$	a_1	a_2	a_3	a_4
t_1	3	3	4	5	t_2	8	2	5	2

TABLE III: Manhattan distance between agents and tasks

For example, $\{a_1\} \circlearrowleft \gamma_1(t_1)$, as a_1 respects the maximal distance from t_1; $\{a_1, a_3, a_4\} \circlearrowleft \gamma_2(t_2)$, as these agents have the minimal energy needed to perform t_2; $\{a_1, a_2\} \circlearrowleft \gamma_3(t_1)$, as a_1 brings the resource C. However, $x_2(a_2) \not\bowtie \gamma_2(t_2)$ as a_2 does not have enough energy to perform t_2.

Agents can form *coalitions* to accomplish a task $t_j \in T$.

Definition 5 (Coalition)**.** *Let $\tau : A \to T$ a function assigning a_i to t_j when $\exists x_k \in X_{a_i}$, $\exists \gamma_l \in \Gamma_{t_j}$ s.t. $x_k \bowtie \gamma_l$, and $\nexists x_m \in X_{a_i}$ s.t. $x_m \not\bowtie \gamma_p$ for any $\gamma_p \in \Gamma_{t_j}$ with $p \neq l$.* Coalition *C_{t_j} whose task is t_j is $C_{t_j} = \{a_i \in A \mid \tau(a_i) = t_j\} \in 2^A$.*

A global task T requires a set of coalitions S, called *coalition structure*. Each $C_{t_j} \in S$ is assigned a task $t_j \in T$. When S can accomplish T it is a *feasible coalition structure*.

Definition 6 (Feasible coalition structure)**.** *Let a coalition structure $S = \{C_{t_1}, C_{t_2}, ..., C_{t_{|T|}}\}$ over T. S is a* feasible coalition structure *(or feasible solution) denoted S^f iff $\forall t_j \in T$ s.t. $C_{t_j} \in S^f$ it holds that $\forall \gamma_l \in \Gamma_{t_j}$, $\exists s_{cc} \subseteq C_{t_j}$ s.t. $s_{cc} \circlearrowleft \gamma_l(t_j)$ and if Λ_{cbt} is a set of requirements concerning combination of tasks then $X_{S^f} \bowtie \Lambda_{cbt}$.*

We use coalition structure modification functions:

- $\oplus : S \times (A \times T) \to S$, to add a specific agent to a specific coalition. $S \oplus (a_i, t_j)$ is a structure in which $C_{t_j} \leftarrow C_{t_j} \cup \{a_i\}$ and $\forall k \neq j$, C_{t_k} does not change.
- $\ominus : S \times (A \times T) \to S$, to remove a specific agent from a specific coalition. $S \ominus (a_i, t_j)$ is a structure in which $C_{t_j} \leftarrow C_{t_j} \setminus \{a_i\}$ and $\forall k \neq j$, C_{t_k} does not change.

Example 1. ***Continued.*** *$S_1^f = \{\{a_1\}, \{a_3, a_4\}\}$ and $S_2^f = \{\{a_3\}, \{a_1, a_4\}\}$ are the only two feasible coalition structures: the requirements of t_1, t_2 and T are all fulfilled.*

One can observe that no other structure is feasible. E.g., $S = \{\{a_1, a_2\}, \{a_3, a_4\}\}$ is not feasible: if t_1, t_2 requirements are fulfilled, the requirement over T is not.

We refer to single-task agents [9], i.e., can accomplish only one task at a time. Thus, $C_{t_j}, C_{t_k} \in S, j \neq k \Rightarrow C_{t_j} \cap C_{t_k} = \emptyset$. Some agents have no task assignment, thus $\bigcup_{j=1}^{|T|} C_{t_j} \subseteq A$.

We define means to evaluate coalition structures w.r.t. T.

Definition 7. *Let CS be the set of all possible coalition structures that could be assigned to a global task T. Let G be a set of criteria s.t. $\forall g_k \in G$ there exists a weak order G_k upon the set CS, $G_k \subseteq CS^2$ s.t. if $(S, S') \subseteq G_k$, then $S \succeq S'$ and $\exists g_k : CS \to \mathbb{R}$, $g_k(S) \geq g_k(S')$. Then, w.l.o.g., we define the decision problem: $\forall k, \max_{S \in CS} g_k(S)$, which should identify the coalition structures S maximizing "simultaneously" the performance of such structures upon all $g_k \in G$ for T.*

To consider task interdependence, we need to define a function that evaluates a coalition structure S as a whole, accounting for coalition interdependence. Provided that the conditions of commensurability, compensation and preferential independence are satisfied among the criteria in G (see [4]), a global additive value function u_{global} is applicable:

$$u_{global}(S) = \Sigma_k u_k(g_k(S))$$

We normalize to the interval [0,1]. $\bar{u}_k$ are the normalized functions and w_k are criteria importance weights. We get:

$$u_{global}(S) = \Sigma_k w_k \bar{u}_k(g_k(S))$$

Our study aims to design an algorithm that finds a feasible coalition structure S^{*f} that maximizes $u_{global}(S)$. Our generic approach allows considering other evaluation functions too.

