Distributed decision-making in multi-UAV systems : exploring methods, rewards tuning, and operating mode adaptation

Mohand Hamadouche

COLLEGE MATHSTIC
DOCTORAL BRETAGNE
BRETAGNE OCEANE

Université de Bretagne Occidentale

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE BRETAGNE OCCIDENTALE

ÉCOLE DOCTORALE N° 644
Mathématiques et Sciences et Technologies de l'Information et de la Communication en Bretagne Océane
Spécialité : *Informatique et architectures numériques*

Par
Mohand HAMADOUCHE

Distributed Decision-Making in Multi-UAV Systems : Exploring Methods, Rewards Tuning, and Operating Mode Adaptation

Thèse présentée et soutenue à Brest, France, le 29 Février 2024
Unité de recherche : Lab-STICC (UMR, CNRS 6285), UBO, Brest - France

Rapporteurs avant soutenance :

Stéphane MOCANU	Maître de Conférences HDR, Université Grenoble Alpes
Nathalie MITTON	Directrice de recherche, INRIA

Composition du Jury :

Président :	Jean-Philippe BABAU	Professeur des Universités, Université de Bretagne Occidentale
Examinateurs :	David ESPES	Professeur des Universités, Université de Bretagne Occidentale
	Káthia DE OLIVEIRA	Professeur des Universités, Université Polytechnique Hauts-de-France
	Stéphane MOCANU	Maître de Conférences HDR, Université Grenoble Alpes
	Nathalie MITTON	Directrice de recherche, INRIA
Dir. de thèse :	Catherine DEZAN	Maître de Conférences HDR, Université de Bretagne Occidentale
Co-dir. de thèse :	Kalinka BRANCO	Maître de Conférences, Université de São Paulo (USP), Brésil

À ma famille

Résumé Long (French summary)

❑ Chapitre 1 : introduction

Le 21ème siècle a été le témoin d'avancées significatives, en particulier dans le domaine de la robotique, alimentées par l'évolution continue de l'électronique et des capacités informatiques. Les véhicules aériens sans pilote (UAV) sont devenus des acteurs essentiels dans les environnements où l'intervention humaine est difficile, répétitive ou dangereuse. Les UAVs sont largement utilisés dans divers domaines tels que la sécurité [Lee+21], l'inspection et la photogrammétrie [Gha+21], la cartographie ou les opérations de vidéo-surveillance [Mal+21], l'agriculture [Rad+20], les opérations internes de nombreux entrepôts et plateformes logistiques [Che+20], les missions de recherche et de sauvetage [AKY21], les missions militaires [CAM20], etc. En particulier, les UAVs se sont révélés être des outils polyvalents pour relever les défis posés par la crise du coronavirus [Res22].

Cependant, compte tenu du contexte géographique du financement de la thèse, un accent particulier a été mis sur les missions situées dans l'environnement maritime. De plus, l'aspect coopératif de l'utilisation des drones maritimes s'étend à des rôles innovants, tels que l'utilisation de drones maritimes en tant que bateaux-mères. Ces bateaux-mères peuvent servir de stations pour recharger les UAVs lorsque leurs batteries sont faibles, prolongeant ainsi leur autonomie opérationnelle.

Néanmoins, dans le cadre de l'exécution de ces missions, l'accent géographique introduit des défis distincts. Un exemple notable est la vulnérabilité d'un composant critique (par exemple, le GPS) aux défaillances. Ces défaillances peuvent être dues à des facteurs tels qu'une couverture satellitaire inadéquate ou des interférences délibérées par le biais d'attaques de spoofing ou de jamming.

Pour relever ces défis, il est nécessaire de mener des recherches scientifiques permanentes et de réaliser des avancées technologiques afin de renforcer ces systèmes aériens et de garantir leur adaptabilité et leurs performances dans le cadre de missions maritimes complexes. Cette approche interdisciplinaire est essentielle pour faire évoluer les capacités des UAVs aériens et optimiser leur contribution à l'exploration maritime et aux efforts de résolution des problèmes.

❑ Positionnement scientifique

Les travaux de thèse de Chabha Hireche [Cha19] ont exploré les modèles de décision utilisés dans le cadre d'une mission de drone. Ils ont confirmé que les modèles probabilistes, en particulier les Processus de Décision Markoviens (MDP), sont largement adoptés pour la prise de décision sous incertitude en robotique.

Pour compléter cet axe, la première problématique de cette thèse part de cette constatation et répond à la question suivante : **quelle est la méthode la plus adaptée pour résoudre un problème de décision sous incertitudes en fonction du contexte d'une mission dans le cadre embarqué ?**

Cependant, dans diverses applications, les limitations énergétiques, de capacité informatique et d'exécution rendent souvent un seul drone (UAV) inadéquat pour une couverture étendue. Les systèmes multi-drones (UAV), tels que les essaims, visent à obtenir une couverture plus large, une surveillance améliorée et une exécution plus efficace des missions par une action coopérative pour recueillir rapidement et précisément des informations.

La figure 1 schématise les autres problématiques scientifiques auxquelles cette thèse se propose d'apporter des réponses.

Figure 1 – Mission collaborative entre drones aériens et maritimes.

Au sein d'un système multi-UAV, les décisions peuvent être en conflit et s'écarter des objectifs de la mission, nécessitant ainsi un mécanisme de gestion des conflits. Cette observation nous amène à une deuxième problématique : **quels sont ces conflits et comment les résoudre dans un cadre d'une mission multi-UAV modélisé à l'aide de MDPs ?**

Enfin, dans le cadre des missions collaboratives avec des systèmes multi-UAV, une

troisième et dernière problématique se pose qui est la suivante : **comment peut-on s'assurer qu'un drone peut réaliser sa mission en complète autonomie ? comment mettre en œuvre la résolution de conflits au sein d'un système multi-UAV tout en minimisant les communications via le cloud et en réduisant le volume de données à transmettre de façon à favoriser au maximum l'autonomie des drones ?**

Avant d'entamer l'exploration détaillée de la résolution de conflits dans un cadre d'une mission multi-UAV, un état des lieux approfondi de la prise de décision distribuée pour les systèmes de véhicules aériens multi-UAV sous incertitudes est présenté.

❑ Chapitre 2 : état des lieux de la prise de décision distribuée pour les systèmes de véhicules aériens multi-UAV sous incertitudes

Dans la première partie de ce chapitre, nous avons présenté une vue d'ensemble des drones, les classifiant en fonction de leur poids, altitude et endurance. La deuxième partie a principalement énuméré les avantages de l'utilisation d'un système multi-UAV, comprenant notamment la parallélisme, la rentabilité, l'efficacité temporelle, l'efficacité énergétique, la complémentarité et la scalabilité. La troisième partie a approfondi cet aspect en explorant les applications des systèmes multi-drones utilisés au cours de la dernière décennie. Cela a conduit à une expansion significative des missions des drones vers des domaines diversifiés, englobant la surveillance vidéo, l'inspection et le suivi, les réseaux, la photogrammétrie et la SLAM, ainsi que la recherche et le sauvetage. Cette diversification témoigne de l'ampleur des opportunités offertes par ces systèmes. La quatrième partie propose d'abord une taxonomie pour ces systèmes, puis présente une revue de l'état de l'art des modèles probabilistes de prise de décision utilisés dans les systèmes multi-drones. Ces travaux sont ensuite comparés en fonction de caractéristiques essentielles telles que les études de cas, l'organisation collective, l'homogénéité, le type de contrôle de mission, le type de communication, les données partagées, l'interdépendance entre les membres du système, le modèle de prise de décision, le type de tâches, l'environnement de validation, l'extensibilité du système et les critères utilisés pour mesurer la pertinence des publications.

En examinant les questions ouvertes dans ce domaine, nous avons identifié plusieurs points cruciaux :

- les conflits potentiels de gestion entre les membres du système multi-UAV deviennent une préoccupation croissante. Nous avons noté que les conflits peuvent être internes à chaque membre en raison du non-respect des contraintes externes, ou ils peuvent survenir au niveau du système multi-UAV. Les conflits internes peuvent résulter

d'actions incompatibles (par exemple, lorsqu'un UAV1 pénètre dans une zone non autorisée alors qu'un UAV2 a accès à cette zone), etc. Les conflits au niveau du système multi-UAV peuvent apparaître lorsque les contraintes de mission ne sont pas respectées, par exemple, le nombre d'UAV exécutant la tâche est dépassé, etc.

- les modèles de prise de décision doivent intégrer pannes matérielles et logicielles des systèmes embarqués (capteurs, autopilote, processeur embarqué).
- afin de favoriser l'autonomie des drones, il est important de réduire les interactions entre les UAV au niveau du système multi-UAV en échangeant le moins possible de données. Cette réduction des données transmises peut réduire, en même temps, le risque potentiel de transmission incorrecte ainsi que le nombre de ressources et la consommation d'énergie due à la communication.

Pour finir, il est suggéré dans ce chapitre qu'une solution future pourrait consister à gérer une mission avec un modèle décisionnel probabiliste distribué pour un système multi-UAV, alimenté par des modèles de prédiction de défaillance et de détection des cyber-menaces et pouvant interagir les différents acteurs de la mission. Un échange minimal de données (signal de défaillance du capteur, signal d'une cyber-menace, cible verrouillée pour le suivi, tâche prise, etc.) peut permettre une bonne adaptation et synchronisation entre les membres du système multi-UAV, réduisant ainsi la consommation de ressources matérielles et énergétiques pendant les missions.

❑ Chapitre 3 : contribution 1 - comparaison des méthodes fondamentales de planification de mission

Dans la littérature, il existe plusieurs modèles probabilistes pour la prise de décision sous incertitudes, tels que les processus de décision de Markov (MDP) [HDB20; Han+19], les processus de décision de Markov partiellement observable (POMDP) [CMO16], les langages de diagramme d'influence dynamique relationnel (RDDL) [YWW20], les réseaux de Petri (PN) [RCB15], etc. La description sous la forme de MDPs est généralement adopté en robotique pour la prise de décision et la planification sous incertitudes.

Ce travail s'est concentré sur l'approche MDP et ses méthodes de résolution. Il existe trois classes fondamentales de méthodes pour la résolution de problèmes de décision de Markov finis [SB98] qui sont la programmation dynamique (DP), les méthodes de Monte Carlo (MC) et l'apprentissage par différence temporelle (TD). Chacune de ces classes a ses avantages et ses inconvénients.

Les méthodes de Programmation Dynamique (DP) sont bien développées sur le plan mathématiques mais nécessitent un modèle complet de l'environnement [Bel66]. Les méthodes de Monte Carlo (MC) sont conceptuellement plus simples et ne nécessitent pas de

modèle complet de l'environnement. Cependant, elles ne conviennent pas pour une estimation de politique précise et efficace en raison de leur approche non incrémentale [MU49]. Enfin, les méthodes d'apprentissage par différence temporelle (TD) ne nécessitent pas de modèle. Elles sont entièrement incrémentales mais sont plus complexes à analyser et à ajuster. Les approches diffèrent également sur plusieurs aspects concernant leur efficacité et leur vitesse de convergence [Tes95]. Les méthodes de Monte Carlo (MC) ne sont pas très utiles en robotique car, théoriquement, elles ne nécessitent que de l'expérience (sous forme d'épisodes d'échantillons) des états, des actions et des récompenses résultant d'une interaction réelle avec un environnement. Ainsi, la politique apprise à partir de l'expérience réelle ne peut pas prendre en compte les décisions futures de l'agent.

Après un rappel sur les modèles de probabilistes utilisé dans la prise de décision, une analyse des méthodes de résolution en termes de complexité est présentée.

❖ **Processus de décision de Markov**

Un processus de décision de Markov (MDP) discret [Put14] modélise un processus dynamique, l'agent observant l'environnement à chaque étape. Il choisit une action, exécute l'action choisie (qui modifie l'environnement), et l'agent reçoit une récompense en fonction du changement. Un MDP est formellement défini comme un **tuple** $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ où $\mathcal{S}$ représente un ensemble fini d'états, $\mathcal{A}$ est un ensemble fini d'actions, $\mathcal{T}(s, a, s')$ est la probabilité [0,1] entre deux états, et $\mathcal{R}(s, a)$ correspond à la récompense immédiate reçue après l'exécution de l'action a_1 dans l'état s

❖ Une **politique** π $\mathcal{S} \times \mathbb{N} \mapsto \mathcal{A}$ (où $\mathbb{N}$ représente le nombre d'états) est une solution d'un MDP et constitue un plan de décision qui détermine pour un agent à chaque état s de $\mathcal{S}$, l'action a de $\mathcal{A}$ qui doit être effectuée.

La politique optimale (π^*) est généralement celle qui maximise les récompenses cumulées attendues, V_π, de manière à satisfaire l'équation de Bellman[Bel58], définie par :

$$\pi^* = \arg\max_\pi V_\pi(s), \quad \forall s \in \mathcal{S}. \tag{1}$$

$$V_\pi(s) = \max_a \{\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') V_\pi(s')\} \tag{2}$$

La fonction optimale **Q-function** $Q^*(s, a)$ est la récompense totale attendue par un agent qui commence en s et qui choisit une action a se comportera de manière optimale par la suite, définie par :

$$Q^*(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') V^*(s'). \tag{3}$$

Il est donc possible de calculer la politique la plus efficace par l'équation suivante:

$$\pi^* = arg\ \max_a Q^*(s,a) \quad \forall s \in \mathcal{S}. \tag{4}$$

❖ **Comparaison de complexité des trois méthodes**

Le Tableau 1 résume les données d'entrée des trois méthodes et présente leur comparaison de complexité.

		Méthode		
		Valeur Iteration	Policy Iteration	Q-Learning
Données d'entrée	Matrice T	✓	✓	-
	Matrice R	✓	✓	✓
	γ	✓	✓	✓
	α	-	-	✓
	ϵ	✓	-	✓
	ϵ_{decay}	-	-	✓
	$Iter_{max}$	✓	✓	✓
Opérateur de la boucle interne		Bellman*	Bellman*	Max
Nombre de boucles		1	1	2
Condition d'arrêt	$iter = iter_{max}$	✓	✓	✓
	Convergence	✓	✓	-

Table 1 – Résumé des données d'entrée et comparaison de complexité pour les trois méthodes [Ham+21].

Tous ces paramètres et données d'entrée influent sur les performances et le temps de résolution de chaque méthode. De plus, il n'existe aucune méthode permettant de trouver facilement les meilleures valeurs pour ajuster ces paramètres pour une application donnée.

❖ **Résumé de la comparaison des trois méthodes**

Comme le montrent les deux figures 2-a et 2-b, la recherche d'une politique optimale globale (version 1) pour un problème est plus pratique en utilisant les méthodes DP (méthode Value Iteration ou méthode Policy Iteration). La méthode Q-Learning est inefficace en termes de temps d'exécution car elle nécessite beaucoup de temps pour explorer l'espace d'état et d'action avant de converger. Ainsi, pour ce type de scénario simple, la méthode Policy Iteration est le meilleur candidat pour trouver le meilleur ensemble d'actions permettant d'atteindre l'état cible à partir de n'importe quel état.

Comme me montre la figure 2-c, la méthode Q-Learning est plus efficace que les méthodes Dynamic Programming (méthode Value Iteration ou méthode Policy Iteration) pour trouver un chemin entre l'*état de départ* et l'*état cible*.

❖ Les travaux de ce chapitre se sont principalement focalisés sur trois méthodes de résolution des MDPs : Les méthodes Value Iteration, Value Iteration (DP) et la méthode Q-Learning (TD). nous proposons de nouveaux critères pour adapter la méthode de prise de décision au problème d'application , soulignant que cette adaptation est particulièrement pertinente dans le contexte de l'embarqué. A travers des expériences expérimentales,

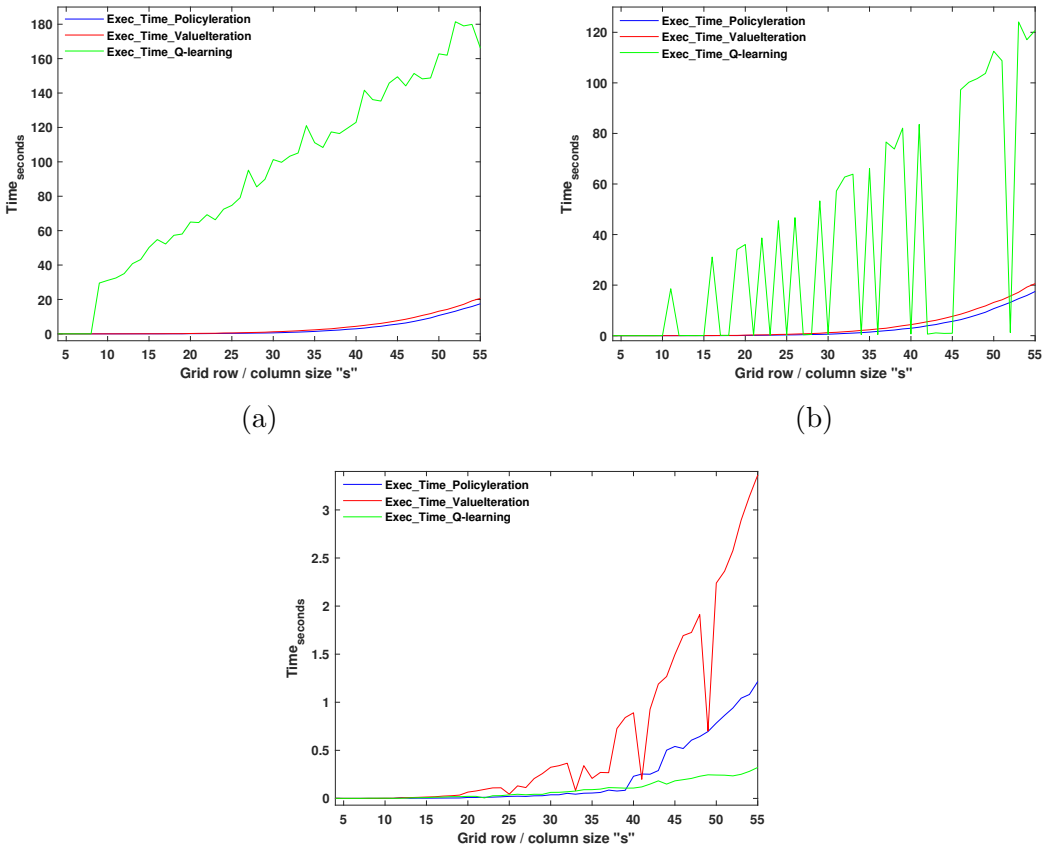


Figure 2 – Comparaison de Value Iteration, Policy Iteration et du Q-Learning [Ham+21]. (a) temps d'exécution en version 1-a pour une taille de 4×4 à 55×55. (b) temps d'exécution en version 1-b pour une taille de 4 × 4 à 55 × 55. (c) Comparaison version 2.

ce travail démontre que la méthode Q-Learning est intéressante dans les cas simples et réguliers et démontre que dans les cas irréguliers, les méthodes classiques de résolution de MDP sont plus appropriées à mettre en œuvre sur des systèmes critiques.

Cette contribution a fait l'objet d'une publication à la conférence IEEE ICUAS 2021 [Ham+21].

❑ Chapitre 4 : contribution 2 - réglage des récompenses pour l'auto-adaptation de politiques dans la planification de missions

L'approche MDP est l'un des meilleurs candidats pour gérer la prise de décision et la planification en situation d'incertitude, ce qui est courant dans l'environnement des UAVs. Elle se compose d'un agent et d'un environnement. L'agent interagit avec l'environnement en observant les transitions des états et en recevant des récompenses pour avoir trouvé une action optimale dans chaque état. Un changement dans la situation de l'agent ou

dans l'environnement entraîne des modifications dans les probabilités de transition et/ou les récompenses associées aux actions. Ces modifications impactent l'action optimale, qui n'est pas immédiatement prise en compte dans d'autres approches d'apprentissage par renforcement. Cela est dû au fait que l'action optimale nécessite un temps d'apprentissage pour s'ajuster aux nouvelles conditions. En revanche, les MDP utilisent les probabilités pour accélérer la résolution des problèmes en anticipant et en incorporant ces changements dans leur modèle. En outre, les MDP peuvent inclure des contraintes physiques dans leur modèle. Ils prennent en compte les exigences de sécurité de la mission, ajoutant ainsi une dimension cruciale à la planification des actions. La résolution de conflits est également intégrée dans le moteur de prise de décision des MDP. Cette approche globale améliore la sécurité, la précision et l'efficacité des missions en permettant au système d'ajuster dynamiquement ses actions en fonction de diverses contraintes et exigences spécifiques.

L'objectif consiste à exploiter les paramètres intrinsèques définissant le MDP, en mettant l'accent sur les récompenses associées aux actions. L'expérimentation a révélé que l'ajustement des récompenses associées aux actions au niveau de l'état a un impact significatif sur la politique résultante du problème modélisé. En conséquence, cette adaptation des récompenses permet de mieux aligner la mission sur les contraintes physiques, les exigences de sécurité et la résolution de conflits intégrées dans le moteur de prise de décision des MDP. Ainsi, en optimisant les récompenses, le système peut supprimer efficacement les conflits et améliorer la performance globale de la mission.

La méthode proposée peut s'appliquer à deux contextes, le contexte multi-UAV (2-a) et le contexte d'un seul UAV, comme suit.

1. **Contribution 2-a : application de la méthode dans un contexte multi-UAV**

L'un des défis du développement des véhicules autonomes est de définir un moteur décisionnel approprié capable de traiter une variété de problèmes de contrôle [EMA20; AK20; Wan+20; FC18; Bar19] afin d'augmenter leur niveau d'autonomie et de sécurité.

Pour surmonter les défis inhérents à la gestion des véhicules autonomes au sein d'un système multi-UAV, nous avons développé une nouvelle approche. Cette approche vise à résoudre les conflits émergents dans un environnement où plusieurs MDP sont simultanément en cours d'exécution. La résolution des conflits se fait par des ajustements dynamiques des plans de mission individuels des UAVs, notamment en modifiant les récompenses associées aux actions conflictuelles du système multi-UAV. De plus, notre approche propose une méthode systématique en ligne permettant une adaptation dynamique des récompenses en temps réel, avec un partage minimal des données entre les UAVs. En outre, elle intègre une dimension énergétique, où les actions des UAVs prennent en compte leur niveau d'énergie, considéré comme une contrainte interne pouvant influencer le déroule-

ment des actions à entreprendre par un UAV. Ces innovations contribuent à optimiser la coordination et la performance du système multi-UAV, tout en garantissant la réalisation des objectifs de la mission.

1.1. **Détection des conflits et des violations de contraintes**

(a) **Conflits et contraintes**

Pour aborder les défis complexes inhérents à la gestion des véhicules autonomes au sein d'un système multi-UAV, notre approche novatrice se concentre sur la résolution des conflits et l'adaptation dynamique des récompenses, tout en prenant en compte des aspects pratiques. Un aspect crucial de cette approche est la détection proactive des conflits et des violations de contraintes au sein de l'essaim de UAVs.

Lorsqu'une prise de décision distribuée est effectuée entre les membres de l'essaim, des conflits peuvent émerger en raison du non-respect des contraintes, que ce soit au niveau individuel ou au niveau de l'essaim dans son ensemble. Ces conflits peuvent être variés, comprenant des défaillances matérielles, des limitations de performance, des conflits de trajectoire, des conflits externes et des conflits de mission. Par exemple, des défaillances matérielles peuvent conduire à des divergences entre les plans de mission, tandis que des limitations de performance peuvent générer des conflits liés à la capacité de réaliser certaines tâches. De même, des conflits externes peuvent survenir en raison d'interactions avec des entités extérieures au système multi-UAV. Ces scénarios détaillés dans la Section 4.1.1 du chapitre 4 illustrent la complexité des situations rencontrées dans un environnement multi-UAV.

(b) **Priorité entre les UAVs**

Pour gérer efficacement les conflits, il est essentiel d'identifier les agents (UAVs) qui peuvent maintenir leur plan de mission et ceux qui doivent en changer. Pour ce faire, nous pouvons définir une priorité entre les UAVs de l'essaim. Nous pouvons distinguer deux types de priorité. La priorité statique, définie hors ligne par l'expert, reste constante pendant la mission. En revanche, la priorité dynamique permet des ajustements en fonction de seuils définis par l'expert sur des probabilités de transition ou d'autres paramètres. Les membres de l'essaim échangent des données pendant la mission pour déterminer la priorité de chaque MDP selon des critères spécifiques.

Dans des missions opérationnelles réelles, l'utilisation de la priorité dynamique des UAVs devient particulièrement pertinente en raison des disparités de performances entre les membres de l'essaim. Au fil de l'évolution de la mission, ces performances peuvent subir des changements aléatoires, influencés par les aléas de l'environnement extérieur. Ainsi, la prise en compte de cette priorité dynamique permet d'ajuster efficacement la répartition des responsabilités au sein de l'essaim.

(c) **Gestion des conflits (*Check_Conflicts*)**

Les conflits sont vérifiés à deux niveaux : la vérification des conflits internes à UAV et la vérification des conflits au niveau de l'essaim. Le premier type de gestion des conflits permet à UAV de se gérer lui-même sans intervention humaine pendant sa mission. Le deuxième type de gestion des conflits permet à l'UAV de se comporter de manière appropriée dans une situation qui survient dans l'essaim. Dans ce cas, le conflit est résolu en attribuant de nouvelles priorités aux actions des UAVs. Nous supposons ici qu'une autorité centrale attribue des priorités dans le cloud.

Chaque UAV calcule sa politique et vérifie périodiquement les conflits, comme illustré dans la Figure 3. Si le conflit est interne à UAV, ce dernier adaptera sa mission pour le

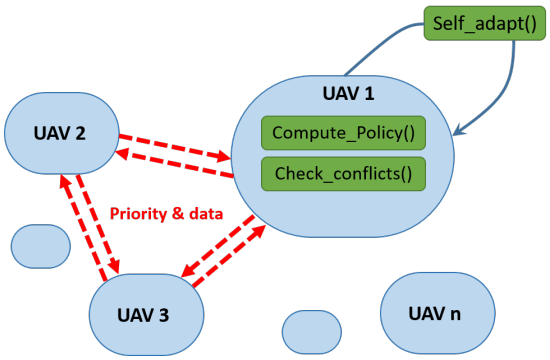


Figure 3 – Diagramme de l'auto-adaptation de politique entre les UAVs de l'essaim [Ham+21].

résoudre. Si le conflit concerne l'un des membres de l'essaim et que celui-ci est incapable de le résoudre, il transmettra la priorité à l'un des membres qui prendra le relais pour accomplir la mission.

1.2. **Résolution des conflits et des violations de contraintes**

Après avoir identifié les contraintes/conflits, une priorité est attribuée à l'UAV. Nous pouvons sélectionner le MDP dont la politique doit être modifiée en fonction de cette priorité.

(a) **Principes d'adaptation des récompenses**

Cette méthode oblige le MDP à modifier sa politique en raison des conflits d'action. Comme chaque valeur de la matrice Q est calculée à l'aide de l'équation (3), la valeur Q de l'action prioritaire est augmentée. Par conséquent, la politique est modifiée pour éviter les violations de contraintes, et certaines actions sont prioritaires par rapport à d'autres.

La politique est obtenue à partir de la matrice Q^* en utilisant l'équation (4)

$$\mathcal{Q}^* = \begin{pmatrix} \mathcal{Q}_{s_1,a_1} & \mathcal{Q}_{s_1,a_2} & . & . & \mathcal{Q}_{s_1,a_m} \\ . & . & \mathcal{Q}_{s_i,a_\sigma} & . & . \\ \mathcal{Q}_{s_n,a_1} & . & . & . & \mathcal{Q}_{s_n,a_m} \end{pmatrix}.$$

En cas de conflit avec l'action a_σ associée à l'état s_i, l'expert choisit l'action a_ϕ comme alternative, parmi les actions possibles de cet état. Cette substitution respecte les contraintes du système sans générer de nouveaux conflits.

En utilisant l'equation (3), nous avons dérivé l'équation permettant de calculer la nouvelle récompense comme suit :

$$\begin{aligned} \mathcal{R}(s_i, a_\phi) &> \mathcal{Q}^*_{s_i,a_\sigma} - \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s_i, a_\phi, s') V^*(s'), \\ \mathcal{R}(s_i, a_\phi) &= \lceil \mathcal{Q}^*_{s_i,a_\sigma} - \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s_i, a_\phi, s') V^*(s') \rceil. \end{aligned} \tag{5}$$

La méthode systématique d'ajustement des récompenses proposée est employée pour examiner tous les états d'une politique. En cas de découverte d'un état conflictuel, une nouvelle récompense est calculée. Enfin, la nouvelle politique est recalculée en utilisant la nouvelle récompense.

(b) **Auto-adaptation**

La méthode d'auto-adaptation est utilisée pour adapter la mission ou la tâche de l'UAV afin de supprimer le conflit entre ce UAV et l'autre membre de l'essaim. Il existe plusieurs scénarios pour résoudre ce conflit.

Les états conflictuels sont marqués dans chaque scénario en fonction de l'action alternative et des paramètres du scénario. Ensuite, l'action alternative , le MDP et ses états conflictuels sont transmis en tant que paramètres à la méthode de résolution de conflit basé sur l'adaptation de récompenses. En cas de conflit, la méthode utilisera la priorité (si reçue) et la tâche alternative fournie par le cloud pour adapter automatiquement la mission de l'UAV à la mission de l'essaim. En l'absence de réception de la priorité, la méthode utilisera la tâche alternative locale pour adapter automatiquement la mission de l'UAV à la mission de l'essaim.

Cette contribution a fait l'objet d'une publication lors de la conférence IEEE ICUAS 2023 [Ham+23].

2. **Contribution 2-b : application de la méthode dans un contexte d'un seul UAV**

Dans le contexte d'une mission individuelle d'un seul UAV, face à l'accroissement de la complexité des missions, il peut etre intéressant d'exprimer le problème de mission en utilisant plusieurs MDP fonctionnant en parallèle [Hir+18]. Cette décomposition implique la prise de décisions simultanées entre différents agents à l'intérieur du même UAV. Cependant, l'exécution d'actions concurrentes peut générer des conflits potentiels. De plus, la survenue de problèmes liés aux défaillances du système et des capteurs durant la mission peut entraîner des conséquences négatives, telles que le crash de l'UAV causé par une défaillance de la batterie.

On peut appliquer la méthode d'adaptation des récompenses de la manière suivante.

2.1. **Détection des conflits au cours d'une mission d'un seul UAV gérée au moyen de plusieurs MDP parallèles**

La gestion d'une mission individuelle d'un UAV à travers plusieurs MDP parallèles souligne l'importance de déceler d'éventuels conflits. Ainsi, la prochaine phase de notre étude approfondira la compréhension des challenges liés à cette gestion complexe, se penchant particulièrement sur la nature des conflits, des contraintes et de la priorité entre politiques susceptibles de se manifester.

Conflits et contraintes

On peut distinguer deux types de contraintes : la contrainte associée au conflit de comportement résultant d'actions antagonistes qui ne peuvent se produire simultanément, et la contrainte liée à l'état de santé du système ou d'un capteur.

Priorité entre politiques

Il est nécessaire de déterminer quel agent conserve sa politique et lequel doit la modifier. Nous avons établi une priorité entre les politiques des MDPs, où le MDP ayant la plus haute priorité conserve sa politique. Il existe deux types de priorité : statique, définie hors ligne par l'expert, et dynamique, ajustée en ligne en fonction de seuils fixés par l'expert.

2.2. **Résolution de conflits dans un seul UAV géré par plusieurs MDP parallèles** Dans le cas de plusieurs MDP, nous proposons les étapes suivantes :

- Étape 1 : nous calculons la politique pour chaque MDP (par exemple, en utilisant la méthode Policy Iteration).
- Étape 2 : nous vérifions si la contrainte est respectée ou s'il y a un conflit entre les politiques (voir l'algorithme 8 de la sous-section 4.2.2). Cet algorithme renvoie les variables booléennes indiquant les politiques des MDP contenant des actions en conflit et les états impliqués.
- Étape 3 : si la contrainte est respectée ou si des conflits sont détectés à l'étape 2, nous résolvons les conflits en modifiant une ou plusieurs politiques en fonction de leur priorité (voir l'algorithme 7). Cet algorithme utilise l'algorithme 5 pour augmenter la récompense de l'action souhaitée exécuter dans la politique ayant la priorité la plus basse.

Cette contribution a fait l'objet d'une publication dans un workshop de la conférence IEEE/IFIP DSN-W 2020 [HDB20].

3. **Simulation sur la plate-forme logicielle de simulation de robot Coppeliasim et MATLAB.**

Pour démontrer l'efficacité de la méthode d'adaptation des récompenses pour la gestion des contraintes, nous avons effectué une simulation sur CoppeliaSim et MATLAB. CoppeliaSim offre un environnement complet pour la conception, la simulation et le test

de systèmes robotiques. Les utilisateurs peuvent implémenter le contrôle via des scripts intégrés, des plugins, des nœuds ROS ou des clients API distants. Nous avons décidé de modéliser la mission considérée dans cette partie en utilisant Matlab, car tous nos algorithmes et évaluations sont écrits en Matlab. La simulation Matlab reçoit des données de CoppeliaSim via des fonctions API distantes.

❖ **Interface GUI Matlab**

Une interface à été conçue pour faciliter la gestion des missions d'inspection des bâtiments. L'interface comprend deux boutons distincts permettant de sélectionner les types de simulation, en particulier les simulations impliquant une défaillance de la batterie et celles sans défaillance.

L'interface comprend deux tables, chacune correspondant aux politiques des UAVs individuels. Les tables se mettent à jour dynamiquement. Lors du lancement de la mission, les modifications en temps réel des politiques des UAVs sont reflétées dans les tables affichées, fournissant une représentation en direct de l'évolution de la mission et des événements pertinents.

❖ **Scène CoppeliaSim**

Une mission d'inspection de bâtiments avec deux UAVs a été mise en place dans CoppeliaSim. Chaque UAV a une application distincte qui influence la méthodologie d'inspection. Les bâtiments à inspecter varient en dimensions.

Deux scénarios sont envisagés :

UAVs sans panne : les UAVs suivent un chemin prédéfini, surmontant des obstacles comme des arbres et des poteaux. UAV avec panne : ce scénario explore une panne de batterie. L'UAV affecté ajuste sa mission et signale à son voisin de prendre le relais de l'inspection. L'UAV voisin s'adapte en incorporant les coordonnées transmises et les fonctions d'inspection pour le bâtiment prioritaire. L'inspection du bâtiment initialement prévu peut être accommodée en fonction des ressources énergétiques restantes.

❖ Les travaux de ce chapitre présentent une méthode de résolution de conflits pour les systèmes multi-UAV. La méthode d'auto-adaptation intégrée ajuste dynamiquement les valeurs de récompense, évitant les conflits potentiels. Elle est efficace, avec une latence minimale et une faible consommation d'énergie. L'étude de cas démontre sa capacité à identifier et résoudre les conflits comportementaux, améliorant la robustesse de la planification de mission. L'évaluation complète des méthodes de planification de mission basées sur MDP pour les UAVs souligne la robustesse et la résilience de l'approche proposée.

Pour répondre à la nécessité d'adapter la méthode de résolution de conflits présentée dans le chapitre 4 aux systèmes multi-UAV, il devient impératif de considérer les défis liés à la communication entre les UAVs. Ces échanges d'information sont inévitablement sujets à des défaillances potentielles. Afin de minimiser la dépendance des UAVs à des

communications externes, privilégiant autant que possible la communication locale, et de ne recourir aux communications via le cloud qu'en cas de nécessité pour résoudre des conflits, l'introduction d'un module prédictif d'autonomie des UAVs s'avère essentielle. Ce module permet de prédire avec précision l'autonomie des UAVs, déterminant ainsi le mode de communication nécessaire pour accomplir la mission de manière optimale.

❑ Chapitre 5 : contribution 3 - adaptation avec coopération via le réseau local / le cloud

Dans le cadre d'une mission distribuée impliquant un système collaboratif multi-UAV opérant dans des environnements incertains, les UAVs sont confrontés à des défaillances potentielles et à des cyber-menaces. Les travaux précédents ont proposé un système adaptatif pour relever ces défis. Cependant, il est essentiel d'établir un système qui améliore la gestion de la collaboration entre les différents membres du système multi-UAV. L'approche proposée repose sur un mécanisme intégrant des réseaux Bayésiens. Cette approche permet d'estimer la sûreté des UAVs et facilite la communication directe avec les UAVs voisins afin de promouvoir l'adaptation interne. En cas de compromission des communications locales, le processus d'adaptation peut être transféré de manière transparente vers le cloud. En conséquence, cette approche contribue à une diminution du volume de données transmises par le système, ce qui se traduit par une réduction de la consommation d'énergie au niveau global du système multi-UAV.

❖ Auto-adaptation de politique en mode local et hybride

Le choix entre les modes local et hybride implique des compromis en termes de contrôle, d'utilisation des données, de détermination des actions alternatives et de priorité des UAVs. Le mode local utilise un contrôle décentralisé et des informations locales pour la prise de décision. Le mode hybride combine le contrôle décentralisé et centralisé, exploitant les données locales et du cloud.

Le choix du mode dépend des exigences de l'application et du compromis entre l'autonomie locale et l'assistance du cloud. Le mode local offre simplicité et autonomie, tandis que le mode hybride fournit une approche plus adaptable et riche en informations.

❖ Réseaux de communication des UAV

Il existe plusieurs moyens de communication entre les UAVs d'un système multi-UAV, notamment l'architecture de système multi-UAV basée sur l'infrastructure [NAL21] ; l'architecture Flying Ad Hoc Network (FANET) [Jos+22] ; le réseau maillé sans fil [Cui+17] ; les réseaux cellulaires [LTW+22]; les communications par satellite [Lee+22] ; les communications via le cloud [Jun+21].

❖ Prédiction de l'autonomie du drone et mode de communication nécessaire (mode local / hybride)

Un module de commutation de mode local/hybride sera intégré dans chaque UAV, et

il sera utilisé pour estimer le degré d'autonomie de chacun. En analysant le comportement du système UAV en situation d'incertitude, en tenant compte des données des capteurs et des données de réseau des voisins locaux, ce module peut estimer le meilleur choix de mode de fonctionnement, comme le montre la figure 4.

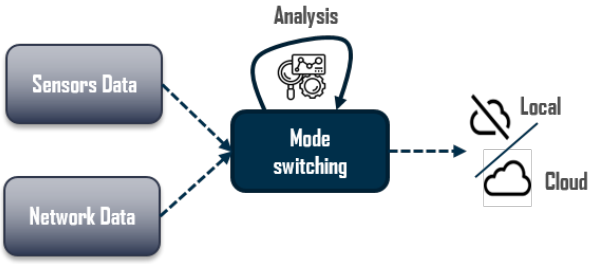


Figure 4 – Diagramme de commutation de mode.

1. **Données d'entrée et mesures de performance**

Les données d'entrée du module de commutation se divisent en deux catégories principales :

a. **Données de l'UAV** : incluent des informations sur la santé du système embarqué provenant de l'autopilote, des capteurs et de la batterie.

b. **Données réseau** : pour évaluer les performances globales de la Qualité de Service (QoS) des communications, deux types de communications sont distingués :

Communications Locales : impliquent l'échange de données et d'informations entre les UAVs au sein du système multi-UAV.

Communications via le cloud : utilisent des services et des plateformes basés sur le cloud pour la communication entre les UAVs.

Pour déterminer les performances des deux types de communication, nous analysons et comparons principalement les mesures les plus couramment utilisées dans les communications dans un système multi-UAV [PPB19; Sha+19; Sin+19; YL19; Yog21]. L'examen de la littérature existante nous a permet de sélectionner trois principales mesures de performance le taux de livraison des paquets (PDR), le délai de bout en bout (E2E) et le débit (Thp).

2. **Principes de la prédiction du mode de fonctionnement**

La méthode de prédiction de l'autonomie du drone vise à trouver et à adapter le mode de fonctionnement des UAVs en ligne. Sur la base de l'analyse des données des capteurs de l'UAV, des données du réseau local (communications inter-UAVs directes du système multi-UAV) et des données du réseau avec le cloud, la méthode estimera le bon fonctionnement et de l'autonomie de l'UAV.

