Architectures délibératives pour la robotique autonome - des algorithmes au logiciel embarqué

Charles Lesire

ONERA
THE FRENCH AEROSPACE LAB

THÈSE

En vue de l'obtention de l'

HABILITATION A DIRIGER DES RECHERCHES

Délivrée par : *l'Institut National Polytechnique de Toulouse (INP Toulouse)*

Présentée et soutenue le *04/04/2019* **par :**

Charles LESIRE

Architectures délibératives pour la robotique autonome
– des algorithmes au logiciel embarqué –

JURY

Jean-Philippe BABAU	Professeur Univ. de Bretagne Occidentale	Rapporteur
Noury BOURAQADI	Professeur Institut Mines-Telecom de Douai	Examinateur
Didier CRESTANI	Professeur Univ. de Montpellier	Rapporteur
Félix INGRAND	Chargé de Recherche LAAS/CNRS	Membre invité
Simon LACROIX	Directeur de Recherche LAAS/CNRS	Examinateur
Olivier SIMONIN	Professeur INSA de Lyon	Rapporteur
Catherine TESSIER	Maître de Recherche ONERA, Centre de Toulouse	Examinateur

Table des matières

Tableaux et figures

Algorithmes et listings

Chapitre 1

Curriculum Vitæ

1.1 Etat civil

Nom d'usage	Lesire-Cabaniols
Nom patronymique	Lesire
Prénoms	Charles, Louis, Sylvain
Date de naissance	31 août 1980
Nationalité	Française
Situation familiale	Marié, 3 enfants

1.2 Parcours Universitaire

depuis 2008	Chercheur affilié à l'équipe d'accueil ONERA-ISAE CSDV (Commande des Systèmes et Dynamique du Vol), Ecole Doctorale Systèmes (ED 309)
2007	Qualification aux fonctions de Maître de Conférences, sections 27 et 61
2006	Doctorat de l'ISAE-Supaero, spécialité Systèmes Décisionnels, mention Très Honorable
2003	Diplôme d'Etudes Approfondies, spécialité Systèmes Informatiques, Université de Toulouse, mention Bien
2003	Diplôme d'Ingénieur de l'Ecole Nationale de l'Aviation Civile, spécialité Informatique et Trafic Aérien
1998	Baccalauréat Scientifique, spécialité Mathématiques

1.3 Parcours Professionnel

03/2017 -	Ingénieur de Recherche, ONERA/DTIS, Toulouse
09/2007 - 02/2017	Ingénieur de Recherche, ONERA/DCSD, Toulouse
01/2007 - 08/2007	Chercheur Post-Doctorant, LAAS-CNRS, Toulouse "Validation d'architectures logicielles pour la robotique"
10/2003 - 09/2006	Doctorant, ONERA/DCSD, Toulouse "Estimation de modes dans les systèmes hybrides"
02/2003 - 08/2003	Stagiaire DEA, ONERA/DCSD, Toulouse "Suivi de l'activité de pilotage"
07/2002 - 08/2002	Stagiaire, Eurocopter, Marignane "Développement d'une base de données de gestion des risques"

1.4 Encadrements

1.4.1 Encadrement de Masters

Etudiant	Sujet	Période	Co-encadrants
Amiot, David	Supervision d'architecture pour la robotique autonome	03/2018 08/2018	David Doose (ONERA/DTIS)
de Gouvello, Alix	Planification et supervision d'équipes multi-robots	03/2018 08/2018	Christophe Grand (ONERA/DTIS)
Iglesis, Enzo	Fault tolerance for autonomous robots	04/2018 09/2018	Jérémie Guiochet (LAAS/CNRS)
Piedade, Sébastien	Langage de spécification de missions autonomes pour un opérateur	03/2017 08/2017	Magali Barbier Guillaume Infantes (ONERA/DTIS)
Guyon, Gautier	Décision distribuée pour des missions multi-robots	03/2017 08/2017	Christophe Grand (ONERA/DTIS)
Heckler, Marine	Exploration active d'environnement inconnu pour la robotique autonome	03/2015 08/2015	Guillaume Infantes (ONERA/DTIS)
T'Hooft, Jorrit	Architecture de planification pour la navigation autonome d'un robot	03/2014 08/2014	Florent Teichteil (ONERA/DTIS)
Gabrielides, Odysseas	Ordonnancement de tâches multi-agents avec contraintes de communication	03/2013 08/2013	Cédric Pralet (ONERA/DTIS)

Etudiant	Sujet	Période	Co-encadrants
Bigot, Damien	Planification et supervision de mission pour une équipe de véhicules autonomes	03/2011 08/2011	Magali Barbier (ONERA/DTIS)
Gateau, Thibault	Supervision de mission pour un ensemble de drones	03/2009 08/2009	Magali Barbier (ONERA/DTIS)
Seban, Pablo	Vers un modèle dynamique de l'attention visuelle d'un opérateur	03/2017 08/2017	Frédéric Dehais (ISAE)

1.4.2 Encadrement de Doctorants

Nom du docteur	**Thibault Gateau**
Sujet	Supervision de mission pour un ensemble de véhicules autonomes hétérogènes
Date de soutenance	11/12/2012
Taux d'encadrement	50 %
Liste des travaux co-publiés	Thibault Gateau, Gaetan Séverac, Charles Lesire, Magali Barbier et Eric Bensana (2012a), « Knowledge base for planning, execution and plan repair », *in* : *ICAPS Workshop on Planning and Execution (PlanEx)*, Altibaia, Brazil Thibault Gateau, Charles Lesire et Magali Barbier (2012c), « Robust strategies for multi-robot team collaboration under uncertain communications », *in* : *International Symposium on Distributed Autonomous Robotic Systems (DARS), Poster session*, Baltimore, MD, USA Thibault Gateau, Charles Lesire et Magali Barbier (2012a), « HiDDeN, une architecture décisionnelle distribuée pour la coopération de véhicules individuellement autonomes », *in* : *Congrès Francophone de Reconnaissance des Formes et d'Ingelligence Artificielle (RFIA)*, Lyon, France Thibault Gateau, Charles Lesire et Magali Barbier (2013b), « HiDDeN : Cooperative plan execution for heterogeneous robots in dynamic environments », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*, Tokyo, Japan

	Charles Lesire, Guillaume Infantes, Thibault Gateau et Magali Barbier (2016), « A distributed architecture for supervision of autonomous multi-robot missions - Application to air-sea scenarios », *in* : *Autonomous Robots* 40.7, p. 1343–1362
Situation actuelle	Ingénieur, ISAE/Supaero, Toulouse

Nom du docteur	**Nicolas Gobillot**
Sujet	Validation d'architectures logicielles pour la robotique
Date de soutenance	29/04/2016
Taux d'encadrement	50 %
Liste des travaux co-publiés	Nicolas Gobillot, Charles Lesire et David Doose (2013a), « A Component-Based Navigation-Guidance-Control Architecture for Mobile Robots », *in* : *ICRA Workshop on Software Development and Integration for Robotics (SDIR)*, Karlsruhe, Germany Nicolas Gobillot, Charles Lesire et David Doose (2014a), « A Modeling Framework for Software Architecture Specification and Validation », *in* : *International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR 2014)*, Bergamo, Italy Nicolas Gobillot, David Doose, Charles Lesire et Luca Santinelli (2015), « Periodic state-machine aware real-time analysis », *in* : *IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2015)*, Luxembourg, Luxembourg Nicolas Gobillot, Fabrice Guet, David Doose, Christophe Grand, Charles Lesire et Luca Santinelli (2016), « Measurement-Based Real-Time Analysis of Robotic Software Architectures », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea Nicolas Gobillot, Charles Lesire et David Doose (2019), « A design and analysis methodology for component-based real-time architectures of autonomous systems », *in* : *Journal of Intelligent Robots and Systems (JINT)*
Situation actuelle	Ingénieur de Recherche, IFREMER, Toulon

Nom du docteur	**Patrick Bechon**
Sujet	Planification multi-robots pour des missions de surveillance
Date de soutenance	26/05/2016
Taux d'encadrement	50 %
Liste des travaux co-publiés	Patrick Bechon, Magali Barbier, Guillaume Infantes, Charles Lesire et Vincent Vidal (2014b), « HiPOP : Hierarchical Partial-Order Planning », *in* : *European Starting AI Researchers Symposium (STAIRS 2014)*, Prague, Czech Republic Patrick Bechon, Magali Barbier, Charles Lesire, Guillaume Infantes et Vincent Vidal (2015), « Using hybrid planning for plan reparation », *in* : *European Conference on Mobile Robots (ECMR 2015)*, Lincoln, United Kingdom Patrick Bechon, Magali Barbier, Christophe Grand, Simon Lacroix, Charles Lesire et Cédric Pralet (2018), « Integrating planning and execution for a team of heterogeneous robots with time and communication constraints », *in* : *IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia Patrick Bechon, Charles Lesire et Magali Barbier (2018), « Hybrid planning and distributed iterative repair for multi-robot missions with communication losses », *in* : *Autonomous Robots*, Under Review
Situation actuelle	Ingénieur de l'Armement, DGA, Bagneux

Nom du doctorant	**Erwan Lecarpantier**
Sujet	Reinforcement Learning for Non-Stationnary Markov Decision Processes
Début de thèse	décembre 2016
Taux d'encadrement	10 %
Liste des travaux co-publiés	Erwan Lecarpentier, Guillaume Infantes, Charles Lesire et Emmanuel Rachelson (2018b), « Open Loop Execution of Tree-Search Algorithms », *in* : *International Joint Conference on Artificial Intelligence (IJCAI 2018)*, Stockholm, Sweden

Nom du doctorant	**Sébastien Piedade**
Sujet	Synthèse de plans conditionnels pour la décision dans l'incertain
Début de thèse	octobre 2017
Taux d'encadrement	70 %
Liste des travaux co-publiés	Sébastien Piedade, Charles Lesire et Guillaume Infantes (2018), « Conditional plans synthesis for decision under uncertainty applied to satellite acquisitions », *in* : *IJCAI Workshop on Planning and Learning*, Stockholm, Sweden

1.4.3 Encadrement de Post-Doctorants

Nom du chercheur	**Khakim Habibi**
Sujet	Décision distribuée pour des missions multi-robots
Contrat	ONERA
Période	03/2018 - 03/2019
Co-encadrement	Christophe Grand, Cédric Pralet (ONERA/DTIS)
Liste des travaux co-publiés	Muhammad Khakim Habibi, Christophe Grand, Charles Lesire et Cédric Pralet (2019), « Solving Methods for Multi-Robot Missions Planning with Energy Capacity Consideration », *in* : *International Conference on Robotics and Automation (ICRA)*, Montreal, Canada

1.5 Responsabilités collectives

Depuis janvier 2018, je suis responsable de l'unité "Systèmes Embarqués, Autonomes et Sûrs" de l'ONERA/DTIS. Cette unité compte (fin 2018) 11 chercheurs permanents, 6 doctorants, 2 post-doctorants. Le travail de responsable d'unité consiste en :

— la représentation de l'unité devant la direction de l'ONERA,

— l'organisation de la répartition annuelle du travail,

— l'évaluation annuelle des chercheurs permanents.

1.6 Activités d'expertises

1.6.1 Expertise de projets

— Expertise de projets ANR JCJC

1.6.2 Participation à des jurys de thèse

— Membre du jury de thèse de Arnaud Degroote, Université de Toulouse, LAAS- CNRS, octobre 2012

— Membre du comité de suivi de thèse de Quentin Gaudel, Université de Toulouse, LAAS-CNRS, février 2015

— Membre du comité de suivi de thèse de Andrea De Maio, Université de Toulouse, LAAS-CNRS, septembre 2018

— Membre du comité de suivi de thèse de Valentin Bouziat, Université de Toulouse, ONERA, octobre 2018

1.6.3 Expertise d'articles

— Relectures d'articles de journaux (IEEE Trans. on Robotics, Journal of Intelligent Robots and Systems, Journal of Software Engineering for Robotics)

— Relectures d'articles de conférences (ICRA, IROS, ECAI, IJCAI, AAMAS)

1.7 Diffusion Scientifique

1.7.1 Participation au comité scientifique de manisfestations

— Membre du comité de programme de SHARC 2017

1.7.2 Vulgarisation scientifique

— Patrick Bechon, Magali Barbier et Charles Lesire (2016), « Comment faire coopérer des robots autonomes », *in* : *La Jaune et la Rouge* 718

— Magali Barbier, Martial Sanfourche, Yoko Watanabe, Charles Lesire et Philippe Bidaud (2015), « Des robots qui coopèrent entre-eux ! », *in* : *Magazine des Ingénieurs de l'Armement* 105

— Charles Lesire (2009b), « Donner de l'autonomie aux drones », *in* : *Le Transpondeur* 106

1.7.3 Conférencier invité

— Conférencier invité à la *CESEC Summer School on Critical Embedded Systems*, 2013, Toulouse, France, sur le sujet *Developping robotic applications using Component-Based Software Engineering (CBSE) and Software Architecture in the Loop (SAIL) simulations* ;

— Conférencier invité au *Workshop on Fielded Multi-Robot Systems*, organisé dans le cadre de l'*International Conference on Robotics and Automation (ICRA)*, Stockholm, Suède, 2016, sur le sujet *Deploying teams of heterogeneous robots for surveillance missions*.

1.7.4 Prix et distinctions

— Prix du meilleur papier à la conférence JFSMA 2015, pour l'article Guillaume Casanova, Charles Lesire et Cédric Pralet (2015), « Gestion des réseaux temporels simples multi-agents dynamiques », *in* : *Journée Francophones sur les Systèmes Multi-Agents (JFSMA)*, Rennes, France ;

— Deuxième place au concours de robotique autonome de la conférence IROS 2014.

1.7.5 Communautés savantes

— Membre du comité technique de l'IEEE Robotics and Automation Society (RAS) *TC-SOFT (Technical Committee on Software Engineering for Robotics and Automation)* depuis 2013 ;

— Membre du comité technique de l'IEEE Robotics and Automation Society (RAS) *TC-MRS (Technical Committee on Multi-Robot Systems)* depuis 2015 ;

— Participation régulière aux ateliers du GT4 du Groupe de Recherche en Robotique :

  — Ananda Basu, Matthieu Gallien, Charles Lesire, Thanh-Hung Nguyen, Saddek Bensalem, Félix Ingrand et Joseph Sifakis (2007), « Incremental Component-Based Construction and Verification of a Robotic System », *in* : *National Conference on Control Architectures of Robots (CAR)*, Paris, France

  — Magali Barbier, Hung Cao, Simon Lacroix, Charles Lesire, Florent Teichteil-Königsbuch et Catherine Tessier (2009), « Decision issues for multiple heterogeneous vehicles in uncertain environments », *in* : *National Conference on Control Architectures of Robots (CAR)*, Toulouse, France

  — Guillaume Infantes, Charles Lesire, Henry de Plinval et Florent Teichteil-Königsbuch (2009), « An Orocos-based decisional architecture for the Ressac missions », *in* : *National Conference on Control Architectures of Robots (CAR)*, Toulouse, France

— Thibault Gateau, Magali Barbier et Charles Lesire (2010), « Local Plan Execution and Repair in a Hierarchical Structure of Sub-Teams of Heterogeneous Autonomous Vehicles », *in* : *National Conference on Control Architectures of Robots (CAR)*, Douai, France

— Charles Lesire, David Doose et Hugues Cassé (2011a), « Validation of real-time properties of a robotic software architecture », *in* : *National Conference on Control Architectures of Robots (CAR)*, Grenoble, France

— Florent Teichteil-Königsbuch, Charles Lesire et Guillaume Infantes (2011b), « A generic framework for anytime execution-driven planning in robotics », *in* : *National Conference on Control Architectures of Robots (CAR)*, Grenoble, France

— Thibault Gateau, Gaetan Séverac, Charles Lesire, Magali Barbier et Eric Bensana (2012b), « Knowledge base for planning, execution and plan repair », *in* : *National Conference on Control Architectures of Robots (CAR)*, Nancy, France

— Nicolas Gobillot, Charles Lesire et David Doose (2013b), « A Component-Based Navigation-Guidance-Control Architecture for Mobile Robots », *in* : *National Conference on Control Architectures of Robots (CAR)*, Angers, France

— Magali Barbier, Charles Lesire et Simon Lacroix (2013), « Coopération autonome d'un drone aérien et d'un robot terrestre », *in* : *National Conference on Control Architectures of Robots (CAR)*, Angers, France

— Nicolas Gobillot, David Doose, Christophe Grand, Charles Lesire et Luca Santinelli (2015), « Real-time analysis of robotic software architectures », *in* : *National Conference on Control Architectures of Robots (CAR)*, Lyon, France

— Jorrit T'Hooft, Charles Lesire et Caroline Ponzoni Carvalho Chanel (2016b), « Proactive Planning and Execution Strategies with Multiple Hypotheses », *in* : *National Conference on Software and Hardware Architectures for Robots Control (SHARC)*, Brest, France

— David Doose, Christophe Grand et Charles Lesire (2017c), « MAUVE Runtime : a component-based middleware to reconfigure software architectures in real-time », *in* : *National Conference on Software and Hardware Architectures for Robots Control (SHARC)*, Toulouse, France

1.8 Enseignements

Entre 2008 et 2012, j'ai participé à plusieurs enseignements en tant que vacataire à l'ISAE/Supaero et à l'ENAC, sur les enseignements suivants :

— Applications robotiques autonomes (TP et BE)

— Introduction à l'Intelligence Artificielle (cours et TP – 20h)

— Optimisation (cours et TP – 6h)

— Planification (cours et TP – 4h)

— Systèmes à événements discrets (cours et TP – 12h)

Depuis 2013 je ne participe plus aux enseignements dans ces établissements, notamment du fait de leur réorganisation interne qui a mené à affecter ces enseignements à des enseignants-chercheurs de ces établissements.

Je donne par ailleurs des cours dans le cadre des formations scientifiques proposées par l'Ecole Doctorale Systèmes :

— Systèmes à événements discrets, systèmes hybrides (EDSYS, formation 2009, cours de 3h)

— Planification embarquée (EDSYS, formation 2013, cours de 3h)

— Architectures logicielles pour la robotique (EDSYS, formation 2013, cours de 3h)

— Architectures logicielles pour la robotique (EDSYS, formation 2017, cours de 4h)

1.9 Participation à des projets

1.9.1 Collaboration à des projets internationaux

ANR ANCHORS

Sujet	UAV-Assisted Ad Hoc Networks for Crisis Management
Période	2012-2015
Client	ANR (coopération franco-allemande)
Consortium	CEA, Airbus DS, LS Telecom, Groupe INTRA, ONERA + 8 partenaires allemands
Rôle	Responsable technique ONERA et responsable du Work-Package "Demonstrations"
Contribution	— Développement d'algorithmes de décision pour le déploiement de réseaux de communication lors de tâches d'observation multi-robots, — Organisation des démonstrations du consortium français, organisée à Esperce (31) en octobre 2015.

FP7 Aeroceptor

Sujet	UAV-Based Innovative Means for Land and Sea Non-Cooperative Vehicles Stop
Période	2013-2015
Client	Union Européenne (FP7)
Consortium	INTA, ISDEFE, GMV, ONERA, PIAP, Univ. Bologna, Austria Inst. of Tech., Israel Aerospace Indus., TOFAS, SFU, Ministerio del Interior (Spain), RTS, Lacroix, Isreali Police, Zabala
Contribution	Intégration des algorithmes développés dans le projet sur les drones ONERA pour les démonstrations finales du projet.

H2020 CPSE-Labs

Sujet	Cyber-Physical Systems Engineering Labs
Période	2015-2017
Client	Union Européenne (H2020)
Consortium	fortiss, OFFIS, KTH, LAAS/CNRS, ONERA, Newcastle Univ., INDRA, Tech. Univ. Madrid
Contribution	Application de la chaîne MAUVE sur des cas d'études fournis par des industriels robotiques (Sterela, Naïo).

1.9.2 Projets industriels ou institutionnels

PEA ACTION

Sujet	Etude de coopération de multivéhicules hétérogènes
Période	2007-2015
Client	DGA (PEA)
Partenaires	LAAS/CNRS
Contribution	— Développement d'algorithmes de planification et de supervision multi-robots, — Participation à la réalisation des expérimentations, — Participation au développement du simulateur.

Etudes dans le domaine maritime

Sujet	Chasse aux mines par véhicule autonome sous-marin
Période	2007-2009
Client	Thales Underwater Systems
Contribution	Développement de l'architecture logicielle basée sur ProCoSA (notamment des interfaces avec le simulateur et l'architecture TUS), et mise en œuvre des tests unitaires et d'intégration.

Sujet	Supervision d'une mission de protection par véhicule autonome de surface
Période	2017-2018
Client	Naval Group
Contribution	Développement de l'architecture logicielle basée sur MAUVE, et définition d'un superviseur de mission utilisant une modélisation par réseaux de Petri.

Etudes CNES

Sujet	Architecture logicielle embarquée
Période	2010-2012
Client	Programme commun ONERA/CNES Robotique d'Exploration Planétaire
Rôle	Responsable technique ONERA sur la tranche de l'année 2011-2012
Contribution	Tests de faisabilité pour le développement d'une architecture logicielle basée composants sur des calculateurs embarquables.

Sujet	Superviseur Bord pour le Spatial
Période	2017-2018
Client	R&T CNES
Contribution	Concept d'architecture modulaire pour la planification d'activités d'observation par un satellite.

1.9.3 Coordination de projets

Sujet	Reconfiguration of multi-Robot teams in Dynamic environments and valiDation
Période	2014-2018
Client	ONERA (Projet de Recherche interne)
Rôle	Chef de projet (6 personnes impliquées dans le projet, 2eqtp/an)
Contribution	— Développement de la chaîne MAUVE, qui permet de spécifier des architectures robotiques à base de composants. Cette chaîne comporte des outils de vérification de propriétés temps-réel, de synthèse d'observateurs temporels, et de procédés de reconfiguration de l'architecture logicielle ; — Réalisation des démonstrations robotiques mettant en avant les capacités de reconfiguration.

1.10 Production Scientifique

1.10.1 Résume de la Production Scientifique

Brevets	1
Articles de revues internationales	4
Articles de revues nationales	1
Communications avec actes dans un congrès international	33
Communications avec actes dans un congrès national	6

1.10.2 Brevets

Dehais, Frédéric, Charles Lesire, Catherine Tessier et Laure Christophe (2009), « Method and device for detecting piloting conflicts between the crew and the autopilot of an aircraft », PCT/FR2009/000648.

1.10.3 Articles de revues internationales

Gobillot, Nicolas, Charles Lesire et David Doose (2019), « A design and analysis methodology for component-based real-time architectures of autonomous systems », *in* : *Journal of Intelligent Robots and Systems (JINT)*.

Ponzoni Carvalho Chanel, Caroline, Alexandre Albore, Jorrit T'Hooft, Charles Lesire et Florent Teichteil-Königsbuch (2019), « AMPLE : an anytime planning and execution framework for dynamic and uncertain problems in robotics », *in* : *Autonomous Robots* 43.*1*, p. 37–62.

Doose, David, Christophe Grand et Charles Lesire (2017a), « MAUVE Runtime : a component-based middleware to reconfigure software architectures in real-time », *in* : *Journal on Software Engineering for Robotics (JOSER)* 8.*1*, p. 128–140.

Lesire, Charles, Guillaume Infantes, Thibault Gateau et Magali Barbier (2016), « A distributed architecture for supervision of autonomous multi-robot missions - Application to air-sea scenarios », *in* : *Autonomous Robots* 40.*7*, p. 1343–1362.

1.10.4 Articles de revues nationales

Casanova, Guillaume, Charles Lesire et Cédric Pralet (2016), « Gestion des réseaux temporels simples multi-agents dynamiques », *in* : *Revue d'Intelligence Artificielle* 30.*1-2*, p. 11–33.

1.10.5 Communications internationales avec actes

Habibi, Muhammad Khakim, Christophe Grand, Charles Lesire et Cédric Pralet (2019), « Solving Methods for Multi-Robot Missions Planning with Energy Capacity Consideration », *in* : *International Conference on Robotics and Automation (ICRA)*, Montreal, Canada.

Lesire, Charles, Stéphanie Roussel, David Doose et Christophe Grand (2019), « Synthesis of Real-Time Observers from Past-Time Linear Temporal Logic and Timed Specification », *in* : *IEEE International Conference on Robotics and Automation (ICRA)*, Montreal, Canada.

Bechon, Patrick, Magali Barbier, Christophe Grand, Simon Lacroix, Charles Lesire et Cédric Pralet (2018), « Integrating planning and execution for a team of heterogeneous robots with time and communication constraints », *in* : *IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia.

Lecarpentier, Erwan, Guillaume Infantes, Charles Lesire et Emmanuel Rachelson (2018b), « Open Loop Execution of Tree-Search Algorithms », *in* : *International Joint Conference on Artificial Intelligence (IJCAI 2018)*, Stockholm, Sweden.

Lesire, Charles et Franck Pommereau (2018), « ASPiC : an Acting system based on Skill Petri net Composition », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018)*, Madrid, Spain.

Santinelli, Luca, David Doose, Guy Durrieu, Frédéric Boniol, Charles Lesire-Cabaniols et Christophe Grand (2018), « Schedulability analysis for mixed critical

cyber physical systems », *in* : *IEEE Industrial Cyber-Physical Systems (ICPS 2018)*, Saint Petersburg, Russia.

Doose, David, Christophe Grand et Charles Lesire (2017b), « MAUVE Runtime : a component-based middleware to reconfigure software architectures in real-time », *in* : *IEEE International Conference on Robotic Computing (IRC)*, Taichung, Taiwan.

Casanova, Guillaume, Cédric Pralet, Charles Lesire et Thierry Vidal (2016a), « Solving Dynamic Controllability Problem of Multi-Agent Plans with Uncertainty using Mixed Integer Linear Programming », *in* : *European Conference on Artificial Intelligence (ECAI)*, The Hague, Netherlands.

Gobillot, Nicolas, Fabrice Guet, David Doose, Christophe Grand, Charles Lesire et Luca Santinelli (2016), « Measurement-Based Real-Time Analysis of Robotic Software Architectures », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea.

T'Hooft, Jorrit, Charles Lesire et Caroline Ponzoni Carvalho Chanel (2016a), « Online Proactive Planning with Multiple Hypotheses », *in* : *European Starting AI Researchers Symposium (STAIRS 2016)*, The Hague, Netherlands.

