Interdependent task allocation via coalition formation for cooperative multi-agent systems

Douae Ahmadoun

Université Paris Cité

École doctorale Informatique, Télécommunications et Électronique (EDITE) 130

Laboratoire d'Informatique Paris Descartes (LIPADE)

Interdependent Task Allocation via Coalition Formation for Cooperative Multi-Agent Systems

Par Douae AHMADOUN

Thèse de doctorat d'Intelligence Artificielle

Dirigée par Pr. Pavlos MORAITIS

Présentée et soutenue publiquement le 20/01/2022

Devant un jury composé de :

Pavlos MORAITIS (Professeur), Université de Paris, directeur de thèse
Elise BONZON (MCF), Université de Paris, co-encadrante
Cédric BURON (PhD), Thales Research and Technology, encadrant industriel
Pierre SAVÉANT (PhD), Thales Research and Technology, encadrant industriel
Alexis TSOUKIAS (DR, CNRS), PSL, Université Paris Dauphine, CNRS, président
Georgios CHALKIADAKIS (Associate Professor), School of Electrical and Computer Engineering, Technical University of Crete, Grèce, rapporteur
Juan Antonio RODRIGUEZ AGUILAR (Professeur), Artificial Intelligence Research Institute, Espagne, rapporteur
Aurélie BEYNIER (MCF, HDR), Sorbonne Université, examinatrice
Onn SHEHORY (Professeur), Bar Ilan University, Israel, examinateur

Titre : Allocation de tâches interdépendantes via la formation de coalitions pour les systèmes multi-agents coopératifs

Résumé :

L'allocation des tâches à plusieurs agents autonomes devant accomplir des tâches complexes a été l'un des domaines de recherche récents sur les systèmes multi-agents. Dans de nombreuses applications, les agents sont coopératifs et doivent effectuer des tâches qui nécessitent chacune une combinaison de différentes capacités dont peut se doter un sous-ensemble d'agents. Dans ce cas, nous pouvons utiliser la formation de coalitions comme paradigme pour affecter des coalitions d'agents à des tâches. Les solutions à ce problème d'allocation de tâches, pour les systèmes robotiques en particulier, trouvent plusieurs applications dans le monde réel et prennent de plus en plus de l'importance dans les domaines de la défense, de l'espace, de la gestion des catastrophes, de l'exploration sous-marine, de la logistique, de la fabrication de produits et de l'assistance dans les services de santé.

De multiples mécanismes de formation de coalitions et d'allocation de tâches ont été introduits dans l'état de l'art, tenant rarement compte des tâches interdépendantes. Cependant, il est récurrent de trouver des tâches dont la qualité ne peut être évaluée sans considérer les autres tâches dans des applications réelles. Ces tâches sont appelées interdépendantes par opposition aux tâches indépendantes qui, elles, peuvent être évaluées individuellement, ce qui entraîne une évaluation globale de l'allocation des tâches qui additionne simplement toutes les évaluations des tâches.

La recherche dans le passé a conduit à de nombreuses méthodes d'allocation de tâches qui traitent le cas des tâches indépendantes sous différents angles et sous différents paradigmes. D'autres travaux résolvent le cas des tâches interdépendantes, mais ils le font soit de manière centralisée avec une complexité très élevée, soit uniquement pour le cas des dépendances de précédence. Cependant, de nombreuses formes d'interdépendance peuvent exister entre les tâches dans les applications du monde réel. Ces applications nécessitent que les mécanismes d'allocation des tâches soient décentralisés et anytime, pouvant renvoyer une solution à tout moment quitte à l'améliorer s'il reste du temps, pour répondre à des problèmes de sensibilité au temps et de robustesse.

Dans cette thèse, nous considérons des environnements multi-agents coopératifs où les tâches sont multi-agents et interdépendantes, et les méthodes d'allocation des tâches doivent être décentralisées et anytime. À cet égard, nous proposons une formalisation du problème qui considère les attributs qualitatifs et quantitatifs des agents et des tâches, et qui capture les dépendances des tâches que ça soit au niveau des exigences ou au niveau de l'évaluation des allocations. Nous introduisons une nouvelle approche avec un mécanisme de formation de coalition décentralisé anytime qui permet aux agents dotés de capacités complémentaires de former, de manière autonome et dynamique, des structures de coalitions faisables qui accomplissent une tâche globale et composite. Cette approche est basée sur la formation d'une structure de coalition faisable permettant aux agents de décider quelle coalition rejoindre et donc quelle tâche accomplir afin que toutes les tâches soient faisables. Ensuite, les structures formées sont progressivement améliorées via des remplacements d'agents pour optimiser l'évaluation globale de l'allocation, le but étant d'accomplir les tâches avec les meilleures performances possibles. Nous analysons la complexité de nos algorithmes et montrons que, bien que le problème général soit NP-complet, notre mécanisme fournit une solution dans un temps acceptable. Des scénarios d'application simulés sont utilisés pour démontrer la valeur ajoutée de notre approche.

Mots clefs :

Allocation de tâches ; formation de coalition ; systèmes multi-agents ; tâches interdépendantes ; programmation par contraintes

Title : Interdependent Task Allocation via Coalition Formation for Cooperative Multi-Agent Systems

Abstract :

Task allocation among multiple autonomous agents that must accomplish complex tasks has been one of the focusing areas of recent research in multi-agent systems. In many applications, the agents are cooperative and have to perform tasks that each requires a combination of different capabilities that a subset of agents can have. In this case, we can use coalition formation as a paradigm to assign coalitions of agents to tasks. For robotic systems, in particular, solutions to this task allocation problem have several and increasingly important real-world applications in defense, space, disaster management, underwater exploration, logistics, product manufacturing, and support in healthcare facilities support.

Multiple coalition formation and task allocation mechanisms were introduced in the prior art, seldom accounting for interdependent tasks. However, it is recurrent to find tasks whose quality cannot be evaluated without considering the other tasks in real-world applications. These tasks are called interdependent in contrast to independent tasks that can be individually assessed, resulting in a global evaluation of the tasks' allocation that sums all the tasks' evaluations. Research in the past has led to many task allocation algorithms that address the case of independent tasks from different angles and under different paradigms. Other works solve the case of the interdependent tasks, but they do it either centrally with very high complexity or only for the case of precedence dependencies. However, many forms of interdependence may exist between tasks in real-world applications. In addition, these applications need task allocation mechanisms to be decentralised and available at anytime to allow them to return a solution at any time and to improve it if there is time left, to respond to their time-sensitivity and robustness issues.

In this dissertation, we consider cooperative multi-agent environments where tasks are multi-agent and interdependent, and task allocation methods have to be decentralized and available at anytime. In this regard, we propose a problem formalisation that considers the agents' and the tasks' qualitative and quantitative attributes and captures the tasks' dependencies on the requirements level and the allocation evaluation level. We introduce a novel approach with a token-passing anytime decentralised coalition formation mechanism. The approach enables agents with complementary capabilities to form, autonomously and dynamically, feasible coalition structures that accomplish a global, composite task. It is based on forming a feasible coalition structure that allows the agents to decide which coalition to join and thus which task to do so that all the tasks can be feasible. Then, the formed structures are incrementally improved via agent replacements to optimise the global evaluation. The purpose is to accomplish the tasks with the best possible performance. The analysis of our algorithms' complexity shows that although the general problem is NP-complete, our mechanism provides a solution within an acceptable time. Simulated application scenarios are used to demonstrate the added value of our approach.

Keywords :

Task allocation ; coalition formation ; multi-agent systems ; interdependent tasks; constraint programming

Interdependent Task Allocation via Coalition Formation for Cooperative Multi-Agent Systems

by

Douae AHMADOUN

A thesis submitted to

University of Paris

in fulfilment of the requirements for the degree of

Doctor of Philosophy

in

Artificial Intelligence

Defense date: 20 January 2022

Director

Pavlos Moraitis (Professor)
University of Paris, France

Supervisors

Elise Bonzon (MCF)
University of Paris, France

Cédric Buron (PhD)
Thales Research and Technology, France

Pierre Savéant (PhD)
Thales Research and Technology, France

Jury

Alexis Tsoukiàs (DR, CNRS), Chair
PSL, Paris Dauphine University, CNRS, France

Georgios Chalkiadakis (Associate Professor), Reviewer
School of Electrical and Computer Engineering, Technical University of Crete, Greece

Juan Antonio Rodriguez Aguilar (Professor), Reviewer
Artificial Intelligence Research Institute, Spain

Aurélie Beynier (MCF, HDR), Examiner
Sorbonne Université, France

Onn Shehory (Professor), Examiner
University of Bar Ilan, Israel

Declaration of Original and Sole Authorship

I, Douae AHMADOUN, declare that this thesis entitled *Interdependent Task Allocation via Coalition Formation for Cooperative Multi-Agent Systems* and the data presented in it are original and result from my own work.

I confirm that:

- No part of this work has previously been submitted for a degree at this or any other university,
- References to the work of others have been clearly acknowledged. Quotations from the work of others have been clearly indicated, and attributed to them,
- In cases where others have contributed to part of this work, such contribution has been acknowledged and distinguished from my own work.

Date:
Signature:

Acknowledgments

My research study has been conducted both in academic and industrial environments: LIPADE (*Laboratoire d'Informatique de Paris DEscartes*) at University of Paris and LDO (*Laboratoire de Décision et d'Optimisation*) at Thales' research center in France, Thales Research and Technology. I am thankful to both institutions for allowing me to join their teams and pursue this degree. Not only was it an incredibly enriching experience scientifically, intellectually, and professionally, I highly cherish the growth I gained and the life lessons I learned these three years. Although the road was long and challenging at times, I am grateful for every moment that brought me to this place and the person I am today.

In carrying out this Ph.D. project, several individuals played major roles in helping me make it to the end. This work would not have been possible without the unstinting support of many people I profoundly thank.

First and foremost, I would like to express my special acknowledgment to my esteemed academic director and co-supervisor, Pavlos Moraitis and Elise Bonzon, for their invaluable supervision, immense patience, and generous knowledge sharing. I express my appreciation to all LIPADE's staff and colleagues for their help and kindness.

Many thanks go to my Thalesian colleagues for always being present, supportive, and open to communicate their expertise. I am fortunate to have had great industrial supervisors, Cédric Buron and Pierre Savéant, whom I thank for their unlimited support and thoughtful mentoring. I plainly convey my words of acknowledgements to Cédric Buron, Claude Bouscarle, and Florence Aligne for believing in me since day one.

I am also very grateful for Aurélie Beynier, Georgios Chalkiadakis, Juan Antonio Rodriguez Aguilar, Onn Shehory, and Alexis Tsoukiàs. They donated their valuable time and offered me insightful comments and brilliant constructive criticisms on my research. They made me follow their high research standard and see my work from the domain's experts' perspective. Special thanks to Onn

Shehory, co-author and collaborator, who played a key role in developing this dissertation.

In appreciation for a lifetime of support and encouragement, I thank my parents, Samira and Abdelhakim. They taught me how noble science is, believed in me, and gave me the confidence to pursue this dream. I am also grateful to my brothers, Mohamed and Atae, and my little sister, Lina, for their infinite unconditional love and support. You all inspire me a lot.

I am also extremely thankful to my friends, among whom Ali, Reda, Fadoua, Diyaa, Mohamed, Ayoub, Ismail, Anas, Zakaria, Oussama, Afaf, Oumaima, Aimane, Fabien, Denis, Abderrahmane, Sirine and Sanae. I appreciate your care, backup, listening, and efforts for me to stay happy through those difficult years. I also want to express my tremendous gratitude to my Thales fellow students Quentin, Roman, Erwann, Adam, Paul, Edgar, and Chahinez for creating such an enjoyable working environment. I cherish all the unforgettable moments that we had.

I finally cannot skip thanking all the professors and teachers I have ever had for their generosity and contribution to my learning journey.

Dedication

Dedicating it to old, present and future me,
none of whom none the wiser.

Abstract

Task allocation among multiple autonomous agents that must accomplish complex tasks has been one of the focusing areas of recent research in multi-agent systems. In many applications, the agents are cooperative and have to perform tasks that each requires a combination of different capabilities that a subset of agents can have. In this case, we can use coalition formation as a paradigm to assign coalitions of agents to tasks. For robotic systems, in particular, solutions to this task allocation problem have several and increasingly important real-world applications in defense, space, disaster management, underwater exploration, logistics, product manufacturing, and support in healthcare facilities support.

Multiple coalition formation and task allocation mechanisms were introduced in the prior art, seldom accounting for interdependent tasks. However, it is recurrent to find tasks whose quality cannot be evaluated without considering the other tasks in real-world applications. These tasks are called interdependent in contrast to independent tasks that can be individually assessed, resulting in a global evaluation of the tasks' allocation that sums all the tasks' evaluations.

Research in the past has led to many task allocation algorithms that address the case of independent tasks from different angles and under different paradigms. Other works solve the case of the interdependent tasks, but they do it either centrally with very high complexity or only for the case of precedence depen-

dencies. However, many forms of interdependence may exist between tasks in real-world applications. In addition, these applications need task allocation mechanisms to be decentralised and available at anytime to allow them to return a solution at any time and to improve it if there is time left, to respond to their time-sensitivity and robustness issues.

In this dissertation, we consider cooperative multi-agent environments where tasks are multi-agent and interdependent, and task allocation methods have to be decentralised and available at anytime. In this regard, we propose a problem formalisation that considers the agents' and the tasks' qualitative and quantitative attributes and captures the tasks' dependencies on the requirements level and the allocation evaluation level. We introduce a novel approach with a token-passing anytime decentralised coalition formation mechanism. The approach enables agents with complementary capabilities to form, autonomously and dynamically, feasible coalition structures that accomplish a global, composite task. It is based on forming a feasible coalition structure that allows the agents to decide which coalition to join and thus which task to do so that all the tasks can be feasible. Then, the formed structures are incrementally improved via agent replacements to optimise the global evaluation. The purpose is to accomplish the tasks with the best possible performance. The analysis of our algorithms' complexity shows that although the general problem is NP-complete, our mechanism provides a solution within an acceptable time. Simulated application scenarios are used to demonstrate the added value of our approach.

Résumé

L'allocation des tâches à plusieurs agents autonomes devant accomplir des tâches complexes a été l'un des domaines de recherche récents sur les systèmes multi-agents. Dans de nombreuses applications, les agents sont coopératifs et doivent effectuer des tâches qui nécessitent chacune une combinaison de différentes capacités dont peut se doter un sous-ensemble d'agents. Dans ce cas, nous pouvons utiliser la formation de coalitions comme paradigme pour affecter des coalitions d'agents à des tâches. Les solutions à ce problème d'allocation de tâches, pour les systèmes robotiques en particulier, trouvent plusieurs applications dans le monde réel et prennent de plus en plus de l'importance dans les domaines de la défense, de l'espace, de la gestion des catastrophes, de l'exploration sous-marine, de la logistique, de la fabrication de produits et de l'assistance dans les services de santé.

De multiples mécanismes de formation de coalitions et d'allocation de tâches ont été introduits dans l'état de l'art, tenant rarement compte des tâches inter-dépendantes. Cependant, il est récurrent de trouver des tâches dont la qualité ne peut être évaluée sans considérer les autres tâches dans des applications réelles. Ces tâches sont appelées interdépendantes par opposition aux tâches indépendantes qui, elles, peuvent être évaluées individuellement, ce qui entraîne une évaluation globale de l'allocation des tâches qui additionne simplement toutes les évaluations des tâches.

La recherche dans le passé a conduit à de nombreuses méthodes d'allocation de tâches qui traitent le cas des tâches indépendantes sous différents angles et sous différents paradigmes. D'autres travaux résolvent le cas des tâches interdé-

pendantes, mais ils le font soit de manière centralisée avec une complexité très élevée, soit uniquement pour le cas des dépendances de précédence. Cependant, de nombreuses formes d'interdépendance peuvent exister entre les tâches dans les applications du monde réel. Ces applications nécessitent que les mécanismes d'allocation des tâches soient décentralisés et *anytime*, pouvant renvoyer une solution à tout moment quitte à l'améliorer s'il reste du temps, pour répondre à des problèmes de sensibilité au temps et de robustesse.

Dans cette thèse, nous considérons des environnements multi-agents coopératifs où les tâches sont multi-agents et interdépendantes, et les méthodes d'allocation des tâches doivent être décentralisées et *anytime*. À cet égard, nous proposons une formalisation du problème qui considère les attributs qualitatifs et quantitatifs des agents et des tâches, et qui capture les dépendances des tâches que ça soit au niveau des exigences ou au niveau de l'évaluation des allocations. Nous introduisons une nouvelle approche avec un mécanisme de formation de coalition décentralisé *anytime* qui permet aux agents dotés de capacités complémentaires de former, de manière autonome et dynamique, des structures de coalition faisables qui accomplissent une tâche globale et composite. Cette approche est basée sur la formation d'une structure de coalition faisable permettant aux agents de décider quelle coalition rejoindre et donc quelle tâche accomplir afin que toutes les tâches soient faisables. Ensuite, les structures formées sont progressivement améliorées via des remplacements d'agents pour optimiser l'évaluation globale de l'allocation, le but étant d'accomplir les tâches avec les meilleures performances possibles. Nous analysons la complexité de nos algorithmes et montrons que, bien que le problème général soit NP-complet, notre mécanisme fournit une solution dans un temps acceptable. Des scénarios d'application simulés sont utilisés pour démontrer la valeur ajoutée de notre approche.

Contents

Declaration of Original and Sole Authorship v

Acknowledgments vii

Dedication ix

Abstract xi

Résumé xiii

Contents xv
List of figures xviii
List of tables xix
List of algorithms xxi

Abbreviations xxv

Notations xxix

1 General Introduction 1
1.1 Introduction 2
1.2 Context and Motivation 2
1.2.1 Task allocation in Multi-Agent Systems 2
1.2.2 Independent and Interdependent Tasks 4
1.2.3 Motivating Case 5
1.3 Background 6
1.3.1 Agents 7
1.3.2 Multi-Agent Systems 9
1.3.3 Cooperative Agent Teams 10
1.3.4 Task Allocation 11
1.3.5 Coalition Formation 12
1.3.6 Decentralisation 12

1.4 Problem Addressed and Research Questions 14
1.5 Research Contributions 17
1.6 Thesis Layout 17

2 Task Allocation State of the Art 19
2.1 Introduction 20
2.2 A taxonomy of task allocation methods 20
2.3 Independent Task Allocation 22
2.3.1 Market-Based Algorithms 22
2.3.2 Distributed Constraint Optimisation Problems 27
2.4 Interdependent Task Allocation 35
2.4.1 Interdependent Tasks in Organization Design 35
2.4.2 Interdependent Tasks in Coalition Formation 37
2.4.3 Temporally Interdependent Tasks 39
2.5 Analysis and Discussion 43
2.6 Conclusion 46

3 Modelling Interdependent Task Allocation 47
3.1 Introduction 48
3.2 Classical Independent Task Allocation Problem Formalisation . 48
3.2.1 Allocation of tasks requiring a single agent 49
3.2.2 Allocation of tasks requiring many agents 49
3.2.3 Discussion 50
3.3 Modelling Interdependent Task Allocation 50
3.3.1 Agents 51
3.3.2 Tasks 51
3.3.3 Fulfilment 54
3.3.4 Coalition Structures 57
3.3.5 Global Utility Function 58
3.4 Conclusion 61

4 Solving the Interdependent Task Allocation problem 63
4.1 Introduction 64
4.2 Our Approach 64

4.3 Feasible Interdependent Coalition Structure Anytime Method (Feasible Interdependent Coalition Structure Anytime Method (FICSAM)) . . . 66
4.3.1 General Process . . . 66
4.3.2 Decision Process . . . 70
4.3.3 Feasible Structure Generation Process . . . 76
4.3.4 Feasibility Check Process . . . 78
4.3.5 Run-Through Example . . . 79
4.4 Improved Feasible Interdependent Coalition Structure Anytime Method (Improved feasible Interdependent Coalition Structure Anytime Method (IFICSAM)) . . . 80
4.4.1 Swap Improvement Process . . . 81
4.4.2 Combinations Improvement Process . . . 84
4.4.3 CSP Improvement Process . . . 85
4.5 Complexity Analysis . . . 93
4.6 Empirical Evaluation . . . 93
4.6.1 The Scenarios Generator . . . 94
4.6.2 Implementation . . . 95
4.6.3 Setup . . . 95
4.6.4 Centralised Solution . . . 96
4.6.5 Results Evaluation . . . 100
4.6.6 Results for the many-to-one swapping improvements . . . 103
4.7 Discussion . . . 105

5 Conclusions & Outlook **107**
5.1 Introduction . . . 108
5.2 Thesis Results . . . 108
5.3 Limitations and Recommendations for Future Work . . . 110

Bibliography **113**

List of Publications **127**

Appendices **129**

A Résumé long en Français **131**
A.1 Introduction . . . 131
A.1.1 Contexte et Motivation . . . 131
A.1.2 Problématique de Recherche . . . 132
A.1.3 Contributions de la Thèse . . . 133
A.1.4 Plan du Résumé . . . 133
A.2 Etat de l'art . . . 134
A.3 Modélisation du problème . . . 135
A.4 Algorithmes et Résultats . . . 136
A.5 Conclusion et Perspectives . . . 141
A.5.1 Conclusion . . . 141
A.5.2 Perspectives pour de futurs travaux . . . 142

B ICTAI paper **145**

List of Figures

1.1 Simple representation of an agent 8
1.2 Task allocation problem for multi-agent tasks with inter-dependencies and heterogeneous cooperative agents 16

2.1 Gerkey's cube for classifying task allocation state of the art methods [Gerkey et al. 2004] 21
2.2 Contractual Network Protocol Sequence Diagram 24
2.3 Consensus Based Auction Algorithm (CBAA) main phases . . . 25
2.4 Interdependence levels on a multi-agent system inspired by [Zhao et al. 2020] 37
2.5 Representation of the different task allocation methods families 43

3.1 Agents and tasks scattered in the grid 55

4.1 First agent process 68
4.2 Global process 70
4.3 Decision process 75
4.4 Example of a tree of Constraint Satisfaction Problems (CSPs) in [Binshtok et al. 2007] 87
4.5 Example of the application tree of algorithm 5 88
4.6 Class diagram of our experiments' implementation 96
4.7 Parameters of the centralised method for the experimentations in the MiniZinc language 98
4.8 Variables, constraints and objective of the centralised method for the experimentations in the MiniZinc language 99

A.1 Processus du premier agent 138
A.2 Processus global 139
A.3 Processus de décision 140

List of Tables

2.1 Example of the utility function for a task allocation problem modeled with DCOP . . . 31
2.2 Summary table of some DCOP algorithms characteristics (Optimality, Runtime and Communication), taken from [Fioretto et al. 2018] . . . 34
2.3 Summary table of multi-agent task allocation approaches with their classification in Gerkey's taxonomy and characteristics regarding completeness and complexity . . . 45

3.1 Example of agent characteristics . . . 51
3.2 Example of single task requirements . . . 52
3.3 Example of task combination requirements . . . 54
3.4 Manhattan distance between agents and tasks in the example . 56

4.1 Description of the algorithm 2's parameters . . . 71
4.2 Example of agents' characteristics . . . 77
4.3 Example of single tasks' requirements . . . 78
4.4 Example data to illustrate the preference specification . . . 90
4.5 Experimental results on average time and utilities for FICSAM and IFICSAM in its swap (1-to-1) variant compared with the centralised method . . . 100
4.6 Experimental results on the average number of messages for FICSAM and IFICSAM in its swap (1-to-1) variant compared with the centralised method . . . 102
4.7 Experimental results on average time and utilities for FICSAM and IFICSAM in its two variants swap and bestcombinations (1-to-1 and many-to-many) compared with the centralised method 104

List of Algorithms

1 coalition-formation($a_i, T, A, \Gamma_T, \Lambda_{cbt}, msg(perf(a_l, a_i, < content >))$) 67
2 decide($a_i, X_{a_i}, T, \Gamma_T, \Lambda_{cbt}, S, X_S, R, nd, A, is\_feasible(S)$) 73
3 improve_ificsam_swap(a_i, T_{a_i}, S) 82
4 improve_ificsam_combinations(a_i, T_{a_i}, S) 84
5 best_csp($a_i, t_j, H_{t_j}, P_{t_j}, A, X_A$) 91
6 improve_ificsam_bestcombinations(a_i, T_{a_i}, S) 92

Abbreviations

All abbreviations used in the thesis are listed here in alphabetical order. Although the abbreviations are explained through the text at their first appearance, this assists the reader who may not read all thesis sections.

ADOPT	Asynchronous Distributed OPTimisation
AI	Artificial Intelligence
BB-CSP	Branch and Bound over Constraint Satisfaction Problems
CBAA	Consensus Based Auction Algorithm
CBBA	Consensus Based Bandle Algorithm
CBC	Coin-OR Branch-and-Cut
CFG	Characteristic Function Game
CFSTP	Coalition Formation with Spatial and Temporal constraints Problem
COP	Constraint Optimisation Problem
CSP	Constraint Satisfaction Problem
DAI	Distributed Artificial Intelligence
DCOP	Distributed Constraint Optimisation Problem
Dec-POMDP	Decentralised Partially Observable Markov Decision Process
D-Gibbs	Distributed Gibbs
DSA	Distributed Stochastic Algorithm
DUCT	Distributed Upper Confidence Tree

FMS	Fast Max Sum
FICSAM	Feasible Interdependent Coalition Structure Anytime Method
HD	High Definition
IA	Instantaneous Allocation
IFICSAM	Improved feasible Interdependent Coalition Structure Anytime Method
IoT	Internet of Things
MA	Multi-Agent
MAA*	Multi-Agent A-star
MA-MDP	Multi-Agent Markov Decision Process
MAS	Multi-Agent System
MCDM	Multi-Criteria Decision Making
MDP	Markov Decision Process
MGM	Maximum Gain Message
MIP	Mixed-Integer Programming
MR	Multi-Robot
MRTA/TOC	Multi-Robot Task Allocation with Temporal and Ordering Constraints
MT	Multi-Task
OR	Operations Research
PFG	Partition Function Game
POMDP	Partially Observable Markov Decision Process
POSG	Partially Observable Stochastic Game

SAT	SATisfiability problem
SPoF	Single Point of Failure
SR	Single-Robot
ST	Single-Task
STAAMS	Simultaneous Task Allocation and Motion Scheduling
SyncBB	Synchronized Branch and Bound
TA	Time extended Allocation
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle
UMRTA	Uncertain Multi-Robot Task Allocation
UTA	Additive UTilities
UUV	Unmanned Underwater Vehicle
VCS	Vacancy Chain Scheduling

Notations

$\top$	Boolean True
$\bot$	Boolean False
A	Set of agents
T	Set of tasks and global task
a_i	An agent
s	A subset of agents
X	Set of agents characteristics
X_{a_i}	Instantiation of agent a_i characteristics
x_k	An agent characteristic
s	A set of agents
X_s	Instantiation of characteristics of agents in the subset s
D_k	Domain definition of the agent characteristic x_k
T	Set of tasks, global task
t_j	A task
Γ_{t_j}	Set of requirements for task t_j
γ_l	A task requirement
E_l	Domain definition for the attribute of the task requirement γ_l
T_{cbt}	The set of the possible task combinations
cbt_j	A task combination
Λ_{cbt_j}	Set of task combination requirements for task combination cbt_j
λ_l	A task combination requirement

F_l	Domain definition of the task combinations requirement λ_l
$\bowtie$	Requirement satisfaction
$\circlearrowleft$	Fulfilment relation
T_{a_i}	The set of tasks that agents a_i can contribute too
τ	Assignment function
C_{t_j}	The coalition of agents assigned to task t_j
u_{global}	Global utility
S	A coalition structure
S^f	A feasible coalition structure
CS	The set of all possible coalition structures
$\oplus$	The addition function of an agent to a specific coalition in a coalition structure
$\ominus$	The removal function of an agent from a specific coalition in a coalition structure
G	A set of criteria
g_p	A criterion
G_p	A week order linked to the criterion g_p
R	The set of agents rounds
nd	The number of unchanged decisions
S_{max}	A feasible coalition structure with the maximum utility found
S_{old}	A coalition structure the agent had before receiving the token
S_{new}	A new coalition structure the agent is constructing
cb	A combination of agents
H_{t_j}	The set of hard constraints to realize task t_j
C_{t_j}	The set of soft constraints to realize task t_j
Q	Queue
N	Node

1 General Introduction

Contents

1.1 Introduction **2**
1.2 Context and Motivation **2**
1.2.1 Task allocation in Multi-Agent Systems 2
1.2.2 Independent and Interdependent Tasks 4
1.2.3 Motivating Case 5
1.3 Background **6**
1.3.1 Agents 7
1.3.2 Multi-Agent Systems 9
1.3.3 Cooperative Agent Teams 10
1.3.4 Task Allocation 11
1.3.5 Coalition Formation 12
1.3.6 Decentralisation 12
1.4 Problem Addressed and Research Questions **14**
1.5 Research Contributions **17**
1.6 Thesis Layout **17**

1.1 Introduction

This chapter provides a prelude to the work presented in this dissertation. First, the motivation, the research challenges, and the main contributions are briefly outlined. Also, the fundamental concepts examined in this work are introduced and reviewed. The explored research questions are pointed out, as well as the contributions stemming from this Ph.D. project. The chapter concludes with an overview of the thesis' structure and content of the thesis, along with the contributions with which we answer the research questions.

1.2 Context and Motivation

In this section, we present the global context in which this thesis was done. We also underline the core concepts used in this thesis. Finally, we highlight a brief summary of a part of the work that cannot be exhibited in this manuscript, which inspired the research directives of this thesis.

The motivational context of the thesis involves several concepts and notions that are introduced in section 1.3 in detail.

1.2.1 Task allocation in Multi-Agent Systems

In today's era of rising technological applications such as robotics, autonomous engines – be they air-bound such as Unmanned Aerial Vehicles (UAVs), ground-bound such as Unmanned Aircraft Systems (UASs) or water-bound such as Unmanned Underwater Vehicles (UUVs) – and home appliances, there has been a renewed interest in multi-agent task allocation technologies. Indeed, the decrease in robots and drones production costs on the one hand, and the advances in sensor technologies, communication protocols and computational capacities on the other hand, encourage the usage of these technologies and accelerate their advent.

In particular, it is interesting to employ these machines in missions that qualify as complex, critical and/or time-sensitive. A mission is defined by a set of tasks that compose the global task entrusted to the machines. We use "mission" and "global task" as equivalent terms for the rest of this document. Indeed, a system with multiple robots has many advantages over single robot systems. Some configurations are difficult or impossible to handle by a single robot yet much

easier to solve using a team of robots. These missions typically involve different tasks each requiring different skills, tools or resources. In general, these tasks are complementary and inter-related in terms of shared resources, execution time or efficiency.

Taking full advantage of these robots requires an embodied intelligence that makes decisions about their cooperation and organises their ability to work as a team of individuals sharing a common goal: to achieve their mission efficiently. This intelligence enables the robots to coordinate their actions by allocating tasks or workloads according to their capabilities to be used most conveniently and efficiently for the problem at hand. To do that, a robot needs information about the tasks' requirements and the evaluation of these tasks realisation by particular groups of agents. This way, they can make sure to accomplish the tasks if they have enough capabilities to do it, and more than this, to complete them with the best possible performance based on tasks evaluation.

In many real-world applications, the evaluation of these tasks cannot be done for each one of them individually. This happens, for instance, when two tasks require a shared resource, that must be present either among the agents dedicated to the first task or among those dedicated to the second one. In such a case, the two tasks cannot be evaluated independently: a task cannot be evaluated without considering the other. This also occurs when tasks have some ordering constraints. For example, a task may have another task as a precondition, meaning that the latter must be completed before the task can be performed. It may also happen that the quality of execution of the second task depends on the quality of execution of the first one.

As an example, consider the case of a search and rescue mission. A search task is first done by agents that have good vision technology. Those agents gather information about the number and localisation of the victims in a specific area. Based on this information, a number of agents with specific arms perform a rescue task to retrieve the victims. The quality of the rescue task execution depends on the information received from the agents that performed the search task. If these agents were not enough, had bad vision cameras or not enough fuel to stay longer in the area, they may have reported incomplete or inaccurate information. This directly affects the quality of the rescue task, which may send fewer agents than required and may not rescue all the victims.

1.2.2 Independent and Interdependent Tasks

The task allocation literature differentiates between two different situations. In some applications, the mission or global task to be performed is composed of several tasks, but these tasks are entirely independent, in the sense that the quality of execution of a task does not depend on the other tasks but only on the agents assigned to it. By contrast, in other applications, one cannot evaluate the allocation of specific agents to a task without considering other tasks' allocations. This happens, for example, when two tasks share an essential resource, a task's execution quality is influenced by another task's execution quality, or tasks have some execution order. Such tasks are said to be *interdependent*.

Interdependence between tasks can be formalised using the concept of utility. Before explaining this formalisation, let's first precise some notions. We consider utility functions over groups of tasks. Formally, we define $u : 2^T \rightarrow \mathbb{R}$, where T is a set of tasks. For a given set $S = \{t_1, \ldots, t_n\}$ of n tasks in T, the value $u(S)$ is a measure of the quality of the allocation of the tasks in S. In particular, individual tasks have an associated utility and so does the global task, which is defined as the collection of all tasks in T.

Furthermore, utility functions are said to be additive [Brandt et al. 2016] if

$$\forall S_1, S_2 \subseteq 2^T : u(S_1 \cup S_2) = u(S_1) + u(S_2) - u(S_1 \cap S_2) \tag{1.1}$$

Admittedly, when the global task is composed of independent tasks, the utility of the global task is additive. This is reasonable thanks to the tasks' independence that makes it possible to evaluate each task allocation locally without considering any external parameter to the task. Also, there is no intersection between the different tasks. Thus, the utility is indeed additive. This way, the utility additivity suggests that the utility of the global task is the sum of the local utilities among all the tasks composing it, taken individually. Formally:

$$u(T) = \sum_{t_j \in T} u(t_j) \tag{1.2}$$

By contrast, when tasks are interdependent, the tasks utilities cannot be additive. This is due to the impossibility of local evaluation for all the tasks. The reason is that the interdependent tasks influence each others' utilities. Thus, the utility of the global task cannot simply be the sum of the utilities of its parts, which are the tasks. If a global task is composed of interdependent tasks, its

utility can then be calculated only as a whole even if its components (*i.e.* the tasks) are discrete.

Besides, task interdependence can be present in different ways. We can have two considerations of the notion of interdependence: interdependence on tasks feasibility constraints and interdependence in the global task allocation utility.

Principally, the first consideration represents cases where two or more tasks cannot be accomplished if they do not verify some condition(s) together. If these conditions are not satisfied, the tasks cannot be executed even if their locally defined conditions are satisfied. This can be illustrated for example by the necessity of a shared resource between two tasks. For example, if the first task is to clean the table by one agent and the second one is to clean the floor by another agent, the two tasks need each a towel so that agents assigned to them can do the cleaning. But one of the agents must have the detergent product and pass it to the other so that the two tasks can be completed, since the goal of the two agents team is to clean the room. At the task allocation level, it does not matter who of the two agents has this shared resource as long as one of them has it.

Likewise, the second consideration of interdependencies is expressed in the different possible task allocation utilities. In several realistic configurations, a task value, mirroring the performance of the agents assigned to it, might be influenced by the cooperative arrangements of other agents and by their performance on their assigned tasks [T. Sandholm et al. 1999]. For example, it may happen that, in a Search and Rescue mission, the agents allocated to the search task do not report the correct number of victims in a specific area. Consequently, based on the wrong information concluded from the search task due to its agents' bad performance, just a few agents will be allocated to the rescue task, and thus, they might fail in rescuing all the victims. On the contrary, if the search task was well accomplished and its agents succeeded in communicating the accurate number of victims, the rescuers' team has more chances to succeed in their task.

1.2.3 Motivating Case

The work proposed in this thesis, being realised in an industrial framework, is based on an industrial use case scenario. The problem we examined for that use case was task allocation for autonomous, cooperative, and heterogeneous agents that should perform a set of complex tasks.

Assuming tasks are independent, we initially modeled the problem under a specific paradigm that suits our application's particularities. After that, we have chosen a task allocation algorithm [Macarthur et al. 2011] that goes with the specific aspects of our application.

Then, as the application's domain experts were not able to help with the formulation of a utility function, we had to build an approach using Multi-Criteria Decision Making (MCDM) techniques [Siskos et al. 2005].