La probabilité de l'autonomie de l'UAV est calculée en combinant les différentes

métriques les plus utilisées dans les communication de systèmes multi-UAV. Nous avons construit le réseau bayésien (BN) suivant:

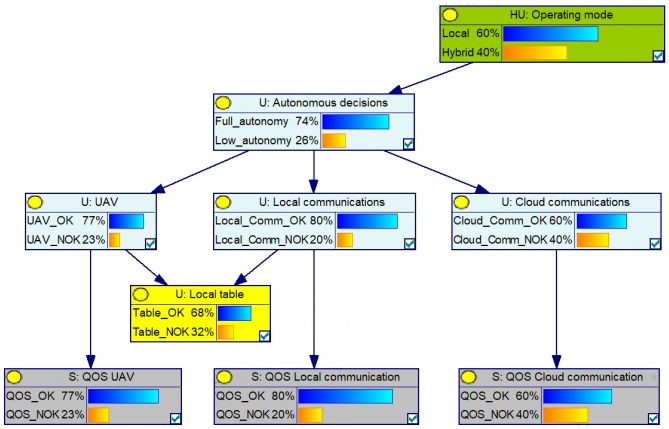


Figure 5 – Mode switching modeled with BN.

La circulation de l'information au sein de ce réseau bayésien permet de tirer des conclusions ou à faire des prédictions sur les variables incertaines du réseau, dans notre cas estimer le degré d'autonomie d'un UAV, et donc de commuter sur le bon mode de fonctionnement permettant d'optimiser la mission.

❖ Les travaux de ce chapitre introduisent un module pemettant d'améliorer les capacités d'adaptation des systèmes multi-UAV collaboratifs par le biais d'une adaptation du mode de fonctionnement basée sur un réseau bayésien. L'accent est mis sur l'auto-adaptation des politiques en mode local et hybride (via le cloud), ainsi que sur l'exploration des réseaux de communication des UAVs afin d'optimiser la planification des missions pour les UAVs. À travers cette contribution, nous avons proposé une approche pour prédire le degré d'autonomie des drones pour définir le mode de communication le plus adapté (réseau local ou hybride).

❑ Chapitre 6 : conclusion générale et perspectives

Dans cette thèse, on s'est intéressé particulièrement aux problèmes de la prise de décision distribuée dans un système multi-UAV dans un environnement incertain. En effet, dans diverses applications et dans des environnements de mission en évolution, les limites d'un système avec UAV unique, telles que les contraintes énergétiques, la capacité de calcul, la capacité d'exécution et la capacité de service, entravent la couverture d'une zone étendue. Les systèmes multi-UAV sont censés remédier à ces limitations en permettant une action coopérative pour recueillir des informations complètes de manière efficace et précise. Toutefois, les décisions prises au niveau du système multi-UAV peuvent entrer en conflit avec les objectifs de la mission, ce qui nécessite un mécanisme de gestion des conflits au sein du système multi-UAV.

À travers cette thèse, nous avons pu :

(a) faire une comparaison des méthodes fondamentales de planification des missions en introduisant de nouveaux critères pour adapter le choix de la méthode de résolution d'un problème.

(b) proposer une nouvelle approche pour la résolution de conflits avec adaptation de récompenses (rewards) associé à des actions dans le cadre d'un système collaboratif multi-UAV.

(c) enfin, proposer un nouveau module utilisant des réseaux Bayésiens pour estimer la sécurité des UAVs et de leur communications. Ce module permet de favoriser l'utilisation les données locales (via les communications local) pour résoudre les conflits sans passer via le cloud. Dans le cas échéant, les communications via cloud sont alors utilisée pour résoudre les conflits et améliorer la collaboration entre les UAVs du système.

Dans les perspectives de travail, voici quelques idées qui peuvent être proposées pour améliorer et étendre l'approche actuelle.

1. **Court-termes**
   (a) **Intégration de la prédiction du mode de fonctionnement à la planification de mission :** fusionner prédiction de l'autonomie du drone et mode de communication nécessaire basée sur un réseau bayésien (BN) avec le processus de décision Markovien (MDP) pour la planification de missions, en évaluant leur fonctionnement synergique via des simulations.
   (b) **Raffinement de la prédiction de l'autonomie du drone et mode de communication :** affiner la prédiction de l'autonomie du drone basé sur BN en définissant des estimateurs de Qualité de Service (QoS) pour UAV, communication locale et cloud, renforçant la gestion de l'auto-adaptation.
   (c) Développement d'outils informatiques facilitant la spécification, la génération et l'intégration des estimateurs QoS au niveau des UAV ou des systèmes multi-UAV.
   (d) **Exploration des supports matériels :** examiner les options CPU, GPU, FPGA pour mettre en œuvre efficace des moniteurs et organes de décision, optimisant la performance et l'efficacité énergétique dans les applications UAV.
2. **Moyen à plus long termes**
   (a) **Coopération entre drones hétérogènes :** étendre la collaboration à des drones hétérogènes (aériens/terrestres/marins) pour améliorer l'efficacité et la performance des missions dans divers scénarios opérationnels.

(b) **Étude du *Federated Learning (FL)* :** explorer le paradigme d'apprentissage machine collaboratif "Federated Learning (FL)", améliorant la prise de décision décentralisée des drones tout en préservant la confidentialité des données.

Abstract

Unmanned Aerial Vehicle (UAV)s thrive in complex, repetitive, and hazardous environments, enhancing mission quality, productivity, and safety. Operating in non-deterministic environments with unpredictable events, these autonomous vehicles face challenges necessitating independent and real-time decision-making for effective mission management.

This research focuses on the decision-making on multi-UAV system in case of collaborative mission. Three aspects are mainly covered: 1) the choice of the method for mission planning according to specific criteria, 2) the self-adaptation of policies based on rewards tuning for mission planning, 3) Bayesian Network (BN)-based operating mode adaptation for policy self-adaptation in a collaborative multi-UAV system. For mission planning, a focus on the Markov Decision Process (MDP) framework and a comparative study of the three fundamental MDP resolution methods are proposed to facilitate the choice of a resolution method according to the criteria of the problem to be solved. Using MDP-based decision engines at the multi-UAV system level can lead to conflicts. The conflict management mechanism based on the rewards adaptation approach proposed shows how to detect conflicting member UAVs by the embedded edge devices and how UAV adjusts its mission plan to avoid conflicts in the multi-UAV system and resolve conflicts. Finally, we explore a local and hybrid mode switching mechanism based on BN to enable the self-adaptation of policies. This adaptation will be driven by continuous Quality of Service (QoS) monitoring of both the UAV and communication networks, aiming to optimize mission planning for achieving total drone autonomy. This work makes it possible to draw up an assessment and some perspectives.

Keywords: Mission Planning, Decision-Making, Markov Decision Process, Self-adaptation, Bayesian Networks, Autonomous Vehicles.

Abstract (in French)

Les véhicules aériens sans pilote (UAV) prospèrent dans des environnements complexes, répétitifs et dangereux, améliorant ainsi la qualité, la productivité et la sécurité des missions. Opérant dans des contextes non déterministes avec des événements imprévisibles, ces véhicules autonomes sont confrontés à des défis nécessitant une prise de décision indépendante et en temps réel pour une gestion de mission efficace.
Cette recherche porte sur la prise de décision sur un système multi-UAV en cas de mission collaborative. Trois aspects sont principalement abordés : 1) le choix de la méthode de planification de mission selon des critères spécifiques, 2) l'auto-adaptation des politiques basées sur le réglage des récompenses pour la planification de mission, 3) l'adaptation du mode de fonctionnement basé sur BN pour auto-adaptation des politiques dans un système collaboratif multi-UAV. Pour la planification de mission, un accent sur le cadre du Processus de Décision Markovien (MDP) et une étude comparative des trois méthodes fondamentales de résolution de MDP est proposée pour faciliter le choix d'une méthode de résolution selon les critères du problème à résoudre. L'utilisation de moteurs de décision basés sur MDP au niveau du système multi-UAV peut conduire à des conflits. Le mécanisme de gestion des conflits basé sur l'approche d'adaptation des récompenses proposée montre comment détecter les UAV membres en conflit par les dispositifs de périphérie intégrés et comment un UAV ajuste son plan de mission pour éviter les conflits dans le système multi-UAV et résoudre les conflits. Enfin, nous explorons un mécanisme de commutation de mode local et hybride basé sur un réseau Bayésien pour permettre l'auto-adaptation des politiques. Cette adaptation sera pilotée par une surveillance continue Quality of Service (QoS) du UAV et des réseaux de communication, visant à optimiser la planification des missions pour atteindre une autonomie totale des drones.

Mots clés : Planification de mission, Prise de décision, Processus de décision de Markov, Auto-adaptation, Réseaux Bayésiens, Véhicules autonomes.

Acknowledgement

First and foremost, I want to express my heartfelt gratitude to the jury members for agreeing to evaluate this work and allowing me to defend it. I especially want to thank the reviewers, namely Stéphane MOCANU and Nathalie MITTON, for their trust and meticulous examination of this document, and to Jean-Philippe BABAU, David ESPES, and Káthia OLIVEIRA for their presence on the jury.

None of the contributions presented in this work would have come to fruition without the support of my thesis advisor, Catherine Dezan, and my co-advisor, Kalinka Branco from USP (Brazil), along with the valuable collaboration of David Espes. I sincerely thank them for their exceptional patience and academic guidance throughout this research journey, and specifically, I extend my appreciation to Kalinka for welcoming me to the Critical Embedded System Laboratory - LSEC at ICMC/USP. Their extensive knowledge, motivation, and patience have enriched my doctoral experience and steered my research efforts. I also thank the members of my thesis monitoring committee for their time, guidance, and contributions to my thesis.

A special acknowledgment goes to all the teachers I had throughout my studies, who each contributed in their own way to shaping the person I have become today.

A sincere thank you to all the teaching staff of the computer science department for their good spirits and the pleasure I have had in teaching alongside them. Your collaboration and positive atmosphere have greatly enriched my teaching experience.

Thanks to the system engineers and administrative staff for their availability, kindness, and efficiency. Your support has been invaluable, and I appreciate our positive and effective collaboration.

I would like to thank my father and mother, to whom I would like to dedicate the only book I wished you could read, you who never learned to read or write. My gratitude goes to my wife, Lynda, for all the moral support. My gratitude towards my sister Lahna and my brothers Omar, Amazigh, and Yacine is immeasurable. Their love, support, exceptional understanding, and encouragement have made this possible. My feelings of love and gratitude for them can hardly be expressed in words.

Additionally, I would like to express my gratitude to my friends and colleagues who gave me their moral and intellectual support.

Thank you to everyone who enabled me to complete this thesis under the best conditions.

Finally, I apologize to anyone who may feel overlooked on this list. I thank you and kindly ask for your understanding of this unintentional omission.

As Albert Einstein said:

"***The only source of knowledge is experience.***
Logic will get you from A to B.
Imagination will take you everywhere."

Table of Contents

List of Figures

List of Tables

Acronyms

A2C Advantage Actor-Critic 62
AC Actor-Critic 62
ACO Ant colony optimization 63
API Application Programming Interface 116
ARED Allocations de Recherche Doctorale 46

BN Bayesian Network 20, 23, 25, 50, 104, 121, 130–135, 137–140, 142, 144, 145, 149, 150, 166

CAV connected autonomous vehicles 62, 63
CDE Contrat d'établissement doctoral 47
CESER Conseil économique, social et environnemental régional 45
COT cost of time 63
CPT conditional probability table 130, 135

DAG Directed Acyclic Graph 130
DBN Dynamic Bayesian Networks 46
DP Dynamic Programming 8, 10, 48, 70, 74, 75, 81–83, 85, 90–93, 149
DTs Decision Trees 71

E2E End-to-End 19, 62, 124, 126, 127, 129, 130, 133

F.I Flying information (speed, acceleration, and angles) 64, 65
FANET Flying Ad Hoc Network 18, 123, 128
FL Federated Learning 151

GCS Ground Control Station 59
GPS Global Positioning System 5, 45, 89, 104–106, 118, 132, 140, 148
GUI Graphical user interface 116

ICTs Information and Communication Technologies 46
IMU Inertial Measurement Unit 89, 132

L.I. Location information (longitude, latitude, and height) 64, 65

MC Monte Carlo 8, 9, 48, 149

MDP Markov Decision Process 6, 8–17, 23, 25, 46–52, 62, 65, 69, 70, 72–75, 78, 81, 82, 85, 88–93, 96, 98–102, 104–111, 115, 118, 119, 148–150, 166

ML Machine Learning 69, 70, 151

MoTie Monitoring of mobile Things with Intelligent and embedded adaptations for secure services 46

N.sp Not specified 64, 65

NN Neural Networks 46

PDR Packet Delivery Ratio 19, 126, 129, 130, 133

PI Policy Iteration 10, 16, 48, 72, 75, 76, 78, 79, 81–83, 85, 88, 91–93, 101, 105, 107, 113, 114

PN Petri Net 8, 48, 149

POMDP Partially Observable Markov Decision Process 8, 48, 61, 62, 65, 149

QoS Quality of Service 23, 25, 126, 132–134, 136–139, 150, 166

RDDL Relational Dynamic influence Diagram Language 8, 48, 149

RL Reinforcement Learning 62, 70, 71

ROS Robot Operating System 116

S&A Sense and Avoidance 65

SAMM Systèmes autonomes en Milieu Maritime 46, 47

SIMFC Swarm Intelligence-inspired Multi-layer Clocking Control 64, 65

SMD MAR Systèmes Multi-Drones multi-milieux appliqués au domaine MARitime 45, 46

SVMs Support Vector Machines 71

TD Temporal Difference 8, 9, 48, 70, 76, 81, 82, 90, 91, 149

Thp Throughput 19, 127, 129, 130, 133

UAV Unmanned Aerial Vehicle 5–7, 11–21, 23, 25, 43, 44, 47, 49–51, 53–67, 70, 73, 74, 89–93, 95–99, 101–108, 111, 115, 117–119, 121–126, 128, 130, 132–134, 136–145, 147–151, 166

UGVs Unmanned Ground Vehicles 46

UTM Unmanned Aircraft Traffic 62, 65

V-REP Virtual Robot Experimentation Platform 116

VI Value Iteration 10, 48, 72, 74, 75, 78, 79, 81–83, 85, 88, 91–93, 149

Nomenclature

c_i	Constraint i
α	Learning rate
Δ	Precision factor that defines the difference between two values function
ϵ	Stopping condition
$\epsilon - greedy$ policy	Policy that chooses the best action
ϵ_{decay}	Value which will be deduced from ϵ at each iteration
γ	Discount factor
$\mathcal{A}$	Finite set of actions
$\mathcal{O}$	Method complexity
$\mathcal{R}(s, a)$	Immediate reward
$\mathcal{S}$	Finite set of states
$\mathcal{T}(s, a, s')$	Probability [0,1] of moving to state s' when action a is executed in state s
π	Policy (Solution of an MDP)
π^*	Optimal policy
a_i	Action i
$Average\_{Exec\_time}$	Average execution time
$Average\_{Iter}$	Average of iterations
$Bool_{Conf}$	Boolean label used to identify the MDP containing conflicting states
cc_1	Condition of the convergence 1: find a path from any state s of the grid to the Goal state

cc_2 condition of the convergence 2: find a path from the Start state (square(1,1)) of the grid to the Goal state

$Exec\_time_{\_max}$ Maximum execution time

$Exec\_time_{\_min}$ Minimum execution time

$Init_r$ Initialization of the initial state (1^{st} instruction) randomly

$Init_s$ Initialization of the initial state (1^{st} instruction) to the start state (square(1,1) of the grid

$iter$ Number of iteration

$Iter_{\_max}$ Maximum of iterations

$Iter_{\_min}$ Minimum of iterations

$iter_{max}$ Maximum number of iterations

$Q^*(s, a)$ Optimal Q-function $Q^*(s, a)$

s_i State i

ST_{Conf} Conflicting states

$T_{\_Land}$ Time needed for resolution of the MDP Landing

$T_{\_Nav}$ Time needed for resolution of the MDP Navigation

$T_{\_Parallel}$ Time needed for resolution parallel execution of the tree MDPs

$T_{\_Sequential}$ Time needed for resolution sequential execution of the tree MDPs

$T_{\_Track}$ Time needed for resolution of the MDP Tracking

$terminal_{state}$ State **s** is terminal and has no action to take from it to go to another state **s'**

$V_\pi^*(s)$ Optimal Value function

$V_\pi(s)$ Value function

Q-table Table of Q-values

Ver. Algorithm version

CHAPTER 1

Introduction

Contents

1.1 General context

The 21^{st} century has witnessed significant advancements, particularly in robotics, fueled by the continuous evolution of electronics and computing capabilities. This progress has paved the way for developing more precise, faster, and autonomous robots, strengthening the reliability and accuracy of on-board intelligence systems. Unmanned Aerial Vehicle (UAV) have emerged as pivotal players in environments where human intervention is challenging, repetitive, or dangerous. UAVs are being widely used in various fields such as security [Lee+21], inspection and photogrammetry [Gha+21], mapping or video surveillance operations [Mal+21], agriculture [Rad+20], internal operations of many warehouses and logistics platforms [Che+20], search and rescue missions [AKY21], military

missions [CAM20] and etc. Notably, UAVs have proven to be versatile tools in addressing the challenges posed by the coronavirus crisis [Res22]. Their applications include detecting and responding to abnormal situations, such as unauthorized gatherings, using features like thermal imaging cameras. Furthermore, UAVs mitigate human interaction by disseminating health advisories through drone-mounted loudspeakers, delivering COVID tests, and executing various tasks while maintaining social distancing.

However, considering the geographical context of the thesis funding, particular emphasis has been directed toward missions situated within the maritime environment. Within this maritime domain, aerial drones manifest as indispensable instruments, offering a supplementary perspective alongside traditional maritime vehicles. Collaboration with singular or multiple aerial drones presents clear advantages for maritime missions (Figure 1.1), especially in researching and identifying specific objects or individuals, detecting sea and coastal pollution, surveillance of port areas, and reconstructing 3D images of floating structures or lost objects.

Figure 1.1 – Collaborative mission between areal and maritime drones.

Moreover, the cooperative aspect of maritime drone usage extends to innovative roles, such as employing maritime drones as motherships. These motherships can serve as stations for recharging UAVs when their batteries are low, thereby extending the operational endurance of the UAV fleet. This cooperative strategy not only optimizes mission duration but also enhances the overall effectiveness of the aerial drone fleet in maritime exploration.

Nevertheless, within the execution of these missions, the maritime environment introduces distinct challenges and presents inherent complexities, including unpredictable weather conditions, potential communication interference, and the need for resilient navigation systems to overcome adversities. For instance, the precise identification of objects

or individuals may be restricted by factors such as poor visibility due to fog or adverse weather conditions. A notable example is the vulnerability of a critical component (e.g., the Global Positioning System (GPS)) to failure. These failures can be caused by factors such as inadequate satellite coverage or deliberate interference via spoofing or jamming attacks.

Addressing these challenges highlights the necessity for ongoing scientific research and technological advancements to strengthen these aerial systems, ensuring adaptability and robust performance in maritime mission complexity. This interdisciplinary approach is crucial to advancing the capabilities of aerial drones and optimizing their contribution to maritime exploration and problem-solving efforts.

1.2 Drone projects in Brittany

In 2010, the concept of "maritime informatics" (*marétique*) emerged in Brittany, integrating computer systems for managing maritime operations [DA19]. This concept aligns with the region's commitment to sustainable development and leverages historical digital expertise [DA19].

The convergence of the maritime sector with digital technologies offers opportunities such as understanding marine ecosystems, facilitating the energy transition, and optimizing land-sea communications.

The Regional Economic, Social, and Environmental Council (*Conseil économique, social et environnemental régional (CESER)*) is exploring the role of maritime informatics in specific Brittany use cases:

- ➥ Marine ecosystem management: understand, monitor, and protect marine ecosystems, including the exploitation of marine food resources;
- ➥ Maritime engineering and operations: design, produce, maintain ships, and navigate for maritime activities;
- ➥ Integrated coastal development: manage the flow of people and goods at the land-sea interface, produce and distribute renewable marine energies, and engage in maritime activities while transmitting maritime culture.

The different projects conducted within the CESER in Brittany related to the thesis are:

- ❖ **SMD MAR project**:

  Autonomous underwater robots demonstrate continuous, high-quality measurements in the ocean. The **Multi-Drone Systems in Multiple Environments Applied**

to the Maritime Domain (*Systèmes Multi-Drones multi-milieux appliqués au domaine MARitime (SMD MAR)*) project is a CPER project (2015-2022) co-financed by the French State and Brittany region aims to boost robotic resources by adopting a global and cooperative approach. The system includes underwater robots, surface and aerial drones, and a deployable control/command platform for land or ship use. This investment project made it possible to purchase drones and on-board equipment used during the thesis.

The objective of this project is to increase the level of autonomy of drones by significantly:

1. Enhancing drone autonomy by incorporating on-board computing power for demanding tasks (e.g., target detection, obstacle avoidance, navigation) without increasing energy consumption, volume, or weight.
2. Embedding the necessary intelligence (e.g., Neural Networks (NN), Markov Decision Process (MDP), Dynamic Bayesian Networks (DBN)) for autonomous decision-making in complex missions and optimal hardware configuration.

❖ **SAMM project**:

Like Unmanned Ground Vehicles (UGVs), autonomous ships are getting much attention, especially in Nordic countries. France must remain caught up to stay abreast of this significant trend impacting maritime transportation and other sectors. The Brittany Region supports the research program "Autonomous Systems in Maritime Environments" (*Systèmes autonomes en Milieu Maritime (SAMM)*), including different Brest establishments (ENSTA -*École nationale supérieure de techniques avancées*, IMT Atlantique - Institut Mines-Télécom Business school, UBO - *Université de Bretagne Occidentale*, ...) aiming to develop an excellent industry in Information and Communication Technologies (ICTs) specifically for maritime drones. It involves sustained regional support across five domains: intelligence, sensors and algorithms, embedded systems, data mining, and human-machine interactions.

❖ **MoTie project**:

The Monitoring of mobile Things with Intelligent and embedded adaptations for secure services (MoTie) is an international cooperation project with Brazil established within the research collaboration framework with USP (University of Sao Paulo), providing a practical context to the thesis regarding the security of autonomous vehicle services. It aligns with the regional SAMM project (Doctoral Research Grants - ARED) on drones in the maritime domain, focusing primarily on embedded and autonomous systems. The objective is to propose a contextual adaptation and onboard implementation of intelligent monitoring systems for the security of autonomous vehicle services to counter the various hazards of the missions. This can be done with

permanent monitoring of possible attacks or failures at the level of the multi-UAV system of drones by setting up intelligent monitoring systems capable of estimating the state of the on-board system and on-board applications and ordering communications by considering the context of the mission. These devices will be integrated into a drone decision-making engine.

This thesis receives 50% of its funding from the Brittany Region through the regional SAMM project, with the remaining support secured through a Doctoral Establishment Contract (CDE) from UBO (*Université de Bretagne Occidentale*).

1.3 Problem statement

Chabha Hireche's thesis work [Cha19] explored decision models used in the context of a drone mission. It confirmed that probabilistic models, particularly Markov Decision Processes (MDP), are widely adopted for decision-making under uncertainty in robotics.

To complete this axis, the first problem of this thesis starts from this observation and answers the following question: **what is the most suitable method for solving a decision problem under uncertainty of the context of a drone mission in an embedded context?**

However, in various applications, energy, computing capacity, and execution limitations often make a single (UAV) unsuitable for extensive coverage. Multi-drone systems (UAV), such as swarms, aim to achieve wider coverage, improved surveillance, and more efficient mission execution through cooperative action to gather information quickly and accurately.

Figure 1.1 illustrates the other scientific issues this thesis aims to address.

Within a multi-UAV system, decisions may conflict and deviate from the mission objectives, necessitating a conflict management mechanism. This observation leads to a second issue: **what are these conflicts, and how can they be resolved in a distributed multi-UAV framework modeled using MDP?**

Finally, in the context of collaborative missions with multi-drone systems, a third and final problem arises: **how can we guarantee that a drone can carry out its mission in total autonomy? how can we implement conflict resolution within a multi-UAV system while minimizing communications via the cloud and reducing the volume of data to be transmitted to maximize drone autonomy?**

1.4 Contributions

1.4.1 Contribution 1: comparison of fundamental methods for mission planning

In the literature, there are several probabilistic models for decision-making under uncertainty, such as Markov Decision Process (MDP) [HDB20; Han+19], Partially Observable Markov Decision Process (POMDP) [CMO16], Relational Dynamic influence Diagram Language (RDDL) [YWW20], Petri Net (PN) [RCB15], etc. One of the most common frameworks for decision-making and planning under uncertainty in robotics is to use MDPs, leveraging their capacity to model sequential decision problems and incorporate uncertainty, thus providing a structured framework for optimal decision strategies in dynamic environments.

This work focused on the MDP framework and its resolution methods. Three fundamental classes of methods for solving finite Markov decision problems [SB98] are Dynamic Programming, Monte Carlo methods, and Temporal Difference learning. Each class of methods has its strengths and weaknesses.

Dynamic Programming (DP) methods are well-developed mathematically but require a complete and definite model of the environment [Bel66]. In contrast, conceptually straightforward and model-free, Monte Carlo (MC) methods encounter challenges in efficiently estimating policies due to their non-incremental nature [MU49]. On the other hand, TD methods do not require a model and offer full incrementality, although their analysis and tuning are more complex. These approaches diverge in terms of efficiency and convergence speed [Tes95]. It's worth noting that Monte Carlo (MC) methods may be less practical in robotics, as they theoretically rely solely on past experiences (sample episodes) without accounting for an agent's future decisions. This limitation restricts the policy learned from real-world interactions.

This work mainly focused on three methods for solving MDPs: Value Iteration and Policy Iteration methods (DP) and the Q-Learning method (TD). It proposes new criteria to adapt the decision-making method to the application problem, highlighting that this adaptation is particularly relevant in the embedded context. Through experimental experiences, this work demonstrates that the Q-Learning method is interesting in simple and regular cases and demonstrates that in irregular cases, classical MDP resolution methods are more reasonable to implement in critical systems.

This contribution was the subject of a publication at the IEEE ICUAS 2021 conference [Ham+21].

1.4.2 Contribution 2: self-adaptation based rewards tuning for mission planning

The MDP approach is one of the best candidates for handling decision-making and planning under uncertainty, which is common in the UAV environment. It consists of an agent and an environment. The agent interacts with the environment by observing the transitions of states and receiving rewards for finding an optimal action in each state. A change in the agent's situation or the environment will lead to a change in the transition probabilities and/or the rewards of the actions, which in turn leads to a change in the optimal action, which is not immediately considered in other decision-making approaches because it requires learning time, whereas MDPs use probabilities to accelerate problem-solving. Moreover, MDP can also include physical constraints, mission safety requirements, and conflict resolution in the decision-making engine, thereby improving mission safety, accuracy, and efficiency.

The objective is to utilize parameters characterizing the MDP, primarily the rewards associated with actions. It has been determined through experimentation that tuning the rewards associated with actions at the state level leads to modifying the resulting policy of the modeled problem, therefore adapting the mission by removing the conflicts.

The proposed method can be applied to the multi-UAV context (2-a) and the single-UAV context.

❖ **Contribution 2-a: application to multi-UAV context**

One challenge in the development of autonomous vehicles is to define an appropriate decision-making engine that can handle a variety of control problems [EMA20; AK20; Wan+20; FC18; Bar19] to increase their autonomy level and safety. However, at the multi-UAV level, each UAV has its own MDP, and the decisions they make are unique and optimal at their level. However, the decisions made at the multi-UAV level may be contradictory and may not meet the mission objectives. A conflict management mechanism, therefore, is proposed within the multi-UAV system.

We introduced a new approach that allows:

➥ Conflicts resolution: in a multi-UAV system or team of UAVs where multiple MDPs are being executed concurrently, conflicts may arise. To resolve them, the mission plans of the individual UAVs are adjusted (by modifying the rewards of the conflicting multi-UAV system actions).

➥ Dynamic reward shaping: a systematic online method is proposed to adapt rewards in real-time with minimal data sharing between UAVs in the multi-UAV system.

➥ Energy-aware strategies: UAV actions are aware of their energy level. The energy level is perceived as an internal constraint and can alter the actions performed by a UAV.

This contribution was the subject of a publication at the IEEE ICUAS 2023 conference [Ham+23].

❖ **Contribution 2-b: application to UAV context**

With the increased complexity of missions, decomposing an MDP into several sub-MDPs becomes necessary. The decomposition involves parallel decisions between different agents inside the same UAV, but the execution of concurrent actions can lead to conflicts. In addition, problems due to the system and sensor failures may appear during the mission that can lead to negative consequences (e.g., a crash of a UAV caused by a drop in battery charge). We present a new method to prevent behavior conflicts that can appear within distributed decision-making and to emphasize the action selection if needed to ensure the system's safety and various requirements. This method considers the different constraints due to antagonist actions and some thresholds on transition functions to promote specific actions that guarantee the system's safety. Then, it automatically computes the rewards of the different MDPs related to the mission to establish safe planning.

This contribution was the subject of a publication at the IEEE/IFIP DSN-W 2020 conference workshops [HDB20].

1.4.3 Contribution 3: adaptation with cooperation through the cloud/the local network

In the context of a distributed mission involving a collaborative multi-UAV system operating in uncertain environments, UAVs face potential failures and cyber threats. The previous work proposed an adaptive system to address these challenges. However, it is essential to establish a system that enhances the management of collaboration among the various members of the multi-UAV system. The suggested approach involves the utilization of Bayesian Networks. This approach makes it possible to estimate the safety of UAVs and facilitates direct communication with neighboring UAVs to promote internal adaptation. The goal is to maintain autonomy while fostering effective collaboration. In case of compromised local communications, the adaptation process can seamlessly transition to the cloud. This operational model has the advantage of reducing inter-drone communication, thereby enhancing the efficiency of the multi-UAV system during missions. Additionally, it contributes to a decrease in the volume of data transmitted through the system, resulting in reduced energy consumption.

This contribution is currently being written for submission to a conference.

1.5 Organisation of the document

This document is organized as follows:

- ✍ Chapter 2 defines the general concepts we will need throughout this manuscript and outlines distributed decision-making approaches by introducing the state-of-the-art on distributed decision-making and the challenges and open questions that need to be addressed.
- ✍ Chapter 3 describes Contribution 1. Focusing on decision-making models, we compare the three fundamental methods for solving MDPs. Theoretically and experimentally, via numerical simulation, we give new criteria to adapt the decision-making method to the application problem, with the parameters' explanations.
- ✍ Chapter 4 concerns Contribution 2, which involves detecting and resolving conflicts with the self-adaptation of policies. We have identified the conflicts that can appear during UAV missions or in multi-UAV systems. This chapter is, therefore, structured to address these problems.
- ✍ Chapter 5 concerns Contribution 3, which involves adapting the operational mode for policy self-adaptation in multi-UAV systems. We aim to enable drones to resolve conflicts autonomously and communicate through the cloud only when no direct connection alternative exists.
- ✍ Chapter 6 concludes this thesis and proposes some research perspectives from this work.

1.6 Publications

International conferences and Workshops

1. M. Hamadouche, C. Dezan, D. Espes and K. Branco, "**Online reward adaptation for MDP-based distributed missions**" *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, Warsaw, Poland. [Ham+23]
2. M. Hamadouche, C. Dezan, D. Espes and K. Branco, "**Comparison of Value Iteration, Policy Iteration and Q-Learning for solving Decision-Making problems**" *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, Athens, Greece. [Ham+21]

3. M. Hamadouche, C. Dezan and K. R. L. J. C. Branco, "**Reward Tuning for self-adaptive Policy in MDP based Distributed Decision-Making to ensure a Safe Mission Planning**" *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Valencia, Spain. [HDB20]

National conferences

1. M. Hamadouche, C. Dezan, D. Espes and K. Branco, "**Prise de décision distribuée et embarquée dans les systèmes autonomes multi-UAV pour une sécurité des services d'engins autonomes**" *Conféerence d'informatique en Parallélisme, Architecture et Système (ComPAS' 23)*, Annecy, juillet 2023.
2. M. Hsaini, M. Hamadouche and C. Dezan, "**Classifieur embarqué pour la détection d'intrusions dans le contexte des véhicules autonomes**" *Conféerence d'informatique en Parallélisme, Architecture et Système (ComPAS' 22)*, Lyon, Juin 2021.

Submission planned

1. M. Hamadouche, C. Dezan, D. Espes and K. Branco, "**BN-based operating mode adaptation for policy self-adaptation in a collaborative multi-UAV system**".

CHAPTER 2

State-of-the-art in distributed decision-making for multi-Unmanned Aerial Vehicle (UAV) systems under uncertainty

Contents

Introduction

This chapter comprehensively introduces multi-UAV systems, covering foundational aspects, advancements, challenges, and prospects. Section 2.1 provides a multi-Unmanned Aerial Vehicle (UAV) systems overview, starting with an outline of UAVs and their operational characteristics. It details the advantages of multi-UAV systems, showcasing enhanced efficiency through collaboration, highlights their diverse applications, and proposes

a taxonomy of multi-UAV systems offering a structured classification. Section 2.2 summarizes the state-of-the-art in distributed and probabilistic decision-making for multi-UAV systems. Section 2.3 addresses challenges, open issues, and future directions in this field. Section 2.4 summarizes key points and suggests potential future research directions.

2.1 Multi-Unmanned Aerial Vehicle (UAV) system

2.1.1 Unmanned Aerial Vehicle

Unmanned Aerial Vehicle (UAV) (or uncrewed aerial vehicle [Hig+16]) and drone are among the different terms used in UAV fields. These additional terms are generally derived from other specifications and concepts depending on their uses and research areas [SRZ15].
For the remainder of the article, UAV is defined as an aircraft (or aerial vehicle) without a human pilot on board [VV15], incorporating an embedded system (computer) platform that makes it possible to be controlled remotely and/or to execute high-performance programs that allow it to make decisions autonomously.

UAV	Weight (kg)	Altitude (km)	Endurance (h)
Micro	0.1	0.25	1
Mini	<30	0.15–0.3	<2
Short range	200	3	2–4
Medium range	150–500	3–5	30–70
Long range	-	5	6–13
Endurance	500–1500	5–8	12–24
Medium altitude, long endurance	1000–1500	5–8	24–48
High altitude, long endurance	2500–12,500	15–50	24–48

Table 2.1 – Unmanned Aerial Vehicles UAVs classification. Adapted from the work in [SRZ15].

UAVs can be classified according to their weight, altitude, and endurance [SRZ15] as shown in Table 2.1. The UAV shown in Figure 2.1 is a mini UAV DJI FPV, according to the classification in Table 2.1. This mini-UAV weighs around 7.95 kg (including batteries and propellers), has a flight time of 20 minutes and a range of 6 km.

2.1.2 Advantages of using a multi-UAV system

A multi-UAV system is an autonomous system inspired by the self-organized mechanism of biological groups (e.g., insects, bees, and ants), which has a lot of advantages.

Some of the advantages of using a multi-UAV system are discussed in many papers [Chu+18; SBS20]. The retained advantages of using a multi-UAVs system instead of a

Figure 2.1 – DJI FPV, a mini-UAV weighing around 7.95 kg (including batteries and propellers), with a flight time of 20 minutes and a range of 6 km [Dà-].

single UAV are outlined in Figure 2.2. The main advantages are:

- **Parallelism:** for large-scale tasks that are distributed over a wide area in the environment, the multi-UAV system can deal with multiple targets in one task, e.g., in search or tracking of multiple targets missions.
- **Cost:** in some cases, having a complex single UAV could be an expensive solution. To perform adequately in a large or complicated area, the UAV has to embed heavy and costly sensors. In some cases, multiple UAVs could be cheaper due to the lower manufacturing and maintenance costs.
- **Time efficiency:** in tasks such as target search, exploration, etc., the duration of the missions can be drastically reduced with the use of multiple UAVs.
- **Energy efficiency:** since the members of a multi-UAV system are smaller and more straightforward than a complex single UAV, their power consumption is reduced in the way that they can use a small battery. As the missions are solved quickly, a multi-UAV system's energy consumption is smaller than a single, complex UAV solution.
- **Complementary:** in some tasks that require different types of sensors, each member of the multi-UAV system will have a specific set of these sensors. The cooperation between the UAVs during a mission would complement each of them.
- **Scalablity:** the interaction between members of the multi-UAVs system is done locally. New members can join the multi-UAV system anytime to support other members to finish their mission. Entering or leaving the multi-UAV system does not interrupt the general mission. So, the multi-UAV system can adapt to the change using dynamic task re-allocating schemes.

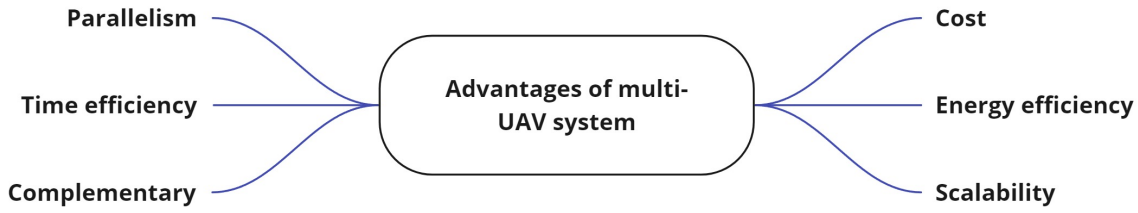


Figure 2.2 – Advantages of using a multi-UAV system.

2.1.3 Applications of the multi-UAV system

Currently, the use of UAVs (as multi-UAV system) is growing faster. They inspire and motivate developments and other innovations in various places. Multiple applications are already listed, new ones keep appearing, and precisely anticipating their development is challenging. The line between actual and potential applications is changing rapidly. With advancements in camera technologies and other types of remote sensing equipment, drone missions have expanded into consequential areas, including:

- **Video surveillance, Inspection & Tracking**
  There are several proposals for using multiple drone systems. Most of the works are focused on the audiovisual field, particularly with the shots used in most television shows (documentaries, magazines, sporting events, etc.) [Mad+19]. These systems are also used for monitoring and inspecting infrastructures (e.g., railways, power lines, etc.) [Hoa+18; Liu+19], locating and tracking targets in military applications [Gu+18], etc.

- **Network**
  In specific complex missions, a multi-UAV system is used to form a communication architecture to offer connectivity in large areas where the infrastructure is either down due to an emergency (e.g., natural disasters, terrorist attacks, and others) or inexistent for special events (e.g., concerts) [Hon+20; Raj+20].

- **Photogrammetry & SLAM**
  Photogrammetry is the science of extracting reliable quantitative data from physical objects by measuring and interpreting photographic images. It is often used by surveyors, architects, or even engineers to create topographic maps, meshes, point clouds, or drawings based on the real world by making measurements in a scene through images acquired (using UAVs in this case) according to different points of view [Aue+18].
  Simultaneous Localization and Mapping (SLAM) is a process used for building or updating a map of an unknown environment and locating the robot in that map simultaneously. Mainly, it is used for a single UAV, but more and more research is

being done to use it with multiple UAVs [SC17].

❖ **Search and Rescue**

Search and rescue operations using a single UAV are not a new concept in the literature, but the use of a multi-UAV system remains a branch of research not fully explored [AAK19] and can significantly improve these operations due to the benefits of these systems (see Section 2.1.2).

2.1.4 Taxonomy of the multi-UAV system

There are different ways in which these systems can be classified. For example, they can be classified based on the number of members in the system, mission control type, autonomy level, or coordination. After reviewing the publications on multi-UAV systems, we propose a taxonomy of these systems according to the most relevant characteristics and attributes. The retained taxonomy is outlined in Figure. 2.3.