Watanabe, Yoko, Augustin Manecy, Alexandre Amiez, Charles Lesire et Christophe Grand (2016), « Non-cooperative ground vehicle tracking and interception by multi-RPA collaboration », *in* : *International Council of the Aeronautical Sciences (ICAS)*, Daejeon, Korea.

Bechon, Patrick, Magali Barbier, Charles Lesire, Guillaume Infantes et Vincent Vidal (2015), « Using hybrid planning for plan reparation », *in* : *European Conference on Mobile Robots (ECMR 2015)*, Lincoln, United Kingdom.

Casanova, Guillaume, Cédric Pralet et Charles Lesire (2015), « Managing Dynamic Multi-Agent Simple Temporal Network », *in* : *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2015)*, Istanbul, Turkey.

Gobillot, Nicolas, David Doose, Charles Lesire et Luca Santinelli (2015), « Periodic state-machine aware real-time analysis », *in* : *IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2015)*, Luxembourg, Luxembourg.

Bechon, Patrick, Magali Barbier, Guillaume Infantes, Charles Lesire et Vincent Vidal (2014b), « HiPOP : Hierarchical Partial-Order Planning », *in* : *European Starting AI Researchers Symposium (STAIRS 2014)*, Prague, Czech Republic.

Gobillot, Nicolas, Charles Lesire et David Doose (2014a), « A Modeling Framework for Software Architecture Specification and Validation », *in* : *International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR 2014)*, Bergamo, Italy.

Ponzoni Carvalho Chanel, Caroline, Charles Lesire et Florent Teichteil-Königsbuch (2014), « A Robotic Execution Framework for Online Probabilistic (Re)Planning »,

in : *International Conference on Automated Planning and Scheduling (ICAPS 2014)*, Portsmouth, New Hampshire, USA.

Pralet, Cédric et Charles Lesire (2014a), « Deployment of Mobile Wireless Sensor Networks for Crisis Management : A Constraint-Based Local Search Approach », *in* : *International Conference on Principles and Practice of Constraint Programming (CP 2004)*, Lyon, France.

Gateau, Thibault, Charles Lesire et Magali Barbier (2013b), « HiDDeN : Cooperative plan execution for heterogeneous robots in dynamic environments », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*, Tokyo, Japan.

Ponzoni Carvalho Chanel, Caroline, Florent Teichteil-Königsbuch et Charles Lesire (2013), « Multi-Target Detection and Recognition by UAVs Using Online POMDPs », *in* : *AAAI Conference on Artificial Intelligence (AAAI 2013)*, Bellevue, Washington, USA.

Echeverria, Gilberto, Séverin Lemaignan, Arnaud Degroote, Simon Lacroix, Michael Karg, Pierrick Koch, Charles Lesire et Serge Stinckwich (2012a), « Simulating Complex Robotic Scenarios with MORSE », *in* : *International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, Tsukuba, Japan.

Ponzoni Carvalho Chanel, Caroline, Florent Teichteil-Königsbuch et Charles Lesire (2012c), « POMDP-based online target detection and recognition for autonomous UAVs », *in* : *European Conference on Artificial Intelligence (ECAI)*, Montpellier, France.

Teichteil-Königsbuch, Florent, Charles Lesire et Guillaume Infantes (2011a), « A generic framework for anytime execution-driven planning in robotics », *in* : *IEEE International Conference on Robotics and Automation (ICRA 2011)*, Shangai, China.

Lesire, Charles (2010a), « An Iterative A* Algorithm for Planning of Airport Ground Movements », *in* : *European Conference on Artificial Intelligence (ECAI)*, Lisbon, Portugal.

Watanabe, Yoko, Charles Lesire, Alain Piquereau, Patrick Fabiani, Martial Sanfourche et Guy Le Besnerais (2010a), « The ONERA ReSSAC Unmanned Autonomous Helicopter : Visual Air-to-Ground Target Tracking in an Urban Environment », *in* : *Annual Forum of the American Helicopter Society (AHS)*, Pheonix, AZ, USA.

Lesire, Charles (2009a), « Automatic planning of ground traffic », *in* : *AIAA Aerospace Sciences Meeting (ASM)*, Orlando, FL, USA.

Basu, Ananda, Matthieu Gallien, Charles Lesire, Thanh-Hung Nguyen, Saddek Bensalem, Félix Ingrand et Joseph Sifakis (2008b), « Incremental Component-Based Construction and Verification of a Robotic System », *in* : *European Conference on Artificial Intelligence (ECAI 2008)*, Patras, Greece.

Abdeddaïm, Yasmina, Eugene Asarin, Matthieu Gallien, Félix Ingrand, Charles Lesire et Mihaela Sighireanu (2007), « Planning Robust Temporal Plans : A Comparison Between CBTP and TGA Approaches », *in* : *International Conference on Automated Planning and Scheduling (ICAPS)*, Providence, RI, USA.

Lesire, Charles et Catherine Tessier (2007), « Particle Petri Net-based estimation in hybrid systems to detect inconsistencies », *in* : *IFAC Workshop on Dependable Control of Discrete Systems (DCDS)*, Paris, France.

Bonnet-Torrés, Olivier, Patrice Domenech, Charles Lesire et Catherine Tessier (2006), « Exhost-PIPE : PIPE Extended for Two Classes of Monitoring Petri Nets », *in* : *International Conference on Application and Theory of Petri Nets and Concurrency (ATPN)*, Turku, Finland.

Lesire, Charles et Catherine Tessier (2006a), « A Hybrid Model for Situation Monitoring and Conflict Prediction in Human Supervised Autonomous Systems », *in* : *AAAI Spring Symposium "To Boldly Go Where No Human-Robot Team Has Gone Before"*, Stanford, CA, USA.

Lesire, Charles et Catherine Tessier (2006b), « Estimation and Conflict Detection in Human Controlled Systems », *in* : *International Workshop on Hybrid Systems : Computation and Control (HSCC)*, Santa Barbara, CA, USA.

Watanabe, Yoko, Charles Lesire, Alain Piquereau, Patrick Fabiani et Martial Sanfourche (2006), « System development and flight experiment of vision-based simultaneous navigation and tracking », *in* : *AIAA InfoTech@Aerospace*, Atlanta, GA, USA.

Dehais, Frédéric, Alexandre Goudou, Charles Lesire et Catherine Tessier (2005), « Towards an anticipatory agent to help pilots », *in* : *AAAI Fall Symposium "From Reactive to Anticipatory Cognitive Embodied Systems"*, Arlington, VI, USA.

Lesire, Charles et Catherine Tessier (2005a), « Particle Petri Nets for Aircraft Procedure Monitoring Under Uncertainty », *in* : *International Conference on Application and Theory of Petri Nets and Concurrency (ATPN)*, Miami, FL, USA.

1.10.6 Communications nationales avec actes

Lecarpentier, Erwan, Guillaume Infantes, Charles Lesire et Emmanuel Rachelson (2018a), « Open Loop Execution of Tree-Search Algorithms », *in* : *Journées Francophone Planification, Décision et Apprentissage pour la conduite des systèmes (JFPDA 2018)*, Nancy, France.

Casanova, Guillaume, Cédric Pralet, Charles Lesire et Thierry Vidal (2016b), « Synthèse de plans d'exécution multi-agents robustes aux incertitudes et à l'absence de communications », *in* : *Journée Francophones sur les Systèmes Multi-Agents (JFSMA)*, Rouen, France.

Casanova, Guillaume, Charles Lesire et Cédric Pralet (2015), « Gestion des réseaux temporels simples multi-agents dynamiques », *in* : *Journée Francophones sur les Systèmes Multi-Agents (JFSMA)*, Rennes, France.

Gateau, Thibault, Charles Lesire et Magali Barbier (2012a), « HiDDeN, une architecture décisionnelle distribuée pour la coopération de véhicules individuellement autonomes », *in* : *Congrès Francophone de Reconnaissance des Formes et d'Ingelligence Artificielle (RFIA)*, Lyon, France.

Seban, Pablo, Frédéric Dehais et Charles Lesire (2008), « Vers un modèle dynamique de l'attention visuelle d'un opérateur humain », *in* : *Congrès Francophone de Reconnaissance des Formes et d'Intelligence Artificielle (RFIA)*, Amiens, France.

Dehais, Frédéric, Charles Lesire, Catherine Tessier et Laurent Chaudron (2004), « Conflits et contre-mesures dans l'activité de pilotage », *in* : *Congrès Francophone de Reconnaissance des Formes et d'Intelligence Artificielle (RFIA)*, Toulouse, France.

1.10.7 Autres communications

Bechon, Patrick, Charles Lesire et Magali Barbier (2018), « Hybrid planning and distributed iterative repair for multi-robot missions with communication losses », *in* : *Autonomous Robots*, Under Review.

Piedade, Sébastien, Charles Lesire et Guillaume Infantes (2018), « Conditional plans synthesis for decision under uncertainty applied to satellite acquisitions », *in* : *IJCAI Workshop on Planning and Learning*, Stockholm, Sweden.

Doose, David, Christophe Grand et Charles Lesire (2017c), « MAUVE Runtime : a component-based middleware to reconfigure software architectures in real-time », *in* : *National Conference on Software and Hardware Architectures for Robots Control (SHARC)*, Toulouse, France.

Bechon, Patrick, Magali Barbier et Charles Lesire (2016), « Comment faire coopérer des robots autonomes », *in* : *La Jaune et la Rouge* 718.

Gobillot, Nicolas, David Doose, Charles Lesire et Luca Santinelli (2016), « Periodic state-machine aware real-time analysis », *in* : *Journée Formalisation des Activités Concurrentes (FAC)*, Toulouse, France.

Sarkis, Mireille, François Montaigne, Hervé Durin, Charles Lesire et Pascal Izydorczyk (2016), « ANCHORS – UAV-Assisted Ad Hoc Networks for Crisis Management and Hostile Environment Sensing », *in* : *Workshop Interdisciplinaire sur la Sécurité Globale (WISG)*, Troyes, France.

T'Hooft, Jorrit, Charles Lesire et Caroline Ponzoni Carvalho Chanel (2016b), « Proactive Planning and Execution Strategies with Multiple Hypotheses », *in* : *National Conference on Software and Hardware Architectures for Robots Control (SHARC)*, Brest, France.

Barbier, Magali, Martial Sanfourche, Yoko Watanabe, Charles Lesire et Philippe Bidaud (2015), « Des robots qui coopèrent entre-eux ! », *in* : *Magazine des Ingénieurs de l'Armement* 105.

Gobillot, Nicolas, David Doose, Christophe Grand, Charles Lesire et Luca Santinelli (2015), « Real-time analysis of robotic software architectures », *in* : *National Conference on Control Architectures of Robots (CAR)*, Lyon, France.

Gobillot, Nicolas, Alessandra Melani, Fabrice Guet, Luca Santinelli, Eric Noulard, David Doose, Charles Lesire et Jérôme Morio (2015), « Building System Awareness : Cache Characterization through Probabilities », *in* : *Journée Formalisation des Activités Concurrentes (FAC)*, Toulouse, France.

Lesire, Charles, Patrick Bechon et Guillaume Casanova (2015), « Multi-robot Planning and Execution for Surveillance Missions », *in* : *ONERA-DLR Aerospace Symposium (ODAS)*, Toulouse, France.

Bechon, Patrick, Magali Barbier, Guillaume Infantes, Charles Lesire et Vincent Vidal (2014c), « HiPOP : Hierarchical Partial-Order Planning », *in* : *Journées Francophones de Planification, Décision et Apprentissage (JFPDA)*, Leuven, Belgium.

Infantes, Guillaume, Charles Lesire et Cédric Pralet (2014b), « Multi-Robot Planning and Execution for an Exploration Mission : a Case Study », *in* : *ICAPS Workshop on Planning and Robotics (PlanRob)*, Portsmouth, NH, USA.

Barbier, Magali, Charles Lesire et Simon Lacroix (2013), « Coopération autonome d'un drone aérien et d'un robot terrestre », *in* : *National Conference on Control Architectures of Robots (CAR)*, Angers, France.

Gobillot, Nicolas, Charles Lesire et David Doose (2013a), « A Component-Based Navigation-Guidance-Control Architecture for Mobile Robots », *in* : *ICRA Workshop on Software Development and Integration for Robotics (SDIR)*, Karlsruhe, Germany.

Gobillot, Nicolas, Charles Lesire et David Doose (2013b), « A Component-Based Navigation-Guidance-Control Architecture for Mobile Robots », *in* : *National Conference on Control Architectures of Robots (CAR)*, Angers, France.

Gateau, Thibault, Charles Lesire et Magali Barbier (2012c), « Robust strategies for multi-robot team collaboration under uncertain communications », *in* : *International Symposium on Distributed Autonomous Robotic Systems (DARS), Poster session*, Baltimore, MD, USA.

Gateau, Thibault, Gaetan Séverac, Charles Lesire, Magali Barbier et Eric Bensana (2012a), « Knowledge base for planning, execution and plan repair », *in* : *ICAPS Workshop on Planning and Execution (PlanEx)*, Altibaia, Brazil.

Gateau, Thibault, Gaetan Séverac, Charles Lesire, Magali Barbier et Eric Bensana (2012b), « Knowledge base for planning, execution and plan repair », *in* : *National Conference on Control Architectures of Robots (CAR)*, Nancy, France.

Lesire, Charles, David Doose et Hugues Cassé (2012b), « Mauve : a Component-based Modeling Framework for Real-Time Analysis of Robotic Applications », *in* : *ICRA Workshop on Software Development and Integration for Robotics (SDIR)*, Saint-Paul, Minnesota, USA.

Ponzoni Carvalho Chanel, Caroline, Florent Teichteil-Königsbuch et Charles Lesire (2012a), « Détection et reconnaissance de cibles en ligne pour des UAV autonomes avec un modèle de type POMDP », *in* : *Journées Francophones sur la Planification, la Décision et l'Apprentissage (JFPDA)*, Nancy, France.

Ponzoni Carvalho Chanel, Caroline, Florent Teichteil-Königsbuch et Charles Lesire (2012b), « Planning for perception and perceiving for decision : POMDP-like online target detection and recognition for autonomous UAVs », *in* : *International Conference on Automated Planning and Scheduling (ICAPS) Applications Workshop (SPARK)*, Sao Paulo, Brazil.

Verfaillie, Gérard, Cédric Pralet, Vincent Vidal, Florent Teichteil-Königsbuch, Guillaume Infantes et Charles Lesire (2012), « Synthesis of plans or policies for controlling dynamic systems », *in* : *AerospaceLab* 4.

Lesire, Charles, David Doose et Hugues Cassé (2011a), « Validation of real-time properties of a robotic software architecture », *in* : *National Conference on Control Architectures of Robots (CAR)*, Grenoble, France.

Teichteil-Königsbuch, Florent, Charles Lesire et Guillaume Infantes (2011b), « A generic framework for anytime execution-driven planning in robotics », *in* : *National Conference on Control Architectures of Robots (CAR)*, Grenoble, France.

Gateau, Thibault, Magali Barbier et Charles Lesire (2010), « Local Plan Execution and Repair in a Hierarchical Structure of Sub-Teams of Heterogeneous Autonomous Vehicles », *in* : *National Conference on Control Architectures of Robots (CAR)*, Douai, France.

Lesire, Charles (2010b), « Iterative planning of airport ground movements », *in* : *International Conference on Research in Air Transportation (ICRAT)*, Budapest, Hungary.

Barbier, Magali, Hung Cao, Simon Lacroix, Charles Lesire, Florent Teichteil-Königsbuch et Catherine Tessier (2009), « Decision issues for multiple heterogeneous vehicles in uncertain environments », *in* : *National Conference on Control Architectures of Robots (CAR)*, Toulouse, France.

Infantes, Guillaume, Charles Lesire, Henry de Plinval et Florent Teichteil-Königsbuch (2009), « An Orocos-based decisional architecture for the Ressac missions », *in* : *National Conference on Control Architectures of Robots (CAR)*, Toulouse, France.

Lesire, Charles (2009b), « Donner de l'autonomie aux drones », *in* : *Le Transpondeur* 106.

Andrieu, Quentin, Frédéric Dehais, Alexandre Izaute, Charles Lesire et Catherine Tessier (2008), « Towards a dynamic computational model of visual attention », *in* : *Conference on Humans Operating Unmanned Systems (HUMOUS)*, Brest, France.

Basu, Ananda, Matthieu Gallien, Charles Lesire, Thanh-Hung Nguyen, Saddek Bensalem, Félix Ingrand et Joseph Sifakis (2008c), « Incremental Component-Based Construction and Verification of a Robotic System », *in* : *International Conference on Intelligent Robots and Systems (IROS) Workshop on Current Software Frameworks in Cognitive Robotics*, Nice, France.

Mercier, Stéphane, Frédéric Dehais, Charles Lesire et Catherine Tessier (2008), « Resources as basic concepts for authority sharing », *in* : *Conference on Humans Operating Unmanned Systems (HUMOUS)*, Brest, France.

Mercier, Stéphane, Catherine Tessier, Frédéric Dehais et Charles Lesire (2008), « Basic concepts for human/robot shared authority », *in* : *International Conference on Human Centered Processes (HCP)*, Delft, Netherlands.

Basu, Ananda, Matthieu Gallien, Charles Lesire, Thanh-Hung Nguyen, Saddek Bensalem, Félix Ingrand et Joseph Sifakis (2007), « Incremental Component-Based Construction and Verification of a Robotic System », *in* : *National Conference on Control Architectures of Robots (CAR)*, Paris, France.

Bonnet-Torrés, Olivier, Aziz El Bouzidi, Charles Lesire et Catherine Tessier (2006), « Exhost-PIPE : Playing Plan Petri Nets and Particle Petri Nets », *in* : *International Conference on Application and Theory of Petri Nets and other models of concurrenty (ATPN) Tools session*, Turku, Finland.

Dehais, Frédéric, Charles Lesire, Patrick Fabiani et Catherine Tessier (2006), « Situation monitoring and conflict detection in human-centric autonomous systems », *in* : *Moving Autonomy Forward Conference*, Grantham, UK.

Lesire, Charles (2006), « Estimation numérico-symbolique pour le suivi d'activités hybrides », thèse de doct., Toulouse, France : ISAE-Supaero.

Lesire, Charles et Catherine Tessier (2005b), « Réseaux de Petri particulaires pour l'estimation symbolico-numérique », *in* : *Journées Formalisation des Activités Concurrentes (FAC)*, Toulouse, France.

Lesire, Charles (2004a), « A numerical/symbolic estimator for activity tracking : a preliminary report », *in* : *International Conference on Knowledge Representation and Reasoning (KR) Doctoral Consortium*, Whistler, BC, Canada.

Lesire, Charles (2004b), « Suivi de l'activité de pilotage par prédiction et recalage », *in* : *Journal Information, Savoirs, Décision, Médiations (IDSM)* 13.

Lesire, Charles (2004c), « Suivi de l'activité de pilotage par prédiction et recalage », Thèse de master, Toulouse, France : Ecole Nationale de l'Aviation Civile.

Lesire, Charles (2003), « Suivi de l'activité de pilotage par prédiction et recalage », *in* : *Manifestion des Jeunes Chercheurs en Sciences et Techniques de l'Information et de la Communication (MaJeCSTIC)*, Marseille, France.

Chapitre 2

Introduction

Pour les pionniers, *robotique autonome* est un pléonasme. Les travaux fondateurs de la robotique ont dès le départ l'ambition de concevoir des robots intelligents, autonomes : des robots qui puissent adapter leur comportement à leur évolution et à celle de leur environnement. Shakey, le premier robot autonome (Nilsson 1969), était ainsi capable à la fois d'adapter ses déplacements pour éviter des obstacles, et, pour atteindre des buts fournis par un utilisateur, de planifier des séquences d'actions au moyen des premiers planificateurs STRIPS.

Depuis, l'Intelligence Artificielle a pris une part prépondérante dans les travaux en robotique, de la modélisation de problèmes complexes, mêlant contrôlabilité partielle, observabilité partielle, contraintes temporelles, etc., à l'élaboration d'algorithmes efficaces pour résoudre ces problèmes. La robotique a aujourd'hui intégré des approches pour résoudre des problèmes partiellement observables, avec observations et actions probabilistes, comme dans le cadre des POMDP (Kaelbling, Littman et Cassandra 1998), qui ont par exemple permis au robot Xavier d'être en service pendant plusieurs années, parcourant quelques centaines de kilomètres, au sein de l'université de Carnegie Mellon (Simmons et al. 2000). Les travaux sur la planification de tâches ont permis de conférer de l'autonomie aux robots et satellites d'exploration spatiale en planifiant, et replanifiant en cas de nouvelles informations ou de défaillances, des tâches de déplacement ou d'observation pour réaliser des missions scientifiques (Chien et al. 1999). Et finalement, plus récemment, les méthodes d'apprentissage profond ont permis de franchir une étape importante dans les capacités des systèmes autonomes à interpréter les données acquises sur leur environnement (Goodfellow, Bengio et Courville 2016).

Ces différentes fonctions ou différents algorithmes contribuent à élever l'intelligence des systèmes robotiques. Mais concevoir un système robotique autonome nécessite d'utiliser plusieurs fonctionnalités de manière structurée et cohérente, au sein d'une *architecture délibérative*. Ingrand et Ghallab (2017) ont classé les différentes méthodes et algorithmes nécessaires à la prise de décision au sein d'un robot autonome selon différentes fonctions (figure 2.1) résumées ci-dessous :

— *planifier* : repose sur des algorithmes de recherche pour calculer un plan réalisable pour atteindre un objectif en fonction d'un état courant et de modèles ;

— *agir* : raffine les actions planifiées jusqu'à des commandes adaptées au contexte courant, et réagit à des événements ;

— *observer* : détecte et reconnaît des caractéristiques pouvant définir l'état courant du système, et de son environnement ;

— *surveiller* : compare une prédiction et l'observation des états du monde pour en déduire des écarts, diagnostiquer leur cause, et éventuellement les réparer ;

— *raisonner sur les buts* : évalue la pertinence des actions en cours au regard de l'évolution du robot et de son environnement ; en déduit si des buts doivent être mis à jour, suivis, ou abandonnés ;

— *apprendre* : permet au système d'acquérir ou d'adapter ses fonctionnalités à partir d'expériences sur la façon dont l'environnement réagit aux actions du robot.

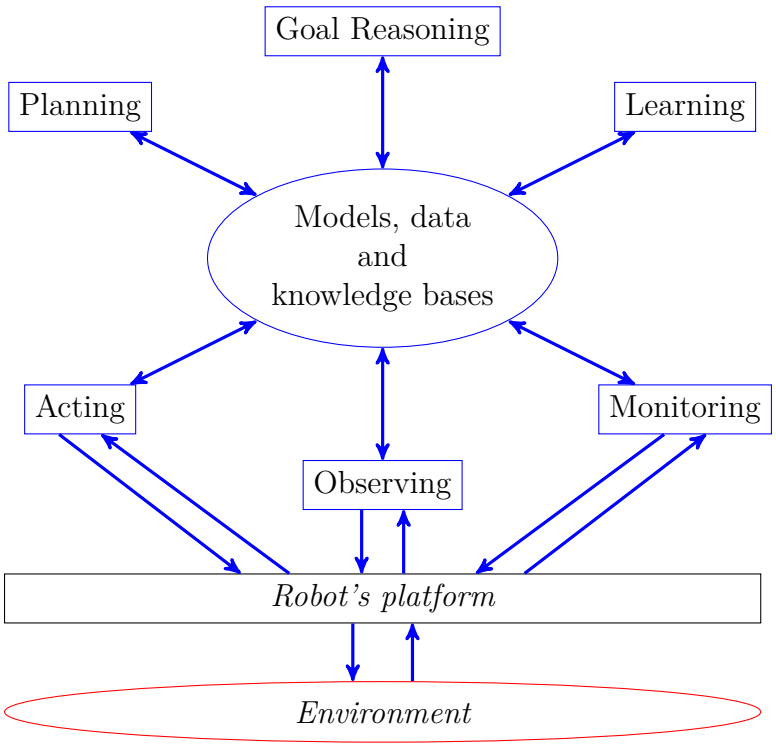


FIGURE 2.1 – Reproduction de la figure "Schematic view of deliberation functions", INGRAND et GHALLAB (2017)

Le constat dressé par INGRAND et GHALLAB (2017) est que de nombreux travaux ont porté sur le développement des fonctions d'apprentissage, de planification et de raisonnement sur les buts, notamment car ces fonctions correspondent aux travaux fondateurs de l'Intelligence Artificielle. Probablement que ces fonctions donnent également une image plus "formelle", contrairement aux autres fonctionnalités qui, de par leur lien avec la plate-forme robotique, peuvent avoir une connotation plus empirique ou expérimentale. INGRAND et GHALLAB (2017) argumentent ainsi pour des travaux plus fondamentaux centrés sur l'action, avec une attention particulière sur les liens entre des modèles descriptifs généralement utilisés pour raisonner ou planifier, et des modèles opérationnels, généralement au cœur des systèmes d'exécution, jusqu'aux langages de programmation

permettant d'accéder aux capteurs et actionneurs du système robotique.

Suivant le même constat, les travaux présentés dans ce mémoire se sont focalisés sur la mise au point des méthodes et outils pour la conception de systèmes robotiques autonomes. L'objectif de ces travaux est donc de contribuer au développement de robots autonomes en fournissant à leur développeur, programmeur ou concepteur, des approches méthodologiques, outillées, qui permettent d'intégrer de manière rigoureuse et pratique, des algorithmes d'Intelligence Artificielle dans des architectures délibératives. Pour cela, ces travaux ont un objectif certes de pertinence vis-à-vis des problématiques d'autonomie et de prise de décision, mais essaient également de prendre en compte des problématiques de sûreté de fonctionnement et de robustesse, qui sont deux critères indispensables pour franchir la barrière de l'utilisation de robots autonomes, autant dans le domaine industriel que public.

Ce mémoire présente trois contributions :

— Un cadre pour la planification et l'exécution en-ligne des problèmes combinatoires en garantissant la réactivité du système de décision ; ce cadre, appelé **AMPLE** pour *Anytime Meta-PLannEr*, permet notamment de raisonner sur différentes situations futures possibles du système robotique et de son environnement, par exemple en considérant les différents effets possibles des actions réalisées lorsqu'il utilise le cadre (PO)MDP (chapitre 3).

— Une chaîne outillée de développement d'architectures logicielles à base de composants, reposant sur un langage de conception spécifique, **MAUVE**, et sur des outils de génération de code, de déploiement, de recueil de traces et d'analyse du comportement temps-réel (chapitre 4).