$$u_{global}(S^{*f}) = \max_{S \in CS} \Sigma_k w_k \bar{u}_k(g_k(S))$$

Example 1. ***Continued.*** *We define 5 criteria to evaluate a coalition structure S:*

- *# agents near (w.r.t a threshold) the task they are allocated: $g_1(S) = \Sigma_{C_{t_j} \in S} |\{a_i \in C_{t_j} | dist(a_i, t_j) \leq th_1^{t_j}\}|$*
- *# agents whose energy is greater than a threshold: $g_2(S) = \Sigma_{C_{t_j} \in S} |\{a_i \in C_{t_j} | x_2(a_i) \geq th_2^{t_j}\}|$*
- *# agents that bring the resources that are needed: $g_3(S) = \Sigma_{C_{t_j} \in S} |\{a_i \in C_{t_j} | x_3(a_i) = th_3^{t_j}\}|$*
- *# agents allocated to each task, given task threshold: $g_4(S) = \Sigma_{C_{t_j} \in S} (1 \text{ if } |C_{t_j}| \geq \gamma_4(t_j); |C_{t_j}|/\gamma_4(t_j))$*
- *otherwise, g_5: 1 if the maximum # agents allocated to the global task is respected, 0 otherwise.*

With equal criteria weights, we define the normalized functions $\bar{u}_k(g_k(S))$ to compute the utilities of coalition structures.

$$u_{global}(S) = 1/5\big((g_1 + g_2 + g_3)/|A| + g_4/|T| + g_5\big)$$

If we assume that $th_1^{t_1} = th_1^{t_2} = 3$, $th_2^{t_1} = 4$, $th_2^{t_2} = 9$, $th_3^{t_1} = C$, $th_3^{t_2} = C \vee R$, we have:

$$u_{global}(S_1^f) = 1/5\big((2 + 1 + 3)/4 + 2/2 + 1\big) = 0.7$$

$$u_{global}(S_2^f) = 1/5\big((2 + 2 + 3)/4 + 2/2 + 1\big) = 0.75$$

III. Our Approach

A. General Description

As stated above, we aim at finding the best feasible coalition structure (w.r.t. u_{global}), if it exists, or detect nonexistence early on. In our solution, a structure comprises agent coalitions and a task assigned to each coalition. We propose a decentralized solution approach based on token passing among the agents. The process is divided into rounds. In each round the token is circulated among the agents. The round ends when all agents have received the token once. Agent ordering depends on application-based criteria.

At start, each agent knows its own characteristics and the global task to be accomplished. Information about the other agents and the evolution of the coalition structure formation arrives via the token passing process. When a_i sends the token to a_j, it adds information on the best feasible coalition structure S formed so far, X_S, and the round of the process. Holding the token, a_i may decide to join (or initiate) a coalition C_{t_j} assigned to t_j, if it can contribute to the accomplishment of t_j or its participation can increase u_{global}. Where applicable, a_i updates the information it received with the changes it has applied. Then, a_i passes the token to the next agent. We assume agents can exchange messages, yet the underlying communication infrastructure is beyond this article's scope.

Our mechanism has 2 stages. Stage I coincides with the first round, implementing the FICSAM method. It finds a feasible coalition structure S^f, if one exists. Hence, an agent joins a coalition only if it can satisfy some task requirements not yet satisfied by other coalition members, regardless of the impact on u_{global}. If no S^f is found in stage I, no such S^f exists.

Stage II implements the IFICSAM method, incrementally improving feasible the feasible coalition structure found so far. It starts from round 2, once S^f was found in round 1. Agents improve u_{global} via replacements or swaps between agents. The resulting improved structure must preserve feasibility, respecting all requirements.

B. Agent Decision Process

Algorithm 1 describes the decision process of agent a_i, triggered when a_i gets a message from some a_j during the coalition formation process. Its decision depends on the message content and the round R of the process. Firstly (line 3), a_i checks the feasibility of the received coalition structure S according to feasibility criteria of the given application. Then, if $R[i] = 1$ and S not yet feasible, it computes a feasible structure w.r.t. T, task-level and combination-level requirements, and its potential contribution if it joins S. For this, it uses procedure *s-f-st* (line 5) that models this problem as a constraint satisfaction problem (CSP) where variables represent agents (i.e. $\{a_i\} \cup S$) decisions, each variable's domain is the task set T and constraints implement tasks requirements satisfaction w.r.t. agents characteristics, i.e., $(X_{a_i} \cup X_{S \setminus a_i}) \bowtie \Gamma_T$ and $(X_{a_i} \cup X_{S \setminus \{a_i\}}) \bowtie \Lambda_{cbt}$. We use the Lazy Clause Generation based constraint solver Chuffed [6] but other solvers can be used as well.

As said earlier, when a_i holds the token it adds the information about its characteristics to X_S. Thus, in round 1, when a_i gets the token, it has to solve a centralized coalition formation problem based on the knowledge accumulated so far. a_i tries to find whether the characteristics in X_S are sufficient to satisfy all requirements of both individual tasks and task combinations. Requirements are modeled as hard constraints [8] and a CSP is solved. Other centralized coalition formation methods, e.g., [19], can be applied too. If no feasible structure is found in round 1 (i.e., the CSP has no solution), the process terminates as there is no solution. Else, the process proceeds to gradually improve the initial feasible structure.