Along these lines, our end-to-end approach includes making agents allocate themselves to tasks dynamically and measuring the quality of that allocation for critical, dynamic, and complex missions. We have implemented this approach and its associated techniques and launched many experiments on a simulator specifically designed for the use case [Gayraud et al. 2021].

We run empirical experiments on the simulator using the chosen task allocation algorithm with the designed utility function. The experiments results had led to two observations. When tasks did not influence each other, we had promising results in reasonable times. However, the results were unexpected when there were unavoidable influences between tasks and did not correspond to operational forecasts. Analysing this inconsistency between experts' evaluation and the resulting task allocation made us notice the interdependent character of tasks that we had not considered in our modelisation.

With a deeper look into task interdependence, present in several configurations and numerous real-world applications (see next section), we decided to channel the research direction of this thesis towards the case of interdependent tasks.

The work we have done on the use case, including modelisation, suggested approach and results, is documented in a patent [Ahmadoun et al. 2020]. This patent was filed as classified by the French Government Defense Agency. This is why we cannot give further details about it in this dissertation.

1.3 Background

Before exploring the thesis research questions, literature, and contributions, let us set the stage by defining some general concepts related to this project's subject. This section introduces the fundamental vocabulary in the multi-agent task allocation community. Since the goal is to remove ambiguity on the field's

terms by providing basic definitions, acquainted and more advanced readers can skip this section.

1.3.1 Agents

Studying agents is a reasonably recent research domain. Being a part of the largest field of computer science and directly associated with robotic systems, in particular, much attention has been paid to this field as one of the most prominent and promising technologies.

Seemingly due to its novelty, no consensus on the definition of an agent has been reached. Still, numerous proposals, with some slight variations, have been more and more embraced.

One of the most popular and trusted definitions is Wooldridge and Jennings's definition [Wooldridge and Nicholas R Jennings 1995]. The two researchers suggested: "An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its delegated objectives". They link the agent with the ecosystem in which it exists and executes some purposeful actions on its own to reach specific predefined goals. The agent is autonomous, which means it can operate without the intervention of a human or another system. In this way, an agent can be considered as intelligent, and it should have the following characteristics: reactivity (*i.e.*, ability to perceive the environment and react to its changes in a timely manner), proactivity (*i.e.*, ability to take initiatives based on a goal-oriented behaviour), and sociability (*i.e.*, ability to interact with other agents). It is also considered that an agent has a list of actions it can perform to change its environment. The critical problem an agent faces is deciding which of these actions to do to meet its design objectives [Woolridge et al. 2001].

With the same concepts but on a more general level, Russell and Norvig defined an agent as: "Anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors" [Russell et al. 2002]. With this definition, animals like ants, fish, and birds, or even humans are considered agents. We have senses (such as eyes, tongue, hands) that allow us to sense our environment and different organs and body parts as effectors. Also, a software agent is an agent since it collects knowledge about

the environment through the information it receives and encodes bit strings to make actions. Russel and Norvig consider goal-directed behaviour the essence of intelligence. They named a goal-oriented agent who always deliberatively chooses to act with the optimal foreseen outcome, with a term borrowed from economics, a "rational" agent. With this generic definition, a top-level view of an agent and its interaction with its environment can be illustrated as shown in figure 1.1.

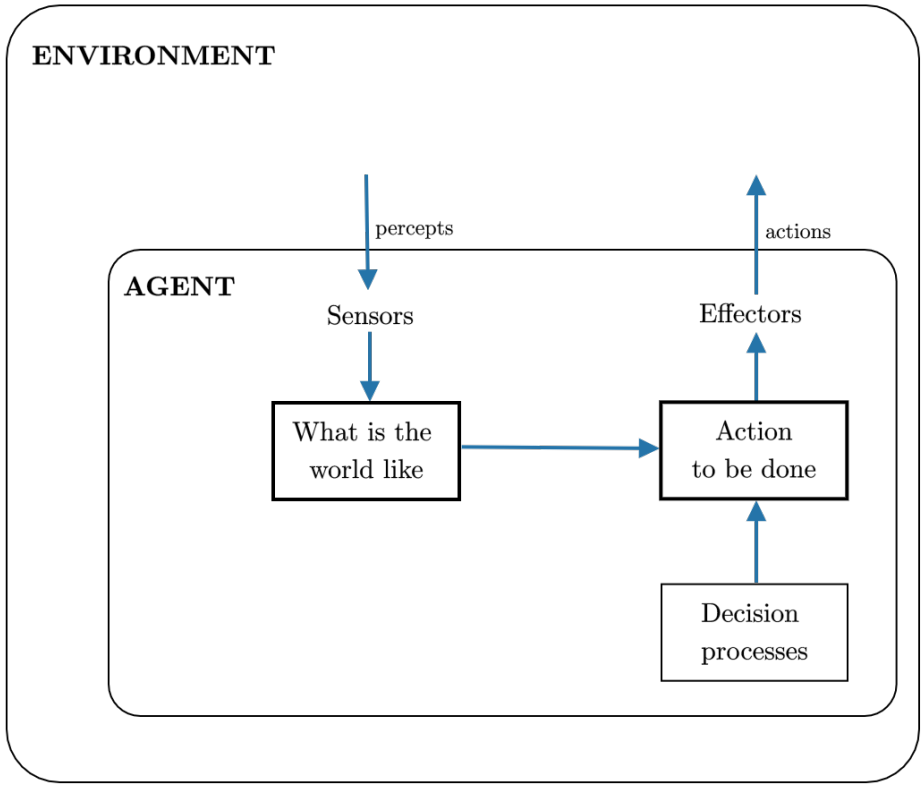


Figure 1.1: Simple representation of an agent

With these two definitions, we implicitly assume that the agents we deal with are intelligent. We will also use the general term "agent" throughout this thesis as a generalisation for "robot" and an abbreviation for "intelligent agent".

1.3.2 Multi-Agent Systems

As its name implies, a Multi-Agent System (MAS) is composed of multiple agents populating the same environment and carrying out their actions in this shared environment. The agents of a MAS are naturally led to interact with each other to achieve their goals. This capability of interaction, whether to set a competitive configuration or to counterbalance each other's deficiencies, makes the MASs applicable to several applications. Some examples include computer networks due to their increasing complexity as a result of edge computing and Internet of Things (IoT) emergence [Kovtunenko et al. 2019; X. Liu et al. 2020; Munir et al. 2019]; robotics as the most intuitive and natural application especially with their growing use nowadays [Kitano 2000; J. Liu et al. 2018; Stone et al. 2000]; complex systems basically to model them as agents for more flexibility and expressivity [Bai et al. 2017; Boes et al. 2017; Rzevski 2012]; business management for optimisation purposes [Coria et al. 2014; J.-H. Lee et al. 2008; Żytniewski 2016]; health care and medical technologies [Moreno 2003; Nealon et al. 2003; Tapia et al. 2009]; and smart grids to address their multiple challenges using agents [Pipattanasomporn et al. 2009; Rogers et al. 2012; Wang et al. 2020].

Additional characteristics of MAS environments are that they can be either static or dynamic. An environment is considered static when it can be assumed it stays unchanged except by its agents' actions. On the other hand, when an environment undergoes changing processes beyond the control of the agents, it is considered dynamic and thus more complex to deal with [Russell et al. 2002]. The dynamics can affect, for example, the agents by adding or removing some of them. It can regard the agents' objectives, which may vary throughout the agents' execution. It may also involve the other elements of the environment or agents' communication by the presence of a specific noise.

As stated before, thanks to their important features, MASs applicability in several domains has significantly improved. Open research questions about the MASs are being approached. These involve, among others, coordination, negotiation, argumentation, communication, security, learning, and task allocation. The task allocation challenge is studied in this thesis and is more long-windedly detailed in section 1.3.4.

We note that in a MAS, agents can be homogeneous or heterogeneous. Namely,

homogeneous agents have the same capabilities and can be interchangeable. By contrast, when agents are heterogeneous, they have different capabilities.

1.3.3 Cooperative Agent Teams

As stated in section 1.3.2, the interactions between MAS agents can take different forms: cooperation, coordination, negotiation, competition. From an agent perspective, we distinguish two types: cooperative agents and competitive agents.

In a cooperative setting, each agent is principally concerned with maximising the social welfare of the entire system, albeit it does not necessarily maximise its personal utility. Cooperative multi-agent systems refer to the particular case of multi-agent systems where agents have to interact with each other to reach common goals. An example would be MASs that intend to resolve a complex problem that an individual agent or a monolithic system cannot solve easily or cannot solve at all.

By contrast, other configurations can involve agents with different stakeholders where each agent has its objectives, preferences, and utilities to maximise regardless of the consequences on other agents. Those agents are called self-interested or competitive agents.

Cooperative multi-agent systems represent a very active research area, as it will be shown in the next chapter. This thesis contributes to a part of it.

From an organisational perspective, the system designer needs to ensure a particular organisation upon the agents. The designer has to set up the agent roles, relationships, and authority structures governing their behaviours. Major organisational paradigms in MASs are reviewed in [Horling et al. 2004]. Among these paradigms, we are interested in the coalitions paradigm that will be more detailed later in section 1.3.5, but also in the teams paradigm. This is because we are interested in cooperative agents who, by design, agreed to work together toward a shared goal [Kaminka et al. 2002; Scerri et al. 2005; Tambe et al. 1999]. In agents teams, agents coordinate their decisions in a manner supported by their individual actions and consistent with their objective as a team.

1.3.4 Task Allocation

Tasks are the actions that agents can perform to achieve their design objectives. In a cooperative setting, the common goal of the agents is a set of different tasks. Each task has some specific requirements to be performed. For example, the task of surveillance for a specific area needs specific cameras and a certain lifespan enabling the agent who performs this task to have enough time to scan the area. In another example, lifting a heavy table needs either two agents with lifting arms and high specific strength or four with lifting arms with a medium strength level, but no more than four agents to avoid overcrowding.

Task allocation is an essential requirement for multi-agent systems operating in cooperative environments. It allows agents to know their individual goals to improve the overall system performance. The objective of task allocation in a MAS is to optimise use of the available resources, namely agents, in the most beneficial way for the MAS application.

A particular case may occur when the available agents cannot in any way perform the present tasks. For example, if there are two surveillance tasks, but among the agents, only one agent has a camera and a very limited remaining lifespan. In this case, no assignment is possible since the two tasks cannot be performed together. Otherwise, performance levels are defined, and the goal is to find the rearrangement with a good performance outcome. This is applicable for cases were many different rearrangements between the agents and the tasks are possible regarding the tasks' requirements and the agents' skills.

The performance level is defined by what we call a utility. The utility is derived by a function that maps a state to a real number describing the level of performance. In our case, a state is a specific tasks' distribution among the agents. This function, being the explicit mirroring of the cooperation quality, expresses the whole system's welfare.

More specifically, a multi-agent mission outcome can be influenced by many factors; this includes the heterogeneity of agents, their different capabilities, tasks disparity, and even auxiliary problems like localisation and navigation.

A utility is defined as a valuation quantity to shape the task allocation problem and measure the likelihood of its success. It synthesises different task allocation model aspects depending on the application. These aspects may include, for instance, execution time, traveled distances, or execution quality.

1.3.5 Coalition Formation

A coalition formation is a temporary grouping of agents into coalitions to perform complex tasks, otherwise unfeasible by a single agent and requiring the association of several agents with different skills.

This concept has been widely studied in the game theory field [T. W. Sandholm et al. 1995; Shenoy 1979]. Its proposed methods are focused on rational agents seeking to maximise their own utility via coalitions. In addition, the first solutions proposed were centralised and computationally exponential. However, Distributed Artificial Intelligence (DAI) researchers have developed coalition formation algorithms that are applicable to MASs and focus on distributing the computations, reducing complexity and making the task allocation efficient. They also have developed algorithms, where the agents, as in our problem, are completely cooperative [Abdallah et al. 2004; Aumann et al. 1974; O. M. Shehory et al. 1997]. Thus, with these solutions, the agents aim to form a coalition that provides the highest overall utility - without considering their individual utility.

Similarly, in a cooperative multi-agent scenario, the process of coalition structure generation aims to produce a distribution of coalitions, each consisting of a subset of agents allocated to a specific task. This way, each task has its own coalition. In addition, in the context of cooperative game theory, many works have used the coalition formation paradigm as the outcome of a cooperative game can be defined as a coalition structure together with a payoff vector [Chalkiadakis et al. 2011].

The coalition formation is an adequate paradigm to model the problem when a problem involves complex tasks, and each needs more than one agent and a specific combination of capabilities or resources to be performed. The aspiration is to allocate tasks to agents and, as a result, regroup assigned agents for each task in a group (aka a coalition). The coalition's agents cooperate to accomplish their assigned task or perform it with a certain level of efficiency. In this way, for each task, the coalition structure, indicating the set of agent coalitions assigned each to a task, is formed via task allocation.

1.3.6 Decentralisation

Many communication disconnections, mission changes or breakdowns may occur in real-world applications. This is why we need multi-agent task allocation

mechanisms enabling several cooperating robots (being considered as agents) to achieve their missions in a completely decentralised and robust manner. The reason is that in a decentralised configuration, all the agents have the capability and the intelligence to make their own decisions. Thus, contrary to a centralised configuration where only a central point can do so, agents are not paralysed and unable to continue their mission execution if an eventual breakdown happens to that central point or if they lose communication with it.

These methods can distribute tasks among robots and assign computation charges to nodes in a network. In addition, in many real-world multi-agent applications, such as human-agent teams [Losey et al. 2020; Zhang et al. 2012], sensor networks [Farinelli, Rogers, Petcu, et al. 2008; Mainland et al. 2005], disaster rescue missions [Beck et al. 2016; Hooshangi et al. 2017], satellite constellations [Schetter et al. 2003; Yao et al. 2019], to whom decentralisation is inherent, centralised decision-making is not practical and sometimes not possible.

Despite the indisputable advantage in the quality of the results achieved by centralised methods over decentralised ones, it is worth considering these results' regarding their solution applicability in many scenarios.

In addition, the centralised methods face the risk of a Single Point of Failure (SPoF). This is when the central point (or computation engine) responsible for the decision-making of the whole MAS breaks down. In this case, the system stays paralysed with no decision nor acting capacities, and thus, the goals cannot be achieved. When all the system's agents have the decision making capability, the system is more autonomous and more robust to the eventual breakdowns.

Besides, centralised methods first demand a global view on the system and strong computation capacities to perform all required calculations and generate the allocation decisions for all the tasks. Hence, it is impossible to implement such method on small engines with limited memory and model computational power, such as drones.

Moreover, even when there are no communication disturbances or noise, the central point must be able to communicate with all the system's agents to give them orders depending on the resulting task allocation. By contrast, in the decentralised methods, all the agents do not have to be connected to one point based on some communication network.

In contexts where communication disturbances are possible, a SPoF problem can happen if the central point loses communication with a part or all of the agents.

Task allocation decentralisation is not adopted to distribute and simplify the computations but to make the suggested solutions more practical, robust, compatible with new technologies, and applicable to real-world applications.

After discussing these elements, we state that a centralized approach does not apply to the problem settings in this thesis.

1.4 Problem Addressed and Research Questions

The aim of this Ph.D. project is to study multi-agent decentralised task allocation methods for tasks with dependencies.

In particular, we focus our research on the case where agents cannot execute more than one task at a time and are heterogeneous by their different capabilities. These agents must realise, cooperatively as a team, a mission composed of complex tasks where each of these tasks requires a subset of agents with a specific combination of capabilities for its achievement. This is why agents should form coalitions to accomplish such tasks.

Furthermore, the tasks are interdependent. The quality of a task does not depend only on the agents assigned to it but can also depend on other tasks. Regarding the mission execution environment, we start by considering a static environment, but we can extend to a dynamic environment where some agents can break down while others can join the mission on the road, and some tasks can fail while new agents can be added during the mission. For this reason, we focus on a decentralised configuration in order to adapt to the nature of the intended applications that can be critical with communication instability and where agents are robots or drones.

In the following, we present the main problem of interest in this thesis along with the questions and the related challenges it raises. The decision process has to be decentralised to make it applicable in real-world applications. Also, it has to be able to propose anytime solutions. This means that agents can have a solution at each moment if the process has to stop, and they can improve its quality if more time is given.

The central problem studied in this Ph.D. research project is :

Multi-Agent decentralised Task Allocation via Coalition Formation for Interdependent Tasks

We can formulate the problem with the question: "How to allocate interdependent tasks to agents and form coalitions in a decentralised manner?"

The problem of multi-agent task allocation can be described as follows. Given a set of tasks and a set of agents, we want to define an association between subsets of agents and tasks to match the capabilities of the agents with the requirements of the tasks and maximise a utility function. To cooperate, accomplish their mission, and ensure good performances, the agents (whether robotic or software ones) need to have the ability to decide which task each agent of the team should perform. This decision should be based on each agent's capabilities, other team agents' capabilities, and the tasks' requirements. The mechanism that allows them to do so must be resilient and robust to the eventual communication problems or changes concerning agents or tasks.

Figure 1.2 illustrates our problem. The problem's purpose is to obtain an online mapping assigning the tasks that form the mission (*i.e.*, the global task) to the agents who form a team. The output must be a set of agent coalitions assigned each to a specific task. The agents are heterogeneous, meaning that they are not inter-replaceable. Also, there are several interdependencies between tasks, and thus the tasks cannot be evaluated separately. The mechanism should be completely decentralised, and each agent must have the capacity to decide based on the information it has and communicate with other agents.

To deal with this problem, the main questions to be addressed are :

- How can the task allocation mechanism assure the satisfaction of the tasks requirements?
- How can agents form coalitions when tasks are interdependent?
- How to measure an allocation quality?
- How to make the allocation mechanism decentralised and anytime?

The tasks we consider in our setting are complex, because they need more than one agent and combinations of specific capabilities to be performed. This

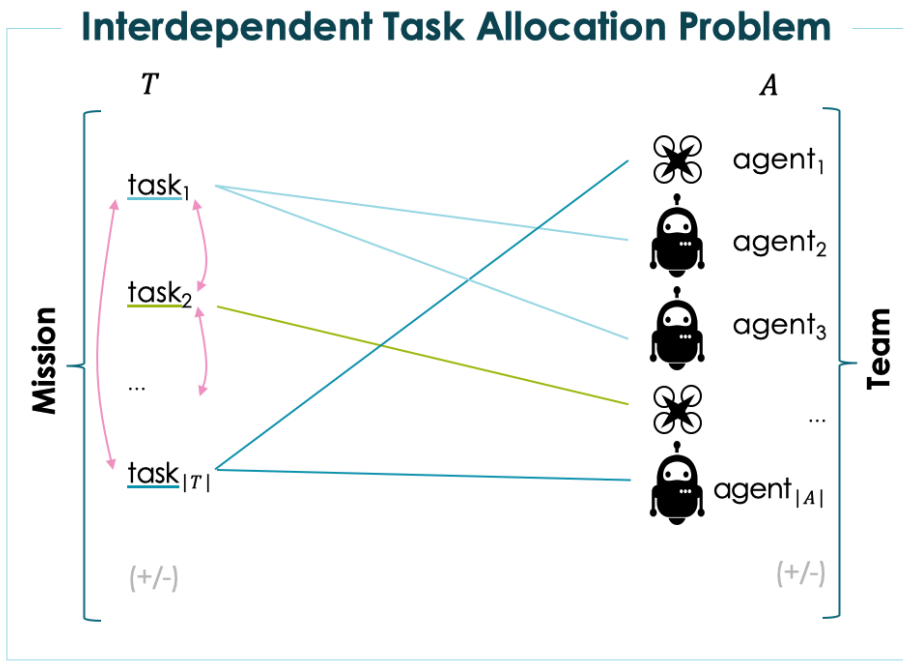


Figure 1.2: Task allocation problem for multi-agent tasks with inter-dependencies and heterogeneous cooperative agents

is our proposed allocation mechanism modeled under the **coalition formation** paradigm. Thus, the adopted approach proposes to form coalitions and assign them to tasks in a way that optimises the global performance.

To measure this performance, we need to model a utility function. This function should represent a numeric value expressing when a system is more or less efficient combining different execution criteria. To aggregate the different execution criteria into one utility function, MCDM techniques are needed. This part is included in the secret patent work.

1.5 Research Contributions

The main contributions of this work can be listed as follows:

- A survey of independent and interdependent task allocation approaches representing different methods families with a detailed analysis on formalisation and resolution aspects and a global comparison based on different factors.
- An end-to-end approach allocating independent tasks in critical, dynamic, and complex missions (hidden by the secret patent).
- An informal approach to guide designers in modelling a utility function for independent tasks when the experts cannot provide one (hidden by the secret patent).
- A generic modeling of interdependent task allocation problems considering both qualitative and quantitative tasks and agents properties and where interdependence appears at two different levels: the tasks requirements and the utility function.
- A novel approach for a feasible coalition structure formation for interdependent task allocation. It introduces two stages algorithm for interdependent task allocation via coalition formation that is anytime and completely decentralised with three different extensions and the inter-agents messages exchange protocol.

These results have been published in a paper [Ahmadoun et al. 2021] and three patents [Ahmadoun et al. 2021; Ahmadoun et al. 2020; Gayraud et al. 2021] one of which is characterised as confidential by the French Government Defense Agency. For this reason, these results cannot appear in this document. Future research papers are initiated as well (see our list of perspectives in section 5.3).

These contributions are presented in a detailed manner in section 5.2.

1.6 Thesis Layout

In the remainder of this thesis, we outline the algorithms presented in the task literature for independent or interdependent tasks. We then present a complete

formalisation of the interdependent task allocation. We finally introduce a new approach with several algorithms to solve the problem of the generation and improvement of feasible coalition formation for interdependent task allocation in a decentralised manner.

This is achieved through the course of the remaining chapters. The following is a general description of this dissertation's contents. The present doctoral thesis is organised into two parts and six chapters: the two parts concern independent and interdependent task allocation problems.

- Chapter 2 presents a survey of the most relevant research work related to the problem discussed in this thesis. Our survey is composed of two parts. The first concerns independent task allocation and examines different method families with a discussion for each of them as related to our problem. The second part exhibits the interdependent task allocation works in the literature. An analysis of the outlined survey concludes the chapter.
- Chapter 3 starts by stating the classical task allocation formalisation for the independent task allocation problem. Then, it proposes a generic and complete formalisation of the interdependent task allocation problem that covers agents and tasks specifics under the coalition formation paradigm.
- Chapter 4 describes our new two stages approach to interdependent task allocation via coalition formation. It presents our new anytime algorithms and mechanisms for the problem that ensures to produce a solution for the task allocation covering all tasks requirements, even the ones representing tasks interdependencies, and to improve its utility. This chapter also presents and discusses the implementation and the experimentation results.
- Finally, chapter 5 provides the conclusions to this work, focusing on the contributions and the limitations of the approaches and algorithms developed. It also identifies the most promising directions for future work that can be carried out to extend and enhance the proposed algorithms.

2 Task Allocation State of the Art

Contents

2.1 Introduction **20**
2.2 A taxonomy of task allocation methods **20**
2.3 Independent Task Allocation **22**
2.3.1 Market-Based Algorithms 22
2.3.2 Distributed Constraint Optimisation Problems . . . 27
2.4 Interdependent Task Allocation **35**
2.4.1 Interdependent Tasks in Organization Design . . . 35
2.4.2 Interdependent Tasks in Coalition Formation . . . 37
2.4.3 Temporally Interdependent Tasks 39
2.5 Analysis and Discussion **43**
2.6 Conclusion **46**

2.1 Introduction

This chapter outlines an overview of the main approaches to the task allocation problem proposed in the literature. We fundamentally focus on a number of the most popular task allocation method families. We highlight the strengths and limitations of each family of methods motivating, thus, the research objectives of this thesis. The discussed methods' characteristics are discussed in-depth, starting from their algorithmic complexity and quality bounds to their classification regarding the large state of the art. This review chapter first covers the task allocation state of the art in the case of independent tasks, followed by the case of interdependent tasks.

2.2 A taxonomy of task allocation methods

The multi-agent task allocation problem was first treated from a robotics perspective. Several works have therefore been realized on multi-robot task allocation. These works were generally based on a taxonomy of task allocation problems, which has evolved with the field.

Gerkey's proposal taxonomy [Gerkey et al. 2004] was widely adopted in the field, and then extended. It provides a link between each method and the types of problems it can deal with. [Gerkey et al. 2004] also included a literature review of the task allocation methods, facilitating the identification of many problems to one of the classes therein discussed. This taxonomy proposes to divide the methods of task allocation according to three axes:

- SR or MR : Single-Robot or Multi-Robot tasks,
- ST or MT : Single-Task or Multi-Task robots,
- IA or TA : Instantaneous or Time-extended Allocation process.

We use the term "agent" instead of "robot" (present in the taxonomy) to make it more general and adequate to the terms used in this dissertation.

All the task allocation methods can be found in one of the sub-cubes of Gerkey's cube in figure 2.1. The cube allows to analyze, classify, evaluate and compare the solutions proposed with their equivalent instances. This formalisation has been extended by [Korsah et al. 2013; Miloradović et al. 2019].

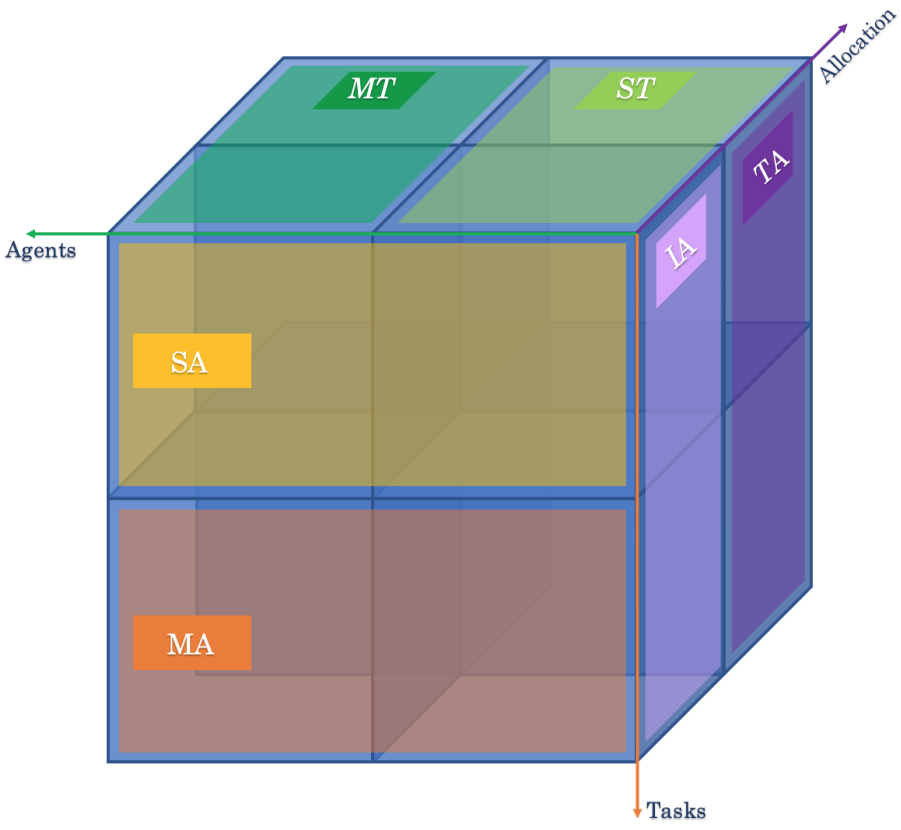


Figure 2.1: Gerkey's cube for classifying task allocation state of the art methods [Gerkey et al. 2004]

As a generalisation, we use the terms Single-Agent and Multi-Agent instead of Single-Robot and Multi-Robot for task categorisation. We also use the term agents instead of robots as a generalisation for the agent categorisation.

It is to be mentioned here that the problem addressed in this thesis lies in the category of Multi-Agent (MA) tasks, Single-Task (ST) agents, and either Instantaneous Allocation (IA) or Time extended Allocation (TA) categories depending on the problem's specificities.

2.3 Independent Task Allocation

We present in this section different method families for decentralised task allocation for cooperative agents when tasks are multi-agent and independent.

2.3.1 Market-Based Algorithms

Similar to economics auctions, market-based approaches propose that the cooperative agents perform "auctioning" on the tasks rather than on objects. Each agent bids on different tasks using the tasks utilities and then negotiate with the other agents until they agree on the best bids on the whole set of tasks.

This model implies an iteration of three stages. A task is first published, and all agents are aware of it. Then, the agents bid with calculated utilities representing offers. Finally, a winner is identified, and the task is assigned to it, and so on [Mosteo et al. 2010].

Formalisation

In auction-based methods, the task allocation problem is modeled with a tuple $< A, T, \mathbb{U} >$ where:

- A is the set of agents,
- T is the set of tasks,
- $\mathbb{U} = (\mathbb{u}_{ij})_{i \leq |A|, j \leq |T|}$ is the vector of utilities where $\mathbb{u}_{ij}$ is the utility for an agent i to do a task j.

Distributed auction allocation methods aim to maximize the sum of the utilities of each task-agent allocation:

$$\max \sum_{i \leq |A|, j \leq |T|} \alpha_{ij} \, \mathbb{u}_{ij} \tag{2.1}$$

where $\alpha_{ij} = 1$ if the task j is allocated to the agent i, 0 otherwise.

Several algorithms have been proposed for the resolution of the model based on collaborative auctions. We present some of them in the following sections, particularly the contract nets and the family of consensus auctions algorithms.

Solving with Contract Net

Contract net is one of the first algorithms introduced in this category; its application initially concerned computers and distributed sensors [Smith 1980]. This protocol is essentially based on the formalisation of interactions in multi-agent systems. It is presented as a task allocation mechanism built on the tasks subcontracting principle, using the protocol for drawing up contracts in public contracts where a relationship is established between managers, who propose the tasks to be resolved, and bidders, who bid on the proposed tasks and can become contractors as well.

The protocol consists of four stages [Wooldridge 2009]:

1. Call-for-proposals : the manager sends a description of the task to the agents.
2. Sending of proposals: by the contractors to the manager based on the description received.
3. Accepting a proposal: by the manager to the best bidder after having received and evaluated the proposals of all the contactors.
4. Establishment of the contract: between the manager and the winning bidder who confirms his commitment. Otherwise, we go back to step 3.

The agents do not have fixed roles as managers and contractors. The roles can be interchanged if needed.

Figure 2.2 presents the sequence diagram of the contractual network protocol. It shows detailed operations and interactions between the cooperative agents, namely the contract net manager and the different contractors.

Solving with consensus auction algorithms

The family of auction and consensus algorithms [Choi et al. 2009] is relatively new and has received considerable attention. The CBAA version is limited to the case of ST agents. Its generalisation to a Multi-Task (MT) multi-allocation problem is formulated in Consensus Based Bandle Algorithm (CBBA).

The CBAA algorithm is based on the iteration of two mechanisms :

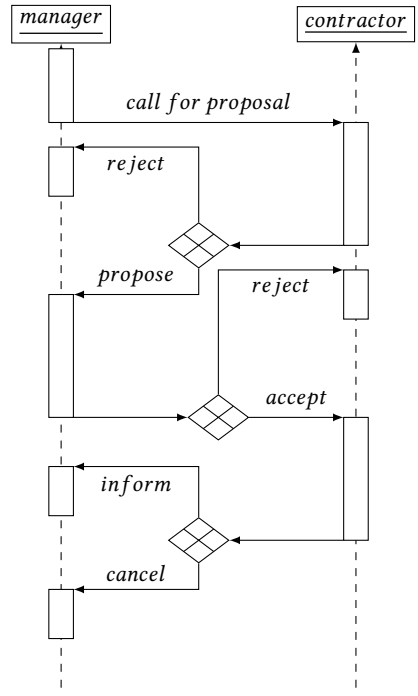


Figure 2.2: Contractual Network Protocol Sequence Diagram

Tasks selection : The auction strategy is internally done for each agent. This makes it possible for the agent to assign to itself the winning task, the one that brings it the highest reward (*i.e.*, utility).

Conflicts resolution : This consists of an agreement following local communication on winning offers with consensus mechanisms that allow for adaptability in different communication topologies.

As illustrated in figure 2.3, the iteration is ensured by the agent's awareness that it is allocated to a task. If it does not have one, it relaunches the auction to assign itself a new task. Otherwise, it makes a consensus again to unify the winning offers list with its neighbors and eventually yields the task to an agent who makes the most of it, if there is one. As a result, the agents agree on offers instead of agreeing on world perceptions to avoid inconsistencies. That significantly reduces the amount of data to be exchanged between neighbors

and removes the requirement of environmental stability. These algorithms have completely decentralised functionality due to the dual role of the agents: the auctioneer who organizes the auction and the bidder who participates in the auction.

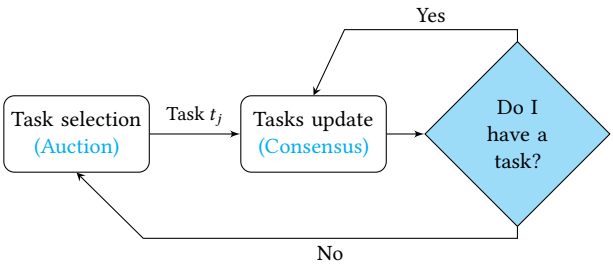


Figure 2.3: Consensus Based Auction Algorithm (CBAA) main phases

Although they are based on the same principles, the difference between CBAA and CBBA is the number of tasks to assign to each agent each time. While in CBBA, the agents seek to allocate several tasks simultaneously, when the agents are multitasking (according to Gerkey in section 2.2), CBAA responds to a problem of assignment of a task by a single-task agent.

CBBA takes place in two iterative stages. Being an MT multi-allocation algorithm, its first step consists of constructing the bundle, where each agent assigns to itself a list of tasks to realize. Then, to resolve potential conflicts, the next step is to make a consensus with other agents. The goal is to converge to a list of winning bids to decide whether to update its assignment to a task, reset it or leave it as it is.

Nonetheless, these algorithms present some limitations regarding the task allocation for heterogeneous cooperative agents problems and its relevance to real-world applications. CBAA and CBBA algorithms do not deal with the real-time aspect of applications since they assume that the task list is predefined. They also converge only in the case of constant offers or even increasing offers [Brunet 2008]. This rigidity prevents those algorithms from readjusting to the evolution of the system and the dynamism of its elements. To solve a part of this issue, [Buckman et al. 2019] suggested an extension of the CBBA algorithm, making it possible to address the case where new tasks can continuously appear. The idea is to re-allocate the new tasks during the execution of the mission without having to

relaunch the allocation each time a new task arrives. Also, [ElGibreen et al. 2019] extended the CBAA algorithm and addressed the realistic case of heterogeneous agents by considering the environment's dynamism and the uncertainty. The algorithm introduces a semantics system expressing the agents different physical capabilities used in a self-ranking negotiation matrix. This matrix is used by the agents to evaluate themselves based on their capabilities, workload, and an incremental task cost function that can be updated progressively.

Discussion

First, the problem formalisation of the market-based methods is based on the use of additive utilities. This is because the core of these methods is to make proposals on the tasks, each task individually, before agreeing on which agent is the best at doing it. Thus, the possibility of evaluating a task locally is primordial and intrinsic to the elementary concepts of market-based solutions. For that reason, market-based methods cannot solve the interdependent task allocation problem.

A problem that can arise with the Contract Net algorithm is when allocating a task to a less qualified contractor while a more qualified contractor is busy when the tasks are announced. In addition, it is not very effective when the quality of the communication is not good [Wooldridge 2009], which is the case for many realistic application cases, notably the case of drones. The reason is principally the long messages exchanged for establishing the contract. This negatively affects the allocation adaptability with possible dynamic changes in the environment.

The CBAA and CBBA algorithms and extensions have proven advantages regarding convergence to allocation without conflicts, robustness to inconsistencies of perception, and dynamism of the communication topology, as long as the communication is maintained through a connected graph including all the agents.

However, several problems of realistic use cases have not been treated by this family of algorithms. First, the interdependence between tasks in its general aspect has not been addressed and may not be handled in the classical form of these algorithms in particular and the market-based algorithms in general. However, some works have focused on presenting solutions for the specific

case of temporal interdependencies. [Luo et al. 2013] presented a market-based method, different from the ones we have presented, and conveyed the temporal aspect of the tasks, identifying each task by a specific deadline. In this work, a temporal constraint for a task, particularly its deadline, is local to that task and does not concern the other tasks, making the tasks independent.