— Une architecture délibérative pour une équipe de robots autonomes hétérogènes, mise en œuvre dans le cadre du projet **ACTION**, pour laquelle la robustesse à l'absence de communication permanente est importante ; cette architecture met en œuvre des mécanismes de planification et réparation de plan décentralisés, ainsi que des mécanismes d'adaptation temporelle du plan des différents robots (chapitre 5).

Ces différents travaux contribuent aux fonctionnalités nécessaires à l'autonomie, et décrites précédemment, selon le schéma de la figure 2.2.

Ces trois contributions participent à l'élaboration d'architectures délibératives pour la robotique autonome :

— Le cadre **AMPLE** est un cadre d'*exécution* de politiques, qui coordonne la résolution

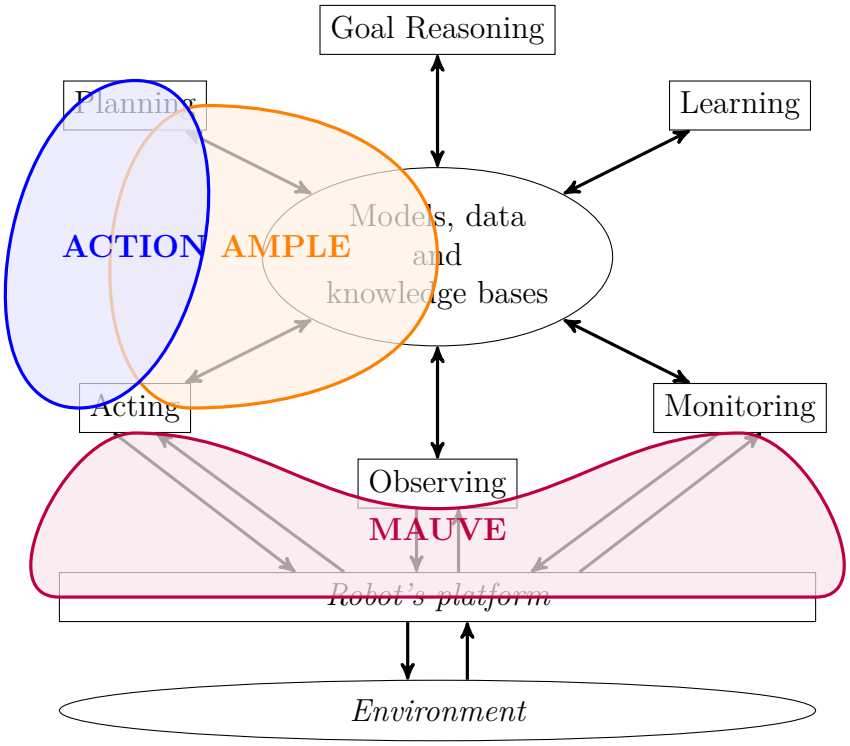


FIGURE 2.2 – Placement des contributions selon les différentes fonctions délibératives

de différents problèmes de *planification* en gérant les contraintes de réactivité, et l'incertitude sur l'évolution du monde ; il repose pour cela sur une *modélisation* commune des plans et des requêtes de planification.

— La chaîne **MAUVE** fournit des outils de modélisation de la *plate-forme robotique* et des différents composants et traitements associés ; elle permet donc la réalisation des fonctionnalités de commande et de perception du robot, et fournit un socle pour l'*interface* avec des fonctions délibératives.

— L'architecture **ACTION** intègre le développement d'un algorithme de *planification* hybride pour une équipe de robots hétérogènes, ainsi que des principes d'*exécution* distribuée qui permettent d'une part de propager les contraintes temporelles au sein d'un réseau temporel simple multi-agent (MaSTN, Multiagent Simple Temporal Network) de manière robuste aux communications intermittentes, et une logique de *réparation* tenant compte des communications et de *modèles* d'actions temporels et hiérarchiques.

Chacun des chapitres suivants décrit l'une de ces contributions, en la resituant par rapport à son contexte et à la littérature, au regard de la problématique de l'autonomie évoquée dans cette introduction. Chacune de ces contributions est également discutée,

notamment en vu d'en lister les limites et d'évoquer des pistes d'amélioration.

Le chapitre 6 restructure l'ensemble de ces pistes en élaborant des perspectives de recherche pour les architectures délibératives, selon deux axes principaux : 1. la fonction d'agissement et le lien entre l'exécution de plans et les modèles d'architectures logicielles, et 2. l'utilisation des modèles et d'algorithmes différents au sein d'architectures délibératives complexes.

Chapitre 3

AMPLE : un cadre *anytime* pour la planification et l'exécution

3.1 Contexte

3.1.1 Décision séquentielle dans l'incertain en robotique

Un des enjeux des algorithmes de planification et de prise de décision pour des systèmes robotiques est la gestion de l'incertitude. Ces incertitudes peuvent venir des effets même du système sur son environnement, de l'imperfection des capteurs ou des actionneurs, etc.

Ces problèmes sont généralement modélisés sous la forme de Processus Décisionnels de Markov (MDP ; Puterman 1994) lorsque l'état du système est observable, et que l'incertitude est modélisée uniquement sur les effets des actions, et de Processus Décisionnels de Markov Partiellement Observable (POMDP ; Kaelbling, Littman et Cassandra 1998) lorsque l'état du système est également incertain, alors représenté sous forme d'un état de croyance.

Un **PO**[†]MDP est défini par :

— des *états* $\mathcal{S}$, des *actions* $\mathcal{A}$, et des ***observations*** $\mathcal{O}$

— des *effets probabilistes* : $T(s, a, s') = \Pr(S_{t+1} = s' \mid S_t = s, A_t = a)$

— des ***observations probabilistes*** : $O(s', a, o) = \Pr(O_{t+1} = o \mid S_{t+1} = s', A_t = a)$

— des *récompenses* : $r(s, a)$

La fonction de transition T définit une distribution de probabilités sur les états d'arrivée possibles à l'étape $t+1$, en fonction de l'état de départ (s_t) et de l'action appliquée (a_t). La fonction d'observation O définit une distribution de probabilités sur les observations qui seront disponibles en fonction de l'état dans lequel le système se retrouve (s') et de l'action appliquée (a). Une représentation graphique de ce modèle est donnée sur la figure 3.1.

Cette modélisation a été notamment prisée en robotique car elle fournit un cadre pour raisonner sur la qualité de la localisation, ou le coût de navigation, basé sur des fonctions d'utilité (Carrillo et al. 2015 ; Makarenko et al. 2002).

†. les éléments en gras ne sont définis que dans le cadre partiellement observable

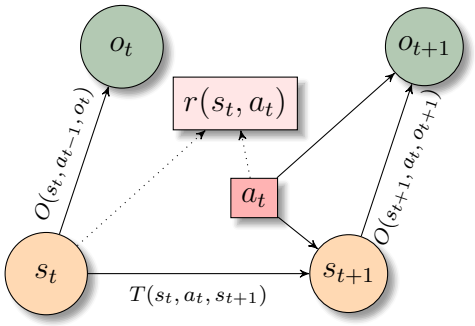


FIGURE 3.1 – Représentation graphique d'un modèle POMDP

Le problème à résoudre sur la base de ce modèle est alors de calculer une *politique* $\pi : \mathcal{S} \to \mathcal{A}$ qui optimise un critère, typiquement l'espérance des récompenses cumulées :

$$E\left[\sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t)\right], \qquad 0 < \gamma < 1, \qquad \pi(s_t) = a_t \tag{3.1}$$

La politique π associe ainsi à chaque état possible du système une action optimale à réaliser. La complexité de ce problème, dans le cadre MDP, est polynomiale en fonction du nombre d'états. La représentation la plus usuelle étant une représentation implicite, c'est-à-dire utilisant des variables d'états, la complexité devient exponentielle en fonction du nombre de variables d'état.

Afin de résoudre de manière pratique des problèmes complexes, de nombreux algorithmes calculent une politique partielle, définie uniquement sur un ensemble d'états *pertinents* (HANSEN et ZILBERSTEIN 2001 ; BONET et GEFFNER 2003 ; TEICHTEIL-KÖNIGSBUCH, KUTER et INFANTES 2010 ; PINEAU, GORDON et THRUN 2006 ; KURNIATI, HSU et LEE 2008 ; ROSS et CHAIB-DRAA 2007).

3.1.2 Planification *anytime* dans l'incertain

Les applications utilisant des (PO)MDP en robotique résolvent des problèmes très spécifiques, de navigation (ONG et al. 2010), de manipulation (HSIAO, KAELBLING et LOZANO-PEREZ 2007), ou d'identification d'objets (FARGES et al. 2018). Ces travaux résolvent le problème hors-ligne, en se basant sur les incertitudes présentes dans le modèle.

Cependant, pour être intégrés dans une architecture délibérative plus complexe, qui va planifier ou organiser la réalisation de tâches plus spécifiques pour résoudre une mission dans sa globalité, il est nécessaire de pouvoir résoudre ces problèmes en ligne. En effet, les problèmes de navigation peuvent dépendre des explorations ou cartographies réalisées, les tâches de manipulation ou d'identification, des objets détectés. Un point important dans l'intégration de ces méthodes au sein d'une architecture délibérative est donc leur *réactivité* par rapport aux évolutions de l'environnement ou des tâches à effectuer.

De plus, dans certaines applications critiques, cette réactivité doit être bornée et

garantie. Une des applications que l'on considère est l'atterrissage autonome d'urgence : lors d'une mission d'exploration, un drone aérien subit une panne qui l'oblige à atterrir en urgence. La zone dans laquelle il essaie d'atterrir était inconnue en début de mission, donc il est nécessaire de résoudre le problème en ligne. La représentation de ce problème est faite sous forme de MDP, où l'incertitude est représentée sur les chances de succès d'un atterrissage, qui dépendent du temps passé à explorer la zone. L'objectif est alors de maximiser l'espérance d'atterrir de manière sécurisée. L'énergie disponible suite à la panne étant limitée, la résolution du problème doit se faire en temps borné.

Les approches pour traiter cette question de la réactivité se sont orientées vers la définition d'algorithmes *anytime*, initialement dans le domaine de la planification déterministe (Korf 1990), dans lesquels un plan faisable mais non optimal est calculé rapidement, puis ensuite amélioré si le temps le permet. Ces approches *anytime* ont également été appliquées pour la résolution de (PO)MDP (Barto, Bradtke et Singh 1995 ; Pineau, Gordon et Thrun 2006 ; Keller et Eyerich 2012 ; Ross et al. 2008). Des travaux récents en planification déterministe ont cherché à prendre en compte le temps de calcul des plans au sein même de la recherche de solution (Burns, Ruml et Do 2013 ; Lin et al. 2015), mais n'apportent pas une solution satisfaisante, ni garantie, au problème de réactivité.

Le problème auquel on s'intéresse est un problème de *planification en temps contraint*. Il est nécessaire de fournir une solution à l'instant t, c'est-à-dire une action a_t à exécuter dans le cas d'une politique. Les approches *anytime* n'apportent pas une solution satisfaisante à ce problème car :

1. elles ne peuvent pas contrôler le temps passé à trouver une première solution ;
2. elles reposent sur le calcul de politiques partielles, et ne peuvent donc garantir que l'état dans lequel se trouve le système a été exploré ;
3. l'analyse du temps de calcul sur des problèmes types ne peut pas *garantir* une réactivité lors de la résolution en ligne.

3.2 Contribution

Nous avons proposé de résoudre le problème de planification en temps contraint par une approche *architecturale*, à travers le cadre AMPLE (Anytime Meta-PLannEr), qui peut être vue comme une approche *anytime* pour la planification *et* l'exécution. En ce sens, il peut être vu comme une architecture de *planification continue en-ligne intégrée avec le processus d'exécution* (Nau, Ghallab et Traverso 2015).

3.2.1 Architecture AMPLE

Afin de traiter le problème de planification et exécution en temps contraint, l'architecture AMPLE proposée est composée (figure 3.2) d'un module responsable de la résolution de problèmes de planification (*planning thread*), et d'un module responsable de spécifier des problèmes, de demander la résolution de ces problèmes, et de récupérer les actions à exécuter (*execution thread*).

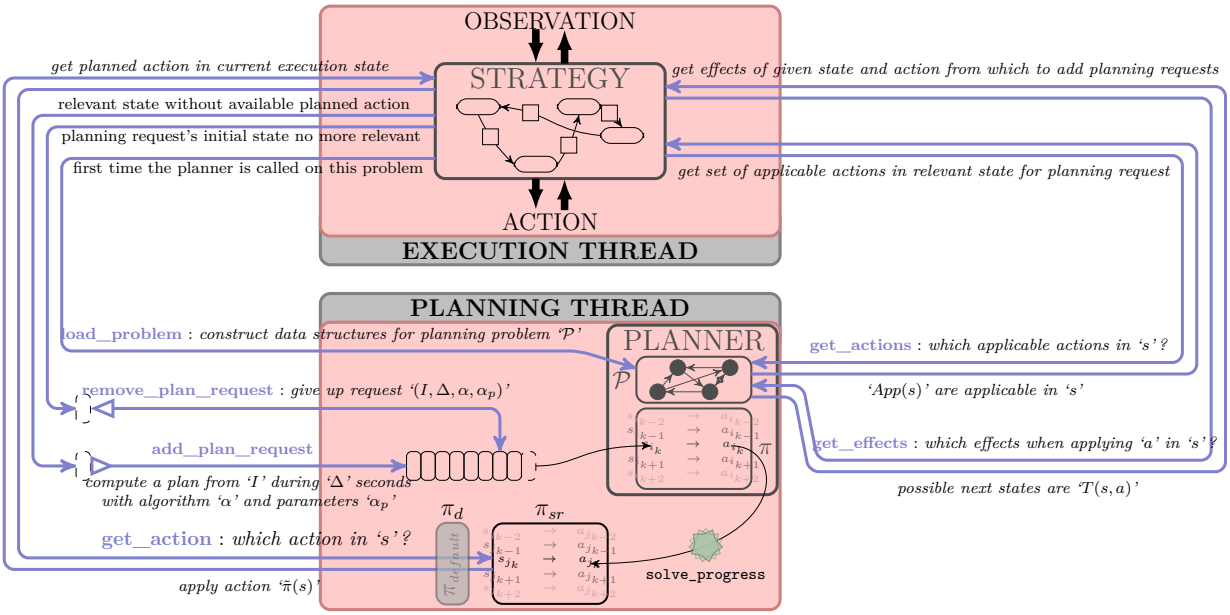


FIGURE 3.2 – Architecture AMPLE, tirée de (PONZONI CARVALHO CHANEL et al. 2019).

Ces deux modules interagissent à travers un certain nombre de méthodes qui permettent d'ajouter/retirer des requêtes de planification, d'obtenir l'action planifiée pour un état donné, ou d'interroger le problème de planification.

3.2.2 Module de planification

Le module de planification repose sur trois principes fondamentaux qui assurent la planification en temps-contraint.

Interface générique

Le module de planification encapsule un algorithme de planification, et fournit ainsi une interface générique à différents algorithmes de résolution de (PO)MDP. Cette encapsulation générique est basée sur le découpage de l'algorithme en fonctions élémentaires. La boucle principale de l'algorithme est alors gérée par le module de planification, ce qui permettent de maîtriser le temps passé à résoudre un problème. Dans (PONZONI CARVALHO CHANEL et al. 2019), nous avons analysé les approches usuelles de résolution de MDP et POMDP, et nous avons montré que la plupart des algorithmes peuvent se décomposer selon ce schéma.

Requêtes de planification

Le module de planification gère une file de requêtes de planification. Chaque requête est définie par l'état initial du problème à résoudre, les paramètres éventuels de l'algorithme de résolution, et une contrainte de temps.

Politique locale

Le module de planification maintient une politique locale, qui est mise à jour à chaque boucle d'exécution de l'algorithme de planification. Cette politique locale est initialisée par une *politique par défaut*, qui permet de garantir la réactivité, même lorsque aucune action n'a été optimisée pour l'état courant par l'algorithme de résolution. Cette politique par défaut doit être définie en fonction de l'application.

Applications

Ce module de planification a été décrit dans (Teichteil-Königsbuch, Lesire et Infantes 2011). Dans ce travail, les requêtes de planification étaient gérées par un module ad-hoc. Les principes du module de planification ont été évalués sur des modèles MDP aléatoires, en comparaison avec une approche *anytime* classique. Cette comparaison a montré que l'architecture AMPLE intégrant un algorithme optimal était plus efficace qu'un algorithme *anytime* en terme de temps de calcul et de taux de réussite, au prix de plans de moins bonne qualité puisque qu'aucune garantie d'optimalité n'est donnée sur les politiques exécutées par AMPLE.

Une intégration de l'algorithme AEMS (Ross et Chaib-Draa 2007) a été réalisée pour résoudre un modèle POMDP d'une mission de reconnaissance d'objets par un drone (Ponzoni Carvalho Chanel, Teichteil-Königsbuch et Lesire 2013). Dans ce travail, la comparaison entre la version AMPLE et une version *anytime* d'AEMS a également montré un meilleur taux de succès et une durée de mission plus courte avec l'architecture AMPLE.

3.2.3 Module d'exécution

Le module d'exécution est responsable de définir les requêtes de planification demandées au module de planification, et de gérer l'interface avec le reste du système, par le lancement des actions et l'observation de l'état du système.

Le comportement du module d'exécution est défini par une *stratégie*, qui décrit comment les requêtes sont générées et résolues. Nous avons proposé deux stratégies génériques pour résoudre des (PO)MDP.

Stratégie Next

Considérons que le système se trouve dans l'état s_t. Le principe de la stratégie **Next**, illustrée sur la figure 3.3, est alors de :

1. récupérer l'action a_t de la politique locale (action optimisée par une requête précédente, ou action par défaut) ;
2. pour chacun des états suivants possibles s^i_{t+1}, lancer une requête de planification à partir de cet état ;
3. lancer l'exécution de l'action a_t pendant la résolution de ces requêtes.

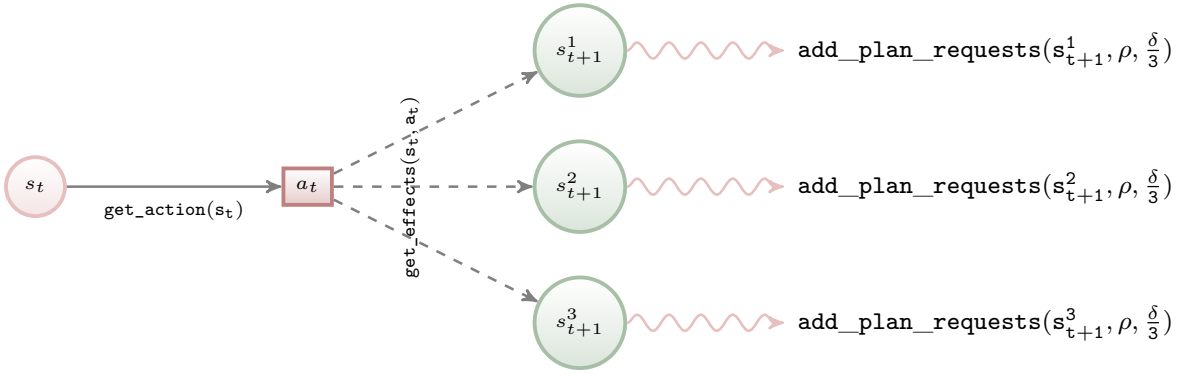


FIGURE 3.3 – Schéma de principe de la stratégie **Next**

Dans cette stratégie, l'objectif est de faire en sorte que l'algorithme de planification ait exploré une partie de l'espace d'état à partir de chacun des états suivants, améliorant ainsi les chances d'avoir une action de meilleure qualité que l'action de la politique par défaut. Le temps alloué à chaque requête est basé sur une répartition équitable de la durée estimée δ pour réaliser l'action a_t. Ceci ne garantit pas la réactivité, mais permet d'améliorer les chances que chaque état suivant ait été exploré. La réactivité est assurée, dans le cas où l'action dure moins longtemps que prévu, par la politique par défaut qui assure d'avoir une action réalisable.

Stratégie Path

La stratégie **Path** a pour objectif d'explorer de manière plus approfondie les chemins les plus probables, afin d'améliorer la qualité des politiques obtenues sur ce chemin. A partir de l'état s_t, la stratégie, illustrée sur la figure 3.4, est de :

1. récupérer l'action a_t de la politique locale ;
2. pour l'état suivant le plus probable (ici s^2_{t+1}), lancer une requête de planification à partir de cet état ;

3. pour chacune des actions réalisables dans l'état suivant le plus probable, récupérer l'effet le plus probable de cette action, et lancer une requête de planification à partir de cet état ;
4. lancer l'exécution de l'action a_t pendant la résolution de l'ensemble de ces requêtes.

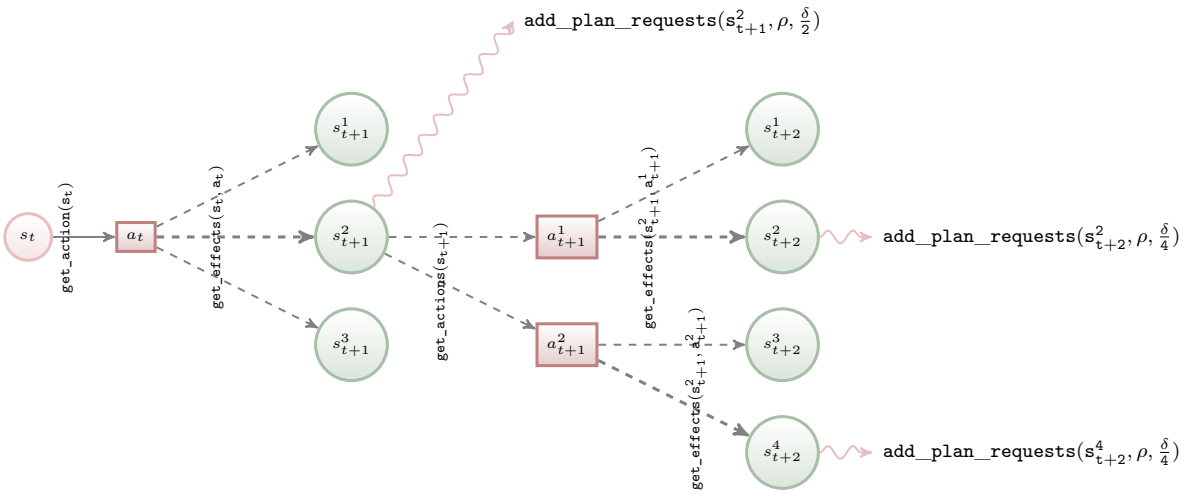


FIGURE 3.4 – Schéma de principe de la stratégie **Path**

La profondeur, c'est-à-dire le nombre d'actions successives à partir desquelles on génère une requête de planification (de 2 dans l'exemple précédent), peut être un paramètre de la stratégie. Le temps alloué à chacune de ces requêtes est inversement proportionnel à leur profondeur dans l'arbre ainsi construit, de sorte à laisser plus de temps pour résoudre les requête qui doivent optimiser les actions à réaliser à plus court terme.

Applications

Dans (PONZONI CARVALHO CHANEL, LESIRE et TEICHTEIL-KÖNIGSBUCH 2014), nous avons décrit le module d'exécution et les stratégies **Next** et **Path**, et comparé ces stratégies pour la résolution de MDP aléatoires, et pour le problème POMDP de reconnaissance d'objets par un drone. Cette comparaison a permis de montrer que la stratégie **Path** réalise moins de calculs lorsque le problème a une trajectoire prépondérante. Lorsque les probabilités sur les effets des actions sont plus équilibrés, **Path** utilise de nombreuses fois la politique par défaut, avec pour conséquence une durée de mission plus longue que pour la stratégie **Next**.

Dans (T'HOOFT, LESIRE et PONZONI CARVALHO CHANEL 2016), nous avons décrit une stratégie ad-hoc pour résoudre de manière efficace un problème de navigation par un robot mobile, en environnement dynamique, en utilisant un algorithme déterministe, et en gérant l'incertitude au niveau du module d'exécution par la génération de requêtes de planification avec des états initiaux divers. Cette stratégie a été formalisée et généralisée dans (PONZONI CARVALHO CHANEL et al. 2019) par la stratégie **Portfolio**.

3.3 Synthèse et perspectives

L'architecture AMPLE proposée répond au problème de planification en temps contraint par :

1. un contrôle fin du temps passé à résoudre chaque requête de planification, basé sur une décomposition des algorithmes de résolution en fonctions élémentaires ;
2. des stratégies d'exécution qui permettent de contrôler comment l'espace d'état est exploré, et qui peuvent être adaptées à l'application ;
3. la présence d'une politique par défaut qui garantit l'existence d'une action à exécuter même si aucune requête de planification n'a fourni de solution pour cet état.

Les perspectives d'évolution de ces travaux sont les suivantes.

3.3.1 Qualité de la politique exécutée

Une des limites actuelle de l'architecture AMPLE est qu'elle n'apporte aucun élément permettant de juger de la qualité de la politique locale maintenue par le module de planification. Même dans le cas où l'algorithme encapsulé dans ce module est un algorithme qui garantit une convergence vers un optimum, rien à ce jour ne permet de reporter cette propriété sur l'architecture AMPLE dans son ensemble.

Il semble donc nécessaire de mener une telle analyse, qui sera bien sûr dépendante à la fois des propriétés de l'algorithme encapsulé, et également de la stratégie d'exécution qui décrit comment cet algorithme est utilisé par AMPLE.

Dans un cadre plus général, indépendant de l'algorithme ou de la stratégie, il est tout de même envisageable de fournir des éléments quant à :

— la qualité de la politique par défaut,

— la mise à jour de la politique locale.

Dans les applications traitées avec AMPLE, les politiques par défaut ont été définies de différentes manières :

— une simple *action par défaut* dans l'application de navigation par un robot mobile, qui consiste à faire réaliser un tour au robot pour mettre à jour sa carte locale ;

— une politique *paramétrée* dans l'application d'atterrissage autonome d'urgence, qui consiste à aller vers la zone la plus proche, et tenter un atterrissage ;

— une politique calculée en résolvant un *problème relaxé* (sans observabilité partielle) dans l'application de reconnaissance d'objets.