In line 6 a_i checks the feasibility of S returned by *s-f-st* by executing *check-feasibility*. The latter is implemented as a CSP whose input includes agent variables, tasks, constraints corresponding to local and global requirements and the solution represented as the variables domains. If the structure is feasible the solver returns it. Otherwise it proves that no solution exists. If S is feasible, we found the first feasible solution S^f. This structure is sent by a_i to all members of the coalitions of the structure formed so far, along with X_S and initializes counter $nd \leftarrow 0$ ("nd" stands for "number of decisions"). nd is initialized when a new solution is introduced in the process by an agent. It is incremented when an agent is unable to improve the current feasible structure.

When a_i cannot find a feasible structure with X_S, it examines ways to contribute to feasible structures by agents that have not yet had the token in that round. It examines tasks whose requirements match its characteristics, i.e., tasks in T_i (line 16). Then a_i picks a task $t_j \in T_i$ (line 18), joins $C_{t_j} \in S$, and adds its characteristics to X_S. As before, a_i checks whether the updated S improves u_{global}. a_i sends the token to the next agent, including S and X_S in the message. However, if a_i is the last to receive the token, it implies that there is no feasible solution. a_i informs the agents in S about this.

If $R[i] > 1$, a feasible solution was already found (line 34) and the agents seek another feasible solution that maximizes u_{global}. For this, a_i can use Algorithm 2, (line 35), presented later. If the returned improved coalition structure is feasible (line 37) then it becomes the best current feasible one (line 38). In this case a_i sends this solution to all the agents in the structure. If the returned structure is not feasible, it is not further considered. Hence, a_i reconsiders the feasible structure that it received from the previous agent (line 41).

Lines $42-47$ concern cases where S is feasible but a_i could not improve it. Here, if $nd < |A|$, a_i increments nd and passes the token to the next agent. Otherwise S cannot be improved anymore and a_i informs the agents that the process is ending with S as the best feasible solution found so far.

C. Improved FICSAM (IFICSAM)

Algorithm 2 allows agents to improve in $R > 1$. Once a_i receives the token, it assumes that the received structure S maximizes u_{global} (line 1). It then checks whether it can increase u_{global} in two ways. The first consists of a_i switching to

Algorithm 1: decide($a_i, X_{a_i}, T, \Gamma_T, \Lambda_{cbt}, S, X_S, R, nd, A$, $is\_f(S)$)

1 $S_{old} \leftarrow S$; $R[i] \leftarrow R[i] + 1$
2 **if** $R[i] = 1$ **then**
3 **if** $not(is\_f(S))$ **then**
4 $S_{max} \leftarrow S$
5 $S \leftarrow$ *s-f-st*$(T, \Gamma_T, \Lambda_{cbt}, \{a_i\} \cup S, a_i, X_{a_i} \cup X_{S \setminus \{a_i\}})$
6 $is\_f(S) \leftarrow$ *check-feasibility*$(T, \Gamma_T, \Lambda_{cbt}, A, S)$
7 **if** $is\_f(S)$ **then**
8 $S^f \leftarrow S$; $nd \leftarrow 0$
9 $S_{max} \leftarrow S^f$
10 **for** $C_{t_j} \in S$ **do**
11 **for** $a_l \in C_{t_j}$ **do**
12 send(*propose*$(a_i, a_l, \langle "ats", S, X_S, R, nd \rangle)$)
13 **else**
14 **if** $R \neq \langle 1, ..., 1 \rangle$ **then**
15 $T_i \leftarrow \emptyset$
16 **for** $t_j \in T$ s.t. $X_{a_i} \bowtie \Gamma_{t_j}$ **do**
17 $T_i \leftarrow T_i \cup \{t_j\}$
18 Get $t_j \in T_i$
19 $S \leftarrow S \oplus (a_i, t_j)$
20 $X_S \leftarrow X_S \cup X_{a_i}$
21 **if** $u_{global}(S) > u_{global}(S_{max})$ **then**
22 $S_{max} \leftarrow S$; $nd \leftarrow 0$
23 send(*propose*$(a_i, next(a_i), \langle "gt", S, X_S, R, nd \rangle)$)
24 $S \leftarrow S_{max}$; $nd \leftarrow nd + 1$
25 send(*propose*$(a_i, next(a_i), \langle "gt", S, X_S, R, nd \rangle)$)
26 **else**
27 (i.e. $R = \langle 1, ..., 1 \rangle$)
28 $S^f \leftarrow \langle [], [], ..., [] \rangle$
29 **for** $C_{t_j} \in S$ **do**
30 **for** $a_l \in C_{t_j}$ **do**
31 send(*inform*$(a_i, a_l, \langle "end", \emptyset, \emptyset, 1 \rangle)$)
32 **else**
33 (i.e. $R[i] > 1$)
34 $S^f \leftarrow S$
35 call IFICSAM(a_i, T_i, S)
36 $is\_f(S) \leftarrow$ *check-feasibility*$(T, \Gamma_T, \Lambda_{cbt}, A, S)$
37 **if** $is\_f(S)$ and $S \neq S^f$ **then**
38 $S^f \leftarrow S$; $nd \leftarrow 0$
39 **for** $t_j \in T$ **do**
40 **for** $a_l \in C_{t_j}$ **do**
 send(*propose*$(a_i, a_l, \langle "ats", S, X_S, R, nd \rangle)$)
41 **else** $S \leftarrow S_{old}$
42 **if** $nd < |A|$ and $S = S_{old}$ **then**
43 $nd \leftarrow nd + 1$
44 send(*propose*$(a_i, next(a_i), \langle "gt", S, X_S, R, nd \rangle)$)
45 **else**
46 **for** $a_l \in A$ **do**
47 send(*inform*$(a_i, a_l, \langle "end", S, X_S, R \rangle)$)