Furthermore, even with the guarantee of a minimum level of performance of CBAA algorithm and its different extensions, these algorithms are bounded within 50% of the optimal solutions (meaning that it is proven that the final utility is never less than half the optimal utility) under the assumption of precise knowledge of the environment. This way, they give the same results as the centralised greedy algorithms. The greedy algorithms are based on the idea of choosing the best available agent for each task. They make locally optimal choices at each stage by following the problem-solving heuristics.

The original CBAA and CBBA algorithms do not solve all the cases of interdependence between the tasks and different aspects of many real-world applications. They also suppose in practice that agents are "black boxes" with quasi-perfect capacities ignoring their performance in carrying out the tasks once assigned. Yet, they are sufficiently extensible to be adapted to the different problem cases and thus increase their performance and applicability [Dias et al. 2006].

2.3.2 Distributed Constraint Optimisation Problems

Constituting a good part of research in Artificial Intelligence (AI) and Operations Research (OR), Constraint Satisfaction Problems (CSPs) are mathematical problems whose solving methods seek to find for a set of variables a combination of values that satisfies a set of constraints [Dechter, Cohen, et al. 2003].

The Constraint Optimisation Problem (COP) is an optimisation framework that generalises CSP by replacing the boolean constraints satisfaction with degrees of satisfaction over constraints. The objective of this framework is to optimise the constraints rather than to satisfy them firmly. However, hard constraints can also be modeled with a boolean satisfaction degrees system. Otherwise, the other optimised constraints are considered as preferences specifying the extent of satisfaction of the associated constraint. They must be maximized if they represent gain, utility, or minimized if they represent cost, loss.

When the precedent optimisation problem has to be performed in a decentralised manner by different actors or agents, it is called Distributed Constraint Optimisation Problem (DCOP). Hence, the DCOP framework is the distributed

version of the COP framework. It can be considered as multi-agent systems paradigm, where the agents communicate so that each agent can decide the affectation value of its variable, aiming to optimise a global objective function. A complete and exhaustive presentation of the framework can be found in [Fioretto et al. 2018].

As it is an optimisation paradigm, DCOP has been employed in several MAS applications like scheduling, recommendation systems, radio frequency allocation, service-oriented computing, traffic control and coordination [Fioretto et al. 2018]. Task allocation for cooperative agents is one of those applications thanks to the possibility to formulate a multi-agent task allocation problem under the DCOP paradigm (see the next subsection). Indeed, many efforts have been deployed to represent the multi-agent cooperation and the tasks allocation problem as a DCOP. In particular, RoboCup challenges have demonstrated how this type of techniques can be used for a task allocation problem [Farinelli, Rogers, and Nick R Jennings 2014; Parker et al. 2018; Pujol-Gonzalez, Jesus Cerquides, Farinelli, Meseguer, and Rodríguez-Aguilar 2014; Pujol-Gonzalez, Jesus Cerquides, Meseguer, et al. 2018; Ramchurn, Farinelli, et al. 2010].

Formalisation

We assume here, that by optimisation, we mean the maximisation of a utility function. The formalisation is the same if we had cost functions to minimize.

A DCOP, for the case of task allocation, is a tuple $< A, X, D, F, \alpha >$ where:

- $A = \{a_1, \ldots, a_m\}$ is a set of agents,
- $X = \{x_1, \ldots, x_n\}$ is a set of variables, representing the decision of what task to undertake,
- $D = \{D_1, \ldots, D_n\}$ is the set of finite domains of the variables in X, with D_i being the domain of variable x_i. In the task allocation context, each D_i represents the set of tasks that agent a_i can do,
- $F = \{f_1, \ldots, f_p\}$ is a set of utility (or cost) functions, with $f_j : \prod_{x_i \in \mathrm{x}^j} D_i \rightarrow \mathbb{R}^+ \cup \{\bot\}$ where each f_j represents the utility of a task t_j, x^j is the scope of f_i (the decisions of agents that can do task t_j) and the $\bot$ symbol means

that a combination of the values of the variables x^j is not allowed. The utility functions can also be called constraints since they represent the constraints to optimise,

- $\alpha : X \rightarrow A$ is a surjective function that assigns the control of each variable x_i to an agent $\alpha(x_i)$. In the task allocation context, we assume that each agent controls one variable that represents its decision, with: $\forall x_i \in X \; \exists a_k \in A : \alpha(x_i) = a_k$ and $i = k$. Thus $|A| = |X|$.

▶ **Definition 2.1 (Complete Assignment).**
A *complete assignment* ω is a value assignment for all the variables in X. ◀

▶ **Definition 2.2 (Solution).**
A *solution* to a DCOP is a complete assignment that satisfies (*i.e.*, does not violate) all its constraints, *i.e.*, utility functions. A constraint or a utility function is satisfied by an assignment ω when $f_j(\omega_{\mathrm{x}^j}) \neq \bot$. ◀

▶ **Definition 2.3 (Optimal solution).**
An *optimal solution* to a DCOP is a complete allocation of values to all the problem variables, that maximizes the sum of the utility functions. ◀

The goal in DCOP is to find an optimal solution (*i.e.*, an allocation of tasks to agents represented in the values of the decisions' variables); a solution that maximize the total problem utility functions:

$$\omega^* = \underset{\omega \in \Omega}{\operatorname{argmax}} \sum_{f_j \in F} (\omega_{\mathrm{x}^j}) \tag{2.2}$$

where Ω is the set of all possible solutions and ω_{x^j} is a partial assignment to the variables relevant to utility function f_j in Ω.

We mention here that we can have, in a given problem, a mix of hard (to satisfy firmly) and soft (to optimise) constraints. Hard constraints are represented in this case by a utility function that can have only two values $f_j : \prod_{x_i \in \mathrm{x}^j} D_i \rightarrow \{1\} \cup \{\bot\}$ where 1 means satisfied and $\bot$ unsatisfied. Soft constraints are represented normally $f_j : \prod_{x_i \in \mathrm{x}^j} D_i \rightarrow \mathbb{R}^+ \cup \{\bot\}$ with different satisfaction levels.

▶ **Example 2.4.**
Here is an example of the application of the DCOP paradigm for a task allocation problem.

Consider we have a set of agents $A = \{a_1, a_2, a_3\}$ that should accomplish a set of tasks $T = \{t_1, t_2\}$.

The formalisation will be as follows:

- $A = \{a_1, a_2, a_3\}$ the agents.
- $X = \{x_1, x_2, x_3\}$ the variables representing the agents decisions.
- $D = \{D_1, D_2, D_3\}$ the variables domains. We consider that each agent can do any task in T. Then: $T = D_1 = D_2 = D_3$.
- $F = \{f_1, f_2\}$ the set of utility functions where each f_j is the utility function of task t_j and each function f_j has as scope the set of all agents decisions, formally: $\mathrm{x}^j = X$.
- α the control assignment function where $\alpha(x_i) = a_i$ for $i \in \{1, 2\}$.

In this case, the optimal solution is the complete assignment $x_1 = t_1$, $x_2 = t_2$ and $x_2 = t_1$. ◀

Several algorithms have been developed to solve DCOPs. By highlighting quality guarantees, and based on completeness, we can classify them as complete algorithms, approximate algorithms with error bounds and approximate algorithms without error bounds. Below is a description of those categories with some examples of algorithms. The exploration processes in DCOP algorithms are different and can be presented in three categories [Yeoh 2010]. The first category is based on research techniques to explore the space of possible solutions. The second category is derived from the fields of dynamic programming and the propagation of beliefs, and is thereby based on inference , allowing agents to exploit the structure of the problem graph to aggregate information and gradually reduce the size of the problem. The last category is based on sampling the research space to approximate a probability distribution as a product of statistical inference.

x_1	x_2	x_3	$f_1(\mathrm{x}^1)$	$f_2(\mathrm{x}^2)$
t_1	t_1	t_1	8	0
t_1	t_1	t_2	6	2
t_1	t_2	t_1	7	4
t_1	t_2	t_2	3	6
t_2	t_1	t_1	7	3
t_2	t_1	t_2	4	5
t_2	t_2	t_1	4	6
t_2	t_2	t_2	0	7

Table 2.1: Example of the utility function for a task allocation problem modeled with DCOP

Solving with complete algorithms

The complete DCOP algorithms are those that present mathematical proof of obtaining the optimal solution. Thanks to an exhaustive search of the problem space, they can guarantee the optimality of their results. However, these algorithms are NP-complete, take a long time, and consume considerable computing power to produce a solution when the given a problem with a considerable size.

Several works have developed algorithms for this category. We name two examples, Synchronized Branch and Bound (SyncBB) [Hirayama et al. 1997] and Asynchronous Distributed OPTimisation (ADOPT), two complete algorithms based on the space search for possible solutions. SyncBB, as its name suggests, is a synchronous distributed version of the classic Branch-and-Bound algorithm, based on a complete heuristic order between variables. ADOPT, on the other hand, is asynchronous and based on the concept of maintaining and tightening the lower and upper limits of the utilities of each agent until the two limits are equal.

Solving with approximate algorithms with error bound

For real-world applications, particularly those on a large scale, requiring real-time results or involving robotics, problem-solving is done in distributed environ-

ments with limited computing resources. Finding optimal solutions to DCOP problems is NP-hard. Thus, solving the problem with a complete algorithm is sometimes not applicable for some real-world applications. It is, therefore, necessary to consider faster incomplete algorithms even when losing the guarantee of optimality. In fact, unlike complete algorithms, incomplete algorithms generally do not offer any guarantee on the quality of the calculated solutions. However, some approximate algorithms have an error bound, which allows them to guarantee a specific performance in their results. The approximate algorithms without any guarantee of quality are presented in the next subsection.

Distributed Upper Confidence Tree (DUCT) [Ottens et al. 2017] and Distributed Gibbs (D-Gibbs) [Nguyen et al. 2013] are part of these approximate algorithms with error bounds as a guarantee of the quality of the solutions. The two algorithms are incomplete, synchronous, and based on sampling. DUCT is inspired by the Monte Carlo Trees Search and uses confidence limits to solve DCOPs. D-Gibbs, meanwhile, extends the Gibbs sampling process by adapting it for DCOPs in a decentralised manner.

Solving with approximate algorithms without error bound

There are approximate algorithms that do not provide any theoretical error bounds but can have very good experimental performances. These algorithms try to limit computation time, complexity and memory usage, making them practical in contexts where decisions must be taken in real-time.

Most incomplete algorithms fall into this category [Okimoto et al. 2011]. Among others, two popular synchronous algorithms were presented: Distributed Stochastic Algorithm (DSA) [Fitzpatrick et al. 2003; Maheswaran et al. 2004] and Max-Sum [Farinelli, Rogers, Petcu, et al. 2008]. DSA is based on a stochastic variant of another incomplete algorithm, Maximum Gain Message (MGM) [Maheswaran et al. 2004]. It is built on stochastic decision-making to escape local minima. Max-Sum, on the other hand, is based on inference and beliefs propagation, and works on factor graphs, where each node represents an agent's decision or the utility function related to a task. The main concept is based on considering the impact of value assignment in the marginalized utility function. This algorithm has the advantage of being fast with a limited communication load in memory. It is guaranteed that Max-Sum converges to an optimal solution on acyclic graphs, but convergence is not guaranteed on cyclic ones. We mention here that this algorithm has also been used in modern reinforcement

learning research for cooperative MASs to propagate payoffs and determine an approximately maximizing joint action [Kok et al. 2006].

Discussion

DCOP provides an interesting framework for the task allocation problem, with a large panel of algorithms. Being based on an explicit and precise formalisation, it allows mathematical modeling of any problem, as well as a detailed study of the behavior of its methods.

The above proposed methods are presented in table 2.2 according to the following characteristics that are proposed by the survey [Fioretto et al. 2018] and that allow choosing the most adapted algorithm for a specific application by matching them with the addressed problem's requirements:

- **Completeness**, indicates if the algorithm can prove the possibility or not of convergence to a globally optimal solution, and if not, if it converges to an approximate solution with a certain error bound;
- **Complexity** per agent is the asymptotic algorithmic complexity carried by each agent (since the DCOP algorithms is decentralised);
- **Anytime** indicates if the algorithm can produce a valid solution at any time if it is interrupted (*i.e.*, even before it finishes);
- **Number of messages** to exchange before convergence;
- **Size of messages** exchanged in the algorithm;
- **Local communication**, indicates if the problem can be solved by interacting only with neighboring nodes (graphs being the principle representation of DCOPs).

Completeness	**Algorithm**	Complexity	Anytime	# msgs	Msgs size	Local com.
Complete	**SyncBB**	$O(d^n)$	✓	$O(d^n)$	$O(n)$	✗
Complete	**ADOPT**	$O(d^n)$	✗	$O(d^n)$	$O(n)$	✓
Bounded error	**DUCT**	$O(\ell l d)$	✓	$O(\ell n)$	$O(n)$	✓
Bounded error	**D-Gibbs**	$O(\ell l d)$	✓	$O(\ell n l)$	$O(1)$	✓
Unbounded error	**DSA**	$O(\ell l d)$	✓	$O(\ell n l)$	$O(1)$	✓
Unbounded error	**Max-Sum**	$O(\ell d^l)$	✓	$O(\ell n l)$	$O(d)$	✓
Unbounded error	**FMS**	$O(\ell d^l)$	✓	$O(\ell n l)$	$O(d)$	✓

Table 2.2: Summary table of some DCOP algorithms characteristics (Optimality, Runtime and Communication), taken from [Fioretto et al. 2018]

We summarize in table 2.2 a set of DCOP algorithms with their different characteristics using the following notations:

- $n = |A|$ the number of variables. It is equal to the number of agents, since we consider that each agent has exactly one variable.
- $d = \max_{D_i \in D} |D_i|$ the largest domain size (number of elements).
- $l = \max_{a_i \in A} |N_{a_i}|$ the largest number of neighbouring agents, where N_{a_i} is the list of agent a_i's neighbors.
- ℓ the number of iterations (for incomplete algorithms).

This table presents for each algorithm its completeness (*i.e.* its ability to prove optimality or unsatisfiability), and if it is incomplete the existence or not of an error bound. It also presents whether the algorithm is anytime or not (*i.e.* it offers a possible solution at any moment), the number and the size of the exchanged messages and finally the fact that agents only need to communicate with their direct neighbors or not.

The DCOP framework takes advantage of its interactions with different paradigms, notably decision theory, constraint programming, and game theory, to extend to new models ranging from dynamic DCOPs (for dynamic environments with changing utility functions) [Ramchurn, Farinelli, et al. 2010; Yeoh

et al. 2015] to probabilistic DCOPs (for environments with stochastic behaviors) [Nguyen et al. 2012; Wu et al. 2014]. Thus, this model offers a panoply of methods with different properties adaptable to various application cases, executing in environments with various behaviors (deterministic or stochastic), that evolve differently (static or dynamic). The different DCOP methods also apply to agents with very varied aspects, at the level of their behavior (deterministic or stochastic), their knowledge (total or partial), or their positioning with regard to cooperation between agents (cooperative or competitive).

The paradigm applies to different multi-agent applications, including task allocation applications. Given that DCOPs are NP-hard, approximate algorithms as in Max-Sum and its extension Fast-Max-Sum have been used for task allocation for different applications such as RoboCup challenges [Pujol-Gonzalez, Jesus Cerquides, Farinelli, Meseguer, and Rodriguez-Aguilar 2015; Ramchurn, Farinelli, et al. 2010], sensor networks [Farinelli, Rogers, and Nick R Jennings 2014; Vinyals et al. 2011] and mobile sensing robots [Yedidsion et al. 2018]. The Fast Max-Sum algorithm, proposed in [Ramchurn, Farinelli, et al. 2010], optimises the algorithm Max-Sum exploiting the binary relation in an agent-task couple (allocated or not allocated). This algorithm also reduces the number and size of messages and the computation time in comparison with Max-Sum. It is shown to be robust to dynamism regarding the numbers of the problem's tasks. However, being based on additive utilities as in equation (2.1), DCOP methods only define local utilities regarding the tasks and therefore do not address the task allocation problem when the tasks are interdependent.

2.4 Interdependent Task Allocation

As seen earlier in this chapter, an important part of the task allocation literature focuses on the independent tasks case. Nevertheless, real-world applications go beyond the inter-tasks independence. In fact, it is common that tasks exhibit different types of interdependencies with different forms of influence between them.

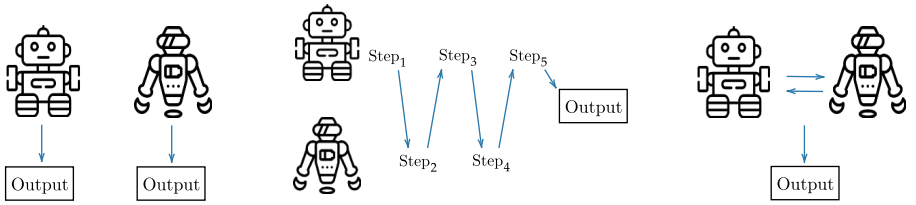
2.4.1 Interdependent Tasks in Organization Design

When we talk here about interdependent tasks or task utilities, it is essential to mention that they differ from interdependent valuations, mainly used for

competitive agents. For example, interdependent valuations are described in [Ramchurn, Mezzetti, et al. 2009] where they are introduced in competitive agent scenarios with the purpose of separate utilities among several agents: for each task, an agent gives a valuation that is affected by the other agents' valuations for this task, based on a trust model. By contrast, we focus in our work on completely cooperative agents where there is no notion of dividing utility. Also, in a competitive configuration, the underlying concept is to consider other agents' perspectives to calculate its individual contribution and then own it. However, in a cooperative configuration, it is instead the team that is considered; we may need to calculate individual incomes in specific methods but with the objective of gathering and not dividing.

Now, let us define the task interdependence we are considering. Task interdependence is a familiar concept with the organization design domain. This domain mainly examines processes, roles, workflows, structures, and systems to ensure organizations' goals effectively. In the same field, [Puranam et al. 2012] defined task interdependence as follows: "two tasks are interdependent if the value generated from performing each one is different when the other task is performed versus when it is not". Typically, the studied interdependence, in this work, is called agent-agnostic. This is when tasks are interdependent regardless of who accomplishes them. Different levels of task interdependence are described to characterize the influence between the outputs of different organizational units [Thompson et al. 2017]. In [Coyote et al. 1967], three levels of interdependence are distinguished, capturing increasing complexity along a Guttman scale (a single ordinal scale designed to arrange items with respect to an attribute [Guttman 1944]): tasks can be characterized by pooled, sequential, or reciprocal interactions. Firstly, pooled interdependence is when tasks are performed separately with minimal interaction between tasks performers without affecting each other's performance. Secondly, sequential interdependence is when there is a predefined order over the execution of the tasks, and thus a task's execution state affects the performance of the following tasks. Finally, reciprocal interdependence concerns high levels of interdependence with specific agents' skills or characteristics. Temporal interactions might occur between tasks but not necessarily; there may be cases where tasks can be executed following flexible orders. In figure 2.4, we are illustrating this categorisation through a multi-agent system. Due to the joint activity that humans may have with robots in human-robots systems, the human-robot teams' research community studies task interdepen-

dence in human-robot teams on different levels of interdependence [Johnson et al. 2014; Lematta et al. 2019; Zhao et al. 2020].



(a) Pooled Interdependence (b) Sequential Interdependence (c) Reciprocal Interdependence

Figure 2.4: Interdependence levels on a multi-agent system inspired by [Zhao et al. 2020]

In the multi-agent task allocation context, existing methods often assume that tasks are independent and rarely consider tasks and groups dynamics. This may be because autonomous decomposition of tasks into smaller sub-tasks has not yet been extensively studied [Brutschy et al. 2014]. On the other hand, due to the apparent presence of inter-tasks sequentiality in different applications and the problem in task planning and coordination solutions, tasks with temporal interdependence, such as sequentiality, present a big part of the interdependent task allocation literature as discussed in section 2.4.3.

2.4.2 Interdependent Tasks in Coalition Formation

In this thesis, the case we are dealing falls in the single-task robots and multi-robot tasks (ST-MR) category, which is among the classes of task allocation problems defined by [Gerkey et al. 2004]. Since we intend to address the inter-tasks dependencies in their generality and not only the temporal ones, the last category (instantaneous or time-extended assignment) is not specified. As introduced in section 1.3.5, allocation of tasks to subgroups of agents has been dealt with by several approaches, including coalition formation methods [O. Shehory et al. 1998].

The models for coalition structure formation, falling into the game theory framework, are called coalition function games [Hajduková 2006]. In coalition formation, two types of games exist following the influencing factors for

a coalition's value: Characteristic Function Games (CFGs) and Partition Function Games (PFGs). In CFGs, the value of each coalition only depends on its composition, *i.e.*, its members. By contrast, the PFGs address the problem of interdependent coalitions where the value of each coalition depends not only on its members' identities but also on the other coalitions compositions [Myerson 1977].

Multiple methods have been used to solve CFGs. One approach is to rely on COP and Constraint Satisfaction Problems [Dechter 2003] and their solutions, to find suitable ways to form coalitions, while enforcing constraints on the coalition structure. For instance, [Ramchurn, Polukarov, et al. 2010] dealt with the problem of task allocation with spatial and temporal constraints. Their method allocates agents to tasks so that coalitions may be feasible with respect to the locations, tasks workloads, deadlines, and the number of completed tasks is maximized. However, the method is centralised and does not generalise to other constraints. CSPs have also been used in other contexts relevant to task allocation.

In a CFG, a specific coalition has one and only one value. However, in a PFG, the same coalition has as many possible values as the number of possible partitions of the agents outside it. Hence, we can observe that CFGs are a subclass of PFGs [Rahwan, T. P. Michalak, et al. 2015]. Consequently, it is much easier to work with CFGs, which can explain their significant presence in the literature. By contrast, since PFGs coalition values also depend on the way non-members are partitioned, computing coalition structures in PFGs is very challenging on a computational level.

Given an arbitrary partition function, an exhaustive search is required to provide an optimal coalition structure [Präntare et al. 2020] unless externalities-based models are studied for the partition function to reduce complexity and allow a solution in a reasonable time. An externality is a natural aspect of PFGs describing the difference in a coalition value, in a specific coalition structure, that results from the merge of two other coalitions in that same coalition structure [Rahwan, T. P. Michalak, et al. 2015]. This is because the utility of a specific coalition is affected by external moves related to the formation of other coalitions. There are specific sub-classes of PFGs based on this. We have, for example, games with negative externalities noted PFG^{-} representing games where the merge of any two coalitions is beneficial to the other coalitions (as when agents have overlapping or partially overlapping objectives). We also have games with

positive externalities noted PFG$^+$ representing situations detrimental to other coalitions (when self-interested agents have to use shared bounded resources) [T. Sandholm et al. 1999]. In addition, [Rahwan, T. Michalak, et al. 2012] defined a specific kind of externalities representing inter-coalition effects (for example, assumptions on utility functions and coalition mergers). Since in our problem, agents are cooperative, meaning they have completely overlapping goals, we are more interested in the PFG$^+$. In such games configuration, agents that are not members of a specific coalition may change the world to the closest state of the coalition's goal, making it more affordable for the coalition's members to achieve their goal. Nevertheless, our use of the coalition formation paradigm in a task allocation application fixes the number of coalitions to the number of tasks in the problem. For that, we cannot consider merging the coalitions and thus use the externalities-based methods.

Several approaches were proposed to solve the Partition Function Form Game problem [Kóczy 2018]. However, these methods are centralised, and the notion of agents is either absent or subject to a centralised allocation. The agents are not completely autonomous and do not make their own decisions: their orders are provided by the centralised planning agent. Such a centralised approach is inapplicable in our case.

Because of their centralised property, these methods are unable to deal with cases where the agents cannot communicate with the central point, either because it is too far or because the communication with it is unreliable. This central point can be one of the agents or a specific external calculator.

Finally, in interdependence cases other than precedence dependencies, [O. Shehory et al. 1998] proposed to combine interdependent tasks into unified tasks and to solve dependencies with CSP techniques. Nevertheless, the proposed algorithms cover only the precedence order and the resource consumption task dependencies cases.

2.4.3 Temporally Interdependent Tasks

In the task allocation literature, many studies address the interdependent task allocation problem. Nevertheless, these works mainly focus on the specific case of temporal interdependencies. Such interdependencies are expressed through the consideration of constraints of precedence and required concurrency between tasks. Those are often mutually studied for the planning problem [Coles

et al. 2009]. Basically, tasks without temporal constraints can all be executed simultaneously, in parallel, if enough agents are available with sufficient resources. However, even with this abundance of agents and resources, we might have an application where temporal constraints are inherent. Let's consider an example where agents have munitions that need to be recharged and that are needed in specific tasks. An agent has to start by recharging its munition, it then moves to the task's position where it finally uses the munition; there is a natural precedence order between these two tasks. Hence, precedence order constraints are when agents assigned to a task should wait for the accomplishment of another task to start executing theirs, following a specific precedence order. On the other hand, required concurrency constraints between two tasks, for example, necessitates concurrent execution of these tasks. The two tasks are thus conditioned to be executed in parallel. An example is of two tasks of mending fuses in the dark and lighting a match. The first task cannot be executed without the second and the second, has no utility in the absence of the first [Coles et al. 2009].

[Behrens et al. 2019] examined tasks with spatio-temporal requirements and task ordering constraints, on a use case application treating industrial dual-arm and multi-arm robots working on manipulation and assembly tasks. The issue addressed by [Behrens et al. 2019] covers not only the allocation of task steps and actions to the individual arms but also optimal planning. The problem is named Simultaneous Task Allocation and Motion Scheduling (STAAMS). The goal is to compute executable optimal motion plans while considering different allocations of partially-ordered tasks to the individual robot arms. This work proposes a descriptive model to define the tasks and their temporal requirements. In addition, it suggests a centralised constraint programming approach to obtain the arm allocation and the plan of tasks. For this, the solver must integrate the task and robot motion models into the constraint optimisation problems to be solved and resolve them with heuristics for higher efficiency.

Another work that examines the temporal and ordering constraints in the problem of multi-robot task allocation is [Gini 2017]. It suggests to name this class of problems Multi-Robot Task Allocation with Temporal and Ordering Constraints (MRTA/TOC). For this problem, it offers a literature review that covers different aspects of time-extended assignments using several temporal models, optimization objectives, and typical solution approaches. These solution

approaches include decentralized methods mainly based on auction models or DCOPs.

[Beck et al. 2016] deals with the same precedent problem, joining the collaboration and task allocation aspect to task planning due to tasks' temporal and order conditions. However, it studies a different configuration of problems and is applied to Collaborative Search and Rescue, a classic application in the literature. Search and Rescue missions consist of heterogeneous robots performing actions for the rescue part and piling up knowledge about future tasks for the search part. For these reasons, the work proposes an online approach, thus dynamic to adapt to newfound tasks; and that considers incomplete information, consisting of having a set of probable uncertain tasks in addition to the set of known and certain tasks. According to [Gerkey et al. 2004]'s taxonomy, the problem examined here falls into the category ST-SR-TA for Single-Task robots, Single-Robot tasks, and Time-extended Assignment. Due to the spatial character of the tasks, coordinating robots' tasks is mandatory to increase performance. The coordination need is answered by holding dependency between tasks and hence forming a preferred space-based ordering. The paper suggests an algorithm to solve what it denominates as Uncertain Multi-Robot Task Allocation (UMRTA) and an online planning approach resulting in a joint plan to coordinate the collaborative robots.

Similar to the search and rescue application, a use case describing the temporal interdependent tasks of harvesting and storing objects is studied in [W. Lee et al. 2020]. Tasks are simultaneously performed at different locations, and the posterior tasks should be processed after completing the prior tasks to complete the overall task. The method, applied to a swarm robotic system, suggests that robots continuously calculate their response threshold of tasks that varies depending on the task demand and the number of neighboring robots performing the task. The response threshold-based model is continuously updated in each robot, making the whole self-organized swarm converge to the objective task distribution.

In [Dahl et al. 2009], interdependencies between tasks are referred to as the group dynamics. As the focus is on spatially explicit environments and due to different group densities and eventual clutters and collisions, the studied group dynamics perceive interactions from a temporal point of view. The problem

considered falls into the ST-SR-TA's category in the taxonomy of [Gerkey et al. 2004], and the multi-robot task allocation problem is regarded through the scheduling prism where jobs have to be assigned to machines. A Vacancy Chain Scheduling (VCS) model, a bio-inspired resource allocation process loaning the concept from the example of attributing vacancies to employees after a senior's retirement following a promotions Vacancy Chain process answers this problem. The model focuses on homogeneous agents but can be generalised to heterogeneous ones and calculates robots' contributions to different tasks. Breaking global performance into these individual contributions simplifies the general scheduling problem. A VCS algorithm based on this model and using Reinforcement Learning techniques is proposed. Concretely, Q-learning is adopted to assess the local task utility that grasps the group dynamics effects subject to other robot allocations. The reward function employed in the learning process explicitly exposes the temporal character of the performance. This algorithm relies on the emanation of optimal allocation patterns from robot interactions stigmergy.

In the context of swarm robotics, [Brutschy et al. 2014] suggests a near-optimal self-organized method for sequential task allocation. To answer robustness, scalability and dynamism challenges, difficult to tackle with optimal methods in the literature, the global allocation in this method is the result of individual local decisions. Any task change is mainly based on the time the robots wait in a task, denoted the interface delay, relative to an other amount of time waited in another task, relying on local interactions and each robot perception.

Finally, [Capezzuto et al. 2021] deals with the problem of a multi-agent task allocation problem where tasks have deadlines and workloads and agents need to cooperate in coalitions to be efficient and complete the maximum of tasks. It names the problem as Coalition Formation with Spatial and Temporal constraints Problem (CFSTP) and extends the work of [Ramchurn, Polukarov, et al. 2010] by optimising the mathematical programming formulation of the CFSTP and designing a distributed version of the suggested algorithm with the use of the DCOP paradigm. The algorithm is tested on a realistic test framework that simulates the mobilization of firefighters. The results show that the algorithm stands out in situations where the number of agents monotonically decreases over time.

2.5 Analysis and Discussion

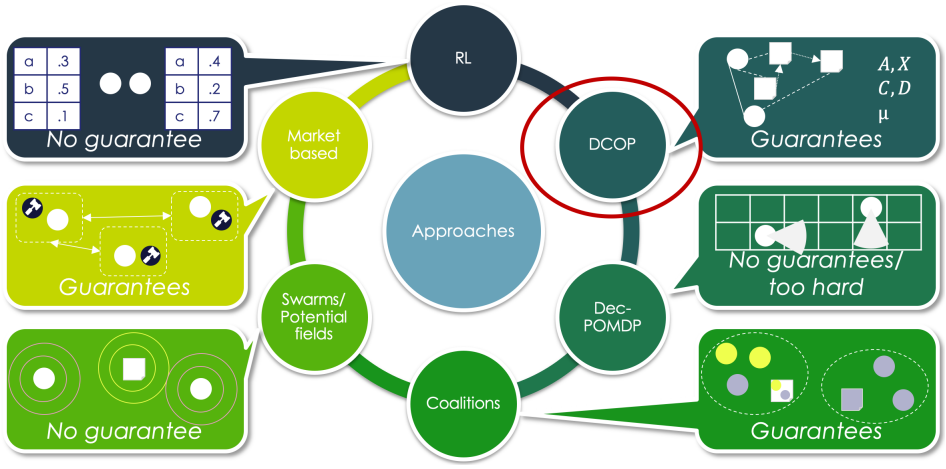


Figure 2.5: Representation of the different task allocation methods families

Figure 2.5 presents the discussed methods with an illustration of their mechanisms and guarantees.

Table 2.3 outlines an overview using the taxonomy given in section 2.2, in addition to the two elements mentioned above: computational complexity and guarantees in terms of optimality. The presented approaches have very different characteristics.

It is sometimes more important to have guarantees in terms of optimality rather than having a reduced calculation time, for example, when long computations need to be distributed on nodes. In this case, it may be useful to use an optimal algorithm, or at least one with performance guarantees, even if the allocation process is longer. Other critical applications require ensuring the treatment of all the tasks. It is then necessary to formalise the compromises to achieve the best success for the mission. Having access to a formal utility function, which represents these compromises, and having algorithms that can find the optimal solution is essential, even if it means taking a little more time to converge to an allocation. DCOP exact algorithms are the most suitable for these problems. In other cases, it may be necessary to quickly find an allocation, even

if it means finding a sub-optimal allocation. For example, this can be the case for a drone, particularly in the case where the environment is changing rapidly, and where the allocation has to be revalued. Approximate and market-based approaches are therefore more appropriate in this case.

On the other hand, the study of interdependent task allocation and the coalition formation literature leads us to two conclusions.

First, the few works addressing the task allocation problem with interdependencies focus on the specific case of temporal interdependencies, where constraints of precedence and concurrency are considered. Different techniques are used to solve the task allocation in cases of precedence dependencies between tasks as extensions of market-based methods and the constraint satisfaction problems. However, no method that surpasses the precedence dependencies to more general dependencies for the task allocation exists.

Secondly, the interdependent coalitions are expressed in the coalitions formation literature by PFG. The current state of the art presents this coalition's case as only solvable by brute-force search unless externalities-based models are analyzed. However, PFGs are based on the notion of payoffs and utility partition among coalitions which corresponds more to rational competitive contexts rather than altruist cooperative ones. This notion of payoff configurations and their allocation deriving from the coalitional utility is, thus, not relevant to the problem at hand, rendering as such PFGs and related models not beneficial in this thesis work. Those externalities are related to merging coalitions, which are not adapted to task allocation where the number of coalitions is fixed (same as the number of tasks). Besides, since the PFGs solving is highly complex without considering externalities, it has only been addressed in a centralised configuration.

Finally, another trait of externalities-based models that impedes their use in our work is their assumption that externalities are known a priori. At the same time, this is problematic in most real-world settings we are targetting.

This being said, the literature research works do not propose solutions to our problem. No method has been suggested to find a task allocation among agents under the coalition formation in a decentralized manner for a multi-agent system in the case where there is an interdependence between the tasks in the general meaning of tasks interdependence.

	SA	MA	ST	MT	IA	TA		
Contract net	•		•	•	•		P	BOUNDED
CBAA	•		•		•		P	BOUNDED
CBBA	•		•	•	•	•	P	BOUNDED
DCOP – exact methods	•	•	•		•		NP	OPT
DCOP – bounded error	•	•	•		•		NP	BOUNDED
DCOP – approximated	•	•	•		•		ANYTIME	∅

Table 2.3: Summary table of multi-agent task allocation approaches with their classification in Gerkey's taxonomy and characteristics regarding completeness and complexity

2.6 Conclusion

This chapter reviewed task allocation methods existing in the literature for both independent and interdependent tasks cases. Since this thesis problem concerns interdependent tasks, let us start with the interdependent tasks.

Among all the works we have reviewed, the proposed task allocation methods for interdependent tasks only solve the precedence interdependence. Though this is a natural and prevalent interdependence type in real-world applications, general interdependence cannot be reduced only to that case. Tasks may show dependencies other than ordering and temporality-related ones, as in resource sharing and quality influence situations. The coalition formation, in which our problem can be situated due to the multi-agents property of the tasks we consider, proposes to deal with interdependent coalitions. Nevertheless, because of the complexity of the problem, the only solutions proposed to reduce that complexity involve externalities. A central aspect of cooperative game theoretic work is sharing payoffs among the agents of emerging coalitions. However, there is no need to tackle this problem in this work since the agents are entirely cooperative and do not need to calculate their individual payoffs. Therefore, utilising models or algorithms in the literature concerning cooperative games in general (including PFGs) is not valuable for the problem at hand and thus of not much interest in this thesis.

Hence, the current works are not dealing with the general case of multi-agent task interdependencies for the task allocation problem in entirely cooperative contexts.

However, a vast majority of task allocation methods for cooperative agents have been carried out upon the assumption of task independence. This is why we have outlined, earlier, task allocation approaches in the case of independent tasks. We concluded for each family of methods by discussing the methods' properties and examining the possibility of adapting them for interdependent tasks.