Dans le premier cas, l'action par défaut a un rôle difficilement quantifiable, car elle repose sur une connaissance experte de l'impact de cette action sur l'état du robot (meilleure

connaissance de son environnement), et de la définition ad-hoc de la stratégie d'exécution. Cependant, il est possible de constater que l'action par défaut assure l'intégrité du système, car elle ne met pas en danger la sécurité du robot ou de son environnement (contrairement à une action par défaut qui aurait été d'avancer).

Dans le cas de la politique paramétrée, il est possible d'une part d'évaluer la qualité de cette politique (en terme d'espérance de gain – voir (3.1) – ou de succès de la mission), et d'autre part d'appliquer des méthodes de type *model-checking* probabiliste (Kwiatkowska, Norman et Parker 2002) pour évaluer les probabilités d'atteindre des états interdits ou dangereux.

Dans le dernier cas, l'analyse repose sur le problème relaxé, et sur l'algorithme utilisé pour résoudre ce problème. Cependant, cette façon de calculer la politique par défaut repose le problème de la réactivité, car rien ne peut garantir que le problème relaxé sera résolu avant la limite de temps donnée. Dans cette situation, il est préconisé d'avoir une politique par défaut paramétrée, qui garantisse la réactivité, et qu'il est possible d'analyser, et si l'application le nécessite, de mener de front la résolution du problème relaxé et la résolution du problème original, dans une architecture AMPLE qui pourrait être multi-problèmes ou multi-modèles.

3.3.2 Architecture multi-modèles

La stratégie **Portfolio** que nous avons proposée dans (Ponzoni Carvalho Chanel et al. 2019) consiste à utiliser le module de planification avec des paramètres différents. Ces paramètres permettent notamment de spécialiser l'algorithme pour la résolution d'une requête particulière. Dans le cas de l'application de navigation autonome, un des paramètres de l'algorithme était la carte (sous forme de grille d'occupation) utilisée par l'algorithme pour planifier des trajectoires. Le module d'exécution lançait donc des requêtes utilisant des cartes différentes (carte complète, carte locale avec obstacles).

On voit déjà dans cette stratégie que le module d'exécution n'est pas forcément lié à un problème unique. L'extension directe serait même de n'avoir qu'un module d'exécution pour gérer plusieurs modules de planification, ce qui veut dire qu'une stratégie pourrait par exemple faire plusieurs requêtes sur le même état à des modules différents (et donc résoudre des problèmes différents), ou au contraire, associer à certains états un module de planification spécifique.

Les cas pratiques d'une telle stratégie pourraient être :

— une décomposition hiérarchique des problèmes, par exemple en lançant, pour un problème de logistique, des requêtes à un planificateur de tâches pour planifier l'ordre dans lequel déposer des colis, et un planificateur de déplacement pour calculer les trajectoires pour rejoindre les lieux de livraison ;

— la gestion de situations critiques, par exemple en lançant, pour chaque action, des

requêtes à la manière de **Next**, ainsi qu'une requête prenant en compte une panne possible, pour un planificateur de trajectoire de retour à la base.

Ces stratégies posent des questions plus générales de cohérence des modèles, des politiques, des critères optimisés. Ces questions seront abordées dans le chapitre 6.

Chapitre 4

MAUVE : une chaîne outillée pour le développement logiciel en robotique autonome

4.1 Contexte

Pour assurer un comportement autonome du système robotique, notamment face à des incertitudes et des perturbations, différents mécanismes ou différentes architectures peuvent être mis en place, tels que l'architecture AMPLE présentée précédemment (chapitre 3) qui assure une réactivité du système face à l'évolution de l'environnement. Divers concepts d'architectures tolérantes aux fautes ont également été proposés dans la littérature pour assurer un comportement autonome face à des pannes dans le système (Khalastchi et Kalech 2018 ; Crestani, Godary-Dejean et Lapierre 2015).

Pour fonctionner de manière correcte, ces architectures nécessitent que le développement logiciel des différentes fonctions, ainsi que leur exécution sur la plateforme physique, suivent d'une part des principes de *modularité* et mettent en œuvre des mécanismes *temps-réel* garantissant une exécution correcte.

4.1.1 Environnements de développement logiciel pour la robotique

Depuis plusieurs années, la problématique du développement et de l'implantation de fonctionnalités sur un système robotique a reçu un intérêt croissant. Une des raisons vient du besoin d'adopter des processus de développement, tels que ceux issus du génie logiciel, pour maîtriser la complexité croissante des systèmes robotiques, qui intègrent de plus en plus de capteurs, de capacités de perception, d'apprentissage, de planification, etc. Pour aider à ce développement, des middlewares spécifiques à la robotique ont d'abord vu le jour, parmi lesquels les plus connus et les plus utilisés à ce jour sont *Orocos-RTT* (Soetens et Bruyninckx 2005) et *ROS* (Quigley et al. 2009).

Au delà de ces middlewares, des approches à base de composants se sont imposées afin d'apporter de la généralité, de la maintenabilité, de la robustesse au développement de logiciel embarqué (Brugali et Scandurra 2009 ; Brugali et Shakhimardanov 2010). Cette démarche peut être résumée par la figure 4.1, dans laquelle on peut distinguer le

développement d'architectures à composants, et son implémentation et exécution qui, à travers un middleware, repose sur l'exécution de tâches sur le système d'exploitation.

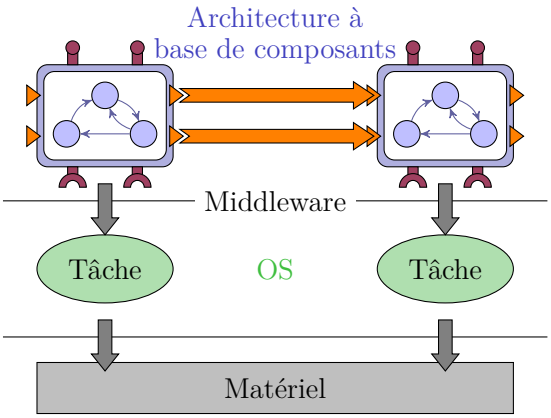


FIGURE 4.1 – Développement d'applications robotiques : des composants aux tâches temps-réel.

Comme représenté dans la figure 2.2, la vocation de ces middlewares ou de ces approches à composants est notamment de soutenir l'implantation des fonctions d'agissement (*acting*), d'observation (*observing*) ou de surveillance (*monitoring*) du système. En plus de l'aspect *modularité* et exécution *temps-réel*, il nous semble donc utile de disposer d'un *modèle* des fonctions logicielles qui sont implantées dans le système.

Dans (GOBILLOT, LESIRE et DOOSE 2019), nous avons analysé plusieurs middlewares, ainsi que plusieurs approches de développement basées sur des modèles, selon plusieurs critères : la modularité, l'implantation temps-réel, la présence d'un langage de spécification (DSL), un processus de génération de code, une modélisation du comportement des composants, l'évaluation de pire temps d'exécution (Worst-Case Execution Time – WCET) et de réponse (Worst-Case Response Time – WCRT). Cette analyse est résumée dans le tableau 4.2.

Cette analyse montre que deux approches basées sur des modèles possèdent quasiment l'intégralité des caractéristiques demandées, $G^{en}{}_{o}M$ et CPAL. Cependant :

— CPAL est basé sur un langage de description de fonctions de contrôle embarquées, et est peu adapté à l'intégration de fonctions typiques de la robotique comme des fonctions de traitement d'image, ou d'accès à des interfaces physiques ;

— $G^{en}{}_{o}M$ est un langage de définition de modules, mais il ne contient pas d'éléments de description d'architectures (et donc des interactions entre modules) ; il n'y a donc pas de modèle complet des fonctions exécutées dans l'architecture.

TABLE 4.2 – Analyse des middlewares robotiques et d'approches basées modèles selon différentss caractéristiques (✓ : caractéristique présente, ∼ : partiellement présente ou limitée, ✗ : absente). Tableau tiré de (GOBILLOT, LESIRE et DOOSE 2019).

		modular	real-time	DSL	code generator	behaviour model	WCET	WCRT
Middlewares	cisst (JUNG et al. 2013)	✓	✓	✗	✗	✗	✗	✗
	RTC (ANDO et al. 2011)	✓	✓	✗	✓	✗	✗	✗
	RoboComp (MANSO et al. 2010 ; MARTÍNEZ et al. 2010)	✓	✓	✓	✓	✗	✗	✗
	Orocos-RTT (SOETENS et BRUYNINCKX 2005)	✓	✓	✗	✗	✓	✗	✗
	ROS (QUIGLEY et al. 2009)	✓	✗	✗	✗	✗	✗	✗
Model-based processes	BIP (BASU et al. 2008)	✓	✓	∼	✓	✓	✗	✗
	CPAL (NAVET et FEJOZ 2016)	✓	✓	✓	✗	✓	✓	✓
	ROCK p.d.	✓	✓	∼	✓	✓	✗	✗
	BRIDE (BRUYNINCKX et al. 2013)	✓	✗	✓	✓	✓	✗	✗
	$G^{en}{}_{o}M$ (MALLET, PASTEUR et HERRB 2010 ; FOUGHALI et al. 2016)	✓	∼	✓	✓	✓	∼	∼
	SmartSoft (SCHLEGEL et al. 2010 ; STECK et SCHLEGEL 2010)	✓	✓	✓	✓	✗	✗	∼

4.2 Contribution

Pour pallier ces limitations, et envisager des extensions plus simples vers les fonctions délibératives de surveillance, d'observation, et d'agissement, nous avons développé MAUVE, une chaîne outillée pour le développement d'applications robotiques, basée sur :

— un langage de modélisation, spécifique aux applications robotiques, le DSL (Domain Specific Language) MAUVE (GOBILLOT, LESIRE et DOOSE 2014) ; ce langage fournit un cadre qui permet d'une part d'appliquer des bonnes pratiques de conception, qui reprennent des concepts issus de pratiques répandues en robotique : la séparation des rôles (SCHLEGEL et al. 2010), le découplage entre modèle et algorithme (MALLET, PASTEUR et HERRB 2010), des patrons de communication (SCHLEGEL 2006) ; ce langage fournit d'autre part un modèle formel des composants et de l'architecture déployés sur le robot ;

— des outils de génération de code embarqué, reposant sur les middlewares Orocos-RTT et ROS, notamment pour supporter l'interopérabilité des applications développées ;

— un algorithme d'évaluation des échéances temps-réel (GOBILLOT et al. 2015) adapté au modèle précédent.

4.2.1 Le DSL MAUVE

Le DSL MAUVE (Gobillot, Lesire et Doose 2014) est basé sur la notion de composants. Chaque composant représente une entité exécutable de l'architecture logicielle.

Composants

Un composant est décrit par :

— une *coquille* (*Shell*), qui décrit l'interface du composant (propriétés, ports d'entrée et de sortie) ;

— un *cœur* (*Core*), qui décrit le comportement du composant au moyen d'une machine à états, dans laquelle chaque état appelle des fonctions élémentaires appelées *codels* et implantées en C++.

Les listings 4.1 et 4.2, tirés de (Gobillot, Lesire et Doose 2019), montrent la description de la coquille du composant *Navigation* (qui calcule et exécute des itinéraires pour rejoindre un objectif sur une carte), et du cœur du composant *P3DX* (qui gère la communication avec le contrôleur physique du robot).

Listing 4.1 – Shell of the *Navigation* component (Gobillot, Lesire et Doose 2019)

```
1  shell NavigationShell {
2    property resolution: double [0.1;1] = 0.3
3    property position_threshold: double [0;1] = 0.3
4
5    input port map: OccupancyGrid
6    input port pose: PoseStamped
7    input port goal: PoseStamped
8
9    output port next: PoseStamped
10 }
```

Architectures

Une architecture spécifie les instances de composants, ainsi que les communications entre ces composants. Le listing 4.3, tiré de (Gobillot, Lesire et Doose 2019), importe l'architecture *NavigationArchitecture*, et y ajouter les composants *Exploration*, *Hokuyo* et *P3DX*. Un schéma de l'architecture obtenue est montrée sur la figure 4.3.

Déploiement et génération de code

La description du déploiement permet de spécifier les caractéristiques d'exécution temps-réel de chaque composant : sa période, sa priorité, et le cœur sur lequel il s'exécute.

MAUVE fournit un générateur de code (figure 4.4), qui prend en entrée la définition des composants, architectures et déploiements, ainsi que le code correspondant aux codels, et génère :

— du code C++ utilisant le middleware Orocos pour les composants et les architectures,

Listing 4.2 – Core of the *P3DX Driver* component (Gobillot, Lesire et Doose 2019)

```
core P3DXCore (P3DXShell) {
  var cmd: TwistStamped;
  var vel: TwistStamped;
  var p: PoseStamped;
  var robot: ArRobotPtr;
  var connector: ArRobotConnectorPtr;

  configure = {
    aria_init(robot, connector, device, baudrate);
    return aria_connect(robot, connector);
  }

  start = { return aria_start(robot, sonarOn, motorOn); }

  update = {
    if (read(command, cmd) == new_data) then {
      aria_command(robot, cmd);
    } else {}
    aria_loop(robot);
    p = aria_pose(robot, odometry_frame);
    write(pose, p);
    vel = aria_velocity(robot, robot_frame);
    write(velocity, vel);
  }

  stop = { aria_stop(robot, true, true); }

  cleanup = { aria_disconnect(robot); }
}
```

Listing 4.3 – Exploration architecture (Gobillot, Lesire et Doose 2019)

```
architecture ExplorationP3dxArchitecture {
  import NavigationArchitecture

  instance exploration: Exploration
  instance hokuyo: Hokuyo
  instance p3dx_driver: P3DX

  connection pose.target -> exploration.pose
  connection exploration.goal -> navigation.goal
  connection gmapping.map -> exploration.map
  connection p3dx_driver.pose -> gmapping.odometry
  connection p3dx_driver.pose -> pose.source
  connection hokuyo.scan -> switch.scan
  connection hokuyo.scan -> guidance.scan
  connection hokuyo.scan -> gmapping.scan
  connection switch.command -> p3dx_driver.command
}
```

— des fichiers de configuration ROS pour déployer l'architecture,

— un modèle de chaque composant sous forme de machine à états périodique, utilisé pour l'analyse d'ordonnançabilité (voir section suivante).

Le générateur de code intègre également des *tracepoints* dans le code généré. Ceci permet, à travers la librairie de trace LTTng †, d'obtenir des traces d'exécution.

4.2.2 Analyse d'ordonnançabilité d'une architecture MAUVE

A partir de la description de l'architecture et de ses composants, et des caractéristiques d'exécution temps-réel du déploiement, l'objectif est d'analyser *a priori*, si l'architecture

†. LTTng : an open source tracing framework for Linux, https://lttng.org/

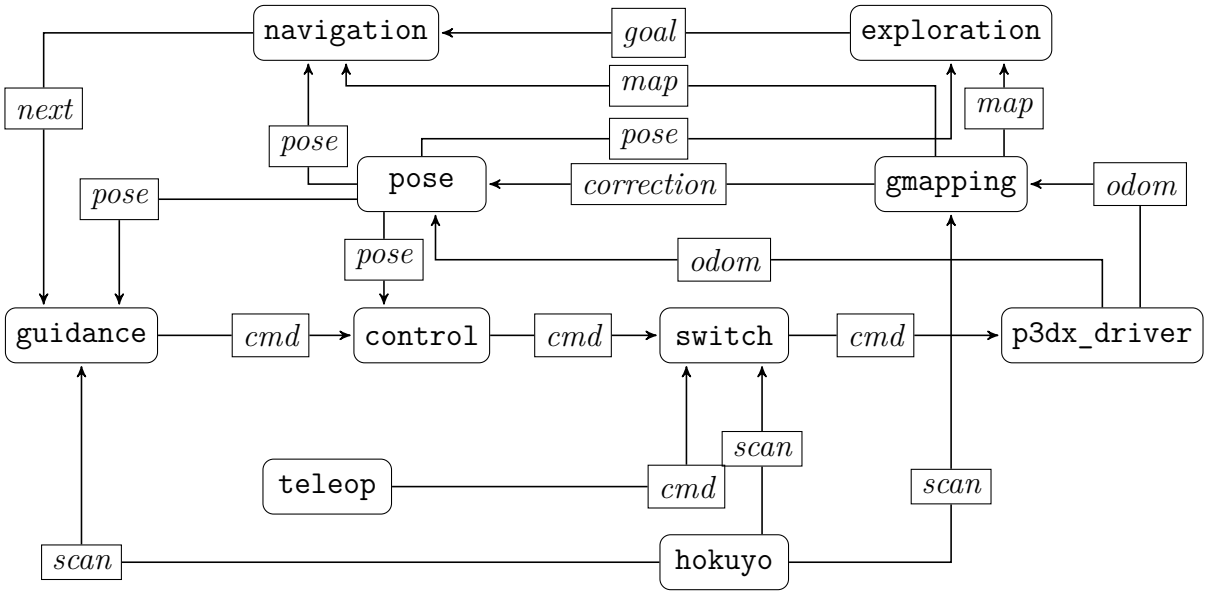


FIGURE 4.3 – The component-based exploration architecture. Rounded rectangles represent components, rectangles with italic labels represent connections between components ports. (GOBILLOT, LESIRE et DOOSE 2019)

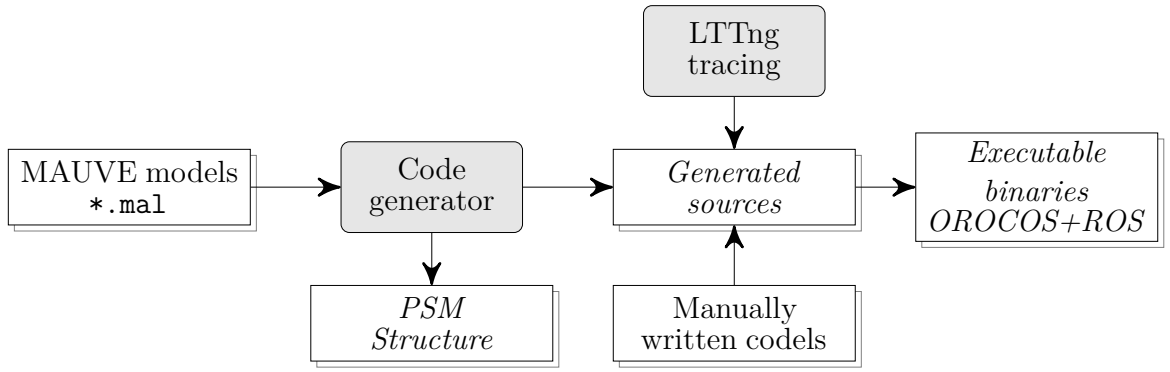


FIGURE 4.4 – Processus de génération de code à partir du DSL MAUVE, tiré de (GOBILLOT, LESIRE et DOOSE 2019).

est ordonnançable.

L'évaluation de cette ordonnançabilité est basée sur le calcul, pour chaque composant, du pire temps de réponse (WCRT – Worst-Case Response Time), c'est à dire de la durée la plus longue entre le début d'une activation du composant et la fin de son exécution (en prenant donc en compte un délai entre activation et début d'exécution, et les éventuelles préemptions).

Les méthodes classiques de calcul de WCRT (LIU et LAYLAND 1973) utilisent, pour chaque tâche, donc dans notre cas pour chaque composant, une seule estimation du pire temps de calcul (WCET – Worst-Case Execution Time). Des travaux plus récents prennent un compte un modèle plus fin du comportement de chaque tâche (TCHIDJO MOYO et al. 2010 ; STIGGE et al. 2011 ; ZENG et DI NATALE 2012), mais le modèle des tâches ne

correspond pas parfaitement à notre modèle de composant, avec pour conséquence une surestimation du WCRT.

Nous avons donc choisi de développer une méthode d'analyse de WCRT adaptée au modèle de composant et au modèle d'exécution MAUVE. Une description détaillée de cette méthode a été publiée dans (Gobillot et al. 2015). Elle repose sur :

— la représentation de l'exécution de chaque composant sous forme de PSM (Periodic State Machine), qui reprend la structure de la machine à état du cœur du composant, et intègre les WCET de chaque codel appelé dans cette machine à état ;

— le calcul de l'ensemble des traces d'exécution de cette PSM ;

— le calcul d'une borne supérieure de la *request bound function* (*rbf*, Baruah, Rosier et Howell 1990) de l'ensemble de ces traces ;

— l'adaptation de la méthode de calcul de WCRT de Liu et Layland (1973) pour utiliser cette *rbf* à la place d'un WCET unique de chaque composant.

Dans (Gobillot et al. 2015), nous avons montré un gain pouvant aller jusqu'à 32% sur l'estimation du WCRT de certains composants de l'architecture de la figure 4.3.

4.2.3 Estimation des pires temps de calcul

L'évaluation du WCRT des composants repose sur une estimation du WCET de chaque codel. Les approches classiques d'estimation du WCET reposent sur une analyse statique du code exécutable (Wilhelm et al. 2008). Cette analyse identifie le pire nombre d'instructions exécutées, mais pour cela nécessite un modèle précis du matériel supportant l'exécution.

L'utilisation de l'analyse statique sur nos architectures nous a montré que cette approche est peu adaptée aux architectures logicielles robotiques (Lesire, Doose et Cassé 2012). Premièrement car les processeurs et cartes utilisées en robotique sont du matériel souvent "grand public", dont le fonctionnement est peu documenté. Deuxièmement, les analyses statiques sont peu adaptées à l'accès à du matériel (capteurs/actionneurs) à travers des appels système. Finalement, il est souvent difficile de borner le fonctionnement d'algorithmes complexes tels que du SLAM ou de la planification de trajectoire, ce qui rend le WCET estimé assez pessimiste.

L'approche alternative est l'analyse probabiliste basée sur des mesures des temps d'exécution (MBPTA – Measurement-Based Probabilistic Timing Analysis), qui estime un WCET probabiliste (ou pWCET) à partir de ces mesures (Cucu-Grosjean et al. 2012). Nous avons appliqué cette approche sur nos architectures (Gobillot et al. 2016), en récupérant des traces d'exécution de l'ensemble des codels, et en appliquant une approche MBPTA utilisant la théorie des valeurs extrêmes pour estimer une distribution de probabilité sur le WCET et qui fournit des métriques permettant d'évaluer la fiabilité de l'estimation (Guet, Santinelli et Morio 2016).

A titre d'exemple, la figure 4.5, tirée de (Gobillot et al. 2016), montre les mesures obtenues sur le codel `avoid_collision` du composant de guidage. La figure 4.6 montre l'estimation du pWCET réalisée à partir de ces mesures.

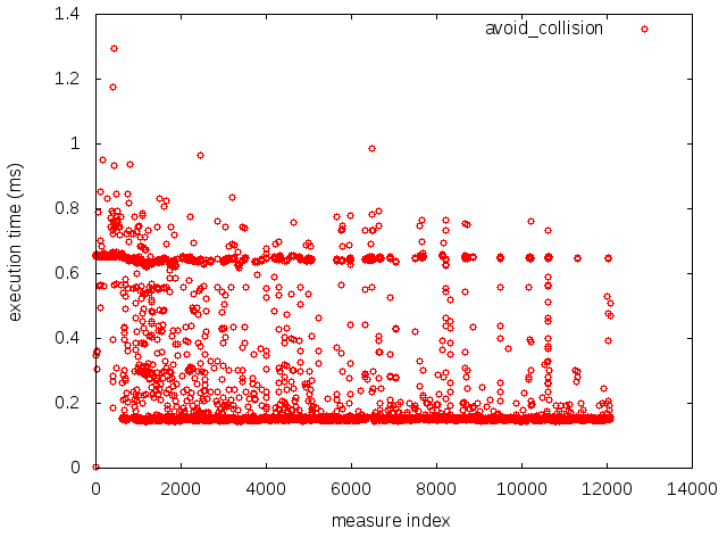


Figure 4.5 – Execution time measures for codel `avoid_collision`. Each circle represents a measure. (Gobillot et al. 2016)

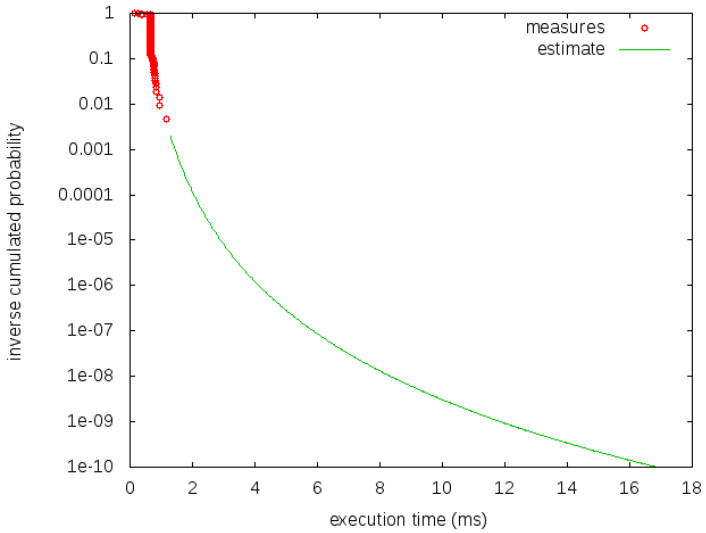


Figure 4.6 – pWCET estimate for codel `avoid_collision`. (Gobillot et al. 2016)

La méthode de calcul du WCRT présentée dans la section précédente ne permet pas de raisonner sur une distribution de pWCET. Par conséquent, à partir des pWCET de chaque codel, nous avons dû choisir un seuil de fiabilité (par exemple, à 10^{-4}) pour avoir une valeur unique de WCET pour chaque codel afin d'appliquer la méthodologie de calcul du WCRT.

4.2.4 Le runtime MAUVE

Les travaux précédents ont mis en avant des limites liées au middleware utilisé, en l'occurrence Orocos, à la fois pour l'évaluation du WCRT, pour faciliter l'utilisabilité de la chaîne MAUVE, et également pour envisager des extensions vers des mécanismes de tolérance aux fautes et de supervision qui soient plus simples à implanter.

Synchronisation des tâches temps-réel

Lors de la spécification du déploiement MAUVE, il est possible de définir, pour chaque composant, une période et une priorité. Cependant, Orocos ne permet pas de contrôler les dates d'activation des différentes tâches qu'il crée sur le système d'exploitation. Ainsi, il nous est impossible de savoir si les dates d'activation de deux tâches de même période seront exactement synchronisées, et sinon, de connaître le décalage entre ces dates. Hors, ceci nous amène, dans le cadre de l'évaluation du WCRT, à considérer le pire-cas, c'est-à-dire celui qui amène un temps de réponse plus élevé. Il nous a semblé donc indispensable d'améliorer ce comportement pour maîtriser la synchronisation (ou le décalage) des dates de réveil.