another task (and coalition). Agent a_i examines its contribution to its coalition $C_{t_j} \in S$. There may be $t_k \in T_i$ that, if a_i contributes to its performance instead of contributing to t_j, u_{global} increases. Task switching is relevant if no requirement of t_j is violated and if u_{global} increases when a_i participates in the accomplishment of t_k instead of t_j (line 3). In that case the new structure becomes the maximal one (line 4).

The second consists of agent switching. a_i checks whether it can replace another agent a_l that currently fulfils the requirements of t_j assigned to $C_{t_j} \in S$ (i.e. a_l leaves S) or to swap with it, by fulfilling the requirements of t_k currently fulfilled by a_i in the coalition assigned to t_k. If the change increases u_{global} it is implemented, and S is updated to a new structure S', $u_{global}(S') > u_{global}(S)$ (lines 6-13). Feasibility of this new solution is verified in Algorithm 1 (line 36).

Algorithm 2: IFICSAM(a_i, T_i, S)

1 $S_{max} \leftarrow S$
2 **for** $t_j \in T_i \setminus \tau(a_i, S)$ **do**
3 **if** doesn't exist $\gamma_m \in \Gamma_{t_j}$ s.t. $X_{a_i} \not\bowtie \gamma_m(t_j)$ and $u_{global}(S \oplus (a_i, t_j) \ominus (a_i, \tau(a_i, S)) > u_{global}(S_{max})$ **then**
4 $S_{max} \leftarrow S \oplus (a_i, t_j) \ominus (a_i, \tau(a_i, S))$
5 **for** $a_k \in C_{t_j}$ s.t. doesn't exist $\gamma_l \in \Gamma_{\tau(a_i,S)}$ with $X_k \not\bowtie \gamma_l(\tau(a_i, S))$ **do**
6 **if** $u_{global}\big(S \oplus (a_i, t_j) \ominus (a_k, t_j) \ominus (a_i, \tau(a_i, S))\big) > u_{global}\big(S \oplus (a_i, t_j) \oplus (a_k, \tau(a_i, S)) \ominus (a_i, \tau(a_i, S)) \ominus (a_k, t_j)\big)$ **then**
7 $S_{new} \leftarrow S \oplus (a_i, t_j) \ominus (a_k, t_j) \ominus (a_i, \tau(a_i, S))$
8 **else**
9 $S_{new} \leftarrow S \oplus (a_i, t_j) \oplus (a_k, \tau(a_i, S)) \ominus (a_i, \tau(a_i, S)) \ominus (a_k, t_j))$
10 **if** $u_{global}(S_{new}) > u_{global}(S_{max})$ **then**
11 $S_{max} \leftarrow S_{new}$
12 **if** $u_{global}(S_{new}) > u_{global}(S_{max})$ **then**
13 $S_{max} \leftarrow S_{new}$
14 **return** S_{max}

D. Coalition Formation Global Procedure

Algorithm 3 implements the global behavior of agents participating in the process and acting either as process initiators (lines 1-15) or as candidate members of coalitions in S (lines 16-34). An initiator starts by initializing round 1 (line 2), then looking for tasks in T with requirements that match its characteristics (i.e. $X_{a_i} \bowtie \Gamma_{t_j}$) (line 4). It builds $T_i \subseteq T$, the list of tasks it can perform (line 5). It picks one, say t_j (line 6), and initializes a coalition in S (line 7) that is assigned to t_j. It adds its characteristics to X_S (initially empty) (line 8). X_S will accumulate the characteristics of all members of the coalitions in S. Then, it checks feasibility of S using *check-feasibility*. In the (less probable) case that S is feasible (e.g., if T contains only task t_j), S becomes an anytime solution. a_i considers S as a feasible structure to explore ($nd \leftarrow 0$) and sends this proposal to all agents in A (lines 12-13). Thus, it initiates a process that checks whether there exists another feasible coalition structure S' that improves $u_{global}(S)$. If S is not feasible, a_i initializes a coalition structures formation process by sending a message to the next agent a_j (line 15). With this message a_i passes the token to a_j who will enter the process. a_i informs a_j about S, the characteristics accumulated so far and the current round (i.e. round 1).