3 Modelling Interdependent Task Allocation

Contents

3.1 Introduction **48**

3.2 Classical Independent Task Allocation Problem Formalisation **48**

3.2.1 Allocation of tasks requiring a single agent 49

3.2.2 Allocation of tasks requiring many agents 49

3.2.3 Discussion 50

3.3 Modelling Interdependent Task Allocation **50**

3.3.1 Agents 51

3.3.2 Tasks 51

3.3.3 Fulfilment 54

3.3.4 Coalition Structures 57

3.3.5 Global Utility Function 58

3.4 Conclusion **61**

3.1 Introduction

The previous chapter discussed the shortfall of the task allocation literature methods concerning the problem of decentralised multi-agent task allocation where tasks have inter-tasks dependencies in their general case. We studied the task allocation literature where each task valuation depends only on the task itself and the agents assigned to it, but also when an overall evaluation cannot be avoided. Those are the independent and interdependent tasks respectively. The task allocation methods discussed, albeit very popular, have some limitations that we address below. Mainly, they do not address problems where tasks have dependencies, which is a recurrent problem in real-world applications.

Concretely, this chapter starts by recalling the classical task allocation formulation. Then, it introduces a formalisation for the interdependent task allocation problem. This formulation, which is one of this thesis contributions, describes the components of the task allocation problem, essentially the agents and the tasks, considering both qualitative and quantitative information about them. Not only the agents and tasks are modeled, but also their interactions and goal. The task dependencies in this model appear at both levels of the tasks' requirements and the utility function.

3.2 Classical Independent Task Allocation Problem Formalisation

Before we introduce our modelisation of the interdependent task allocation problem, we start in this section by exposing the classical task allocation problem formalisation for independent tasks.

The task allocation problem formalisation differs depending on whether the tasks require the participation of a single agent or a group of agents. When tasks require a single agent, which we call single-agent tasks in chapter 2, the task allocation is ensured between the set of tasks and the set of agents. However, in the case of multi-agent tasks, the task allocation is done between the tasks' set and the agent's subsets.

3.2.1 Allocation of tasks requiring a single agent

For single-agent tasks, the task allocation can be formalised as follows :

- $A = \{a_1, a_2, \ldots, a_{|A|}\}$ is a set of agents.
- $T = \{t_1, t_2, \ldots, t_{|T|}\}$ is a set of tasks.
- $f : A \times T \rightarrow \mathbb{R}$ is a utility function. It is a function that associates a cardinal utility value $f(a_i, t_j)$ in $\mathbb{R}$ to a pair made of an agent $a_i \in A$ and a task $t_j \in T$.

Denoting μ an allocation of tasks to agents such that for each agent a_i and each task t_j, $\mu_{a_i,t_j} = 1$ if t_j is allocated to a_i and 0 if not. The objective is to find an allocation μ that maximises the global utility:

$$\mu^* = \underset{\mu \in \{0,\,1\}^{|A|\times|T|}}{\operatorname{argmax}} \sum_{a_i \in A} \sum_{t_j \in T} \mu_{a_i,t_j} \times f(a_i, t_j) \tag{3.1}$$

We mention here that in specific applications, some constraints can be specified to make it possible or not for an agent to accomplish a task.

3.2.2 Allocation of tasks requiring many agents

As presented in the section 2.2, a task can be single-agent or multi-agent. A formalisation concept of the multi-agent aspect of these tasks is the coalition formation paradigm, defined in section 1.3.5.

The allocation of multi-agent tasks can be modeled as follows:

- $A = \{a_1, a_2, \ldots, a_{|A|}\}$ is a set of agents.
- $T = \{t_1, t_2, \ldots, t_{|T|}\}$ is a set of tasks.
- $f : 2^A \times T \rightarrow \mathbb{R} \cup \{\perp\}$ is a utility function. It is a function that associates a cardinal utility value $f(s, t_j)$ in $\mathbb{R}$ to a pair made of an agents subset $s \subseteq A$ and a task $t_j \in T$. The $\perp$ symbol is used to represent that some combinations of the values of the variables in $s \in 2^A$ are not allowed for the task t_j.

This allocation output, as a coalition formation generation process, is a coalition structure S that covers all the tasks in T such as $S = \{C_{t_1}, C_{t_2}, \ldots, C_{t_{|T|}}\}$ where $\forall t_j \in T$, the coalition $C_{t_j} \in S$ is composed of the agents to which the task t_j is allocated.

Similarly, the goal is to find an allocation of agents, *i.e.* a coalition structure that maximises the value of the global utility under a certain set of constraints (that are application-dependent) with:

$$S^* = \underset{S \subseteq 2^A}{\operatorname{argmax}} \sum_{C_j \in CS} \sum_{t_j \in T} f(C_j, t_j) \tag{3.2}$$

Since we are in the Multi-Robot (MR)-ST case according to Gerkey's taxonomy (cf. section 2.2), all the coalitions are disjoint : $i \neq j \Rightarrow C_i \cap C_j = \emptyset$.

3.2.3 Discussion

The task allocation formalisms we outlined in previous sections are general and describe the problem at a very high level. We presented them to spot the details we want to focus on for our interdependent task allocation problem formalism.

First, is necessary to consider the agents-tasks matching in terms of skills and needs before considering the quality of the allocation. This is why introducing these elements is essential. Second, since we consider a multi-agent tasks case, our task allocation problem can be formalised under the coalition formation paradigm. Finally, since we are dealing with interdependent tasks, the utility function cannot be separated from the different tasks and should be globally designed.

3.3 Modelling Interdependent Task Allocation

We aim to fill the lack of decentralised allocation solutions for tasks with general types of interdependencies. We propose in the next section a complete generic problem formulation that will be adopted for the rest of this work.

Here we consider the problem of decentralised allocation for a set of interdependent tasks $T = \{t_1, t_2, \ldots, t_{|T|}\}$ necessary to accomplish a global task T, to a set of heterogeneous cooperative agents $A = \{a_1, a_2, \ldots, a_{|A|}\}$. The global task T represents the global goal that the team of the cooperative agents has to achieve.

3.3.1 Agents

A specific profile can describe each agent. For instance, in a given application, an agent can be described by its type, ownership or not of a resource, position on a grid, remaining fuel, or distances to the different tasks.

▶ **Definition 3.1 (Agent characteristics).**
Consider a set of heterogeneous agents $A = \{a_1, a_2, \ldots, a_{|A|}\}$.

An agent $a_i \in A$ is described by a set of attributes $X = \{x_1, \ldots, x_{|X|}\}$ where $x_k(a_i)$ denotes the value of a_i under attribute k. Without loss of generality, we consider each attribute as a function $x_k : A \rightarrow D_k$, D_k is the domain of values for attribute k. We call these attributes *agent characteristics.* ◀

We notice here that our model characteristics capture the **qualitative** values, like the type and possession of a resource, and the **quantitative** values, like the position, quantity of the remaining fuel, and distances to tasks simultaneously.

▶ **Example 3.2.**
Let us consider a set of six agents $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ scattered on a square grid of 8m side length.

The agents are described by three characteristics $X = \{x_1, x_2, x_3\}$. Notably, the agents are characterised by their remaining energy x_1, their location x_2 and their payload x_3 (either a normal camera C or a thermal one R).

Agent characteristics domains and values are presented in table 3.1. ◀

x_k	D_k	$x_k(a_1)$	$x_k(a_2)$	$x_k(a_3)$	$x_k(a_4)$	$x_k(a_5)$	$x_k(a_6)$
x_1	$[0, 10]$	4	4	7	8	9	6
x_2	$[0, 10]^2$	$(0, 0)$	$(3, 3)$	$(1, 2)$	$(5, 3)$	$(6, 2)$	$(6, 5)$
x_3	$\{C, R\}$	C	R	C	C	R	C

Table 3.1: Example of agent characteristics

3.3.2 Tasks

The agents have to perform tasks as defined below:

▶ **Definition 3.3 (Single task requirements).**
A task $t_j \in T$ is described by a set of attributes $\Gamma_{t_j}=\{\gamma_1, \ldots \gamma_m\}$ where $\gamma_l(t_j)$ is

the value of task t_j under the attribute l. Without loss of generality, we consider each attribute as a function $\gamma_l : T \longrightarrow K_l$, K_l is the domain of values for attribute l. We call the attributes Γ_{t_j} *single task requirements.*

◀

As seen in agent characteristics, task requirements also depend on the application and can be **qualitative** or **quantitative**. A requirement concerning a specific type of resources or agents is a qualitative requirement. By contrast, a minimum number of available agents or maximal distances are quantitative requirements. The following example illustrates the established concepts.

Example 3.2, continued.
In continuation to our example, we suppose that agents must perform three different tasks, $T = \{t_1, t_2, t_3\}$.

Each task has four requirements $\Gamma = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$. *We consider here that all the tasks have the same set of single requirements.* γ_1 *states the minimum energy necessary for each task to perform the task.* γ_2 *represents a composed requirement stating the position of the task, and the maximal allowed distance between that position and each of the agents positions.* γ_3 *states the set of the minimum required typed payloads for the task brought by all the agents (it is possible to have more). Finally,* γ_4 *states the minimum number of agents, that possess a payload, needed to accomplish the task.*

Table 3.2 presents the requirements for each task t_j *in* T.

γ_l	K_l	$\gamma_l(t_1)$	$\gamma_l(t_2)$	$\gamma_l(t_3)$
γ_1	$[0, 10]$	≥ 3	≥ 7	≥ 6
γ_2	$\langle [0, 10]^2, [0, 10] \rangle$	$\langle (3, 0), \leq 4 \rangle$	$\langle (4, 4), \leq 10 \rangle$	$\langle (2, 6), \leq 7 \rangle$
γ_3	$\langle (4, 4), 10 \rangle$	$\supseteq \{(C, 1)\}$	$\supseteq \{(C, 1), (R, 1)\}$	$\supseteq \{(C, 1)\}$
γ_4	$[\![1, 4]\!]$	≥ 1	≥ 2	≥ 1

Table 3.2: Example of single task requirements

We mention that in our example, all the tasks have the same set of requirements $\Gamma = \Gamma_{t_1} = \Gamma_{t_2} = \Gamma_{t_3}$ *with different attribute values for each task.*

Besides single task requirements, verifying the mapping between each task and some agents, macro-level requirements may be required to express the presence of interdependencies and to check their manageability.

▶ **Definition 3.4 (Task combination requirements).**
Consider a set of attributes $\Lambda_{cbt_j}=\{\lambda_1, .., \lambda_n\}$ concerning task combinations such that $\lambda_l(cbt_j)$ is the value of task combination $cbt_j \in T_{cbt}$, $T_{cbt} \subseteq 2^T$, under attribute l. Without loss of generality, we consider each attribute as a function $\lambda_l : T_{cbt} \rightarrow F_l$, F_l is the domain of values for attribute l. We call the attributes Λ_{cbt_j} *task combinations requirements.* ◀

Examples of single task requirements can be the number of agents that allow accomplishing a task or other application related constraints that have to be satisfied for the tasks to be accomplished. Task combination requirements can be seen as constraints that have to be satisfied to address the interdependence among tasks (for example, temporal constraints imposing accomplishment order).

These task combination requirements are a general representation of the inter-task dependencies. In real-world applications, several tasks can have joint requirements. For example, in a given situation, two tasks (or more) may need to have a minimal number of resources as a single task requirement and need mutually an additive resource, that of the tasks' coalitions might lend if needed. A dependency, or a task combinations requirement, can even concern all the tasks at once.

We continue in following our example by illustrating the task combination requirements.

Example 3.2, continued.
We have the set of task combinations $T_{cbt} = \{\{t_1, t_2, t_3\}, \{t_2, t_3\}, \{t_1, t_3\}\}$ *that we denote* $T_{cbt} = \{cbt_1, cbt_2, cbt_3\}$.

For each cbt_j *in* T_{cbt}*, one task combination requirement is defined and denoted* $\Lambda_{cbt_j} = \{\lambda_j\}$.

For the first task combination $cbt_1 = \{t_1, t_2, t_3\}$*, the task combination requirement* λ_1 *sets the maximum number of agents allocated to all the tasks (in certain applications, it may be necessary to keep some agents available for a potential task coming in the way). For the second task combination* $cbt_2 = \{t_1, t_2\}$ *and the*

third task combination $cbt_3 = \{t_1, t_3\}$, task combination requirements λ_2 and λ_3 are defined for each one, respectively. They both state that for each task combination, in addition to the locally required payload (expressed by the single task requirements) an additional payload of type C is required to be a joker payload in case of need.

Below, table 3.3 outlines the task combinations requirements for each task combination $cbt_j \in T_{cbt}$ and each associated requirement $\lambda_j \in \Lambda_{T_{cbt_j}}$.

cbt_j	F_j	λ_j	$\lambda_j(cbt_j)$
$cbt_1 = \{t_1, t_2, t_3\}$	λ_1	$[\![1, \lvert A\rvert]\!]$	≤ 5
$cbt_2 = \{t_1, t_2\}$	λ_2	$\{C, T\} \times [\![1, 2]\!]$	$\supseteq \{(C, 1)\}$
$cbt_3 = \{t_1, t_3\}$	λ_3	$\{C, T\} \times [\![1, 2]\!]$	$\supseteq \{(C, 1)\}$

Table 3.3: Example of task combination requirements

To accomplish the global task T, all the single task requirements and task combination requirements should be fulfilled (as it will be defined later).

Task requirements and task combination requirements are easily translated into **constraints**. Single task requirements (for example, the cardinality of the coalitions that allow accomplishing a task or other application-related constraints) can be seen as constraints defined locally on tasks. These must be satisfied for the tasks to be performed. Task combination requirements can be seen as constraints over the global task T that must be satisfied to address dependencies among tasks (for example, temporal constraints imposing accomplishment order or dependence of a task's accomplishment on the achievement degree of another one).

Combined characteristics of a set of agents may allow fulfilling task requirements, or generate conflicts with such requirements.

3.3.3 Fulfilment

▶ **Definition 3.5 (Fulfillment relation).**
Let $s = \{a_1, ..., a_{|s|}\}$ be 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 (with respect to a mathematical operator, for example $=, \leq, \geq$,... 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 $\exists x_k \in cc$ such that x_k is in conflict with this requirement γ_l denoted as $x_k \not\bowtie \gamma_l$. ◀

When a subset of agents is in conflict on a task requirement, it means that the subset cannot "contribute" to the task according to this specific requirement. The violation of the requirement cannot be amended adding any agent or agents of any possible combination of characteristics.

To illustrate these concepts, we present examples of their application related to the example we started earlier.

Example 3.2, continued.
To assess how a set of agents can satisfy some task requirements, particularly for the second tasks requirement γ_2, we first compute agent distances from task positions (see table 3.4). Agents scattering is represented in figure 3.1. For the sake of simplicity and without loss of generality, we use here Manhattan distances.

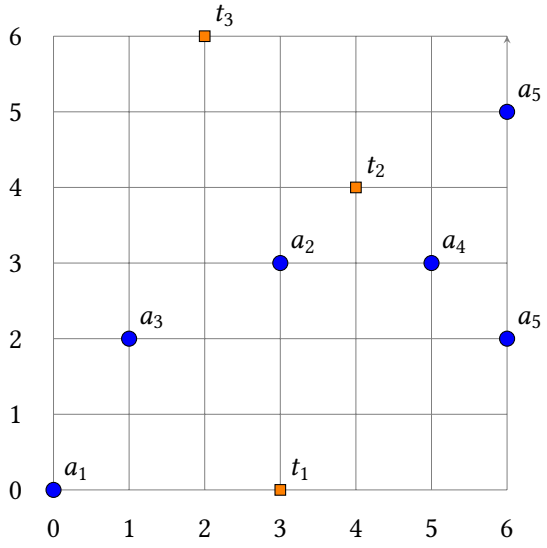


Figure 3.1: Agents and tasks scattered in the grid

$dist(a_i, t_j)$	a_1	a_2	a_3	a_4	a_5	a_6
t_1	3	3	4	5	5	8
t_2	8	2	5	2	4	3
t_3	8	4	5	6	8	5

Table 3.4: Manhattan distance between agents and tasks in the example

Here are some examples:

- $\{a_1\} \circlearrowleft \gamma_2(t_1)$
  a_1 respects the maximal distance from t_1, then a_1 satisfies $\gamma_2(t_1)$.

- $\{a_1, a_3, a_4\} \circlearrowleft \gamma_1(t_2)$
  a_1, a_2 and a_3 have the minimal energy needed to perform t_2, then $\{a_1, a_3, a_4\}$ satisfies $\gamma_1(t_2)$.

- $\{a_1, a_2\} \circlearrowleft \gamma_3(t_1)$
  a_1 brings the resource C, then $\{a_1, a_2\}$ satisfies $\gamma_3(t_1)$.

- $a_2 \not\circlearrowleft \gamma_1(t_2)$
  a_2 does not have enough energy to perform t_2, then a_2 is in conflict with $\gamma_1(t_2)$.

- $\{a_4, a_5\} \not\circlearrowleft \gamma_2(t_3)$
  a_5 does not respect the maximal distance from t_3, then $\{a_4, a_5\}$ is in conflict with $\gamma_2(t_3)$.

For the task combination requirements :

- *For $ss = \{\{a_1, a_2\}, \{a_3, a_4\}, \{a_5, a_6\}\}$ and $cbt_1 = \{t_1, t_2, t_3\}$:*
  $ss \not\circlearrowleft \lambda_1(cbt_1)$
  ss contains six agents, then it is in conflict with $\lambda_1(\{t_1, t_2, t_3\})$.

- *For $ss = \{\{a_1, a_2\}, \{a_3, a_5\}\}$ and $cbt_3 = \{t_1, t_3\}$:*
  $ss \not\circlearrowleft \lambda_2(cbt_3)$
  An additional common C payload is missing in ss, then ss is in conflict with $\lambda_2(cbt_2)$.

3.3.4 Coalition Structures

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

▶ **Definition 3.6 (Coalitions).**
Let a set of agents $A = \{a_1, a_2, ..., a_{|A|}\}$, a set of tasks $T = \{t_1, t_2, ..., t_{|T|}\}$ composing the global task T and $\tau : A \rightarrow T$ a function assigning an agent a_i to a task t_j when $\exists x_k \in X_{a_i}, \exists \gamma_l \in \Gamma_{t_j}$ such that $x_k \bowtie \gamma_l$, and $\nexists x_m \in X_{a_i}$ such that $x_m \bowtie \gamma_p$ for any $\gamma_p \in \Gamma_{t_j}$ with $p \neq l$. A *coalition* $C_{t_j} \in 2^A$ whose task is t_j is a subset of agents $C_{t_j} \in 2^A$ such that $C_{t_j} = \{a_i \in A \mid \tau(a_i) = t_j\}$. ◀

To perform a global task T we need a set of coalitions S, called *coalition structure*. Each coalition $C_{t_j} \in S$ is assigned a task $t_j \in T$. When S can accomplish T we call it a *feasible coalition structure*. Formally:

▶ **Definition 3.7 (Feasible coalition structure).**
Let a global task $T = \{t_1, t_2, ..., t_{|T|}\}$ and a coalition structure $S = \{C_{t_1}, C_{t_2}, ..., C_{t_{|T|}}\}$ over the set of tasks T. S is a *feasible coalition structure* (called also feasible solution) denoted S^f if and only if $\forall t_j \in T$ such that $C_{t_j} \in S^f$ it holds that $\forall \gamma_l \in \Gamma_{t_j}$, there exists a set $s_{cc} \subseteq C_{t_j}$ such that $s_{cc} \cup \gamma_l(t_j)$ and if Λ_{cbt} is a set of requirements concerning the combination of tasks then $X_{Sf} \bowtie \Lambda_{cbt}$. ◀

Note that agents are *single-task*, i.e., they can accomplish only one task at a time (see [Gerkey et al. 2004]). Thus, an agent should exist in no more than one coalition of the coalition structure. From the perspective of coalitions, this is equivalent to stating that two different coalitions cannot contain the same agent, formally: $C_{t_j}, C_{t_k} \in S$, $j \neq k$ then $C_{t_j} \bigcap C_{t_k} = \varnothing$. However, some agents can have no task assigned to them, and thus not exist in any coalition of the structure, formally: $\bigcup_{j=1}^{|T|} C_{t_j} \subseteq A$.

Example 3.2, continued.
$S_1^f = \{\{a_1\}, \{a_3, a_4, a_5\}, \{a_6\}\}$ is a feasible coalition structure: the requirements of the three tasks are all fulfilled, as well as the task combination requirements over the global task.

Another feasible coalition structure is : $S_2^f = \{\{a_1, a_2\}, \{a_3, a_4, a_5\}, \{a_6\}\}$.

However, $S_3 = \{\{a_1, a_2\}, \{a_3, a_4\}, \{a_6\}\}$ is not a feasible coalition structure: task t_2 is not fulfilled since the requirement $\gamma_3(t_2)$ is not satisfied by the corresponding coalition $C_{t_2} = \{a_3, a_4\}$. This is because a_3 and a_4 possess only a C camera each

and thus cannot satisfy the requirement γ_3 of having at least one C camera and one R camera for the task t_2.

Another infeasible coalition structure is $S_4 = \{\{a_1, a_2\}, \{a_3, a_5\}, \{a_4, a_6\}\}$. Even if all the single task requirements are fulfilled by the coalitions for all the tasks, the requirement $\lambda_2(\{t_1, t_2\})$ over the global task is not.

Similarly $S_5 = \{\{a_1, a_2\}, \{a_3, a_4, a_5\}, \{a_6\}\}$ is not a feasible coalition structure either since the requirement $\lambda_1(\{t_1, t_2, t_3\})$ is not satisfied.

Let us note CS is the set of all the possible coalition structures.

Since agents may be added or removed as things progress in the task allocation process, we need coalition structure modification functions.

- $\oplus$ is the addition function $\oplus : CS \times (A \times T) \rightarrow CS$, used to add a specific agent to a specific coalition.
  The resulting coalition structure from $S \oplus (a_i, t_j)$ is a coalition structure in which $C_{t_j} \leftarrow C_{t_j} \bigcup \{a_i\}$, unless a_i was already in C_{t_j} in S, and $\forall k \neq j$ C_{t_k} are the same as in S.

- $\ominus$ is the removal function $\ominus : CS \times (A \times T) \rightarrow CS$, used to remove a specific agent from a specific coalition.
  The resulting coalition structure from $S \ominus (a_i, t_j)$ is a coalition structure in which $C_{t_j} \leftarrow C_{t_j} \setminus \{a_i\}$ and $\forall k \neq j$, C_{t_k} do not change.

3.3.5 Global Utility Function

A utility function is defined following the application needs. It should consider the globality of the tasks to cover the interdependencies between tasks and not be simply a sum of utilities over tasks.

We can now suggest a definition of how to evaluate coalition structures with respect to the accomplishment of a global task.

▶ **Definition 3.8 (Coalition Structure Evaluation).**
Consider $CS = \{S_1, \ldots, S_n\}$ be the set of all possible coalition structures that could be assigned to a global task T. We introduce a set of criteria G such that for each $g_k \in G$ there exists a weak order G_k upon the set CS, $G_k \subseteq CS^2$ such that if $(S, S') \subseteq G_k$, then $S \geq S'$ and $\exists g_k : CS \rightarrow \mathbb{R}$, $g_k(S) \geq g_k(S')$. ◀

Then, without loss of generality, we can define the following decision problem: $\forall k \max_{S_h \in CS} g_k(S_h)$, which should identify the coalition structures S_h maximising "simultaneously" the performance of such structures upon all the criteria in G for a global task T.

Note that we consider interdependent tasks. Hence, it is necessary to define a function that evaluates a coalition structure S as a whole considering the interdependence among the coalitions in the structure. Evaluating the whole structure based only on individual evaluations of the coalitions would not be sufficient due to interdependencies between tasks and thus between coalitions.

Provided the conditions of commensurability, compensation and preferential independence are satisfied among the criteria in G (see [Bouyssou et al. 2000]), a global additive value function u_{global} of the following type is applicable. This function is of the type:

$$u_{global}(S) = \Sigma_k u_k(g_k(S)) \tag{3.3}$$

If $u_k(g_k(S))$ is normalized to the interval [0,1] we have:

$$u_{global}(S) = \Sigma_k w_k \bar{u}_k(g_k(S)) \tag{3.4}$$

where $\bar{u}_k$ represents the normalised functions and w_k are weights representing the relative importance of the criteria.

We want to design an algorithm that finds a feasible coalition structure S^{*f} that maximises $u_{global}(S)$.

$$u_{global}(S^{*f}) = \max_{S \in CS} u_{global}(S) \tag{3.5}$$

For our specific example of a utility function:

$$u_{global}(S^{*f}) = \max_{S \in CS} \Sigma_k w_k \bar{u}_k(g_k(S)) \tag{3.6}$$

However our approach is generic and therefore other types of evaluation functions could be considered.

Example 3.2, continued.
Several utility functions can be defined for this running example. Here, we define 5 criteria to evaluate a coalition structure S:

- *Number of agents that are near (with respect to a given threshold) the task they are allocated:*

  $g_1(S) = \Sigma_{C_{t_j} \in S} |\{a_i \in C_{t_j} | dist(x_1(a_i), \omega_2(t_j)_1 \leq th_1^{t_j}\}|$

- *Number of agents whom energy is greater than a threshold:*

  $g_2(S) = \Sigma_{C_{t_j} \in S} |\{a_i \in C_{t_j} | x_1(a_i) \geq th_2^{t_j}\}|$

- *Number of agents that bring the needed resources:*

  $g_3(S) = \Sigma_{C_{t_j} \in S} |\{a_i \in C_{t_j} | x_3(a_i) = th_3^{t_j}\}|$

- *Number of coalitions which allocated agents number respects the minimum threshold of the coalition's task:*

  $g_4(S) = \Sigma_{C_{t_j} \in S} (1 \text{ if } |C_{t_j}| \geq \gamma_4(t_j);\ \text{else } |C_{t_j}|/\gamma_4(t_j))$

- *And on the global level:*

  $g_5(S) = 1$ *if the maximum number of agents in* S *allocated to the global task* T *(and thus to the sum of the tasks) is respected,* $g_5(S) = 0$ *otherwise.*

Assuming that the weights are the same for all criteria, we define the functions $\bar{u}_k(g_k(S))$ *(normalised between 0 and 1) as follows, to compute the utility of each coalition structure.*

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

If we assume that $th_1^{t_1} = th_1^{t_2} = 3$, $th_2^{t_1} = 6$, $th_2^{t_2} = 9$, $th_3^{t_1} = C$, $th_3^{t_2} = C \vee R$, *we have:*

$$u_{global}(S_1^f) = \frac{1}{5}\left(\frac{2+3+4}{6} + \frac{3}{3} + 1\right) = 0.7$$

$$u_{global}(S_2^f) = \frac{1}{5}\left(\frac{3+4+5}{6} + \frac{3}{3} + 1\right) = 0.8$$

3.4 Conclusion

This thesis aims to address the general case of task interdependencies for the multi-agent task allocation problem. Therefore, this chapter sets the stage by introducing a modelisation of this problem.

We provide generic modelling of the interdependent task allocation problem components, properties, and evaluation techniques, along with an illustrative example. In this formalisation, agents are described by characteristics, and tasks are described by requirements. Depending on the application, the agents' characteristics and tasks' requirements can take any possible values, qualitative or quantitative. The problem's goal is to find a coalition structure in which coalitions are assigned to the tasks. The due coalition structure has to fulfil the defined requirements and thus be feasible. Tasks dependencies are modelled in a general manner and on two different levels. First, the dependencies concern the task combination requirements and thus the common feasibility constraints. Second, the interdependencies are represented by a global task allocation utility that takes the whole coalition structure as a parameter. Hence, the utility function tackles the dependencies between the coalitions of the coalition structure in question and thus between the tasks. The suggested model is used for the rest of this work. This modelisation has been presented in [Ahmadoun et al. 2021].

In the next chapter, we present algorithms that are proposed as solutions to our problem.

4 Solving the Interdependent Task Allocation problem

Contents

4.1 Introduction **64**
4.2 Our Approach **64**
4.3 Feasible Interdependent Coalition Structure Anytime Method (FICSAM) **66**
4.3.1 General Process 66
4.3.2 Decision Process 70
4.3.3 Feasible Structure Generation Process 76
4.3.4 Feasibility Check Process 78
4.3.5 Run-Through Example 79
4.4 Improved Feasible Interdependent Coalition Structure Anytime Method (IFICSAM) **80**
4.4.1 Swap Improvement Process 81
4.4.2 Combinations Improvement Process 84
4.4.3 CSP Improvement Process 85
4.5 Complexity Analysis **93**
4.6 Empirical Evaluation **93**
4.6.1 The Scenarios Generator 94
4.6.2 Implementation 95
4.6.3 Setup 95
4.6.4 Centralised Solution 96
4.6.5 Results Evaluation 100
4.6.6 Results for the many-to-one swapping improvements 103
4.7 Discussion **105**

4.1 Introduction

This chapter presents the algorithms that are proposed to the interdependent task allocation problem. In practical use cases, agents need to find quickly a task allocation where the task requirements are met. If there is enough time, they may improve the allocation they found so that they can improve the global utility and thus their team's performance. We thus present algorithms that allow a two stages decision. The goal of the first stage is to find a coalition structure in which agents fulfill all the task requirements. In the second stage, agents start from the feasible coalition structure found in stage 1 and try to find another one with a greater global utility. These algorithms allow the decision making process to be anytime and applicable to real-world contexts. The suggested approach is published in a paper [Ahmadoun et al. 2021] and a patent [Ahmadoun et al. 2021].

4.2 Our Approach

As stated above, the main aim of our work is to find the best feasible coalition structure with respect to the global utility u_{global} if it exists, or detect the nonexistence of a feasible coalition structure as early as possible. We propose a decentralised solution approach based on token-passing among the agents, candidate members of different coalitions assigned to the tasks composing the global task.

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. For example, in an application with a specific hierarchy, the token may be sent from the most important, and thus possibly best candidate in the team, to the less important, the one with fewer resources and qualities for the application. In a spatial-placed application, the token can be passed from an agent to its nearest agent.

At the beginning, each agent knows its own characteristics and the global task T to be accomplished. Information about the other agents and the evolution of the coalition structure formation arrives via the token-passing process. When agent a_i sends the token to agent a_j, the next agent (defined according to different application-based criteria) adds information on the best (with respect

to u_{global}) feasible coalition structure S formed so far, the accumulated expertise (*i.e.* characteristics) X_S of the participating agents in S, and the number of agents nd who have not changed the coalition structure (have not decided to join a coalition).

Holding the token, agent a_i may decide to join (or initiate) a coalition C_{t_j} assigned to task t_j, if it can contribute to the accomplishment of task t_j or if its participation can increase the global utility function u_{global} . Where applicable, agent a_i updates the information it received with the changes it had applied. Then, agent a_i passes the token to the next agent. If it has not joined a coalition, it passes the token to the next agent by forwarding the information from the previous agent.

We assume that agents can exchange messages and we describe the communication protocol that organizes the messages exchange between agents (in section 4.3.1). Yet, the underlying communication infrastructure (*i.e.* studying questions such as whether there is a message board, the agents acknowledge message receipts, communication is asynchronous, there is any noise) is beyond this work's scope.

Concretely, our mechanism has two stages:

Stage I coincides with the first round. In this stage, agents try to find 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} (*i.e.*, the improvement of the u_{global} value is not a precondition).
If no S^f is found in stage I, no such S^f exists considering the requirements of the tasks and the characteristics of the available agents.
We call the method implementing this stage FICSAM.

Stage II implements our second method incrementally improving the feasible coalition structure found so far.
This method starts from round 2. Once a feasible coalition structure was found at the end of round 1, agents improve u_{global} via replacements or swaps between single or groups of agents. The resulting improved structure must preserve feasibility by respecting all the single task and task combination requirements.

We call the method implementing the first stage combined to this second stage IFICSAM.

In the following sections, we will start by presenting the general algorithm that allows an agent to join (or not) a specific coalition to distributively create a feasible coalition structure. We will then detail different methods we propose for an agent to decide and participate in improving the global utility.

4.3 Feasible Interdependent Coalition Structure Anytime Method (FICSAM)

4.3.1 General Process

Here, we present the general process for the decentralisation of the coalition formation generation.

Algorithm 1 implements the behavior of an agent participating in the process. This agent acts either as the process initiator (lines 1-15) or as a candidate member of some coalitions of a certain coalition structure S (lines 16-34). These two roles are illustrated in their globality in figure 4.1 for the initiator agent and in figure 4.2 for the other agents, or the initiator agent when it receives a message later on in the process.

It is to be mentioned here that the messages in these algorithms follow the FIPA ACL messages structure [Fipa 2002]. Thus, a message is structured this way: $message = performative(sender, receiver, < content >)$. The performative denotes the type of the communicative act (either an informative or proposition message in our case). The first parameter represents the agent that sends the message and the second parameter represents the agent receiving it. Finally, in our case, the content of the message depends on the message's type that is represented in the first parameter of the content (either "gt" for the token passage messages, "ats" for the anytime solution messages or "end" for the end process messages).

As described in algorithm 1, an initiator agent a_i starts its activity by initializing its maximal coalition structure S_{max} (regarding the global utility u_{global}) and its current coalition structure S and incrementing its round to the first

Algorithm 1: coalition-formation(a_i, T, A, Γ_T, Λ_{cbt}, $msg(perf(a_l, a_i, < content >))$)

1 **if** agent a_i makes the first proposal **then**
2 $S_{max} \leftarrow \emptyset$; $S \leftarrow \emptyset$; $R[i] \leftarrow R[i] + 1$
3 $X_S \leftarrow \emptyset$; $T_{a_i} \leftarrow \emptyset$; $nd \leftarrow 0$
4 **for** $t_j \in T$ such that $a_i \bowtie \Gamma_{t_j}$ **do**
5 $T_{a_i} \leftarrow T_{a_i} \cup \{t_j\}$
6 Get $t_j \in T_{a_i}$
7 $S \leftarrow S \oplus (a_i, t_j)$
8 $X_S \leftarrow X_S \cup X_{a_i}$
9 $is\_feasible(S) \leftarrow check\text{-}feasibility(T, \Gamma_T, \Lambda_{cbt}, A, S)$
10 **if** $is\_feasible(S)$ **then**
11 $S_{max} \leftarrow S$
12 **for** $a_l \in A$ **do**
13 send($propose(a_i, a_l, \langle$"$ats$"$, S, X_S, R, nd\rangle)$)
14 send($propose(a_i, next(a_i), \langle$"$gt$"$, S, X_S, R, nd\rangle)$)
15 **while** true **do**
16 Get $msg(perf(a_l, a_i, < content >))$
17 **switch** $perf(a_l, a_i, < content >)$ **do**
18 **case** $propose(a_l, a_i, \langle$"$ats$"$, S, X_S, R, nd\rangle)$ **do**
19 **if** *requirement_anytime_solution* **then**
20 $decision \leftarrow$ "$success$"
21 End of coalition formation process with a feasible coalition structure
22 **else**
23 $is\_feasible(S) \leftarrow true$
24 call $decide(a_i, X_{a_i}, T, \Gamma_T, \Lambda_{cdt}, S, X_S, R, nd, A, is\_feasible(S))$
25 **case** $propose(a_l, a_i, \langle$"$gt$"$, S, X_S, R, nd\rangle)$ **do**
26 $is\_feasible(S) \leftarrow false$
27 call $decide(a_i, X_{a_i}, T, \Gamma_T, \Lambda_{cdt}, S, X_S, R, nd, A, is\_feasible(S))$
28 **case** $inform(a_l, a_i, \langle$"end"$, \emptyset, \emptyset, 1\rangle)$ **do**
29 $decision \leftarrow$ "$failure$"
30 End of the coalition process in first round with no feasible coalition structure found
31 **case** $inform(a_l, a_i, \langle$"end"$, S, X_S, R\rangle)$ **do**
32 $decision \leftarrow$ "$success$"
33 End of coalition formation process with a feasible and improved (with respect to global utility) solution

round $R[i]$ (line 2). The agent a_i initiates (line 3) X_S the set of characteristics of agents in S, T_{a_i} the set of tasks it can perform, and *nd* the number of unchanged decisions (*i.e.*, how many agents did not change coalition structure until now). It then continues by building T_{a_i}, the tasks with requirements that match its characteristics (*i.e.*, $X_{a_i} \bowtie \Gamma_{t_j}$) and thus can perform (lines 4–5). It then picks one of them, say t_j (line 6), and initializes a coalition in the structure S (line 7) that is assigned to t_j. We mention here that S is at the beginning a coalition structure with empty coalitions. Following this, the agent adds its characteristics to X_S (initially empty) (line 8) that accumulate the characteristics of all the members of the coalitions in structure S. Next, it checks the feasibility of the structure S (line 9) by using the procedure *check-feasibility* (explained in section 4.3.4).