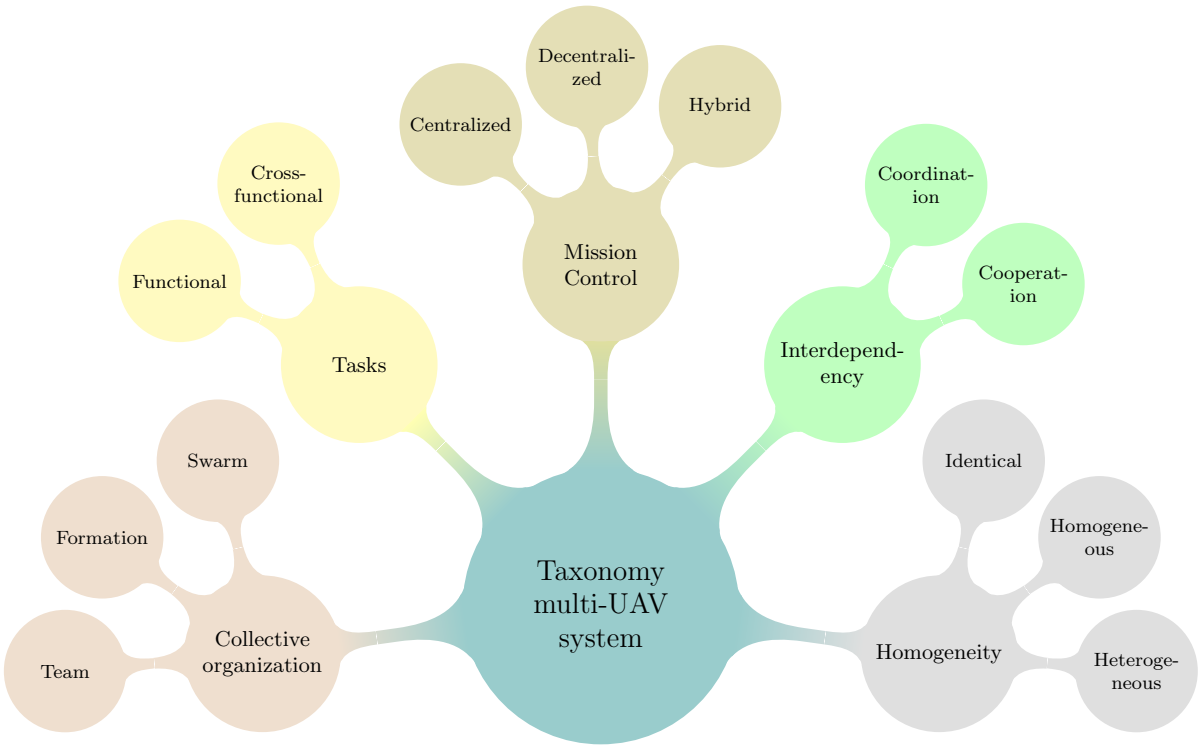


Figure 2.3 – Proposed Taxonomy for multi-UAV systems.

1. **Collective organization**

In [Chu+18], the authors have classified these systems based on the number of agents and their interaction to *teams*, *formations*, or *swarms*.

➥ In a **team**, the members are less than ten UAVs, and each member seeks to optimize individual and independent objectives. In some cases, the interaction

between the agents can be done competitively to maximize local rewards exclusively. Conversely, in certain cases, the interaction can be cooperative since the locally optimal behaviors align with approximately maximizing overall rewards. Such cooperative relationships are often modeled using game theory methods and auction algorithms.

- ➥ A **formation** has also less than ten UAVs, and each UAV has a specific task. Contrary to a *teams*, a *formation* nearly always consists of cooperative interactions, and the interdependence between the states of the agents is greatly specified for objectives such as energy efficiency.
- ➥ The largest one is the ***swarm*** that has more than ten UAVs. Each member interacts with each other competitively or cooperatively depending on the type of mission.

2. **Homogeneity**

   In [SBS20], the authors have taken over the classification given in [Dud+96]: *identical*, *homogeneous* and *heterogeneous UAVs*.

   - ➥ **Identical UAVs:** the UAVs of the system are of the same type (fixed-wing, multi-rotor, etc.) and have the same type of payloads and sensors (e.g., cameras, sniffers, meteorological sensors, etc.) and the same applications for different payloads. For example, in [VR04], they propose a framework for a cooperative strategy for multiple agents whose aim is to search for moving and evading targets in a hazardous environment using identical UAVs.
   - ➥ **Homogeneous UAVs:** the UAVs of the system are of the same type and can have different usages for the payloads and sensors. For example, in [PL06], they investigate the use of a self-organization (SO) framework for evolving UAV swarm behavior. Employing a genetic algorithm (GA) within this framework enables the swarm to locate and neutralize stationary targets efficiently. The homogeneous nature of the UAVs promotes seamless coordination, making them highly responsive to dynamic scenarios. This cohesion enhances their collective capabilities, making them well-suited for tasks that require synchronized and concerted efforts.
   - ➥ **Heterogeneous UAVs:** the UAVs of the system are not of the same type, and they have different sensors and execute various applications. In [FFG19], heterogeneous UAVs coexist within the same swarm, showcasing the flexibility of swarm behavior. With its diverse capabilities, the collective swarm is expert at searching for targets in a given area. Upon identifying new targets, tasks are intelligently assigned to individual UAVs based on their specific resources.

This inherent diversity allows for a fine and custom approach to mission tasks, making heterogeneous UAV swarms highly adaptable and effective in complex environments.

According to our literature review, the homogeneity of a UAV system depends on the specific objectives of the mission for which it is being used. For missions in the same geographical area, the UAVs tend to present greater homogeneity or heterogeneity. Conversely, the dominant trend in missions covering different geographical regions is towards deploying a uniformly identical set of UAVs. This variation in UAV homogeneity is closely linked to the geographical context and strategic requirements of the missions undertaken.

3. **Mission control and monitoring**
   A distinctive feature of classifying multi-UAV systems is how to control the UAVs belonging to the system. We distinguish:

   - **Centralized:** the Ground Control Station (GCS) or a vehicle having sufficient computing resources (UAV or ground robot) in the multi-UAV system receives all the important data (e.g., location information such as longitude, latitude, and height, flying status such as speed, acceleration, and angles) from all the agents and then manages the flight parameters and the path of all UAVs.
   - **Decentralized:** each UAV is assumed to be an agent that manages its flight parameters and path and makes decisions independently from the others, i.e., the decision is only taken according to its data.
   - **Hybrid:** GCS and UAVs are used to manage the flight parameters of the path and make decisions. For example, in [Yan+18], the GCS notifies the center control agent that determines the search area after receiving an alarm. Then, each UAV goes into this area to search for a target.

4. **Interdependency**
   Mainly, interdependence is the interaction between the members of the multi-UAV system and the environment in which they operate. We distinguish two types of interaction introduced in [Maz+15]:

   - **Coordination:** it is when all members of the system share resources (e.g., when a UAV plays the role of a radio re-transmitter from/to other UAVs and the central station) or operate in the same place in a safe manner (e.g., for the surveillance at different altitudes or from different viewpoints of the same object). For example, drones can be synchronized and perform a light show, such as the 1,218 drones used during the opening ceremony of the 2018 Olympics [Inta] that performed different patterns such as Olympic rings, a snowboarding

athlete, etc., or the 300 drones that were used during the light show of Super Bowl in 2017 [Intb]. In May 2021, 5,164 UAVs were used simultaneously, breaking the dazzling record in Shenzhen for the largest swarm of UAV [Gui].

- ➥ **Cooperation:** it is when all the system members execute a set of tasks and make decisions collaboratively to achieve a common behavior and overall goal, to receive a common reward finally. For example, in [Spy+21], multirotor UAVs were equipped with received signal strength indicator (RSSI) sensors and organized in a swarm. They cooperate between them to approximate and trail a moving target.

5. **Tasks:**
   Generally, a mission is composed of several tasks. When a multi-UAV system is considered for doing a mission, it is important to know the characteristics of the tasks. In [SBS20], they classified them into two types:
   - ➥ **Functional:** when all UAVs of the multi-UAV system perform a similar task.
   - ➥ **Cross-functional:** when the UAVs of the multi-UAV system perform dissimilar tasks.

2.2 State-of-the-art on distributed and probabilistic decision-making

We evaluated 708 publications from three digital libraries (ACM, IEEE, and Scopus) in a systematic mapping review carried out between 2019 and 2023. However, most of these articles were not relevant enough regarding the main topic of this thesis. After analyzing the remaining articles, we only found 13 articles presenting decision-making methods. The filtering parameters used to limit the number of articles are:

1. non-English publications. Some publications in other languages were found, although the searches were carried out with English keywords. We have kept only those that were written in English so that we can analyze them in depth;
2. papers that are not downloadable online or those that have content that is not fully accessible;
3. publications that are not related to distributed decision-making, probabilistic model, or UAVs as defined in this work (i.e., UAV-related papers that address other research fields);
4. papers focusing on distributed decision-making other than based on the probabilistic model applied to a multi-UAV system.

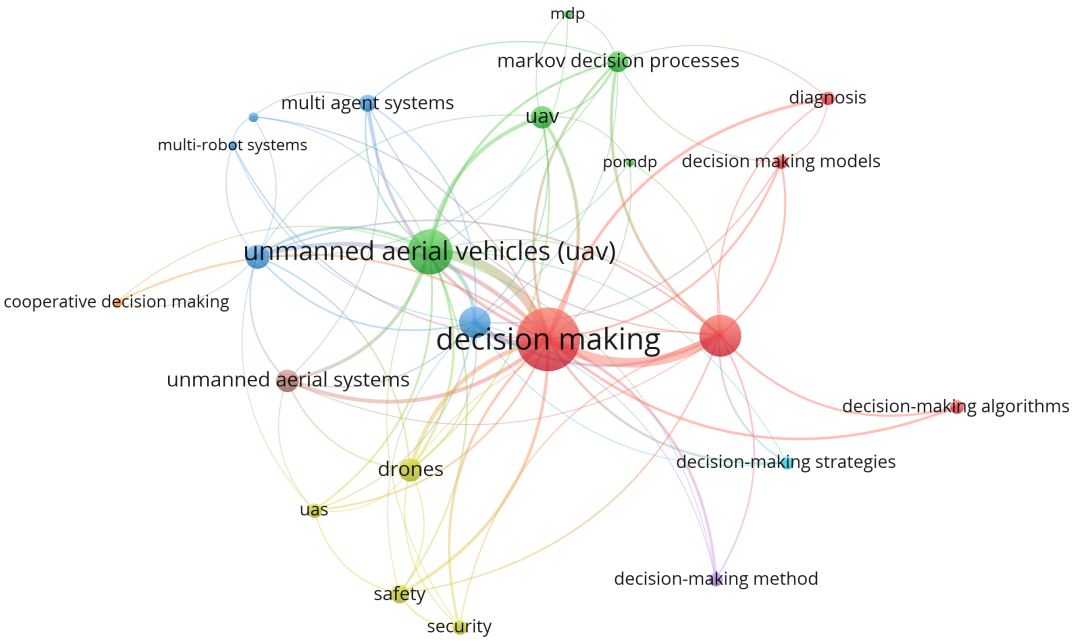


Figure 2.4 – Keywords graph and occurrence mapping for the articles selected in this review.

Figure 2.4 presents a keywords graph and occurrence mapping for the articles selected in this review. We consider keywords with at least two occurrences and at least one link to other keywords. Thereby, the keyword mapping consists of 23 keywords in total. In Figure 2.4, it is possible to note many links with the UAV and decision-making keywords. However, although the keywords related to diagnosis, safety, and security are present in Figure 2.4, there are few links correlating them to multi-robot/multi-agent systems.

To measure how each selected publication is relevant regarding the area of research on probabilistic-based distributed decision-making methods, we analyze each paper regarding the following aspects (criteria):

1. the work presents a theoretical analysis (**C1**);
2. the work presents real-world results (**C2**);
3. the work involves decision-making for UAVs in the distributed multi-UAV systems (**C3**);
4. the work is centered on decision-making for UAVs, considering diagnosis, safety and security threats, or/and the countermeasures (**C4**).

Each aspect is analyzed for each publication and classified according to the taxonomy proposed in Section 2.1.4.

Instead of solving a joint multi-UAV Partially Observable Markov Decision Process

(POMDP), Capitan et al. [CMO16] propose to split the original problem into simpler ones. UAV behaviors are auctioned during the mission. Firstly, they solve one policy for factored POMDP, where one UAV tracks one target. Then, the factored policy is used to emulate different behaviors allocated dynamically to the UAVs. They use a data fusion system to maintain a belief in the joint factored state.

In [Han+19], Han et al. propose different Reinforcement Learning (RL) algorithms to avoid random dynamic obstacles automatically. These algorithms are based on the experience shared between multi-UAVs. First, the obstacle avoidance problem is modeled as a Markov Decision Process (MDP). They introduce an Actor-Critic (AC) algorithm that combines Q-Learning and Policy Gradients and an Advantage Actor-Critic (A2C) algorithm that shares the experience between UAVs to solve this problem.

Ong et al. [OK15] propose several algorithms for conflict avoidance systems for Unmanned Aircraft Traffic (UTM). They formulate the conflict resolution problem as a stochastic problem in a large MDP and decompose it into computationally tractable subproblems (pairwise encounters). They solve these encounters offline, then iteratively combine them online to produce a locally optimal coordination solution.

Wang et al. [Wan+18] present a decentralized approach for multi-UAVs real-time conflict resolution based on pre-order flight information. They assume that each UAV can get historical flight information and current position information of UAVs within a given detectable radius when the distance of recent locations between two UAVs is smaller than a collision radius. First, they use a game theory approach to compute the safety and efficiency payoffs (reward) of the safety level strategy (actions corresponding to speed and heading adjustments) they can carry out. After that, they compute the efficiency that reflects goal achievement according to the flight mission of each maneuver strategy for collision avoidance. Then, according to the pre-order flight information, a UAV builds a memory pool and obtains the most rewarding collision avoidance strategy to be executed.

Cook et al. [CCK16] propose a fuzzy logic approach to mitigate the risk of collision between two UAVs in a congested low altitude airspace that uses a UTM system. They develop two controllers. The first one aims to solve potential collisions between two UAVs that are close to each other. Each UAV uses a robust sensing system that helps them to make a decision based only on local state information. This avoiding system is totally independent from the other UAVs and does not require any communication or coordinated maneuver. The second controller is in charge of the control of an UAV End-to-End path. It guarantees that a minimal distance between two UAVs is respected all along their paths and chooses which UAV has to change its path to keep this distance.

Shen et al. [She+18] propose a distributed resolution scheme for multiple connected autonomous vehicles to make on-ramp merge maneuvers and plan longitudinal trajecto-

ries. They use heuristics that minimize the cost of time spent on planning and finding the optimal lane change maneuver. In the beginning, a free-time algorithm is used to guess the initial time to merge the vehicles. Then, a cost of time (COT) algorithm optimizes the scheduling of the vehicles in the lane and decides which vehicle has to change lanes.

Nie et al. [Nie+16] propose a decentralized cooperative lane-changing decision-making framework for connected autonomous vehicles (CAV). The framework is composed of three core modules. The *state prediction* module employs a cooperative car-following model to predict the following state (represented by the positions, accelerations, and speeds of CAV and its surrounding interacting vehicles) of related vehicles. In the *candidate decision generation* module, the candidate decision (i.e., lane-changing or lane-keeping) is determined to optimize the advantage of the subject vehicle (if it executes the candidate decision). Once the candidate decision is generated, the candidate decision is broadcast to the surrounding interacting vehicles. After receiving a candidate decision, the *candidate decision coordination* module makes a final decision and publishes it to the neighbor vehicles.

Ebert et al. [Ebe+20] introduce a decentralized Bayesian algorithm for robot swarms to identify and cartography different features of an environment. The robots use a go/no-go decision model to map areas with different colors. Each robot performs a pseudo-random walk to cover the environment without any localization system and behaves as a Bayesian estimator while exchanging and integrating observations from nearby robots.

Dai et al. [Dai+19] propose an autonomous flocking control scheme to help a swarm of UAVs keep the same topology during a flying mission. This scheme guarantees the quality of service and communication between them and improves the overall energy consumption of the swarm. Each UAV follows another UAV that acts as a leader. This pattern avoids any collision that could occur between two UAVs.

In [LWW18], Lin et al. propose to form a cooperative formation based on an obstacle avoidance algorithm for a multi-UAV system. All UAVs are part of the same formation and are controlled as a whole. Inside the system, a communication topology is set up. They use a consensus algorithm to maintain the shape during the mission and avoid collisions. They use an improved artificial potential field method outside the system to avoid obstacles.

Yang et al. [Yan+18] use an improved Ant colony optimization (ACO) algorithm for search and rescue missions. Heavy computational tasks are distributed on each UAV of the swarm to reach a target in sea rescue and search missions. The system uses edge computing to complete calculation tasks such as the positioning of the UAVs in the environment and UAVs path determination according to the information shared between UAVs.

Adamey et al. [AOÖ17] present a collaborative multi-sensor agent and multi-target

tracking and surveillance approach. It is developed in conjunction with the Bayesian tracking framework (as the decision-making procedure relies on the belief space). The approach is based on decomposing the entire bounded environment into separate regions. Then, these regions are allocated to individual mobile sensor agents to be explored. The approach utilizes a data structure called a region allocation tree, which encodes all the candidate regions, organizes them in accordance with their subset/superset relationships, and keeps track of their exploration utilities.

In [Gen+16], Gentilini et al. focus on a path-planning strategy for multiple Unmanned Aircraft during surveillance missions. The objective is to find the set of commands for the aircraft network to minimize a cost function (depending on the relative position between the aircraft and the target). First, they developed a target tracking algorithm to provide information on target and drone motion in the surveillance area using a Kalman Filter and Bayesian network. Then, they define the objective functions as the relative position between the aircraft and the target. Finally, they use a heuristic approach to find the set of commands for the UAV that maximize the utility function.

Based on the discussion mentioned above, a side-by-side comparison in terms of case studies, collective organization, homogeneity, mission control type, communication type, data shared inside the swarm, interdependency between the members of the swarm, decision-making model used, tasks type, validation environment, scalability of the system, criteria C1 to C4 are given in Table 2.2 for the different decision-making approaches that were presented previously. The abbreviations used in this table are:

- ☞ N.sp ➡ Not specified,
- ☞ L.I. ➡ Location information (longitude, latitude, and height),
- ☞ F.I ➡ Flying information (speed, acceleration, and angles),
- ☞ SIMFC ➡ Swarm Intelligence-inspired Multi-layer Clocking Control,

Publication	Case study	Collective organisation	Homogeneity	Mission control	Communication	Data shared	Interdependency	Decision-making model	Tasks	Environment	Scalability	C1	C2	C3	C4
Capitan et al. [CMO16]	Tracking multi-targets	Team	Identical	Decentralized	Point-to-Point	On/Off detection estimation	Cooperation	POMDP	Functional	Real	High	✓	✓	✓	-
Han et al. [Han+19]	Obstacle avoidance in multi-UAVs	Team	N.sp	Decentralized	Broadcast in range	Experiences (actions)	Coordination	MDP	N.sp	Simulation	Medium	✓	-	✓	-
Ong et al. [OK15]	Conflict avoidance system for UTM	N.sp	Identical	Centralized then Decentralized	Via central node/server	L.I. & F.I	Coordination	MDP	N.sp	Simulation	High	✓	-	✓	*S&A*
Wang et al. [Wan+18]	Conflict resolution	Swarm	N.sp	Decentralized	Broadcast in range	L.I.	Coordination	Their own model based on the pre-order flight information	Functional	Simulation	Medium	✓	-	✓	*S&A*
Cook et al. [CCK16]	Mitigating the risk of a collision between UAVs using a UTM system	N.sp	Heterogeneous	Hybrid	Via central node/server	L.I. & F.I	Coordination	Fuzzy logic	Cross-functional	Simulation	High	✓	-	✓	*S&A*
Shen et al. [She+18]	Lane change decision-making	N.sp	Heterogeneous	Decentralized	Point-to-Point	S.I.	Cooperation	Estimation using heuristics	Functional	Simulation	Low	✓	-	-	*S&A*
Nie et al. [Nie+16]	Lane changing decision-making	Swarm	Heterogeneous	Decentralized	Broadcast in range	S.I.	Cooperation	Their own framework	Functional	Simulation	Low	✓	-	-	*S&A*
Ebert et al. [Ebe+20]	Classifying distributed feature of an environment	Swarm	Identical	Decentralized	Broadcast in range	Observed color in a grid cell	Cooperation	Bayesian Decision-Making	Functional	Real	High	✓	✓	-	-
Dai et al. [Dai+19]	Autonomous flocking control in UAV Networks	Swarm	Identical	Decentralized	Point-to-Point	L.I. & F.I	Cooperation	SIMFC	Functional	Real	Medium	✓	✓	✓	*S&A*
Lin et al. [LWW18]	Formation and obstacle avoidance	Formation	Identical	Decentralized	Point-to-Point	L.I. & F.I	Cooperation	Improved artificial potential field + consensus algorithm	Functional	Simulation	Low	✓	-	✓	*S&A*
Yang et al. [Yan+18]	Search and rescue	Swarm	Heterogeneous	Hybrid	Via central node/server	L.I. & tasks	Cooperation	Ant colony method	Cross-functional	Simulation	Medium	✓	-	✓	-
Adamey et al. [AOÖ17]	Multi-target tracking and surveillance	Swarm	Identical	Centralized	Via central node/server	The belief of the track target represented as a grid	Cooperation	Bayesian tracking framework + region allocation tree	Functional	Real	Medium	✓	-	✓	-
Gentilini et al. [Gen+16]	Path planning for surveillance missions	Swarm	Identical	Centralized	Via central node/server	L.I.	Cooperation	Target tracking algorithm based on Bayesian network + Kalman Filter	Functional	Simulation	Medium	✓	-	✓	-

Table 2.2 – Reviewed publications and their explored topics. On columns C1 to C4, cells filled with "✓" that represents "yes" for the respective column and with "-" that represents "no" otherwise. For column C4, if they are taking into account only the "Sense and Avoidance", cells filled with "S&A".

Through this comparative table of the reviewed works, very few works have been able to go as far as presenting real-world results, even though they have all given theoretical results. Moreover, no work has focused on distributed decision-making in a multi-UAV system considering diagnosis, safety, and security threats. Very few works that have tried to include safety and security aspects have only considered detecting and avoiding obstacles.

2.3 Challenges, open issues, and future directions

With the emergence of new technologies in robotics, UAVs are becoming more and more autonomous, fast, and robust. A multi-UAV system (swarm or team of UAVs) is usually used to achieve goals faster and more precisely and improve the efficiency of missions for complex scenarios. New approaches to utilizing multi-UAV systems in complex missions still do not benefit from the current technological advancements, mainly due to the dynamic structure of the network of the multi-UAV systems.

As shown in Section 2.2, although there is much work on decision-making in multi-UAV systems, only a small part is handled with probabilistic decision models.

As highlighted in the preceding section, there is a growing trend among academic, industrial, and military organizations to adopt heterogeneous multi-UAV systems, driven by their various advantages in mission execution. This is particularly evident in critical missions where resource optimization and precision are critical.

The open issues are:

- the potential management conflicts between members of the multi-UAV system become an increasing need. Conflicts can be internal to each member due to the non-respect of external constraints, or conflicts can be at the multi-UAV system level. *Internal conflicts* can result from incompatible actions (e.g., when a UAV_1 has to do the mission with insufficient battery power), etc. *Conflicts at the multi-UAV system level* can appear when the mission constraints are not respected; for example, the number of UAVs executing the task is exceeded (e.g., during a search mission, a target T_1 is detected and only one or even two UAVs are needed to follow it, the other UAVs will continue searching for other targets), etc.
- the decision-making models should consider the faults, including hardware failures in embedded systems such as sensors, autopilots, and embedded processors, as well as potential cyber threats. Implementing diagnostic capabilities during the mission presents an invaluable opportunity to mitigate system failures and avert potential disasters, including those that may impact human lives.
- the interactions between UAVs in a multi-UAV system should be reduced by exchanging the least possible volume of data. This reduction of transmitted data can

reduce, at the same time, the potential risk of miss transmission as well as the number of resources and the energy consumption due to the communication.

A future solution could be to manage a mission with a distributed and shared probabilistic decision model for a multi-UAV system. This model can be fed by failure prediction and cyber-attack detection models. A minimalist data exchange (sensor failure signal, cyber-attack signal, target locked for tracking, task taken, etc.) can allow good adaptation and synchronization between the members of the multi-UAV system and reduce the consumption of material and energy resources during missions.

2.4 Final consideration

In the first part of this chapter, an overview of UAVs and their classification according to their weight, altitude, and endurance is presented. In the second part, we have mainly listed the applications of the multi-UAV systems used in the last decade. In the third part, firstly, we propose a taxonomy for these systems and then present a review of the state-of-the-art on probabilistic decision-making models used in multi-UAV systems. Afterward, these works are compared based on the essential features such as case studies, collective organization, homogeneity, mission control type, communication type, data shared inside the multi-UAV system, interdependency between the members of the multi-UAV system, decision-making model used, tasks type, validation environment, scalability of the system, and criteria used for measuring the pertinence of the publications. Finally, the open issues and the uncovered or insufficiently covered items are identified, allowing a road map and a starting point for future research projects.

CHAPTER 3

Comparison of fundamental methods for mission planning

Contents

Introduction

This chapter explores the fundamental concepts of Machine Learning (ML) and decision-making, specifically focusing on Markov Decision Processes (MDP). The Section 3.1 introduces ML, outlining its basic principles and categorizing it into supervised, unsupervised,

and Reinforcement Learning (RL). Reinforcement learning is examined as a dynamic iteration in which agents adapt their decisions through a feedback loop of rewards and penalties. Subsequently, in Section 3.2, the focus extends further into MDP-based decision-making, presenting MDPs as a robust framework for modeling decisions in stochastic environments. The concept of policy, serving as a guiding strategy for an agent's actions, is defined, followed by a practical example involving a racing UAV. Section 3.3 explores in-depth MDP solution methods, such as Dynamic Programming (DP) and Temporal Difference (TD), emphasizing computational complexities. In Section 3.4, experimental insights are provided, including case studies of robot navigation in a grid and mission planning for a drone. These experiments provide insight into the application and adaptability of MDPs in real-world scenarios, with a detailed comparison of solution methods. The Section 3.5 concludes by summarizing key concepts and drawing conclusions from the experimental results.

3.1 General concepts related to Machine Learning

3.1.1 Introduction to Machine Learning

Definition 1. ***Machine Learning (ML)*** *Machine learning is the domain of study that provides computers the ability to learn without being explicitly programmed to perform the desired action, which means a program that improves its performance at some task through experience without programming the modifications [Mit97].*

In other words, a learning algorithm aims to learn in an interactive environment, through experience E, to perform tasks T or solve a problem efficiently. Image recognition is one of the most significant applications of Machine Learning used to classify and identify objects or elements within digital images. Task T is classifying and identifying objects, and Experience E is the database images (with their class) used to train the model.

3.1.2 Types of Machine Learning

In the literature, the ML algorithms are based on the learning style, model training, and according to the input data (data-set) characteristics (presence /or not of labels in the data set representing the information we want to predict) on which they act.

- **Supervised learning**: In supervised learning, the model is trained on a labeled data set, where each input data point (observation) is associated with a corresponding target or output. The goal is to learn a mapping from inputs to outputs so the model can make predictions or classifications on new, unseen data. The supervised

learning algorithm measures its accuracy through the loss function, adjusting it until the error has been sufficiently minimized [Nas17].

Supervised learning can be classified into two types of problems in data mining: classification and regression. In classification, the label is discrete, while in regression, the label is continuous.

- ➥ **Classification**: map the input space into predefined classes. The Common classification algorithms are linear classifiers, Support Vector Machines (SVMs), Decision Trees (DTs), and Random forests,
- ➥ **Regression**: map the input space into a real-value domain. It is used to understand the relationship between dependent and independent variables. The common regression algorithms are Linear regression and Logistic regression.

❖ **Unsupervised learning**: unlike supervised learning, data observations (data-set outputs) are not labeled in unsupervised learning, and algorithms must autonomously extract relationships between them by finding patterns, structure, or relationships within the data [Gha03].

Unsupervised learning is often used for data exploration, anomaly detection, clustering, dimensionality reduction, and generative modeling.

❖ **Reinforcement Learning**: focuses on developing computational algorithms and models for autonomous agents to learn to make sequences of decisions in an environment to maximize a cumulative reward signal [Glo00].

3.1.3 Reinforcement Learning

Definition 2. ***Reinforcement Learning (RL)*** *constitutes a comprehensive category of learning problems intrinsic to autonomous agents interacting in an environmental context. These problems manifest as sequential decision-making scenarios wherein rewards are subject to delay. Reinforcement learning algorithms are designed to learn a policy, denoting a mapping from states to actions, aiming to maximize the cumulative reward over time [SW11].*

RL involves the agent's ability to learn through trial and error, explore different strategies, and optimize actions to achieve a specific goal or objectives in a constantly dynamic and uncertain environment.

Otherwise formulated, reinforcement learning is learning what to do, i.e., how to map situations to actions to maximize a numerical reward signal. In this approach, the learner is not receiving explicit instructions on which actions to choose; instead, it must explore and experiment to determine which actions lead to the highest rewards.

3.2 Decision Making based on MDPs

3.2.1 Markov Decision Process for decision-making

A discrete Markov Decision Process (MDP) [Put14] models a dynamic process, with the agent observing the environment in each step. It selects an action, performs the selected action (which modifies the environment), and the agent receives a reward depending upon the change. An MDP is formally defined as a **tuple** $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ where:

— $\mathcal{S}$ is a finite set of states s_1,s_2,s_3,...,s_n.

— $\mathcal{A}$ is a finite set of actions a_1,a_2,a_3,...,a_m.

— $\mathcal{T}(s, a, s')$ is the probability [0,1] of moving to state s' when action a is executed in state s.

— $\mathcal{R}(s, a)$ is the immediate reward received after taking action a in state s.

3.2.2 Policy

Policy $\pi : \mathcal{S} \times \mathbb{N} \mapsto \mathcal{A}$ ($\mathbb{N}$ representing the number of states) is a solution of an MDP and is a decision plan that can be invoked at any time within the decision-making process. It determines for an agent at each state s in $\mathcal{S}$, the action a in $\mathcal{A}$ that should be carried out.

The optimal policy π^* is the policy that corresponds to the optimal value function, as in:

$$\pi^* = arg \max_{\pi} V_\pi(s), \quad \forall s \in \mathcal{S}. \tag{3.1}$$

The optimal policy (π^*) is usually the one that maximizes the expected accumulated rewards, V_π, such that it satisfies Bellman's Equation[Bel58], as in:

$$V_\pi(s) = \max_a \{\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') V_\pi(s')\} \tag{3.2}$$

In order to determine the optimal actions and corresponding values, two common algorithms are used: Policy Iteration (PI) and Value Iteration (VI) [Put14].

These algorithms introduce another function, the **state-action pair Q function**. The optimal **Q-function** $Q^*(s, a)$ means that the expected total reward received by an agent starting in s and which picks an action a will behave optimally afterward, as in:

$$Q^*(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') V^*(s'). \tag{3.3}$$

Since $V^*(s)$ is the maximum expected total reward when starting from state s, $V^*(s)$

becomes:

$$V^*(s) = \max_a Q^*(s,a) \quad \forall s \in \mathcal{S}. \tag{3.4}$$

Thus, utilizing Equation (3.1), it is possible to compute the most efficient strategy, as in:

$$\pi^* = arg\,\max_a Q^*(s,a) \quad \forall s \in \mathcal{S}. \tag{3.5}$$

3.2.3 Example of decision making with MDP (racing UAV)

To illustrate the properties mentioned above and the modeling of a decision-making process with MDPs (see Figure 3.1), the control of a racing UAV is given as an example.

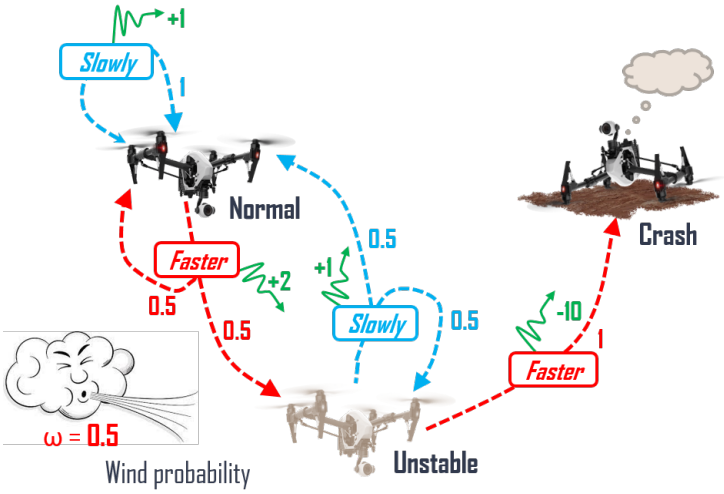


Figure 3.1 – MDP racing UAV control [Ham+21].

The racing UAV has two goals: 1) fly far (to finish the race) and 2) fly quickly (to win the race). It can take three states corresponding to its health state: s_1: *Normal*, s_2: *Unstable* s_3: *Crash*.

The racing UAV is controlled by two actions: a_1: fly *Slowly* or a_2: fly *Faster*. There is a probability ω that a wind poses a risk to the UAV after taking an action. Let's get $\omega = 0.5$, the probability of wind occurrence during a period.

In each period, an action is decided and carried out. If there is wind at the end of a period and the chosen action is to fly *Faster*, the racing UAV will be in a dangerous state (i.e., unstable or crashing), and the state at the beginning of the next period will be $min(s_{i+1}; s_3)$. The other courses of action will be safe.

The transition matrix $\mathcal{T}(s,a,s')$ for this problem can then be defined as follows:

$$\mathcal{T}_{a_{slow}} = \begin{pmatrix} 1 & 0 & 0 \\ 1-\omega & \omega & 0 \\ 0 & 0 & 1 \end{pmatrix}, \mathcal{T}_{a_{Faster}} = \begin{pmatrix} 1-\omega & \omega & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

The reward matrix R(s, a) is defined as follows:

$$\mathcal{R}_{s,a} = \begin{pmatrix} \mathcal{R}_{s_1,a_1} = 1 & \mathcal{R}_{s_1,a_2} = 2 \\ \mathcal{R}_{s_2,a_1} = 1 & \mathcal{R}_{s_2,a_2} = -10 \\ \mathcal{R}_{s_3,a_1} = 0 & \mathcal{R}_{s_3,a_2} = 0 \end{pmatrix}.$$

The racing UAV must find a policy to finish the Fastest in the race. The policy must maximize the γ-discounted reward. Let us assume that $\gamma = 0.95$. Usually, two algorithms are used to solve a problem: policy Iteration or Value Iteration algorithms. The policy Iteration method is usually more efficient than the Value Iteration method because it takes fewer iterations to converge toward the optimal solution. Therefore, it is applied here.

After the convergence of the algorithm, the result of the **Q-function** $Q^*(s_i, a_i)$ is:

$$\mathcal{Q}^*_{s_i,a_i} = \begin{pmatrix} \mathcal{Q}^*_{s_1,a_1} = 14.95 & \mathcal{Q}^*_{s_1,a_2} = 15.50 \\ \mathcal{Q}^*_{s_2,a_1} = 14.50 & \mathcal{Q}^*_{s_2,a_2} = 2.0 \\ \mathcal{Q}^*_{s_3,a_1} = 0.0 & \mathcal{Q}^*_{s_3,a_2} = -10.0 \end{pmatrix}.$$

By using Equation (3.4),

$$V^*(s) = \begin{pmatrix} \mathcal{V}^*_{s_1} \\ \mathcal{V}^*_{s_2} \\ \mathcal{V}^*_{s_3} \end{pmatrix} = \begin{pmatrix} 15.50 \\ 14.50 \\ 0 \end{pmatrix}.$$

From Equation (3.5), the policy that corresponds to the optimal Q-function is: $\pi^* = \begin{pmatrix} a_2 \\ a_1 \\ . \end{pmatrix}$. The agent cannot perform an optimal action in state s_3 because the UAV is in a crashed state.

We note that any change in the transition matrix $\mathcal{T}(s, a, s')$ does not affect the policy, which remains unchanged. If the wind is weak, it will not strongly affect the drone's travel speed. Changing the immediate reward of an action affects the decision-making process and the final policy.

Although rewards are typically randomized, finding a systematic approach for adjusting them remains challenging.

3.3 MDP resolution methods

This Section presents and compares the three most used methods for solving finite Markov Decision Problems based on Dynamic Programming (DP) and Temporal-Difference Learning methods.

It is important to be clear that $iter_{max}$ used in the three methods is the "episodes" and is sometimes called "trials" in the literature.

3.3.1 Dynamic Programming

Dynamic Programming (DP) references a collection of algorithms that can be used to compute optimal policies given a perfect environment model as an MDP. The concept of DP is the use of the Value function (formula (3.2) described in Section 3.2) to organize and structure the search for good policies.

1. **Value Iteration**

   Value Iteration consists of improving the Value Function calculated iteratively using the Bellman equation (3.2) until it converges. The Value Iteration algorithm is shown in Algorithm 1.

Algorithm 1: Value Iteration

Data: Transition and Reward matrices, γ, $iter_{max}$, ϵ
Result: Optimal policy π^*

Initialization $V \in \mathcal{R}$ arbitrarily for all $s \in \mathcal{S}$ (e.g., $V(s) = 0$ for all s)
iter $\longleftarrow$ 0
repeat
    $\Delta \longleftarrow 0$
    Compute new_V using the Bellman operator (and the next policy new_π)
    $\Delta \longleftarrow max(\Delta, new\_V - V)$
    iter $\longleftarrow$ iter + 1
until *((*$\Delta < \epsilon$*) Or (*$iter = iter_{max}$*))*;

   Where $iter_{max}$ is the maximum number of iterations and ϵ is the precision factor that defines the difference between two values function computed in two successive iterations.

   Theoretically, Value iteration requires infinite iterations to converge exactly to V^*. In practice, we stop once the value function changes by only a small amount $\Delta < \epsilon$ or after a sufficiently large iteration number $iter_{max}$ for which the method must give a decision-making policy.

2. **Policy Iteration**

   Policy Iteration consists of evaluating and improving a policy calculated iteratively using the Bellman equation until it converges. The Policy Iteration algorithm is shown in Algorithm 2.

   As for Value Iteration, theoretically, Policy iteration requires infinite iterations to converge exactly to π^*. In practice, we stop once the two successive resulting policies are equal ($new\_\pi \approx \pi$) or after a sufficiently large iteration number $iter_{max}$ for which the method must give a decision-making policy.

Algorithm 2: Policy Iteration

Data: Transition and Reward matrices, γ, $iter_{max}$
Result: Optimal policy π^*

```
Initialization V ∈ R and π(s) arbitrarily for all s ∈ S
iter ⟵ 0
repeat
    Policy Evaluation (Compute of Transition T' and Reward R' matrices of π,
     Then Compute V_π with the new matrices T' and R')
    Policy Improvement (Compute of the next policy new_π) and Compute of
     the new_V using Bellman operator
    iter ⟵ iter + 1
until ((new_π ≈ π) Or (iter = iter_max));
```

3.3.2 Temporal-Difference learning

Temporal Difference (TD) methods can learn directly from raw experience without a model of the environment's dynamics (model-free). They update estimates based on other learned estimates without waiting for the final result.

1. **Q-Learning** is an off-policy TD control algorithm, which means the state-action values function Q converges to the optimal state-action values function Q^* whatever the policy followed. This method was developed by Watkins [WD92], and the recursively form of $Q(s,a)$ is defined by:

$$Q(s,a) \longleftarrow Q(s,a) + \alpha[\,\mathcal{R}(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{3.6}$$

   where α is the learning rate, which determines how Fast the new experience replaces the old ones (or how the agent abandons the previous Q-value in the Q-table for a given state-action pair for the new Q-value). ϵ-greedy is the policy that allows us to set the exploration and exploitation rate for the agent.

   The Q-learning algorithm is shown in Algorithm 3.

   Notice that the $terminal_{state}$ means the state **s** is terminal and has no action to take from it to go to another state **s'**.