When agent a_i acts as a candidate member of a coalition structure its activity depends on the messages it receives from other agents. In case of a *propose*$(a_l, a_i, \langle$"ats"$, S, X_S, R, nd\rangle)$ message, if an anytime solution is required, the process terminates with S as the best feasible structure found so far (w.r.t. u_{global}). This can occur either at the end of round 1 or in the middle of another round. Otherwise the receiving agent a_i considers that S is a feasible structure (line 23) that can be possibly further improved (wrt u_{global}) and for that it uses procedure *decide*. The message *propose*$(a_l, a_i, \langle$"gt"$, S, X_S, R, nd\rangle)$ means that S is not a feasible solution and the sender a_l passes the token (gt) to a_i who will use *decide* to examine whether it can contribute to finding a feasible solution. The message *inform*$(a_l, a_i, \langle$"end"$, \emptyset, \emptyset, 1\rangle)$ informs the agents that no feasible structure was found in round 1 and the process ends with failure, while message *inform*$(a_l, a_i, \langle$"end"$, S, X_S, R\rangle)$ informs of an end with a feasible and improved solution.

E. Complexity

We discuss the complexity of Algorithm 1 and Algorithm 3. These algorithms may rely on others, in which case we may discuss the complexity of those algorithms too. Algorithm 1 appears as a simple procedure, linear in $|T|$ and $|A|$. However, one can observe that it calls other procedures, i.e., *s-f-st* and *check_feasibility*, that are solving CSPs [8] whose complexity is NP-hard (one can show reduction from the 3-SAT problem). Hence, Algorithm 1 is NP-hard as well. This may seem prohibitive but CSP solvers like the one we use (Chuffed [6]) allow to efficiently deal with high complexity problems. Algorithm 3 also seems linear in $|T|$ and $|A|$. However, it calls Algorithm 1. Hence, it is NP-hard too, but solvable in practice.

IV. Experimental Evaluation

We illustrate the added value of our approach and evaluate its performance by benchmarking on a sample application. We also compare performances to a centralized method.

A. The scenario generator

We evaluate our approach with a set of scenarios generated automatically. These are used to benchmark *FICSAM*, *IFICSAM* and a centralized solution. The scenarios are generated according to the following settings: a fleet of Unmanned Aerial Vehicles (UAVs) are assigned a mission in a seaport. The (UAV) agents must inspect hulls of boats in the port. The UAVs are lying on Unmanned Surface Vehicles (USVs), where they can charge. There are typically tens of UAV agents. However, to stretch-test our approach and compare it to a centralized approach, we experiment with up to 100 agents and 20 tasks. The USVs are scattered across the port, so that

Algorithm 3: coalition-formation(T, A, Γ_T, Λ_{cbt}, *msg*(*perf*$(a_l, a_i, <$ *content* $>)$))

if agent a_i makes the first proposal **then**
  $S_{max} \leftarrow \emptyset$; $S \leftarrow \emptyset$; $R[i] \leftarrow R[i] + 1$
  $X_S \leftarrow \emptyset$; $T_i \leftarrow \emptyset$; $nd \leftarrow 0$
  for $t_j \in T$ s.t. $X_{a_i} \bowtie \Gamma_{t_j}$ **do**
    $T_i \leftarrow T_i \cup \{t_j\}$
  Get $t_j \in T_i$
  $S \leftarrow S \oplus (a_i, t_j)$
  $X_S \leftarrow X_S \cup X_{a_i}$
  $is\_f(S) \leftarrow$ *check-feasibility*$(T, \Gamma_T, \Lambda_{cbt}, A, S)$
  if $is\_f(S)$ **then**
    $S_{max} \leftarrow S$
    for $a_l \in A$ **do**
      send(*propose*$(a_i, a_l, \langle$"ats"$, S, X_S, R, nd\rangle)$)
  else
    send(*propose*$(a_i, next(a_i), \langle$"gt"$, S, X_S, R, nd\rangle)$)
while true **do**
  Get *msg*(*perf*$(a_l, a_i, <$ *content* $>)$)
  switch *msg*(*perf*$(a_l, a_i, <$ *content* $>)$) **do**
    case *propose*$(a_l, a_i, \langle$"ats"$, S, X_S, R, nd\rangle)$ **do**
      if *requirement_anytime_solution* **then**
        $decision \leftarrow$ "$success$"
        End of coalition formation process with a feasible solution
      else
        $is\_f(S) \leftarrow true$
        call *decide*$(a_i, X_{a_i}, T, \Gamma_T, \Lambda_{cbt}, S, X_S, R, nd, A, is\_f(S))$
    case *propose*$(a_l, a_i, \langle$"gt"$, S, X_S, R, nd\rangle)$ **do**
      $is\_f(S) \leftarrow false$
      call *decide*$(a_i, X_{a_i}, T, \Gamma_T, \Lambda_{cbt}, S, X_S, R, nd, A, is\_f(S))$
    case *inform*$(a_l, a_i, \langle$"end"$, \emptyset, \emptyset, 1\rangle)$ **do**
      $decision \leftarrow$ "$failure$"
      End of the coalition process in first round with no feasible solution found
    case *inform*$(a_l, a_i, \langle$"end"$, S, X_S, R\rangle)$ **do**
      $decision \leftarrow$ "$success$"
      End of coalition formation process with a feasible and improved (wrt global utility) solution

the UAVs can easily charge to handle new tasks. Our scenario generator implements this by random positioning of USVs (with a uniform distribution) on the port grid.