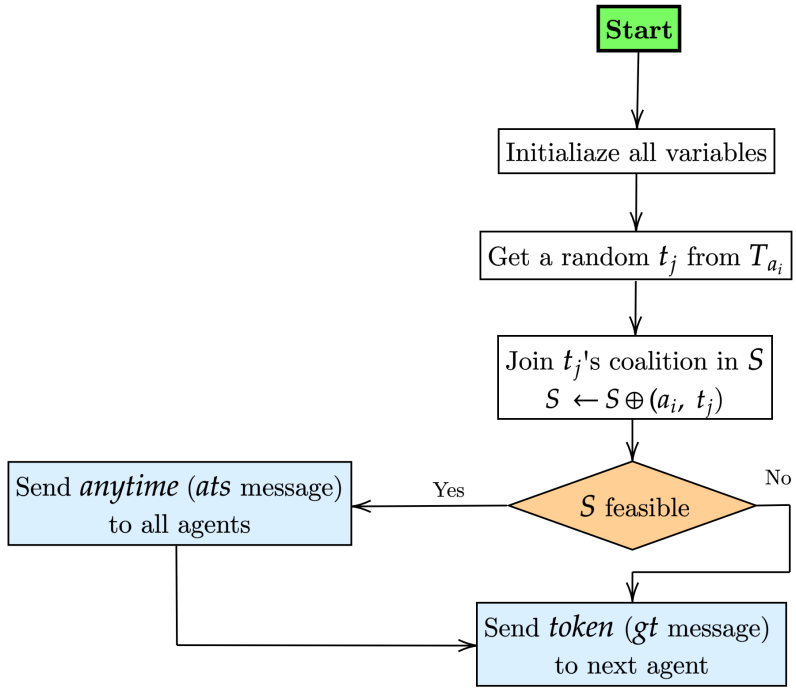


Figure 4.1: First agent process

In the unlikely case where structure S is feasible (e.g., if T contains only the task t_j and t_j is fulfilled by this exact initiator agent), this coalition structure

becomes an anytime coalition structure. Agent a_i considers S as a feasible coalition structure to explore and sends this proposal to all the agents in A via a message $propose(a_l, a_i, \langle "ats", S, X_S, R, nd\rangle)$ (lines 12-13). The first parameter of the message content "ats", for *anytime solution*, means it is a message that informs that an anytime solution has been found. Then, in both cases, whether coalition structure S is feasible or not, agent a_i initializes the decentralised process of coalition structure formation by sending a message to the next agent a_j (line 14). With this message, agent a_i passes the token to agent a_j, who enters the process. a_i informs a_j about the structure S, the characteristics gathered so far X_S, the current round R (*i.e.*, first round), and the number of agents that did not change their decision in the last round nd (here zero). Figure 4.1 presents globally the main steps the first agent goes through in the process.

When an agent a_i acts as a candidate member of a coalition structure, its activity depends on the messages it receives from other agents $msg(perf(a_l, a_i, < content >))$ (lines 16-17). This is illustrated in figure 4.2. Three types of messages exist.

In the case of a $propose(a_l, a_i, \langle "ats", S, X_S, R, nd\rangle)$ message, if an anytime solution is required, the coalition formation process terminates with S as the best (with respect to u_{global}) feasible coalition structure found so far. This can occur either at the end of the first round or in the middle of any other round (*i.e.* all the available agents do not have the possibility to check whether they can contribute to improving this feasible coalition structure).

Otherwise, the receiving agent considers that S is a feasible coalition structure (line 23) that might be further improved (with respect to u_{global}). For that, the receiving agent uses procedure *decide* (see algorithm 2). The message $propose(a_l, a_i, \langle "gt", S, X_S, R, nd\rangle)$ means that S is not a feasible coalition structure and the sender passes the token (gt meaning give token) to the receiving agent who uses procedure *decide* for looking whether it can contribute to the search for a feasible coalition structure. The message $inform(a_l, a_i, \langle "end", \emptyset, \emptyset, 1\rangle)$ informs the agents that no feasible coalition structure has been found at the end of first round and therefore the process ends with failure, while message $inform(a_l, a_i, \langle "end", S, X_S, R\rangle)$ informs agents that the process ends with a feasible and improved (with respect to the feasible coalition structure found in the first round) solution, meaning that none of the available agents can furthermore improve the current feasible coalition structure.

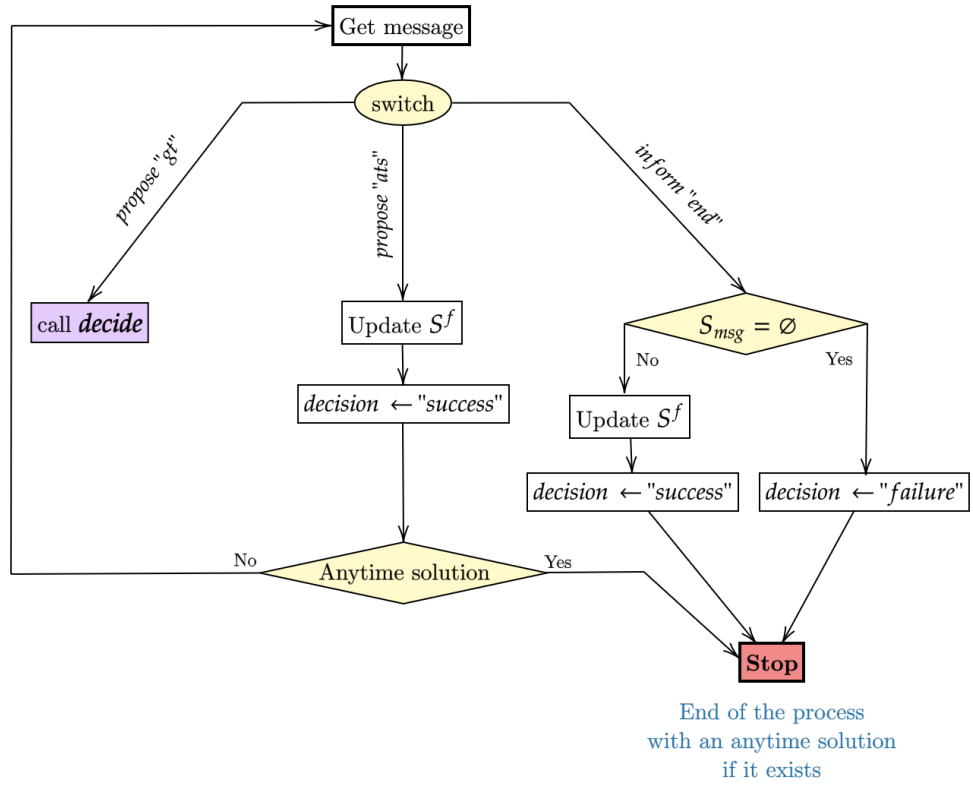


Figure 4.2: Global process

4.3.2 Decision Process

Algorithm 2 describes the decision process of an agent a_i that is triggered when it gets a message from another agent during the coalition formation process. The agent's decision depends on the received message content and the round R in which the process is found. Table 4.1 gathers the description of the parameters of the decision algorithm.

First (line 3), agent a_i checks the feasibility of the received coalition structure S. This is done using a feasibility checking procedure (see section 4.3.4) according to the feasibility criteria of the given application. Notably, the feasibility is based on the single task and task combinations requirements of the application.

Parameter	Description
a_i	The agent running the decision algorithm
X_{a_i}	The agent a_i's characteristics
T	The list of tasks
Γ_T	The single task requirements of the tasks in T
Λ_{cbt}	The task combination requirements
S	The last received coalition structure
X_S	The characteristics of the agents in S's coalition
R	The rounds of the agents
nd	The number of unchanged decisions
A	The list of agents
$is\_feasible(S)$	A boolean parameter indicating if S is feasible or no

Table 4.1: Description of the algorithm 2's parameters

Then, if $R[i] = 1$, *i.e.*, the process is in the first round, and the received coalition structure S is not yet feasible, the agent computes a feasible structure, accounting for the set of tasks T that have to be accomplished, task-level and combination-level requirements Γ_T and Λ_{cbt}, and its potential contribution in case it joins S. This is done by using the procedure *s-f-st* (line 5) that implements a Constraint Satisfaction Problem (CSP) to generate a feasible coalition structure if it exists. More details about this procedure are presented in section 4.3.3.

As said, when agent a_i holds the token, it adds the information about its characteristics to X_S, the set of agent characteristics gathered by preceding agents. Thus, when agent a_i gets the token in the first round, it has to solve a locally centralised coalition formation problem based on the knowledge accumulated so far. This knowledge concerns more precisely the agents that have already joined the coalition structure and their characteristics. Based on this knowledge, agent a_i tries to find whether preceding agents, regarding their characteristics

in X_S, along with a_i are sufficient for satisfying all of the requirements of both individual tasks and task combinations.

As said above, requirements are modeled as hard constraints (see, for example [Dechter 2003]), and a CSP problem is solved (see section 4.3.3). Note that in the case where no feasible coalition structure is found in the first round of the decision process (*i.e.*, the CSP problem has no solution with the accumulated knowledge on agents until now), the process is to be terminated as there is no solution at all. In this case, the absence of a solution is proved the following way. Thanks to the transfer of the agents' characteristics in X_S with the token-passing, at the end of round 1 marking a complete passage of the token to all the agents, the last agent has complete knowledge of all the agents' characteristics. If a solution existed, the last agent would have found it using this knowledge running the CSP. Therefore, if all the agents received the token and the last one did not find a solution, it is concluded that no solution exists, and the process can stop by sending end process messages to all the agents. Otherwise, the process can proceed to gradually improve the initial feasible structure, as it will be discussed in the next section.

The use of this procedure during the first round allows to detect early on whether there is a feasible coalition structure, *i.e.* whether the characteristics of the available agents in A are sufficient to accomplish the tasks in T. Subsequently, either we seek afterwards to improve the performance of the coalition structure (with respect to the u_{global}) based on a decentralised approach (knowing that we have at least one feasible coalition structure) or we abandon the effort by avoiding to consume resources unnecessarily. That is why this procedure does not consider the maximization of u_{global} when looking for a feasible coalition structure during the first round. This avoids losing time and computation by trying to maximize the global utility for a problem for which we could discover later on that it has no solution considering the available resources (*i.e.*, the agents A). That also allows an anytime method, which finds an applicable solution in a minimal time and then improves it gradually.

Let's continue at line 6 where agent a_i checks the feasibility of the coalition structure S returned by the *s-f-st* algorithm (see section 4.3.3). The feasibility test is done using the procedure *check-feasibility* (see section 4.3.4). If the answer is positive that means that we found the first feasible coalition structure S^f. This feasible coalition structure S^f corresponds to an anytime solution and is sent by

Algorithm 2: decide($a_i, X_{a_i}, T, \Gamma_T, \Lambda_{cbt}, S, X_S, R, nd, A, is\_feasible(S)$)

```
1  S_old ← S
2  R[i] ← R[i] + 1
3  if R[i] = 1 then
4      if not(is_feasible(S)) then
5          S_max ← S
6          S ← s-f-st(T, Γ_T, Λ_cbt, {a_i} ∪ S, a_i, X_{a_i} ∪ X_{S\{a_i}})
7          is_feasible(S) ← check-feasibility(T, Γ_T, Λ_cbt, A, S)
8          if is_feasible(S) then
9              S^f ← S; nd ← 0
10             S_max ← S^f
11             for C_{t_j} ∈ S do
12                 for a_l ∈ C_{t_j} do
13                     send(propose(a_i, a_l, ⟨"ats", S, X_S, R, nd⟩))
14         else
15             if R ≠ ⟨1, ..., 1⟩ then
16                 T_{a_i} ← ∅
17                 for t_j ∈ T such that a_i ⋈ Γ_{t_j} do
18                     T_{a_i} ← T_{a_i} ∪ {t_j}
19                 Get t_j ∈ T_{a_i}
20                 S ← S ⊕ (a_i, t_j)
21                 X_S ← X_S ∪ X_{a_i}
22                 if u_global(S) > u_global(S_max) then
23                     S_max ← S; nd ← 0
24                     send(propose(a_i, next(a_i), ⟨"gt", S, X_S, R, nd⟩)
25                 nd ← 0
26                 send(propose(a_i, next(a_i), ⟨"gt", S, X_S,R, nd⟩))
27             else
28                 S^f ← ⟨[], [], ..., []⟩
29                 for C_{t_j} ∈ S do
30                     for a_l ∈ C_{t_j} do
31                         send(inform(a_i, a_l, ⟨"end", ∅, ∅, 1⟩))
32 else
33     S^f ← S
34     call improve_ificsam_variant(a_i, T_{a_i}, S) where
         variant ∈ {swap, combinations, bestcombinations}
35     is_feasible(S) ← check-feasibility(T, Γ_T, Λ_cbt, A, S)
36     if is_feasible(S) and S ≠ S^f then
37         S^f ← S; nd ← 0
38         for t_j ∈ T do
39             for a_l ∈ C_{t_j} do send(propose(a_i, a_l, ⟨"ats", S, X_S, R, nd⟩))
40     else S ← S_old
41 if nd < |A| and S = S_old then
42     nd ← nd + 1
43     send(propose(a_i, next(a_i), ⟨"gt", S, X_S, R, nd⟩))
44 else
45     for a_l ∈ A do
46         send(inform(a_i, a_l, ⟨"end", S, X_S, R⟩))
```

agent a_i to all the members of the coalitions of the structure formed so far, along with X_S and initialized counter $nd \leftarrow 0$ as it stands for "number of unchanged decisions". nd is initialized by an agent when a feasible new coalition structure is introduced in the process by an agent. It is incremented when an agent is unable to improve the current feasible coalition structure. This corresponds to a statement "nothing to say" and serves to terminate the whole process when a solution cannot be further improved by any available agents.

When agent a_i cannot find a feasible coalition structure with X_S (*i.e.*, the gathered characteristics so far), it examines ways of contribution to future feasible coalition structures that are found (if any exist) by agents that have not yet had the token in that round, provided it is not the last agent to receive the token. This is done by examining tasks it can perform with requirements that match its characteristics, *i.e.*, tasks in T_{a_i} ($X_{a_i} \bowtie \gamma_l$) (line 16). Then, a_i picks a task $t_j \in T_{a_i}$ (line 18), joins $C_{t_j} \in S$, and adds its characteristics to X_S.

As before, agent a_i checks whether the updated coalition structure S improves u_{global}. If it is the case, this coalition structure becomes the best among those (not feasible yet) built so far. In any case, agent a_i sends the token to the next agent, including S, X_S, nd, and R (here round 1) in the message. However, if agent a_i is the last to receive the token, it implies that there is no feasible coalition structure. a_i informs the agents in S that the process must end as a feasible coalition structure cannot be found.

If $R[i] > 1$, *i.e.*, the current round is greater than one, this means that a solution was already found (line 34), and this is where the FICSAM algorithm ends.

Lines $40 - 45$ concern the situation where a feasible coalition structure S is found and a_i cannot improve it. In this case, if all the agents have not received the token yet, *i.e.* $nd < |A|$, a_i increments nd and passes the token to the next agent. Otherwise (*i.e.*, $nd \geq | A |$), that means that the current feasible coalition structure cannot be improved anymore, and a_i informs the agents that the process is ending with S as the best feasible coalition structure found so far.

Figure 4.3 presents the process of the decision algorithm.

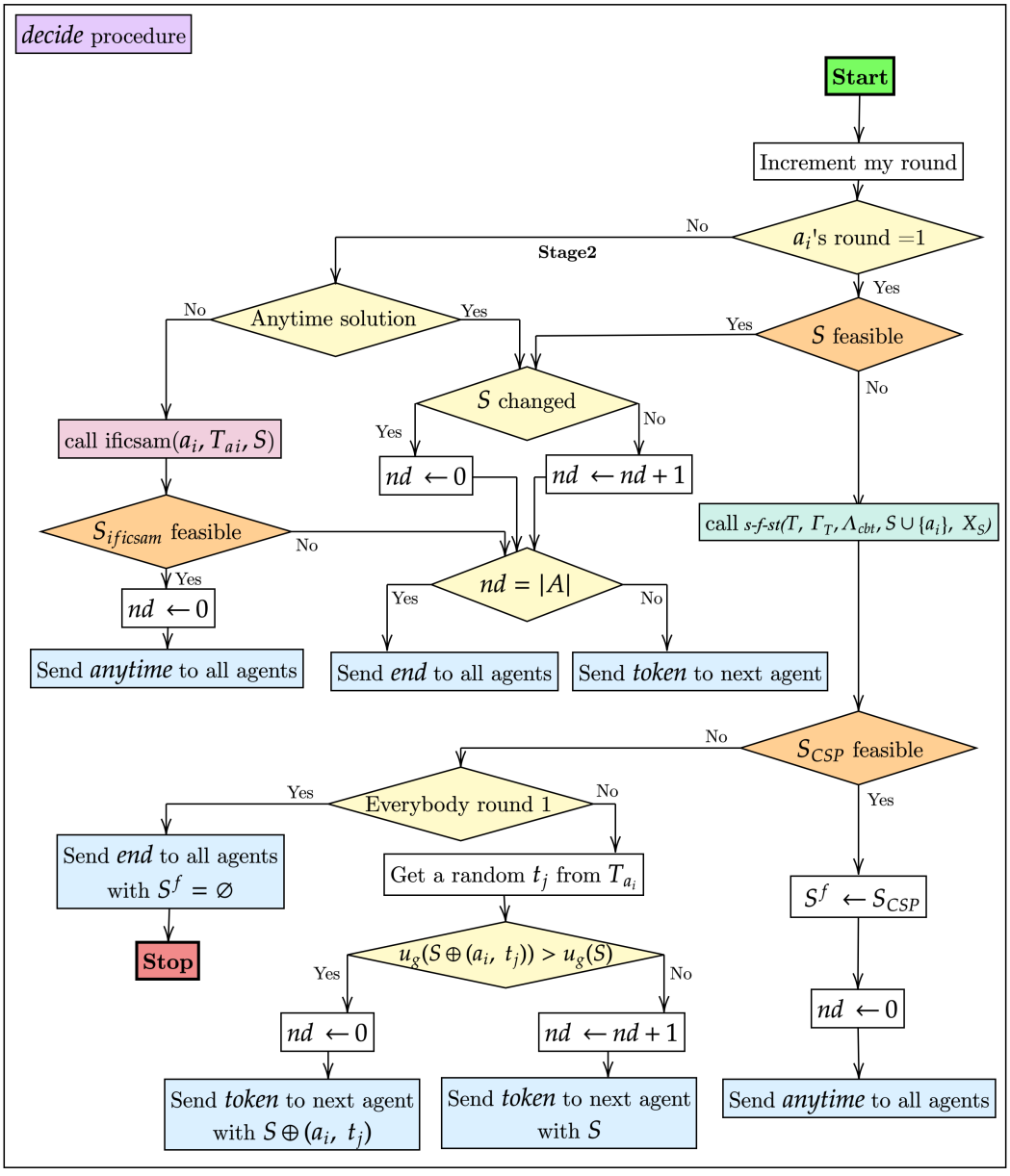


Figure 4.3: Decision process

4.3.3 Feasible Structure Generation Process

The *s-f-st* procedure is used in the first stage of our method. It is called exactly at line 6 of algorithm 2 with $s\text{-}f\text{-}st(T, \Gamma_T, \Lambda_{cbt}, \{a_i\} \cup S, a_i, X_{a_i} \cup X_{S \setminus \{a_i\}})$. This procedure allows the deciding agent to find a feasible coalition structure or to conclude the temporary non-existence of such an allocation. This agent uses the knowledge it has of agents characteristics X_S and of itself X_{a_i} to look for distribution of the tasks among agents, represented by a coalition structure, where the requirements in Γ_T and Λ_{cbt} are fulfilled. It is a sort of a locally centralised feasible coalition structure generator that only considers feasibility but not utility.

The goal is to find a feasible coalition structure where agents are assigned to tasks in T. A coalition structure feasibility results from fulfilling of the tasks and task combinations requirements by its coalitions. Many centralised coalition formation solutions can answer this need. For example, the methods of [O. Shehory et al. 1998] can be applied. We have chosen to use a Constraint Satisfaction Problem (CSP) to make use of the mirroring between tasks requirements and constraints.

A CSP is normally defined by three elements: a set of variables, a set of these variables' respective domains of values and a set of constraints. In our CSP model, variables represent agents' assignments and each variable's domain is the set of singletons of tasks in T and the empty set. This way, the decision of an agent, represented by its corresponding variable, can have as a value one of the tasks, if a task is allocated to him, or no task, otherwise. This allocation of tasks in the domains to the agents in the variables can be seen as a coalition formation where each task has a coalition composed of agents whose variables have this task as a value. Finally, the constraints of our CSP implement the single task requirements and the task combination requirements satisfaction with respect to the agents characteristics. The satisfaction of a constraint is equivalent to the fulfilment of its requirement. Formally, the output of the CSP should be a coalition formation S_{CSP} where:

$$\begin{cases} X_{S_{CSP}} \circlearrowleft \Gamma_T \\ X_{S_{CSP}} \circlearrowleft \Lambda_{cbt} \end{cases} \tag{4.1}$$

At each run, the *s-f-st* procedure starts by creating from its parameters the

variables, the domains, and the constraints. With these elements as inputs, it runs the CSP via a solver. If the solver returns a solution, it is reconstructed to a coalition structure and returned. If not, an empty coalition structure is returned.

In the following, we are presenting an example that illustrates the use of this procedure and the CSP modelisation.

▶ **Example 4.1.**
As in example 3.2 in the previous chapter but a lighter version, we consider a set of agents and a set of tasks.

Consider 4 agents, $A = \{a_1, a_2, a_3, a_4\}$ scattered on a square grid of 10m side length, that must perform two different tasks, $T = \{t_1, t_2\}$. The agents have 3 characteristics: their location x_1, energy x_2, and payload x_3 (either a normal camera C or a thermal one R). Agent characteristics values are presented in table 4.2.

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

Table 4.2: Example of agents' characteristics

Each task has 4 requirements. γ_1 states the position of the task, and the maximal allowed distance of an agent from that position. γ_2 states the minimum energy necessary to perform the task, and γ_3 states the set of the minimum required payloads for the task brought by all the agents. Finally, γ_4 states the minimum number of agents needed to accomplish the task. Requirements of $\{t_1, t_2\}$ are presented in table 4.3.

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

To illustrate the s-f-st procedure use, here are the outputs of the procedure if called with these specific parameters:

- *If agent a_1 calls the procedure with s-f-st($T, \Gamma_T, \Lambda_{cbt}, \{a_2\}, \{a_1\}, X_{a_1} \cup X_{a_2}$), the result will be an empty coalition structure $\{\{\}, \{\}\}$. This is because the agents a_1 and a_2 cannot fulfill all the requirements in Γ_T and Λ_{cbt}. Thus, there is no feasible coalition structure involving only those agents.*

γ_I	$\gamma_I(t_1)$	$\gamma_I(t_2)$
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 4.3: Example of single tasks' requirements

- *If agent a_4 calls the procedure with s-f-st($T, \Gamma_T, \Lambda_{cbt}, \{a_1, a_2, a_3\}, \{a_4\}, X_{a_4} \cup X_{\{a_1,a_2,a_3\}}$), the result will be a coalition structure $\{\{a_1\}, \{a_3, a_4\}\}$. Thus, there is a feasible coalition structure that is considered as an anytime solution.*

◀

4.3.4 Feasibility Check Process

To check if a coalition structure S is feasible or not, we also use a CSP. The modelisation of the CSP consists of a set of variables representing the agents assignments, their respective domains where each represents an empty set or the task that is allocated to the agent depending on the agent's assignment in the coalition structure S, and the constraints corresponding to local and global requirements.

If the coalition structure S is feasible, the solver returns it. This means that this assignment satisfies the problem's constraints meaning that all the requirements are fulfilled by the agents coalitions in S and thus that S is feasible. Otherwise, it returns that no solution exists, meaning that the coalition structure violates one or many constraints of the problem, making it unfeasible.

The CSP used to check the feasibility of a coalition structure bears similarities with the one used to search for a feasible coalition structure. The difference is at the variables domains. In the feasibility check CSP, we already have a coalition structure, and we just want to check if this coalition structure satisfies all the constraints of the problem. This is why we only give the agents assignments in the coalition structure in question for the variables domains. The CSP returns the same entries if the coalition structure turns out to be feasible, whereas it does not return anything if it is not. In the search for a feasible coalition structure with *s-f-st*, we use a CSP to construct a feasible coalition structure from scratch

thanks to the knowledge we have about tasks and precedent agents, ignoring the unfeasible current coalition structure (we know is not feasible thanks to the test in line 4 of algorithm 2).

4.3.5 Run-Through Example

Here is a continuation of example 4.4.

▶ **Example 4.2.**
For simplicity, we assume that the token goes from a_1 to a_2, then to a_3 and to a_4.

Agent a_1 starts by invoking algorithm 1. Following lines 1 and 2, S, S_{max}, X_S, and T_1 are initialized with the empty set, whereas $R[1] = 1$ and $nd = 0$. On lines 4 and 5, a_1 sets $T_1 = \{t_1, t_2\}$. a_1 gets one of these tasks, t_1, for example, and adds itself to coalition C_{t_1}, to obtain a first coalition structure $S = \{\{a_1\}\}$. This structure is not feasible (line 9), and a_1 passes the token to a_2 (line 15) while waiting to receive a message.

Agent a_2 receives the token and calls decide (line 28). In algorithm 2, a_2 keeps the current coalition structure in S_{old}, and updates its round to 1. Conditions in lines 2 and 3 are satisfied, the best coalition structure so far, S, is put in S_{max}, and a_2 searches for a feasible structure by calling s-f-st. As it only knows the characteristics of a_1 and a_2, it cannot find one. Since check-feasibility(S) is false (line 7), we jump to line 13. As not all agents got the token yet, a_2 looks for tasks it can contribute to, (lines 16-17), and obtains $T_2 = \{t_1\}$. It joins then coalition C_{t_1} in S, that becomes $S = \{\{a_1, a_2\}\}$, and adds its characteristics to X_S. We now have $u_{global}(S) = 0.55$ and $u_{global}(S_{max}) = 0.45$. As the utility of the newly founded S is greater than the one of S_{max}, S becomes the new best solution S_{max}, and nd is reinitialized to 0 (we start working on a new solution). Finally, a_2 passes the token to a_3 (line 23).

Agent a_3 receives the token from a_2, and calls the procedure decide in line 28 of algorithm 1. As a_2 just did, a_3 keeps the current coalition structure in S_{old}, and updates its round to 1. Conditions in lines 2 and 3 are satisfied, $S = \{\{a_1, a_2\}\}$ is put in S_{max}, and a_3 searches for a feasible structure by calling s-f-st. It cannot find one, and jumps to line 13. As a_4 has not had the token yet, a_3 looks for the tasks it can contribute to, in lines 16 and 17, and obtains $T_3 = \{t_1, t_2\}$. Assume that t_2 is selected (line 18), we obtain $S = \{\{a_1, a_2\}, \{a_3\}\}$, with $u_{global}(S) = 0.65$. Once

again, the condition in line 21 is satisfied, S is the new best solution, $nd = 0$, and a_3 passes the token to a_4 in line 23.

As previously, agent a_4 receives the token from a_3, calls decide, updates S_{old} and its round $R[4] = 1$, and looks for a feasible coalition structure (algorithm 2, line 5). As it knows all the information from all the agents, it finds one, for example $S = \{\{a_1\}, \{a_3, a_4\}\}$. As S is feasible (line 7), we go to line 8, S_{max} and S^f take the value of S, $nd = 0$, and a_4 informs a_1 and a_3 that a feasible coalition structure has been found, in case an anytime solution is required.

Here is the end of the first stage, and thus of algorithm FICSAM, with an anytime solution. ◀

4.4 Improved Feasible Interdependent Coalition Structure Anytime Method (IFICSAM)

As stated in the previous section, the algorithm's first stage consists in finding a feasible coalition structure to allow agents to have a decision that deals with the problem's constraints represented by the task requirements. This decision corresponds to an anytime solution. The second stage incrementally improves feasible coalition structures in terms of utility. It starts from the second round, once S^f was found in round 1.

In this stage, agents examine improvements in u_{global} through replacements or swaps between single or groups of agents. The resulting improved structure must preserve the feasibility of the improved solution through the respect of both single tasks and task combinations requirements.

As shown in algorithm 2, if the current round is greater than one, $R[i] > 1$, that means that a feasible coalition structure was already found (line 34), and the agents seek another feasible coalition structure that increases global utility u_{global} (*i.e.*, greater utility than the global utility of the coalition structure found in the first round).

To this end, agent a_i can use different algorithms to search for better solutions. It can try to swap its place with another agent or replace another agent allocated to a different task. It may also swap its place with a certain agent in another coalition. And after each swapping trial, it should check if the coalition structure

that results from this possible change has a greater utility than the old coalition structure. The next sections present in-depth that different versions exist of the improvement trials. After the call of one of these improvement algorithms (line 34), the agent checks the returned coalition structure's feasibility (line 35). If the algorithm returns an improved feasible coalition structure (line 36), then this coalition structure becomes the current best feasible one and it corresponds to a new anytime feasible coalition structure for all the agents (line 37). 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 considered anymore for further improvement. Hence, agent a_i reconsiders the feasible coalition structure that it received from the previous agent (line 40).

Next we present the different alternatives for this improvement implemented as algorithms 3, 4 and 6.

4.4.1 Swap Improvement Process

Algorithm 3 implements a method that allows agents to improve the global utility $u_{global}(S)$ of a feasible structure S with one-to-one swapping with other agents.

Once agent a_i receives the token, it assumes that the received structure S maximizes u_{global} (line 1). It then checks whether it can contribute to improving the global utility u_{global} by exploring two possibilities.

The first consists of assigning agent a_i to another task (and thus to another coalition). For this reason, agent a_i examines the possibility to move to a coalition of one of the tasks that it can contribute to. For each of these tasks, let say t_j, it checks if its move to coalition C_{t_j} can result in a coalition structure $S \oplus (a_i, t_j) \ominus (a_i, \tau(a_i, S))$ that has a greater utility than the coalition structure with the best utility until now S_{max} (line 3). This is built on the idea that there may be a task $t_j \in T_{a_i}$ that, if a_i contributes to its performance instead of its contribution to $\tau(a_i, S)$, its current task in S, u_{global} increases.

Switching its task is relevant if none of the requirements of t_j (*i.e.*, $\gamma_l \in \Gamma_{t_j}$) is violated and if u_{global} increases when a_i participates in the accomplishment of task t_j instead of task $\tau(a_i, S)$ (line 3). In that case, the new coalition structure becomes the best current one (line 4).

The second possibility consists of switching the agents. There is two possible

Algorithm 3: improve_ificsam_swap(a_i, T_{a_i}, S)

1 $S_{max} \leftarrow S$
2 **for** $t_j \in T_{a_i} \setminus \tau(a_i, S)$ **do**
3 **if** $\forall \gamma_l \in \Gamma$ such that $a_i \bowtie \gamma_l(t_j)$ 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** $a_k \in C_{t_j}$ such that $\forall \gamma_l \in \Gamma : a_k \bowtie \gamma_l(\tau(a_i, S))$ **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}

scenarios that are tested. First, the deciding agent can take the place of another agent that is already allocated in a coalition. That means that the replaced agent leaves the coalition structure. If this case happens, the deciding agent can swap its place in its coalition with an agent assigned to another coalition, and thus to another task. To do this, agent a_i examines the two scenarios for each task t_j. Indeed, it checks whether it can replace or swap with another agent a_k that currently fulfills the requirements of task t_j assigned to $C_{t_j} \in S$ by fulfilling the requirements of the task $\tau(a_i, S)$ that is currently fulfilled by a_i in the coalition assigned to the task $\tau(a_i, S)$ in S. If such a change increases the global utility u_{global} of the resulting coalition structure in comparison with the latest coalition structure found with the best utility S_{max}, the resulting coalition structure is kept as the best current coalition structure in terms of utility in S_{max}. Coalition structure S_{max} is updated to obtain a new coalition structure where $u_{global}(S_{max}) > u_{global}(S)$ (see lines 6-13).

The feasibility of this new coalition structure is then verified in algorithm 2 (line 36) to decide if it is considered as a new solution or if the feasibility is broken by the change.

Next is the follow-up of example 4.4 in the case where agents continue to the second stage using the one-to-one improvement in algorithm 3.

▶ **Example 4.3.**
At the end of the first stage, agents have found an anytime solution: $S = \{\{a_1\}, \{a_3, a_4\}\}$. *Now, they continue to the second stage. The last agent to receive the token* a_4 *sends the token to agent* a_1.

Agent a_1 *has the token first, and when receiving the message it goes to line 19 of algorithm 1. We assume that we do not need an anytime solution, and proceed to line 23. By receiving the performative "ats",* a_1 *knows that* S *is feasible (line 24), and calls decide. In line 1 of algorithm 2,* $R[1]$ *is incremented to 2, and* a_1 *jumps to line 33.* S^f *is initialized with the last feasible coalition structure found,* $\{\{a_1\}, \{a_3, a_4\}\}$ *here, and* a_1 *can call algorithm improve-ificsam (i.e., algorithm 3) to check whether a better solution (with respect to the global utility* u_{global}*) can be found.*

After recording the current structure in S_{max}, a_1 *browses the tasks it can contribute to* T_{a_1}, *except for* t_1. *The only remaining task is* t_2, *and* a_1 *checks on line 3 if it can improve the global utility of* S *by switching on* t_2. *The structure would then be* $\{\{\}, \{a_1, a_3, a_4\}\}$, *with a utility of* 0.55, *which does not improve the utility of* $S_{max} = 0.7$. *Next,* a_1 *looks if it can switch places with another agent. By switching places with* a_3 *(line 6),* S_{new} *is created as* $S_{new} = \{\{a_3\}, \{a_1, a_4\}\}$ *(line 7). As* $u_{global}(S_{new}) = 0.75$ *is greater than the utility of* S_{max} *(line 10), this coalition is returned (lines 11,12).*

Returning from the call of algorithm 3, a_1 *checks if* $S = \{\{a_3\}, \{a_1, a_4\}\}$ *is still feasible (algorithm 2, line 36). As it is, and as* S *has changed during the process, condition line 37 is verified.* a_1 *updates* S^f *to the new structure* S, *reinitializes nd to 0 (line 38), and informs* a_3 *and* a_4 *(lines 39 and 40).*

Agent a_3 *gets the token, goes to line 22 of algorithm 1, and calls decide on line 27. On line 1 of algorithm 2,* $R[3]$ *is incremented to 2, and* a_3 *jumps to line 33. As* S *is the best feasible coalition structure for this example, condition line 37 is always false, whatever algorithm is called on line 35.* S *thus retrieves its previous value*

(line 41). Since $nd = 0$ *and* $S = S_{old}$*, we go at line 43,* nd *is incremented to 1, and the token passes to* a_4*.*

All the agents receive this message (line 32 of algorithm 1 1), and the process ends with a feasible and improved solution $S = \{\{a_3\}, \{a_1, a_4\}\}$*.* ◀

4.4.2 Combinations Improvement Process

Instead of the one-to-one swap proposed in the previous algorithm, agents may try to swap with many agents in their efforts to improve the coalition structure utility.

To do so, algorithm 4 is called by algorithm 2 (line 34). Contrary to the previous algorithm where deciding agents consider only one-to-one swapping with other agents, this algorithm proposes to discover more widely the search space by considering many-to-one swaps. For this, using algorithm 4, an agent a_i checks whether, for each task $t_j \in T_{a_i}$, it is beneficial to replace each agent a_k member of the coalition C_{t_j} assigned to t_j, by a combination cb of agents that includes a_i (lines 4-7) whose characteristics X_{cb} match the requirements Γ_{t_j} of task t_j (line 4). These changes should take place if they increase the global utility u_{global} (line 10). All the agents' combinations that can contribute to task t_j are considered. The coalition structure, among the possible coalition structures via these changes that has the greater global utility, is retained in S_{max}.