Adéquation code source / modèle

Afin d'améliorer l'utilisabilité de la chaîne MAUVE, il est apparu important de pouvoir agencer, dans une même architecture, des composants générés depuis le DSL, et des composants développés directement en C++ en utilisant le middleware Orocos. Ceci, d'une part, pour faciliter une utilisation incrémentale du DSL, et d'autre part, pour permettre à un développeur d'utiliser des fonctionnalités du langage C++ qui peuvent parfois être utiles pour une bonne implantation, et dont l'utilisation à travers le DSL (et donc uniquement à travers des codels), peut être laborieuse.

Hors, la différence entre les concepts utilisés dans le DSL et ceux du middleware Orocos est très importante : pas de notion de coquille, des fonctionnement du middleware inexistant dans le DSL car difficiles à analyser, etc.

Il nous a donc semblé utile d'avoir une API en C++ qui expose exactement les mêmes concepts et les mêmes fonctionnalités que le DSL.

Mécanismes de reconfiguration

Le dernier point limitant est le besoin d'implanter, au niveau du middleware, des fonctionnalités de reconfiguration qui permettent de conserver les propriétés temps-réel de l'architecture. De telles fonctionnalités vont de la connexion/déconnexion de ports de données, au remplacement de la partie algorithmique (le cœur) d'un composant.

Le runtime MAUVE

Pour répondre à l'ensemble de ces limitations, et devant la difficulté de les intégrer au middleware Orocos sans reprendre toute la structure du code source, nous avons décidé de développer en C++ un nouveau middleware temps-réel, le *Runtime MAUVE*.

Ce middleware répond à l'ensemble des limitations ci-dessus, en offrant des mécanismes de maîtrise des synchronisations entre composants, en exposant une interface C++ qui soit cohérente avec le modèle formel des composants et architectures, et en intégrant des mécanismes de reconfiguration temps-réel.

Le runtime MAUVE est décrit dans (Doose, Grand et Lesire 2017), dans lequel une évaluation du comportement temps-réel a été réalisée, en comparaison avec une implantation équivalente en ROS (non temps-réel) et en Orocos dans trois configurations différentes : le composant de déploiement en mode *background* (non temps-réel), plus prioritaire que les autres composants, ou isolé sur un cœur du processeur. Cette évaluation est résumée par les tableaux 4.7 et 4.8, qui donnent des statistiques sur le temps d'exécution et la périodicité des deux composants A et B dans les différentes configurations.

	Execution time				Periodicity			
	A		B		A		B	
	mean	stdev	mean	stdev	mean	stdev	mean	stdev
ROS	35.358	2.561	19.912	1.813	100.002	0.628	100.002	0.658
OROCOS-bg	31.936	1.447	12.028	1.301	99.998	0.831	100.0	0.057
OROCOS-prio	30.097	2.076	12.015	1.276	99.992	2.299	100.0	0.214
OROCOS-isolated	31.573	1.598	12.021	1.194	99.991	1.521	100.0	0.244
MAUVE	20.059	0.725	11.975	1.254	99.997	1.589	100.0	0.216

Table 4.7 – Statistics on component executions in milliseconds. The execution time is the time between the start and the end of each execution (including preemptions). The periodicity is the time between two successive executions. (Doose, Grand et Lesire 2017)

	First execution offset				First response time			
	A		B		A		B	
	mean	stdev	mean	stdev	mean	stdev	mean	stdev
ROS	7.024	1.280	50.272	5.196	37.826	3.040	70.141	6.641
OROCOS-bg	101.332	4.504	100.002	0.013	134.281	0.909	114.324	0.901
OROCOS-prio	104.143	6.603	100.007	0.011	135.018	2.306	114.200	1.120
OROCOS-isolated	105.486	6.920	100.033	0.036	134.154	1.500	113.751	1.278
MAUVE	14.288	1.086	0.110	0.030	36.242	2.373	14.232	1.080

Table 4.8 – Statistics on first execution of each component in milliseconds. The execution offset is the time between the start/launch of a component and its first execution. The reponse time is the time between the start and the end of the first execution. (Doose, Grand et Lesire 2017)

Ces résultats montrent que, si les périodes sont respectées par tous les middlewares (période spécifiée de 100ms), les temps d'exécution, notamment du composant A qui est

le moins prioritaire, sont moins rigoureux avec ROS et Orocos. On voit de plus sur le tableau 4.8 que la première exécution est très pénalisée par Orocos, avec des écarts-types importants.

Le runtime MAUVE est disponible en open-source, sous licence LGPL. Les instructions et la documentation sont disponibles sur `https//mauve.gitlab.io`.

4.3 Synthèse et perspectives

La chaîne MAUVE fournit des outils de développement et d'analyse d'architectures logicielles qui facilitent le développement, en se basant sur des modèles à la fois des composants, de leur exécution temps-réel, et de leurs connexions. MAUVE a aujourd'hui été utilisé sur plusieurs robots du laboratoire (robots terrestres d'intérieur et d'extérieur, drones aériens multi-rotors, drones aériens hélicoptères, maquette de drone avion), ainsi que sur un véhicule autonome de surface et sur un robot agricole.

L'objectif est maintenant, comme illustré sur la figure 2.2, que MAUVE puisse servir de socle pour accueillir des fonctions d'observation, de surveillance et d'agissement. Pour cela, il est nécessaire de d'abord établir des modèles des différentes capacités, ressources, variables, manipulées au sein de l'architecture fonctionnelle, et également manipulables par l'architecture délibérative.

Pour aller dans ce sens, nous avons initié deux travaux :

— le premier autour de la modélisation par réseaux de Petri des capacités fournies par l'architecture fonctionnelle, et leur combinaison à travers des opérateurs algébriques ; ce travail préliminaire a été publié dans (Lesire et Pommereau 2018) ;

— le second autour de la spécification d'observateurs en combinant logique temporelle et contraintes temporisées, et la synthèse de ces observateurs en garantissant un comportement temps-réel ; ce travail préliminaire a été soumis dans (Lesire et al. 2019).

Ces travaux, et leur apport aux architectures délibératives, seront rediscutés dans le chapitre de perspectives (chapitre 6).

Chapitre 5

Planification et exécution pour des missions multi-robots

5.1 Contexte

Lorsqu'on dispose d'une équipe de robots autonomes, et que ces robots ont des capacités différentes et complémentaires, des mécanismes de décision permettent de tirer parti de cette complémentarité et d'une redondance pour être robuste à des aléas (Parker 2008). De nombreux projets étudient aujourd'hui l'apport d'équipes de robots autonomes pour des missions de recherche et sauvetage (De Cubber et al. 2013 ; Marconi et al. 2012), ou de recueil d'information (Pralet et Lesire 2014 ; Briñón-Arranz, Pascoal et Aguiar 2014). Les missions typiques que l'on considère ici sont des missions de surveillance de zones ou de sites, que ce soient des zones inaccessibles (sous-marines par exemple), ou des sites d'intervention après catastrophe ou accident. Pour ces missions, les opérateurs ou les forces d'intervention ont besoin d'une assistance pour gérer de manière efficace la réalisation de la mission, et donc que l'équipe de robots déployée pour ces missions soit autonome, c'est-à-dire capable de délibérer.

Ces missions demandent que les architectures délibératives de systèmes multi-robots soient ainsi capables de gérer :

— des *incertitudes* liées à l'environnement (présence d'obstacles, position de personnes blessées, ...) ou à la réalisation de la mission (temps nécessaire pour réaliser une tâche, observations imparfaites) ;

— des *pannes* qui peuvent réduire les capacités des robots (capacités de franchissement limitées, capteurs défaillants) ;

— des pannes ou de l'incertitude sur les *communications*, qui peuvent empêcher les robots de communiquer entre eux ou avec le centre d'opération ;

— des interactions avec des *opérateurs*, premièrement en intégrant les contraintes ou les souhaits de l'opérateur dans la planification de la mission, et deuxièmement en fournissant des moyens d'interagir avec l'équipe de robots en cours de mission.

5.1.1 Approches distribuées

Différentes approches distribuées ont été proposées pour réaliser des missions multi-robots. Ces approches reposent sur une résolution locale, par chaque agent, d'un problème de planification ou de la gestion de l'exécution d'un plan, et les agents s'échangent des tâches ou des buts. Il n'y a donc pas de résolution d'un problème global, commun à l'ensemble des agents.

L'approche emblématique est l'approche Market-Based (Smith 1980), appliquée à plusieurs reprises pour des missions d'exploration multi-robots (Zlot et Stentz 2003 ; Dias et al. 2006). Ces travaux souffrent d'une part du manque de garanties de performances, même si Lagoudakis et al. (2005) ont réalisé des analyses permettant d'obtenir des bornes sur ces performances ; et d'autre part de la sensibilité aux problèmes de communication défaillante. Otte, Kuhlman et Sofge (2017) ont réalisé une évaluation empirique de différentes variantes des approches Market-Based en présence de communications défaillantes. Cette évaluation a mené à constater et analyser les différentes variantes relativement aux nombres de tâches non affectées, réalisées à plusieurs reprises, ainsi que l'oisiveté des robots.

Des approches non basées sur des enchères, mais où les robots s'échangent des informations (position des robots, tâches non réalisées), ont également été proposées pour des missions d'exploration multi-robots. Belbachir, Ingrand et Lacroix (2012) ont par exemple étendu l'architecture T-REX (McGann et al. 2008) en une architecture multi-robot, dans laquelle les agents s'échangent leurs cartes, qui sont ensuite gérées localement par chaque architecture. Santos et Egerstedt (2018) réalisent une couverture de zones de manière distribuée par un découpage en cellules de Voronoi qui prend en compte l'hétérogénéité des capteurs ; les robots doivent s'échanger des informations sur leurs positions et leurs capteurs pour converger vers un découpage pertinent.

5.1.2 Décision séquentielle dans l'incertain multi-agents

Les approches de décision séquentielle dans l'incertain, basées sur des modèles (PO)MDP (introduits au paragraphe 3.1.1), ont été étendues dans un cadre multi-agents. Ces extensions permettent de prendre en compte le fait que les agents vont choisir de réaliser des actions concurrentes, mais n'auront qu'une observation (partielle ou complète) locale de leur environnement. Les premières extensions ont mené à définir le cadre Multiagent MDP (MMDP, Boutilier 1996), le cadre Decentralized MDP (Dec-MDP, Becker et al. 2004), et le cadre Decentralized POMDP (Dec-POMDP, Bernstein et al. 2000). La complexité d'une résolution optimale de ces problèmes (NEXP-complets) rend leur utilisation rédhibitoire, menant à des approches sous-optimales ou heuristiques pour leur résolution.

De nombreuses variantes ont été proposées pour prendre en compte les contraintes de communication entre robots, par la modélisation explicite d'actions de communica-

tion (Goldman et Zilberstein 2003 ; Williamson, Gerding et Jennings 2009), ou par des modèles spécifiques, par exemple avec interactions ponctuelles (Dec-SIMDP, Melo et Veloso 2011).

Ces approches reposent cependant sur une modélisation a priori probabiliste des actions réalisées par les robots, de l'environnement (localisation des objets à trouver par exemple), et des communications. Ces modèles sont parfois difficiles à construire, et peuvent devenir complexes si on souhaite y intégrer une modélisation de défaillances ou l'arrivée de nouveaux objectifs de mission.

Une approche alternative intéressante est celle proposée par Unhelkar et Shah (2016) : le modèle d'action est supposé déterministe, car les auteurs considèrent que l'asservissement des robots est assez robuste. En revanche, l'incertitude vient de la méconnaissance de l'environnement, et donc des états dans lesquels on applique ces actions. L'approche consiste donc en la résolution, de manière distribuée, d'un Dec-MDP avec une estimation déterministe de la fonction de transition. Les robots s'échangent ensuite des informations sur leurs observations pour mettre à jour cette fonction de transition, et recalculer une politique locale adaptée. Cette approche est intéressante car elle évoque la résolution continue d'un problème de décision séquentielle dans l'incertain décentralisé, mais comporte des limites, comme la quantité d'informations à échanger (modèles et politiques locales des différents robots).

5.1.3 Planification et réparation

Les approches précédentes résolvent le problème de manière centralisée, en prenant en compte l'incertitude, puis les politiques sont exécutées de manière décentralisée, éventuellement en s'échangeant des informations.

Une alternative est de créer un plan initial pour l'équipe de robots, qui prenne ou non en compte une partie de l'incertitude, puis de modifier ce plan pendant l'exécution en réponse à différents événements (pannes, observations, nouveaux objectifs, ...)

Ces approches peuvent se mettre en œuvre dans deux types d'architectures : des architectures centralisées, où une seule entité est responsable de calculer le plan initial, et de modifier/réparer/recalculer ce plan lors de l'occurrence d'événements, ou des architectures décentralisées, dans lesquelles les modifications du plan sont faites au niveau des robots de l'équipe.

Architectures centralisées

Les architectures centralisées décrites dans la littérature peuvent se regrouper en différentes catégories selon les approches algorithmiques utilisées :

1. les approches basées optimisation ou programmation par contraintes, qui modélisent un problème d'allocation de tâches, ou de routage, éventuellement en prenant en

compte des synchronisations entre robots (Yu, Budhiraja et Tokekar 2018 ; Banfi et al. 2018) ; dans ces travaux le problème est modélisé sous forme de problème de programmation linéaire en nombre entiers (ILP) ou mixtes (MILP) ; la réparation du plan est abordée dans (Banfi, Basilico et Carpin 2018) pour redéployer les robots de manière à maintenir les communications ;

2. les approches basées planification hiérarchique, dans lesquelles un planificateur centralisée utilise un HTN (Hierarchical Task Network, Nau et al. 1999 ; Nau et al. 2003) pour représenter le problème multi-robots, puis décompose ce HTN et alloue des tâches individuelles aux différents robots ; lors de l'arrivée d'événements, Gancet et al. (2005) replanifient le HTN, alors que dans l'architecture Retsina (Paolucci, Shehory et Sycara 2000), les tâches sont redistribuées entre robots, mais sans réutiliser la structure initiale du HTN ; l'architecture HOTRiDE (Fazil Ayan et al. 2007) propose un schéma de réparation intéressant, qui utilise à la fois la structure hiérarchique du HTN et les liens causaux entre tâches, mais n'a été définie et utilisée que pour un contexte mono-robot.

Quelques travaux n'utilisent pas directement un cadre HTN ou similaire, mais proposent des approches hiérarchisées :

— Di Rocco, Pecora et Saffiotti (2013) utilisent une approche mixte, dans laquelle un planificateur centralisé assure la résolution d'un problème représenté par un réseau de contraintes, mais dont la description des *activités* est hiérarchique ; l'exécution du plan est faite de façon distribuée, mais la propagation des contraintes dans le réseau, et la replanification, sont faites de manière centralisée ;

— Schillinger, Bürger et Dimarogonas (2018b) décomposent une mission spécifiée par une formule LTL en tâches indépendantes, qui sont alloués à des agents tout en planifiant leurs actions pour réaliser ces tâches ; la décomposition du problème est donc hiérarchisée par l'algorithme de résolution, mais aucune information hiérarchique n'est réellement modélisée dans le problème ;

— Gombolay, Wilcox et Shah (2018) proposent une architecture centralisée qui résout un problème d'allocation et d'ordonnancement par un algorithme itératif, qui résout un problème de programmation linéaire en nombres entiers ; ils modélisent une tâche par un réseau de contraintes sur des sous-tâches, et les capacités des agents par une relation agent – sous-tâche.

Architectures décentralisées

Des architectures décentralisées proposent de modifier le plan en cours de mission en fonction des événements ou aléas survenus. Kantaros et Zavlanos (2018) calculent de manière régulière un plan sur un horizon borné, pour prendre en compte les nouvelles tâches arrivant à la volée. Le problème résolu de manière décentralisé est modélisé sous

forme MILP, et prend en compte la nécessité d'établir une communication en fin de plan afin de se synchroniser pour recalculer le plan suivant. Popescu, Rivano et Simonin (2016) adoptent une modélisation similaire pour résoudre un problème de patrouille avec des capacités de communication limitées.

Schillinger, Bürger et Dimarogonas (2018a) proposent une architecture décentralisée, avec la synthèse initiale d'une politique pour réaliser une mission modélisée par un MDP avec options ; les robots utilisent ensuite une approche basée enchères pour s'attribuer des options à réaliser lors de l'exécution de la mission.

5.2 Contribution

La conclusion de l'état de l'art synthétique précédent est que peu de travaux s'intéressent à la réparation ou la replanificaiton de mission de manière décentralisée, dans un contexte où les communications sont restreintes et potentiellement défaillantes. Les architectures les plus proches replanifient des graphes de communication, afin de pouvoir utiliser une replanification centralisée.

De plus, dans le contexte de missions opérationnelles de surveillance de sites, ou de soutien à des équipes d'intervention, deux points sont essentiels afin de définir des architectures et algorithmes qui soient acceptés et qui correspondent au besoin :

— la prise en compte de contraintes ou procédures expertes ; les architectures de l'état de l'art prenant en compte une telle expertise sont celles basées HTN (Gancet et al. 2005 ; Paolucci, Shehory et Sycara 2000), mais elles sont centralisées ;

— le calcul d'un plan initial, qui permet aux opérateurs d'avoir une certaine confiance au comportement de l'équipe de robots, et d'anticiper son évolution ; lors des replanifications, fournir un plan d'équipe (lorsque la communication le permet) participe aussi à cette confiance en l'équipe de robots autonomes déployée.

5.2.1 Une architecture de supervision décentralisée basée HTN

Contexte

Les premières missions que nous avons considérées sont des missions de sécurisation de zones dans un contexte maritime, au moyen de robots aériens et sous-marins. Dans cette mission, les contraintes opérationnelles sont importantes : l'équipe de robots doit respecter des procédures définies. De plus, les utilisateurs du système robotique veulent imposer des comportements spécifiques en réaction à certains événements ou certaines pannes du système.

Architecture HiDDeN

Dans ce contexte, nous avons opté pour une représentation de l'ensemble de ces exigences opérationnelles (procédures à respecter, comportements spécifiques) sous forme hiérarchique. La planification hiérarchique, dont l'approche emblématique est la planification HTN (Hierarchical Task Network, Nau et al. 1999 ; Nau et al. 2003), permet en effet d'intégrer plus facilement des connaissances expertes, ou des contraintes liées à ces procédures, en spécifiant des ensembles de tâches qui doivent s'organiser d'une manière prédéfinie pour réaliser une tâche plus abstraite.

Pour cela, nous avons développé une architecture multi-robots basée sur l'exécution de plans hiérarchiques, issus d'un planificateur HTN conventionnel. La spécificité de l'architecture proposée est d'une part, à partir du HTN global, de distribuer le plan aux différents robots en y synthétisant des actions de communication, et d'autre part, de mettre en place des mécanismes de réparation locale. Ces mécanismes de réparation reposent sur deux principes :

— lorsqu'une tâche échoue, elle est remplacée par une nouvelle branche hiérarchique si une réparation existe ;

— si aucune réparation locale n'est possible, on essaie de réparer la tâche de niveau supérieur.

La figure 5.1 représente le plan hiérarchique de la mission de sécurisation maritime. La mission (tâche `scenII`) est décomposée en une séquence de sous-tâches, parmi lesquelles des tâches de synchronisation, insérées pour assurer l'exécution décentralisée (tâches `ack_beg` et `ack_end`), des tâches attribuées à l'opérateur (qui doit acquitter la réalisation de tâches affectées aux robots), et des tâches de l'équipe de robots.

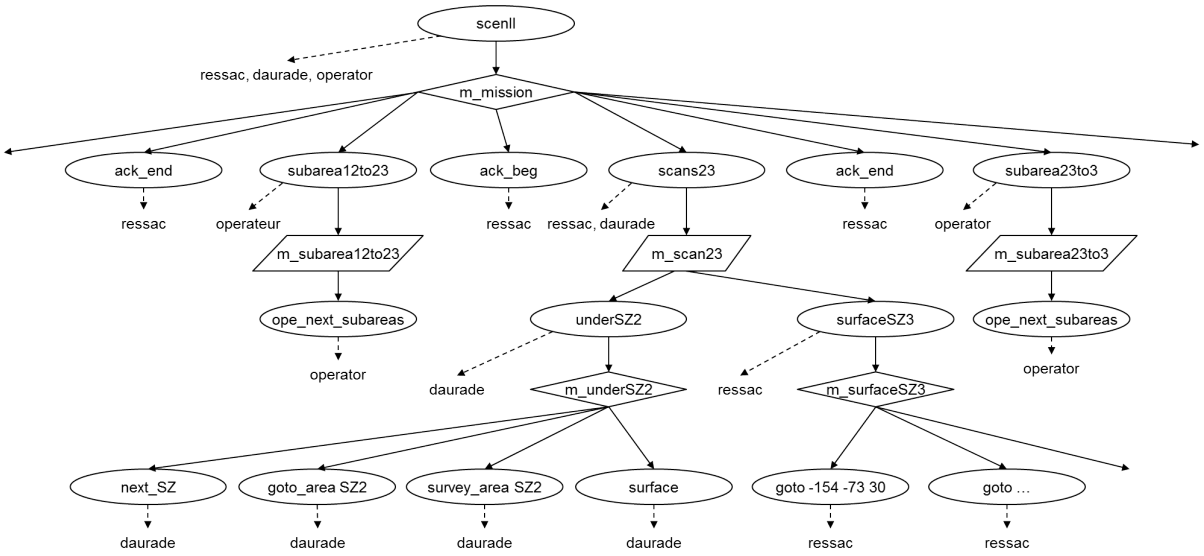


Figure 5.1 – Plan hiérarchique pour deux robots réalisant la mission de sécurisation maritime, tiré de (Lesire et al. 2016).

Les principes généraux de cette architecture ont été décrits dans (Gateau, Lesire et Barbier 2013). Son application à une mission de sécurisation de zone maritime par un drone aérien et un sous-marin autonome, dont les expérimentations ont été réalisées sur le lac de Castillon en 2014 (figure 5.2), est décrite dans (Lesire et al. 2016).

Figure 5.2 – Photos du robot sous-marin et du drone aérien lors des expérimentations, tirées de (Lesire et al. 2016).

Conclusion

L'architecture HiDDeN, basée sur une décomposition hiérarchique de la mission ainsi que des réactions à événements, est adaptée pour modéliser des procédures opérationnelles et ainsi maîtriser le comportement et l'évolution du système multi-robots, ce qui est apprécié des opérateurs humains.

Cependant, il est difficile de décrire de telles décompositions de manière exhaustive lorsque la mission à réaliser devient plus complexe, à cause du nombre de robots impliqués, de la nécessité d'optimiser des critères (de couverture, de détection), ou de raisonner sur des contraintes temporelles (nécessité de se recharger, gestion des durées de déplacement, comptes-rendus périodiques, ...)

5.2.2 Architecture de planification/réparation hybride

Planification hybride POP et HTN

Afin d'intégrer dans notre architecture un algorithme de planification qui permette d'une part de raisonner sur la base de buts à atteindre, et de contraintes sur les actions à réaliser (préconditions, effets, contraintes temporelles), et d'autre part de pouvoir intégrer des contraintes opérationnelles sous forme hiérarchique, nous avons opté pour le paradigme de la *planification hybride*, qui mixe une planification hiérarchique (par décomposition des méthodes abstraites, comme en planification HTN) et une planification partiellement ordonnée (Partial-Order Planning – POP – Penberthy et Weld 1992 ; Younes et Simmons 2003 – aussi appelée planification dans l'espace des plans ou Partial-Order

Causal-Link planning). Différents travaux ont défini des cadres de planification hybride, tels que UCP (Kambhampati, Mali et Srivastava 1998), CHIMP (Stock et al. 2015), PANDA (Schattenberg 2009) ou encore FAPE (Dvorák et al. 2014).

Comme aucun de ces travaux ne propose d'architecture multi-robots ou de principe de réparation, nous avons choisi de développer un nouvel algorithme de planification hybride, HiPOP (Bechon et al. 2014). L'algorithme principal de HiPOP (algorithme 5.1) est le même algorithme que celui d'un POP classique. Le plan initial $\mathcal{I}$ contient uniquement l'état initial et le but du problème de planification. L'algorithme est constitué d'une boucle principale comprenant 3 étapes :

1. la récupération du meilleur plan Π parmi les plans partiels de l'arbre de recherche $\mathcal{P}$,
2. le choix d'un défaut f de ce plan Π,
3. la résolution de ce défaut, et l'ajout des plans résultants à l'arbre de recherche $\mathcal{P}$.

Algorithm 5.1: Algorithme POP

Input : $L, A, \mathcal{I}$

$\mathcal{P} = \{\mathcal{I}\}$;
while $\mathcal{P} \neq \emptyset$ **do**
  $\Pi = PopBestPlan(\mathcal{P})$;
  if $\mathcal{F}(\Pi) = \emptyset$ **then**
    return Π ;
  end
  $f = PopBestFlaw(\mathcal{F}(\Pi))$;
  $\mathcal{P} = \mathcal{P} \cup Resolvers(A, \Pi, f)$;
end
return $\emptyset$

Les algorithmes POP considèrent deux types de défauts : les *liens ouverts*, c'est-à-dire les préconditions d'actions qui ne sont pas liées à un effet d'une autre action, et les *menaces*, c'est-à-dire les effets d'actions qui sont en conflit avec un lien causal établi. En plus de ces deux défauts, HiPOP considère les *défauts abstraits*, c'est-à-dire les actions abstraites du plan (au sens HTN) qui n'ont pas été décomposées par une méthode en sous-tâches. La résolution de chacun de ces défauts peut se faire de la façon suivante :

— un lien ouvert est résolu en créant un lien entre un effet d'une action du plan et la précondition ouverte, ou en ajoutant une nouvelle action (éventuellement abstraite) au plan courant ;

— une menace est résolue en ajoutant une contrainte de précédence entre les actions impliquées dans le lien causal menacé et l'action produisant l'effet menaçant ;

— un défaut abstrait est résolu en décomposant l'action abstraite selon une de ses méthodes.

Dans (Bechon et al. 2014), nous avons de plus proposé et évalué différentes heuristiques pour décider du meilleur plan (fonction *PopBestPlan*) et du défaut à résoudre (fonction *PopBestFlaw*).

La figure 5.3 montre un problème simple dans lequel deux véhicules doivent observer deux cellules (marquées par une croix rouge), mais le véhicule aérien ne peut observer les cellules occupées par des arbres.