Inspection tasks may require various sensors. Here, we rely on two sensor types: HD cameras and LASERs (see e.g., [1]). The quantity of each resource required by a task depends on boat hull. In our scenario generator, the number of resources of each type is uniformly sampled between 0 and $\frac{n_{agents}}{2 \cdot n_{tasks}}$. This maintains scenario diversity and simplifies comparison across scenarios and settings, as averages are the same and can be compared without normalization. In addition to resource constraints, the generator introduces task interdependence via constraints on sets of tasks. Finally, each task has a deadline.

The scenario generator also generates UAV agents. For the sake of simplicity, all UAVs have the same maximum speed. Therefore, the travel time to a task is proportional to the distance between the task and the UAV. Given a grid size G, the generator randomly and uniformly draws UAV distances from $[G/2 \ldots G]$. The agents are provided with sensors s.t. 25% of them have both sensors, 37.5% have only a LASER and 37.5% have only an HD camera. The token passing strategy implemented is based on inter-agents distances. An agent that holds the token sends it to the nearest agent that hasn't yet received the token in the current round. Finally, the global utility function is a normalized additive function computed for tasks and task combinations by accumulating their values, meeting their requirements, matched against coalitions' and agents' characteristics.

The generator produces both feasible and infeasible scenarios. The latter are of interest for method comparison, as it requires that the algorithms prove unsatisfiability, which might take a long time.

B. Setup

To understand the impact of the number of agents $|A|$ and tasks $|T|$, simulations are performed considering sample mission scenarios with $|A| \in [5..100]$ and $|T| \in [2..20]$. Tasks are handled by multiple agents, hence $|A| > |T|$. The reported results include algorithms' execution time, utilities of the structures they return and the number of exchanged messages. For each $\{|A|, |T|\}$ pair, we executed 200 runs. The execution platform was a 3.70 GHz Intel(R) Core(TM) i9-10900X CPU running Python and the MiniZinc tool chain.

C. Centralized Solution

Our decentralized approach allows agents to make local decisions and avoid a single point of failure. However, comparison to a centralized solution facilitates evaluation of our solution's distance from optimum, and execution time.

For the centralized method, we modeled our allocation problem as a Constraint Optimization Problem (COP) and solved it with the Lazy Clause Generation based constraint solver Chuffed [6] through the constraint modeling language MiniZinc [14].

Since the proof of unsatisfiability or the proof of optimality might be very long, we instrumented our code with a timeout. Whenever the search is interrupted, we consider the problem as unsatisfiable for the first case and as the best solution found so far for the second case. In addition, in order to prevent from pathological cases, we filter the instances with a series of necessary and sufficient conditions. No need to bother the solver in such cases, which might take a long time to prove unsatisfiability.

D. Results Evaluation

We present in Table IV the results of *FICSAM*, *IFICSAM* and the aforementioned centralized approach for each metric. Each single result in the table is an average over experiments with 200 scenarios that were randomly generated by the scenario generator. The three methods were all tested on the same scenarios. The remainder of this section presents the

Agents number	Tasks number	Utilities			Execution time in sec (error type)			Messages number	
		FICSAM	IFICSAM	Centralized	FICSAM	IFICSAM	Centralized	FICSAM	IFICSAM
5	2	0.58	0.73	0.83	0.9 (± 0.0)	1.4 (± 0.0)	0.2 (± 0.0)	18.4	25.9
10	2	0.59	0.79	0.85	1.6 (± 0.0)	3.0 (± 0.1)	0.2 (± 0.0)	36.8	71.7
10	5	0.55	0.70	0.78	1.7 (± 0.0)	2.9 (± 0.1)	48.0 (± 5.3)	38.0	61.8
20	2	0.53	0.73	0.85	3.6 (± 0.01)	6.8 (± 0.1)	0.2 (± 0.0)	71.5	177.2
20	5	0.53	0.75	0.86	3.4 (± 0.1)	6.1 (± 0.1)	0.2 (± 0.0)	71.8	170.7
20	10	0.53	0.70	0.79	42.0 (± 6.9)	43.9 (± 7.0)	1184.4 (± 9.0)	75.6	158.2
50	2	0.51	0.67	0.85	7.0 (± 0.2)	11.6 (± 0.3)	4.6 (± 6.0)	178.3	557.6
50	5	0.49	0.67	0.86	69.1 (± 11.1)	76.4 (± 11.3)	0.3 (± 0.0)	176.0	576.3
50	10	0.54	0.71	0.86	387.6 (± 25.4)	398.4 (± 25.4)	25.1 (± 7.2)	181.7	526.8
50	20	0.44	0.61	0.80	212.6 (± 25.1)	222.2 (± 25.1)	1195.7 (± 4.5)	180.3	511.7
100	2	0.48	0.62	0.85	70.2 (± 14.2)	83.4 (± 14.3)	1.8 (± 0.4)	351.3	1349.9
100	5	0.48	0.61	0.86	189.7 (± 24.1)	209.3 (± 24.0)	1.2 (± 7.3)	349.7	1204.1
100	10	0.48	0.63	0.86	446.5 (± 24.7)	468.9 (± 24.4)	32.0 (± 7.3)	350.6	1419.5
100	20	0.53	0.65	0.83	1043.9 (± 45.8)	1073.8 (± 45.8)	1158.6 (± 12.6)	362.2	1147.1

TABLE IV: Experimental results

results in terms of utility for the system, runtime and number of messages exchanged for the decentralized version (this metric has no meaning for the centralized approach). For space reasons, only the results for feasible scenarios are presented.