Algorithm 4: improve_ificsam_combinations(a_i, T_{a_i}, S)

```
1 S_max ← S
2 S' ← S
3 for t_j ∈ T_{a_i} do
4     for cb ∈ 2^A such that cb ⋈ Γ_{t_j} and a_i ∈ cb do
5         for a_k ∈ C_{t_j} do  S' ← S ⊖ (a_k, t_j)
6         for a_l ∈ cb do  S' ← S ⊕ (a_l, t_j)
7         if u_global(S') > u_global(S) then  S_max ← S'
8 return S_max
```

Algorithm 4 with its many-to-one swapping trials allows discovering a wider part of the search space on the possible coalition structures. Nonetheless, due to its combinatorial complexity, it does not scale well to large problems. To deal

with problems of large dimensions (number of tasks and number of agents), a large number of agent combinations has to be checked for each task by each agent that possesses the token. This way, the cost of improving the solution's utility with algorithm 4 may be too high.

4.4.3 CSP Improvement Process

To deal with the highly combinatorial problem related to the many-to-one swaps in algorithm 4, we propose algorithm 6 using a CSPs approach. This algorithm suggests using single task requirements as hard constraints (*i.e.*, required to be satisfied) to reduce the search space. The idea is to eliminate the coalition structures in the search space that do not satisfy the problem's constraints and thus the tasks' requirements before processing them and calculate their utility. Algorithm 6 is called by algorithm 2 (line 34) and is explained in details later in this section.

The goal of algorithm 6 is to optimise the combinatorial part of the many-to-one swap improvement algorithm in case of scalability. In algorithm 4, each agent verifies if it can with any subset of agents replace an agent to perform a task and thus increase the global utility. Agents do not benefit from their knowledge of the tasks requirements and the global utility in this search. This is why they perform the tests on global utility improvement even for subsets that do not fulfill the requirements of the task in question. Hence, the latest algorithm (algorithm 4) consumes useless computations for subsets that cannot be a part of a feasible coalition structure since they do not satisfy the requirements of the task they are candidates for. Instead of checking all the agents' subsets, the variant of algorithm 6 uses CSPs to lighten the space of possibilities of these subsets by selecting the locally optimal one. In addition to the single task requirements used as hard constraints for the CSP, we can design soft constraints (or preferences) that can help orient the research towards subsets allowing higher chances to increase the global utility function. Implementing such concepts in our algorithm 6 is equivalent to doing for each task the selection of an optimal subset of agents regarding the application's requirements and preferences.

Selecting the optimal agents' subset according to the requirements of the tasks and the preferences dictated by the utility function in the application amounts to computing optimal subsets of a set of items. For that, [Binshtok et al. 2007]

proposed the Branch and Bound over Constraint Satisfaction Problems (BB-CSP) algorithm that allows finding the optimal items' subset regarding a preference specification. Given that this work exactly answers our demand, we chose to employ and adapt its algorithm to our context and explain it as follows.

A preference specification is a description of the problem's elements, aka the items, the item subsets, and a preference order over the properties of these subsets. BB-CSP introduces a formalism for the preference specification. It starts with the item properties designation based on which the set properties $(P_k)_{k \leq n}$ are defined. Once the set properties are defined, set preferences are built over the comparison of the values of these set properties through conditional value preference statements or relative importance statements. The conditional value preference statement is when for specific values of the properties $P_{i_1}, P_{i_2}, ..., P_{i_j}$, we prefer a value p_k for the property P_k over a value p'_k. The relative importance statements are when for specific values of the properties $P_{i_1}, P_{i_2}, ..., P_{i_j}$, the property P_k is more important than the property P_l and thus we prefer a better value for P_k even if we compromise on P_l's value. We can set preferences defining an order over the properties and their values based on these comparison statements. This order generates a tree with different properties combinations. Each node in this tree represents a combination of properties and is associated with a set of candidate subsets. The leaf nodes represent combinations composed of all the properties and assign a value for each property in $(P_k)_{k \leq n}$.

For example, let us consider a preference specification where there is only two set properties P_1 and P_2, that are both boolean (*i.e.*, they either take the value true or false and we note P_i when it is true and $\overline{P_i}$ otherwise) and where P_1 is more important than P_2 and true properties are the preferred ones. The resulting tree is as presented in figure 4.4.

In searching for an optimal set, [Binshtok et al. 2007] proposed a Branch & Bound search. This is used to prune the nodes which property combinations are sub-optimal. Then, a CSP is run in each tree node and in the specified order by the preference tree. In this CSP, the variables representing items take 1, if the item in question appears in the output subset or 0 otherwise. The CSP aims to look for a subset of items that has associated preferred properties to its node. Since each tree node is mapped to a CSP, the entire tree is viewed as a tree of CSPs. The B&B on the tree-of-CSPs enables finding an optimal subset of items denoted W_{opt} regarding the preference specification. That is, a set $W_{opt} \subseteq W$

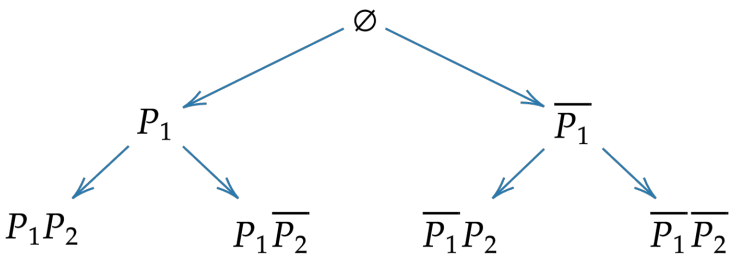


Figure 4.4: Example of a tree of CSPs in [Binshtok et al. 2007]

such that for any other set $W' \subseteq W$, we have that the properties that the subset W_{opt} satisfies are no less desirable than the properties W' satisfies.

We found the model and algorithm relevant to our problem of selecting the best subset of agents for the many-to-one swap and applicable after some adaptation. Analogically, the items in our problem are the agents, the item properties are the agent characteristics, and the set properties are the requirements' state of fulfillment by the set of agents in the set. We can, in addition, define some new desired requirements that can lead to utility increase. For example, if a given task requires at least two agents, but the more agents we have, the greater the utility function, we can define the desired requirement of having four agents.

In [Binshtok et al. 2007], only preferences are considered. Hence, the set properties preferred values can be seen as soft constraints, and the goal is to satisfy the maximum preferred ones. In our problem, however, for a given task $t_j \in T$, t_j' requirements are hard constraints that we denote H_{t_j}. Besides, we add some soft constraints, denoted as P_{t_j} for each task $t_j \in T$, in a specific decreasing order of importance (*i.e.* from the more important to the less important), helping orient the search towards the subset of agents that, by doing the concerned task, increases the global utility. Hence, the main adaptation concerns the addition of the hard constraints in the preference specification and thus in the CSPs's tree. Instead of having an empty root for the tree of CSPs, we are starting with the set of true properties representing the hard constraints we want our subsets of agents to satisfy. This guarantees the assigned coalition's local feasibility. In figure 4.5, an example of the tree of CSPs over which we apply our adaptation of the algorithm in [Binshtok et al. 2007], where a task has three task requirements

($| \Gamma_{t_j} |= 3$), and thus three hard constraints and two soft constraints were added based on the utility function.

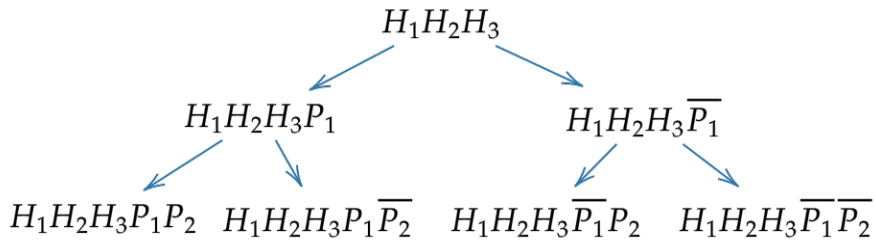


Figure 4.5: Example of the application tree of algorithm 5

▶ **Example 4.4.**
We illustrate here how the preference specification can be set for the CSP improvement algorithm (algorithm 6).

Preference specification

What follows is an illustration of the preference specification that can be used in our adaptation of the method of [Binshtok et al. 2007]. In this method, preference specification of subsets is based on items properties.

For a given task t_j:

Item properties *The items' properties are equivalent to agents' characteristics in our modelisation (see chapter 3). Let's consider that each agent a_i has four characteristics:*

- $x_1(a_i)$ *is the type of the agent a_i where the agents types are $D_1 = \{\delta_1, \delta_2\}$,*
- $x_2(a_i)$ *is the distance to the task of the agent a_i,*
- $x_3(a_i)$ *is the remaining autonomy of the agent a_i,*
- $x_4(a_i)$ *is a boolean variable to verify if agent a_i has the needed resource or not.*

Set properties *Let's describe each subset s of agents in A by a set of boolean properties that we denote here $(\rho_j)_{j \leq n}$:*

- *The subset of agents contains the minimum necessary number of agents of each type for the task t_j*

  $\rho_1 : \langle |\{a_i \in s \mid x_1(a_i) = \delta_1\}| \geq 1 \text{ and } |\{a_i \in s \mid x_1(a_i) = \delta_2\}| \geq 1 \rangle$

- *All the subset's agents are at a maximal distance allowed for this task t_j which is equal to 3*

  $\rho_2 : \langle \max_{a_i \in s} x_2(a_i) \leq 3 \rangle$

- *All the subset's agents have enough autonomy regarding the minimal autonomy that is required for task t_j which is 5 (the agent with the less remaining autonomy has more autonomy than the minimal threshold)*

  $\rho_3 : \langle \min_{a_i \in s} x_3(a_i) \geq 5 \rangle$

- *The number of agents for each type necessary for task t_j that does not exceed the maximum allowed number, namely 3 agents of type δ_1 and 5 agents of type δ_2*

  $\rho_4 : \langle |\{a_i \in s \mid x_1(a_i) = \delta_1\}| \leq 3 \text{ and } |\{a_i \in s \mid x_1(a_i) = \delta_2\}| \leq 1 \rangle$

- *The subset of agents have more than the required resources for task t_j that is 1*

  $\rho_5 : \langle |\{a_i \in s \mid x_4(a_i) = \top\}| > 1 \rangle$

The three first properties are equivalent to the tasks requirements that define the minimal number of agents of each type for the task, the maximal distance that the agents should not exceed and the minimal autonomy that the agents need to perform the task.

We consider the two last properties as soft constraints and that the utility function, for that application, optimises the number of participating agents. The utility function gives greater utility to tasks assigned with less than a threshold number of agents of each type. It also tends to valuate more when extra resources are present to be used if the required ones are wasted. Nevertheless, it is more valuable not to use too many agents than to have supplementary resources.

As an example, we consider a set of agents $A = \{a_1, a_2, a_3, a_4, a_5\}$ *where:*

a_i	$x_1(a_i)$	$x_2(a_i)$	$x_3(a_i)$	$x_4(a_i)$
a_1	δ_1	2	5	$\bot$
a_2	δ_1	3	6	$\top$
a_3	δ_1	2	4	$\top$
a_4	δ_1	4	7	$\top$
a_5	δ_2	2	5	$\top$
a_6	δ_2	2	8	$\bot$

Table 4.4: Example data to illustrate the preference specification

We denote $\alpha_i = 1$ *if agent* a_i *is in the solution, the subset at the output, and* $\alpha_i = 0$ *otherwise. As mentioned earlier, we denote the hard constraints by* H_j *and the soft ones by* P_j*. Next is how the set properties are translated to constraints that are fed to the CSP. For the example in table 4.4, we list the constraints used in the CSPs to generate the best subset of agents in A to do task* t_j *regarding this preference specification:*

- $H_1 : \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \geq 1$ *and* $\alpha_5 + \alpha_6 \geq 1$
- $H_2 : \alpha_3 = 0$
- $H_3 : \alpha_4 = 0$
- $P_1 : \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \leq 3$ *and* $\alpha_5 + \alpha_6 \leq 1$
- $P_2 : \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 \geq 1$

Based on these constraints, their categorization among hard and soft constraints, and the order of soft constraints (here P1 preferred to P2), a tree of CSPs can be generated as in figure 4.5 and we can run algorithm 5 on it. ◀

Algorithm 5 presents the adaptation of the BB-CSP algorithm. It allows agent a_i to compute the best local combination of agents concerning the requirements

of task t_j and the utility dictated desirability via the resolution of a tree of CSP problems.

For this CSP variant, when the agent a_i calls $improve\_ificsam$ (line 34 of algorithm 2), algorithm 5 is called for each task t_j in T_{a_i} by the improvement algorithm we present later. First, let us describe the best subset selecting algorithm used to find coalitions improving the coalition structure, provided in algorithm 5.

Algorithm 5: best_csp$(a_i, t_j, H_{t_j}, P_{t_j}, A, X_A)$

```
1  Q ← {H_{t_j}}
2  best ← ⟨0, ..., 0⟩
3  while Q ≠ ∅ do
4      N ← Pop(Q)
5      (solution, upper) ← CSP(N)
6      if solution ≠ False and upper > value(best) then
7          if value(solution) > value(best) then
8              best ← solution
9          if P_{t_j}(t_j) ≠ ∅ then
10             p ← Pop_first(P_{t_j})
11             Q ← Q ∪ {N ∪ {p̄}, N ∪ {p}}
12 C_best ← {a_i ∈ A | best_i = 1}
13 return C_best
```

Algorithm 5 works as follows: agent a_i first initiates a queue Q with the hard constraints H_{t_j} (line 1). For a given application and a task t_j, the hard constraints, since they represent the task's requirements, are formulated based on Γ_{t_j}. The algorithm then iterates on possible sets of constraints (initially only the set of hard constraints). The agent a_i then keeps popping the constraints of the set in the order of importance. Each time, the agent puts the popped constraint in the set of active constraints N (lines 3–5). Then, the agent a_i proceeds by solving the CSP with the active constraints (line 5), where the variables are boolean (with 1 and 0 as values) for each agent representing its presence or not in the combination, the constraints are the currently selected set of constraints, and the objective function $value$ is a function aggregating the soft constraints. If a

solution is found with an upper bound better than the current best solution, the solution is kept and considered as candidate for the locally optimal subset (line 6). Then, a_i computes the solution's value and compares it to the current best value (lines 7). If the new solution is better, a_i sets the best solution as the newly found solution and saves it in *best* as the new best agents combination (line 8). If there still are soft constraints, agent a_i removes the first soft constraint from the set (to respect the preference order) and adds it at the end of the queue of all constraints (lines 9–11). The lines 12–13 are only used to retrieve the best coalition structure and return it.

After having described the locally optimal subset of agents process, we present the algorithm of CSP improvement in this variant. Algorithm 6 is the one called when the agent a_i calls $improve\_ificsam$ (line 34 of algorithm 2).

Algorithm 6: improve_ificsam_bestcombinations(a_i, T_{a_i}, S)

```
1  S_max ← S
2  for t_j ∈ T_{a_i} do
3      S' ← S
4      H ← H_{t_j}
5      C'_{t_j} ← best_csp(a_i, t_j, H_{t_j}, P_{t_j}, A, X_A)
6      while S'_{t_j} ≠ ∅ and u_global(S') ≤ u_global(S) (or after a number of tries)
        do
7          H ← H ∪ {C_{t_j} ≠ C'_{t_j}}
8          C'_{t_j} ← best_csp(a_i, t_j, H, P_{t_j}, A, X_A)
9      if u_global(S') > u_global(S) then
10         S_max ← S'
11 return S_max
```

In algorithm 6, we duplicate coalition structure S in S' (line 3), where we replace the coalition C'_{t_j} with the best subset returned by algorithm 5 (line 5). We then repeat the process of the optimal subset selection until we find a coalition with which the resulting coalition structure increases the global utility or after a certain number of tries that we fix depending on our application's computational

capacity or time-sensitivity level (line 6). We remind here that even in time-sensitive contexts, when we are in the improvement process (*i.e.*, in the second stage of the algorithm), the agents already have a feasible coalition structure that indicates what each agent should do and with which all the tasks are feasible. After that, agent a_i must check if, with the final obtained coalition for task t_j, the global utility u_{global} is improved (lines 9 - 10 in algorithm 6). If it is the case, it updates the coalition structure S_{max} with the new coalition structure S', where the coalition assigned to task t_j is replaced by the resulting coalition C'_{t_j} (the same way it is done in algorithm 4). The coalition structure that algorithm 6 returns passes through a feasibility check test in algorithm 2 (line 25) to decide whether to keep it as the current best feasible coalition structure or ignore it.

4.5 Complexity Analysis

We discuss the complexity of Algorithm 2 and Algorithm 1. These algorithms may rely on others, in which case we may discuss the complexity of those algorithms too. Algorithm 2 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 [Dechter 2003] whose complexity is NP-complete (one can show reduction from the 3-SAT problem). Hence, Algorithm 2 is NP-complete as well. This may seem prohibitive but CSP solvers like the one we use (Chuffed [Chu et al. 2018]) allow to deal with high complexity problems efficiently. Algorithm 1 also seems linear in $|T|$ and $|A|$. However, it calls Algorithm 2. Hence, it is NP-complete too, but solvable in practice for the problem sizes addressed in this work.

4.6 Empirical Evaluation

We illustrate the added value of our approach and evaluate its performance by benchmarking on a sample application. We also compare with performances obtained using a centralised method.

4.6.1 The Scenarios Generator

We evaluate our approach with a set of scenarios generated automatically. These are used to benchmark *FICSAM*, *IFICSAM* (the one-to-one and many-to-one variants) and a centralised solution.

The scenarios are generated according to the following settings: a fleet of Unmanned Aerial Vehicles (UAVs) is assigned a mission in a seaport. The (UAV) agents must inspect the hulls of boats in the port. The UAVs are relying 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 centralised approach, we experimented with up to 100 agents and 20 tasks. The USVs are scattered across the port so that the UAVs can easily charge for handling new tasks. Our scenario generator implements this by randomly positioning USVs (with a uniform distribution) on the port grid. We assume that the drones are initially uniformly positioned, but different distributions can be plausible depending on the application.

Inspection tasks may require various sensors. Here, we rely on two sensor types: High Definition (HD) cameras and LASERs (see for example, [Agnisarman et al. 2019]). The quantity of each resource required by a task depends on the 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 $[G/2, G]$.

The agents are provided with sensors such that 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 holding the token sends it to the nearest agent that has not 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. Specifically, the utility function we used is global and includes the agents' different properties regarding their allocated tasks. It sums the average of each of the number of agents that respect the maximal distance defined by their tasks by a certain margin, the number of allocated agents that have at least a resource, the number of agents, the number of tasks with a supplementary agent in their coalitions and the number of tasks with coalitions containing agents with a lifespan that exceeds a certain threshold.

The generator produces both feasible and infeasible scenarios. The latter is of interest for method comparison, as it requires that the algorithms prove unsatisfiability, which might take a long time. It is to be reminded that a centralized approach does not apply to the problem settings of these scenarios, mainly to avoid the single point of failure problem and as a result of the distributed nature of the drones and the possibility of communication failures.

4.6.2 Implementation

The previous scenario generator was built over the class diagram represented in figure 4.6. The base classes represent the core of our algorithms, and the inheriting classes represent their instantiations to our experimental use case.

4.6.3 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. That is why we consider that the number of agents is always larger than the number of tasks, hence $|A| > |T|$. The reported results include algorithms' execution time taken by the algorithms to terminate, utilities of the structures they return and the number of exchanged messages.

For each $\{|A|, |T|\}$ pair, we executed 200 runs. In each run, the agent characteristics and task requirement attributes are randomly generated.

The execution platform was a 3.70 GHz Intel(R) Core(TM) i9-10900X CPU running Python, the MiniZinc tool chain and the Chuffed solver.

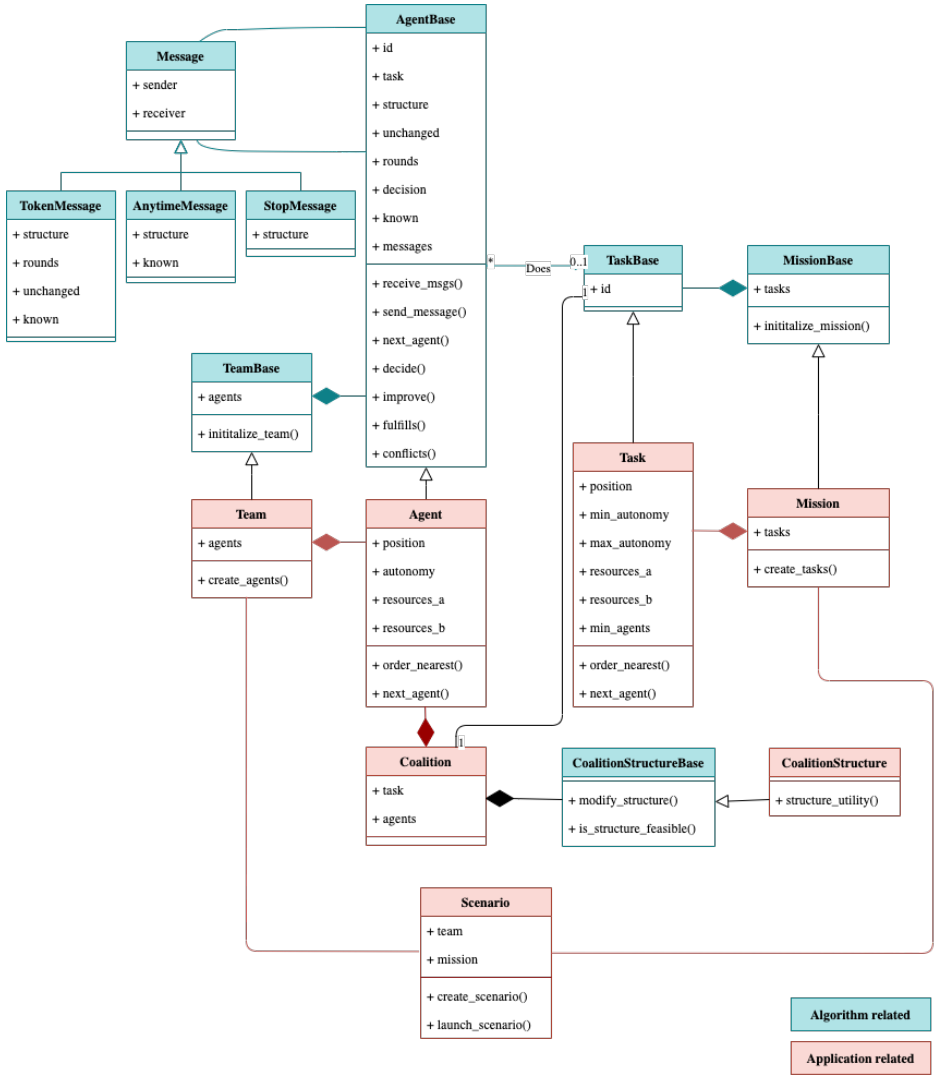


Figure 4.6: Class diagram of our experiments' implementation

4.6.4 Centralised Solution

To compare the results and performances of our algorithms, we developed a decentralised method. Our decentralised approach allows agents to make local

decisions and avoid mission crashes because of a single point of failure. However, the comparison to a centralised solution facilitates evaluating our solution's distance from optimum and execution time.

For the centralised method, we modeled our allocation problem as a Constraint Optimisation Problem and solved it with the Lazy Clause Generation based constraint solver Chuffed [Chu et al. 2018] through the constraint modeling language MiniZinc [Nethercote et al. 2007]. The following figures show the model expressed in the MiniZinc language.

```
int: nb_agents;
int: nb_tasks;
int: margin_autonomy_min;
int: margin_distance_max;
int: margin_nb_agents;

set of int: agents = 1..nb_agents;
set of int: tasks = 1..nb_tasks;
set of int: tasks0 = 0..nb_tasks; % 0 means no task

array[tasks] of int : tasks_autonomy_min;
array[tasks] of int : tasks_autonomy_one;
array[tasks] of int : tasks_distance_max;
array[tasks] of int : tasks_resA;
array[tasks] of int : tasks_resB;
array[tasks] of int : tasks_agents_min;
array[tasks] of tasks : tasks_nearest;
array[tasks] of tasks : tasks_nearest_sd;

array[agents] of int : agents_autonomy;
array[agents] of bool : agents_resA;
array[agents] of bool : agents_resB;

array[tasks, agents] of int: distances;
```

Figure 4.7: Parameters of the centralised method for the experimentations in the MiniZinc language

Figure 4.7 represents the list of parameters that are given as input data to the solver. It contains data of the agents characteristics, the tasks requirements and the thresholds used in the utility function.

In figure 4.8, the assignment array represents the problem's decision variables, which are for each mono-task agent the index of the potential task it can be assigned to and where zero means no assignment. The constraints are then listed to represent the tasks requirements. Finally, the objective function to be maximized is contained in the decision variable *obj*.

```
% C1: Mono-task agents array
array[agents] of var tasks0: assignment;

var int: obj;

% C2: Maximal distance
constraint forall (j in tasks, i in agents)
(assignment[i] == j -> distances[j, i] <= tasks_distance_max[j]);

% C3: Resources A for single tasks
constraint forall (j in tasks)
(sum(i in agents where agents_resA[i])(assignment[i]==j) >= tasks_resA[j]);

% C4: Resources B for single tasks
constraint forall (j in tasks)
  (sum(i in agents where agents_resB[i])(assignment[i]==j) >=
     tasks_resB[j]);

% C5: Resources A for couples of nearest tasks
constraint forall (j in tasks where tasks_resA[j] > 0) (
  (nb_tasks < 5 /\ exists(k in tasks where tasks_nearest[j] == k)
      (sum(i in agents where agents_resA[i]) (assignment[i] == j) +
          sum(i in agents where agents_resA[i]) (assignment[i] == k) >=
                 tasks_resA[j] + tasks_resA[k] + 1))
  \/ (nb_tasks >= 5 /\ exists(k, l in tasks where tasks_nearest[j] == k
                    /\ tasks_nearest_sd[j] == l)
      (sum(i in agents where agents_resA[i]) (assignment[i] == j) +
        sum(i in agents where agents_resA[i]) (assignment[i] == k) +
        sum(i in agents where agents_resA[i]) (assignment[i] == l) >=
            tasks_resA[j] + tasks_resA[k] + tasks_resA[l] + 1))
);

% C6: Minimal number of agents
constraint forall (j in tasks)
  (sum(i in agents)(assignment[i] == j) >= tasks_agents_min[j]);

obj =
(sum(i in agents)(bool2int(exists(j in tasks)(assignment[i] == j /\
 agents_autonomy[i] >= tasks_autonomy_min[j] + margin_autonomy_min))) +
 sum(i in agents)(bool2int(exists(j in tasks)(assignment[i] == j /\
   distances[j, i] <= tasks_distance_max[j] - margin_distance_max))) +
   sum(i in agents where agents_resA[i])
         (bool2int(assignment[i] != 0)) +
   sum(i in agents where agents_resB[i])
         (bool2int(assignment[i] != 0)) +
   sum(i in agents)(bool2int(assignment[i] != 0))) * nb_tasks+
      (sum(j in tasks)(bool2int(sum(i in agents)
          (bool2int(assignment[i] == j /\
           agents_autonomy[i] >= tasks_autonomy_one[j]))
              >= tasks_agents_min[j])) +
       sum(j in tasks)(bool2int(sum(i in agents)
          (bool2int(assignment[i] == j))
    >= tasks_agents_min[j] + margin_nb_agents))) * nb_agents;

solve maximze obj;
```

Figure 4.8: Variables, constraints and objective of the centralised method for the experimentations in the MiniZinc language

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, to prevent pathological cases, we filter the instances with series of necessary and sufficient conditions (for example, if the number of available agents is less than the sum of the required number of agents in all the tasks, it is useless to run the COP). Hence, no need to call the solver in such cases, which might take a long time to prove unsatisfiability.

4.6.5 Results Evaluation

Number of Agents	Number of Tasks	Average Utilities			Average Execution Time (in s)		
		FICSAM	IFICSAM	COP	FICSAM	IFICSAM	COP
5	2	0.58	0.73	0.83	0.9 (± 0.0)	1.4 (± 0.0)	0.2 (± 0.0)
10	2	0.59	0.79	0.85	1.6 (± 0.0)	3.0 (± 0.1)	0.2 (± 0.0)
10	5	0.55	0.70	0.78	1.7 (± 0.0)	2.9 (± 0.1)	48.0 (± 5.3)
20	2	0.53	0.73	0.85	3.6 (± 0.01)	6.8 (± 0.1)	0.2 (± 0.0)
20	5	0.53	0.75	0.86	3.4 (± 0.1)	6.1 (± 0.1)	0.2 (± 0.0)
20	10	0.53	0.70	0.79	42.0 (± 6.9)	43.9 (± 7.0)	1184.4 (± 9.0)
50	2	0.51	0.67	0.85	7.0 (± 0.2)	11.6 (± 0.3)	4.6 (± 6.0)
50	5	0.49	0.67	0.86	69.1 (± 11.1)	76.4 (± 11.3)	0.3 (± 0.0)
50	10	0.54	0.71	0.86	387.6 (± 25.4)	398.4 (± 25.4)	25.1 (± 7.2)
50	20	0.44	0.61	0.80	212.6 (± 25.1)	222.2 (± 25.1)	1195.7 (± 4.5)
100	2	0.48	0.62	0.85	70.2 (± 14.2)	83.4 (± 14.3)	1.8 (± 0.4)
100	5	0.48	0.61	0.86	189.7 (± 24.1)	209.3 (± 24.0)	1.2 (± 7.3)
100	10	0.48	0.63	0.86	446.5 (± 24.7)	468.9 (± 24.4)	32.0 (± 7.3)
100	20	0.53	0.65	0.83	1043.9 (± 45.8)	1073.8 (± 45.8)	1158.6 (± 12.6)

Table 4.5: Experimental results on average time and utilities for FICSAM and IFICSAM in its swap (1-to-1) variant compared with the centralised method

We present in tables 4.5 and 4.6 the results of the application of our algorithms FICSAM, IFICSAM with the one-to-one swapping variant (algorithm 3), and the

aforementioned centralised approach for each metric. Every single result in the table is an average of experiments with 200 randomly generated scenarios by the scenario generator. The three methods were all tested on the same scenarios. The remainder of this section presents the results in terms of global utility value for the system, runtime, and the number of messages exchanged for the decentralised version (this metric has no meaning for the centralised approach).

Not surprisingly, the utilities of the solutions generated by the decentralised approaches are below those of the centralised 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 centralised approach. IFICSAM utilities are always above 70% of the utilities of the centralised approach utilities, and are at 75% from optimum on average. IFICSAM's search of a better solution through inversion shows that even without reaching optimality, this policy allows finding significantly better solutions in a decentralised manner. We observed that, performance slightly degrades for a large number of agents (50 or 100). We believe that this may result from difficulties in finding an initial solution (as discussed below).

Notice that our algorithms terminate quickly. For the largest instances with 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 centralised algorithm on almost all the tests we performed. Further, the latter, given a 1200 seconds timeout, sometimes terminates without reaching an optimal solution. That is the case where the problem is the most combinatorial, typically when the ratio of number of available agents per task is the tightest. Otherwise, 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 a larger numbers of tasks, a smaller average number of agents is needed for each task. Therefore, the problem becomes simpler in some cases as its combinatorial complexity is lower, which favors decentralised algorithms. For instance, for 50 agents, scenarios with 20 tasks take less time than with 10 tasks. Eventually, there are scenarios where the centralized solution is clearly out of the question with respect to the execution time. Consequently,

not only our approach comes close to the optimal in terms of the solution's utility but it can still provide solutions in realistic times when the centralized method fails to.

Additional tasks, keeping the number of agents fixed, increase the difficulty for agents to find the first solution and send an anytime message to other agents. This can explain the drop in the number of exchanged messages when there are 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 centralised COP solution, imposing additional computation. This can explain why the centralised algorithm for 50

Number of Agents	Number of Tasks	Average Number of Messages	
		FICSAM	IFICSAM
5	2	18.4	25.9
10	2	36.8	71.7
10	5	38.0	61.8
20	2	71.5	177.2
20	5	71.8	170.7
20	10	75.6	158.2
50	2	178.3	557.6
50	5	176.0	576.3
50	10	181.7	526.8
50	20	180.3	511.7
100	2	351.3	1349.9
100	5	349.7	1204.1
100	10	350.6	1419.5
100	20	362.2	1147.1

Table 4.6: Experimental results on the average number of messages for FICSAM and IFICSAM in its swap (1-to-1) variant compared with the centralised method

and 100 agents, requires more time to solve scenarios with two tasks compared to scenarios with five.

As mentioned above, some of the generated scenarios are infeasible because 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 mentioned failure. In fact, the token is passed via a message to all agents, from each to the next one. Then the last one that receives the token without succeeding finding a solution sends to its previous end message and so on until the first agent is reached.

Moreover, FICSAM and IFICSAM take significantly less time than the centralised algorithm to terminate when there is a large number of agents and tasks. For 20 agents and 10 tasks, for example, they terminate after 70 seconds on average, while the centralised algorithm takes 400 seconds. For 50 agents and 10 tasks, they terminate after 360 seconds while the centralised, 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 decentralised approaches presented in this paper.

4.6.6 Results for the many-to-one swapping improvements

We also wanted putting in evidence the added value of algorithm 6. We recall that this algorithm uses CSP techniques for each task to find the better coalition locally. In this way, the decision-maker agent performs a many-to-one swapping of this coalition with the one that is already assigned to that task, aiming to find another coalition structure with a greater utility. We used the same generator of experiments, and we replaced the LASER sensors with arms. The agents with the arms can lift objects, and two armed agents are needed to lift each object. This is why we added a constraint requiring a couple of armed agents for each task since one-armed agent cannot perform the lifting alone.

In addition to the application's hard constraints, we also used an ordered list of preferences (*i.e.*, soft constraints) in algorithm 6. With the first preference we considered only agents that are not assigned to the current coalition structure in order to minimize the chances of breaking the feasibility of the resulting coalition structure after the swapping. With the second preference we considered only

agents whose positions are closer to the task's position by also considering a supplementary margin. With the third preference we considered only agents with greater autonomy than the minimum required for this task by also considering an additional margin. These preferences helped us orienting the research towards feasible coalition structures with better utilities.

For each $\{|A|, |T|\}$ couple, we executed 1000 runs.

Number of Agents	Number of Tasks	Average Utilities				Cases where IFICSAM (many to 1) solutions are optimal
		FICSAM	IFICSAM		COP	
			1 to 1	many to 1		
5	2	0.46	0.48	0.85	0.88	60%
10	2	0.44	0.53	0.64	0.89	8%
10	5	0.52	0.53	0.52	0.86	6 %

Table 4.7: Experimental results on average time and utilities for FICSAM and IFICSAM in its two variants swap and bestcombinations (1-to-1 and many-to-many) compared with the centralised method

In these experiments (with the way we generate our scenarios), we planned to run the algorithm 6 a limited number of times. We fixed the number of iterations at five by looking for a local better coalition that allows increasing the coalition structure utility (line 6 of algorithm 6).

For small instances of our scenarios (*i.e.*, where the number of agents does not exceed 10), the use of algorithm 6 allows (with the defined hard and soft constraints) finding the locally best subset that increases the global utility in the five first iterations. However, for instances with more agents, the scenarios generator must be updated for generating scenarios with more constraints on combinations involving several agents, which are recurrent in real-world applications. Moreover, the number of iterations has to be configured (*i.e.*, increased) with respect to the application time limits and the problem's size for approaching the optimal solution. Finally, more soft constraints can be added to help algorithm 6 become more efficient.

Nevertheless, the results on small instances show that, on average, algorithm 6 outperforms algorithm 3 in many instances just by adding one single task requiring specific combinations (here a couple) of agent characteristics (here armed agents). In this case, the first variant of our IFICSAM algorithm (*i.e.*, the one-to-one swapping algorithm 3) fails to obtain a coalition structure with a greater utility. The reason is that this algorithm can only add or exchange an agent with one agent each time and cannot add couples of armed agents to the coalitions.

Consequently, algorithm 6 may improve global utility. Our experiments focused on small instances, but our objective for future work is to create an optimized scenario generator to demonstrate its efficiency also for larger instances.