   By Watkin [WD92], it has been proven that Q-Learning converges to the optimal policy given "sufficient" updates for each state-action pair, decreasing the learning rate. It means that if each action is executed in each state an infinite number of times on an unlimited run and the learning rate α is decayed appropriately, the Q values will converge to Q^*. To get this balance between exploitation and exploration, we use what is called an ϵ-greedy strategy to find the trade-off. For convergence to occur, the agent must use exploitation while engaging in exploration to avoid being

Algorithm 3: Q-Learning

Data: Reward matrix, α, γ, $iter_{max}$, ϵ-greedy policy (ϵ and ϵ_{decay})
Result: Optimal policy π^*

Initialization $Q(s,a) \in \mathcal{R}$, $\forall s \in \mathcal{S}$, $\forall a \in \mathcal{A}(s)$ arbitrarily and $(Q(terminal_{state}, .) = 0)$
iter $\longleftarrow$ 0
repeat
    Initialize **s**
    repeat
        Choose ***a*** from ***s*** using policy derived from $\mathcal{Q}$ (e.g., ϵ-greedy policy)
        Take action ***a***, observe R, ***s'***
        $Q(s,a) \longleftarrow Q(s,a) + \alpha[\,\mathcal{R}(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
        s $\longleftarrow$ s'
    until *(**s** = $terminal_{state}$)*;
    iter $\longleftarrow$ iter + 1
until *(iter = $iter_{max}$)*;

trapped in local optima.

With this strategy, we define an exploration rate $\epsilon(= 0.99)$ initially (99% the agent will start by exploring the environment). This exploration rate is the probability that our agent will explore the environment rather than exploit it. As the agent learns more about the environment, at the start of each new iteration (episode), ϵ will decay by ϵ_{decay} (= 0,001). So, the exploration likelihood becomes less and less probable as the agent learns. Once the agent has explored and learned, it will use its knowledge.

To determine whether the agent will choose exploration or exploitation at each time step, we generate a random number between 0 and 1. If this number exceeds ϵ, the agent will choose its next action via exploitation (i.e., the action with the highest Q-value for its current state from the Q-table). Otherwise, its next action will be chosen via exploration, i.e., randomly choosing its action (i.e., exploring).

3.3.3 Complexity comparison

Table 3.1 resumes the input data of the three methods and shows their complexity comparison.

Notes:

- ☞ The Bellman*: contains two elementary functions: a Max and a Sum function.
- ☞ $iter_{max}$: Number max of iteration is different between them.

All these parameters and data inputs influence each method's performance and res-

		Method		
		Value Iteration	Policy Iteration	Q-Learning
Input data	T matrix	✓	✓	-
	R matrix	✓	✓	✓
	γ	✓	✓	✓
	α	-	-	✓
	ϵ	✓	-	✓
	ϵ_{decay}	-	-	✓
	$Iter_{max}$	✓	✓	✓
Inner loop operator		Bellman*	Bellman*	Max
Number of loops		1	1	2
Stop condition	$iter = iter_{max}$	✓	✓	✓
	Convergence	✓	✓	-

Table 3.1 – Summary of input data and complexity comparison for the three methods [Ham+21].

olution time. Moreover, there is no method to easily find the best values to tune these parameters for a given application. We see that both methods, Value Iteration and Policy Iteration, have the same input data except the ϵ parameter. However, the Q-Learning method is the most complex; it contains more input data, which are the main parameters.

In Section 3.4.1, we study the effect of these parameters on a scalable case study. In the first part, we used the Matlab MDP toolbox v3.0 [Cha+05] to perform the calculations.

3.4 Experimental experiences

This Section focuses on the case study used to compare the three methods. It shows the effect of the parameters listed in Table 3.1 on a scalable case study. It begins by describing the case study and then gives the evaluation results and their analysis.

3.4.1 Case study 1: moving of a robot in a grid

All the policies presented here are obtained by solving each grid's corresponding MDP. The experiments were conducted with a computer with 2 processors, Intel Xeon Gold 5122 CPU @ (3.60 GHz and 3.59 GHz) and 64 GB of RAM.

1. **Grid for robot navigation**

   (a) **Generic regular grid**

   In this case, to quantify the number of iterations and the execution time of each method, also to determine the most efficient one, we implement in Matlab the decision-making problem which generates the grid (Figure 3.2) of size

$s \times s$ squares where $s \in [4, 55]$. So, the number of states ranges from 16 to 3025.

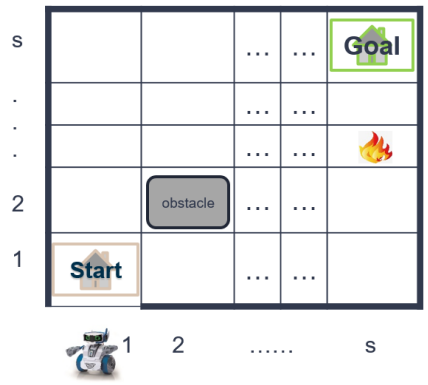


Figure 3.2 – Generic grid of size $S = s \times s$ [Ham+21].

(b) **Irregular grid**

In this case, the number of actions in each state is increased in a synthetic way to see the effect of dissimilarity of actions in the decision-making process. We consider a grid with 16 cases (states) and then create five action classes. In each class, we found the four actions needed to move in the grid (Up, Down, Left, and Right) and other miscellaneous actions as mentioned in Figure 3.3. As shown in Figure 3.4, each state randomly assigns a class.

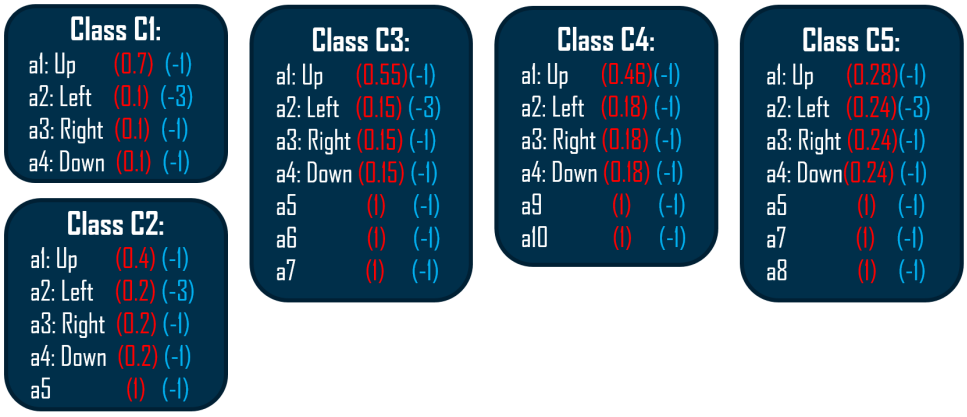


Figure 3.3 – Actions classes for the irregular case with miscellaneous actions (the probability of transition in red, the rewards in blue) [Ham+21].

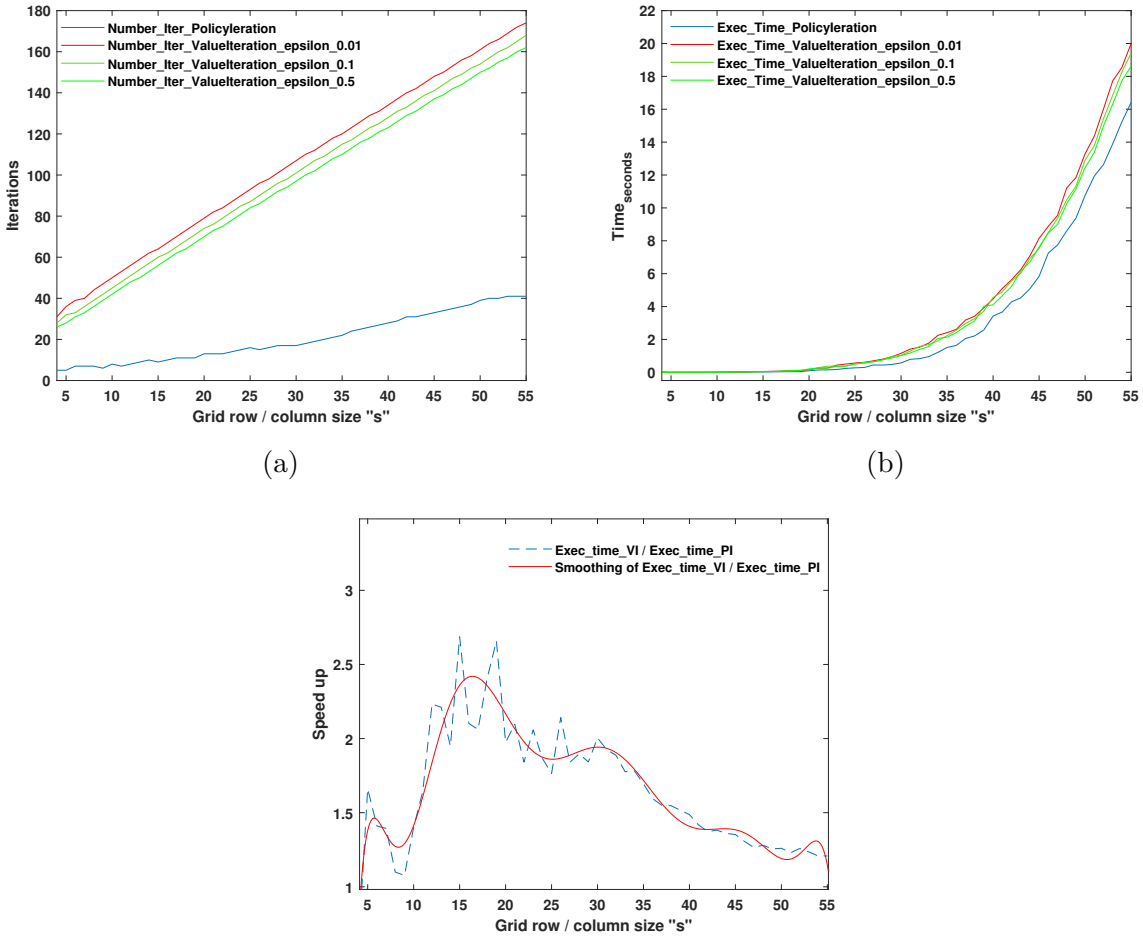
2. **Experiments setup and parameters tuning**

(a) **Selection of the optimal ε-value:**

Unlike Policy Iteration, Value Iteration has one more parameter: ϵ. As it is used in the stop condition of this algorithm, it has to be chosen carefully. The best candidate is a value for which the best policy is found with the lowest number

Figure 3.4 – Classes assignments to grid states [Ham+21].

of iterations. To find this value, we perform tests on the generic regular grid for three ϵ-values: $\epsilon = 0.01$, $\epsilon = 0.1$, and $\epsilon = 0.5$ to evaluate the best solution in terms of execution time as shown in Figure 3.5.



(a) (b)

(c)

Figure 3.5 – Comparison Value Iteration and Policy Iteration [Ham+21]. (a) The number of iterations for size 4×4 to 55×55. (b) Execution time for size 4×4 to 55×55. (c) Speed up of Value Iteration / Policy Iteration.

In Figure 3.5a, the number of iterations of the Value Iteration method decreases slightly by increasing the value of the ϵ parameter e.g., for the size 55×55 the number of iteration is 174 for ($\epsilon = 0.01$), 168 for ($\epsilon = 0.1$) then 162 for ($\epsilon = 0.5$). Also, as shown in Figure 3.5b, the higher the ϵ-value is, the lower the execution time is. For a 55×55 grid, the execution time is 20 seconds (resp. 19.42 and 18.62 seconds) for an ϵ-value equals 0.5 (resp. 0.1 and 0.01). So, the increase of the ϵ parameter reduces the number of iterations needed to find the best policy and its execution time (about $\approx 6\%$ of decrease). This reduction in the number of iterations of the method implies a loss of precision in the Value functions of the values table, which means that the Value Iteration method may converge towards a non-optimal policy. As all the ϵ-values are very low regarding the value of the rewards, the risk that the algorithm does not return the best policy is also very low. In the rest of the document, we consider ($\epsilon = 0.01$) for more precision in resolving the problems using the Value Iteration method.

Speed up between the two DP methods

The curves of the speed up in Figure 3.5c are obtained by calculating the average of five executions of the resolution of the problems modeled with the grids (size 4×4 to 55×55). We solve each grid's corresponding MDP five times using Value Iteration and Policy Iteration. Then, we compute the average time needed to solve the MDP. Finally, to evaluate the speed up, we divide the $Exec\_time\_ValueIteration$ (average of execution time using Value Iteration) by $Exec\_time\_PolicyIteration$ (average of execution time using Value Iteration), then we plot the result. We can notice that the speed-up is strictly greater than 1, and almost 70% of the time, the speed-up is greater than 1.5, and 20% of the time, it is greater than 2, this confirms that Policy Iteration is the most efficient.

(b) **Variants of the DP/TD methods**

To compare the three methods, the maximum number of iterations for each one of them is set to 1,000,000, i.e., $iter_{max} = 1,000,000$. As the goal to achieve is known, the convergence condition of the Q-Learning algorithm can be modified to stop when the goal state is reached, as shown in Table 3.2. This new version of the Q-Learning algorithm may not return the best policy, but it will significantly improve its search time.

Two stop conditions will be studied:

i. **Version 1**: Value Iteration and Policy Iteration methods find the optimal policy that guarantees a path from any system state to the *Goal state*. For the Q-Learning method, the stop condition cc_1 is added to the algorithm.

A. **Version 1-a** initializes the first state of the search to the Start state (square(1,1) of the grid),

B. **Version 1-b** randomly selects the first state of the search.

ii. **Version 2**: to find a path from the the *Start state* to the *Goal state*, we added the condition cc_2 to the three methods.

In brief, all the parameters used during the experiments are presented in Table 3.2.

		Method		
		Value Iteration	Policy Iteration	Q-Learning
Generic regular grid	Ver. 1-a	-	-	cc_1, $Init_s$
	Ver. 1-b	-	-	cc_1, $Init_r$
	Ver. 2	cc_2	cc_2	cc_2, $Init_s$
Irregular grid	Ver. 1-a	-	-	cc_1, $Init_s$
	Ver. 1-b	-	-	cc_1, $Init_r$
	Ver. 2	cc_2	cc_2	cc_2, $Init_s$

Table 3.2 – Summary of the parameters to tune in the three methods comparison [Ham+21].

Notes:

☞ Ver. ⇒ version.

☞ cc_1 (condition of the convergence 1)⇒ ***Find a path from any state*** s ***of the grid to the Goal state.***

☞ cc_2 (condition of the convergence 2)⇒ ***Find a path from Start state (square(1,1)) of the grid to the Goal state.***

☞ $Init_s$⇒ Initialization of the initial state (1^{st} instruction) to the start state (square(1,1) of the grid).

☞ $Init_r$⇒ Initialization of the initial state (1^{st} instruction) randomly.

3. **Comparison between the DP and TD methods**

(a) **Generic regular grid**

i. **Version 1:** in this version we solve MDPs using three methods. The results are shown in figures 3.6 and 3.7. We can notice that:

☞ Policy Iteration method is more efficient than Value Iteration for any version 1 (Versions 1-a or 1-b).

☞ For version 1-a (see Figure 3.6a), the Q-Learning method fails to converge over the size $S = 8 \times 8 = 64$ states and reaches the $iter_{max} = 1,000,000$ iterations. Its execution time increases exponentially.

☞ For version 1-b, the Q-Learning method fails to converge in most sizes and reaches the $iter_{max} = 1,000,000$ iterations due to the random initialization of the state s (see Figure 3.7a). Its execution time increases exponentially in the same way as in version 1-a.

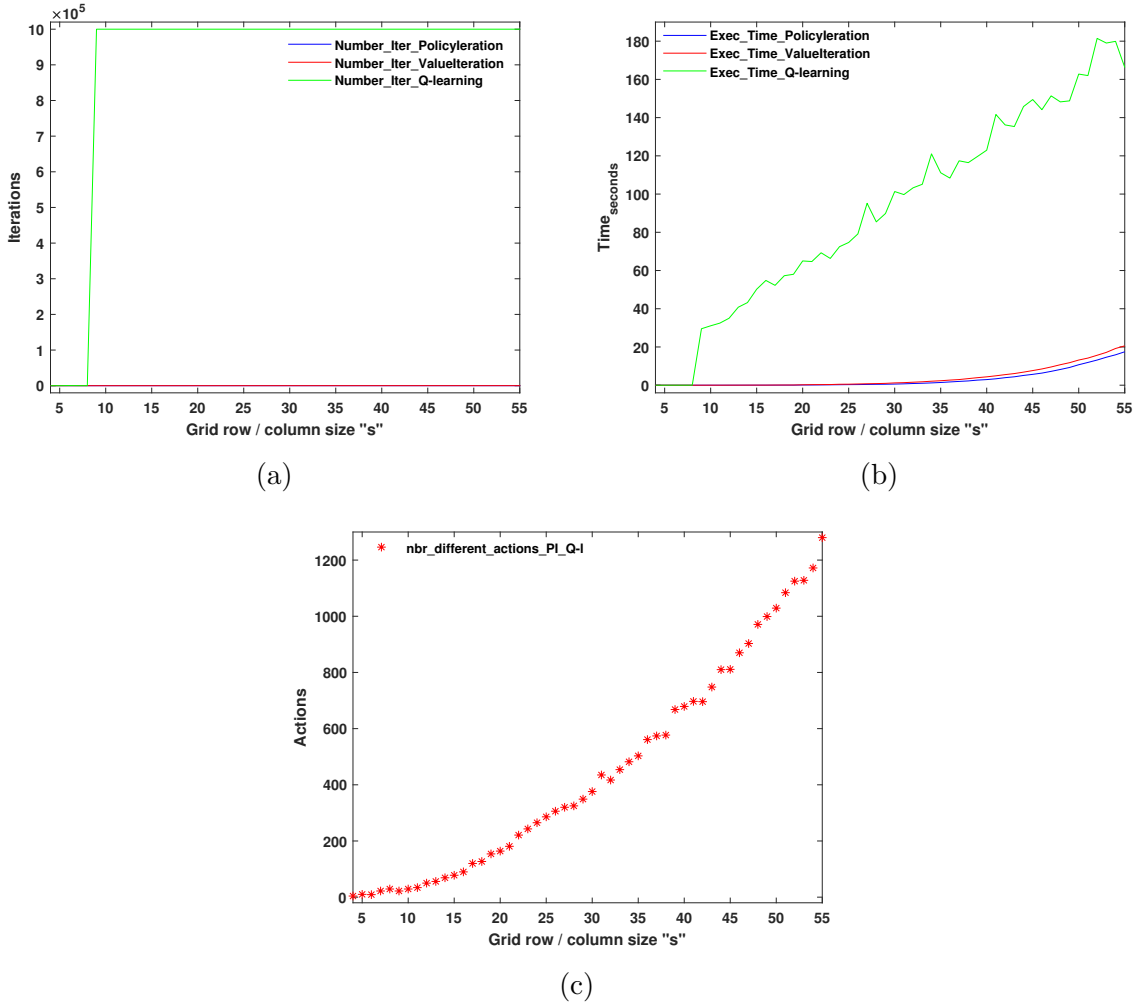
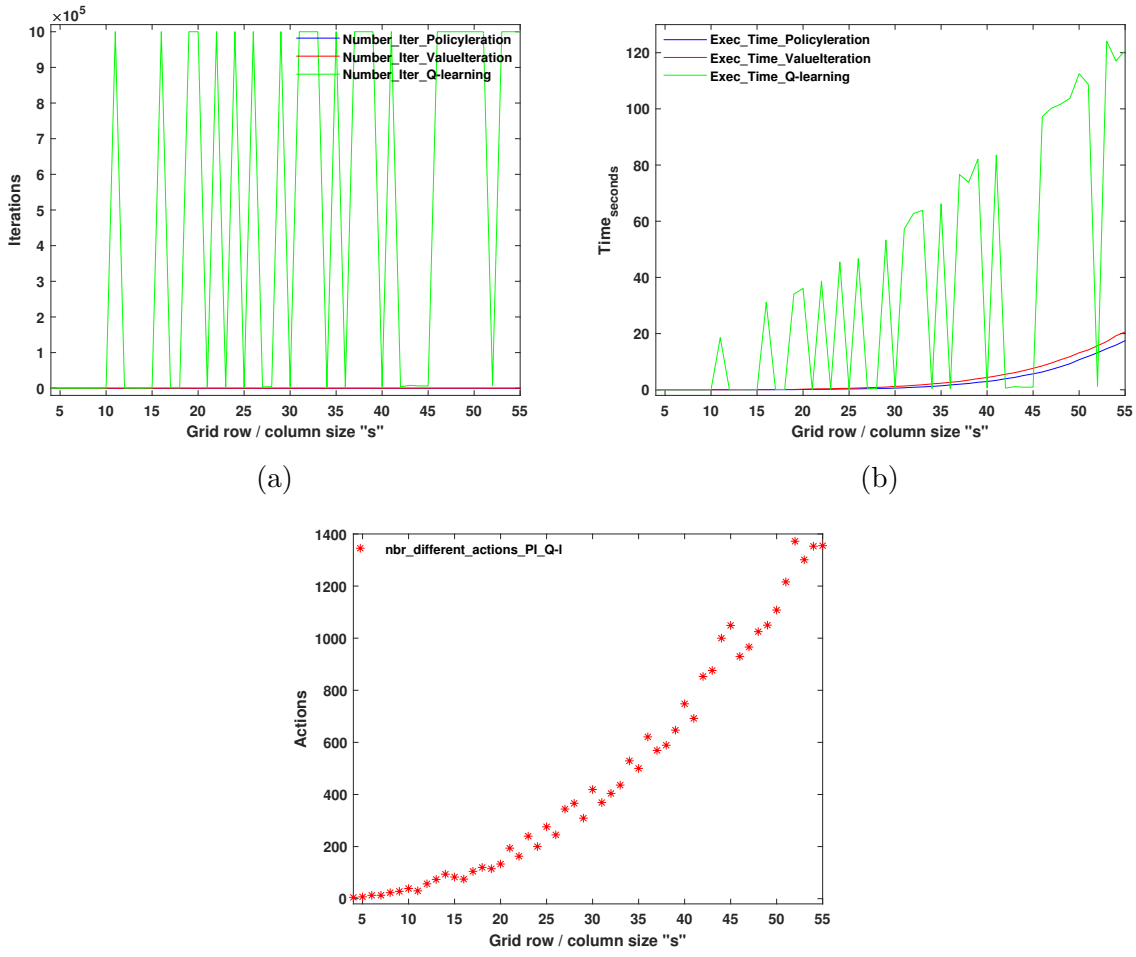


Figure 3.6 – Comparison version 1-a. (a) Number of iterations for size 4×4 to 55×55. (b) Execution time for size 4×4 to 55×55. (c) Number of different actions between VI and Q-l for size 4×4 to 55×55.

We conclude that the search for a global optimal policy for a problem is more practical using the DP methods (Value Iteration method or Policy Iteration method). The Q-Learning method is inefficient in terms of execution time because it needs a lot of time to explore the state and action space before converging. So, for this type of simple scenario, Policy Iteration is the best candidate to find the best set of actions to reach the goal state.



(a) (b)

(c)

Figure 3.7 – Comparison version 1-b [Ham+21]. (a) Number of iterations for size 4×4 to 55×55. (b) Execution time for size 4×4 to 55×55. (c) Number of different actions for size 4×4 to 55×55.

ii. **Version 2:** in this version, a condition of the convergence cc_2 is added to Value Iteration, Policy Iteration and Q-Learning methods. We solve the MDPs using the three methods; the results are shown in Figure 3.8.
We can notice that:

- ☞ The number of iterations of the Q-Learning method increases exponentially and exceeds that of the Value Iteration method and Policy Iteration method (Figure 3.8a).
- ☞ Concerning the execution time (see Figure 3.8b), if the size of the problem is less than $S = 11 \times 11 = 121$ states, their efficiency is approximately the same, their execution times are at 1.5 millisecond scale $\pm$ 0.5 millisecond. Beyond this size up to the size $S = 39 \times 39 = 1521$ states, Q-Learning and Policy Iteration have the same efficiency and are better than Value Iteration. Beyond this size ($S = 39 \times 39 = 1521$), the Q-Learning method is the best.

We conclude that the Q-Learning method is more efficient than Dynamic Programming methods (Value Iteration method or Policy Iteration method) to find a path from the *Start state* to the *Goal state*. This fast convergence of Q-Learning is attributed to its intrinsic balance between exploration and exploitation during learning. By initiating the initialization of the initial state (the first instruction of the algorithm) at the starting state (square (1,1) of the grid), it converges more rapidly. On the other hand, DP methods cannot incorporate the initialization of a state into their algorithms.

Speed up of MDP methods for generic regular grids

The speed-up comparison curves between Q-Learning and Value Iteration (or Policy Iteration) in Figure 3.9 are obtained with the same methodology used to obtain the speed-up between Value Iteration and Policy Iteration in Figure 3.5c. In Figure 3.9a and Figure 3.9b, the speed-up of Q-Learning compared to Policy Iteration is notably higher than the speed-up of Q-Learning compared to Value Iteration. Additionally, it is strictly greater than 1, affirming that Policy Iteration is the more efficient method. In Figure 3.9c, the speed-up of Q-Learning compared to Policy Iteration exceeds 1 when the problem size is less than 40×40. For these smaller grids, the time required to solve the problem (search for a path) is very brief, typically in the order of a few milliseconds. Hence, their efficiency is almost the same. When the size of grids is greater than this size, the curve is under 1; this means that, in that case, the Q-Learning is the better one. Thus, we can conclude that the Q-learning method is more efficient for searching a path.

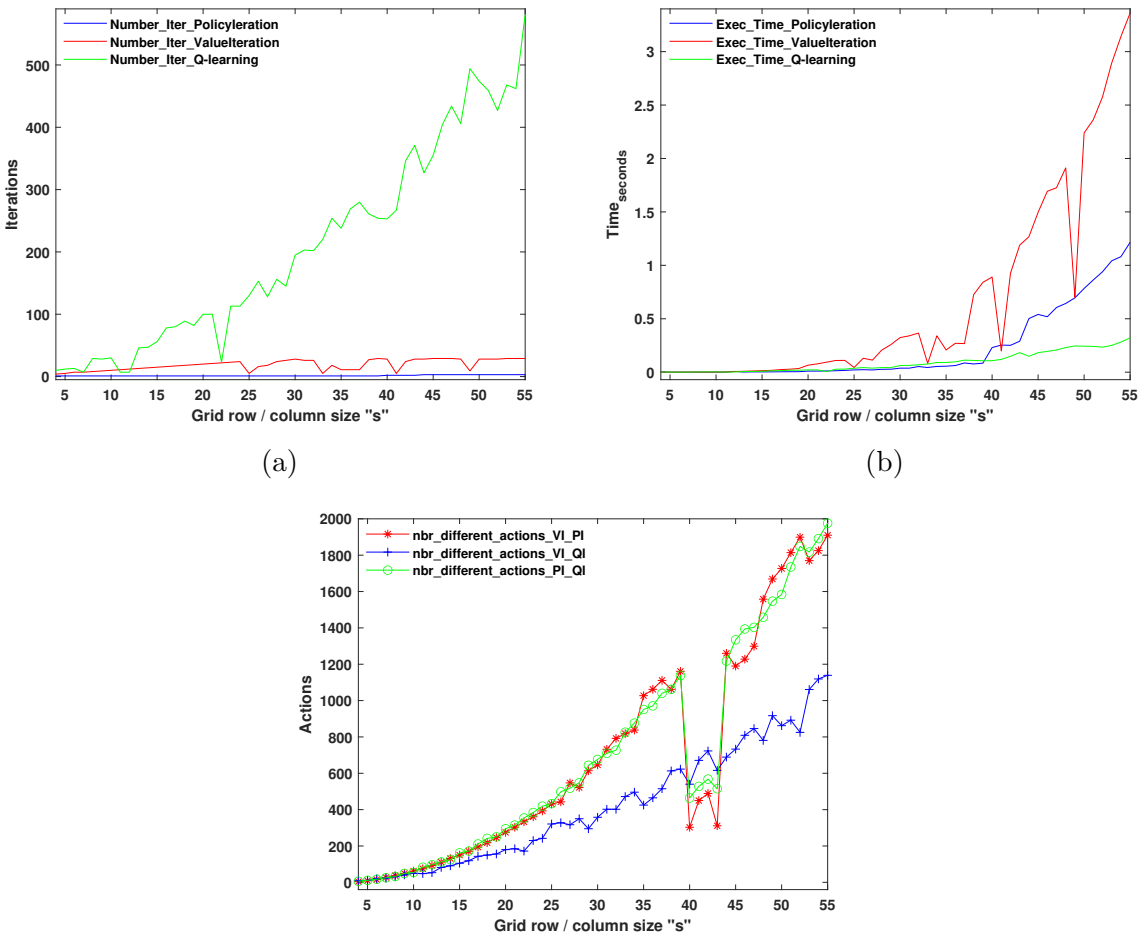


Figure 3.8 – Comparison version 2 [Ham+21]. (a) Number of Iterations for size 4×4 to 55×55. (b) Execution time for size 4×4 to 55×55. (c) Number of different actions for size 4×4 to 55×55.

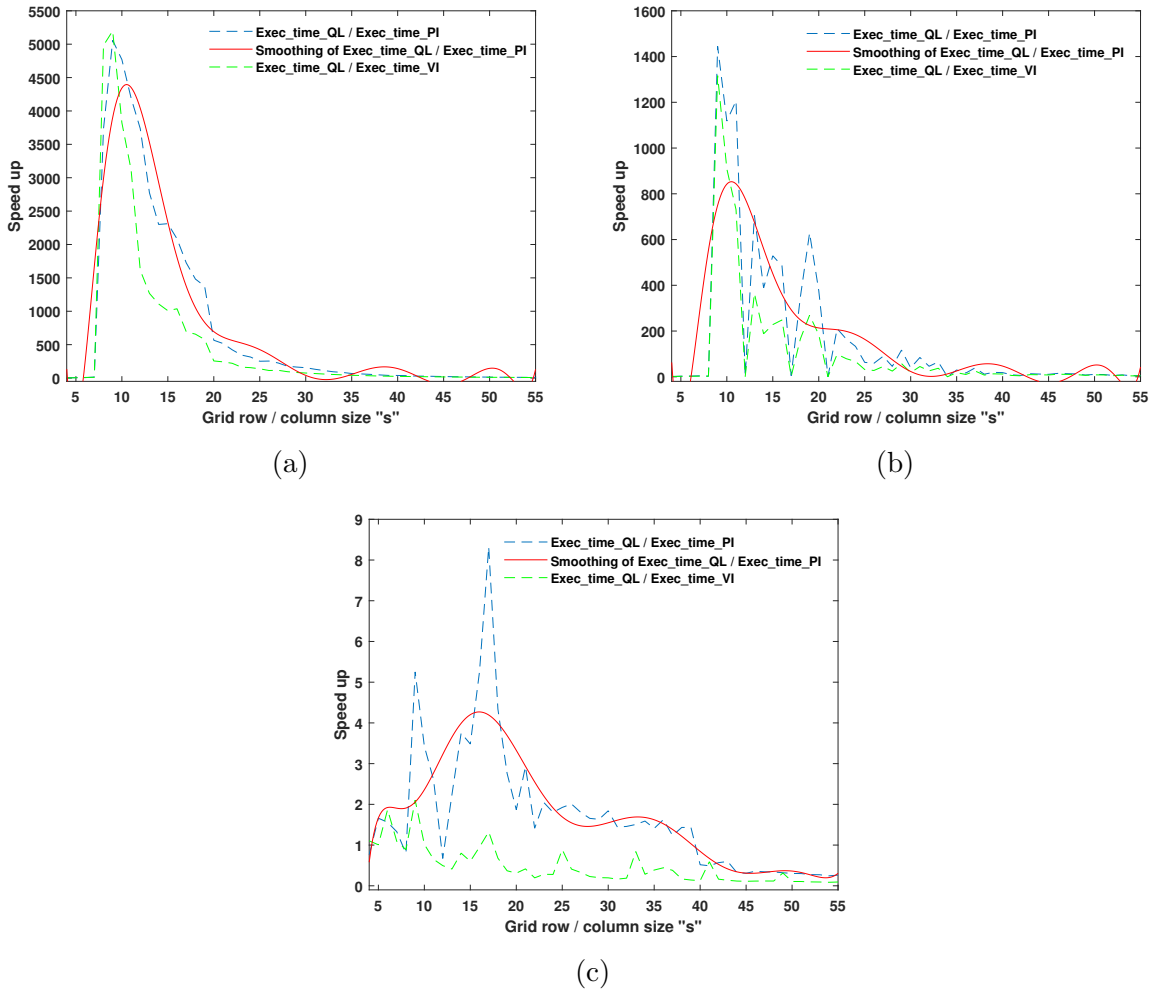


Figure 3.9 – Speed up between the methods for regular grids [Ham+21]. (a) Version 1-a. (b) Version 1-b. (c) Version 2.

(b) **Irregular grid**

To see the effect of non-similarity (irregularity) of actions in each state, a grid with 16 states is considered. Each state is assigned a class of actions (shown in Figure 3.3). The result of 100 executions of the three methods for each version to solve the problem described in this Section is resumed in the following tables (Table 3.3, Table 3.4).

100 Executions		Version 1-a			Version 1-b		
		Value Iteration	Policy Iteration	Q-Learn.	Value Iteration	Policy Iteration	Q-Learn.
Iterations	$Iter\_{min}$	1,375	3	1,000,000	1,375	3	6
	$Iter\_{max}$	1,375	3	1,000,000	1,375	3	1,000,000
	$Average\_{Iter}$	1,375	3	1,000,000	1,375	3	890,009
Time	$Exec\_time\_{min}$	0.03433	0.00050	11.21280	0.03444	0.00032	0.00012
	$Exec\_time\_{max}$	0.07609	0.16397	14.00078	0.05924	0.01831	10.42652
	$Average\_{Exec\_time}$	0.03608	0.00232	11.48608	0.03676	0.00081	7.25192
Speed up Q-l. / MDP Methods		318	4951	1	197	8953	1

100 Executions		Version 2		
		Value Iteration	Policy Iteration	Q-Learn.
Iterations	$Iter\_{min}$	1,375	3	11
	$Iter\_{max}$	1,375	3	1,000,000
	$Average\_{Iter}$	1,375	3	855,015
Time	$Exec\_time\_{min}$	0.03474	0.00032	0.00029
	$Exec\_time\_{max}$	0.05743	0.02480	10.35899
	$Average\_{Exec\_time}$	0.03801	0.00091	7.36136
Speed up Q-l. / MDP Methods		194	8089	1

Table 3.3 – Summary of 100 executions of the three methods of each version to solve the irregular grid [Ham+21].

100 executions			Version 1-a	Version 1-b	Version 2
Number of different actions over 13 states/actions	VI Vs. PI	Min	0		
		Max	0		
		Average	0		
	(VI / PI) Vs. Q-l	Min	7		
		Max	7	13	
		Average	7	7.47	8.03

Table 3.4 – Summary of 100 executions of the three methods of each version to solve the irregular grid (suite) [Ham+21].

We notice that the Q-Learning method often fails to converge even for the size $iter_{max} = 1,000,000$ iterations in each version. So, its execution time is very long compared to DP methods. With DP, the problem is solved in a few milliseconds. With Q-Learning, the problem is solved in 7 seconds in versions 1-b and 2 and 11 seconds in version 1-a.

Following this illustration with the irregular grid, it can be figured that the efficacy of the Q-Learning method decreases in scenarios characterized by highly irregular state-action pairings.

The examples shown in this Section were explored using the context of a ground vehicle. The Q-learning method was evaluated for an irregular number of actions per state for a grid. In the next section, we present a mission plan for an aerial vehicle, representing a concrete set of irregular actions by the state for the decision-making method in the navigation system in the UAV. This scenario brings new challenges and allows a better comparison and understanding of the studied methods.

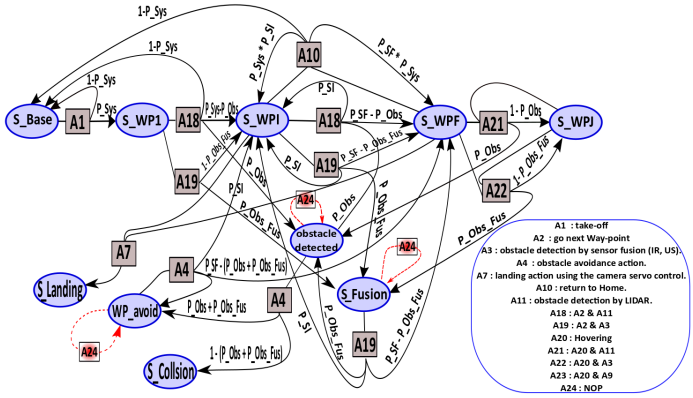
3.4.2 Case study 2: mission planning of a UAV

Navigation is the foundation of mobile robots. In order to plan its path and successfully move in the environment, a robot must know its position in it. Most applications for mobile robots depend on correct localization for path planning and navigation. The navigation system usually obtains the UAV position and altitude, which uses data from a range of sensors, like Inertial Measurement Unit (IMU), Global Positioning System (GPS), and others. Because of complementary characteristics, this information can be improved when sensor data are fused, especially from IMU and GPS.

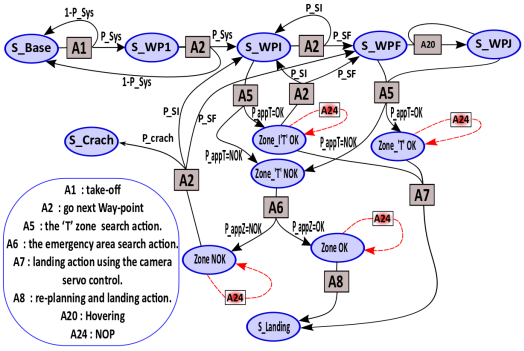
Path-following is one of the most important vehicle navigation tasks, allowing the aircraft to follow a predefined path. Several approaches were created to solve this problem, usually based on geometric and control techniques [SSS14].

In this case study, we consider the tracking mission of [Hir+18; HDB20] as a scenario where the UAV follows a predefined trajectory (set of Way-points). Upon arrival at the search area, the UAV tracks the potential target when it is detected. The tracking mission is defined by means of three main MDPs (Navigation, Landing, and Tracking) that guarantee the success of the tracking mission. In the MDP Navigation (Figure 3.10a), we find the actions of navigation and safety. We find the actions required for landing in the MDP Landing (Figure 3.10b). In the MDP Tracking (Figure 3.10c), we find the actions for the tracking mission related to different algorithmic versions of the tracking application. The actions (A) are briefly described in Figure 3.10. These three MDP problems are resolved using value iteration, policy iteration, and Q-Learning to compare the most efficient methods to solve complex MDP problems.

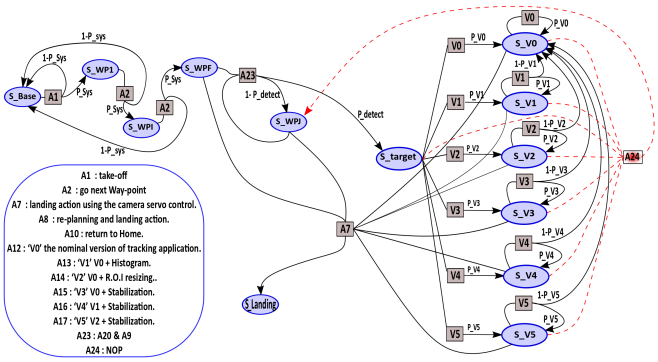
In the previous part 4.1.3, we added a convergence condition to the Q-Learning method because we know the state of arrival. But in this specific case study, the MDPs have several states of arrival (stop), and the fact of being in a state depends on several probabilities (transition probabilities), which are related to the environment. Also, the actions are not



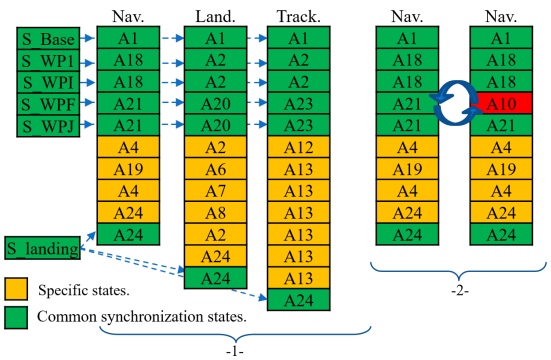
(a) MDP Navigation.



(b) MDP Landing.



(c) MDP Tracking.



(d) -1- Optimal policies of the three MDPs computed using DP Methods . -2- Policies of the MDP Navigation computed using the Q-Learning method.