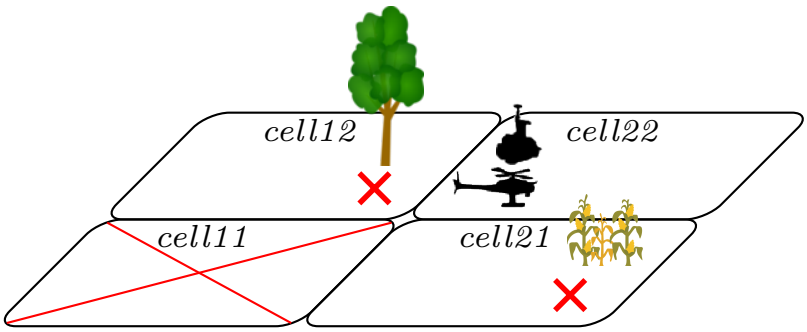


Figure 5.3 – Exemple de problème d'exploration à deux véhicules.

La figure 5.4 montre un plan trouvé par HiPOP pour ce problème. Les tâches abstraites sont représentées en rouge, les liens causaux par des flèches entre tâches.

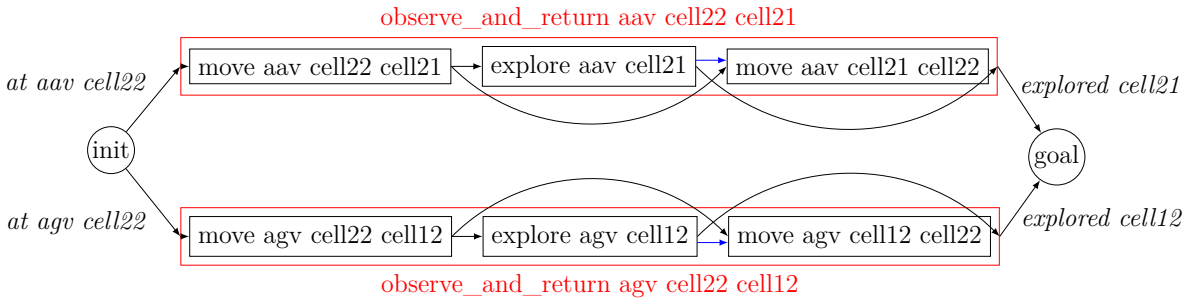


Figure 5.4 – Plan calculé par HiPOP pour l'exemple de la figure 5.3.

Dans le cadre du PEA ACTION, HiPOP a été utilisé pour planifier des missions d'observation de points d'intérêts sur une zone de type village, en utilisant deux robots aériens et six robots terrestres. Le plan obtenu est représenté sur les figures 5.5 (trajectoires planifiées) et 5.6 (ordonnancement des tâches par robot).

Exécution décentralisée de plans temporels

Le plan synthétisé par HiPOP doit être exécuté par chacun des robots. Les premiers événements que nous avons considérés sont les retards dans la réalisation des actions. Ces retards peuvent avoir plusieurs impacts : tout d'abord une perte globale d'efficacité, vu que le plan a été calculé pour réduire le durée totale de la mission, et également une violation

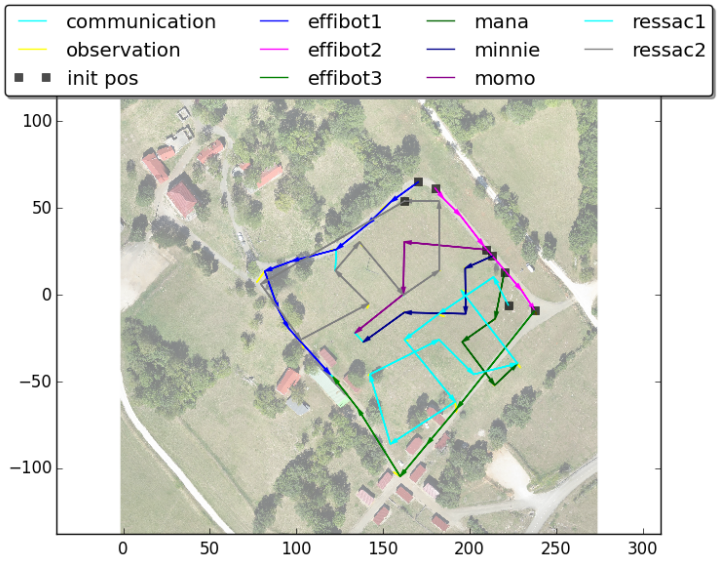


FIGURE 5.5 – Trajectoires planifiées pour un scénario du projet ACTION, figure tirée de (BECHON et al. 2018).

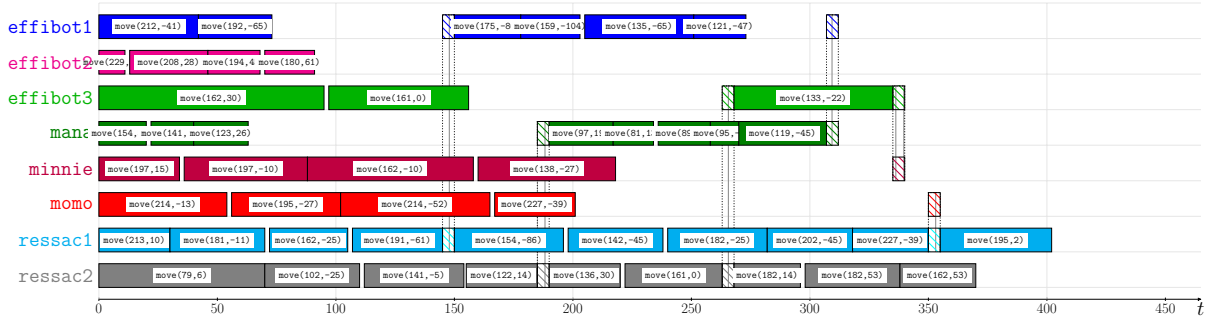


FIGURE 5.6 – Représentation temporelle du plan initial ; chaque ligne représente le plan d'un robot ; les actions d'observation sont courtes, donc invisibles sur cette représentation ; les actions de communication sont représentées par les barres verticales hachurées. Figure tirée de (BECHON et al. 2018)

de contraintes temporelles imposées par l'opérateur. En effet, dans les scénarios considérés, l'opérateur peut imposer d'avoir un retour des informations provenant des robots en fixant une échéance pour une tâche de communication entre un robot terrestre et un robot aérien par exemple (lequel relaie les informations vers l'opérateur). Les tâches de communication ont donc des échéances à respecter, et un retard d'un des robots peut rendre ces échéances insatisfiables.

Les approches classiques de suivi et d'adaptation de contraintes temporelles reposent sur le formalisme des réseaux temporels simples (Simple Temporal Network – STN – DECHTER, MEIRI et PEARL 1991). Ce cadre fournit des algorithmes permettant de propager une contrainte sur un réseau de manière efficace. Ce cadre a été étendu au cas multi-agents (Multiagent Simple Temporal Network – MaSTN – BOERKOEL et al. 2013), mais les

algorithmes proposés ne permettent pas une mise à jour incrémentale efficace du MaSTN. Nous avons donc développé des algorithmes de mise à jour incrémentale, avec le soucis de concevoir un algorithme robuste aux communications intermittentes.

Le problème que l'on cherche à résoudre est, à partir d'un MaSTN correspondant au plan courant, de propager une modification d'une contrainte temporelle (typiquement un retard, c'est-à-dire la modification de la durée maximale d'un arc du réseau temporel) de sorte à recalculer l'ensemble des dates de réalisation (au plus tôt et au plus tard) des actions du plan.

Dans (Casanova, Pralet et Lesire 2015), nous avons proposé quatre algorithmes pour résoudre le problème de propagation dans un MaSTN, représentés sur la figure 5.7 :

— CIP (Centralized Incremental Propagation) implémente une architecture centralisée : l'agent A connaît l'ensemble du MaSTN, et lorsqu'un retard survient sur un autre agent, ce dernier envoie l'information à A, qui propage ce retard (avec des méthodes STN classiques), puis renvoie le STN local aux agents ;

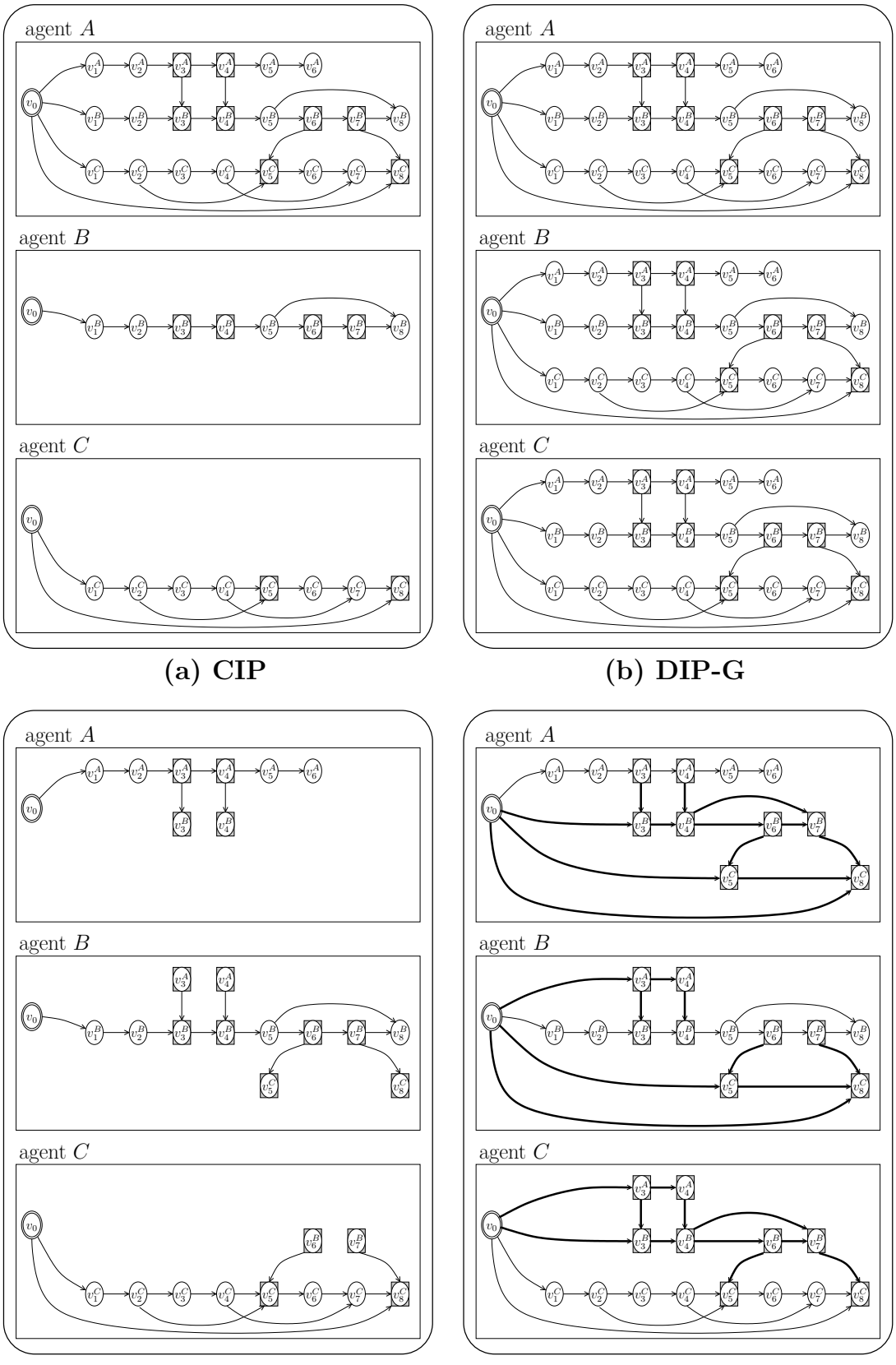
— DIP-G (Distributed Incremental Propagation with Global information sharing), une architecture décentralisée dans laquelle chaque agent a une vue complète du MaSTN ; les agents s'échangent les informations brutes sur les retards observés, et chaque agent fait ensuite la propagation sur sa copie du STN global ;

— DIP-L (Distributed Incremental Propagation with Local information sharing) : les agents n'ont qu'une vue locale du MaSTN (leurs tâches plus les contraintes externes) ; le processus de propagation est donc complètement distribué, et les agents doivent s'échanger des informations afin de faire converger la propagation ;

— DIP-M (Distributed Incremental Propagation with Macro information sharing) : les agents ont une vue détaillée de leurs actions, et une vue macroscopique des actions des autres agents ; les agents s'échangent les contraintes portant sur la partie macroscopique, mais chaque agent est responsable de faire sa propre propagation.

Dans (Casanova, Pralet et Lesire 2015), nous avons évalué ces quatre variantes, et conclu sur l'utilisation privilégiée de DIP-M, qui offre un bon compromis entre temps de calcul et robustesse à des pertes de communication, étant donné que chaque agent peut propager des contraintes sur son macro-STN même s'il n'a pas reçu l'ensemble des mises à jours des autres agents.

DIP-M a ainsi été intégré à l'architecture multi-robots déployée dans le cadre du PEA ACTION. La figure 5.8 montre la réalisation temporelle de la mission dont le plan initial est décrit sur la figure 5.6. Cette mission a pu être terminée en effectuant uniquement des propagations décentralisées de retards.



(a) CIP **(b) DIP-G**

(c) DIP-L **(b) DIP-M**

FIGURE 5.7 – Algorithmes proposées pour la propagation incrémentale de MaSTN, tiré de (CASANOVA, PRALET et LESIRE 2015).

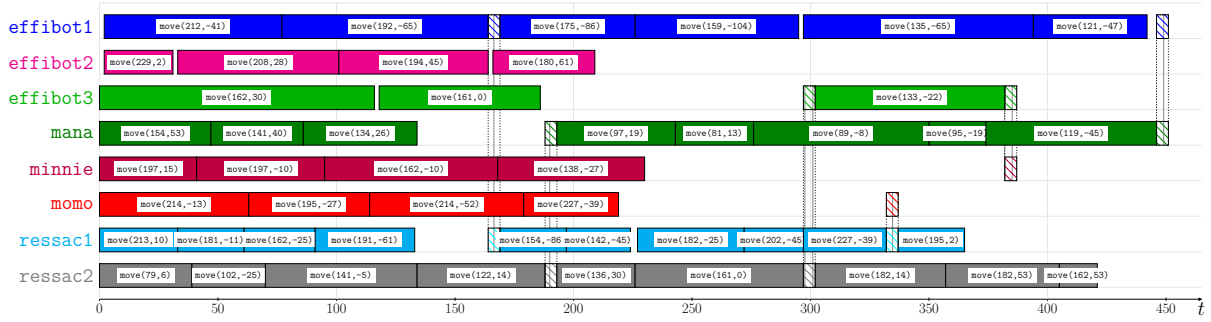


FIGURE 5.8 – Représentation temporelle du plan exécuté ; chaque ligne représente le plan d'un robot ; les actions d'observations sont courtes, donc invisibles sur cette représentation ; les actions de communication sont représentées par les barres verticales hachurées. Figure tirée de (BECHON et al. 2018)

Architecture de réparation décentralisée

Il peut également arriver que la propagation de ces retards échoue, car il est impossible de respecter une contrainte temporelle. Dans ce cas, il est nécessaire de modifier le plan de l'équipe de robots. Ces réparations sont également nécessaires si un robot ne peut réaliser une action, ou s'il est hors-service à cause d'une panne.

Pour répondre à ces situations, nous avons mis en place une architecture de réparation décentralisée. Nous avons tout d'abord proposé un algorithme de réparation utilisant HiPOP (BECHON et al. 2015). Cet algorithme (figure 5.9) :

1. lit un plan initial (le plan courant) ;
2. enlève de ce plan les éléments obsolètes (l'action ayant échoué, ou l'ensemble des actions non réalisées d'un robot hors-service) ;
3. regarde si une échéance est violée, et dans ce cas
4. enlève l'ensemble des actions entre l'instant courant et l'échéance, puis regarde si une autre échéance est violée (retour en 3) ;
5. si aucune échéance n'est violée, on lance une recherche en utilisant HiPOP, avec comme plan initial le plan partiel ainsi construit ;
6. si HiPOP trouve une solution, le plan est réparé ;
7. si HiPOP ne trouve pas de solution, et que le plan est vide, alors
8. aucune solution n'est possible ;
9. si HiPOP ne trouve pas de solutions et que le plan n'est pas vide, on supprime davantage d'actions, en enlevant les actions établissant un lien causal avec des actions enlevées dans l'itération précédente, puis on itère sur l'étape 5.

Pour appliquer cet algorithme de réparation dans le cas de communications défaillantes ou limitées, nous l'avons intégré de manière décentralisée dans l'architecture du projet

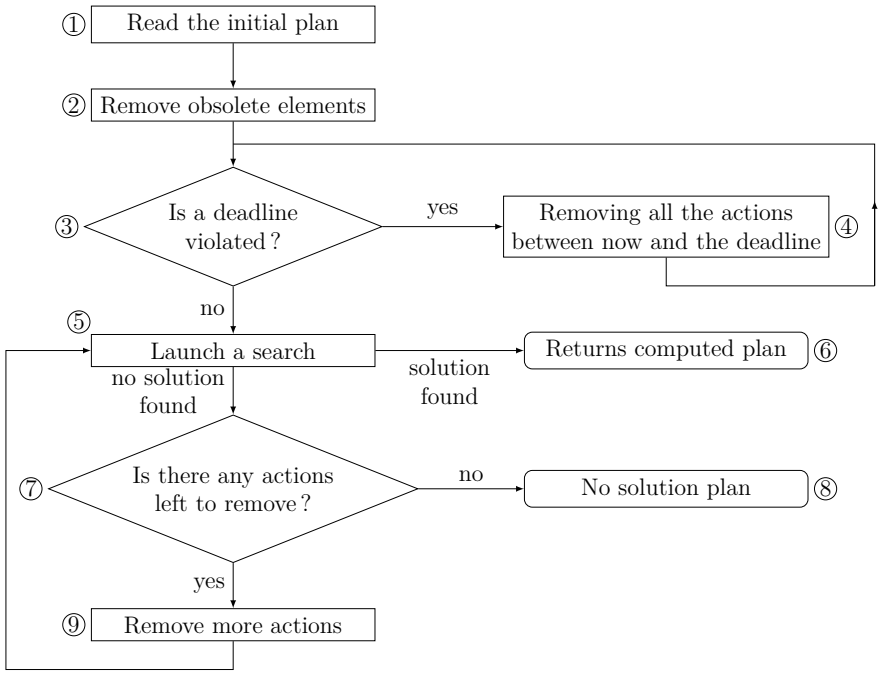


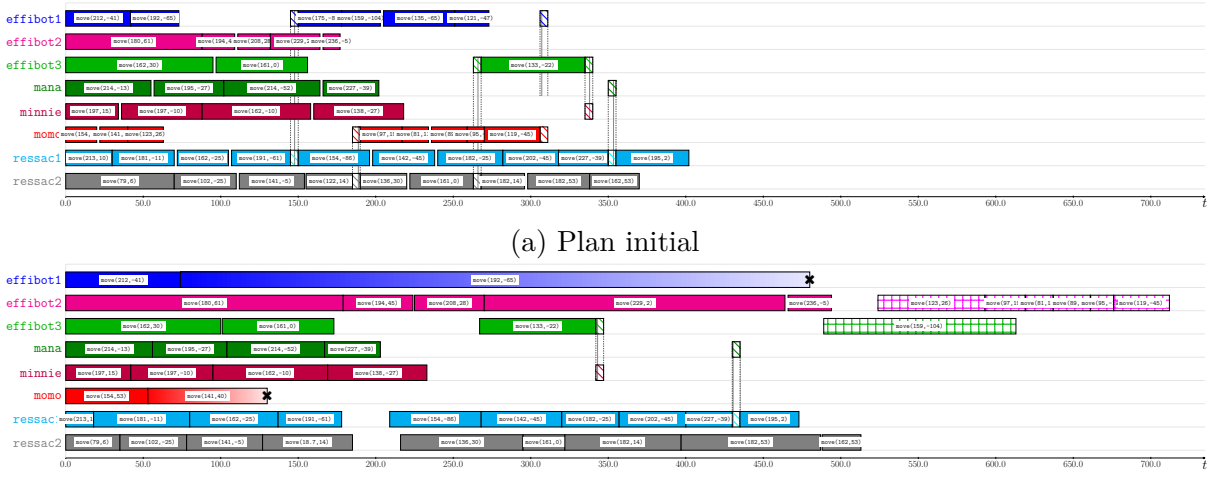
FIGURE 5.9 – Iterative Repair Algorithm (BECHON et al. 2015)

ACTION. Pour cela, le protocole retenu lors de l'observation d'un événement entraînant une réparation (échéance violée, échec d'une action, robot hors-service) est :

- le robot observant l'événement devient le robot *réparateur* ;
- il contacte l'ensemble des robots à portée de communication, et leur demande de se mettre en mode *réparation* ;
- les robots en réparation envoient leur plan courant au robot *réparateur* ;
- ce dernier met en œuvre l'algorithme itératif de réparation, à ceci prêt qu'il n'est pas possible d'enlever du plan les actions des robots qui ne sont pas en communication ;
- si une solution est trouvée, les plans sont renvoyés aux robots en réparation et la mission reprend ;
- si aucune solution n'est trouvée, la mission s'arrête.

Ce protocole permet notamment d'utiliser la capacité des plans partiels à être facilement fusionnables : si jamais les robots hors de portée ont propagé des retards sans replanifier, le plan global est toujours cohérent, et il sera possible de faire une fusion au rétablissement de la communication. Si les robots hors de portée ont également réparé le plan (sans toucher aux actions des premiers robots ayant réparé), les plans pourront également être fusionnés, mais il est possible qu'une incohérence soit levée (à cause d'une échéance violée), et il pourrait être nécessaire de refaire une étape de réparation au rétablissement de la communication.

Les plans de la figure 5.10 montrent, pour un des scénarios du projet ACTION, le plan initial (figure 5.10a) et le plan réalisé par l'équipe de robots (figure 5.10b).



(a) Plan initial

(b) Plan executé par les robots

FIGURE 5.10 – Représentation temporelle des plans d'un scénario avec deux robots hors-service en cours de mission. Figure tirée de (BECHON et al. 2018)

Dans ce scénario, deux robots sont tombés en panne : *momo*, puis *effibot1*. Les réparations ont mené à la ré-affectation de tâches d'observation aux robots *effibot2* et *effibot3*.

5.3 Synthèse et perspectives

Dans un contexte opérationnel où l'équipe de robots doit être autonome pour planifier sa mission, l'exécuter en gérant des retards, et délibérer de manière décentralisée pour réparer sa mission face à des aléas, nous avons proposé des architectures délibératives décentralisées, basée HTN dans le cadre de l'architecture HiDDeN, ou basée planification hybride et MaSTN pour les missions de surveillance et d'observation. L'utilisation de représentations hiérarchiques a permis de capturer les connaissances ou les procédures opérationnelles. Les approches POP ont permis de faciliter la gestion des communications partielles de par leur propriété d'être fusionnable plus naturellement que les autres cadres de planification.

Les contextes d'utilisation d'équipes de robots sont cependant des contextes plus généraux, dans lequels le système multi-robots ne doit pas seulement réaliser des tâches de surveillance d'une zone particulière, mais également être capable de gérer d'autres tâches, comme le transport de marchandise, le déploiement de réseaux de communication, l'observation, ponctuelle ou récurrente, d'un lieu en particulier, ...

Pour aller vers une autonomie permettant une telle utilisation du système multi-robots, plusieurs problèmes et perspectives se posent, notamment d'un point de vue des architectures délibératives. Les perspectives brièvement évoquées ci-dessous seront rediscutées dans le chapitre 6.

5.3.1 Représentation des capacités des robots

Les missions plus générales décrites précédemment nécessitent de pouvoir utiliser des robots pour réaliser différents objectifs. Ceci nécessite de disposer de modèles des capacités des robots, en terme d'actions réalisables, et de ressources disponibles ou nécessaires à la mise en œuvre de ces capacités. Dans l'architecture basée sur HiPOP, cette représentation est faite dans le langage PDDL, qui impose de faire le lien entre la description d'actions et les capacités des robots à travers des prédicats logiques, ce qui rend d'une part la modélisation peut aisée, et d'autre part peut produire des problèmes de planification complexes, pour lesquels les temps de résolution peuvent exploser.

Cette représentation de capacités (actions, ressources) doit de plus être en lien avec l'architecture fonctionnelle déployée sur chaque robot, comme discuté dans les perspectives du chapitre précédent (paragraphe 4.3).

5.3.2 Architecture délibérative hiérarchisée

Si le système multi-robots doit être capable de réaliser, dans une même mission, des tâches ou objectifs de nature très différents, il semble pertinent d'utiliser des algorithmes de planification, voire des architectures délibératives, adaptés à ces objectifs. Si l'architecture proposée basée sur HiPOP semble en effet pertinente et adaptée pour des problèmes d'allocation de tâches avec contraintes temporelles, il semble peu pertinent de l'utiliser pour faire du suivi de cible, du déplacement coordonné, ou d'autres tâches pour lesquelles des algorithmes plus spécifiques ont été développés dans la littérature.

Il me semble ainsi pertinent de *hiérarchiser* l'architecture délibérative, en distinguant, à plus haut niveau, une modélisation globale de la mission, qui sera raffinée par sous-objectifs ou sous-tâches. Ce principe hiérarchique est très proche de la distinction entre *raisonnement sur les buts* et *planification* (pour réaliser un ou plusieurs buts à un instant donné), telle que discutée dans (Ghallab, Nau et Traverso 2014 ; Ingrand et Ghallab 2017). Cependant, les architectures utilisant un raisonnement sur les buts considèrent généralement que :

— l'opérateur ne peut spécifier la mission et interagir avec le système qu'au niveau le plus haut,

— deux niveaux de délibération suffisent, un niveau de gestion des buts et un niveau de planification de tâches pour réaliser ces buts.

Ces deux points me semblent limitants pour concevoir une architecture délibérative qui réponde à l'enjeu de missions plus générales.