4.7 Discussion

Algorithms FICSAM and IFICSAM are based on a novel approach with token-passing decentralised coalition formation algorithms for multi-agent systems. They enable the cooperative agents to make decisions related to a task allocation problem when tasks are interdependent. A communication protocol underlines the decision-making process for the inter-agents exchange of messages. Hence, the algorithm Feasible Interdependent Coalition Structure Anytime Method (FICSAM) allows finding collectively an anytime solution, a feasible coalition structure, first. When agents require an anytime solution, they can stop here and start their execution. In a second time, with the algorithm IFICSAM in addition, the agents can improve their solution's quality to increase their collective performance expressed by the utility function. The first solution found guarantees that the tasks can be performed by agent coalitions each task is allocated to. This first coalition structure is found in a very reasonable time, even with large instances. The first solutions, especially the solutions after the one-to-one swapping improvement, have very good utilities compared to the centralised solutions that produce optimal utilities. Besides, in the beginning, the agents start making decisions with partial information along with gathering information about other agents' characteristics. At the end of the first round, all the agents have information about all the other agents' characteristics. This is what makes it safe to decide that no solution exists covering the tasks requirements at the end of the first round at the latest.

The algorithms presented in this chapter are entirely decentralised, anytime, solve the interdependent task allocation problem, built on a communication protocol with a reasonable number of short messages to be exchanged, can scale to large instances, and produce coalition structures that are feasible with very good utilities.

The improvements we proposed in this chapter and more extensions can be tested and developed. For example, the impact of the token-passing order on results if multiple orders are possible can be interesting to study. In our approach, and since the information and the solution are constructed, the decision is sequential and only one agent at a time has the decision token. A study of the possibility of paralleling the decision in specific cases can be examined.

5 Conclusions & Outlook

Contents

5.1 Introduction 108
5.2 Thesis Results 108
5.3 Limitations and Recommendations for Future Work . 110

5.1 Introduction

This chapter concludes our dissertation by summarising the work and the contributions presented in the preceding chapters. Furthermore, it discusses some directions for possible improvements and future works on the subject of this thesis.

5.2 Thesis Results

Motivated by the need to design cooperative agents capable of making decisions about their task allocation in contexts where the tasks are interdependent, this dissertation makes several contributions to the field of Multi-agent Systems. Our contributions are a mix of scientific and practical contributions. These contributions are communicated in a paper and three patents listed in section 5.3 and are summarised as follows:

- A survey of task allocation approaches with a detailed analysis on formalisation and resolution aspects. We have presented various solutions that we globally compare based on different factors for each family of methods. The survey tackles both cases of independent and interdependent tasks. In the case of the interdependent tasks, we have examined the problem from a coalition formation perspective and the task allocation perspective with sequential tasks. It focuses on multi-agent tasks and decentralised configurations.

- An end-to-end approach to allocate tasks in critical, dynamic and complex missions. The approach applies to multi-agent independent tasks and heterogeneous cooperative agents. It includes an informal approach to guide designers in modelling a utility function when the experts cannot provide one, and implementing a state of the art task allocation algorithm. We have run empirical experiments on our use case with our simulator. The approach, the use case, and the results cannot be presented in this document for confidentiality reasons.

- We have proposed a general modelisation of the interdependent task allocation problem considering both qualitative and quantitative properties for tasks and agents. Our modelisation is generic and covers the agent

characteristics, the different task requirements, both for a single task or a combination of tasks, the different agents-tasks relations, and the utility function. The tasks being multi-agent, our modelisation of the task allocation problem falls into the coalition formation framework. Furthermore, the dependencies between tasks can appear on two levels. Firstly, they appear in the requirements that concern a task combination rather than a single task. The task combination requirements make the corresponding tasks interdependent since their coalitions must fulfil that requirement together. Secondly, task dependencies are expressed by the global utility function that involves the whole coalition structure as a whole and does not simply represent a sum of defined utilities for each coalition.

- We have introduced a novel approach for a feasible coalition structure formation for interdependent task allocation under the coalition formation paradigm. The approach is fully decentralised, based on a token-passing process and a messages exchange protocol, and composed of two stages. As the anytime solution proposal was the main requirement, the first stage aims to find a feasible coalition structure representing an anytime solution. The second stage allows finding other feasible coalition structures with better utilities. When the goal is limited to finding a feasible coalition structure, the process ends by the end of the first stage. This is noted as the algorithm FICSAM. In this algorithm, we implemented a CSP method to generate a feasible coalition structure or check a coalition structure feasibility, using the mirroring between our modelisation of the task requirements and the constraints. When more time is available for finding a better solution in terms of utility, we have developed for the second stage three extensions to the previous algorithm under the name IFICSAM. In the first extension, agents test different one-to-one swapping with other agents to check if the swap can result in another coalition structure that is feasible too, yet that increases the global utility. In the second extension, agents try to do many-to-one swapping where each tries to shift its place with a combination of agents. We improve this second extension by a third one where we use a Branch & Bound CSP technique to choose the best combination of agents that is a candidate to a swap with the agent in question.

- We have built a scenario generator for a specific use case to test our

approaches. Our generator randomly generates the attributes' values of agent characteristics, single task requirements, and task combination requirements. This allows for an extensive and various benchmark of scenarios.

- We have developed the global procedure algorithm used to orchestrate the token passing to define the underlying communication protocol between agents. This procedure defines the agents' behaviour when they receive each of the different types of messages and their content. Each agent knows only its own characteristics at the beginning of this process. However, this algorithm allows agents to share their characteristics with others and update their knowledge of the other agents. In addition, the token passing organisation allows having mandatory sequential decision-making due to the interdependent character of the tasks.
- We have presented several experimental results showing that the agents can efficiently cooperate and form coalitions and that efficient task allocation can be achieved by decentralisation even when tasks are interdependent. We compared the results of our approach with the results of a centralised method built with COP techniques. We have shown that our decentralised approaches terminate in very reasonable times with near-optimal performances.

5.3 Limitations and Recommendations for Future Work

The present research work leads to several open fronts and research perspectives that merit to be investigated:

- The first research direction concerns the development of use cases with configuration that shows the advantages of the IFICSAM extension with the many-to-one swapping using the B&B CSP techniques. Due to lack of time, we could not generate scenarios that highlight the features of this approach.
- The second research direction regards the study of the token passing strategy. We observed the order by which the agents receive the token and,

thus, the token passing graph influence the resulting coalition structures. The token passing process is, of course, application-dependent. However, if the application allows different configurations, a study on the effects of these different configurations on the results can lead to optimisation suggestions.

- It is to be noticed that, for more effectiveness, our algorithm can benefit from some heuristics. The objective is to obtain not only a feasible initial task allocation but also a "good" initial task allocation. The third research direction concerns adding heuristics to our algorithms. Furthermore, this might be advantageous for the second stage of the algorithm that continues improving the task allocation. The heuristics can be for instance: to be greedy regarding the coalition to join in the first stage, prioritise tasks with more demanding requirements, namely tasks that are more "difficult" to cover by the set of available agents, or start allocating tasks that require the rarest agents characteristics.

- Even though our algorithms are conceived to be dynamic, we have only tested them on static environments. Another research direction concerns the study of our approach in the case of a dynamic environment. The goal is to relax the assumption of having a static environment and make some minor changes to our algorithms to guarantee their robustness to dynamism. The basic idea is to have a trigger announcing a change in the environment. This trigger activates an intermediary stage that makes the agents update their current coalition structure depending on the occurring changes and resume the process normally.

- A research direction, focusing on the constraint solving techniques, regards the implementation of FICSAM and IFICSAM with other solvers. The objective is to analyse the results using different solvers. Notably, we would like to implement the OR-Tools solver, the winner of the Minizinc competition for the last few years. It could be interesting to test IBM's CP or SCIP solvers that are known to have very good performances.

- From a utility function design perspective, a research direction concerns treating more specific cases of interdependent tasks and the appropriate utility function formulation. We observed that expressing the utility function is not always easy without one given by the end-user. The idea is to

provide an approach to guide the utility function design for specific, more common task dependencies cases.

- In the same previous research direction, a sub-direction concerns focusing on coalitional utility functions of specific types or classes to enhance the coalition structure generation in terms of quality and execution time. For example, it is interesting to examine coalition structure generation assuming superadditive or subadditive functions [Bistaffa, Farinelli, Jesús Cerquides, et al. 2014; Bistaffa, Farinelli, Chalkiadakis, et al. 2017; Dang et al. 2006].
- Last but not least, another research direction concerns studying scrupulously the impact of our scenario generator variations on the complexity of constrained allocation problems it is called for.

Bibliography

Abdallah, Sherief and Victor Lesser (2004). **Organization-based cooperative coalition formation**. In: *Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004.(IAT 2004).* IEEE, 162–168 (see page 12).

Agnisarman, Sruthy, Snowil Lopes, Kapil Madathil, Kalyan Piratla, and Anand Gramopadhye (2019). **A survey of automation-enabled human-in-the-loop systems for infrastructure visual inspection**. *Automation in Construction* 97, 52–76 (see page 94).

Ahmadoun, Douae, Élise Bonzon, Cédric Buron, Pavlos Moraitis, Pierre Savéant, and Onn Shehory (2021). **Decentralized coalition structure formation for interdependent tasks allocation**. In: *ICTAI 2021 - 33rd IEEE International Conference on Tools with Artificial Intelligence* (see pages 17, 61, 64, 133).

Ahmadoun, Douae, Élise Bonzon, Pavlos Moraitis, Cédric Buron, and Pierre Savéant (2021). **TSP20X5128 Thales ref at LAVOIX**. Original document in French (see pages 17, 64, 133).

Ahmadoun, Douae, Cédric Buron, Alain Peres, and Mathieu Leconte (2020). **TSP21X5067 Thales ref at LAVOIX**. Original document in French (see pages 6, 17, 132, 133).

Amato, Christopher, Alan Carlin, and Shlomo Zilberstein (2007). **Bounded dynamic programming for decentralized POMDPs**. In: *AAMAS workshop on multi-agent sequential decision making in uncertain domains* (see page 134).

Aumann, Robert J and Jacques H Dreze (1974). **Cooperative games with coalition structures**. *International Journal of game theory*, 217–237 (see pages 12, 134).

Bai, Quan, Fenghui Ren, Katsuhide Fujita, Minjie Zhang, and Takayuki Ito (2017). **Multi-agent and Complex Systems**. Springer (see page 9).

Beck, Zoltán, WLT Teacy, NR Jennings, and AC Rogers (2016). **Online planning for collaborative search and rescue by heterogeneous robot teams**. In: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*. Association of Computing Machinery (see pages 13, 41, 135).

Behrens, Jan Kristof, Ralph Lange, and Masoumeh Mansouri (2019). **A constraint programming approach to simultaneous task allocation and motion scheduling for industrial dual-arm manipulation tasks**. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 8705–8711 (see pages 40, 135).

Binshtok, Maxim, Ronen I Brafman, Solomon Eyal Shimony, Ajay Martin, and Craig Boutilier (2007). **Computing optimal subsets**. In: *AAAI*, 1231–1236 (see pages 85–88).

Bistaffa, Filippo, Alessandro Farinelli, Jesús Cerquides, Juan Antonio Rodríguez-Aguilar, and Sarvapali Ramchurn (2014). **Anytime coalition structure generation on synergy graphs** (see page 112).

Bistaffa, Filippo, Alessandro Farinelli, Georgios Chalkiadakis, and Sarvapali D Ramchurn (2017). **A cooperative game-theoretic approach to the social ridesharing problem**. *Artificial Intelligence* 246, 86–117 (see page 112).

Boes, Jérémy and Frédéric Migeon (2017). **Self-organizing multi-agent systems for the control of complex systems**. *Journal of Systems and Software* 134, 12–28 (see page 9).

Boularias, Abdeslam and Brahim Chaib-Draa (2008). **Exact dynamic programming for decentralized POMDPs with lossless policy compression**. In: *ICAPS*. AAAI Press, 20–27 (see page 134).

Bouyssou, Denis, Thierry Marchant, Marc Pirlot, Patrice Perny, Alexis Tsoukias, and Philippe Vincke (2000). **Evaluation and decision models: a critical perspective**. Vol. 32. Springer Science & Business Media (see page 59).

Brandt, Felix, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia (2016). **Handbook of computational social choice**. Cambridge University Press (see page 4).

Brunet, Luc Luc PV (2008). **Consensus-based auctions for decentralized task assignment**. PhD thesis. Massachusetts Institute of Technology (see pages 25, 134).

Brutschy, Arne, Giovanni Pini, Carlo Pinciroli, Mauro Birattari, and Marco Dorigo (2014). **Self-organized task allocation to sequentially interdependent tasks in swarm robotics**. *Autonomous agents and multi-agent systems (AAMAS)* 28:1, 101–125 (see pages 37, 42, 135).

Buckman, Noam, Han-Lim Choi, and Jonathan P How (2019). **Partial replanning for decentralized dynamic task allocation**. In: *AIAA Scitech 2019 Forum*, 0915 (see pages 25, 134).

Buşoniu, Lucian, Robert Babuška, and Bart De Schutter (2010). **Multi-agent reinforcement learning: An overview**. *Innovations in multi-agent systems and applications-1*, 183–221 (see page 134).

Capezzuto, Luca, Danesh Tarapore, and Sarvapali D Ramchurn (2021). **Large-scale, dynamic and distributed coalition formation with spatial and temporal constraints**. In: *European Conference on Multi-Agent Systems*. Springer, 108–125 (see page 42).

Chalkiadakis, Georgios, Edith Elkind, and Michael Wooldridge (2011). **Computational aspects of cooperative game theory**. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 5:6, 1–168 (see page 12).

Choi, Han-Lim, Luc Brunet, and Jonathan P How (2009). **Consensus-based decentralized auctions for robust task allocation**. *IEEE transactions on robotics* 25:4, 912–926 (see pages 23, 134).

Chu, Geoffrey, Peter J. Stuckey, Andreas Schutt, Thorsten Ehlers, Graeme Gange, and Kathryn Francis (2018). **Chuffed, a lazy clause generation solver**. https://github.com/chuffed/chuffed (see pages 93, 98).

Coles, Andrew, Maria Fox, Keith Halsey, Derek Long, and Amanda Smith (2009). **Managing concurrency in temporal planning using planner-scheduler interaction**. *Artificial Intelligence* 173:1, 1–44 (see pages 39, 40).

Coria, José A García, José A Castellanos-Garzón, and Juan M Corchado (2014). **Intelligent business processes composition based on multi-agent systems**. *Expert Systems with Applications* 41:4, 1189–1205 (see page 9).

Coyote, Bertha Little and James D Thompson (1967). **Organizations in action: Social science bases of administrative theory**. Vol. 10. McGraw-Hill College (see page 36).

Dahl, Torbjørn S, Maja Matarić, and Gaurav S Sukhatme (2009). **Multi-robot task allocation through vacancy chain scheduling**. *Robotics and Autonomous Systems* 57:6-7, 674–687 (see pages 41, 135).

Dang, Viet Dung, Rajdeep K Dash, Alex Rogers, and Nicholas R Jennings (2006). **Overlapping coalition formation for efficient data fusion in multi-sensor networks**. In: *AAAI*. Vol. 6, 635–640 (see page 112).

Dechter, Rina (2003). **Constraint processing**. Elsevier Morgan Kaufmann. ISBN: 978-1-55860-890-0. URL: http://www.elsevier.com/wps/find/bookdescription.agents/678024/description (see pages 38, 72, 93).

Dechter, Rina, David Cohen, et al. (2003). **Constraint processing**. Morgan Kaufmann (see page 27).

Dias, M Bernardine, Robert Zlot, Nidhi Kalra, and Anthony Stentz (2006). **Market-based multirobot coordination: A survey and analysis**. *Proceedings of the IEEE* 94:7, 1257–1270 (see pages 27, 134).

ElGibreen, Hebah and Kamal Youcef-Toumi (2019). **Dynamic task allocation in an uncertain environment with heterogeneous multi-agents**. *Autonomous Robots* 43:7, 1639–1664 (see pages 26, 134).

Farinelli, Alessandro, Alex Rogers, and Nick R Jennings (2014). **Agent-based decentralised coordination for sensor networks using the max-sum algorithm**. *Autonomous agents and multi-agent systems* 28:3, 337–380 (see pages 28, 35).

Farinelli, Alessandro, Alex Rogers, Adrian Petcu, and Nicholas R Jennings (2008). **Decentralised coordination of low-power embedded devices using the max-sum algorithm**. In: *Seventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-08)*, 639–646 (see pages 13, 32, 134).

Fioretto, Ferdinando, Enrico Pontelli, and William Yeoh (2018). **Distributed constraint optimization problems and applications: A survey**. *Journal of Artificial Intelligence Research (JAIR)*, 623–698 (see pages 28, 33, 34, 134).

Fipa, ACL (2002). **Fipa acl message structure specification**. *Foundation for Intelligent Physical Agents, http://www. fipa. org/specs/fipa00061/SC00061G. html (30.6. 2004)* (see page 66).

Fitzpatrick, Stephen and Lambert Meertens (2003). **Distributed coordination through anarchic optimization**. In: *Distributed Sensor Networks*. Springer, 257–295 (see pages 32, 134).

Gayraud, Lionel, Cédric Buron, and Douae Ahmadoun (2021). **TSP21X5158 Thales ref at LAVOIX**. Original document in French (see pages 6, 17, 132, 133).

Gerkey, Brian P and Maja J Matarić (2004). **A formal analysis and taxonomy of task allocation in multi-robot systems**. *The International journal of robotics research* 23:9, 939–954 (see pages 20, 21, 37, 41, 42, 57, 134).

Gini, Maria (2017). **Multi-robot allocation of tasks with temporal and ordering constraints**. In: *Thirty-First AAAI Conference on Artificial Intelligence* (see page 40).

Guttman, Louis (1944). **A basis for scaling qualitative data**. *American sociological review* 9:2, 139–150 (see page 36).

Hajduková, Jana (2006). **Coalition formation games: A survey**. *International Game Theory Review* 8:04, 613–641 (see page 37).

Hansen, Eric A, Daniel S Bernstein, and Shlomo Zilberstein (2004). **Dynamic programming for partially observable stochastic games**. In: *AAAI*, 709–715 (see page 134).

Hausknecht, Matthew and Peter Stone (2015). **Deep recurrent q-learning for partially observable mdps**. In: *2015 aaai fall symposium series* (see page 134).

Hirayama, Katsutoshi and Makoto Yokoo (1997). **Distributed partial constraint satisfaction problem**. In: *International conference on principles and practice of constraint programming*. Springer, 222–236 (see pages 31, 134).

Hooshangi, Navid and Ali Asghar Alesheikh (2017). **Agent-based task allocation under uncertainties in disaster environments: An approach to interval uncertainty**. *International Journal of Disaster Risk Reduction* 24, 160–171 (see page 13).

Horling, Bryan and Victor Lesser (2004). **A survey of multi-agent organizational paradigms**. *The Knowledge engineering review* 19:4, 281–316 (see page 10).

Johnson, Matthew, Jeffrey M Bradshaw, Paul J Feltovich, Catholijn M Jonker, M Birna Van Riemsdijk, and Maarten Sierhuis (2014). **Coactive design: Designing support for interdependence in joint activity**. *Journal of Human-Robot Interaction* 3:1, 43–69 (see page 37).

Kaminka, Gal A, David V Pynadath, and Milind Tambe (2002). **Monitoring teams by overhearing: A multi-agent plan-recognition approach**. *Journal of artificial intelligence research* 17, 83–135 (see page 10).

Kitano, Hiroaki (2000). **Robocup rescue: A grand challenge for multi-agent systems**. In: *Proceedings fourth international conference on MultiAgent systems*. IEEE, 5–12 (see page 9).

Kochenderfer, Mykel J (2015). **Decision making under uncertainty: theory and application**. MIT Press (see page 134).

Kóczy, László (2018). **Partition Function Form Games: Coalitional Games with Externalities**. Springer (see page 39).

Kok, Jelle R and Nikos Vlassis (2006). **Collaborative multiagent reinforcement learning by payoff propagation**. *Journal of Machine Learning Research* 7, 1789–1828 (see page 33).

Korsah, G Ayorkor, Anthony Stentz, and M Bernardine Dias (2013). **A comprehensive taxonomy for multi-robot task allocation**. *The International Journal of Robotics Research* 32:12, 1495–1512 (see page 20).

Kovtunenko, Alexey, Marat Timirov, and Azat Bilyalov (2019). **Multi-agent approach to computational resource allocation in edge computing**. In: *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*. Springer, 135–146 (see page 9).

Lee, J-H and C-O Kim (2008). **Multi-agent systems applications in manufacturing systems and supply chain management: a review paper**. *International Journal of Production Research* 46:1, 233–265 (see page 9).

Lee, Wonki, Neil Vaughan, and Daeeun Kim (2020). **Task allocation into a foraging task with a series of subtasks in swarm robotic system**. *IEEE Access* 8, 107549–107561 (see page 41).

Lematta, Glenn J, Pamela B Coleman, Shawaiz A Bhatti, Erin K Chiou, Nathan J McNeese, Mustafa Demir, and Nancy J Cooke (2019). **Developing Human-Robot Team Interdependence in a Synthetic Task Environment**. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 63. 1. SAGE Publications Sage CA: Los Angeles, CA, 1503–1507 (see page 37).

Liu, Jiming and Jianbing Wu (2018). **Multiagent robotic systems**. CRC press (see page 9).

Liu, Xiaolan, Jiadong Yu, Zhiyong Feng, and Yue Gao (2020). **Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing**. *China Communications* 17:9, 220–236 (see page 9).

Losey, Dylan P, Mengxi Li, Jeannette Bohg, and Dorsa Sadigh (2020). **Learning from my partner's actions: Roles in decentralized robot teams**. In: *Conference on robot learning*. PMLR, 752–765 (see page 13).

Luo, Lingzhi, Nilanjan Chakraborty, and Katia Sycara (2013). **Distributed algorithm design for multi-robot task assignment with deadlines for tasks**. In: *2013 IEEE International Conference on Robotics and Automation*. IEEE, 3007–3013 (see page 27).

Macarthur, Kathryn, Ruben Stranders, Sarvapali Ramchurn, and Nicholas Jennings (2011). **A distributed anytime algorithm for dynamic task allocation in multi-agent systems**. In: *AAAI Conference on Artificial Intelligence* (see page 6).

Maheswaran, Rajiv T, Jonathan P Pearce, Milind Tambe, et al. (2004). **Distributed Algorithms for DCOP: A Graphical-Game-Based Approach.** In: *ISCA PDCS*. Citeseer, 432–439 (see pages 32, 134).

Mainland, Geoffrey, David C Parkes, and Matt Welsh (2005). **Decentralized, adaptive resource allocation for sensor networks**. In: *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, 315–328 (see page 13).

Miloradović, Branko, Mirgita Frasheri, Baran Cürüklü, Mikael Ekström, and Alessandro Vittorio Papadopoulos (2019). **TAMER: Task Allocation in Multi-robot Systems Through an Entity-Relationship Model**. In: *International Conference on Principles and Practice of Multi-Agent Systems (PRIMA)*, 478–486 (see page 20).

Modi, Pragnesh Jay, Wei-Min Shen, Milind Tambe, and Makoto Yokoo (2005). **ADOPT: Asynchronous distributed constraint optimization with quality guarantees**. *Artificial Intelligence* 161:1-2, 149–180 (see page 134).

Moreno, Antonio (2003). **Medical applications of multi-agent systems**. *Computer Science and Mathematics Department, University of Rovira, Spain* (see page 9).

Mosteo, Alejandro R and Luis Montano (2010). **A survey of multi-robot task allocation**. *Instituto de Investigación en Ingeniería de Aragón, Univ. of Zaragoza, Spain, Technical Report No. AMI-009-10-TEC* (see pages 22, 134).

Munir, Md Shirajum, Sarder Fakhrul Abedin, Nguyen H Tran, Zhu Han, Choong Seon Hong, et al. (2019). **A multi-agent system toward the green edge computing with microgrid**. In: *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–7 (see page 9).

Myerson, Roger B (1977). **Values of games in partition function form**. *International Journal of Game Theory* 6, 23–31 (see page 38).

Nealon, John and Antonio Moreno (2003). **Agent-based applications in health care**. In: *Applications of software agent technology in the health care domain*. Springer, 3–18 (see page 9).

Nethercote, Nicholas, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack (2007). **MiniZinc: Towards a Standard CP Modelling Language**. In: *Principles and Practice of Constraint Programming – CP 2007*. Ed. by Christian Bessière, 529–543. ISBN: 978-3-540-74970-7 (see page 98).

Nguyen, Duc Thien, William Yeoh, and Hoong Chuin Lau (2012). **Stochastic dominance in stochastic DCOPs for risk-sensitive applications**. In: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 257–264 (see page 35).

– (2013). **Distributed Gibbs: A memory-bounded sampling-based DCOP algorithm** (see pages 32, 134).

Okimoto, Tenda, Yongjoon Joe, Atsushi Iwasaki, Makoto Yokoo, and Boi Faltings (2011). **Pseudo-tree-based incomplete algorithm for distributed constraint optimization with quality bounds**. In: *International conference on principles and practice of constraint programming*. Springer, 660–674 (see pages 32, 134).

Oliehoek, Frans A, Christopher Amato, et al. (2016). **A concise introduction to decentralized POMDPs** (see page 134).

Ottens, Brammert, Christos Dimitrakakis, and Boi Faltings (2017). **DUCT: An upper confidence bound approach to distributed constraint optimization problems**. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8:5, 1–27 (see page 32).

Parker, James, Alessandro Farinelli, and Maria Gini (2018). **Lazy max-sum for allocation of tasks with growing costs**. *Robotics and Autonomous Systems* 110, 44–56 (see page 28).

Pipattanasomporn, Manisa, Hassan Feroze, and Saifur Rahman (2009). **Multi-agent systems in a distributed smart grid: Design and implementation**. In: *2009 IEEE/PES Power Systems Conference and Exposition*. IEEE, 1–8 (see page 9).

Präntare, Fredrik and Fredrik Heintz (2020). **An anytime algorithm for optimal simultaneous coalition structure generation and assignment**. *Autonomous Agents and Multi-Agent Systems* 34:1, 1–31 (see pages 38, 135).

Pujol-Gonzalez, Marc, Jesus Cerquides, Alessandro Farinelli, Pedro Meseguer, and Juan Antonio Rodriguez-Aguilar (2015). **Efficient inter-team task allocation in robocup rescue**. In: *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*, 413–421 (see page 35).

Pujol-Gonzalez, Marc, Jesus Cerquides, Alessandro Farinelli, Pedro Meseguer, and Juan Antonio Rodríguez-Aguilar (2014). **Binary max-sum for multi-team task allocation in robocup rescue**. In: *Optimisation in Multi-Agent Systems and Distributed Constraint Reasoning* (see page 28).

Pujol-Gonzalez, Marc, Jesus Cerquides, Pedro Meseguer, Juan A Rodriguez-Aguilar, and Milind Tambe (2018). **Decentralized dynamic task allocation for UAVs with limited communication range**. preprint arXiv:1809.07863 (see page 28).

Puranam, Phanish, Marlo Raveendran, and Thorbjørn Knudsen (2012). **Organization design: The epistemic interdependence perspective**. *Academy of Management Review* 37:3, 419–440 (see page 36).

Rahwan, Talal, Tomasz P Michalak, Michael Wooldridge, and Nicholas R Jennings (2015). **Coalition structure generation: A survey**. *Artificial Intelligence* 229, 139–174 (see pages 38, 135).

Rahwan, Talal, Tomasz Michalak, Michael Wooldridge, and Nicholas R Jennings (2012). **Anytime coalition structure generation in multi-agent systems with positive or negative externalities**. *Artificial Intelligence* 186, 95–122 (see page 39).

Ramchurn, Sarvapali D, Alessandro Farinelli, Kathryn S Macarthur, and Nicholas R Jennings (2010). **Decentralized coordination in robocup rescue**. *The Computer Journal* 53:9, 1447–1461 (see pages 28, 34, 35).

Ramchurn, Sarvapali D, Claudio Mezzetti, Andrea Giovannucci, Juan A Rodriguez-Aguilar, Rajdeep K Dash, and Nicholas R Jennings (2009). **Trust-based mechanisms for robust and efficient task allocation in the presence of execution uncertainty**. *Journal of Artificial Intelligence Research* 35, 119–159 (see page 36).

Ramchurn, Sarvapali D, Mariya Polukarov, Alessandro Farinelli, Nick Jennings, and Cuong Trong (2010). **Coalition formation with spatial and temporal constraints**. In: *International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2010)*, 1181–1188 (see pages 38, 42).

Rogers, Alex, Sarvapali Ramchurn, and Nicholas Jennings (2012). **Delivering the smart grid: Challenges for autonomous agents and multi-agent**

systems research. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 26. 1 (see page 9).

Russell, Stuart and Peter Norvig (2002). **Artificial intelligence: a modern approach** (see pages 7, 9).

Rzevski, George (2012). **Modelling large complex systems using multi-agent technology**. In: *2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. IEEE, 434–437 (see page 9).

Sandholm, Tuomas W and Victor R Lesser (1995). **Coalition formation among bounded rational agents**. In: *IJCAI (1)*. Citeseer, 662–671 (see page 12).

Sandholm, Tuomas, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé (1999). **Coalition structure generation with worst case guarantees**. *Artificial intelligence* 111:1-2, 209–238 (see pages 5, 39).

Scerri, Paul, Alessandro Farinelli, Steven Okamoto, and Milind Tambe (2005). **Allocating tasks in extreme teams**. In: *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 727–734 (see page 10).

Schetter, Thomas, Mark Campbell, and Derek Surka (2003). **Multiple agent-based autonomy for satellite constellations**. *Artificial Intelligence* 145:1-2, 147–180 (see page 13).

Shehory, Onn M, Katia Sycara, and Somesh Jha (1997). **Multi-agent coordination through coalition formation**. In: *International Workshop on Agent Theories, Architectures, and Languages*. Springer, 143–154 (see page 12).

Shehory, Onn and Sarit Kraus (1998). **Methods for task allocation via agent coalition formation**. *Artificial intelligence* 101:1-2, 165–200 (see pages 37, 39, 76, 134).

Shenoy, Prakash P (1979). **On coalition formation: a game-theoretical approach**. *International journal of game theory* 8:3, 133–164 (see pages 12, 134).

Siskos, Yannis, Evangelos Grigoroudis, and Nikolaos F Matsatsinis (2005). **UTA methods**. In: *Multiple criteria decision analysis: State of the art surveys*. Springer, 297–334 (see page 6).

Smith, Reid G (1980). **The contract net protocol: High-level communication and control in a distributed problem solver**. *IEEE Transactions on computers* 29:12, 1104–1113 (see pages 23, 134).

Stone, Peter and Manuela Veloso (2000). **Multiagent systems: A survey from a machine learning perspective**. *Autonomous Robots* 8:3, 345–383 (see page 9).

Sutton, Richard S, Andrew G Barto, et al. (1998). **Introduction to reinforcement learning**. MIT press Cambridge (see page 134).

Szer, Daniel and François Charpillet (2005). **An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs**. In: *European Conference on Machine Learning (ECML)*, 389–399 (see page 134).

Tambe, Milind, Jafar Adibi, Yaser Al-Onaizan, Ali Erdem, Gal A Kaminka, Stacy C Marsella, and Ion Muslea (1999). **Building agent teams using an explicit teamwork model and learning**. *Artificial intelligence* 110:2, 215–239 (see page 10).

Tapia, Dante I and Juan M Corchado (2009). **An ambient intelligence based multi-agent system for alzheimer health care**. *International Journal of Ambient Computing and Intelligence (IJACI)* 1:1, 15–26 (see page 9).

Thompson, James D, Mayer N Zald, and W Richard Scott (2017). **Organizations in action: Social science bases of administrative theory**. Routledge (see page 36).

Vinyals, Meritxell, Juan A Rodriguez-Aguilar, and Jesus Cerquides (2011). **A survey on sensor networks from a multiagent perspective**. *The Computer Journal* 54:3, 455–470 (see page 35).

Wang, Pengyuan and Manimaran Govindarasu (2020). **Multi-agent based attack-resilient system integrity protection for smart grid**. *IEEE Transactions on Smart Grid* 11:4, 3447–3456 (see page 9).

Wooldridge, Michael (2009). **An introduction to multiagent systems**. John Wiley & Sons (see pages 23, 26, 134).

Wooldridge, Michael and Nicholas R Jennings (1995). **Intelligent agents: Theory and practice**. *The knowledge engineering review* 10:2, 115–152 (see page 7).

Woolridge, Michael and Michael J Wooldridge (2001). **Introduction to multiagent systems**. John Wiley & Sons, Inc. (see page 7).

Wu, Feng and Nicholas Jennings (2014). **Regret-based multi-agent coordination with uncertain task rewards**. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 1 (see page 35).

Yao, Feng, Jiting Li, Yuning Chen, Xiaogeng Chu, and Bang Zhao (2019). **Task allocation strategies for cooperative task planning of multi-autonomous satellite constellation**. *Advances in Space Research* 63:2, 1073–1084 (see page 13).

Yedidsion, Harel, Roie Zivan, and Alessandro Farinelli (2018). **Applying max-sum to teams of mobile sensing agents**. *Engineering Applications of Artificial Intelligence* 71, 87–99 (see page 35).

Yeoh, William (2010). **Speeding up distributed constraint optimization search algorithms**. Citeseer (see page 30).

Yeoh, William, Pradeep Varakantham, Xiaoxun Sun, and Sven Koenig (2015). **Incremental DCOP search algorithms for solving dynamic DCOP problems**. In: *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. Vol. 2. IEEE, 257–264 (see page 34).

Zhang, Kai, Emmanuel G Collins Jr, and Dongqing Shi (2012). **Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction**. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 7:2, 1–22 (see page 13).

Zhao, Fangyun, Curt Henrichs, and Bilge Mutlu (2020). **Task Interdependence in Human-Robot Teaming**. In: *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 1143–1149 (see page 37).

Żytniewski, Mariusz (2016). **Integration of knowledge management systems and business processes using multi-agent systems**. *International Journal of Computational Intelligence Studies* 5:2, 180–196 (see page 9).

List of Publications

Articles in Refereed Conference Proceedings

[1] **Decentralized Coalition Structure Formation for Interdependent Tasks Allocation**. In: *ICTAI 2021 - 33rd IEEE International Conference on Tools with Artificial Intelligence*. Joint work with Bonzon, Élise, Cédric Buron, Pavlos Moraitis, Pierre Savéant, and Onn Shehory.

Patents

[2] Joint work with Bonzon, Élise, Pavlos Moraitis, Cédric Buron, and Pierre Savéant. **TSP20X5128 Thales ref at LAVOIX**. Original document in French.

[3] Joint work with Buron, Cédric, Alain Peres, and Mathieu Leconte. **TSP21X5067 Thales ref at LAVOIX**. Original document in French.

[4] Joint work with Gayraud, Lionel and Cédric Buron. **TSP21X5158 Thales ref at LAVOIX**. Original document in French.

Appendices

A Résumé long en Français

A.1 Introduction

A.1.1 Contexte et Motivation

Les avancées technologiques dans la construction des robots, les technologies des capteurs et les capacités de calcul ont renouvelé l'intérêt pour les méthodes d'allocation de tâches multi-agents. Avec la baisse des coûts de production, ces progrès accélèrent l'avènement de ces technologies et encouragent leur utilisation dans des missions qualifiées de complexes et critiques avec des configurations difficiles voire impossibles à gérer par un seul robot mais beaucoup plus faciles à résoudre en utilisant une équipe de robots. Tirer pleinement parti de ces robots nécessite une intelligence incarnée leur permettant de s'allouer des tâches en fonction de leurs capacités, des exigences des tâches de la mission et de l'évaluation de la réalisation de ces tâches par des groupes particuliers d'agents.

On peut distinguer deux situations différentes. Dans certaines applications, les tâches composant la tâche globale sont indépendantes, *i.e.* la qualité d'exécution d'une tâche ne dépend que des agents qui lui sont affectés mais pas des autres tâches. Dans ce cas, l'utilité de la tâche globale est additive (la somme des utilités individuelles des tâches). En revanche, dans d'autres applications, il n'est pas possible d'évaluer l'allocation d'agents spécifiques à une tâche localement sans considérer les allocations des autres tâches. Ainsi, les utilités des tâches ne peuvent pas s'additionner et l'utilité de la tâche globale ne peut être calculée que dans son ensemble. Cela se produit, par exemple, lorsque deux tâches partagent une ressource essentielle ou quand les tâches ont un certain ordre d'exécution. Ces tâches sont dites interdépendantes. En plus de cette inclinaison liée à l'utilité globale de l'interdépendance, cette dernière peut être présente au niveau des contraintes de faisabilité. C'est le cas où deux ou plusieurs tâches ne peuvent pas être accomplies si elles ne vérifient pas ensemble certaines contraintes. Sinon, les tâches individuelles concernées ne peuvent pas être exécutées même si chacune remplit ses contraintes locales.