Not surprisingly, the utilities of the solutions generated by the decentralized approaches are below those of the centralized approach (that are optimal, except for cases when the time limit is reached). However, impressively, they are rather close to that optimum. *FICSAM* solution utilities, being the first feasible coalition structures agents find, are below those of *IFICSAM* solutions where agents continue searching for other feasible coalition structures with better utilities. The utilities of *FICSAM* are consistently above 50% of the utilities of the centralized approach. The utilities of *IFICSAM* are always above 70% of the utilities of the centralized approach, and are at 75% from optimum on average. We can observe that, for a large number of agents (50 or 100), performance slightly degrades. We believe that this may result from difficulty in finding an initial solution (as discussed below).

Notice that our algorithms terminate quickly. Surprisingly, for the largest instance of 100 agents and 20 tasks, despite the message exchange overhead and the computations performed by disparate agents, both *FICSAM* and *IFICSAM* are faster than the centralized algorithm. Further, the latter, given a 1200 seconds timeout, sometimes terminates without reaching an optimal solution. The runtime varies across scenarios. It depends not only on the number of agents and the number of tasks, but also on scenario complexity. For instance, in scenarios with larger numbers of tasks, a smaller average number of agents is needed for each task. Therefore, in some cases, the problem becomes simpler as its combinatorial complexity is lower, which favors decentralized algorithms. For instance, for 50 agents, scenarios with 20 tasks take less time than with 10 tasks.

However, additional tasks, keeping the number of agents intact, increase the difficulty for agents to find a first solution and send an anytime message to other agents. This can explain the bigger the drop in the number of exchanged messages where we have more tasks for the same number of agents. When the number of tasks is very small, the problem is inverted; the number of agents required for each task is larger. This agent multiplicity produces many symmetries among the variables representing the agents in the centralized COP solution, imposing additional computation. This can explain why the centralized algorithm, for 50 and 100 agents, requires more time to solve scenarios with 2 tasks compared to scenarios with 5 tasks. A deeper study of the impact of scenario variations on the complexity of constrained allocation problems is called for. We leave this for future work.

As mentioned above, a part of generated scenarios are infeasible whenever task requirements cannot be covered by the generated set of agents. In such cases, the number of exchanged messages in our mechanism is always twice the number of agents. This results from the number of token passing messages, to which we add the number of end messages with the mention failure. Moreover, *FICSAM* and *IFICSAM* take significantly less time than the centralized algorithm to terminate. For 20 agents and 10 tasks, for example, they terminate after 70 seconds on average, while the centralized algorithm takes 400 seconds. For 50 agents and 10 tasks, they terminate after 360 seconds while the centralized, interrupted by the timeout, takes 1200 seconds. This observation sheds light on the cost of computing an unsatisfiability certificate by COP methods. This cost appears significantly larger than the computational cost exhibited by the decentralized approaches presented in this paper.

V. Related Work

Task allocation to groups of agents has been addressed by several approaches, including coalition formation methods [19]. Among the classes of task allocation problems defined by [9], the case we address in this paper falls in the category of single-task robots and multi-robot tasks (ST-MR).

Some coalition formation studies address the problem of interdependent coalitions via Partition Function Games (PFGs) [13], in contrast with Characteristic Function Games (CFGs). In PFGs, a coalition's value depend not only on the identity of its members but also on the way non-members are

partitioned. Therefore, computing coalition structures in PFGs is very challenging: given an arbitrary partition function, an exhaustive search is required to provide an optimal coalition structure [15] unless additional assumptions – externalities – are provided. Indeed, [17] define a specific kind of externalities that represent inter-coalition effects (e.g., assumptions on utility functions, and coalition mergers). That solution approach is inapplicable in our case, where each task is associated with one coalition, as the set of tasks dictates a fixed number of coalitions, and mergers are therefore irrelevant. Other similar coalition structure generation solutions [16] focus on complexity reduction, however distribution and partial information are usually not their main focus.

Several approaches were proposed to solve the Partition Function Form Game problem [10]. However, these methods are centralized, and the notion of agents is either absent or subject to a centralized allocation. The agents are not autonomous and do not make their own decisions: their orders are provided by the centralized planning agent. Such a centralized approach is inapplicable in our case.

Multiple methods have been used to solve CFGs. One approach is to rely on Constraints Optimization Problems (COP) and Constraint Satisfaction Problems [8] and their solutions, to find suitable ways to form coalitions, while enforcing constraints on the coalition structure. For instance, in [18], task allocation with spatial and temporal constraints is presented. That method allocates agents to tasks so that coalitions are feasible w.r.t. the locations, tasks workloads, deadlines, and the number of completed tasks is maximized. However, the method is centralized and does not generalize to other constraints. CSPs have also been used in other contexts relevant to task allocation.