Chapitre 6

Perspectives

Les travaux présentés dans ce mémoire ont apporté des contributions à la définition de concepts et d'outils pour concevoir des architectures délibératives pour des systèmes robotiques. Les travaux sur la chaîne MAUVE ont abouti à un socle d'outils pour la conception d'architectures logicielles. Les travaux sur l'architecture AMPLE et sur les architectures multi-robots ont abouti à la proposition d'architectures délibératives qui permettent de réaliser de manière robuste (à l'incertitude, aux pertes de communication, aux défaillances) des missions bien identifiées : atterrissage autonome d'urgence, localisation et identification de cibles, surveillance de zone maritime, allocation de tâches d'observation.

L'utilisation de systèmes robotiques dans des contextes opérationnels nécessite d'étudier la performance de ces systèmes dans des tâches longues ou récurrentes (surveillance de site 24h/24), ou dans des tâches variées (réaliser une tâche de lever de doute, ou une tâche d'exploration, avec le même système robotique).

Dans ce chapitre, je propose quelques perspectives de recherche ayant pour vocation de définir des concepts d'architectures délibératives, et des outils permettant leur développement, pour répondre à ce défi d'autonomie à long terme, avec deux points de vues qui me semblent cruciaux pour faciliter leur utilisabilité et leur acceptabilité :

— la possibilité d'analyser ces architectures pour apporter des garanties, quant à la fiabilité ou aux performances du système robotique ; ceci repose sur une modélisation formelle cohérente de l'ensemble du système ;

— la facilité de spécifier les comportements ou les objectifs attendus du système robotique.

6.1 De l'architecture fonctionnelle à la gestion de l'exécution des plans

Les travaux sur MAUVE présentés dans le chapitre 4 utilisent les concepts de composants, de ressources et de communications, et d'architectures logicielles qui instancient et connectent ces composants. Cependant, pour lier ces concepts à des mécanismes délibératifs de planification, de surveillance, ou d'action, il est nécessaire de fournir un niveau plus abstrait, en terme de fonctionnalités ou de services rendus par de telles architectures logicielles.

Ahma et Babar (2016) ont réalisé une analyse des articles publiés sur les architectures logicielles pour la robotique, qui montre une évolution des pratiques, des approches orientées-objet dans les années 2000 aux approches à base de composants dans les années 2010, et depuis quelques années, l'émergence des approches orientées service.

6.1.1 Architectures auto-adaptables pour la robotique

Un des objectifs des architectures orientées service est la capacité d'utiliser, en ligne, les services disponibles afin des les composer pour réaliser une tâche. Ces architectures sont donc reconfigurables, ou adaptables à une situation ou à une tâche. Dans le domaine de l'informatique, ces concepts d'adaptation ou de reconfiguration ont été répandus sous la terminologie *informatique autonomique* (Kephart et Chess 2003).

Le système SHAGE (Kim et Park 2006) repose sur une description des services fournis par chaque composant du système. Les stratégies d'adaptation de l'architecture sont décrites par : 1. des contraintes sur l'architecture courante pour appliquer la reconfiguration, 2. des modifications à apporter à l'architecture, en terme de services (à ajouter, supprimer, remplacer). Le système SHAGE raisonne ensuite sur les composants disponibles pour concrétiser la reconfiguration. Des approches similaires sont utilisées par Tajalli et al. (2010) et Sykes et al. (2008) où les services fournis par chaque composant sont modélisés par des préconditions et post-conditions, et les architectures sont générées en utilisant un algorithme de planification basé sur PDDL. Georgas et Taylor (2008) proposent de spécifier ces instances d'architectures dans des *politiques d'adaptation*. Ces politiques sont implantées dans le composant d'*analyse* d'une couche composée d'un composant de *collecte* (qui concentre des données permettant de surveiller le système), d'un composant d'*analyse* qui déduit de ces observations l'adaptation à réaliser, et d'un composant d'*administration* qui agit sur l'architecture (Edwards et al. 2009).

Les architectures orientées service modélisent les services au niveau des composants élémentaires. Si cette représentation est adaptée au domaine de l'informatique (services web, internet des objets), les applications robotiques suivent un processus de développement assez différent, dans lequel le rôle d'intégrateur (différencié du rôle de développeur de composants par Schlegel et al. 2010) implique une expertise dans l'utilisation structurée de composants pour remplir une fonctionnalité, ou fournir un service. Cette expertise est mise en avant par Gherardi et Brugali (2014) sous le terme d'*architecture de référence*.

Quelques travaux abordent la problématique de la fiabilité en proposant des architectures tolérantes aux fautes (Fault-Tolerant Architectures). Par exemple, Crestani, Godary-Dejean et Lapierre (2015) utilisent une méthode systématique (AMDEC) pour identifier les fautes pouvant survenir dans l'architecture d'un robot terrestre, et implantent des modules de détection au niveau de l'architecture logicielle pour détecter ces fautes. Ces modules, ainsi que les mécanismes de reconfiguration (qui correspondent à

différentes modalités de navigation), reposent sur une spécification manuelle.

6.1.2 Contrôle d'exécution et programmation de missions

En robotique autonome, l'approche principalement adoptée a été de mettre en œuvre des mécanismes d'adaptation du plan permettant d'assurer une robustesse du système robotique. Ces approches ont reposé sur des langages de programmation (Simmons et Apfelbaum 1998 ; Bresina et al. 1999 ; Gat 1997), des langages de modélisation représentant des procédures hiérarchisées (Verma et al. 2005 ; Georgeff et Ingrand 1989 ; Ingrand et al. 1996), ou des règles de décision spécifiant les actions à déclencher dans certaines conditions (Noreils 1990).

Ces travaux se sont surtout attachés à représenter la décomposition de plans ou d'actions pour contrôler le système robotique, c'est-à-dire à implanter la fonctionnalité d'*agissement* (figure 2.1). Mais le lien avec l'architecture logicielle du système contrôlé est essentiellement réalisé par une programmation manuelle des procédures, des règles, ou par une implantation ad-hoc des actions du système. Ces travaux ne reposent donc pas sur un modèle de l'architecture et de ses fonctionnalités.

Quelques travaux essaient tout de même de pallier ce manque et sont discutés ci-dessous.

Performance Level Profiles Brafman, Bar-Sinai et Ashkenazi (2016) proposent une description de modules fonctionnels, qui peuvent donc agréger différents composants, et qui est utilisée pour synthétiser d'une part des plans utilisant ces modules en utilisant le cadre RosPlan (Cashmore et al. 2015), et d'autre part des composants de surveillance qui vérifient que le comportement du module (ou ses sorties) correspond à sa spécification. La description de ces modules ne repose pas sur un modèle de l'architecture fonctionnelle, mais sur la description de "profils de performance", qui contiennent des éléments intéressants pour raisonner au niveau de l'architecture délibérative. On peut voir par exemple sur le listing 6.1, la description d'un PLP pour un module de déplacement, décrivant les paramètres du module (`execution_parameters`), les données nécessaires à son exécution (`input_parameters`), les variables non observables (`non_observable`), les variables lues (`variables`), les ressources requises (ici `robot_wheels`), des préconditions, des invariants (`concurrency_conditions`), des effets (`achievement_goal`), des effets de bord (ici la consommation d'énergie), une mesure de progression (ici, la formule `closer_to_target` indique que la distance à la cible diminue), des probabilités de succès (et d'échec, non montrées sur cette liste) en fonction de formules sur les variables, et des modes de défaillance.

Task Coordination Language Lotz et al. (2013) proposent le langage TCL (Task Coordination Language) pour décomposer des blocs de manière hiérarchique. Si les décompositions sont assez limitées (séquentielles avec éventuellement des préemptions de tâches),

Listing 6.1 – PLP d'un module assurant un suivi de consigne de déplacement.

```
<plps:achieve_plp name="walk_given_distance" version="1.0">
 <parameters>
  <execution_parameters>
   <param name="linear_distance" />
  </execution_parameters>
  <input_parameters>
   <param name="laser_scan" read_frequency="5" />
   <param name="energy_level" />
  </input_parameters>
  <output_parameters/>
  <non_observable>
   <param name="clear_path" />
  </non_observable>
 </parameters>

 <variables>
  <var name="current_linear_speed" type="real" />
  <var name="collision_alert" type="boolean" />
  <var name="arm_moving" type="boolean" />
  <var name="energy_consumed" type="real" />
  <var name="distance_to_target" type="real" />
  <var name="robot_at_target" type="boolean" />
 </variables>

 <required_resources>
  <resource name="robot_wheels" ><status type="exclusive" /></resource>
 </required_resources>

 <preconditions>
  <formula_condition key_description="not_moving_linear">
   <expression value="current_linear_speed"/>
    <operator type="="/><expression value="0"/>
  </formula_condition>
 </preconditions>

 <concurrency_conditions>
  <formula_condition key_description="not_arm_moving">
   <expression value="arm_moving" /><operator type="="/><expression value="FALSE" />
  </formula_condition>
 </concurrency_conditions>

 <side_effects>
  <assignment_effect key_description="energy_down">
   <param name="energy_level" /><expression value="energy_level - energy_consumed"/>
  </assignment_effect>
 </side_effects>

 <progress_measures><progress_measure frequency="2">
  <formula_condition key_description="closer_to_target">
   <expression value="distance_to_target" />
   <operator type="less" />
   <expression value="distance_to_target@pre"/>
  </formula_condition>
 </progress_measure></progress_measures>

 <achievement_goal><formula_condition key_description="at_target">
  <expression value="robot_at_target"/><operator type="="/><expression value="TRUE"/>
 </formula_condition></achievement_goal>

 <success_probability><conditional_probability>
  <formula_condition key_description="clear_path">
   <expression value="clear_path"/><operator type="="/><expression value="TRUE"/>
  </formula_condition>
  <probability value="0.95" />
 </conditional_probability></success_probability>

 <failure_modes>
  <failure_mode><formula_condition key_description="has_collision">
   <expression value="collision_alert"/><operator type="="/><expression value="TRUE"/>
  </formula_condition></failure_mode>
 </failure_modes>
</plps:achieve_plp>
```

les actions proposées dans le langage permettent d'agir sur l'architecture. Pour cela, TCL propose les actions suivantes :

— `tcl-param`, pour modifier des paramètres de composants,

— `tcl-state`, pour activer ou désactiver un composant,

— `tcl-wiring`, pour changer la connexion entre composants,

— `tcl-activate-event`, pour activer un événement correspondant à un composant (déclencher un service par exemple),

— `tcl-kb-update` et `tcl-kb-query` pour interagir avec une base de données.

Le lien entre ces actions et l'architecture logicielle utilise les modèles de composants issus du cadre SmartSoft (Steck, Lotz et Schlegel 2011).

Reactive Model-based Programming Language Les travaux de Ingham, Ragno et Williams (2001) et Williams et al. (2003) combinent une représentation du système par des automates, et le langage RMPL (Reactive Model-based Programming Language) pour programmer un contrôleur en utilisant des constructions (parallélisme, séquence, récurrence, conditions, boucles, ...) basées sur des contraintes posées sur les états des automates. Ce langage est ensuite traduit en automates concurrents hiérarchiques, dont l'exécution repose notamment sur la résolution des contraintes sur les états des automates. Les constructions proposées sont extrêmement riches et pertinentes pour des applications de robotique autonome. On pourra remarquer que des travaux plus actuels diffusés dans la communauté robotique (Colledanchise et Ögren 2017 ; Lima et al. 2018 ; Heinzemann et Lange 2018) proposent des constructions plus limitées, voire cherchent à s'étendre vers des constructions plus riches (Colledanchise et Natale 2018) par des constructions inspirées de RMPL. Un exemple de programme RMPL est présenté dans le listing 6.2, qui décrit le contrôleur qui allume un des deux moteurs pour réaliser une insertion en orbite pour un satellite. Les égalités sont des contraintes posées sur l'état des automates des moteurs (`EngineA`, `EngineB`) ou de la caméra. La construction `do-watching` exécute le

Listing 6.2 – Contrôleur en RMPL, tiré de (Ingham, Ragno et Williams 2001).

```
1  OrbitInsert ()::
2    ( do-watching ( (EngineA = Firing) OR (EngineB = Firing) )
3      ( parallel
4        (EngineA = Standby)
5        (EngineB = Standby)
6        (Camera = Off)
7        ( do-watching (EngineA = Failed)
8          ( when-donext ( (EngineA = Standby) AND (Camera = Off) )
9              (EngineA = Firing)
10         )
11       )
12       ( when-donext ( (EngineA = Failed) AND (EngineB = Standby) AND (Camera = Off) )
13           (EngineB = Firing)
14       )
15     )
16   )
```

bloc situé en dessous, jusqu'à ce que la contrainte soit satisfaite (un des deux moteurs allumés en ligne 2). La construction `when-donext` attend que la contrainte soit satisfaite pour exécuter le bloc suivant. Ce programme essaie d'allumer le moteur A. En cas d'échec, il essaie d'allumer le moteur B.

Les travaux de Williams et al. (2003) ont cependant été utilisés dans des cadres très restreints, où certes il est nécessaire de gérer des pannes dans le système, mais où l'architecture logicielle est relativement pauvre comparée aux traitements aujourd'hui embarqués dans des systèmes robotiques. Les modèles d'automates du système sont peu détaillés, n'offrant pas de représentation des compétences du système, de ses modalités, et de ses ressources.

6.1.3 Axes de recherche et propositions

Modélisation des compétences par réseaux de Petri

Du point de vue du processus d'agissement, l'interface avec la plate-forme robotique et son architecture logicielle est décrite par l'ensemble des compétences disponibles. Cette description des compétences doit s'appuyer sur un modèle formel, qui permette de raisonner sur les entrées nécessaires pour exécuter cette compétence, sur le résultat attendu, ainsi que sur les ressources nécessaires. Les éléments proposés par Brafman, Bar-Sinai et Ashkenazi (2016) (durée d'exécution, chances de succès ou d'échec, avancement de l'exécution) sont des informations intéressantes à intégrer à une telle modélisation des compétences.

Un formalisme adapté à la représentation des compétences (dont notamment à l'accès concurrent à des ressources) est le formalisme des réseaux de Petri (Petri 1962 ; Murata 1989). J'ai proposé une première formalisation de compétences sous forme de réseau de Petri à flot de contrôle, c'est-à-dire dans lequel on distingue des places de contrôle (place d'entrée, de sortie, interne), qui peuvent être marquées par des jetons représentant un comportement nominal ou une exception. Ce travail préliminaire a été publié dans (Lesire et Pommereau 2018). Un exemple de réseau de Petri de compétence est présenté sur la figure 6.1. Cette compétence a en entrée une variable `call`, utilise deux ressources de type `lock`, et retourne la valeur de la variable `ret`. La place p_{exec} gère l'exécution réelle de la compétence. Dans ce premier travail, l'exécution des compétences est réalisée de manière ad-hoc en s'interfaçant à des actions ROS, mais la représentation des variables d'états d'entrée/sortie et des ressources utilisées permet de raisonner sur les enchaînements de compétences ou leur concurrence, et de disposer d'outils d'analyse. Par rapport aux éléments proposés dans la spécification des PLP (Brafman, Bar-Sinai et Ashkenazi 2016), ce modèle permet déjà d'intégrer les paramètres d'entrée, les préconditions, les ressources utilisées, et les effets. Les éléments de type probabilité de succès ou d'échec, ou durée d'exécution, pourraient être intégrés au modèle de réseau de Petri de compétence (à

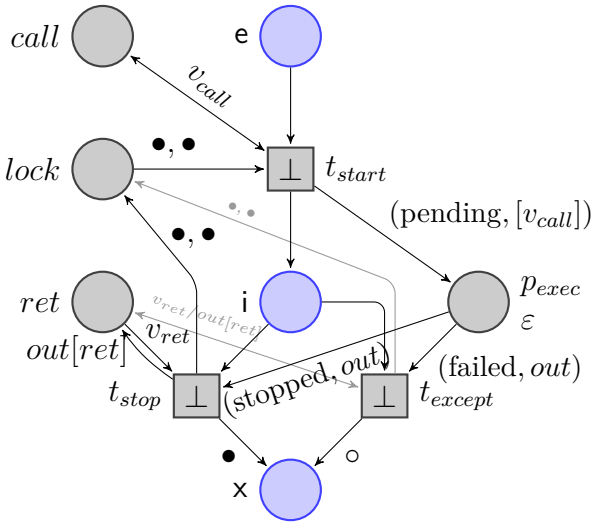


FIGURE 6.1 – Réseau de Petri de compétence. Les places e, i et x sont les places de contrôle d'entrée, interne, et de sortie. Les jetons nominaux sont représentés par un •, les jetons d'exception par un ◦. Figure tirée de (LESIRE et POMMEREAU 2018).

travers des extensions temporisées ou stochastiques), notamment en vue de réaliser des analyses sur le modèle de compétence. Des éléments de type mesure de progression ou invariants pourraient être utiles pour détecter des défaillances dans la réalisation de la compétence, sans attendre un compte-rendu d'échec venant du composant logiciel.

Modélisation des compétences par des architectures de référence

Faire des raisonnements sur les modèles de compétence nécessite d'en assurer une certaine correction, par rapport à l'implantation et l'exécution réelle de ces compétences. Certaines de ces informations sont disponibles, à travers les modèles de composants et d'architectures logicielles, le temps de réponse des composants, ou leur accès à des ressources. Cette modélisation doit notamment reposer sur une connaissance experte du développeur ou de l'intégrateur quant aux composants et ressources qui permettent de réaliser ces compétences, donc sous la forme d'*architectures de référence*, comme proposé par GHERARDI et BRUGALI (2014). Ces modèles peuvent servir dans la fonction d'agissement, pour décider des modalités à utiliser pour remplir au mieux l'action à réaliser (en fonction des ressources utilisables, ou de critères de performance). Ces modèles peuvent également être utilisés dans une fonction de surveillance afin de reconfigurer l'architecture logicielle pour s'adapter à des pannes, en agissant sur l'architecture à la manière de TCL (LOTZ et al. 2013).

Il me semble intéressant de poursuivre ces travaux sur la modélisation de compétences par réseaux de Petri par la définition de modèles communs d'architectures logicielles et de compétences, de sorte à avoir un modèle unifié du logiciel exécuté sur le système robotique, des fonctionnalités ou services qu'il fournit, et de la façon dont ces services sont utilisés

par un processus d'agissement. Cette modélisation peut être envisagée en associant, à chaque compétence, une ou plusieurs architectures de référence, qui décrivent les différentes modalités possibles pour fournir cette compétence, donc intégrant différents composants, et utilisant différentes ressources. Au niveau de la surveillance de l'architecture, les actions de reconfiguration du type *activer un composant* ou *connecter un port* pourront alors être formellement définies à partir de ces modèles d'architectures.

Synthèse d'observateurs temps-réel depuis des spécifications logiques

Pour réaliser ces reconfigurations d'architecture, il est nécessaire de pouvoir *observer*, donc détecter des pannes ou des erreurs dans la réalisation des compétences. Dans les usages des développeurs d'architectures robotiques, ces observateurs sont principalement programmés (sans spécification), sous la forme de tests d'appartenance de certaines valeurs à des intervalles (valeurs minimales et maximales autorisées), comme en ont fait état Crestani, Godary-Dejean et Lapierre (2015). Plusieurs travaux récents proposent des langages de spécification centrés sur la fonction de surveillance, comme par exemple le langage DeRoS (Adam et al. 2016) pour spécifier des observateurs. DeRos permet de combiner ces tests avec des contraintes temporelles, et des règles de sécurité déclenchées par ces observateurs. Un exemple de règle est donné dans le listing 6.3 : le robot est arrêté (action `stop`) si son attitude dépasse des bornes autorisées (`max_tilt`), ou si sa vitesse dépasse la vitesse maximale autorisée pendant un temps donné.

Listing 6.3 – Spécification de règles en utilisant DeRoS, tirée de (Adam et al. 2016, p. 126).

```
use "RobotSafety.system"
import RobotSafety
action stop;
const max_tilt = toRad (10 deg)
const max_speed = 0.5 m/s
input orientation = topic /fmInformation/imu.orientation
input linear_speed = topic /fmKnowledge/encoder_odom.twist.twist.linear.x
entity imu_sensor_system
{
  tilt_not_ok :
    orientation.roll() not in [-max_tilt, max_tilt] or
      orientation.pitch() not in [-max_tilt, max_tilt] ;
}
entity drive_system {
  max_speed_exceeded : linear_speed > max_speed for 2 sec;
}
if imu_sensor_system.tilt_not_ok then { stop ; } ;
if drive_system.max_speed_exceeded for 0.4 sec then { stop ; } ;
```

Les durées de 2 secondes (spécifiée dans l'observateur) et de 0.4 secondes (spécifiée dans la règle) régissent le comportement de la fonction de surveillance du système, et donc il est impératif d'apporter des garanties quant à leur bonne exécution Or, sur cet exemple, on peut premièrement noter l'ambiguïté de la spécification (le temps au bout duquel l'action stop est déclenchée après que la vitesse est devenue trop importante est difficile à évaluer), mais surtout ce temps de réaction est extrêmement dépendant de l'implantation

de la fonction d'observation (et donc de l'évaluation des observateurs – `entity` dans la spécification DeRoS).

De plus, de part la complexité croissante des architectures logicielles en robotique, les tests d'intervalles ne sont pas suffisants pour décrire des observateurs complexes, notamment lorsqu'ils combinent des valeurs produites par différents composants de l'architecture, ou lorsque c'est la dynamique des valeurs prises qui est importante. La définition d'observateurs sur l'architecture logicielle nécessite un langage de description plus riche, permettant de représenter des dynamiques du système, et une implantation temps-réel plus rigoureuse. Pour aller dans ce sens, j'ai proposé, avec mes collègues, une spécification d'observateurs utilisant une logique temporelle linéaire étendue avec des opérateurs du passé (Past-time LTL, Emerson 1990), combinée avec une spécification de contraintes temporelles sur ces formules logiques. Ce travail a été soumis dans (Lesire et al. 2019). Le listing 6.4 donne un exemple de spécification combinant un observateur sur les valeurs d'un scan laser (utilisant de la logique propositionnelle), et un observateur sur la vitesse du robot (utilisant de la logique Past-time LTL). L'observateur de plus haut niveau décrit que le laser est détecté en panne si dans les dix dernières secondes le robot s'est déplacé, mais que le scan du laser est resté identique.

Listing 6.4 – Exemple de spécification proposé, combinant Past-time LTL et spécification temporisée

```
1 atomic-observer laserObs on laser {
2   fixed_scan : sameValuesThanPrev(laser.scan);
3 }
4 atomic-observer velocityObs on robot {
5   robot_moves : H(not isNull(robot.velocity), 100);
6 }
7 observer laserError on laserObs, velocityObs {
8   laser_error : (one robot_moves within 10 sec) and (all fixed_scan within 10 sec);
9 }
```

Ce travail préliminaire demande tout d'abord à être éprouvé sur des cas d'études complexes, de sorte à analyser la pertinence du langage proposé. L'enrichissement de ce langage pourrait se baser sur les schémas proposés par Autili et al. (2015), qui ont défini une logique intégrant des aspects qualitatifs, temporisés et probabilistes, et ont proposés des traductions vers des logiques temporelles. Utiliser un tel langage de spécification permettrait d'enrichir les constructions possibles pour décrire des observateurs (par exemple avec des chaînes de précédences temporisées, difficiles à écrire en LTL), et dispose de traductions vers des cadres de logique temporelle plus conventionnels pour réaliser des analyses. La synthèse d'observateurs ayant un comportement temps-réel intégré à l'architecture reste cependant encore ouverte.

Programmation de comportements par des opérateurs logiques

La programmation de comportements, voire de missions complètes, par un utilisateur, doit reposer sur un langage adapté, et sur une modélisation formelle, pour permettre

à la fois d'apporter des garanties sur le comportement spécifié en pouvant analyser les modèles, et aider à l'utilisabilité de ces modèles. Les travaux sur des langage de spécification ou de programmation ont majoritairement proposé une spécification par décomposition hiérarchique, basée sur des constructions programmatiques, de type conditions, répétition, séquences.

Dans (Lesire et Pommereau 2018), j'ai proposé des opérateurs logiques pour combiner les réseaux de Petri de flot de contrôle représentant les compétences du système. Ces opérateurs sont des constructions usuelles (séquence, parallélisme, choix, branchement, répétition), mais de part l'utilisation de jetons de contrôle, ils permettent de prendre en compte le succès ou l'échec des compétences (par exemple dans la condition de branchement). De plus, ces opérateurs apportent, pour la plupart, une conservation de bonnes propriétés (de type réseau bien formé) par composition. Pour l'utilisateur, l'utilisation de ces opérateurs permet de s'abstraire du modèle de réseau de Petri sous-jacent. Le résultat des compositions étant un réseau de Petri bien formé, il est cependant possible de mener des analyses sur ce réseau. L'utilisabilité de ce cadre de programmation de comportements passe par la définition d'un langage de spécification plus convivial, reposant sur des mots-clés et des constructions du type de ceux utilisés dans RPML (Williams et al. 2003) qui me semblent très adaptés aux différentes situations que j'ai pu rencontrer.

6.2 Décomposition hiérarchique de la décision

Les défis levés par l'utilisabilité de systèmes autonomes (mono- ou multi-robots) dans des contextes opérationnels sont d'une part de mettre en œuvre des architectures délibératives qui permettent de gérer des missions diverses et complexes, et d'autre part que ces architectures délibératives apportent des garanties sur les performances et la sécurité du système robotique, et soient "facilement" utilisables.

Des exemples d'utilisation complexe sont :

— des missions d'exploration dans des zones partiellement connues et peu accessibles (dont aux communications perturbées) ; par exemple pour l'exploration martienne, Ceballos et al. (2011) étudient une mission où il est demandé au système robotique d'aller observer des objets identifiés, tout en étant capable de réaliser des observations opportunistes, de communiquer ces observations dans des fenêtres de temps contraintes, en gérant les limitations d'énergie ;

— des missions de recherche et sauvetage après catastrophe, à l'image de la compétition RoboCup Rescue Simulation League (Kitano et Tadokoro 2001), où le système robotique doit éteindre des incendies, rechercher et secourir des blessés ; certains accès sont obstrués (donc à déblayer), les contraintes temporelles (pour assurer la survie des blessés et bâtiments) et de communication sont prégnantes ;

— des missions de robotique de service ou d'assistance ; Di Rocco, Pecora et Saffiotti (2013) traitent un problème où des robots de livraison et des robots d'assistance à domicile doivent se synchroniser pour réaliser leurs tâches, tout en répondant à des solicitations d'utilisateurs.