Figure 3.10 – MDPs that constitute the tracking mission of a UAV and their optimal policies using DP/TD Methods [Ham+21].

similar in each state. So, we cannot specify the future states according to the current state and the action. However, we need the probabilities, which is a big challenge compared to the previous case study of grids dealing with ground vehicles. We take here the probabilities defined in [HDB20] and briefly shown in Figure 3.10.

The resolution of the three MDPs of the tracking mission using the DP methods (Value Iteration or Policy Iteration) and the TD method (Q-Learning) gives us these three policies as shown in Figure 3.10d. Since the time of resolution varies from one execution to another, we decided to execute the resolution 300 times for each method and to calculate their averages. These averages are given in Table 3.5.

Execution time in seconds			Speed up V.I./DP Method
Using V.I.	Average($T\_Parallel$)	0.0045	1
	Average($T\_Sequential$)	0.0069	1
Using P.I.	Average($T\_Parallel$)	0.00041	11
	Average($T\_Sequential$)	0.00078	9

Table 3.5 – Summary of 300 executions of the Dynamic Programming methods to solve the decision-making problem of a UAV mission planning [Ham+21].

where:
$T\_Parallel = max(T\_Nav, T\_Land, T\_Track)$, i.e., the time needed for resolution parallel execution of the tree MDPs,
$T\_Sequential = (T\_Nav + T\_Land + T\_Track)$, i.e., the time needed for resolution in sequential execution of the tree MDPs,
T_Nav is the time needed for resolution of the MDP Navigation,
T_Land is the time needed for resolution of the MDP Landing,
T_Track is the time needed for resolution of the MDP Tracking.

Execution time in seconds			Speed up Q-l. / MDP Method
Using V.I.	Average(T_Nav)	0.0018	7.8
			1052
Using P.I.	Average(T_Nav)	0.00019	73.7
			9967
Using Q-l	Average(T_Nav) $iter_{max} = 1,000$	0.0140	1
	Average(T_Nav) $iter_{max} = 100,000$	1.8937	1

Table 3.6 – Summary of 300 executions of the Dynamic Programming methods and the Q-Learning method to solve the decision-making problem of a UAV mission planning (MDP Navigation only) [Ham+21].

In the Table 3.5, we can notice that:

- ☞ If we solve the three MDPs sequentially in average we need $T\__{Sequential} \approx 0.0069\,\text{s} \approx 7\,\text{ms}$ using Value Iteration method or $T\__{Sequential} \approx 0.000\,78\,\text{s} \approx 0.8\,\text{ms}$ using Policy Iteration method.
- ☞ But if we solve the three MDPs simultaneously in average we need $T\__{Parallel} \approx 0.0045\,\text{s} = 4.5\,\text{ms}$ using Value Iteration method or $T\__{Parallel} \approx 0.000\,41\,\text{s} \approx 0.0004\,\text{s} = 0.4\,\text{ms}$ using Policy Iteration method.

In Table 3.6 concerning the resolution of the mission planning problem using the Q-Learning method and, as we said previously, we cannot add another condition of convergence for the MDPs. For this, we choose the complex one only (the MDP Navigation).

Firstly, we apply the native Q-Learning method to the MDPs navigation. The policies found are unrealistic (e.g., the optimal action associated with the state S_base is $A4$: "obstacle avoidance action", which cannot be executed if the UAV is not flying). The solution is to restrict the choice of possible actions from a given state (instruction 6 of the algorithm 3). After adding the restriction to the Q-Learning method, we apply it to the MDP Navigation in two ways, using 1,000 and 100,000 iterations.
We can notice that:

- ☞ When we solve the MDP Navigation using the Q-Learning method where $iter_{max} = 1,000$ iterations, in average, we need $T\__{Nav} \approx 0.0142\,\text{s} \approx 14.2\,\text{ms}$. When we solve the MDP Navigation using Q-Learning method where $iter_{max} = 100,000$ iterations, in average we need $T\__{Nav} \approx 1.8937\,\text{s} \approx 1894\,\text{ms}$.
- ☞ We can see in Figure 3.10d-2- even if we restrict the possible actions in each state, using ($iter_{max} = 1,000$) iterations or $iter_{max} = 100,000$ iterations, sometimes the method converges to the right optimal policy. Sometimes, the policy found by the Q-Learning method is different in a state-action that does not converge towards the optimal policy (found using DP methods). Even with the restriction for selecting the actions in each state and the fact that the rewards are given, the Q-Learning method fails to converge. Thus, Q-Learning needs more iterations to explore the state and action spaces. Since the Q-Learning method takes 9967 times longer than the Policy Iteration method with $iter_{max} = 100,000$ iterations, we have stopped the simulations at $iter_{max} = 100,000$.

We conclude that:

- ☞ If the three MDPs are solved sequentially, the Value Iteration method's resolution time is $\cong 9\times$ the resolution time using the Policy Iteration method.
- ☞ If the three MDPs are solved simultaneously, the Value Iteration method's resolution time is $\cong 11\times$ the resolution time using the Policy Iteration method.

- ☞ If the MDP navigation is only solved using the Q-Learning method with $iter_{max} = 1,000$, its resolution time is $\cong 8\times$ (resp. is $\cong 74\times$) the resolution time using Value Iteration method (resp. Policy Iteration method).
- ☞ If the MDP navigation is only solved using the Q-Learning method with $iter_{max} = 100,000$, its resolution time is $\cong 1052\times$ (resp. is $\cong 9967\times$) the resolution time using Value Iteration method (resp. Policy Iteration method).

As mobile robot missions, especially those of UAVs, contain very irregular actions, they depend on the state where the robots are at a given time. Also, it contains uncertain events related to the probability that can alter the mission from one moment to another; the MDPs are much more interesting for this type of mission with irregular actions.

3.5 Final consideration

As robots should be more autonomous during this century, they must be able to perceive their environment and make decisions under uncertainty. This chapter focuses on the MDP framework and on the three fundamental methods for solving it, which are Value Iteration and Policy Iteration, which are DP methods, and Q-Learning, which is a temporal-difference method. We give some parameters to choose a particular method to solve a given decision-making problem. As we can observe in the case study focused on ground vehicles, when the actions are similar in each state of the problem, and the problem is searching a path from point A to point B, the Q-Learning method is more efficient than the DP methods. As we can observe in the case of mission planning, using aerial vehicles, which are more sophisticated and irregular states, when actions are non-similar in each state of the problem, it is more difficult to set up the Q-Learning method, so DP methods are more efficient. We have shown that the regularity of actions by step of decision can be an element of consideration for choosing the resolution method. Depending on the type of actions of the mission planning and the number of states of a mission, we will choose one method over the others, particularly for on-board critical decision-making.

CHAPTER 4

Self-adaptation based rewards tuning for mission planning

Contents

Introduction

This chapter explores applying rewards adaptation techniques to address multi-UAV mission planning constraints and self-adaptation of each UAV in a collaborative mission.

The research applied to two levels of adaptation: the multi-UAV level and the individual UAV level. In Section 4.1, the examination at the multi-UAV level focuses on identifying and resolving conflicts and constraint violations within the mission planning framework. This investigation sets the stage for evaluating the proposed rewards adaptation approach through a case study involving a UAV mission, and the ensuing results are subsequently presented. Section 4.2 further addresses the mission of an individual UAV by employing multiple MDPs. This approach is beneficial for managing complex missions by decomposing the problem into subproblems. The resolution of these subproblems involves the execution of several MDPs in parallel, enhancing the adaptability and efficiency of the overall mission strategy. A case study involving the mission planning of an individual UAV is presented, and the corresponding results are discussed. In Section 4.3, simulations are conducted using the Coppeliasim robot simulation platform and MATLAB to validate the proposed methodology. Section 3.5 summarizes the chapter's key findings, highlighting the potential improvements in multi-UAV mission planning achieved through rewards adaptation for constraints management.

4.1 Rewards adaptation for constraints management at multi-UAV level

In a swarm, multiple MDPs are executed. Each UAV has its own MDP to decide its best course of action. As the UAV decision is independent of the others, conflicts can arise at the swarm level, and the chosen actions can contradict the mission objectives. This results in the need for a method to solve these conflicts. In the first subsection, we introduce the conflicts and constraint violations, the priority assignment when conflicts occur in distributed decision-making based on MDP at a swarm level, and finally, the method to detect these conflicts. In the second subsection, we present the core of the reward adaptation method, the method for resolving conflicts at swarm levels, and the mission management plan.

In our study, "experts" associated with the central authority refers to the decision-makers (operational monitoring systems experienced in the missions under consideration) who express their preferences via different priorities, parameter thresholds, and actions that can resolve conflicts.

4.1.1 Detection of conflicts and constraint violations

As multiple UAVs within a swarm execute individual MDPs, their independent decision-making processes may lead to conflicts at the swarm level, potentially contradicting mis-

sion objectives. This subsection introduces conflicts and constraint violations, describes the prioritization of UAV-related decisions in distributed decision-making, and explains the method used for conflict detection. Finally, we end this subsection by implementing a conflict management mechanism.

1. **Conflicts and constraints**

   When decision-making is distributed between UAVs of the swarm, conflicts may arise among the UAVs. These can occur within each member due to non-compliance with external constraints or at the swarm level.

   - **Hardware failure conflicts**

     UAVs can be subject to hardware failures (network malfunction and loss of communications, system/autopilot malfunction, component failure) that can lead to conflicts with other UAVs in the swarm. This can occur when the UAV with hardware failures has specific and relevant functionality in the swarm. For example, cameras with temperature sensors are expensive; they are integrated only in some swarm UAVs. When the UAV with this camera breaks down, it will influence the swarm's performance.

   - **Conflicts related to performance limitations**

     UAVs can suffer from performance limitations, such as speed, range, and flight duration, leading to conflicts when trying to accomplish similar tasks. For example, in a search and rescue mission, one of the UAVs is more efficient than the others according to one or more criteria such as energy self-sufficiency, wind resistance, and a more interesting payload to carry.

   - **Trajectory conflicts**

     Trajectory conflicts arise when multiple UAVs navigate similar routes or intersect paths, depicted as nodes and links in the control network. It is essential to avoid scenarios where multiple UAVs are concurrently tracking a target to prevent collisions within the swarm and optimize resource utilization. Such situations may lead to overlapping trajectories, increasing collision risk and compromising the expected benefits of efficient swarm coordination.

   - **External conflicts**

     Sometimes, conflicts can happen when UAVs break the rules of where they're allowed to fly. For example, if a UAV goes into an area it's not supposed to or flies too high, it can create a conflict. This might happen because the UAV's navigation system isn't accurate or it doesn't have the latest information about flying rules. Conflicts can also arise due to external factors like bad weather conditions or fog, making it challenging for UAVs to navigate safely.

➥ **Mission conflicts**

These conflicts may arise when mission constraints are violated. For example, exceeding the designated number of UAVs for a specific task or surpassing the allocated number of UAVs for a particular region during the mission can lead to mission conflicts.

2. **UAV priority**

To effectively deal with conflicts, identifying which agents (UAVs) can maintain their mission plan policy and which ones must change them is essential. For this purpose, we can define a priority between the UAVs in the swarm. We can discern two types of priority:

➥ **Static UAV priority**

In this case, the expert defines a UAV priority between the UAVs of the swarm; this priority is determined offline and remains stable throughout the mission.

➥ **Dynamic UAV priority**

In this case, the expert defines thresholds on some probabilities for the transition function or thresholds on other parameters, which makes it possible to adapt the UAV priority in the swarm. During the mission, the members exchange data between them, which will be used to determine the priority of each MDP according to specific criteria.

In real missions, it is more interesting to use the dynamic UAV priority because the performances of the swarm members are different. As the mission evolves, these performances change randomly according to the hazards of the external environment that influences it.

3. **Conflicts management (*Check_Conflicts*)**

The conflicts are checked at two levels: checking the internal conflicts of the UAV and checking the conflicts at the swarm level. The first type of conflict management allows the UAV to manage itself without human intervention during the mission.
The second type of conflict management allows the UAV to behave adequately in a situation that arises in the swarm. In such a case, the conflict is solved by allocating new priorities to the actions of the UAVs. We assume here that a central authority assigns the priorities in the cloud. The decision to choose these priorities is out of the scope of this thesis.
Each UAV computes its policy and checks the conflicts periodically, as shown in Figure 4.1. If the conflict is internal to the UAV, the latter will adapt its mission to resolve it. If the conflict is at the level of one of the members of the swarm, and this one is unable to solve it, it will send the priority to one of the members who will take the relay to achieve the mission.

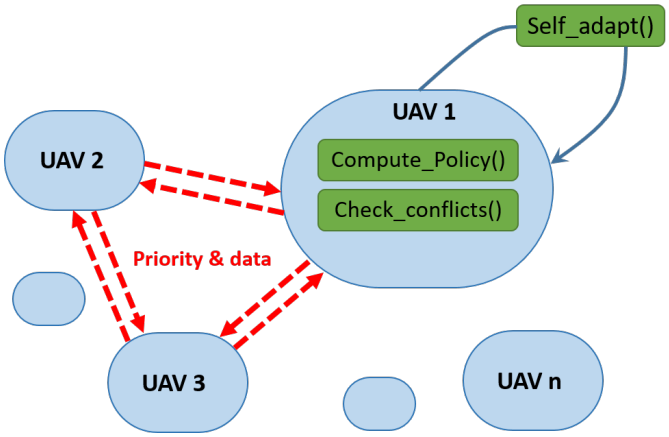


Figure 4.1 – Diagram of self-adaptation between UAVs of the swarm [Ham+23].

Algorithm 4: Check_Conflicts

Input: π : UAV policy, $conflicts$: table of the constraints (c_i) with the corresponding probabilities $P(c_i)$

Output: $Bool_{Conf}$: Boolean indicating conflicts, ST_{Conf}: conflicting states tables

```
Initialize to empty ST_Conf ;
Initialize to false Bool_Conf ;
foreach (s_i, c_k) ∈ S_A × conflicts do
    if π(s_i) = a_{c_k1} then
        ST_Conf (i) ← true;
        Bool_Conf ← true;
```

The *Check_Conflicts()* method (see **Algorithm** 4) assigns a label if there are conflicts on the MDP and defines the corresponding states. This method uses a table of possible conflicting actions (called a *conflicts* Table). This method compares each pair of state s_i actions from the 'conflicts' Table. If so, a Boolean label $Bool_{Conf}$ is used to identify these MDP and the corresponding tables ST_{Conf} and is fulfilled with conflicting states for subsequent resolution. The complexity of this method is $\mathcal{O}(n_s n_c)$, where n_s and n_c are the number of states of the MDP and the number of conflicts in the *conflicts* Table, respectively.

Depending on the mission context (system probabilities/different sensor data) defined as *conflicts*, this MDP and the conflicts are used to identify its conflicting states.

4.1.2 Solving conflicts and constraint violations

After identifying the constraints/conflicts, a priority is assigned to the UAV. We can select the MDP whose policy should be modified based on this priority. In this Section ,

we describe the method for tuning the rewards.

1. **Rewards adaptation principles (Core of the method)**

   This method forces the MDP to change its policy due to action conflicts. As each value of the matrix Q is calculated using Equation (3.3), the Q value of the action with the highest priority is increased. Therefore, the policy is changed to avoid constraint violations, and some actions are prioritized over others.

   The policy is obtained from the matrix Q^* using the Equation (3.5).

$$\mathcal{Q}^* = \begin{pmatrix} \mathcal{Q}_{s_1,a_1} & \mathcal{Q}_{s_1,a_2} & . & . & \mathcal{Q}_{s_1,a_m} \\ . & . & \mathcal{Q}_{s_i,a_\sigma} & . & . \\ \mathcal{Q}_{s_n,a_1} & . & . & . & \mathcal{Q}_{s_n,a_m} \end{pmatrix}.$$

   Let's assume that the action a_σ is associated with the state s_i and that this action is conflicting. In this case, the expert determines and selects the action a_ϕ (among the possible actions from the associated state) that can be executed instead of the action a_σ. Such a principle does not violate the system's constraints or lead to another conflict.

   We have:
   $V^*_{s_i} = \underset{a}{max}\ (\mathcal{Q}^*_{s_i,a_1}, .., \mathcal{Q}^*_{s_i,a_\phi}, .., \mathcal{Q}^*_{s_i,a_\sigma}, .., \mathcal{Q}^*_{s_i,a_m})$
   If $\mathcal{V}^*_{s_i} = \mathcal{Q}^*_{s_i,a_\sigma}$ in case of a conflict, this one can be resolved if $\mathcal{V}^*_{s_i} = \mathcal{Q}^*_{s_i,a_\phi}$

$$\text{i.e. } \mathcal{Q}^*_{s_i,a_\phi} > \mathcal{Q}^*_{s_i,a_\sigma}.$$

   Using Equation (3.3):

$$\mathcal{R}(s_i,a_\phi) + \gamma \textstyle\sum_{s'\in\mathcal{S}} \mathcal{T}(s_i,a_\phi,s')V^*(s') > \mathcal{Q}^*_{s_i,a_\sigma}\ ,$$

   The equation that allows the calculation of the new reward of the **Algorithm** 5 is:

$$\begin{aligned} \mathcal{R}(s_i,a_\phi) &> \mathcal{Q}^*_{s_i,a_\sigma} - \gamma \sum_{s'\in\mathcal{S}} \mathcal{T}(s_i,a_\phi,s')V^*(s'), \\ \mathcal{R}(s_i,a_\phi) &= \lceil \mathcal{Q}^*_{s_i,a_\sigma} - \gamma \sum_{s'\in\mathcal{S}} \mathcal{T}(s_i,a_\phi,s')V^*(s') \rceil. \end{aligned} \tag{4.1}$$

   If $[\mathcal{Q}^*_{s_i,a_\sigma} - \gamma \sum_{s'\in\mathcal{S}} \mathcal{T}(s_i,a_\phi,s')V^*(s')]$ is an integer

$$\mathcal{R}(s_i,a_\phi) = \lceil \mathcal{Q}^*_{s_i,a_\sigma} - \gamma \sum_{s'\in\mathcal{S}} \mathcal{T}(s_i,a_\phi,s')V^*(s') \rceil + 1. \tag{4.2}$$

Algorithm 5: Resolve_Conflicts

Input: π, Q, $\mathcal{R}$, $\mathcal{T}$, ST_{Conf}, a_ϕ
Output: π^*, Q^*, $\mathcal{R}'$

1 $V \leftarrow \underset{a}{max}(Q)$;
2 $\mathcal{R}' \leftarrow \mathcal{R}$;
3 **foreach** $s_i \in \mathcal{S}$ **do**
4 **if** $ST_{Conf}(s_i) = True$ **then**
5 $a_{exec} \leftarrow \pi(s_i)$;
6 $\mathcal{R}'(s_i, a_\phi) = \lceil \mathcal{Q}^*_{s_i, a_{exec}} - \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s_i, a_\phi, s') V^*(s') \rceil$; `// Equation 4.1`
7 π^*, Q^* $\leftarrow$ Compute the optimal Policy using the reward matrix $\mathcal{R}'$;

The systematic rewards tuning algorithm is given in the **Algorithm** 5. a_{exec} is the executed action and a_ϕ is the action expected to solve the conflicts.

The ***Resolve_Conflicts*** method is used to check all states of a policy. If a state is found to be a conflicting state ($ST_{Conf}(s_i) = True$), a new reward is calculated (line 6). Finally, the new policy is recalculated using the new reward (line 7).

In [Ye10], the authors show that the simple Policy Iteration method is indeed a strongly polynomial algorithm for the discounted MDP with a fixed discount rate of $0 \leq \gamma < 1$. The number of its iterations is bounded by $\mathcal{O}(\frac{n^2(m-1)}{1-\gamma}.log(\frac{n^2}{1-\gamma}))$ and each iteration requires at most $\mathcal{O}(n^2m)$ arithmetic operations, where n and m are the number of states and the maximum number of actions per state, respectively. Thus, the complexity of the method shown in **Algorithm** 5 is $\mathcal{O}(n)+\mathcal{O}(\frac{n^2(m-1)}{1-\gamma}.log(\frac{n^2}{1-\gamma}))$.

2. **Self-adaption**

The ***Self_Adapt*** method presented in the **Algorithm** 6 is used to adapt the mission or the task of the UAV to remove the conflict between this UAV and the other member of the swarm. There are several scenarios to resolve this conflict.

The conflicting states are marked in each scenario according to the alternative action a_ϕ and based on the scenario's parameters. Then, the alternative action a_ϕ, the MDP and its conflicting states are passed as parameters to the ***Resolve_Conflicts*** method (see **Algorithm** 5).

If there is conflict, the method will use the parameter *priority* (if received) and the provided alternative task Alt_{task} received to auto-adapt the mission of the UAV to the swarm mission. In the case of not reception of the priority, the method will use the local alternative task $Local_{task}$ to auto-adapt the mission of the UAV to the swarm mission.

The complexity of this method is $\mathcal{O}(n) + \mathcal{O}(\frac{n^2(m-1)}{1-\gamma}.log(\frac{n^2}{1-\gamma}))$.

Algorithm 6: Self_Adapt

Input: Internal: π: UAV policy, Q Q-function table, $\mathcal{R}$ matrix, $\mathcal{T}$ matrix of the MDP,
From the swarm: priority, Alt_{task}
Result: $\pi*$ policy without conflicts

```
(Bool_Conf, ST_Conf) ← Check_Conflicts (π, conflicts); // see Algorithm 4
π* ← π;
if Bool_Conf = True then
    if priority = True then
        π* ← Resolve_Conflicts (π, Q, R, T, ST_Conf, Alt_task); // see
             Algorithm 5
    else
        π* ← Resolve_Conflicts (π, Q, R, T, ST_Conf, Local_task); // see
             Algorithm 5
```

4.1.3 Case study with a collaborative mission planning

In the case study (Figure 4.2), a swarm of UAVs follows a predefined trajectory that searches for targets (with a visual recognition system) upon arrival at the search area. It tracks the potential target(s) when it(they) is(are) detected.

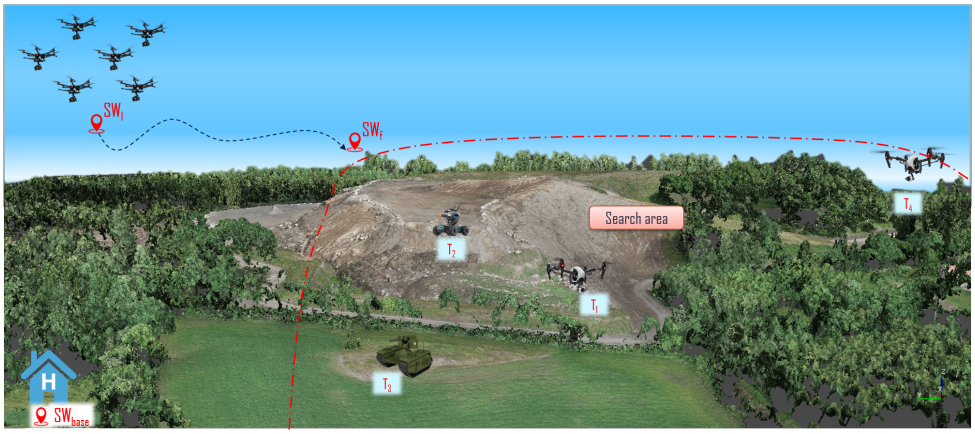


Figure 4.2 – The Scenario of a swarm of UAVs searching and tracking targets [Ham+23].

1. **Case study description**

   The tracking mission is modeled with an MDP. This mission has 22 states and 16 actions, including 5 for safe navigation, 6 for activating different algorithmic versions of the tracking application, and 4 for ensuring a successful search of a landing area. Each UAV in the swarm contains the MDP illustrated in Figure 4.3).

   The parameters of the MDP, in particular, the probabilities of transitions, are produced and updated online by Health Management (HM) blocks [Hir+18]. These

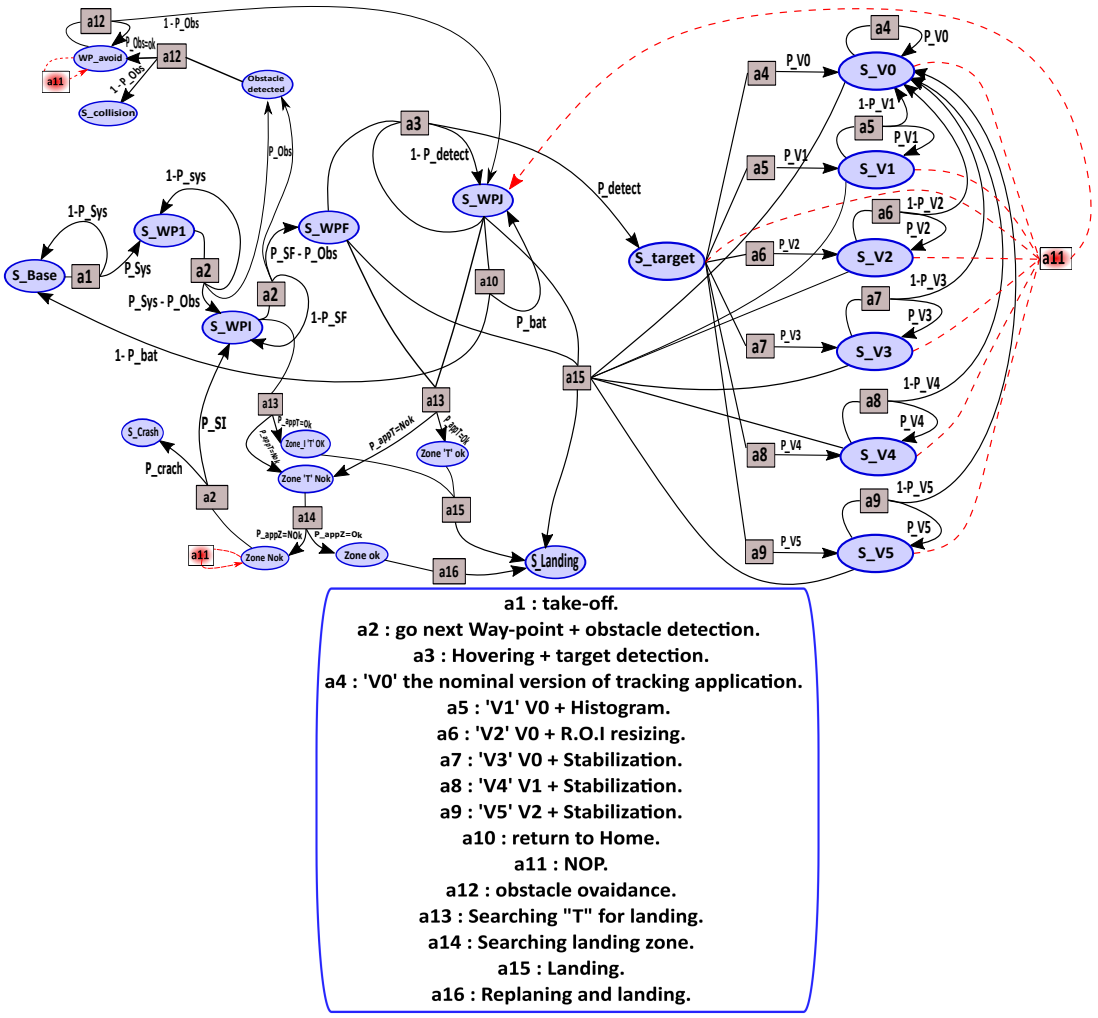


Figure 4.3 – The Scenario of a swarm of UAVs searching and tracking targets [Ham+23].

HM modules are diagnostic blocks that use Bayesian Networks for the real-time evaluation of health states of the on-board system, on-board applications, and communication network, taking into account the context of the mission.

2. **Conflicts description and management**
   To manage the conflicts/constraints between the UAVs of the swarm, we proceed as follows:

   ➥ Enumerate the conflicts/constraints at the swarm level that can occur between these UAVs, such as:
   — C_1**:** The UAV_i follows $Target_t$, and one of its sensors fails, e.g. Camera or GPS.
   — C_2**:** the UAV with the smallest distance to target $Target_t$ is the only one to track it.
   — C_3**:** The UAV_i follows $Target_t$ and its speed is lower than of the $Target_t$.
   — C_4**:** The UAV_i follows $Target_t$ which enters an unauthorized area for the follower.
   — C_5**:** The UAV_i follows $Target_t$ and does not have enough energy to continue the mission and has to make an RTH (return to home/base).

   ➥ By taking the constraints/conflicts mentioned above, we can identify the conflicting state in the MDP where conflicts occur.
   For example, let's assume that UAV_i, which tracks $Target_t$, receives the signal to stop the tracking and leaves the task to another UAV. It has to modify its mission plan and execute an alternative action.

   ➥ Finally, alternative actions to resolve constraints/conflicts are identified.

   In this study, constraint/conflict management consists of the self-adaptation of each UAV in the swarm to avoid constraint violation. The priorities of the conflicts are evaluated offline depending on the context of the mission and the mission's strategy, 'safety' or 'mission first'.
   Priorities between UAVs are evaluated in the cloud according to the mission context and the system constraints of each UAV.

3. **Validation on scenario**
   To validate the method with the tracking mission, we elaborate on the following scenario:

   ➥ **Scenario 1**
   UAV_i follows $Target_t$, and one of its sensors fails, e.g., Camera or GPS. A conflict arises, i.e., either continuing the mission or returning home. For mission

failure avoidance and as the priority to continue the mission is the highest, the cloud sends an alternative action to UAV_i that will increase the priority to the return to home action. So, UAV_i will adapt its mission plan and return to its home station (RTH). The rest of the swarm will autonomously adapt its mission plan and not require that the cloud intervene. The closest UAV to the target will take action and follow $Target_t$.

4.1.4 Results

The results of our experiments for the scenario are presented here. The initial policy of the UAV without conflict (i.e., the system functions properly) is shown in **Figure 4.4**.

S1 base	a1
S2 wp1	a2
S3 wpI	a2
S4 wpF	a3
S5 S_WPj	a3
S6 target	a4
S7 S5V0	a4
S8 S5V1	a7
S9 S5V2	a7
S10 S5V3	a7
S11 S5V4	a7
S12 S5V5	a7

S13 OBSTACLE_det	a12
S14 WP_avoidance	a12
S15 S_Collision	a11
S16 zone_I OK	a15
S17 zone T NOK	a14
S18 zone T OK	a15
S19 zone OK	a16
S20 zone NOK	a2
S21 S_landing	a11
S22 S_crash	a11

Figure 4.4 – Initial policy of the UAV without conflict (computed with Policy Iteration).

The update of the GPS health probability will affect the health probability of the system and the different applications that use it. So, when the GPS fails, the resolution of the MDP for the tracking mission gives us the policy as shown in Figure 4.5.

S1 base	a1
S2 wp1	a2
S3 wpI	a2
S4 wpF	a3
S5 S_WPj	a10
S6 target	a10
S7 S5V0	a5
S8 S5V1	a5
S9 S5V2	a5
S10 S5V3	a5
S11 S5V4	a5
S12 S5V5	a5

S13 OBSTACLE_det	a12
S14 WP_avoidance	a12
S15 S_Collision	a11
S16 zone_I OK	a15
S17 zone T NOK	a14
S18 zone T OK	a15
S19 zone OK	a16
S20 zone NOK	a2
S21 S_landing	a11
S22 S_crash	a11

Figure 4.5 – UAV Policy in the case of GPS failure [Ham+23].

We can notice that some actions are not the same as in the case of a good system functioning. However, we must consider that the UAV is tracking a target in this scenario.

The UAV is in one of the states related to the tracking (from S7 to S12, which are the different algorithmic versions of the tracking application). When the GPS fails, the system tries to execute another version of the tracking application (a5: Version 5), as shown in Figure 4.5.

The *check_conflict* method was able to detect this conflict. The *self-adapt* method then recalculates the rewards of the actions related to the conflicting states. The calculation of this new reward favors the execution of the alternative action (i.e., action a5) that will allow the UAV to adapt itself to the swarm and avoid conflicts. Figure 4.6 shows the resulting policy after solving the conflicts.

S1 base	a1		S13 OBSTACLE_det	a12
S2 wp1	a2		S14 WP_avoidance	a12
S3 wpI	a2		S15 S_Collision	a11
S4 wpF	a3		S16 zone_I OK	a15
S5 S_WPj	a10		S17 zone T NOK	a14
S6 target	a10		S18 zone T OK	a15
S7 S5V0	a15		S19 zone OK	a16
S8 S5V1	a15		S20 zone NOK	a2
S9 S5V2	a15		S21 S_landing	a11
S10 S5V3	a15		S22 S_crash	a11
S11 S5V4	a15			
S12 S5V5	a15			

Figure 4.6 – UAV Policy in the case of GPS failure after resolving the conflicts with the auto-adapt method [Ham+23].

4.2 Rewards adaptation at UAV level

This Section illustrates UAV-level reward adaptation, tackling conflict detection dynamics in a singular UAV mission managed by multiple parallel MDPs. We then delve into conflict resolution within distributed MDPs, emphasizing a specific case study with a single UAV's mission managed through multiple parallel MDPs. Finally, we conclude by analyzing the results obtained.

4.2.1 Detection of conflicts within a mission of a single UAV managed through multiple parallel MDPs

Decomposition methods are the most common for reducing the complexity of MDP models. They split the problem into several smaller and simpler sub-problems. Each sub-problem is represented, treated as a distinct sub-MDP, and solved independently. Conflicts arise when decision-making is distributed between agents due to inconsistent actions or events that violate constraints.

1. **Conflicts and constraints**

   We can distinguish two types of constraints:

   - ➥ Constraint associated with the behavior conflict: the existence of a conflict during the parallel execution of some antagonistic actions. The constraint expresses that the antagonist cannot occur simultaneously; for example, the *Return To Home* and the *Tracking* actions are antagonist actions. They cannot be executed on a UAV simultaneously.
   - ➥ Constraint on system or sensor health: The existence of a critical probability of the transition function can lead to a failure if it goes below a threshold; for instance, suppose the probability of detecting an obstacle is higher than a threshold fixed by an expert according to the speed of the UAV. In that case, we prefer the *Avoidance/Obstacles* action over other actions.

   Then, we introduce the policy priority between MDPs.

2. **Policy priority**

   In addition to the behavior conflicts, we need to know which agent can keep their policy and which must change it. For that, we define a priority between the policies of the MDPs. The MDP with the highest priority keeps its policy, unlike the other MDPs. To modify the policies, we adapt the rewards. We can distinguish two kinds of priority:

   - ➥ Static policy priority:
     In this case, the expert defines a policy priority between the MDPs; this priority is determined offline and remains stable throughout the decision process.
   - ➥ Dynamic policy priority (e.g., mission planning of a UAV in Section 4.1.3):
     In this case, the expert defines a threshold on some probability of the transition function. This will make it possible to adapt the policy priority of the MDPs according to this threshold.

   The dynamic policy priority is helpful for an online adoption of the local policies.

4.2.2 Solving conflicts in a single UAV managed through multiple parallel MDPs

In the case of multiple MDPs, we propose the following steps:

- ☞ **Step 1:** we compute the policy for each MDP (e.g., using Policy Iteration).
- ☞ **Step 2:** we check if the constraint is verified or if there is a conflict between policies (see Algorithm 8). This Algorithm returns the Boolean variables that point out the MDP policies containing any actions in conflict and the states involved.

☞ **Step 3:** if the constraint is verified or conflicts are detected in **Step 2**, we solve the conflicts by modifying one or more policies according to their priority (see Algorithm 7). This Algorithm uses Algorithm 5 to increase the reward of the action a_ϕ in the policy having the lower priority.

Algorithm 7: Solving_Policies_Conflicts ( )

Input: Internal: π_α, π_β, $\alpha_{inconflicts}$, $\beta_{inconflicts}$, $states_{inconflicts_\alpha}$, $states_{inconflicts_\beta}$
Result: $\pi_{\alpha withoutConflict}$, $\pi_{\beta withoutConflict}$

```
if α_inconflicts = True then
    π_αwithoutC = Resolve_Conflicts(π_α, Q_α, R_α, T_α, states_inconflicts_α)
else
    π_αwithoutConflict = π_α
if β_inconflicts = True then
    π_βwithoutC = Resolve_Conflicts(π_β, Q_β, R_β, T_β, states_inconflicts_β)
else
    π_βwithoutConflict = π_β
```

Algorithm 8: Check conflicts(π_α, π_β, order, constraints)

Input: Internal: π_α, π_β, order, constraints
Result: $\alpha_{inconflicts}$, $\beta_{inconflicts}$, $states_{inconflicts_\alpha}$, $states_{inconflicts_\beta}$

```
states_inconflicts_α = NULL;
states_inconflicts_β = NULL;
foreach (s_i, s_j) ∈ S_A × S_β do
    if π_α(s_i) = a_σ and π_β(s_j) = a_σ then
        if order(π_α) > order(π_β) then
            if β_inconflicts = False then
                β_inconflicts = True
            states_inconflicts_β(j) = True
        else
            if α_inconflicts = False then
                α_inconflicts = True
            states_inconflicts_α(i) = True
```

4.2.3 Case study of a mission of a single UAV managed through multiple parallel MDPs

1. **Case study description**

   The tracking mission considers a UAV following a predefined trajectory (set of Waypoints). Once the designated search area is reached, the UAV initiates tracking of

a potential target upon detection. This mission is defined using three main MDPs (Navigation, Landing, and Tracking). These distinct MDPs collectively ensure the successful execution of the tracking mission.

❖ **MDP navigation** (Figure 4.7): in this MDP, we find the actions of navigation and safety. In addition to the common states which we list below, there are

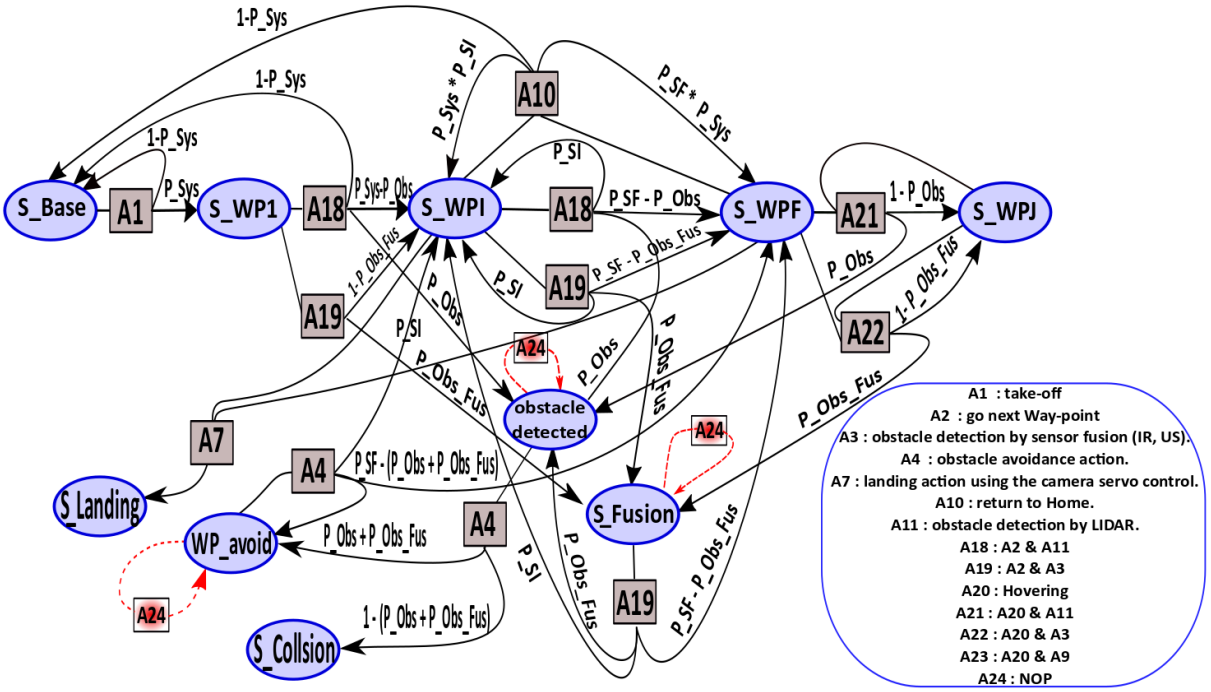


Figure 4.7 – MDP Navigation [Ham+23].

also specific states:

➥ obstacle_detected: an obstacle is detected by the LIDAR or fusion application.