Studies that address the interdependent task allocation problem mainly focus on the specific case of temporal interdependencies, where constraints of precedence and sequentiality are considered (see e.g. [2], [3], [5], [7]). Our study addresses diverse interdependencies, not necessarily temporal.

VI. Conclusion

We propose a novel decentralized approach for dealing with the problem of coalition formation for task allocation. Given the exponential complexity of finding optimal solutions, we opted for an anytime approach implemented in two stages that gradually improves solution quality or prove unsatisfiability. While many practical solutions address only specific types of task interdependence (or none at all), our solution is not limited to specific interdependencies. Furthermore, our solution explicitly handles diverse agent characteristics and task requirements. It facilitates both qualitative and quantitative agent characteristics and task requirements information, allowing matching thereof. By using CSP-based techniques, we provide an efficient way to deal with large scale instances of our running problem. To illustrate the added value of our approach we ran an extensive number of experiments, with a variety of numbers of agents and tasks, that have proven that our approach is very efficient both for finding a first solution at the end of the first stage (i.e., the end of the first round) and then to significantly improve it during the second stage (through several rounds). The algorithm can be interrupted at anytime, yet it will always return a solution if the problem is feasible. In future work we plan to apply our approach in different real world application problems. Several domains may benefit from our approach: for instance, the coordination of mobile or fixed radars, or the coordination of rovers or Autonomous Underwater Vehicles (AUVs) for de-mining. We specifically aim to apply this approach in application domains of the Thales Group corporation and develop it towards deployment in large programs (*e.g.* Maritime Mine Counter Measures).

References

[1] S. Agnisarman, S. Lopes, K. Madathil, K. Piratla, and A. Gramopadhye. A survey of automation-enabled human-in-the-loop systems for infrastructure visual inspection. *Automation in Construction*, 97:52–76, 2019.

[2] Z. Beck, W. Teacy, N. Jennings, and A. Rogers. Online planning for collaborative search and rescue by heterogeneous robot teams. In *Proc. of AAMAS*, 2016.

[3] J. K. Behrens, R. Lange, and M. Mansouri. A constraint programming approach to simultaneous task allocation and motion scheduling for industrial dual-arm manipulation tasks. In *Proc. of ICRA*, pages 8705–8711. IEEE, 2019.

[4] D. Bouyssou, T. Marchant, M. Pirlot, P. Perny, A. Tsoukias, and P. Vincke. *Evaluation and decision models: a critical perspective*, volume 32. Springer Science & Business Media, 2000.

[5] A. Brutschy, G. Pini, C. Pinciroli, M. Birattari, and M. Dorigo. Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous agents and multi-agent systems*, 28(1):101–125, 2014.

[6] G. Chu, P. J. Stuckey, A. Schutt, T. Ehlers, G. Gange, and K. Francis. Chuffed, a lazy clause generation solver. https://github.com/chuffed/chuffed, 2018.

[7] T. S. Dahl, M. Matarić, and G. S. Sukhatme. Multi-robot task allocation through vacancy chain scheduling. *Robotics and Autonomous Systems*, 57(6-7):674–687, 2009.

[8] R. Dechter. *Constraint processing*. Elsevier Morgan Kaufmann, 2003.

[9] B. P. Gerkey and M. J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *IJRR*, 23(9):939–954, 2004.

[10] L. Kóczy. *Partition Function Form Games: Coalitional Games with Externalities*. Springer, 2018.

[11] L. S. Marcolino, A. X. Jiang, and M. Tambe. Multi-agent team formation: Diversity beats strength? In *Proc. of 23rd IJCAI*, page 279–285. AAAI Press, 2013.

[12] T. Michalak, J. Sroka, T. Rahwan, M. Wooldridge, P. McBurney, and N. R. Jennings. A distributed algorithm for anytime coalition structure generation. In *Proc. of AAMAS*, page 1007–1014, 2010.

[13] R. B. Myerson. Values of games in partition function form. *International Journal of Game Theory*, 6:23–31, 1977.

[14] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack. MiniZinc: Towards a Standard CP Modelling Language. In C. Bessière, editor, *CP'07*, pages 529–543, 2007.

[15] F. Präntare and F. Heintz. An anytime algorithm for optimal simultaneous coalition structure generation and assignment. *Autonomous Agents and Multi-Agent Systems*, 34(1):1–31, 2020.

[16] T. Rahwan, T. Michalak, M. Wooldridge, and N. Jennings. Coalition structure generation: A survey. *Artificial Intelligence*, 229:139–174, 2015.

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

[18] S. D. Ramchurn, M. Polukarov, A. Farinelli, N. Jennings, and C. Trong. Coalition formation with spatial and temporal constraints. In *AAMAS*, pages 1181–1188, 2010.

[19] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial intelligence*, 101(1-2):165–200, 1998.

[20] L. Vig and J. A. Adams. Multi-robot coalition formation. *IEEE Transactions on Robotics*, 22(4):637–649, 2006.