La thèse, réalisée dans un cadre industriel, est basée sur un cas d'utilisation dont les agents sont autonomes, coopératifs et hétérogènes, devant effectuer un ensemble de tâches complexes. En assumant l'indépendance des tâches, nous avons modélisé le problème sous un paradigme spécifique, choisi un algorithme d'allocation de tâches convenant aux particularités de l'application, établi un guide avec des techniques MCDM pour la modélisation de la fonction d'utilité et implémenté ces techniques sur un simulateur spécialement conçu [Gayraud et al. 2021]. Les expérimentations empiriques donnaient de bons résultats lorsque les tâches n'avaient pas des effets les unes sur l'exécution des autres. Dans le cas contraire, nous avons eu des résultats inattendus par rapport aux prévisions opérationnelles. L'analyse de cette incohérence a relevé le caractère interdépendant des tâches que nous n'avons pas pris en compte dans notre modélisation. Un examen plus approfondi de l'interdépendance des tâches et l'observation de sa présence dans de nombreuses applications du monde réel, ont précipité l'orientation de ce travail vers le cas des tâches interdépendantes.

Le travail réalisé incluant la modélisation, l'approche suggérée et les résultats est documenté dans un brevet [Ahmadoun et al. 2020] qui a été mis au secret par l'Agence du Gouvernement français de la Défense.

A.1.2 Problématique de Recherche

Ce projet doctoral se focalise sur des agents mono-tâches et hétérogènes, de par leurs différentes capacités. Ces agents doivent réaliser, coopérativement, une mission composée de tâches interdépendantes dont chacune nécessite un sous-ensemble d'agents avec une combinaison spécifique de capacités, d'où le besoin de former des coalitions. L'environnement est considéré statique, mais nous pouvons étendre à un environnement dynamique. Pour s'adapter à la nature des applications du monde réel critiques, avec une communication instable et avec des robots ou des drones comme agents, nous nous concentrons sur une configuration décentralisée et *anytime*.

La problématique centrale étudiée dans cette thèse est: **L'allocation décentralisée de tâches via la formation de coalition pour des tâches interdépendantes**. Le but est de définir une affectation de coalitions d'agents aux tâches en faisant correspondre les capacités des agents avec les exigences des tâches et en maximisant une fonction d'utilité même si les tâches ne peuvent pas être évaluées séparément vu leur interdépendance. Le mécanisme doit être complètement décentralisé et chaque agent doit pouvoir décider en fonction

des informations dont il dispose et communiquer avec d'autres agents. Enfin, nous considérons des contextes où les tâches sont multi-agents vu leur complexité et leur exigence de combinaisons particulières de capacités de la part des agents pour qu'elles soient faisables. D'où la formalisation du problème sous le paradigme de la formation de coalitions.

A.1.3 Contributions de la Thèse

Cette thèse a abouti à de nombreuses contributions. D'abord, une étude des différentes familles de méthodes d'allocation de tâches indépendantes et interdépendantes avec une comparaison globale basée sur un nombre de facteurs. Ensuite, une approche de bout en bout pour allouer des tâches indépendantes dans des missions critiques, dynamiques et complexes accompagné d'un guide informel de la modélisation d'une fonction d'utilité pour les concepteurs [Ahmadoun et al. 2020; Gayraud et al. 2021]. Aussi, une modélisation générale du problème d'allocation de tâches interdépendantes considérant à la fois les propriétés qualitatives et quantitatives des agents et des tâches. En plus, une nouvelle approche pour la formation de structure de coalitions faisable pour l'allocation de tâches interdépendantes [Ahmadoun et al. 2021]. Enfin un algorithme *anytime* et décentralisé en deux étapes pour l'allocation de tâches interdépendantes via la formation de coalitions avec trois extensions différentes ainsi qu'un protocole de communication inter-agents [Ahmadoun et al. 2021; Ahmadoun et al. 2021].

A.1.4 Plan du Résumé

Dans la suite de ce résumé, sont décrites les méthodes phares de la littérature de l'allocation de tâches dans le chapitre A.2. Une formalisation de l'allocation des tâches interdépendantes est ensuite présentée dans le chapitre A.3. Dans le chapitre A.4, une nouvelle approche est introduite avec différents algorithmes décentralisés pour résoudre le problème de la génération puis de l'amélioration de structures de coalitions faisables pour l'allocation de tâches interdépendantes. Ce travail est enfin conclu avec les apports et les limites de nos contributions ainsi que des perspectives pour le travail futur.

A.2 Etat de l'art

Ce chapitre donne un aperçu des principales approches de la littérature abordant le problème d'allocation de tâches. Nous nous concentrons fondamentalement sur un certain nombre des familles de méthodes d'allocation de tâches les plus populaires.

De nombreuses méthodes ont été proposées pour coordonner les agents coopératifs et produire une allocation de tâches indépendantes. Les méthodes basés sur le marché [Dias et al. 2006; Mosteo et al. 2010] proposent des enchères collaboratives sur des tâches avec les algorithmes CBAA et CBBA [Brunet 2008; Buckman et al. 2019; Choi et al. 2009; ElGibreen et al. 2019] ou contract nets [Smith 1980; Wooldridge 2009]. Le processus de décision multi-agents de Markov (l'un des cadres les plus courants pour l'optimisation multi-agents) introduit la programmation dynamique [Boularias et al. 2008; Hansen et al. 2004], la recherche heuristique exacte [Szer et al. 2005], méthodes approximatives [Amato et al. 2007; Kochenderfer 2015; Oliehoek et al. 2016] ou encore l'apprentissage par renforcement multi-agents [Buşoniu et al. 2010; Hausknecht et al. 2015; Sutton et al. 1998]. Les problèmes distribués d'optimisation des contraintes (DCOP) (une présentation exhaustive peut être trouvée dans [Fioretto et al. 2018]), est un paradigme multi-agent qui présente des méthodes allant des algorithmes complets [Hirayama et al. 1997; Modi et al. 2005] aux algorithmes approximatifs avec [**ottens2012duct**; Nguyen et al. 2013] et sans limites d'erreur [Farinelli, Rogers, Petcu, et al. 2008; Fitzpatrick et al. 2003; Maheswaran et al. 2004; Okimoto et al. 2011].

Les méthodes décentralisées d'allocation des tâches peuvent être étudiées selon plusieurs critères. Le premier critère est le type de problème que ces méthodes peuvent traiter tel qu'il est proposé par la classification de Gerkey [Gerkey et al. 2004]. Cependant, selon le problème abordé, d'autres caractéristiques peuvent être essentielles. En particulier, le temps de convergence de l'algorithme en fonction du nombre d'échanges entre agents, ainsi que la complexité des calculs effectués par chaque agent, sont des éléments essentiels selon les cas d'utilisation.

L'aspect spécifique de l'allocation des tâches à un système multi-agents (lié à notre problématique de recherche) est décrit dans la littérature par le concept de formation de coalitions. Ce concept est largement étudié dans le domaine de la théorie des jeux [Aumann et al. 1974; Shenoy 1979] mais il est aussi tout à fait applicable dans un contexte plus général d'allocation de tâches [O. Shehory

et al. 1998]. Dans ce cadre, les coalitions interdépendantes sont exprimées par les PFGs, se référant aux *Partition Function Games*, contrairement aux CFGs, pour *Characteristic Function Games* [Rahwan, T. P. Michalak, et al. 2015]. L'état actuel de l'état de l'art présente le cas des coalitions interdépendantes, qui nous intéresse afin de traiter les tâches interdépendantes, comme résolvable uniquement par recherche par force brute, à moins que des hypothèses supplémentaires, appelées externalités, ne soient placées sur la fonction de partition [Präntare et al. 2020]. Ces externalités sont liées à la fusion des coalitions, ce qui n'est pas adapté à un problème d'allocation de tâches où le nombre de coalitions est fixe (identique au nombre de tâches).

Dans les familles de méthodes évoquées et autres, il est souvent assumé que les tâches sont indépendantes. Les quelques travaux abordant le problème d'allocation des tâches interdépendentes se focalisent sur le cas particulier des interdépendances temporelles, où les contraintes de précédence sont considérées [Beck et al. 2016; Behrens et al. 2019; Brutschy et al. 2014; Dahl et al. 2009].

Ceci dit, les travaux de recherche présents dans la littérature ne proposent pas de solutions à notre problème. Aucune méthode n'a été suggérée pour trouver de manière décentralisée pour un système multi-agents une allocation des tâches entre agents sous le paradigme de la formation de coalitions dans le cas où il existe une interdépendance entre les tâches au sens général de l'interdépendance des tâches et qui vise à être appliquée aux applications du monde réel, et doit ainsi être *anytime*.

A.3 Modélisation du problème

L'objectif de ce chapitre est de modéliser le problème de l'alloacation de tâches sous le paradigme de la formation de coalitions dans le cas des tâches interdépendantes général.

Le problème consiste à définir un ensemble d'agents coopératifs mono-tâches $A = \{a_1, \ldots, a_{|A|}\}$. Chaque agent est décrit par un ensemble de caractéristiques dans $X = \{x_1, \ldots, x_{|X|}\}$ avec $x_k \in X$ un attribut considéré comme une fonction $x_k : A \rightarrow D_k$, où D_k est le domaine de valeurs pour l'attribut x_k et $x_k(a_i)$ est la valeur de la caractéristique x_k pour l'agent a_i.

On définit également un ensemble de tâches composant la tâche globale $T = \{t_1, \ldots, t_{|T|}\}$. Ces tâches ont des exigences qui sont définies sur deux niveaux: au niveau des tâches individuellement et au niveau des combinaisons

des tâches. D'abord, chacune des tâches $t_j \in T$ est décrite par un ensemble d'exigences $\Gamma_{t_j} = \{\gamma_1, ..., \gamma_m\}$ avec $\gamma_l \in \Gamma_{t_j}$ un attribut considéré comme une fonction $\gamma_l : T \rightarrow K_l$, où K_l est le domaine de valeurs pour l'attribut γ_l et $\gamma_l(t_j)$ est la valeur de l'exigence γ_l pour la tâche t_j.

Ensuite, on définit des exigences pour toute combinaison de tâches cbt_j dans $T_{cbt} = \{cbt_1, \ldots, cbt_n\}$ où $T_{cbt} \subseteq 2^T$. Ainsi, chaque combinaison de tâches $cbt_j \in T_{cbt}$ est décrite par un ensemble d'exigences $\Lambda_{cbt_j} = \{\lambda_1, \ldots, \lambda_p\}$ avec $\lambda_l \in \Lambda_{cbt_j}$ un attribut considéré comme une fonction $\Lambda_{cbt_j} : T \rightarrow F_l$, où F_l est le domaine de valeurs pour l'attribut λ_l et $\lambda_l(cbt_j)$ est la valeur de l'exigence λ_l pour la combinaison de tâches cbt_j.

Une structure de coalition $S = \{C_{t_1}, \ldots, C_{t_{|T|}}\}$ est un ensemble de coalitions d'agents dont chaque coalition C_{t_j} (de par ses agents) est affectée à une tâche $t_j \in T$. L'objectif est donc de trouver une structure de coalition S faisable qui maximise l'évaluation d'allocation, exprimée par une fonction d'utilité globale. La faisabilité d'une structure de coalitions est conditionnée par la satisfaction de toutes les exigences, que ça soit celles définies au niveau des tâches ou celles définies pour des combinaisons de tâches.

En plus de la satisfaction de ces exigences qui représentent des contraintes dures, la fonction d'utilité, étant globale et portant sur l'ensemble de la structure de coalitions à la fois, exprime aussi les interdépendances des tâches et est maximisable. Elle doit être définie selon les besoins spécifiques de l'application et doit doit prendre en compte la globalité des tâches pour couvrir les leurs interdépendances plutôt que d'être simplement une somme d'utilités définies sur des tâches.

A.4 Algorithmes et Résultats

Dans ce chapitre, nous avons présenté l'approche que nous proposons pour résoudre la problématique de l'allocation des tâches interdépendantes.

L'objectif principal de notre travail est de trouver la meilleure structure de coalition faisable par rapport à l'utilité globale u_{global} si elle existe, ou de détecter la non-existence d'une structure de coalition faisable le plus tôt possible. Nous proposons une approche de solution décentralisée basée sur le passage de jetons entre les agents candidats des différentes coalitions qui seront affectés aux tâches composant la tâche globale.

Le processus est divisé en tours. A chaque tour, le jeton circule parmi les agents. Le tour se termine lorsque tous les agents ont reçu le jeton une fois. L'ordre du passage de token entre les agents dépend de l'application. Par exemple, dans une application avec une certaine hiérarchie, le jeton peut être envoyé du plus important, et donc peut-être le meilleur candidat de l'équipe, au moins important, celui qui a le moins de ressources et de qualités pour l'application. Dans une application placée dans l'espace, le jeton peut être transmis d'un agent à l'agent le plus proche.

Au départ, chaque agent connaît ses propres caractéristiques et la tâche globale T à accomplir. Les informations sur les autres agents et l'évolution de la formation de la structure de coalitions arrivent via le jeton échangé. Lorsque l'agent a_i envoie le jeton à l'agent a_j qui est son suivant, ce dernier reçoit avec le jeton des informations sur la meilleure structure de coalitions faisable (par rapport à u_{global}) S formée jusqu'à présent, l'expertise accumulée (*i.e.* les caractéristiques) X_S des agents participants à S, et le nombre d'agents nd qui n'ont pas changé de structure de coalitions (n'ont pas décidé de rejoindre une coalition).

Détenant le jeton, l'agent a_i peut décider de rejoindre (ou initier) une coalition C_{t_j} affectée à la tâche t_j, si elle peut contribuer à l'accomplissement de la tâche t_j ou si sa participation peut augmenter la fonction d'utilité globale u_{global}. Cette organisation est assurée par un algorithme global schématisé dans les figures A.1 et A.2. Le cas échéant, l'agent a_i met à jour les informations qu'il a reçues avec les modifications qu'il a appliquées. Ensuite, l'agent a_i passe le jeton à l'agent suivant. S'il n'a pas rejoint une coalition, il passe le jeton à l'agent suivant en transmettant simplement les informations qu'il a obtenues de l'agent précédent.

Nous supposons que les agents peuvent échanger des messages et nous décrivons le protocole de communication qui organise l'échange de messages entre agents (dans figure A.2). Pourtant, l'infrastructure de communication sous-jacente est au-delà de la portée de ce travail.

Notre approche repose principalement sur deux phases:

Phase I C'est la phase qui coïncide avec le premier tour. Dans ce tour, les agents ont comme objectif primaire de trouver une structure de coalitions S^f faisablie, si l'en existe.
Par conséquent, un agent ne rejoint une coalition que s'il peut satisfaire

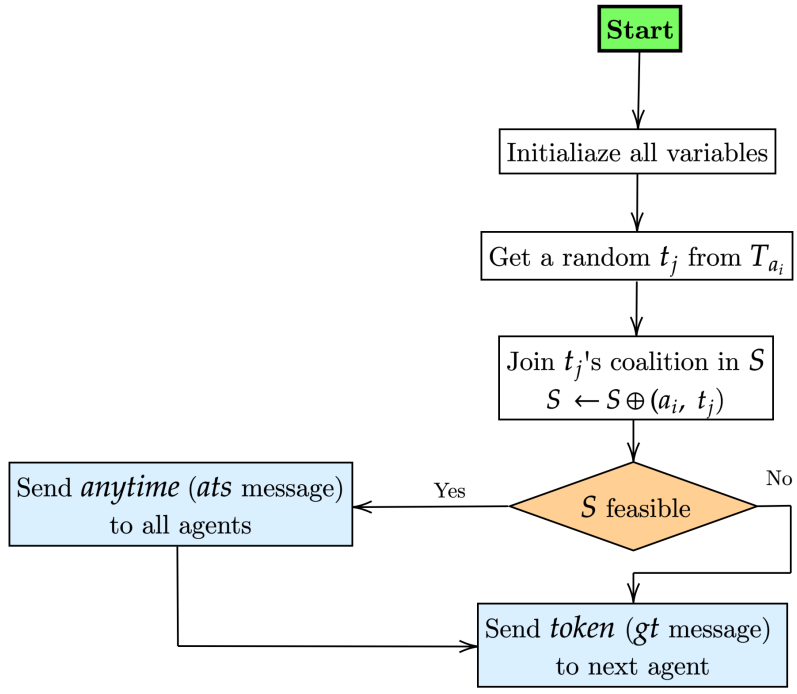


Figure A.1: Processus du premier agent

certaines exigences de tâches non encore satisfaites par d'autres membres de la coalition, quel que soit l'impact sur u_{global} (*i.e.* l'amélioration de la La valeur u_{global} n'est pas une condition préalable à ce stage). Si aucune S^f n'est trouvée à la phase I, on peut déclarer qu'aucune telle structure S^f n'existe compte tenu des exigences des tâches et des caractéristiques des agents disponibles.
Nous appelons la méthode implémentant cette étape FICSAM.

Phase II implémente notre deuxième méthode qui se focalise sur l'amélioration progressive de la structure de coalition faisable trouvée jusqu'à présent. Cette méthode commence à partir du deuxième tour. Une fois qu'une structure de coalitions faisable a été trouvée à la fin du premier tour (et par conséquent de la phase I), les agents essaient d'améliorer u_{global} via des remplacements ou des échanges entre un seul ou des groupes d'agents. La

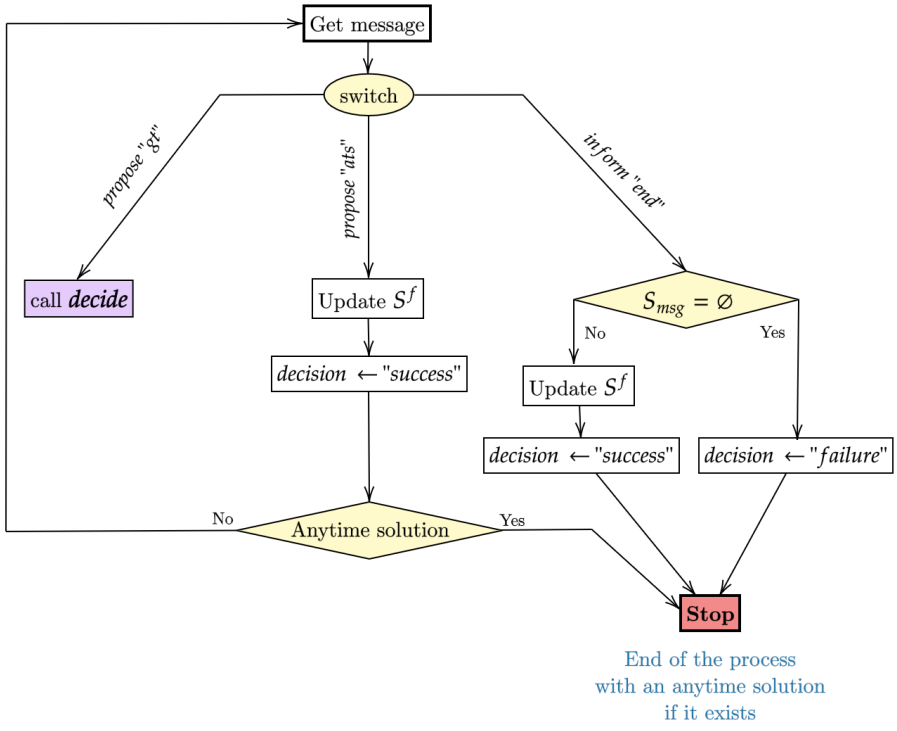


Figure A.2: Processus global

structure de coalitions améliorée qui en résulte doit préserver la faisabilité en respectant toutes les exigences des tâches et des combinaisons de tâches.

Nous appelons la méthode implémentant la première étape combinée à cette deuxième étape IFICSAM.

Figure A.3 schématise l'algorithme que nous proposons pour le processus de décision. En effet, quand un agent a_i reçoit un jeton d'un autre agent pendant le processus de formation de la coalition, ce processus est déclenché. La décision de l'agent dépend du contenu du message reçu et du tour R dans lequel se trouve le processus. Dans la première phase (*i.e.* premier tour), l'agent applique une méthode de formation de coalitions en utilisant ces connaissances accumulés sur les autres agents et ces connaissances sur les tâches pour générer une structure de coaltions. Pour ceci, il appelle la méthode *s-f-st*. Nous avons choisi d'utiliser

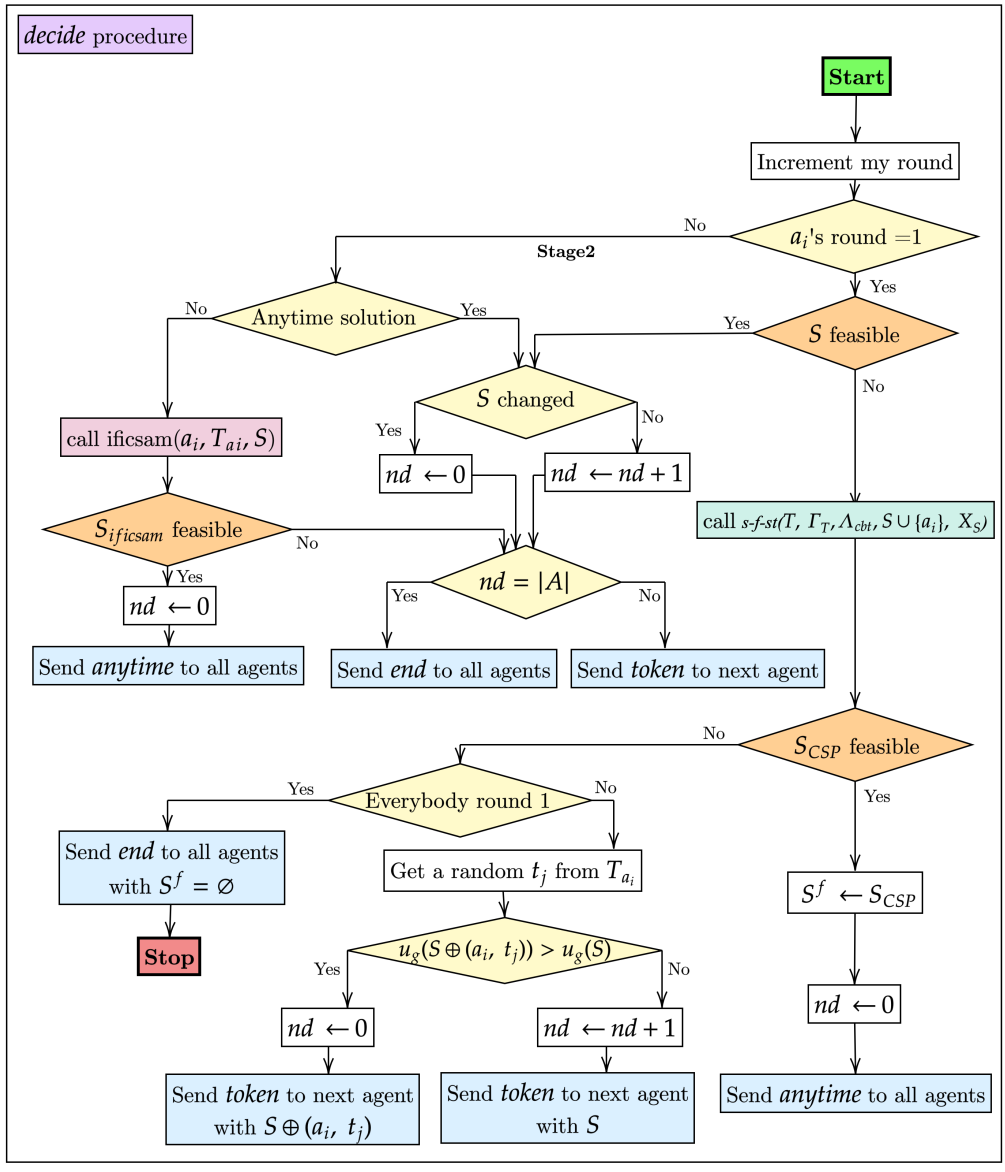


Figure A.3: Processus de décision

un solveur en modélisant ce problème par un CSP. Nous avons utilisé ce solveur CSP également pour vérifier la faisabilité des structures de coalition générées ou construites.

Dans la deuxième phase, et dans la quête d'autres structures de coalitions faisables avec des utilités plus grandes, nous proposons trois variantes différents. Dans la première, l'agent en possession du jeton essaie de rejoindre une coalition ou d'échanger sa place avec un agent dans une coalition et vérifie si un tel changement augmentera l'utilité globale. Dans la seconde, et pour explorer plus de structures de coalitions possibles, l'agent cherche à échanger les places des agents dans une des coalitions par un sous-groupe d'agents dont il appartient. Cette solution permet d'explorer plus de possibilités mais vu le nombre de vérifications à faire, n'est pas scalable. Pour ce, dans la troisième variante nous proposons de faire l'échange des agents d'une coalitions avec un certain sous-groupe d'agents, mais qui est choisi en utilisant un CSP qui repose sur des contraintes solides qui représentent les exigences de la tâche en question, mais aussi des préférences définies selon l'application. Cela permet de trouver la meilleur coalition pour la tâche.

Ces différentes variantes ont été testés sur un benchmark de scénarios. Pour ce, nous avons construit un générateur de scénarios basés sur un cas d'utilisation de drones de surveillance.

Les résultats empiriques montrent que nos algorithmes, notamment FICSAM, IFICSAM avec la première variante et IFICSAM avec la troisième variante prennent un temps raisonnable pour terminer. Les utilités, comparées aux utilités optimales, s'approchent de l'optimal pour les versions IFICSAM. La première variante a des résultats meilleurs lorsque la nature de l'application permet de faire des échanges sans risquer d'invalider les exigences des autres tâches. Tandis que la troisièle a de meilleurs résultats lorsqu'il y a dans l'application plus d'exigences qui ne peuvent être remplis que par une combinaison particulière d'agents.

A.5 Conclusion et Perspectives

A.5.1 Conclusion

L'objectif de cette thèse a été de contribuer à l'état de l'art du problème de l'allocation des tâches dans les systèmes multi-agents dans le cas des tâches

interdépendantes visant des applications du monde réel. Dans ces applications, il est récurrent d'opter pour des solutions décentralisées pour la prise de décision d'abord pour la nature des agents utilisés (robots, drones, ...) mais aussi en raison des contraintes environnementales et des besoins de robustesse. En plus, les tâches dans les applications du monde réel peuvent être interdépendantes sous différentes formes: partage des ressources, influence de l'exécution, contraintes d'ordre, synchronisation, ...

Une étude minutieuse et analytique de l'état de l'art du problème de l'allocation de tâches dans un contexte où les agents sont coopératifs et hétérogènes, a mené à la conclusion que les méthodes existantes supposent l'indépendance des tâches ou le cas échéant une interdépendance limitée à une interdépendance temporelle. Cependant, ayant besoin d'un algorithme pour les applications où les tâches peuvent exhiber une interdépendance plus générale, nous avons proposé une approche couvrant à la fois un algorithme d'allocation décentralisé avec différentes extensions et une définition formelle du problème en formulant les dépendances inter-tâches et en précisant la nuance entre une solution qui satisfait les tâches et une autre solution qui en plus en optimise la performance. Notre approche a été testée sur un bunch d'expérimentations produites avec un générateur de scénarios qui ont montré que nos méchanismes fournissent des solutions faisables et améliorables dans des temps acceptables tout en étant décentralisées et *anytime*. Ainsi, cette thèse couvre efficacement diverses questions visant à faire progresser les mécanismes de prise de décision des agents pour construire de manière robuste des équipes multi-agents.

A.5.2 Perspectives pour de futurs travaux

Notre approche et nos contributions, ayant comme objectif de contribuer à la nouvelle piste de recherche concernant les tâches interdépendantes, ont des limitations. Ainsi, plusieurs pistes se présentent pour les travaux futurs à la suite de ce travail. Premièrement, et afin de tester expérimentalement l'extension IFICSAM qui utilise les techniques B&B CSP pour chercher une solution avec une meilleure utilité, un cas d'utilisation avec une configuration où c'est avantageux de faire des échanges de plusieurs agents avec un agent est à développer. Ensuite, nous pourrons tester d'autres solveurs pour les CSPs appelés par nos algorithmes FICSAM et IFICSAM et faire l'analyse de leurs différents résultats. En particulier, nous aimerions implémenter le solveur OR-Tools, vainqueur du concours minizinc ces dernières années. Aussi, nous aimerions assouplir l'hypothèse du

statisme de l'environnement en rajoutant un déclencheur annonçant un changement dans l'environnement qui, lui, active une étape intermédiaire obligeant les agents à mettre à jour leur structure de coalition en fonction du changement qui se produit. Si l'algorithme, et comme on le prévoit, facilement dynamisable, il pourra s'appliquer à un éventail encore plus large des applications du monde réel. Dans notre approche, l'ordre de passage de jetons est prédéfini. Cependant, il serait intéressant d'étudier l'impact des changements de graphes de passage de jetons sur les structures de coalition résultantes. Enfin, nous aimerions examiner de la possibilité de parallélisation du passge de jeton, et ainsi de la prise de décision, dans des cas particuliers.

B ICTAI paper

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, multi-agent systems, task allocation, task interdependence, constraint programming**

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 optimal, and computation time is very sensible.

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. We 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. 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. We 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 $X_{S^f} \bowtie \Lambda_{cbt}$.*

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.

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

We use coalition structure modification functions:

- $\oplus : CS \times (A \times T) \to CS$, 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.
- $\ominus : CS \times (A \times T) \to CS$, 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.*** *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: $g_4(S) = \Sigma_{C_{t_j} \in S} (1$ if $|C_{t_j}| \geq \gamma_4(t_j)$ else $|C_{t_j}|/\gamma_4(t_j))$*
- *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.

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

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:

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

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

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. Agents communicate by exchanging messages following the communication protocol described in algorithm 3.

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 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 joins a coalition to contribute to feasible structures that can be found by agents that haven't received the token yet. 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 another task (and coalition). Agent a_i examines its contribution

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 ← S; R[i] ← R[i] + 1
2  if R[i] = 1 then
3      if not(is_f(S)) then
4          S_max ← S
5          S ← s-f-st(T, Γ_T, Λ_cbt, {a_i} ∪ S, a_i,
             X_{a_i} ∪ X_{S\{a_i}})
6          is_f(S) ← check-feasibility(T, Γ_T, Λ_cbt, A, S)
7          if is_f(S) then
8              S^f ← S; nd ← 0
9              S_max ← S^f
10             for C_{t_j} ∈ S do
11                 for a_l ∈ C_{t_j} do
12                     send(propose(a_i, a_l, ⟨"ats", S, X_S, R, nd⟩))
13         else
14             if R ≠ ⟨1, ..., 1⟩ then
15                 T_i ← ∅
16                 for t_j ∈ T s.t. X_{a_i} ⋈ Γ_{t_j} do
17                     T_i ← T_i ∪ {t_j}
18                 Get t_j ∈ T_i
19                 S ← S ⊕ (a_i, t_j)
20                 X_S ← X_S ∪ X_{a_i}
21                 if u_global(S) > u_global(S_max) then
22                     S_max ← S; nd ← 0
23                     send(propose(a_i, next(a_i), ⟨"gt", S, X_S,
                         R, nd⟩))
24                 nd ← 0
25                 send(propose(a_i, next(a_i), ⟨"gt", S, X_S,
                     R, nd⟩))
26             else
27                 (i.e. R = ⟨1, ..., 1⟩)
28                 S^f ← ⟨[], [], ..., []⟩
29                 for C_{t_j} ∈ S do
30                     for a_l ∈ C_{t_j} do
31                         send(inform(a_i, a_l, ⟨"end", ∅, ∅, 1⟩))
32 else
33     (i.e. R[i] > 1)
34     S^f ← S
35     call IFICSAM(a_i, T_i, S)
36     is_f(S) ← check-feasibility(T, Γ_T, Λ_cbt, A, S)
37     if is_f(S) and S ≠ S^f then
38         S^f ← S; nd ← 0
39         for t_j ∈ T do
40             for a_l ∈ C_{t_j} do
                 send(propose(a_i, a_l, ⟨"ats", S, X_S, R, nd⟩))
41     else S ← S_old
42 if nd < |A| and S = S_old then
43     nd ← nd + 1
44     send(propose(a_i, next(a_i), ⟨"gt", S, X_S, R, nd⟩))
45 else
46     for a_l ∈ A do
47         send(inform(a_i, a_l, ⟨"end", S, X_S, R⟩))
```

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 $X_{a_k} \not\bowtie \gamma_l(\tau(a_i, S))$ **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 24) 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-complete (one can show reduction from the 3-SAT problem). Hence, Algorithm 1 is NP-complete 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-complete 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 ← ∅; S ← ∅; R[i] ← R[i] + 1
    X_S ← ∅; T_i ← ∅; nd ← 0
    for t_j ∈ T s.t. X_{a_i} ⋈ Γ_{t_j} do
        T_i ← T_i ∪ {t_j}
    Get t_j ∈ T_i
    S ← S ⊕ (a_i, t_j)
    X_S ← X_S ∪ X_{a_i}
    is_f(S) ← check-feasibility(T, Γ_T, Λ_cbt, A, S)
    if is_f(S) then
        S_max ← S
        for a_l ∈ A do
            send(propose(a_i, a_l, ⟨"ats", S, X_S, R, nd⟩))
    else
        send(propose(a_i, next(a_i), ⟨"gt", S, X_S, R, nd⟩))
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, ⟨"ats", S, X_S, R, nd⟩) do
            if requirement_anytime_solution then
                decision ← "success"
                End of coalition formation process with a
                  feasible solution
            else
                is_f(S) ← true
                call decide(a_i, X_{a_i}, T, Γ_T, Λ_cbt, S,
                  X_S, R, nd, A, is_f(S))
        case propose(a_l, a_i, ⟨"gt", S, X_S, R, nd⟩) do
            is_f(S) ← false
            call decide(a_i, X_{a_i}, T, Γ_T, Λ_cbt, S, X_S,
              R, nd, A, is_f(S))
        case inform(a_l, a_i, ⟨"end", ∅, ∅, 1⟩) do
            decision ← "failure"
            End of the coalition process in first round with
              no feasible solution found
        case inform(a_l, a_i, ⟨"end", S, X_S, R⟩) do
            decision ← "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 $[G/2, G]$. 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 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 their 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 (sec)			Messages number	
		FICSAM	IFICSAM	Centralized	FICSAM	IFICSAM	Centralized	FICSAM	IFICSAM
5	2	0.58	0.73	0.83	0.9	1.4	0.2	18.4	25.9
10	2	0.59	0.79	0.85	1.6	3.0	0.2	36.8	71.7
10	5	0.55	0.70	0.78	1.7	2.9	48.0	38.0	61.8
20	2	0.53	0.73	0.85	3.6	6.8	0.2	71.5	177.2
20	5	0.53	0.75	0.86	3.4	6.1	0.2	71.8	170.7
20	10	0.53	0.70	0.79	42.0	43.9	1184.4	75.6	158.2
50	2	0.51	0.67	0.85	7.0	11.6	4.6	178.3	557.6
50	5	0.49	0.67	0.86	69.1	76.4	0.3	176.0	576.3
50	10	0.54	0.71	0.86	387.6	398.4	25.1	181.7	526.8
50	20	0.44	0.61	0.80	212.6	222.2	1195.7	180.3	511.7
100	2	0.48	0.62	0.85	70.2	83.4	1.8	351.3	1349.9
100	5	0.48	0.61	0.86	189.7	209.3	1.2	349.7	1204.1
100	10	0.48	0.63	0.86	446.5	468.9	32.0	350.6	1419.5
100	20	0.53	0.65	0.83	1043.9	1073.8	1158.6	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. For the largest instance (100 agents, 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 drop in the number of exchanged messages when we have more tasks for the same number of agents. In our scenarios, 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 might produce 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 PFGs [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 the problem we are concerned with in this paper. 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).

Acknowledgments

The authors thank Christian Bessiere, Yannis Dimopoulos and Alexis Tsoukias for the fruitful discussions.

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. *Autom. Constr.*, 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. *AAMAS*, 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, 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 *CP'07*, pages 529–543. Springer, 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. *AIJ*, 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. *AIJ*, 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. *AIJ*, 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.