➥ S_fusion: represents the detection obstacle state using a fusion of multiple short/long range infra-red (IR) and ultrasonic (US) sensors.

➥ WP_avoid: way-point resulting from obstacle avoidance.

➥ S_collision: collision state with the obstacle.

❖ **MDP landing** (Figure 4.8): in this MDP, we find the actions required for landing. These actions also correspond to safety actions. In addition to the common states, there are also specific states:

➥ Zone_I‘T’_Ok: a ‘T’ intermediate zone is found.

➥ Zone_‘T’_Ok: a ‘T’ zone is found.

➥ Zone_‘T’_Nok: the ‘T’ zone is not found.

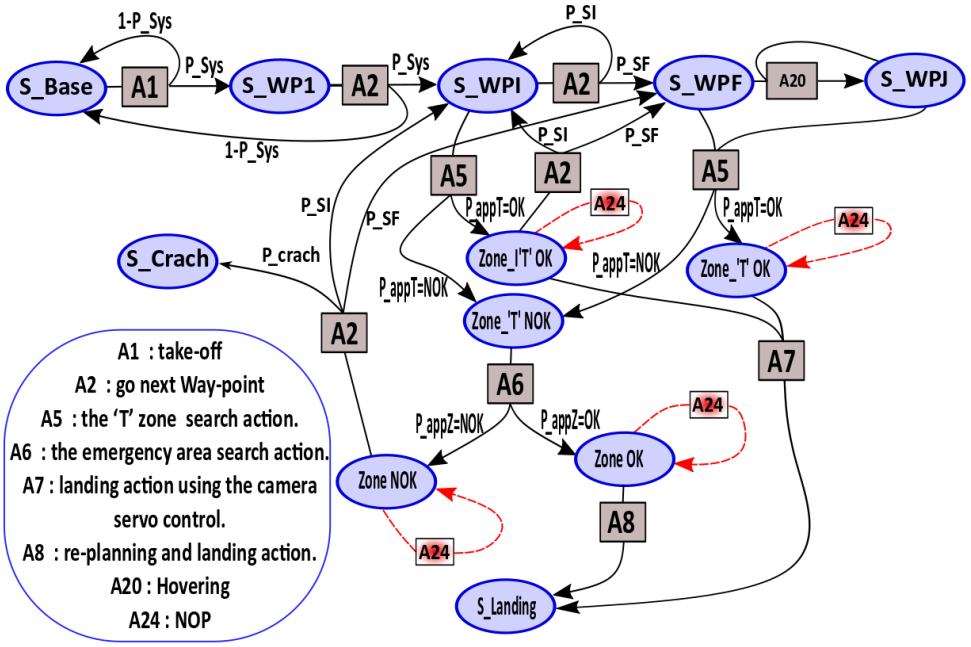


Figure 4.8 – MDP Landing [Ham+23].

- ➥ Zone_Ok: an emergency area is found to land.
- ➥ Zone_Nok: the emergency area is not found.
- ➥ S_crash: crash state.

❖ **MDP Tracking (Figure 4.9):** in this MDP, we find the tracking mission with different algorithmic versions of the tracking application. In addition to

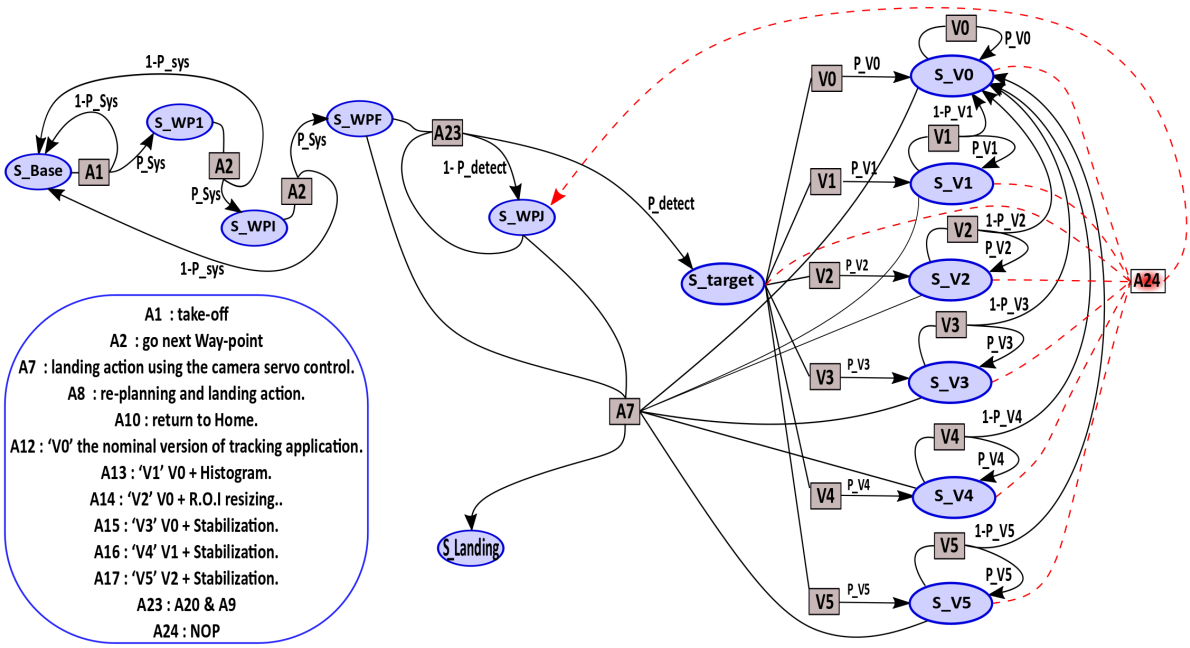


Figure 4.9 – MDP Tracking [Ham+23].

the common states, there are also the following specific states:

- ➥ S_target: represents the target detection state.
- ➥ S_Vi: represents all states corresponding to the different versions of the tracking application.

Some common states are defined for the synchronization of the MDPs; they are below:

- ➥ S_WP1: first way-point of the trajectory (in flight).
- ➥ S_WPI: intermediate way-points of trajectory.
- ➥ S_WPF: final way-point of trajectory corresponding to the tracking area.
- ➥ S_WPJ: way-points of the UAV after the S_WPF.
- ➥ S_landing: landing state.

2. **Constraint description and management**

To manage the conflicts between the three MDPs, we proceed as follows:

- ❖ we enumerate the behavior conflicts that can occur between these MDPs
  - ➥ Obstacle Avoidance & Tracking (V0, V1, ...,V5). [A4 & A12,...,A17]
  - ➥ Obstacle Avoidance & (Re−planning and landing). [A4 & A8]
  - ➥ Obstacle Avoidance & Landing. [A4 & A7]
  - ➥ Obstacle Avoidance & GO_Next_WP. [A4 & A2]
  - ➥ Landing & Tracking (V0, V1, ...,V5). [A7 & A12,...,A17]
  - ➥ RTH (return to home) & tracking (V0, V1, ...,V5). [A10 & A12,...,A17]
  - ➥ GO_Next_WP & Tracking (V0, V1, ...,V5). [A2 & A12,...,A17]
  - ➥ ...etc
- ❖ we identify the conflicting states by taking the abovementioned antagonistic actions.
- ❖ we add the 'NOP' action, allowing us to execute it on one or more of the MDPs containing a previously identified behavioral conflict.

In addition, we consider two constraints for this mission that can lead to its failure. The first one is associated with the probability of system failure. The second one considers the presence of an obstacle on the UAV's trajectory. We propose to define the policy priority according to two parameters:

- ➥ **P_sys**: probability of good health of the system.

Highest priority MDP		*Proba_det_Obst*	
		$< \beta$	$\geq \beta$
P_sys	$< \alpha$	Landing	Landing
	$\geq \alpha$	Tracking	Navigation

Table 4.1 – Highest priority MDP according to *Proba_det_Obst* and P_sys [Ham+23].

- **Proba_detect_obs**: the probability of good obstacle detection, which represents the maximum probability of detection by LIDAR and by fusion (IR, US) $Proba\_det\_Obst = max(P\_Obs\_OK, P\_Obs\_Fus)$

The priority is evaluated dynamically depending on the context of the mission and the system constraints mentioned before.

3. **Validation on scenarios**

   To validate the method with the tracking mission, we elaborate three scenarios as follows:

   - **Scenario 1**: functional system without obstacle detection. [P_sys (=90%) $> \alpha$ and $Proba\_det\_Obst(= 30\%) < \beta$].
   - **Scenario 2**: functional system with obstacle detection. [P_sys (=90%) $> \alpha$ and $Proba\_det\_Obst(= 85\%) > \beta$]
   - **Scenario 3**: system error with a drop in the battery charge. [P_sys (=29%)].

   The two factors α and β can be modifiable by the user according to the criticality of the autonomous system and its speed. For instance, we set α to 30% (if the probability of the system is less than 30%, we prefer the landing over the others) and β to 70% (if the likelihood of obstacle detection is more than 70%, we prefer choosing the action associated with collision avoidance over the others).

   In Figure 4.10, we show the Diagram of the resolution of the MDPs to obtain policies after solving the possible conflicts.

   Our Algorithm periodically scans all three policies and the control variables (P_sys and $Proba\_detect\_obs$) used to assign priority to the MDPs. When one of them is modified (policies) or the threshold values α and/or β of the two variables are reached, the Algorithm recalculates the policies and solves conflicts.

4.2.4 Results

Results of our experiments for the different scenarios validation described in Matlab are presented.

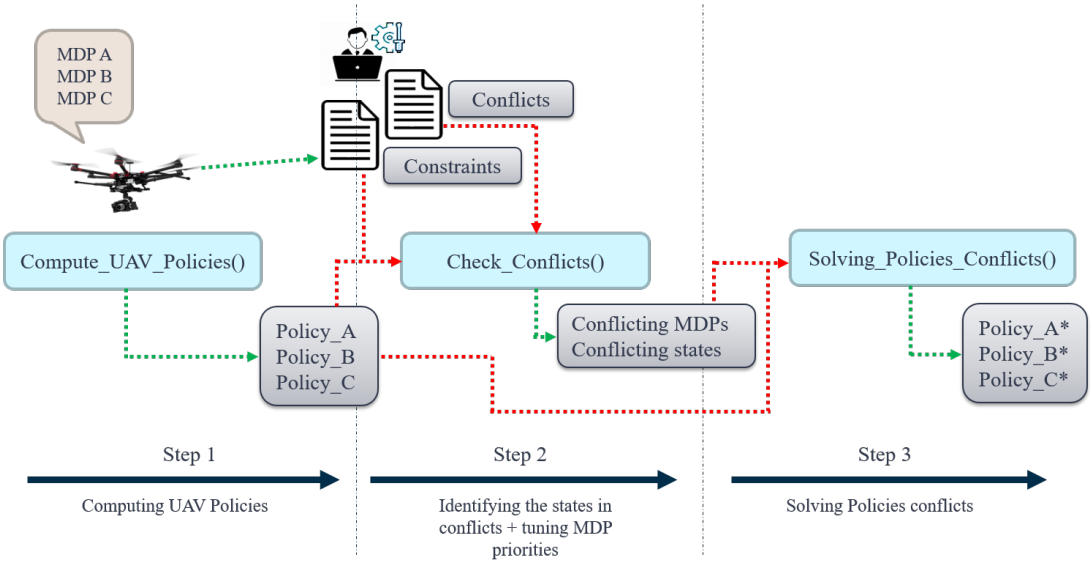


Figure 4.10 – Resolution diagram.

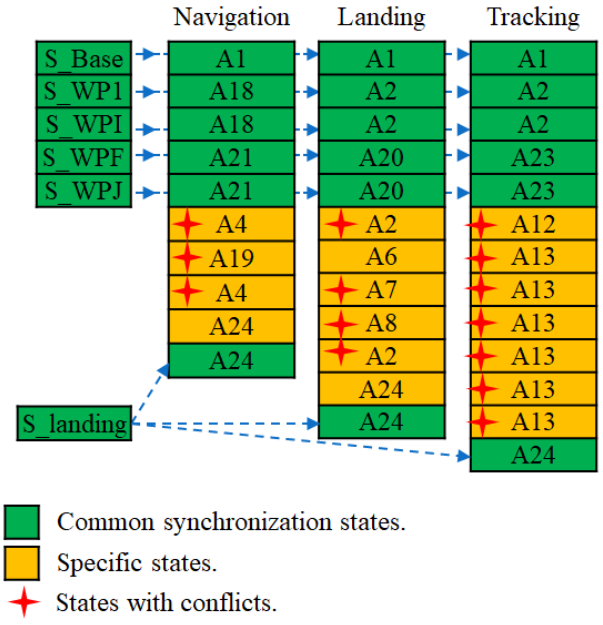


Figure 4.11 – Policies computed with Policy Iteration [Ham+23].

The resolution of the three MDPs of the tracking mission using the Policy Iteration Algorithm gives us these three policies as shown in Figure 4.11.

1. **Scenario 1**

   In the first scenario, we represent the case of the proper functioning of the system without obstacles on the trajectory. With the probabilities given in Table 4.2, we compute the policies of the different MDPs. If we ignore the potential conflicts, we obtain the policies given in Figure 4.11.

Probability	Value	Probability	Value
P_sys	0.9	P_SF	0.6
P_Obs_OK	0.3	P_SI	1 - P_SF
P_Obs_Fus	0.1	P_land	1
P_app_T	0.5	P_app_Z	0.5
P_V0	0.8	P_V1	0.72
P_V2	0.7	P_V3	0.3
P_V4	0.1	P_V5	0.6
P_Det	0.3	P_NOP	0.1

Table 4.2 – Probabilities used in scenario 1 to compute policies and resolve conflicts [Ham+23].

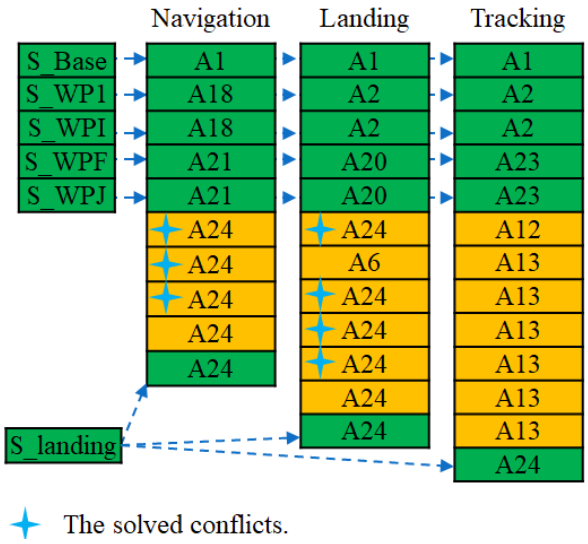


Figure 4.12 – Policies after solving conflicts (scenario 1) [Ham+23].

In Figure 4.12, we give the resulting policies after solving the conflicts. We can notice that the *check_Conflicts* Algorithm can detect conflicts according to *P_sys* and *Proba_detect_obs*. The MDP Navigation receives the highest priority according to the Table 4.1.

2. **Scenario 2**

   In the second scenario, we represent the case of an obstacle appearing on the trajectory. So we change the probability of obstacle detection with Lidar to $P\_Obs\_OK = 85\%$ initially $P\_Obs\_OK = 30\%$. We solve the MDPs, ignoring first the constraints, and we find again the policies depicted via Figure 4.11.

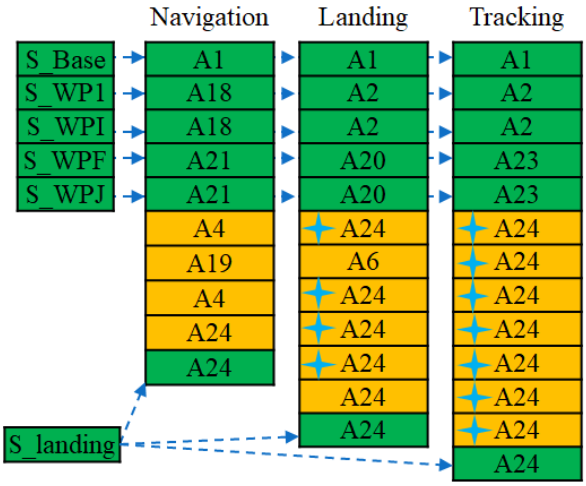


Figure 4.13 – Policies after solving conflicts (scenario 2) [Ham+23].

   Figure 4.13 illustrates the result of solving conflicts using the probabilities of this scenario. We can notice that the *check_Conflicts* Algorithm can detect conflicts. According to the values of *P_sys* and *Proba_detect_obs*, the MDP Tracking receives the highest priority (see Table 4.1).

3. **Scenario 3**

   In the third scenario, we consider the system degrading, e.g., battery malfunction. So we change the probability of good health of the system to $P\_sys = 29\%$ initially $P\_sys = 90\%$ (the other probabilities are the values given in Table. 4.2). We solved the MDPs and determined that the policies correspond to the default policies, as illustrated in Figure 4.11.

   Figure 4.14 illustrates the result after solving the conflicts. We can notice that the *check_Conflicts* Algorithm can detect conflicts. According to the values of *P_sys* and *Proba_detect_obs*, the MDP Landing receives the highest priority (see Table 4.1).

This Section exhibits three potential conflicts between the three concurrent MDPs of the UAV mission. Our method allows us to avoid conflict while executing several MDPs and adapting to different context changes.

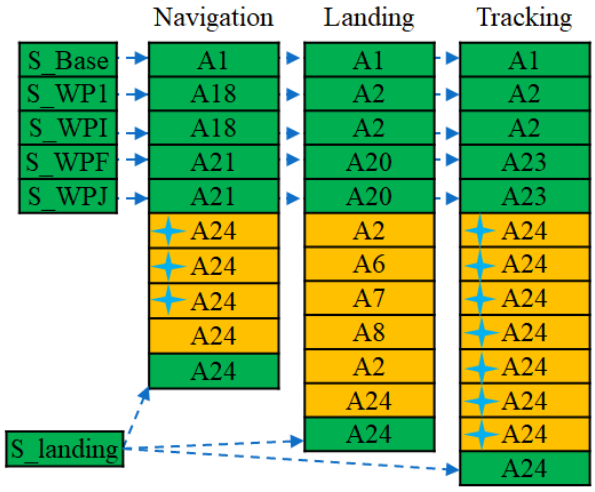


Figure 4.14 – Policies after solving conflicts (scenario 3) [Ham+23].

4.3 Simulation on Coppeliasim robot simulation platform and MATLAB

To demonstrate the effectiveness of the rewards adaptation methodology for constraints management, we conduct simulations on the CoppeliaSim robot simulation platform [RSF13] and MATLAB.

CoppeliaSim, formerly known as Virtual Robot Experimentation Platform (V-REP), provides a comprehensive environment for designing, simulating, and testing robotic systems. The CoppeliaSim platform offers distinctive features, making it a versatile tool for various applications. It employs a distributed control architecture, allowing individual control of each object or model. Users can implement control through embedded scripts, plugins, Robot Operating System (ROS) nodes, or remote Application Programming Interface (API) clients, providing flexibility in the development process. This platform is useful for rapid algorithm development, simulations in factory automation, and quick prototyping and verification. Its functionality and API can be readily extended through plugins and modules, offering a customizable and adaptable environment. Noteworthy existing modules include support for physics, kinematics, path planning, custom user interfaces, openCV integration, and more, making it versatile and ideal for multi-robot applications. Controllers can be written in C/C++, Python, Java, Lua, Matlab, or Octave.

We decided to model the mission considered in this part using Matlab since all our algorithms and evaluations are written in Matlab. The Matlab simulation receives data from CoppeliaSim through remote API functions.

1. **GUI Matlab interface**:

   We have engineered an interface to facilitate the administration of building in-

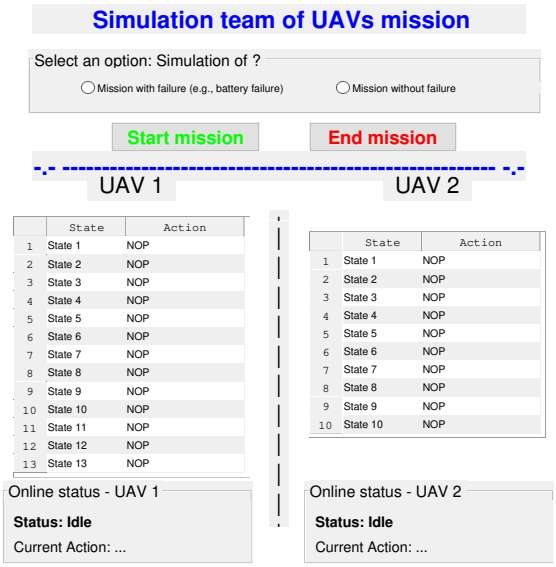


Figure 4.15 – Initial interface of the Matlab GUI used to control the simulation.

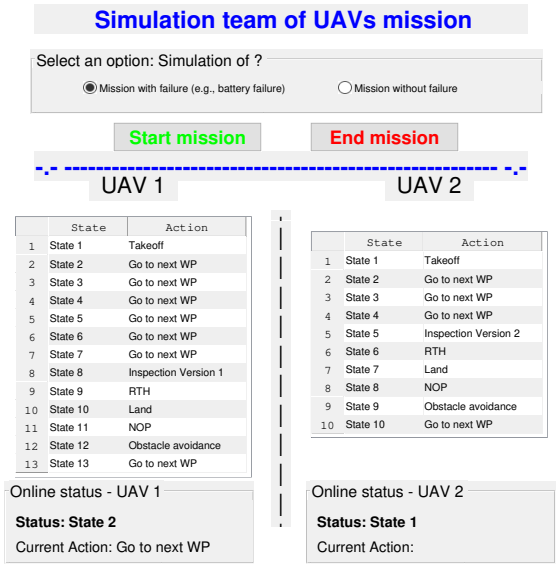


Figure 4.16 – Matlab GUI during the simulation.

spection missions. Illustrated in Figure 4.15, the interface incorporates two distinct buttons enabling the selection of simulation types—specifically, simulations involving battery failure and those without failure. Additionally, it features initiation and termination buttons for launching and concluding missions, respectively.

The foundational segment of the interface comprises two tables, each corresponding to the policies of individual UAVs. The tables dynamically update in a manner exemplified in Figure 4.16. Upon launching the mission, real-time modifications to the UAV policies are reflected in the displayed tables, providing a live representation of the evolving mission status and pertinent events.

2. **CoppeliaSim scene**:

   We have engineered a scene in CoppeliaSim as shown in Figure 4.17. The illustrated scenario evolves within the context of a building inspection, featuring two UAVs positioned initially at the top of the building. Each UAV is equipped with a distinct application, influencing the methodology employed during the inspection process. For the sake of simplicity, these applications are differentiated based on their movement patterns— a zigzag trajectory characterizes one, while the other enables vertical movement only.

   Significantly, the buildings under scrutiny differ in dimensions, height, and width. Notably, priority is given to the two-story building over the three-story one.

   The complexity of the inspection task is further compounded by strategically placed obstacles, namely trees and poles, intentionally positioned along the predetermined flight path of the UAVs.

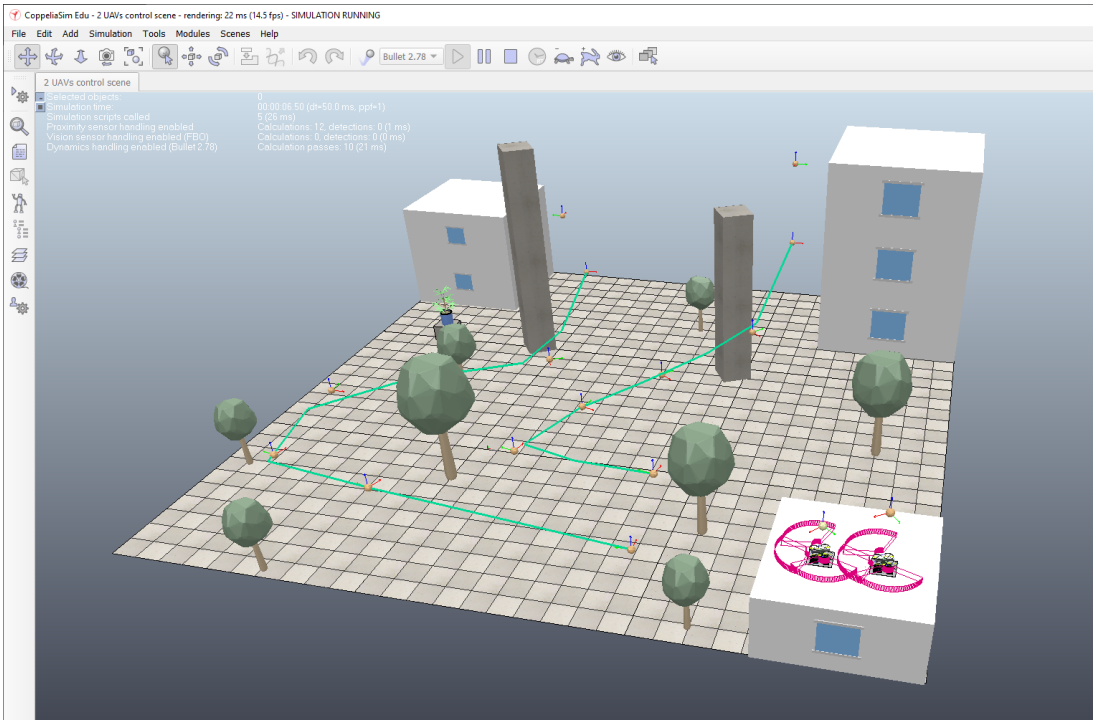


Figure 4.17 – Scene view of the considered buildings inspection mission in CoppeliaSim.

The envisioned scenarios unfold in two distinct cases:

(a) **UAVs without failure**:

In this scenario, the UAVs navigate through their assigned missions, following a predefined path represented by a sequence of GPS points. They encounter and successfully overcome obstacles such as trees and poles along their designated paths. Each UAV executes its mission, demonstrating the efficacy of the MDP-based mission planning. This dynamic evaluation considers not only the fulfillment of individual tasks but also the ability of the UAVs to autonomously navigate complex environments and return to the base, serving as a foundational benchmark for assessing the efficiency of the MDP-based mission planning for UAVs.

(b) **UAV with failure (e.g., battery exhausted in this case)**:

This scenario explores the implications of a failure event illustrated by an exhausted battery. The analysis investigates how the UAVs adapt or respond to unexpected hazards, particularly those affected by the failure. The investigation contains considerations such as re-routing strategies, emergency procedures, and the overall impact on the inspection process. The mission develops similarly to the first scenario (without failure). While UAV_A UAV inspects the two-story priority building, at a specific time, its battery unexpectedly exhausts. The battery monitoring system promptly detects this failure, imme-

diately updating probabilities within MDP. The affected UAV_A quickly adjusts its mission, transmitting a signal to its neighboring UAV_B before beginning a landing.

Upon receiving the signal, the neighboring UAV_B adapts to the situation, incorporating the transmitted coordinates and specific inspection functions for the priority building. The adapted UAV_B inspects the two-story priority building and returns to its base. The inspection of the initially planned three-story building may be accommodated depending on the remaining energy resources.

4.4 Final consideration

This chapter introduces a sophisticated conflict resolution method for mitigating conflicts arising from the parallel execution of multiple MDPs within a multi-UAV system. The embedded, low-overhead self-adaptation method dynamically adjusts reward values, considering internal and collaborative constraints, thus averting potential conflicts at the multi-UAV system. By leveraging cloud capabilities, UAVs can favor alternative actions to resolve conflicts stemming from contradictory mission objectives and states.

The proposed method is distinguished by its efficiency, exhibiting minimal latency and energy consumption. The case study demonstrates its ability to identify and resolve behavioral conflicts during the parallel execution of MDPs, enhancing mission planning robustness. Notably, the method can be systematically employed for dynamic updates to adhere to safety rules, providing adaptability in response to evolving scenarios, including potential cyber-attacks where new security rules may be introduced.

This approach is applied to a particular case where a single UAV is managed by several MDPs in a complex mission. For example, one MDP may suggest increasing altitude to avoid an obstacle, while another may recommend landing. Our conflict resolution method enables the UAV to make a balanced decision by dynamically adjusting reward values according to internal mission constraints to establish safe planning.

As presented in this chapter, the comprehensive evaluation of MDP-based mission planning methods for UAVs underscores the robustness and resilience of the proposed approach under diverse conditions. The systematic conflict resolution and dynamic adaptation contribute to the overall effectiveness of UAV mission planning, showcasing the potential for broader applications and advancements in autonomous systems.

CHAPTER 5

BN-based operating mode adaptation for policy self-adaptation in collaborative multi-UAV system

Contents

Introduction

This chapter introduces a novel approach to enhance the adaptive capabilities of collaborative multi-UAV systems through Bayesian Network (BN)-based operating mode adap-

tation. The focus is on local and cloud-based policy self-adaptation, combined with examination into drone communication networks, to optimize mission planning for UAVs. Section 5.1 begins by presenting the differences in applying the self-adapting policy method in both local and Cloud modes.

Section 5.2 starts by presenting the existing communications networks used in a multi-UAV system, addressing associated issues. These networks primarily involve local (UAV-UAV) and cloud-based communications. In Section 5.3 extensively explores the local/hybrid mode-switching mechanism in detail, including the consideration of data used to estimate the performance of the communications. Section 5.4 outlines different scenarios, detailing the selection criteria for both local and hybrid modes. In Section 5.5, a case study involving collaborative mission planning is presented, illustrating the application of the self-adapting policy method associated with the local and hybrid modes switching in a multi-UAV system searching for targets. The analysis includes scenarios where local communications are secure and effective and situations where potential risks or vulnerabilities in local communications are identified. Section 5.6 summarizes the effectiveness of the self-adaptation policy method and its impact on mission planning.

5.1 Self-adapting policy method in local and hybrid mode

The self-adaptation process for policies within the collaborative multi-UAV system offers flexibility through two distinct modes: local adaptation utilizing the UAV's internal data and a hybrid approach exploiting cloud data. The subtle distinctions between these modes are thoughtfully outlined in the Table 5.1, providing a comprehensive comparison between the local (UAV + neighbors) and hybrid (cloud-assisted) modes when employing the self-adaptation policy method.

We notice that choosing between these modes entails trade-offs in control, data utilization, alternative action determination, and UAV priority.

The local mode relies on decentralized control, using internal context from the UAV and its neighbors for decision-making. In this mode, alternative actions are determined based on local information, and UAV priorities are defined offline, with online adaptability. In contrast, the hybrid mode combines decentralized and centralized control, exploiting both local and cloud-based data sources. This allows for a more comprehensive understanding of the multi-UAV system context, enabling alternative actions to be determined locally and received from the cloud through aggregating local and central data. The UAV priorities in hybrid mode are dynamic, being updated with cloud-provided information.

The mode choice depends on specific application requirements, objectives, and the

Mode	Local (UAV + neighbors)	Hybrid (use the cloud)
Type of control	Use decentralized control	Use the both decentralized and centralized control if needed
Context (data) used	Use the internal context of the UAV (and the internal context of its neighbors if available)	Use the internal context of the UAV (and the internal context of its neighbors if available) and the context of the multi-UAV system
Alternative action	Alternative action determined with the local information	Alternative action can be determined locally or received by the cloud or can be the aggregation of local and central (depending on the objectives/risks)
Priority of the UAVs	Priority of the UAV is defined locally offline (depending on the characteristic of the UAV) and can be adapted online	Priority of the UAV can be defined locally and updated with the priority provided by the cloud depending on the new information of the cloud

Table 5.1 – Differences between using the self-adapting policy method in local and cloud mode

trade-off between local autonomy and cloud assistance. The local mode offers simplicity and autonomy, while the hybrid mode provides a more adaptable approach with a wide volume of data. The decision should align with the system's goals, emphasizing the need for decentralized control and local data in the local mode or the advantages of enhanced context awareness and dynamic adaptation with cloud assistance in the hybrid mode.

5.2 UAV Communication networks

A single UAV concept was long considered the default choice for mission performance. But nowadays, using a multi-UAV system offers many advantages. However, there are several ways for communications between UAVs in multi-UAV system, including Infrastructure-based multi-UAV system architecture [NAL21]; Flying Ad Hoc Network (FANET) architecture [Jos+22]; Wireless Mesh Network [Cui+17]; cellular networks [LTW+22]; satellite communications [Lee+22]; cloud communications [Jun+21].

5.2.1 Local (UAV-UAV) communications and its issues

In many scenarios, the UAV communications network operates by forming a multi-UAV system to accomplish a specific mission in a large geographic area. However, unlike many other wireless networks (e.g., MANET (Mobile Ad hoc Network) and VANET (Vehicular Ad-hoc Network)), the topology of UAV networks remains highly dynamic, with

the number of nodes and links changing, as like the relative positions of the nodes. This would cause an increase in the risk of link failures affecting communication.

- **Issues:**
  However, many issues affect the performance of the communications between UAVs of the multi-UAV system and can negatively impact the performance of the network, including:
  - Routing issues: finding the most efficient route that allows the network scaling and reducing the End-to-End latency,
  - Mobility issues: the connection of the UAV in a multi-UAV system with the other members should remain stable even if the UAV moves.
  - Scalability issues: the increase in the number of drones in the network leads to an increase in latency and a reduction in network throughput, which leads to network congestion,
  - Reliability issues: due to frequent link failure;
  - Power consumption issues: when the number of UAVs in a network increases, the network's energy consumption also increases.
  - Limited onboard energy of UAVs, etc.

5.2.2 Cloud communications and its issues

Cloud communications involve using cloud-based services and platforms to communicate between devices, applications, and users. Cloud communications enable communications between multiple UAVs in a network. Cloud communications provide a wide range of services, such as data sharing, collaboration tools, and a centralized control system for a multi-UAV system that maintains UAVs' coordination and mission objectives.

- **Issues:**

However, many issues affect the performance of the communications between members of the multi-UAV system and can negatively impact the performance of the network, including:

- Latency issues: delays in cloud communication can impact real-time decision-making, mission execution, and coordination among UAVs.
- Bandwidth limitations: limited data transfer capacity can lead to congestion, slowing communication, and impacting the efficiency of information exchange.
- Reliability issues: dependence on cloud services introduces vulnerability to outages or service disruptions, which in turn disrupt the coordination and control of the multi-UAV system.

- ➙ Security and privacy issues: transmitting sensitive data increases concerns about unauthorized access and breaches. This may compromise mission-critical information and network integrity.
- ➙ Dependency on internet connectivity: dependence on the cloud in areas with unreliable internet connectivity creates difficulties in communicating with UAVs.
- ➙ Scalability challenges: the limited ability of the cloud to scale efficiently affects coordination as UAV numbers grow, causing resource constraints.
- ➙ Data synchronization issues: maintaining consistent information among UAVs presents essential synchronization challenges for the efficient operation of the multi-UAV system.

5.3 Local/hybrid mode switching

A local/hybrid mode switching module will be integrated into each UAV, and it will be used to estimate the degree of autonomy of each one.

This module analyzes the behavior of the UAV system under uncertainty by considering sensor and network data from local neighbors. Using this information, the module can estimate the best choice of operating mode, as illustrated in Figure 5.1.

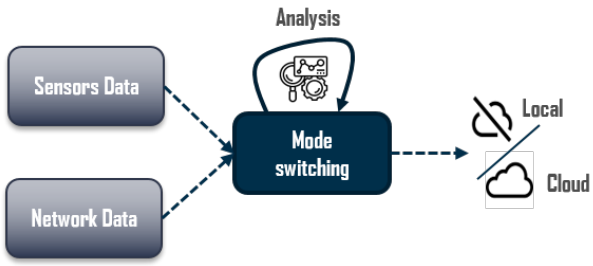


Figure 5.1 – Mode switching diagram.

5.3.1 Input data and performance metrics

The input data of the switching mode module that are used are grouped into two categories:

- ❖ **UAV data**:

  The UAV data are the health data of the onboard system, which come from the autopilot, sensors, battery, etc. For simplicity reasons, in this work, all these data are collected and processed by software monitoring systems integrated into the UAV. These data are then grouped into a new variable that describes the percentage of the health status of the UAV to make decisions.

❖ **Network data**:
To measure the overall performance of the communications Quality of Service (QoS), two types of communications have been distinguished: local communications and cloud communications:

➥ **Local communications:**
Local or inter-UAV communications designate exchanging data and information from a UAV and neighboring ones.

➥ **Cloud communications:**
Cloud communications use cloud-based services and platforms to communicate between devices, applications, and users. Cloud communications can enable communications between multiple UAVs in a network. Cloud communications can offer a wide range of services, such as data sharing and collaboration tools and a centralized control system for a multi-UAV system that maintains UAVs' coordination and mission objectives.

To determine the performance of both types of communication, we mainly analyze and compare the most commonly used metrics in UAV communications [PPB19; Sha+19; Sin+19; YL19; Yog21].

The metrics used are :

☞ ***Packet Delivery Ratio (PDR)***: defined as the number of packets accepted by the destination node and the number of packets transmitted by the source. Its calculation formula is

$$Packet\ Delivery\ Ratio\ (PDR) = \frac{\sum Delivery_{Data}}{\sum Transmit_{Data}} \tag{5.1}$$

☞ ***Packet delivery rate***: shows how successfully a protocol delivers packets from source to destination. It can be characterized as

$$Packet\ Delivery\ Rate = \frac{\sum Delivery_{Data}}{\sum Transmit_{Data}} * 100 \tag{5.2}$$

where $\sum Delivery_{Data}$ is the total packets successfully received, and $\sum Transmit_{Data}$ is the total packets sent.

☞ ***End-to-End (E2E) delay***: is calculated as the time elapsed between its transmission and its reception. It also carries time elapsed in buffering and processing for the data packet transmission. The End to end delay is calculated by the formula:

$$E2E\ Delay = \sum (Time_{end\_packet} - Time_{start\_packet}) \tag{5.3}$$

where: $Time_{end\_packet}$ is the time when the last packet is received, $Time_{start\_packet}$ is the time when the first packet is sent. the multiplication by 8 is a conversion factor used to convert the data size from bits to bytes.

The Average End to end delay is calculated by the formula:

$$Average\ E2E\ Delay = \frac{\sum(Time_{end\_packet} - Time_{start\_packet})}{\sum Total_{Packets}} \tag{5.4}$$

☞ ***Throughput (Thp)***: the average throughput is the number of bits successfully arrived per second (kb/s) at the destination node. This is used to measure the protocol's reliability under different conditions; hence, the average throughput in the network needs to be as high as possible [Gar+18]. Mathematically, it can be defined as,

$$Throughput(Thp) = \frac{\sum Total_{Packets} * Packetsize * 8}{Time_{end\_packet} - Time_{start\_packet}} \tag{5.5}$$

☞ ***Jitter***: is the variation in the arrival times of consecutive data packets. In the Mobile Ad-hoc Networks (RFC5148) context, minimal variability in packet arrival times is essential for optimal performance. The jitter value, calculated according to RFC 3550, represents the degree of fluctuation in packet arrival times and is often measured in milliseconds (ms). The Jitter for the i^{th} packet is denoted as J_i and is calculated by the formula:

$$J_i = J_{i-1} + \frac{|D_{i-1}| - J_{i-1}}{16} \tag{5.6}$$

Where the gain parameter 1/16 is considered optimal, providing a good noise reduction ratio while maintaining a reasonable convergence rate, as Schulzrinne et al. [Sch+03] suggested. D_i represents the deviation from the expected arrival time for the i^{th} packet, calculated using the formula:

$$D_i = (R_i - R_{i-1}) - (S_i - S_{i-1}) \tag{5.7}$$

where R is the arrival time in UDP time tags, and S is the UDP time tag received from the packet.

☞ ***Reliability***: is the ability of a system to consistently and dependably deliver data with minimal or no loss or errors. It quantifies a system's ability to ensure unchanged and timely transmission, with the demanded level varying based on application criticality. High reliability is vital for applications intolerant to packet loss, while others with tolerance for occasional loss accept medium or low reliability. The is

reliability is calculated using the formula:

$$Reliability = \frac{\sum TransmitData}{\sum DeliveryData} * 100 \tag{5.8}$$

☞ ***Packet loss***: is the percentage of data packets that do not successfully reach their destination within a given transmission. Lower packet loss is generally desirable, indicating more robust and reliable communications performance. The formula calculates it:

$$Packet\ loss\ rate = \frac{Number\ of\ lost\ packets}{Total\ number\ of\ sent\ packets} * 100 \tag{5.9}$$

☞ ***Average energy consumption***: refers to the mean amount of energy utilized by the network nodes, typically UAV, within a FANET over a given duration. It is computed by dividing the total energy consumed by all nodes during that period by the duration of the observation.

$$Average\ Energy\ Consumption = \frac{Total\ Energy\ consumed\ by\ all\ Nodes}{Duration} \tag{5.10}$$

☞ ***Routing overheads***: are all control messages (cumulative routing overhead) in bytes (for each routing protocol) divided by the simulation period (time):

$$Routing\ overhead\ ratio = \frac{Total\ Routing\ overhead}{Total\ transmission\ time} \tag{5.11}$$

where *Total Routing Overhead* is the cumulative routing overhead (cost or resources expended on routing-related activities) incurred during the specified time period, and the *Total Transmission Time* is the overall duration of network operation or communication.

☞ ***Traffic Received***: represents the total amount of the total traffic (or data) received in bits per second by all traffic destinations (Nodes) in the entire network.

$$Total\ Traffic\ received = \sum_{i=1}^{n} Traffic_i \tag{5.12}$$

where n is the total number of traffic destinations (Nodes or sources), and $Traffic_i$ is the traffic (or data) received from the i-th source or destination in bits per second.

☞ ***Data dropped***: indicates total higher layer data traffic (in bits/sec) dropped by all the WLAN MACs. This occurs due to persistent retransmission failures, where higher-layer packets are dropped because the MAC fails to receive acknowledgments (ACKs) for (re)transmissions or their fragments. A lower data dropped value reflects

an improved transmission path capability and stability.

$$Data\ Dropped = \sum_{i=1}^{n} Dropped_i \tag{5.13}$$

where n is the total number of traffic destinations (Nodes or aircraft), and $Dropped_i$ represents the data dropped by the i-th node or aircraft in bits per second.

Table 5.2 presents all selected publications in this review with the performance metrics cited before.

Paper	**Performance metrics**										
	Packet Delivery Ratio (PDR)	**Packet delivery rate**	**End-to-End (E2E) delay**	**Throughput (Thp)**	**Jitter**	**Reliability**	**Packet loss**	**Average energy consumption**	**Routing overhead**	**Traffic Received**	**Data dropped**
Garcia et al. [Gar+18]		✓	✓	✓							
Hassan et al. [Has+20]			✓	✓			✓				
He et al. [HSS20]			✓	✓			✓		✓		
Kaur et al. [KSG20]	✓		✓	✓	✓						
Leonov et al. [LL18]			✓	✓	✓	✓					
Lin et al. [Lin+19]	✓		✓								
Pires et al.[PPB19]		✓	✓							✓	
Singal et al. [Sin+19]	✓		✓					✓			
Singh et al. [SV14]	✓		✓	✓							
Singh et al. [SV15]	✓		✓	✓							
Tan et al. [Tan+20]			✓	✓						✓	✓
Yang et al. [YL19]	✓		✓	✓							

Table 5.2 – Performance metrics used to evaluate the performance of network communication

We notice that some metrics seem almost similar, e.g., Packet Delivery Ratio (PDR) and Packet Delivery Rate.

In our case study on network communications quality, the selection of the three key performance metrics — Packet Delivery Ratio (PDR), End-to-End (E2E) delay, and Throughput (Thp) — is based on the review of existing literature, as evidenced by Table 5.2. These metrics emerge as the most prevalent and widely acknowledged indicators in assessing different aspects of network performance. Packet Delivery Ratio (PDR) provides a fundamental measure of reliability, indicating the percentage of successfully delivered packets. End-to-End (E2E) delay captures the responsiveness of the network by quantifying the time taken for packet traversal. Throughput (Thp) offers insights into the network's capacity and efficiency by measuring the throughput or data transfer rate. Focusing our study on these metrics ensures a comprehensive evaluation, aligning with established practices and facilitating meaningful comparisons with existing research.

5.3.2 Mode switching (Core of the method)

The "Mode switching" method aims to find and adapt the UAVs' operating mode online. Based on the UAV sensor data analysis, the local network data (direct inter-drone communications of the multi-UAV system), and the network data with the cloud, the method will estimate the UAV's autonomy (good functioning).

1. **Definition of a Bayesian network**

   Bayesian Network (BN) is used to perform a diagnostic reasoning and root cause analysis [Sch+15; Zer+17], leveraging its capability as a multivariate probability distribution that facilitates reasoning and learning under uncertainty [Pea88; Dar09]. In a BN, random variables are represented as nodes within a Directed Acyclic Graph (DAG), where the edges in the DAG [Cow+07] induce conditional dependencies and independencies between variables. Figure 5.2 illustrates a straightforward BN example. A BN's graphical structure often reflects a domain's causal relationships and generally serves as a concise representation of a conditional probability table. Each BN node is linked to a corresponding conditional probability table (CPT), capturing its causal links to parents and children in the DAG.

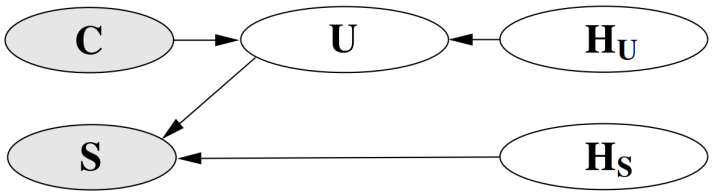


Figure 5.2 – Simple Bayesian network [Sch+15].

Considering the BN [Sch+15] in Figure 5.2, it consists of four different types of interconnected nodes, namely command node C, health node H, sensor node S, and status node U. The health node H has subtypes H_S for a sensor node and H_U for a status node.

In this BN, the observed values of S on the sensors enable the assessment of the health state of the system H_u. This is made possible through Bayesian reasoning by inference based on observations made on the system.

2. **Inference in Bayesian Network**

   Inference in Bayesian networks is a process that involves drawing conclusions or making predictions about the uncertain variables within the network, given observed evidence or information. Bayesian networks, also known as belief networks or probabilistic graphical models, employ Bayes' theorem to update probabilities and model dependencies among variables.

Bayes' theorem

Bayes' theorem, named after Thomas Bayes, is a fundamental concept in probability theory and statistics. It calculates the probability of an event by incorporating prior knowledge of conditions that may be associated with the event.
Bayes' theorem is stated mathematically as the following equation [SKO94]:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{5.14}$$

where A and B are events and $P(B) \neq 0$.

- $P(A|B)$ is a conditional probability: the probability of event A occurring given that B is true. It is also called the posterior probability of A given B.
- $P(B|A)$ is also a conditional probability: the probability of event B occurring given that A is true. It can also be interpreted as the likelihood of A given a fixed B because $P(B|A) = L(A|B)$.
- A and B are the probabilities of observing A and B, respectively, without any given conditions; they are known as the prior probability and marginal probability.

3. **Construction of the BN for mode switching**

The probability of the autonomy of the UAV is calculated by combining the different metrics most used in multi-UAV communication; considering the example in Figure 5.2, we have built the following Bayesian Network (BN) in Figure 5.3.

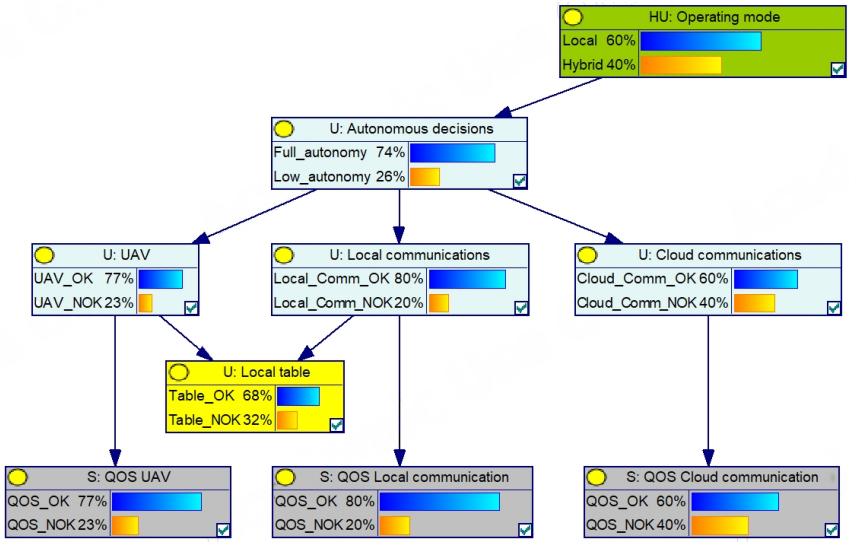


Figure 5.3 – Mode switching modeled with BN.

4. **Construction of the BN**

The systematic process used in the construction of the Bayesian Network (BN) facilitating the determination of the appropriate operating mode follows these steps:

(a) **Definition of all the variables and their states**: identify and define the variables (nodes) of the Bayesian Network along with all possible states for each variable.

Our Bayesian Network is made of several nodes like the following:

- ❖ **The root node "operating mode"** is the BN's estimation node of the conflict resolution module. Depending on this node, we will privilege one operating mode rather than another.
- ❖ The leaf nodes **"QoS UAV"**, **"QoS Local communication"**, and **"QoS Cloud communication"** are the leaf nodes of the BN computed and evaluated with health estimation modules.
  - ☞ **"QoS UAV" node**: serves as a robust indicator for assessing the overall health status of the UAV, focusing particularly on its hardware components. This node contains various aspects, including data processing within the UAV system, onboard processing, processors, memory, and the health status of essential sensors like GPS, IMU, and cameras. It also considers the condition of the batteries sustaining

the UAV's power supply, along with the health of applications running on the UAV. This comprehensive metric evaluates the robustness and efficiency of both the hardware elements and the associated software, contributing to a holistic assessment of the UAV's operational well-being.

- ☞ **"QoS Local communication" node**: serves as a strong indicator for evaluating the overall health status of the local communications within the UAV and its neighboring nodes. This node uses various metrics to estimate the reliability and safety of communications between a UAV and its neighbors. Specifically, it utilizes the three key metrics identified in Section 5.3.1: Packet Delivery Ratio (PDR), End-to-End (E2E) delay, and Throughput (Thp). These metrics provide a comprehensive communications quality assessment, addressing crucial aspects such as packet delivery, End-to-End delay, and data transfer efficiency.
- ☞ **"QoS Cloud communication" node**: serves as a robust indicator for assessing the overall health status of the cloud-based communications within the members of the multi-UAV system. The "QoS Cloud communication" uses the same metrics to estimate reliability and safety for the "QoS Local communication" node. Specifically, it utilizes the three key metrics identified in Section 5.3.1: Packet Delivery Ratio (PDR), End-to-End (E2E) delay, and Throughput (Thp).

❖ The **"Local table" node** depends on two intermediate nodes, which in turn are dependent on two leaf nodes, namely, S: UAV" and "S: Local communications". This table facilitates information sharing among the UAVs within the system, providing a comprehensive overview of the mission.

UAVs share information with their direct neighbors through local communications. In cases where conflicts within this local communication remain unresolved, the "Local table" is updated with additional information sourced from other UAVs within the multi-UAV system, facilitated through cloud communication. This adaptive mechanism ensures continuous information refinement, fostering effective communication and collaboration among the UAVs.

The observation made in the mission context is captured by the leaf nodes of the Bayesian network in the form of evidence according to the observed state of the node.

(b) **Definition of the BN structure**: establishes conditional dependency links among the various nodes in the network. Notably, a BN is inherently an acyclic

graph, prohibiting the presence of loops. Ensuring the smooth circulation of information among the nodes is essential to facilitate adequate reasoning and root cause analysis within the BN framework. D-separation rules govern this adherence to information flow principles [Pat+07], emphasizing the importance of maintaining proper connectivity while avoiding cycles in the graph.

The three cases in Table 5.3 cover all types of connections in a causal network.

Graph	Information circulation rule
V1 $\longrightarrow$ V2 $\longleftarrow$ V3	Type: convergent connection Information flows from V1 to V3 if only if V2 is known
V1 $\longleftarrow$ V2 $\longrightarrow$ V3	Type: divergent connection Information flows from V1 to V3 if only if V2 is not known
V1 $\longrightarrow$ V2 $\longrightarrow$ V3	Type: serial connection Information flows from V1 to V3 if only if V2 is not known

Table 5.3 – Circulation of information according to the type of connection in a BN.

Connections are established in the switching estimation mode based on the guidelines specified in Table 5.3. An example is provided below:

— Node **'U: UAV'** to node **'U: Local table'** ➛ poor QoS of the UAV causes unreliable data in the "Local table",

— Node **'U: Local communications'** to node **'U: Local table'** ➛ poor QoS of the local communications link causes unreliable data in the "Local table",

— etc.

The nodes **'U: UAV'**, **'U: Local table'**, and **'U: Local communications'** form a convergent connection. The nodes **'U: Local table'**, **'U: UAV'**, and **'S: QOS UAV'** form a divergent connection. The nodes **'S: QOS UAV'**, **'U: UAV'**, and **'U: Autonomous decisions'** form a serial connection

(c) **Definition of Bayesian Network (BN) parameters** is crucial for quantifying the probability values associated with each node, reflecting the likelihood of observing State 0 or State 1. For the root nodes of the BN (those without parents), these probabilities are known as a priori probabilities. These values represent the inherent likelihood of each possible state without being influenced by other nodes in the network. In contrast, for child nodes (nodes with parent or causal nodes), these parameters take the form of conditional probabilities, reflecting the probability of a particular state given the states of their parent

nodes. This distinction underscores the foundational role of BN parameters in encapsulating both prior knowledge and conditional dependencies within the network, facilitating accurate probabilistic modeling and reasoning

For each node of our BN, we set a conditional probability table (CPT), which contains the initial probability values as shown in Figure 5.4.

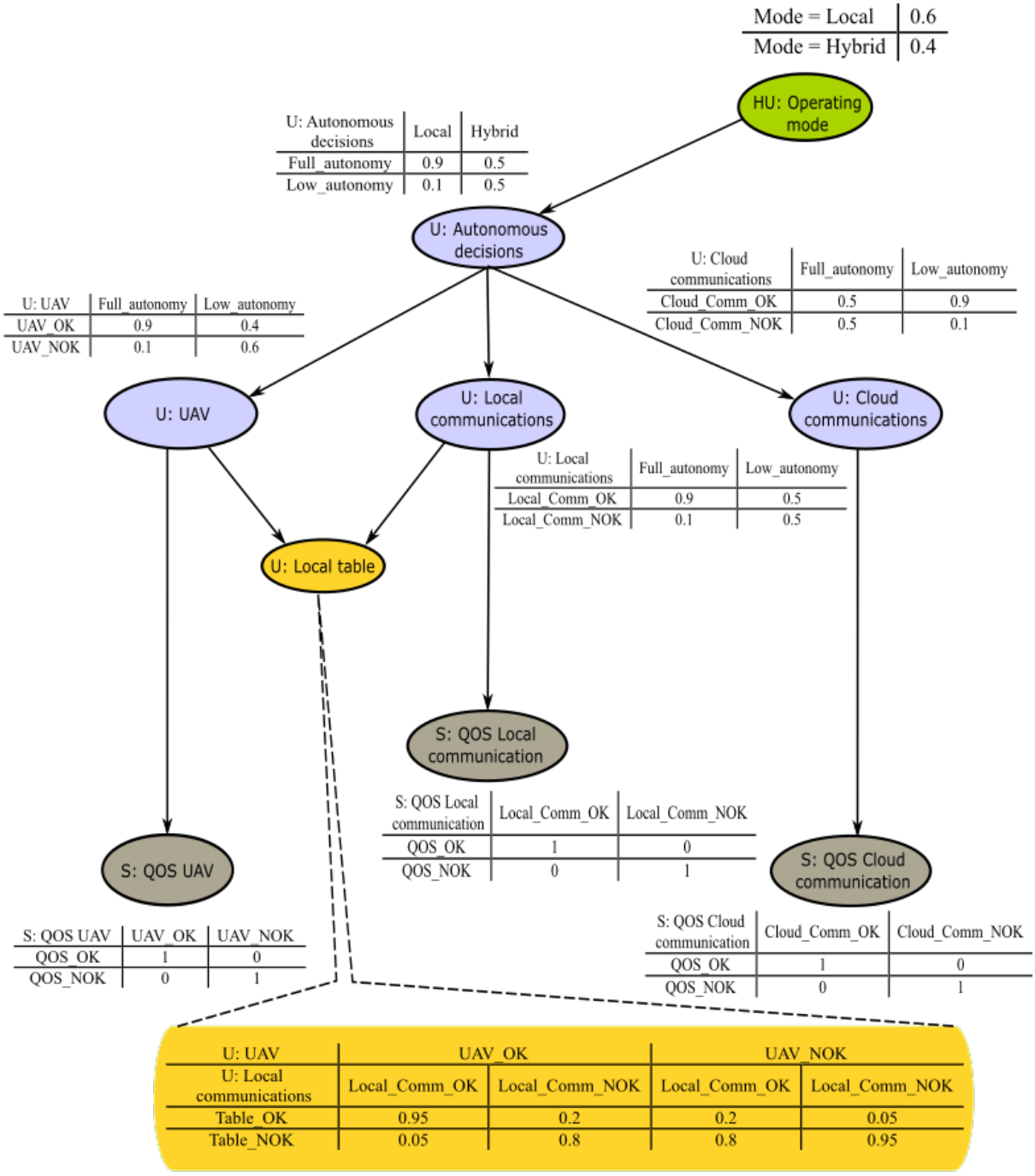


Figure 5.4 – Mode switching modeled with BN showing the conditional probability table (CPT).

5.4 Operating mode selection scenarios

This Section explores various scenarios for selecting operating modes within the Bayesian network framework. Specifically, we delve into selecting the Local mode in subsection 5.4.1 and the Hybrid mode in subsection 5.4.2. The Bayesian network can dynamically determine the most suitable mode based on different conditions and events. Each case within these sections outlines distinct conditions and reasoning processes that guide the Bayesian network in making optimal decisions for selecting either the Local or Hybrid operating modes. Let's examine the different cases within each mode selection to gain insights into the adaptive decision-making capabilities of the Bayesian network.

5.4.1 Selection of the Local mode

In the context of selecting the Local mode, the Bayesian network demonstrates its capacity to opt for the local operating mode in these various scenarios:

➥ **Case 1**: **QoS UAV, QoS Local communication, and QoS cloud communications are OK**, as shown in Figure 5.5.

In this scenario, the three estimators provide insights into the proper functioning of the UAV, local, and cloud communications. The circulation of information within this Bayesian network reinforces reliability, particularly on the Local Communication node, which is shared among UAVs.

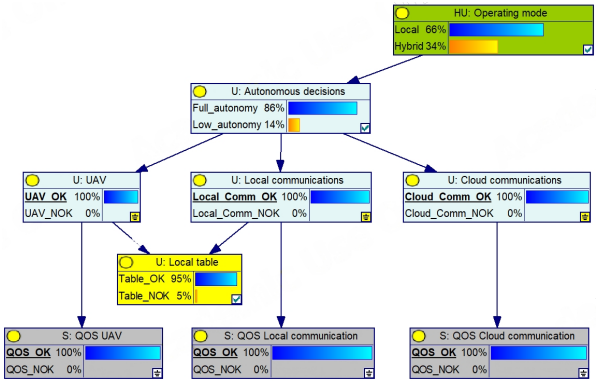


Figure 5.5 – QoS UAV, QoS local communication, and QoS Cloud communications are OK.

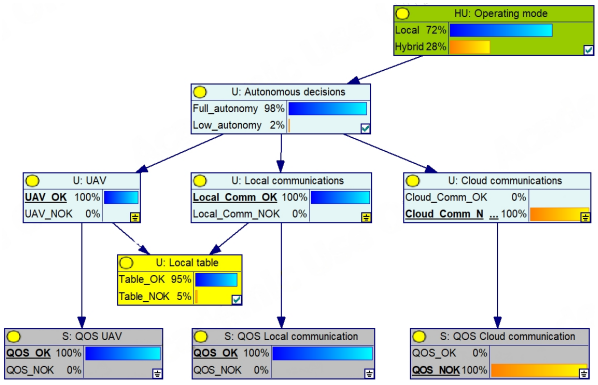


Figure 5.6 – QoS UAV and QoS local communications are OK, and QoS Cloud communications are not OK.

➥ **Case 2**: **QoS UAV and QoS local communications are OK, and QoS Cloud communications are not OK**, as shown in Figure 5.6

In this case, as illustrated in Figure 5.6, the QoS UAV and QoS Local communications are determined to be satisfactory. In contrast, the QoS Cloud communications

are flagged as unsatisfactory. The QoS UAV and QoS Local communications estimators offer insights into the proper functioning of the drone and local communications. Conversely, the cloud communications estimator signals a compromise in communications through the cloud. The circulation of information within this Bayesian network enhances reliability, particularly emphasizing the shared local communications node among the drones.

➥ **Case 3**: **QoS UAV and QoS Cloud communications are not OK, and QoS local communications are OK**, as depicted in Figure 5.7.

In this situation, the QoS local communications estimator provides insight into whether the UAV's local communications are functioning properly. The QoS UAV estimators indicate reduced confidence in the UAV due to issues that may be linked to sensors, battery, or autopilot malfunctions. Simultaneously, QoS cloud communications suggest compromised communications through the cloud.

The circulation of information within this Bayesian network reduces the drone's decision-making reliability and affects the shared table at the drone level. Despite good local communication, the Bayesian Network may delegate tasks to a direct neighbor, bypassing the cloud, to maintain operational efficiency.

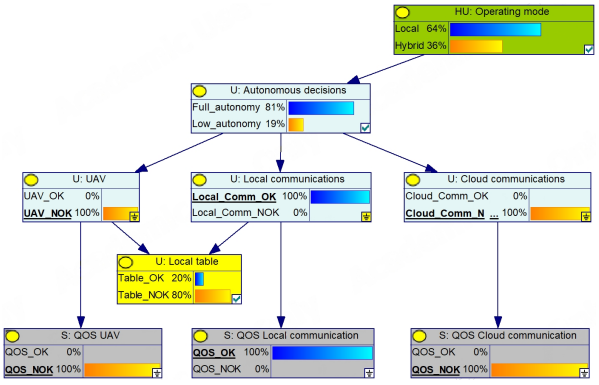


Figure 5.7 – QoS UAV and QoS Cloud communications are not OK, and QoS local communications are OK.

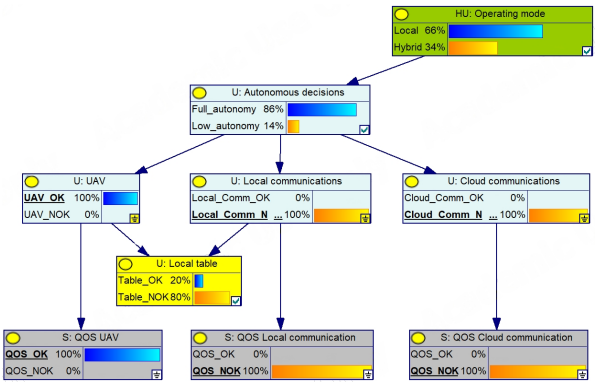


Figure 5.8 – QoS UAV is OK, and QoS Cloud communications and QoS local communications are not OK.

➥ **Case 4**: **QoS UAV is OK, and QoS Cloud communications and QoS local communications are not OK**; is shown in Figure 5.8

In this scenario, the QoS UAV estimator, along with the QoS Local Communication, offers valuable insights into the proper functioning of the UAV and its local communications with direct neighbors. Concurrently, the QoS cloud Communications estimator indicates a cloud communications threat. The circulation of information within the Bayesian network reduces the reliability of both types of communication. However, despite both types of communication being compromised, the Bayesian

Network has no choice other than to delegate the task to a local neighbor since neither cloud nor local communications are dependable.

5.4.2 Selection of the Hybrid mode

Turning our attention to the Hybrid mode selection, the Bayesian network demonstrates its capability to choose the Hybrid operating mode in these cases:

➥ **Case 5**: **QoS UAV is not OK, and QoS local communications and QoS Cloud communications are OK**; is shown in Figure 5.9

In this scenario, the QoS UAV estimator indicates a malfunction in the UAV, reducing its capacity to make safe decisions. Concurrently, the QoS cloud communications and QoS Local communications estimators indicate that direct communications with neighbors are robust, and cloud communications are functional and secure. The circulation of information within the Bayesian network decreases the reliability of the shared "Local table", impacting the effectiveness of local communications with neighbors. However, given the assurance of good cloud communications, the Bayesian Network will delegate the task to another UAV within the system through the cloud, operating in Hybrid mode.

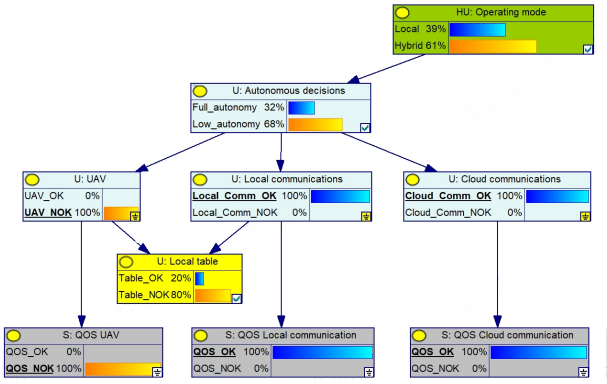


Figure 5.9 – QoS UAV is not OK, and QoS local communications and QoS Cloud communications are OK.

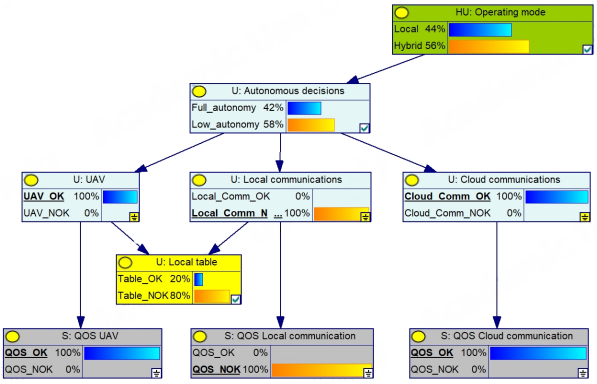


Figure 5.10 – QoS UAV and QoS Cloud communications are OK, and QoS local communications are not OK.

➥ **Case 6**: **QoS UAV and QoS Cloud communications are OK, and QoS local communications are not OK**; is shown in Figure 5.10

In this scenario, the QoS UAV and QoS cloud Communications estimators offer valuable insights into the proper operation of the UAV and cloud communications. Simultaneously, the QoS Local Communication estimator indicates that direct communications with neighbors are compromised and unsafe, while communications via the cloud are functional and secure. The circulation of information within the

Bayesian network reduces the reliability of the shared "Local table", impacting the effectiveness of direct communications with neighbors. Nonetheless, given the reliability of cloud communications and the unsatisfactory state of local communications, the Bayesian Network delegates the task to another UAV within the system via the cloud, operating in Hybrid mode.

➥ **Case 7**: **QoS UAV and QoS local communications are not OK, and QoS Cloud communications are OK**, is shown in Figure 5.11.

In this scenario, the QoS UAV and QoS Local communications estimators provide crucial insights into the malfunction of the UAV and the poor, unsafe state of local communications with neighbors. Conversely, the QoS cloud communications estimator indicates that communications via the cloud are operational and secure. The circulation of information within the Bayesian network reduces the reliability of the shared "Local table", affecting the effectiveness of local communications with neighbors. Nevertheless, given the reliability of cloud communications and the compromised state of local communications, the Bayesian Network chooses to delegate the task to another UAV within the system via the cloud, operating in Hybrid mode.

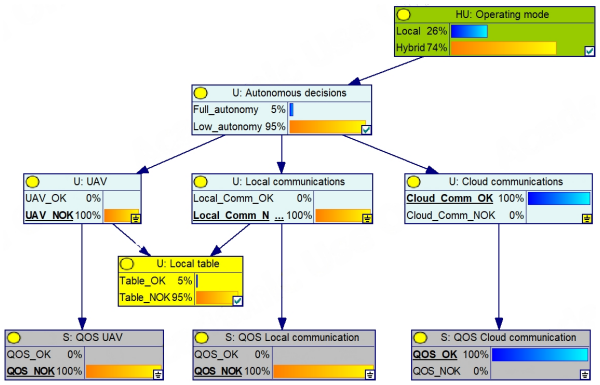


Figure 5.11 – QoS UAV and QoS local communications are not OK, and QoS Cloud communications are OK

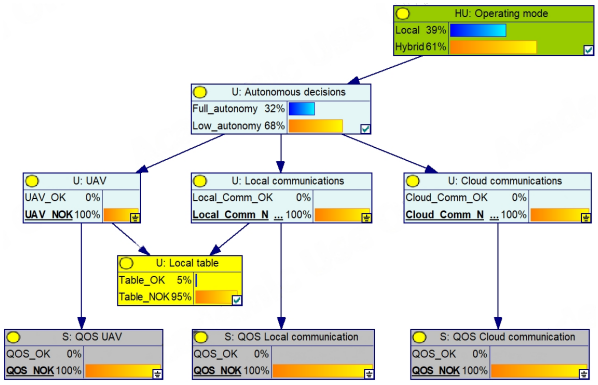


Figure 5.12 – QoS UAV, QoS local communication, and QoS Cloud communications are not OK

➥ **Case 8**: **QoS UAV, QoS local communication, and QoS Cloud communications are not OK**, is shown in Figure 5.12.

In this exceptional scenario, the estimators of QoS UAV, QoS Local communication, and QoS cloud communications are all malfunctioning. Given the UAV's inability to make decisions and the unsafe state of local communications, the circulation of information within the Bayesian network reduces the reliability of the "Local table", impacting both local and cloud communications.

Despite the poor state of communications via the cloud, the Bayesian Network has only one option that is to delegate the task to one of the system's UAVs via the cloud,

operating in Hybrid mode. This decision is driven by the compromised functionalities of both the UAV and local communications, emphasizing the adaptability of the Bayesian Network in responding to diverse scenarios.

5.5 Case study with a collaborative mission planning

These are some examples of conflicts/constraints at the multi-UAV level:

- ☞ **Constraint 1**: The UAV with the smallest distance to target $Target_t$ is the only one to track it.
- ☞ **Constraint 2**: The UAV_i follows $Target_t$, and the speed of the UAV follower $<$ speed of the $Target_t$
- ☞ **Constraint 3**: The UAV_i follows $Target_t$, and one of its sensors fails, e.g., GPS
- ☞ **Constraint 4**: The UAV_i follows $Target_t$ and will not have enough energy resources to continue the mission or make RTH (return to base).
- ☞ **Constraint 5**: The UAV_i follows $Target_t$, which enters an unauthorized area for the follower.

The tracking mission described in Section 4.1.3 is assigned to a multi-UAV system. We have used three UAVs for simplicity's sake. As shown in Figure 5.13, the UAVs in this system will track a target (e.g., the red UAV).

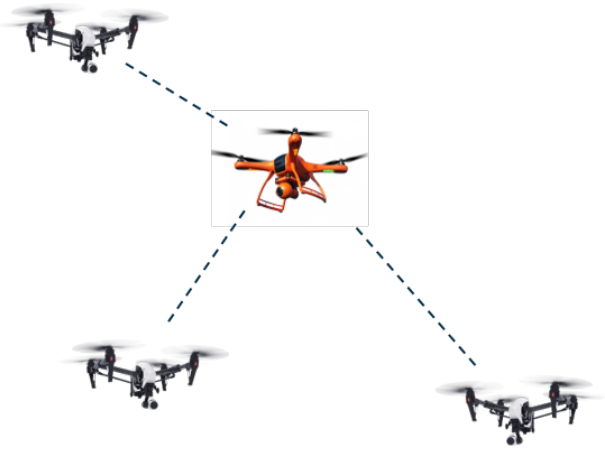


Figure 5.13 – Tracking mission with multi-UAV system

5.5.1 Multi-UAV system searching target process

In multi-UAV system searching targets, UAVs must reach a consensus on a target to follow, which can be a complex task requiring effective coordination. Here are the general steps involved in reaching a consensus between these UAVs:

1. **Communication and coordination**: UAVs must be able to communicate with each other. This can be achieved via radio links, ad hoc communications networks, direct, local UAV-UAV communication, or through the cloud.
2. **Data collection**: Each UAV must collect information about the potential target, such as its position, speed, heading, etc. On-board sensors, such as cameras or radars, can be used for this purpose.
3. **Data exchange**: UAVs share the data they have collected on the target with other group members. The communications can be centralized or decentralized, depending on the control architecture.
4. **Data fusion**: information shared by UAVs is fused to estimate the target better. Data fusion techniques, such as Bayesian data fusion, can be employed to achieve a more comprehensive understanding of the situation
5. **Decision-making**: UAVs use the merged data to decide which target to follow and which UAV will do it. The decision-making may involve an assessment of priorities or pre-defined decision criteria.
6. **Repeating the process**: steps 2 to 5 are repeated periodically to update the decision on the target to follow as the situation evolves. This reiteration may be necessary due to target movements or the arrival of new information.
7. **Consensus**: is reached when the majority of UAVs converge on a common decision about which target to track and which UAV will track it. This may require voting mechanisms or other decision-making protocols.
8. **Target tracking**: once consensus has been reached, especially in the case of UAV target tracking, the UAVs collaboratively coordinate their movements to follow the chosen target precisely. They adjust their trajectories, ensuring seamless coordination while maintaining constant surveillance. In the case of individual tracking, the UAV selected from the consensus will dynamically change its trajectory to track the target effectively.

In this process, UAVs must be equipped with advanced navigation systems, quality sensors, and control software to implement this consensus process effectively. In addition, mission planning, resource management, and allocating roles within the UAV team are also crucial to the success of the target-tracking mission.

Among the conflicts and constraints listed above, we have chosen to take the 1st constraint as a case study to demonstrate the effectiveness of our method.

5.5.2 Local communications are safe and functioning effectively

In the case of the good functioning of local communications (UAV-UAV),

- A team of UAVs are deployed to the search area, and they establish a local communications network among themselves. This network enables real-time data sharing and coordination.
- We assume that the UAVs are programmed to autonomously search for targets in the search area using a combination of high-resolution cameras, thermal imaging, and LIDAR sensors.
- When a UAV detects a potential target using its sensors, it constructs an internal table with the collected data, as shown in Figure 5.14. Its table is periodically shared with its neighbors. After processing the Bayesian Network with various UAV data and communication safety data, it concludes that local communications are safely detected by the monitoring system in the UAV, and the "Local Table" data is not compromised. This increases the probability of operating autonomously, and as this information propagates towards the root node of the BN, the system's health promotes a preference for local operating mode.
- Conflicting UAV relays the information to other UAVs via the local communications network, as shown in Figure 5.15. The UAVs collectively analyze the data to confirm the presence of a target and classify the target according to the pre-defined factors and criteria.

Figure 5.14 – Detection of a target and construction of an internal table.

- UAVs engaged in the search shares their findings, and the group reaches a consensus on the priority of targets based on factors such as proximity, condition, and accessibility.
- The drones work together to assign tasks, so in this case study, the tracking task.
- The UAV assigned to track the target follows the target. The UAV maintains communications with the other system members.

The UAVs continue to share data and update the data in each member's internal table and with the central command center. This data includes the location and velocity of the

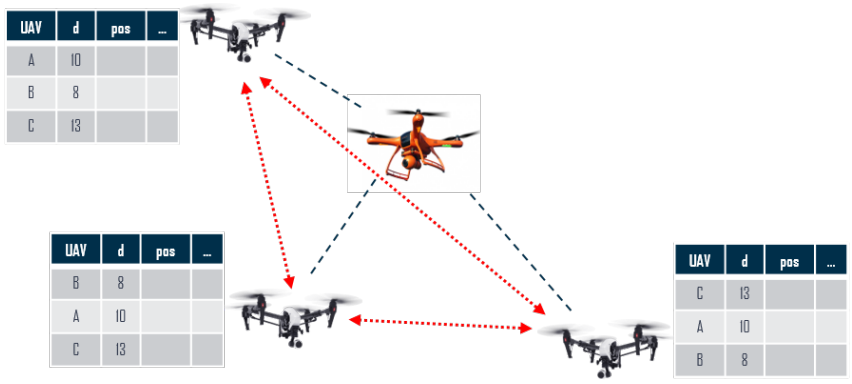


Figure 5.15 – Mission data exchange and fusion.

target, any additional threats, and the overall progress of the tracking mission.

✍ Note that this case study concerns **Case 1** (see subsection 5.4.1) among several instances of local operating mode selection scenarios. This is because communications are secure, and neighboring UAVs trust the data from their "Local table". This trust enables the system to operate locally, bypassing the need to rely on cloud services for conflict resolution.

5.5.3 Local communications are under potential risks or vulnerabilities

In the event of potential risks or vulnerabilities in local communications among UAVs during a mission, as shown in Figure 5.16, the management of this mission will be done as follows:

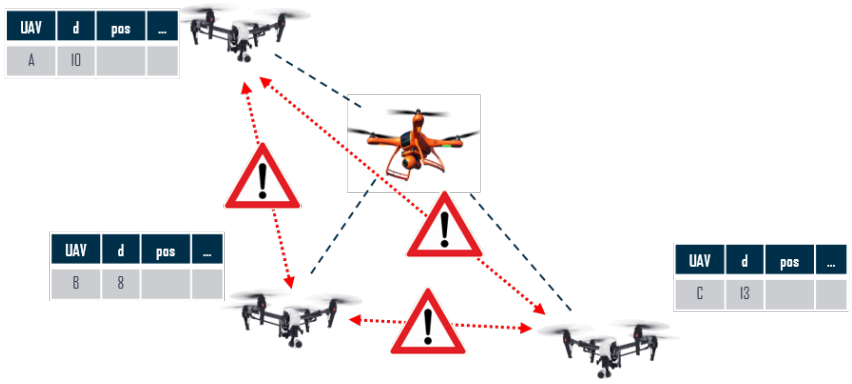


Figure 5.16 – Detection of a target, construction of an internal table for each UAV, and appearance of potential local communications vulnerabilities.

➥ A team of UAVs is deployed to the search area. The UAVs start their search operations with an awareness of the communications challenges because of concerns

about potential communications vulnerabilities. However, they maintain connectivity through the cloud.

- ➥ We assume that the UAVs are programmed to autonomously search for targets in the search area using a combination of high-resolution cameras, thermal imaging, and LIDAR sensors.
- ➥ When a UAV detects a potential target using its sensors, it constructs an internal table with the collected data as shown in Figure 5.14. Its table is periodically shared with its neighbors. After processing the Bayesian Network with these data, it determines that local communications are not secure, and the data in the Local table is also deemed unsafe and may be compromised. This reduction in the probability of autonomous operation propagates this information to the root node of the BN, indicating that the overall system health does not support relying only on direct neighbors. Therefore, there is a need to favor a mode of operation via the cloud.
- ➥ The UAVs use the cloud as a reliable communications channel to exchange data and coordinate their search and rescue efforts. This ensures that they can maintain connectivity even if local communications are intermittent.
- ➥ While local communications remains unreliable, UAVs adapt their decision-making processes. They can still collaborate in real-time and make coordinated decisions based on cloud-shared information, as shown in Figure 5.17.

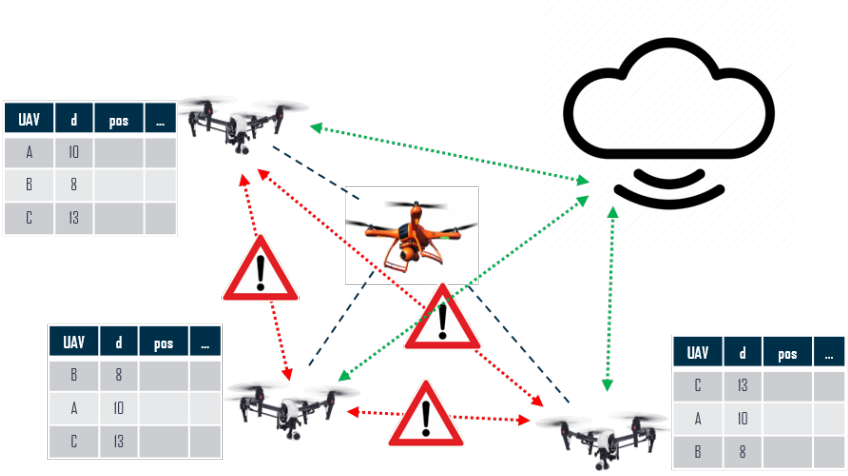


Figure 5.17 – Mission data exchange and fusion through the cloud-based communication.

UAVs engage in consensus-building processes to agree on which targets to prioritize and how to coordinate their efforts. Cloud-based communications support effective decision-making.

- ➥ The tracking task is centrally assigned based on the consensus reached through cloud communication, ensuring efficient allocation of resources.

✍ Note that this case study corresponds to the **Case 7** (see subsection 5.4.2) of selecting Hybrid operating mode. This is because UAV data and local communications are not securely detected by the monitoring system in the UAV. As a result, UAVs no longer trust the data in their "Local table". This reduces confidence in the system to operate locally, thus leading to the need to rely on cloud services for conflict resolution.

Although cloud communications are available as a backup, UAVs can utilize offline storage capabilities for critical data. Also, UAVs will periodically attempt to restore local communications but maintain cloud connectivity as their primary means of communication.

5.6 Final consideration

In conclusion, this chapter underscores the indispensable role of communication in achieving mission objectives, particularly in distributed and cooperative missions within multi-UAV systems. We introduced a sophisticated module to enhance the adaptive capabilities of collaborative multi-UAV systems through a Bayesian Network (BN)-based operating mode adaptation. The focus is on policy self-adaptation in local and hybrid modes (via the cloud) and exploiting drone communication networks to optimize drone mission planning.

The chapter describes the disparities in applying the self-adapting policy method in local and cloud modes. It begins by elucidating the existing communication networks used for multi-UAV system communication, primarily local (UAV-UAV) and cloud-based communications while addressing associated issues. When local communications face potential vulnerabilities, alternative communication channels, such as cloud-based communication, are key in enhancing mission efficiency.

We propose an autonomous local/hybrid mode-switching approach, considering UAV and communication data to predict drone autonomy and required communication mode (local/hybrid mode). This prediction is made by analyzing the UAV system's behavior under uncertainty and considering sensor data and network data from local neighbors. Based on this analysis, the module can estimate the optimal operating mode. Finally, we outline scenarios for both local and hybrid utilization modes, demonstrating the effectiveness of applying the self-adapting policy method combined with local and hybrid mode switching in a collaborative mission with a multi-UAV target search system.

The subsequent chapter provides a comprehensive summary of the work conducted in this thesis and outlines potential avenues for future research.

CHAPTER 6

Conclusion and perspectives

This thesis focuses on distributed decision-making in multi-UAV systems in uncertain environments. This chapter briefly presents the context of this thesis, highlights our contributions, and then outlines prospects for future research.

Contents

6.1 Conclusion

6.1.1 Context of the thesis

Nowadays, robots have invaded almost all fields: industrial (e.g., assembly robot in a production line), agricultural (e.g., analysis of fertilizer needs), environmental (e.g., monitoring of natural disasters), commercial and transportation (e.g., delivery), security sectors (e.g., forest fire detection), military missions, etc. The production and use of drones have increased and require improved decision-making, safety, security, and knowledge of relevant areas. They must continually adapt to accomplish missions facing unpredictable problems.

With a focus on missions in the maritime environment, aerial drones are essential. UAVs complement traditional maritime vehicles, providing advantages in tasks such as

searching, identifying objects or individuals, detecting pollution, monitoring ports, and reconstructing 3D images. Collaboration of maritime vehicles with drones extends to innovative roles, such as using them as motherships to recharge drones, optimize mission duration, and improve overall efficiency. Despite these benefits, executing maritime missions presents challenges due to unpredictable weather conditions, communications interference, and the need for resilient navigation systems. Specific examples include poor visibility and vulnerabilities such as GPS outages caused by factors such as inadequate satellite coverage or deliberate interference. Addressing these challenges requires scientific research and technological advancements to strengthen air systems, ensure adaptability and robust performance in complex maritime missions, and emphasize an interdisciplinary approach for optimal contribution to exploration and problem-solving efforts.

6.1.2 Problem statement

Chabha Hireche's thesis work [Cha19] explored decision models used in the context of a drone mission. It confirmed that probabilistic models, particularly Markov Decision Processes (MDP), are widely adopted for decision-making under uncertainty in robotics.

In the course of this research, our primary goal was to determine the most suitable solving method for a decision problem under uncertainty as a function of the context of a mission in the embedded card.

In various applications, limitations in energy, computing capacity, and execution often made a single Unmanned Aerial Vehicle (UAV) inadequate for extensive coverage. Multi-UAV systems, such as swarms, aimed to achieve broader coverage, enhanced surveillance, and more efficient mission execution through collaborative action for rapid and precise information collection.

Within a multi-UAV system, we focused on uncovering potential conflicts and exploring how they could be resolved in a distributed multi-UAV framework modeled using MDP.

Finally, given our keen interest in collaborative missions involving multi-UAV systems, the ultimate goal was twofold. First, we aimed to address how one can ensure that a drone can carry out its mission autonomously. Second, we sought to implement conflict resolution within a multi-UAV system. This involved minimizing cloud communications and reducing the volume of transmitted data to maximize the autonomy of drones.

6.1.3 Contributions

This section briefly outlines key contributions, thoroughly examining three fundamental methods for MDP resolutions in mission planning. It introduces a novel approach

involving self-adaptation through rewards tuning in mission planning strategies. Finally, it explores the adaptation of collaborative multi-UAV systems via a mode-switching adaptation based on a Bayesian network.

- ✍ **Comparison of fundamental methods for mission planning**:
  The literature encompasses various probabilistic models for decision-making under uncertainty, including Markov Decision Process (MDP), Partially Observable Markov Decision Process (POMDP), Relational Dynamic influence Diagram Language (RDDL), and Petri Net (PN). This study specifically focuses on the widely adopted MDP framework in robotics, exploring its resolution methods. The three fundamental classes for solving finite MDPs are Dynamic Programming (DP), Monte Carlo (MC) methods, and Temporal Difference (TD) learning. Specifically, the research delves into three methods for solving MDPs Value Iteration and Policy Iteration methods (DP) and the Q-Learning method (TD). The study introduces new criteria to adapt decision-making methods to specific application problems. Experimental findings indicate that the Q-Learning method is effective in simple and regular cases. In contrast, classical MDP resolution methods are more suitable for irregular cases, especially in critical systems.
- ✍ **Self-adaptation based rewards tuning for mission planning**:
  MDP is a robust approach for decision-making and planning in uncertain UAV environments. It involves an agent interacting with the environment, adjusting actions based on state transitions and rewards. Unlike other reinforcement learning approaches, MDPs use probabilities and/or rewards to expedite problem-solving, incorporating factors such as physical constraints, mission safety, and conflict resolution in decision-making. The focus is on tuning rewards associated with actions, as experimentation reveals that modifying state-level rewards adapts the mission by resolving conflicts. The application extends to multi-UAV contexts, addressing challenges in autonomous vehicle development. At the multi-UAV system level, individual UAVs have unique MDPs, posing potential conflicts. The proposed conflict management mechanism involves adjusting mission plans, resolving conflicts, dynamic reward shaping through systematic online methods, implementing energy-aware strategies, and ensuring optimal decision-making, safety, and efficiency in multi-UAV system missions.
- ✍ **Adaptation with cooperation through the local network/the cloud**:
  Potential failures and cyber threats pose challenges in the context of a collaborative multi-UAV system operating in uncertain environments. Previous work suggested an adaptive system, but there's a need to enhance collaboration management among system members. The proposed approach utilizes Bayesian Networks to estimate

UAV safety, enabling direct communication with neighboring UAVs for internal adaptation. The goal is to maintain autonomy while fostering effective collaboration. In the event of compromised local communications, the adaptation seamlessly transitions to the cloud, reducing inter-drone communication and enhancing system efficiency. This operational model also decreases the transmitted data volume, reducing energy consumption in the multi-UAV system during missions.

6.2 Research perspectives

This Section presents some perspectives for future work, which we classify according to their term.

6.2.1 Short-term perspectives

In this section, we give the possible improvements of the work presented in this thesis.

1. **Mission planning**: involves integrating the Bayesian Network (BN)-based operating mode adaptation into the Markov Decision Process (MDP)-based decision engine within the context of mission planning. Up to this point, the validation of the two proposed modules in chapters 4 and 5 has been conducted individually. The next step is to pair these modules and synergistically assess their cohesive functioning. Initial validation will be performed through simulation, focusing on scenarios involving challenges at the UAV level or attacks on communication systems. This iterative approach thoroughly evaluates the integrated system's robustness and effectiveness in addressing complex real-world scenarios.
2. **BN-based operating mode adaptation**: as suggested in Chapter 5, the current framework assumes that the BN nodes receive the probability of good functioning from the UAV, local communication, and cloud communication from probabilistic estimators. So, we propose first defining these Quality of Service (QoS) estimators and subsequently integrating them into the proposed BN. This integration aims to enhance self-adaptation management, providing a more comprehensive and refined foundation for the decision-making processes within the system.
3. **Definition of an IT method/tool** to facilitate these estimators' specification, generation (in both software and hardware versions), and integration of estimators within the UAV or multi-UAV system. This innovative method/tool is anticipated to offer a systematic and precise approach to designing and implementing estimators (system monitors).

4. **Exploration of various support options** such as CPU, GPU, and FPGA for implementing monitors is crucial to address real-time and energy constraints in online UAV applications. This becomes particularly significant given the limited flight duration of UAVs, often equipped with small batteries. By considering different hardware support options, we aim to optimize the implementation to meet stringent performance and energy efficiency requirements.

6.2.2 Medium to long-term perspectives

Regarding future prospects, the following are some ideas for improving and extending the current approach.

1. **Study of the cooperation between heterogeneous drones (aerial/terrestrial /marine)**:
   This thesis focuses on collaboration within a homogeneous multi-UAV system, wherein multiple aerial UAVs execute the same decision model as a decision engine. However, we posit that extending the exploration to include heterogeneous UAVs could potentially enhance the effectiveness of mission outcomes. Heterogeneous UAVs, executing diverse decision models, introduce a layer of complexity that may contribute to improved mission adaptability and performance. Future research endeavors could delve into the integration and coordination of such heterogeneous systems, providing a deeper understanding of their collective capabilities and potential advantages in diverse operational scenarios.
2. **Study of the collaborative ML paradigm "Federated Learning (FL)"**:
   Federated Learning (FL) provides another decentralized framework where drones collectively improve their decentralized decision-making capabilities while respecting data confidentiality. It enhances efficiency by minimizing communication with a central server, ensuring failure resilience, and bolstering system robustness. Collaborative learning facilitates knowledge-sharing, strengthening mission resilience. Federated Learning's scalability makes it suitable for large-scale multi-UAV systems, accommodating growing drone numbers efficiently.

References

[AAK19] Ebtehal Turki Alotaibi, Shahad Saleh Alqefari, and Anis Koubaa, « Lsar: Multi-uav collaboration for search and rescue missions », *in*: *IEEE Access* 7 (2019), pp. 55817–55832, DOI: 10.1109/ACCESS.2019.2912306.

[AK20] Shubhani Aggarwal and Neeraj Kumar, « Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges », *in*: *Computer Communications* 149 (2020), pp. 270–299, DOI: 10.1016/j.comcom.2019.10.014.

[AKY21] Amjaad Alhaqbani, Heba Kurdi, and Kamal Youcef-Toumi, « Fish-Inspired Task Allocation Algorithm for Multiple Unmanned Aerial Vehicles in Search and Rescue Missions », *in*: *Remote Sensing* 13.*1* (2021), p. 27, DOI: 10.3390/rs13010027.

[AOÖ17] Emrah Adamey, Abdullah Ersan Oğuz, and Ümit Özgüner, « Collaborative multi-MSA multi-target tracking and surveillance: A divide & conquer method using region allocation trees », *in*: *Journal of Intelligent & Robotic Systems* 87.*3* (2017), pp. 471–485, DOI: 10.1007/s10846-017-0499-4.

[Aue+18] Lukas Auer et al., « Swarm-technology for Large-area Photogrammetry Survey and Spatially Complex 3D Modelling », *in*: *International Journal of Latest Research in Engineering and Technology* 4.*9* (2018), pp. 33–39, URL: https://www.researchgate.net/profile/Markus-Keuschnig/publication/328149194_Swarm-technology_for_Large-area_Photogrammetry_Survey_and_Spatially_Complex_3D_Modelling/links/5bbb996c299bf1049b74fafd/Swarm-technology-for-Large-area-Photogrammetry-Survey-and-Spatially-Complex-3D-Modelling.pdf.

[Bar19] Jayme Garcia Arnal Barbedo, « A review on the use of unmanned aerial vehicles and imaging sensors for monitoring and assessing plant stresses », *in*: *Drones* 3.*2* (2019), p. 40, DOI: 10.3390/drones3020040.

[Bel58] Richard Bellman, « Dynamic programming and stochastic control processes », *in*: *Information and control* 1.*3* (1958), pp. 228–239, DOI: 10.1016/S0019-9958(58)80003-0.

[Bel66] Richard Bellman, « Dynamic programming », *in*: *Science* 153.*3731* (1966), pp. 34–37, DOI: 10.1126/science.153.3731.34.

[CAM20] Marco Calderón, Wilbert G Aguilar, and Darwin Merizalde, « Visual-Based Real-Time Detection Using Neural Networks and Micro-UAVs for Military Operations », *in*: *International Conference of Research Applied to Defense and Security*, Springer, 2020, pp. 55–64, DOI: 10.1007/978-981-15-4875-8_5.

[CCK16] Brandon Cook, Kelly Cohen, and Elad H Kivelevitch, « A fuzzy logic approach for low altitude uas traffic management (UTM) », *in*: *AIAA Infotech@ Aerospace*, 2016, p. 1905, DOI: 10.2514/6.2016-1905.

[Cha+05] Iadine Chadés et al., « Markov decision process (mdp) toolbox v2. 0 for matlab », *in*: *INRA Toulouse, INRA, France* (2005), URL: http://www.inra.fr/internet/Departements/MIA/T/MDPtoolbox.

[Cha19] Hireche Chabha, « Étude et implémentation sur SoC-FPGA d'une méthode probabiliste pour le contrôle de mission de véhicule autonome. », PhD dissertation, UNIVERSITE DE BRETAGNE OCCIDENTALE, 2019.

[Che+20] Shazhou Chen et al., « A Warehouse Management System with UAV Based on Digital Twin and 5G Technologies », *in*: *2020 7th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS)*, IEEE, 2020, pp. 864–869, DOI: 10.1109/ICCSS52145.2020.9336832.

[Chu+18] Soon-Jo Chung et al., « A survey on aerial swarm robotics », *in*: *IEEE Transactions on Robotics* 34.*4* (2018), pp. 837–855, DOI: 10.1109/TRO.2018.2857475.

[CMO16] Jesús Capitan, Luis Merino, and Aníbal Ollero, « Cooperative decision-making under uncertainties for multi-target surveillance with multiples UAVs », *in*: *Journal of Intelligent & Robotic Systems* 84.*1* (2016), pp. 371–386, DOI: 10.1007/s10846-015-0269-0.

[Cow+07] Robert G Cowell et al., *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*, Springer Science & Business Media, 2007, DOI: 10.1111/j.1751-5823.2008.00054_8.x.

[Cui+17] Qiannan Cui et al., « Brief analysis of drone swarms communication », *in*: *2017 IEEE International Conference on Unmanned Systems (ICUS)*, 2017, pp. 463–466, DOI: 10.1109/ICUS.2017.8278390.

[Dà-] Ltd. Dà-Jiāng Innovations Science and Technology Co., *DJI—The Future of Possible.* Available online: https://www.dji.com, Accessed: 2021-04-01.

[DA19] CLOAREC Daniel and COUËTIL Anne, *Vous avez dit marétique ? Des opportunités à saisir à la confl uence de la mer et du numérique en Bretagne*, Last accessed 05 December 2023, 2019, URL: https://www.bretagne.bzh/app/uploads/sites/8/2022/04/ceser_vous_avez_dit_maretique_rapport_web.pdf.

[Dai+19] Fei Dai et al., « Swarm intelligence-inspired autonomous flocking control in UAV networks », *in*: *IEEE Access* 7 (2019), pp. 61786–61796, DOI: 10.1109/ACCESS.2019.2916004.

[Dar09] Adnan Darwiche, *Modeling and reasoning with Bayesian networks*, Cambridge university press, 2009, URL: http://hdl.handle.net/10214/8932.

[Dud+96] Gregory Dudek et al., « A taxonomy for multi-agent robotics », *in*: *Autonomous Robots* 3.*4* (1996), pp. 375–397, DOI: 10.1007/BF00240651.

[Ebe+20] Julia T Ebert et al., « Bayes bots: collective Bayesian decision-making in decentralized robot swarms », *in*: *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 7186–7192, DOI: 10.1109/ICRA40945.2020.9196584.

[EMA20] Mohamed M Eltabey, Ahmed A Mawgoud, and Amr Abu-Talleb, « The Autonomy Evolution in Unmanned Aerial Vehicle: Theory, Challenges and Techniques », *in*: *International Conference on Advanced Intelligent Systems and Informatics*, Springer, 2020, pp. 527–536, DOI: 10.1007/978-3-030-58669-0_47.

[FC18] Benjamin T Fraser and Russell G Congalton, « Issues in Unmanned Aerial Systems (UAS) data collection of complex forest environments », *in*: *Remote Sensing* 10.*6* (2018), p. 908, DOI: 10.3390/rs10060908.

[FFG19] Xiaowei Fu, Peng Feng, and Xiaoguang Gao, « Swarm UAVs task and resource dynamic assignment algorithm based on task sequence mechanism », *in*: *IEEE Access* 7 (2019), pp. 41090–41100, DOI: 10.1109/ACCESS.2019.2907544.

[Gar+18] Alejandro Garcia-Santiago et al., « Evaluation of AODV and DSDV routing protocols for a FANET: Further results towards robotic vehicle networks », *in*: *2018 IEEE 9th Latin American Symposium on Circuits & Systems (LASCAS)*, IEEE, 2018, pp. 1–4, DOI: 10.1109/LASCAS.2018.8399972.

[Gen+16] Desirée Gentilini et al., « Multi agent path planning strategies based on Kalman Filter for surveillance missions », *in*: *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, IEEE, 2016, pp. 1–6, DOI: 10.1109/RTSI.2016.7740591.

[Gha+21] Yahya Ghassoun et al., « Implementation and Validation of a High Accuracy UAV-Photogrammetry Based Rail Track Inspection System », *in*: *Remote Sensing* 13.*3* (2021), p. 384, DOI: 10.3390/rs13030384.

[Gha03] Zoubin Ghahramani, « Unsupervised learning », *in*: *Summer school on machine learning*, Springer, 2003, pp. 72–112, DOI: 10.1007/978-3-540-28650-9_5.

[Glo00] Pierre Yves Glorennec, « Reinforcement learning: An overview », *in*: *Proceedings European Symposium on Intelligent Techniques (ESIT-00), Aachen, Germany*, Citeseer, 2000, pp. 14–15, URL: https://www.researchgate.net/publication/2869722_Reinforcement_Learning_an_Overview.

[Gu+18] Jingjing Gu et al., « Multiple moving targets surveillance based on a cooperative network for multi-UAV », *in*: *IEEE Communications Magazine* 56.*4* (2018), pp. 82–89, DOI: 10.1109/MCOM.2018.1700422.

[Gui] Guinness, *Most remote controlled multirotors/drones airborne simultaneously*, Web page, Accessed: 2023-07-17, URL: https://www.guinnessworldrecords.com/world-records/373319-most%C2%A0unmanned-aerial%C2%A0vehicles%C2%A0uavs-airborne%C2%A0simultaneously%C2%A05-kg-or-less.

[Ham+21] Mohand Hamadouche et al., « Comparison of value iteration, policy iteration and Q-Learning for solving decision-making problems », *in*: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2021, pp. 101–110, DOI: 10.1109/icuas51884.2021.9476691.

[Ham+23] Mohand Hamadouche et al., « Online reward adaptation for MDP-based distributed missions », *in*: *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2023, pp. 1059–1066, DOI: 10.1109/ICUAS57906.2023.10156131.

[Han+19] Xiao Han et al., « Multi-UAV Automatic Dynamic Obstacle Avoidance with Experience-shared A2C », *in*: *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, IEEE, 2019, pp. 330–335, DOI: 10.1109/WiMOB.2019.8923344.

[Has+20] Muhammad Abul Hassan et al., « Unmanned aerial vehicles routing formation using fisheye state routing for flying ad-hoc networks », *in*: *the 4th international conference on future networks and distributed systems (ICFNDS)*, 2020, pp. 1–7, DOI: 10.1145/3440749.3442600.

[HDB20] Mohand Hamadouche, Catherine Dezan, and Kalinka RLJC Branco, « Reward Tuning for self-adaptive Policy in MDP based Distributed Decision-Making to ensure a Safe Mission Planning », *in*: *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, IEEE, 2020, pp. 78–85, DOI: 10.1109/DSN-W50199.2020.00025.

[Hig+16] Kate Highnam et al., « An uncrewed aerial vehicle attack scenario and trustworthy repair architecture », *in*: *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, IEEE, 2016, pp. 222–225, DOI: 10.1109/DSN-W.2016.63.

[Hir+18] Chabha Hireche et al., « Context/resource-aware mission planning based on bns and concurrent mdps for autonomous uavs », *in*: *Sensors* 18.*12* (2018), p. 4266, DOI: 10.3390/s18124266.

[Hoa+18] VT Hoang et al., « Angle-encoded swarm optimization for uav formation path planning », *in*: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 5239–5244, DOI: 10.1109/IROS.2018.8593930.

[Hon+20] Liang Hong et al., « Toward swarm coordination: Topology-aware inter-UAV routing optimization », *in*: *IEEE Transactions on Vehicular Technology* 69.*9* (2020), pp. 10177–10187, DOI: 10.1109/TVT.2020.300335.

[HSS20] Dawei He, Wei Sun, and Lei Shi, « The novel mobility models based on spiral line for aerial backbone networks », *in*: *IEEE Access* 8 (2020), pp. 11297–11314, DOI: 10.1109/ACCESS.2020.2965616.

[Inta] Intel, *Intel Drone Light Show Breaks Guinness World Records Title at Olympic Winter Games PyeongChang 2018*, Web page, Accessed: 2023-07-17, URL: https://newsroom.intel.de/news-releases/intel-drone-light-show-breaks-guinness-world-records-title-at-olympic-winter-games-pyeongchang-2018/.

[Intb] Intel, *Intel Drones Light Up Lady Gaga Performance During Pepsi Zero Sugar Super Bowl LI Halftime*, Web page, Accessed: 2023-07-17, URL: https://www.businesswire.com/news/home/20170205005037/en/Intel-Drones-Light-Up-Lady-Gaga-Performance-During-Pepsi-Zero-Sugar-Super-Bowl-LI-Halftime-Show.

[Jos+22] Abhishek Joshi et al., « Enclosing and monitoring of disaster area boundary using multi-UAV network », *in*: *Journal of Ambient Intelligence and Humanized Computing* (2022), pp. 1–19, DOI: 10.1007/s12652-022-03757-5.

[Jun+21] Soyi Jung et al., « Orchestrated scheduling and multi-agent deep reinforcement learning for cloud-assisted multi-UAV charging systems », *in*: *IEEE Transactions on Vehicular Technology* 70.*6* (2021), pp. 5362–5377, DOI: 10.1109/TVT.2021.3062418.

[KSG20] Parampreet Kaur, Ashima Singh, and Sukhpal Singh Gill, « RGIM: an integrated approach to improve QoS in AODV, DSR and DSDV routing protocols for FANETS using the chain mobility model », *in*: *The Computer Journal* 63.*10* (2020), pp. 1500–1512, DOI: 10.1093/comjnl/bxaa040.

[Lee+21] Hyojun Lee et al., « A Robot Operating System Framework for Secure UAV Communications », *in*: *Sensors* 21.*4* (2021), p. 1369, DOI: 10.3390/s21041369.

[Lee+22] Ju-Hyung Lee et al., « Integrating LEO satellites and multi-UAV reinforcement learning for hybrid FSO/RF non-terrestrial networks », *in*: *IEEE Transactions on Vehicular Technology* (2022), DOI: 10.1109/TVT.2022.3220696.

[Lin+19] Na Lin et al., « A 3D smooth random walk mobility model for FANETs », *in*: *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems*, IEEE, 2019, pp. 460–467, DOI: 10.1109/HPCC/SmartCity/DSS.2019.00075.

[Liu+19] Yao Liu et al., « Two-layer routing for high-voltage powerline inspection by cooperated ground vehicle and drone », *in*: *Energies* 12.*7* (2019), p. 1385, DOI: 10.3390/en12071385.

[LL18] Alexey V Leonov and George A Litvinov, « Considering AODV and OLSR routing protocols to traffic monitoring scenario in FANET formed by mini-UAVs », *in*: *2018 XIV International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*, IEEE, 2018, pp. 229–237, DOI: 10.1109/APEIE.2018.8545667.

[LTW+22] Chuan-Chi Lai, Ang-Hsun Tsai, Li-Chun Wang, et al., « Adaptive and Fair Deployment Approach to Balance Offload Traffic in Multi-UAV Cellular Networks », *in*: *IEEE Transactions on Vehicular Technology* (2022), DOI: 10.1109/TVT.2022.3221557.

[LWW18] Qianyu Lin, Xiaoli Wang, and YuTong Wang, « Cooperative formation and obstacle avoidance algorithm for multi-uav system in 3d environment », *in*: *2018 37th Chinese Control Conference (CCC)*, IEEE, 2018, pp. 6943–6948, DOI: 10.23919/ChiCC.2018.8483113.

[Mad+19] Ioannis Mademlis et al., « High-level multiple-UAV cinematography tools for covering outdoor events », *in*: *IEEE Transactions on Broadcasting* 65.*3* (2019), pp. 627–635, DOI: 10.1109/TBC.2019.2892585.

[Mal+21] Evangelos Maltezos et al., « The INUS Platform: A Modular Solution for Object Detection and Tracking from UAVs and Terrestrial Surveillance Assets », *in*: *Computation* 9.*2* (2021), p. 12, DOI: 10.3390/computation9020012.

[Maz+15] Ivan Maza et al., « Classification of multi-UAV architectures », *in*: *Handbook of unmanned aerial vehicles* (2015), pp. 953–975, DOI: 10.1007/978-90-481-9707-1_119.

[Mit97] Tom M Mitchell, *Machine learning*, 1997, URL: https://www.cin.ufpe.br/~cavmj/Machine%20-%20Learning%20-%20Tom%20Mitchell.pdf.

[MU49] Nicholas Metropolis and Stanislaw Ulam, « The monte carlo method », *in*: *Journal of the American statistical association* 44.*247* (1949), pp. 335–341, DOI: 10.1080/01621459.1949.10483310.

[NAL21] Haque Nawaz, Husnain Mansoor Ali, and Asif Ali Laghari, « UAV communication networks issues: a review », *in*: *Archives of Computational Methods in Engineering* 28 (2021), pp. 1349–1369, DOI: 10.1007/s11831-020-09418-0.

[Nas17] Vladimir Nasteski, « An overview of the supervised machine learning methods », *in*: *Horizons. b* 4 (2017), pp. 51–62, DOI: 10.20544/HORIZONS.B.04.1.17.P05.

[Nie+16] Jianqiang Nie et al., « Decentralized cooperative lane-changing decision-making for connected autonomous vehicles », *in*: *IEEE Access* 4 (2016), pp. 9413–9420, DOI: 10.1109/ACCESS.2017.2649567.

[OK15] H. Y. Ong and M. J. Kochenderfer, « Short-term conflict resolution for unmanned aircraft traffic management », *in*: *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, 2015, 5A4-1-5A4–13, DOI: 10.1109/DASC.2015.7311424.

[Pat+07] Naïm Patrick et al., *Réseaux bayésiens*, Eyrolles, 2007, URL: https://www.editions-eyrolles.com/Livre/9782212119725/reseaux-bayesiens.

[Pea88] Judea Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan kaufmann, 1988, DOI: https://doi.org/10.1016/C2009-0-27609-4.

[PL06] Ian C. Price and Gary B. Lamont, « GA Directed Self-Organized Search and Attack UAV Swarms », *in*: *Proceedings of the 2006 Winter Simulation Conference*, 2006, pp. 1307–1315, DOI: 10.1109/WSC.2006.323229.

[PPB19] Rayner M Pires, Alex Sandro Roschildt Pinto, and Kalinka Regina Lucas Jaquie Castelo Branco, « The broadcast storm problem in FANETs and the dynamic neighborhood-based algorithm as a countermeasure », *in*: *IEEE Access* 7 (2019), pp. 59737–59757, DOI: 10.1109/ACCESS.2019.2915561.

[Put14] Martin L Puterman, *Markov decision processes: discrete stochastic dynamic programming*, John Wiley & Sons, 2014, URL: https://www.google.fr/books/edition/Markov_Decision_Processes/VvBjBAAAQBAJ?hl=fr&gbpv=1&dq=arkov+decision+processes:+discrete+stochastic+dynamic+programming.

[Rad+20] Panagiotis Radoglou-Grammatikis et al., « A compilation of UAV applications for precision agriculture », *in*: *Computer Networks* 172 (2020), p. 107148, DOI: 10.1016/j.comnet.2020.107148.

[Raj+20] Gunasekaran Raja et al., « Dynamic polygon generation for flexible pattern formation in large-scale uav swarm networks », *in*: *2020 IEEE Globecom Workshops (GC Wkshps*, IEEE, 2020, pp. 1–6, DOI: 10.1109/GCWkshps50303.2020.9367501.

[RCB15] Juan Jesús Roldán, Jaime del Cerro, and Antonio Barrientos, « A proposal of methodology for multi-UAV mission modeling », *in*: *2015 23rd Mediterranean Conference on Control and Automation (MED)*, IEEE, 2015, pp. 1–7, DOI: 10.1109/MED.2015.7158721.

[Res22] Ágoston Restás, « Drone Applications Fighting COVID-19 Pandemic—Towards Good Practices », *in*: *Drones* 6.*1* (2022), p. 15, DOI: 10.3390/drones6010015.

[RSF13] E. Rohmer, S. P. N. Singh, and M. Freese, « V-REP: A versatile and scalable robot simulation framework », *in*: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1321–1326, DOI: 10.1109/IROS.2013.6696520.

[SB98] R. S. Sutton and A. Barto, « Reinforcement learning: an introduction », *in*: *IEEE Transactions on Neural Networks* 9 (5 1998), pp. 1054–1054, DOI: 10.1109/tnn.1998.712192.

[SBS20] Georgy Skorobogatov, Cristina Barrado, and Esther Salamıé, « Multiple UAV systems: a survey », *in*: *Unmanned Systems* 8.*02* (2020), pp. 149–169, DOI: 10.1142/S2301385020500090.

[SC17] Patrik Schmuck and Margarita Chli, « Multi-uav collaborative monocular slam », *in*: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 3863–3870, DOI: 10.1109/ICRA.2017.7989445.

[Sch+03] Henning Schulzrinne et al., *RTP: A transport protocol for real-time applications*, tech. rep., 2003, DOI: 10.17487/rfc3550.

[Sch+15] Johann Schumann et al., « Towards real-time, on-board, hardware-supported sensor and software health management for unmanned aerial systems », *in*: *International Journal of Prognostics and Health Management* 6.*1* (2015), DOI: 10.36001/ijphm.2015.v6i1.2243.

[Sha+19] Manmohan Sharma et al., « A comprehensive study of performance parameters for MANET, VANET and FANET », *in*: *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, IEEE, 2019, pp. 0643–0646, DOI: 10.1109/IEMCON.2019.8936159.

[She+18] Minghao Shen et al., « Heuristics based cooperative planning for highway on-ramp merge », *in*: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 1266–1272, DOI: 10.1109/ITSC.2018.8569341.

[Sin+19] Gaurav Singal et al., « UAVs reliable transmission for multicast protocols in FANETs », *in*: *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, IEEE, 2019, pp. 130–135, DOI: 10.1109/IOTSMS48152.2019.8939221.

[SKO94] Alan Stuart, Maurice G. Kendall, and J. K. Ord, *Kendall's Advanced Theory of Statistics: Volume I – Distribution Theory*, Hodder Arnold, 1994, URL: https://search.worldcat.org/fr/title/266788295.

[Spy+21] Yannis Spyridis et al., « Modelling and simulation of a new cooperative algorithm for UAV swarm coordination in mobile RF target tracking », *in*: *Simulation Modelling Practice and Theory* 107 (2021), p. 102232, DOI: 10.1016/j.simpat.2020.102232.

[SRZ15] Yasir Saleem, Mubashir Husain Rehmani, and Sherali Zeadally, « Integration of cognitive radio technology with unmanned aerial vehicles: issues, opportunities, and future research challenges », *in*: *Journal of Network and Computer Applications* 50 (2015), pp. 15–31, DOI: 10.1016/j.jnca.2014.12.002.

[SSS14] PB Sujit, Srikanth Saripalli, and Joao Borges Sousa, « Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless », *in*: *IEEE Control Systems Magazine* 34.*1* (2014), pp. 42–59, DOI: 10.1109/MCS.2013.2287568.

[SV14] Kuldeep Singh and Anil Kumar Verma, « Applying OLSR routing in FANETs », *in*: *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, IEEE, 2014, pp. 1212–1215, DOI: 10.1109/ICACCCT.2014.7019290.

[SV15] Kuldeep Singh and Anil Kumar Verma, « Experimental analysis of AODV, DSDV and OLSR routing protocol for flying adhoc networks (FANETs) », *in*: *2015 IEEE international conference on electrical, computer and communication technologies (ICECCT)*, IEEE, 2015, pp. 1–4, DOI: 10.1109/IEMCON.2019.8936159.

[SW11] Claude Sammut and Geoffrey I Webb, *Encyclopedia of machine learning*, Springer Science & Business Media, 2011, DOI: 10.1007/978-1-4899-7687-1.

[Tan+20] Xiaopeng Tan et al., « Performance analysis of routing protocols for UAV communication networks », *in*: *IEEE access* 8 (2020), pp. 92212–92224, DOI: 10.1109/ACCESS.2020.2995040.

[Tes95] Gerald Tesauro, « Temporal difference learning and TD-Gammon », *in*: *Communications of the ACM* 38.*3* (1995), pp. 58–68, DOI: 10.1145/203330.203343.

[VR04] Patrick Vincent and Izhak Rubin, « A framework and analysis for cooperative search using UAV swarms », *in*: *Proceedings of the 2004 ACM symposium on Applied computing*, 2004, pp. 79–86, DOI: 10.1145/967900.967919.

[VV15] Kimon P Valavanis and George J Vachtsevanos, *Handbook of unmanned aerial vehicles*, vol. 1, Springer, 2015, DOI: 10.1007/978-90-481-9707-1.

[Wan+18] Weiye Wang et al., « Autonomous Conflict Resolution Method for multi-UAVs Based on Preorder Flight Information », *in*: *Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence*, 2018, pp. 1–6, DOI: 10.1145/3302425.3302455.

[Wan+20] Bin Wang et al., « Current technologies and challenges of applying fuel cell hybrid propulsion systems in unmanned aerial vehicles », *in*: *Progress in Aerospace Sciences* 116 (2020), p. 100620, DOI: 10.1016/j.paerosci.2020.100620.

[WD92] Christopher JCH Watkins and Peter Dayan, « Q-learning », *in*: *Machine learning* 8.*3-4* (1992), pp. 279–292, DOI: 10.1007/BF00992698.

[Yan+18] Tingting Yang et al., « Multi Agents to Search and Rescue Based on Group Intelligent Algorithm and Edge Computing », *in*: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, IEEE, 2018, pp. 389–394, DOI: 10.1109/Cybermatics_2018.2018.00092.

[Ye10] Y Ye, *The Simplex and Policy Iteration Methods are Strongly Polynomial for the Markov Decision Problem with a Fixed Discount Rate, Technical paper*, 2010, DOI: 10.1287/moor.1110.0516.

[YL19] Hua Yang and Zhiyong Liu, « An optimization routing protocol for FANETs », *in*: *EURASIP Journal on Wireless Communications and Networking* 2019.*1* (2019), pp. 1–8, DOI: 10.1186/s13638-019-1442-0.

[Yog21] Sumendra Yogarayan, « Wireless ad hoc network of manet, vanet, fanet and sanet: A review », *in*: *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 13.*4* (2021), pp. 13–18, URL: https://jtec.utem.edu.my/jtec/article/view/6119.

[YWW20] Hongfei Yao, Hongjian Wang, and Ying Wang, « UUV autonomous decision-making method based on dynamic influence diagram », *in*: *Complexity* 2020 (2020), pp. 1–14, DOI: 10.1155/2020/8565106.

[Zer+17] Sara Zermani et al., « Embedded context aware diagnosis for a UAV SoC platform », *in*: *Microprocessors and Microsystems* 51 (2017), pp. 185–197, DOI: 10.1016/j.micpro.2017.04.013.

COLLEGE MATHSTIC
DOCTORAL BRETAGNE
BRETAGNE OCEANE

Université de Bretagne Occidentale

Titre : Prise de décision distribuée dans les systèmes multi-UAV : exploration des méthodes, réglage des récompenses et adaptation du mode de fonctionnement

Mot clés : Planification de mission, Prise de décision, Processus de décision de Markov, Auto-adaptation, Réseaux Bayésiens, Véhicules autonomes.

Résumé : Les véhicules aériens sans pilote (UAV) prospèrent dans des environnements difficiles, améliorant la qualité des missions, la productivité et la sécurité. Opérer dans des contextes imprévisibles nécessite une prise de décision indépendante en temps réel pour une gestion efficace des missions.
Ce document se concentre sur les missions collaboratives multi-UAV, couvrant : 1) la sélection d'une méthode de planification de mission basée sur des critères spécifiques, 2) l'auto-adaptation des politiques grâce à l'ajustement des récompenses, et 3) l'adaptation du mode opératoire basée sur un réseau bayésien dans un système collaboratif multi-UAV. Pour la planification de mission, un cadre de processus de décision de Markov (MDP) est mis en avant, avec une étude comparative de trois méthodes fondamentales de résolution des MDP pour aider à la sélection de méthode en fonction des critères du problème. L'utilisation de moteurs de décision basés sur les MDP au niveau du système multi-UAV peut entraîner des conflits, et le mécanisme de gestion des conflits proposé, basé sur l'adaptation des récompenses, aborde la détection et la résolution des conflits parmi les UAVs. Enfin, un mécanisme de commutation de mode local et hybride basé sur un réseau bayésien est exploré pour permettre l'auto-adaptation des politiques, piloté par une surveillance continue de la qualité de Service (QoS) pour optimiser la planification des missions et atteindre une autonomie totale des drones.

Title: Distributed Decision-Making in Multi-UAV Systems: Exploring Methods, Rewards Tuning, and Operating Mode Adaptation

Keywords: Mission Planning, Decision-Making, Markov Decision Process, Self-adaptation, Bayesian Networks, Autonomous Vehicles.

Abstract: Unmanned Aerial Vehicle (UAV)s thrive in challenging environments, improving mission quality, productivity, and safety. Operating in unpredictable settings requires independent real-time decision-making for effective mission management.
This document focuses on multi-UAV collaborative missions, covering: 1) selecting a mission planning method based on specific criteria, 2) self-adapting policies through reward tuning, and 3) Bayesian Network (BN)-based operating mode adaptation in a collaborative multi-UAV system. For mission planning, a Markov Decision Process (MDP) framework is highlighted, with a comparative study of three fundamental MDP resolution methods to help in method selection based on problem criteria. The use of MDP-based decision engines at the multi-UAV system level may lead to conflicts, and the proposed conflict management mechanism, based on rewards adaptation, addresses conflict detection and resolution among member UAVs. Finally, a local and hybrid mode switching mechanism based on BN is explored to enable policy self-adaptation, driven by continuous Quality of Service (QoS) monitoring for optimizing mission planning and achieving total drone autonomy.