Ces exemples montrent la diversité des missions robotiques, où parfois le système à des tâches à réaliser dès le début de la mission (aller observer des objets), parfois les tâches arrivent de manière sporadique en fonction des observations (Parker et al. 2016 pour le problème de RoboCup Rescue). De plus, la mission globale est multi-objectifs (observer, rechercher, détecter, ...) et multi-critères (le plus vite possible, en sauvant le plus de personnes, ...).

6.2.1 De la hiérarchie dans les processus de planification

Comme nous l'avons vu et discuté dans le chapitre 5, le cadre HTN est un bon outil pour représenter, dans le problème de planification, des connaissances ou contraintes procédurales, dictées par un opérateur humain.

Ce type de raisonnement hiérarchique est très présent dans la littérature, pas seulement au niveau d'algorithmes de planification de tâche haut-niveau tel que HiPOP, pour des problèmes mono-robot ou multi-robots. Dans les systèmes d'enchères (Zlot et Stentz 2003), l'allocation de tâches est faite par le processus d'enchères, la planification des activités de chaque robot pour un processus spécifique. Les travaux sur la résolution conjointe de problèmes de planification symbolique et de mouvement mettent souvent en œuvre un processus hiérarchique, parfois même en utilisant le cadre HTN (Kaelbling et Lozano-Pérez 2011 ; de Silva, Pandey et Alami 2013). Bernardini, Fox et Long (2017) proposent une architecture délibérative pour une mission de recherche et sauvetage où certaines actions de recherche ont été construites par des procédures (avec des schémas de déplacement opérationnels). Dans un contexte multi-robots, quelques travaux se basent sur la notion de rôle, qui permet de lier des tâches entre elles dans une décomposition. Bowling, Browning et Veloso (2004) supervisent des stratégie multi-robots où chaque agent se voit attribuer un rôle, c'est-à-dire un plan à exécuter. Hunsberger et Grosz (2000) ont proposé un système d'enchères ou chaque agent peut enchérir sur un rôle, c'est à dire sur un ensemble de tâches structuré sous forme de plan partiel (au sens POP).

Ces différents travaux ont rarement étudié la bien fondé de cette décomposition hiérarchique des problèmes (excepté en planification conjointe tâches et mouvements, par exemple Bidot et al. 2017). Cependant, tous ces travaux mettent en avant le besoin de raisonner à des niveaux différents de décision pour constituer une architecture délibérative.

6.2.2 Architectures hiérarchisées

Au niveau architectural, les approches hiérarchiques sont utilisées dans deux contextes différents : la gestion d'échelles de temps différentes, ou la gestion de modèles ou algorithmes différents. Quelques exemples de travaux, qui ont inspiré les axes de recherche présentés dans les paragraphes suivants, sont décrits ci-dessous.

Chien et al. (1999) ont proposé une architecture de planification pour des missions de satellites, où le plan est laissé abstrait sur le long-terme, partiellement raffiné à moyen terme, et complètement raffiné à court-terme. Un principe similaire, utilisant le cadre HTN, est proposé par Kaelbling et Lozano-Pérez (2011) sous l'appellation *planning in the now*, où les actions sont raffinées au moment où elles vont être exécutées.

L'architecture T-REX (McGann et al. 2008), utilisée dans un contexte spatial dans l'architecture GOAC (Ceballos et al. 2011), décompose l'architecture en *réacteurs*. Chaque réacteur planifie et supervise son plan, en gérant l'évolution d'une partie du système. La structuration interne et les algorithmes de planification et de supervision ne sont pas spécifiés, mais les échanges d'information entre réacteurs sont formalisés à travers l'envoi de buts (*goals*) et de faits (*facts*) sur des chronogrammes (*timelines*).

Hanheide et al. (2017) proposent une architecture dont le cœur est un algorithme de planification déterministe, qui raisonne à partir d'hypothèses faites sur l'environnement. La réalisation de certaines tâches peut faire appel à la résolution (et l'exécution) d'un problème de décision séquentielle dans l'incertain (modélisé dans le cadre POMDP). Un module gère l'agrégation de connaissances de sorte à synthétiser ou mettre à jour les hypothèses courantes.

6.2.3 Axes de recherche et propositions

Les travaux présentés ci-dessus sur les architectures hiérarchisées n'ont pas été déployés dans des applications multi-robots. Dans l'architecture CASPER (Chien et al. 1999), le raffinement du plan (raffiné à court-terme, abstrait à long-terme) est géré par un seul algorithme de planification, même si la mise à jour de l'état, ou de la liste des tâches à raffiner, sont gérées par d'autres modules. Dans l'architecture T-REX (McGann et al. 2008), les échangent entre réacteurs sont spécifiés par des informations sur des timelines (buts ou faits). Cette formalisation des échanges est néanmoins limitée, car elle impose que les buts (envoyés entre modules) soit datés. De plus, l'architecture T-REX utilise en pratique un algorithme de planification commun à tous les réacteurs (EUROPA – Barreiro et al. 2012 – dans la version originale de McGann et al. 2008 ; APSI – Cesta et al. 2011 – dans l'architecture GOAC – Ceballos et al. 2011). Les applications multi-robots mettant en œuvre l'architecture T-REX l'ont fait de manière complètement distribuée, avec soit l'envoi de buts de navigation (Das et al. 2012), soit l'échange de cartes entre robots (Belbachir, Ingrand et Lacroix 2012).

Architecture délibérative d'acteurs hiérarchisés et spécialisés

Ma proposition est d'étendre les concepts développés dans les architectures hiérarchiques "à-la-T-REX", de sorte à mixer au sein d'un même architecture des *acteurs* (élément mis en avant comme central dans une architecture délibérative par Ghallab, Nau et Traverso 2014 – appelés *réacteurs* dans T-REX) qui gèrent planification et supervision de leur "niveau" de plan, notamment en mixant des algorithmes et formalismes différents, comme l'ont fait Hanheide et al. (2017) en combinant planification déterministe à haut-niveau et décision séquentielle dans l'incertain pour des tâches spécifiques. De plus, ce principe d'architecture peut s'appliquer autant à l'autonomie d'un robot, qu'à un système multi-robots. Ce concept architectural doit reposer sur une modélisation (et des algorithmes) permettant d'assurer une cohérence dans la conception de l'architecture, et d'apporter des garanties quant à son fonctionnement. Quelques perspectives sur des éléments de modélisation sont données ci-dessous.

Réseau de décision : un modèle pivot entre acteurs

Utiliser des algorithmes différents met en exergue le besoin de formaliser des modèles communs, ou tout du moins interconnectés, depuis la spécification de mission jusqu'à l'exécution. Ces modèles doivent aller au-delà de buts ou faits sur des chronogrammes. Wilkins et desJardins (2001) présentent différents applications réelles pour lesquelles des connaissances diverses doivent être prises en compte au niveau de l'architecture délibérative : des contraintes temporelles et d'utilisation des ressources, comme dans T-REX, mais également des procédures opérationnelles, des stratégies possibles, des façons de réagir à des événements, des préférences, la prise en compte explicite de l'interaction avec l'utilisateur, ...

Fratini, Pecora et Cesta (2008) proposent une architecture de planification permettant de raisonner sur des contraintes variées en basant leur architecture sur une modélisation par composantes (variables d'état, ressources) du système, et de la dynamique de ces composantes. La partie "planification" construit un *réseau de décision*, qui représente les choix faits par le planificateur (en terme d'affectation de valeur à des variables). Un module de gestion de ce réseau de décision assure la vérification de la cohérence du réseau proposé, sans quoi le planificateur doit revenir sur sa décision. Si l'instanciation de l'architecture est assez limitée (planification à base de chronogrammes, et propagation de contraintes pour la partie vérification ; pour résoudre un seul problème), l'idée de partager un réseau de décision entre acteurs est intéressante pour assurer une cohérence dans les décisions prises à différents niveaux. Ce concept est d'ailleurs assez proche de la façon dont HiPOP fonctionne (un algorithme de type A* pour la partie "décision", un algorithme de propagation STN pour la partie vérification des contraintes temporelles), ou de la façon dont la propagation sur un MaSTN fonctionne dans l'algo-

rithme DIP-M (présenté au paragraphe 5.2.2) : chaque agent "décide" des modifications locales de son STN (engagement d'actions ou observation des temps d'exécution), et met à jour un réseau, le macro-STN, sur lequel la faisabilité du plan est vérifiée (par chaque agent).

Un élément essentiel à définir pour assurer une formalisation d'une architecture d'acteurs est donc de définir un tel *réseau de décision*. La définition de ce réseau de décision est indispensable pour pouvoir qualifier le comportement global de l'architecture (en terme de preuve de fonctionnement correct). Il doit reposer sur des méthodes de propagation de contraintes pour assurer la cohérence des décisions prises à différents niveaux.

Dans un contexte multi-robots, où les acteurs sont répartis sur différents systèmes robotiques, les mécanismes de propagation ou vérification de ce réseau de décision doivent également permettre d'assurer une cohérence du réseau en présence de perturbations des communications (délais, perte de messages). Les travaux initiés dans le cadre des réseaux temporels simples sont une bonne base qu'il faut étendre à des contraintes plus générales.

Décomposition des buts et des tâches, et planification hybride

Nous avons vu dans le paragraphe 5.2.1 l'intérêt de représenter des procédures opérationnelles par des décompositions hiérarchiques, comme dans le cadre de modélisation HTN. Les approches les plus répandues de résolution des HTN (Nau et al. 1999 ; Nau et al. 2003) ont principalement considéré des décompositions de tâches assez simples (séquentielles ou non contraintes), et des algorithmes de recherche en avant, produisant des plans totalement ordonnés. Les travaux originels de Erol, Hendler et Nau (1994) considèrent cependant une méthode de décomposition comme un *réseau de tâches*, c'est-à-dire un ensemble de tâches et un ensemble de contraintes entre ces tâches, de type précédence, contraintes temporisées, contraintes sur des variables. Cette représentation est également adoptée par les algorithmes de planification hybride, car elle correspond à la définition d'un plan partiel dans le cadre POP, présenté dans le paragraphe 5.2.2.

Il me semble donc pertinent de considérer une représentation des objectifs de la mission sous forme de réseaux de tâches, utilisable dans un contexte de planification hybride. Alford et al. (2016) ont proposé un cadre plus général que les HTN, les Goal-Task Networks, qui mixent une décomposition de tâches et une décomposition de buts, plus adaptée à un cadre de planification *générative* (dans laquelle on ne fait pas que suivre la décomposition, mais on peut insérer des tâches). Ce cadre permet de capturer à la fois des notions de procédures par décomposition de tâches, et également des notions d'étapes dans la réalisation de la mission à travers la décomposition de buts.

Les travaux que je souhaite mener sur la planification "haut-niveau" dans une architecture délibérative portent donc sur :

— l'utilisation des informations de décomposition par des réseaux tâches-buts dans un cadre de planification hybride, cadre qui a montré son intérêt dans les scénarios du projet ACTION de part la flexibilité laissée dans le plan résultat, et son utilisation pour réparer des plans localement, notamment en présence de communications limitées ; de plus le cadre de la planification hybride est adapté pour expliquer un plan à un utilisateur (Seegebarth et al. 2012) car les justifications des éléments du plan sont riches (décompositions, liens causaux, liens temporels, résolution de défauts) ;

— l'étude des conditions et hypothèses qui permettent de ne décomposer un plan qu'à haut-niveau, en le laissant le plus abstrait possible ; Marthi, Russell et Wolfe (2007) ont proposé une sémantique *angélique* pour les méthodes d'un HTN, en encadrant les effets d'une méthode par des sur-ensembles et sous-ensembles d'effets, ce qui permet de savoir quelles tâches il est nécessaire de décomposer ; ils supposent ces ensembles définis dans le domaine, ce qui semble réalisable dans un cadre purement HTN (similaire à l'heuristique proposée par Bercher et al. 2017) ; dans le cas d'une architecture d'acteurs, calculer de tels ensembles est plus difficile. De plus, étudier ces conditions nécessitant une décomposition permettrait de formaliser les notions d'horizon de planification (défini dans CASPER – Chien et al. 1999 – et T-REX – McGann et al. 2008 – de manière empirique).

Supervision des acteurs par des réseaux de Petri de compétences

Comme nous l'avons vu dans les quelques exemples du paragraphe 6.2, les objectifs ne sont pas seulement des buts à atteindre, mais également des comportements à adopter, comme par exemple la réalisation récurrente de tâches ("débarrasser la table tant qu'il reste des couverts"), des réactions à événements ("si batterie critique, planifier un chemin de retour à la station"), ou des branches conditionnelles ("si le chemin est bloqué, déblayer, sinon, éteindre l'incendie"). Comme discuté dans le paragraphe 6.1.3, des constructeurs du type de ceux proposés dans RMPL (Williams et al. 2003) semblent adaptés à ce type de spécification. De plus, Wilkins et desJardins (2001), fondateurs des travaux en planification hiérarchique, argumentent pour l'utilisation de connaissances expertes et variées (hiérarchiques, temporelles, mais également procédurales ou sur les ressources) au niveau de la planification de mission, en lien avec un opérateur.

Dans une architecture délibérative d'acteurs, certains acteurs ont une fonction de *supervision* plutôt que de planification, et doivent donc représenter la prise de décision par des constructions "programmatiques", comme c'est le cas pour le processus d'*agissement*. Il me semble intéressant de faire ici le lien entre des fonctions de gestion de l'exécution de plan par la combinaison et réseaux de Petri de compétences par des opérateurs logiques, comme

proposé dans le paragraphe 6.1.3, et la gestion de l'exécution de prises de décision : si l'on considère qu'une compétence ne représente plus une fonction fournie par l'architecture logicielle du système robotique, mais représente une fonctionnalité du processus de décision (de l'acteur ou du réseau de décision), alors il serait possible d'écrire, en utilisant des constructions programmatiques, des stratégies de décision. Cette spécification de stratégies pourrait trouver des similarités avec les stratégies d'exécution présentées dans le cadre AMPLE (paragraphe 3.2.3), où les compétences fournies par le module de planification de l'architecture AMPLE sont l'ajout de requêtes de planification, l'accès à la politique, l'accès à des éléments du modèle POMDP, etc. (voir figure 3.2).

Ce travail pourrait trouver un lien avec la modélisation par réseaux de Petri de problèmes de planification "à-la-STRIPS", proposée par Costelha et Lima (2007). De plus, il permet de mettre en place des stratégies de raffinement contextualisées (dans lesquelles des décompositions de tâches ou buts sont différentes en fonction de l'état courant), un élément de représentation important également soulevé par Wilkins et desJardins.

Bibliographie

Adam, Sorin, Morten Larsen, Kjeld Jensen et Ulrik Pagh Schultz (2016), « Rule-based Dynamic Safety Monitoring for Mobile Robots », *in* : *Journal of Software Engineering for Robotics (JOSER)* 7.*1*, p. 120–141, URL : http://joser.unibg.it/index.php/joser/article/view/115 (cité page 74).

Ahma, Aakash et Muhammad Ali Babar (2016), « Software Architectures for Robotics Systems : A Systematic Mapping Study », *in* : *Journal of Systems and Software* 122, p. 16–39, DOI : 10.1016/j.jss.2016.08.039 (cité page 68).

Alford, Ron, Vikas Shivashankar, Mark Roberts, Jeremy Frank et David Aha (2016), « Hierarchical Planning : Relating Task and Goal Decomposition with Task Sharing », *in* : *International Joint Conference on Artificial Intelligence (IJCAI 2016)*, New York City, New York, USA, URL : http://www.ijcai.org/Abstract/16/429 (cité page 80).

Ando, Noriaki, Shinji Kurihara, Geoffrey Biggs, Takeshi Sakamoto, Hiroyuki Nakamoto et Tetsuo Kotoku (2011), « Software Deployment Infrastructure for Component Based RT-Systems », *in* : *Journal of Robotics and Mechatronics* 23.*3*, p. 87–98, DOI : 10.1007/978-3-540-89076-8_12 (cité page 41).

Autili, Marco, Lars Grunske, Markus Lumpe, Patrizio Pelliccione et Antony Tang (2015), « Aligning Qualitative, Real-Time, and Probabilistic Property Specification Patterns Using a Structured English Grammar », *in* : *IEEE Transactions on Software Engineering* 41.*7*, p. 620–638, DOI : 10.1109/TSE.2015.2398877 (cité page 75).

Banfi, Jacopo, Nicola Basilico et Stefano Carpin (2018), « Optimal Redeployment of Multirobot Teams for Communication Maintenance », *in* : *IEEE/RSJ Internatioal Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain (cité page 54).

Banfi, Jacopo, Alberto Quattrini Li, Ioannis Rekleitis, Francesco Amigoni et Nicola Basilico (2018), « Strategies for coordinated multirobot exploration with recurrent connectivity constraints », *in* : *Autonomous Robots* 42.*4*, p. 875–894, DOI : 10.1007/s10514-017-9652-y (cité page 54).

Barreiro, Javier, Matthew Boyce, Minh Do, Jeremy Frank, Michael Iatauro, Tatiana Kichkaylo, Paul Morris, James Ong, Emilio Remolina, Tristan Smith et David Smith (2012), « EUROPA : A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization. », *in* : *International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS)*, Sao Paulo, Brazil, URL : icaps12.icaps-conference.org/ickeps/ICKEPS2012-EUROPA.pdf (cité page 78).

Barto, Andrew, Steven Bradtke et Satinder Singh (1995), « Learning to act using real-time dynamic programming », *in* : *Artificial Intelligence* 72.*1-2*, p. 81–138, doi : 10.1016/0004-3702(94)00011-O (cité page 31).

Baruah, Sanjoy, Louis Rosier et Rodney Howell (1990), « Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor », *in* : *Real-Time Systems* 2, p. 301–324, doi : 10.1007/BF01995675 (cité page 45).

Basu, Ananda, Matthieu Gallien, Charles Lesire, Thanh-Hung Nguyen, Saddek Bensalem, Félix Ingrand et Joseph Sifakis (2008), « Incremental Component-Based Construction and Verification of a Robotic System », *in* : *European Conference on Artificial Intelligence (ECAI 2008)*, Patras, Greece, doi : 10.3233/978-1-58603-891-5-631 (cité page 41).

Bechon, Patrick, Magali Barbier, Christophe Grand, Simon Lacroix, Charles Lesire et Cédric Pralet (2018), « Integrating planning and execution for a team of heterogeneous robots with time and communication constraints », *in* : *IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, doi : 10.1109/ICRA.2018.8461024 (cité pages 60, 63, 65).

Bechon, Patrick, Magali Barbier, Guillaume Infantes, Charles Lesire et Vincent Vidal (2014), « HiPOP : Hierarchical Partial-Order Planning », *in* : *European Starting AI Researchers Symposium (STAIRS 2014)*, Prague, Czech Republic, doi : 10.3233/978-1-61499-421-3-51 (cité pages 58, 59).

Bechon, Patrick, Magali Barbier, Charles Lesire, Guillaume Infantes et Vincent Vidal (2015), « Using hybrid planning for plan reparation », *in* : *European Conference on Mobile Robots (ECMR 2015)*, Lincoln, United Kingdom, doi : 10.1109/ECMR.2015.7324201 (cité pages 63, 64).

Becker, Raphen, Shlomo Zilberstein, Victor Lesser et Claudia Goldman (2004), « Solving Transition Independent Decentralized Markov Decision Processes », *in* : *Journal of Artificial Intelligence Research (JAIR)* 22, doi : 10.1613/jair.1497 (cité page 52).

Belbachir, Assia, Félix Ingrand et Simon Lacroix (2012), « A cooperative architecture for target localization using multiple AUVs », *in* : *Intelligent Service Robotics* 5.*2*, p. 119–132, doi : 10.1007/s11370-012-0107-1 (cité pages 52, 78).

Bercher, Pascal, Gregor Behnke, Daniel Höller et Susanne Biundo (2017), « An Admissible HTN Planning Heuristic », *in* : *International Joint Conference on Artificial Intelligence (IJCAI 2017)*, Melbourne, Australia, doi : 10.24963/ijcai.2017/68 (cité page 81).

Bernardini, Sara, Maria Fox et Derek Long (2017), « Combining temporal planning with probabilistic reasoning for autonomous surveillance missions », *in* : *Autonomous Robots* 41.*1*, p. 181–203, doi : 10.1007/s10514-015-9534-0 (cité page 77).

Bernstein, Daniel, Robert Givan, Neil Immerman et Shlomo Zilberstein (2000), « The complexity of decentralized control of Markov decision processes », *in* : *Conference on Uncertainty in Artificial Intelligence (UAI 2000)*, Stanford, California, USA, URL : http://rbr.cs.umass.edu/shlomo/papers/BGIZmor02.pdf (cité page 52).

Bidot, Julien, Lars Karlsson, Fabien Lagriffoul et Alessandro Saffiotti (2017), « Geometric backtracking for combined task and motion planning in robotic systems », *in* : *Artificial Intelligence* 247, p. 229–265, DOI : 10.1016/j.artint.2015.03.005 (cité page 77).

Boerkoel, James, Léon Planken, Ronald Wilcox et Julie Shah (2013), « Distributed Algorithms for Incrementally Maintaining Multiagent Simple Temporal Networks », *in* : *International Conference on Automated Planning and Scheduling (ICAPS 2013)*, Rome, Italy, URL : http://www.aaai.org/ocs/index.php/ICAPS/ICAPS13/paper/view/5999 (cité page 60).

Bonet, Blai et Hector Geffner (2003), « Labeled RTDP : Improving the convergence of real-time dynamic programming », *in* : *International Conference on Automated Planning and Scheduling (ICAPS 2003)*, Trento, Italy, URL : http://ftp.cs.ucla.edu/pub/stat_ser/R319.pdf (cité page 30).

Boutilier, Craig (1996), « Planning, Learning and Coordination in Multiagent Decision Processes », *in* : *Conference on Theoretical Aspects of Rationality and Knowledge (TARK 1996)*, De Zeeuwse Stromen, The Netherlands, URL : https://www.cs.toronto.edu/~cebly/Papers/tark96.pdf (cité page 52).

Bowling, Michael, Brett Browning et Manuela Veloso (2004), « Plays as Effective Multiagent Plans Enabling Opponent-Adaptive Play Selection », *in* : *International Conference on Automated Planning and Scheduling (ICAPS 2004)*, Whistler, British Columbia, Canada, URL : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.520.414 (cité page 77).

Brafman, Ronen, Michael Bar-Sinai et Maor Ashkenazi (2016), « Performance level profiles : A formal language for describing the expected performance of functional modules », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)*, Daejeon, South Korea, DOI : 10.1109/IROS.2016.7759280 (cité pages 69, 72).

Bresina, John, Keith Golden, David Smith et Rich Washington (1999), « Increased Flexbility and Robustness of Mars Rovers », *in* : *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 1999)*, Noordwijk, The Netherlands, URL : https://ti.arc.nasa.gov/m/pub-archive/69h/0069%5C%20(Bresina).pdf (cité page 69).

Briñón-Arranz, Lara, Antonio Pascoal et Pedro Aguiar (2014), « Adaptive Leader-Follower Formation Control of Autonomous Marine Vehicles », *in* : *IEEE Conference*

on Decision and Control (CDC 2014), Los Angeles, California, USA, DOI : 10.1109/CDC.2014.7040222 (cité page 51).

BRUGALI, Davide et Patrizia SCANDURRA (2009), « Component-Based Robotic Engineering (Part I) », *in* : *IEEE Robotics and Automation Magazine* 16.*4*, p. 84–96, DOI : 10.1109/MRA.2009.934837 (cité page 39).

BRUGALI, Davide et Azamat SHAKHIMARDANOV (2010), « Component-Based Robotic Engineering (Part II) », *in* : *IEEE Robotics and Automation Magazine* 17.*1*, p. 100–112, DOI : 10.1109/MRA.2010.935798 (cité page 39).

BRUYNINCKX, Herman, Markus KLOTZBÜCHER, Nico HOCHGESCHWENDER, Gerhard KRAETZSCHMAR, Luca GHERARDI et Davide BRUGALI (2013), « The BRICS Component Model : A Model-Based Development Paradigm For Complex Robotics Software Systems », *in* : *ACM Symposium on Applied Computing (SAC 2013)*, Coimbra, Portugal, DOI : 10.1145/2480362.2480693 (cité page 41).

BURNS, Ethan, Wheeler RUML et Minh Binh DO (2013), « Heuristic Search When Time Matters », *in* : *Journal of Artificial Intelligence Research (JAIR)* 47, p. 697–740, DOI : 10.1613/jair.4047 (cité page 31).

CARRILLO, Henry, Philip DAMES, Vijay KUMAR et José CASTELLANOS (2015), « Autonomous robotic exploration using occupancy grid maps and graph SLAM based on Shannon and Rényi Entropy », *in* : *IEEE International Conference on Robotics and Automation (ICRA 2015)*, Seattle, Washington, USA, DOI : 10.1109/ICRA.2015.7139224 (cité page 29).

CASANOVA, Guillaume, Cédric PRALET et Charles LESIRE (2015), « Managing Dynamic Multi-Agent Simple Temporal Network », *in* : *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2015)*, Istanbul, Turkey, URL : https://www.onera.fr/sites/default/files/u518/aamas2015.pdf (cité pages 61, 62).

CASHMORE, Michael, Maria FOX, Derek LONG, Daniele MAGAZZENI, Bram RIDDER, Arnau CARRERAA, Narcis PALOMERAS, Natalia HURTOS et Marc CARRERASA (2015), « ROSPlan : Planning in the Robot Operating System », *in* : *International Conference on Automated Planning and Scheduling (ICAPS 2015)*, Jerusalem, Israel, URL : https://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10619/10379 (cité page 69).

CEBALLOS, Antonio, Saddek BENSALEM, Amedeo CESTA, Lavindra de SILVA, Simone FRATINI, Félix INGRAND, Jorge OCON, Andrea ORLANDINI, Frédéric PY, Kanna RAJAN, Riccardo RASCONI et Michel van WINNENDAEL (2011), « A Goal-Oriented Autonomous Controller for space exploration », *in* : *Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2011)*, Noordwijk, the Netherlands, URL : https://pdfs.semanticscholar.org/48d5/78b7dc5447645fdf2bcce0712dac6a5b86bb.pdf (cité pages 76, 78).

Cesta, Amedeo, Gabriella Cortellessa, Simone Fratini, Angelo Oddi et Giulio Bernardi (2011), « Deploying Interactive Mission Planning Tools - Experiences and Lessons Learned », *in* : *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII)* 15.*8*, p. 1149–1158, doi : 10.20965/jaciii.2011.p1149 (cité page 78).

Chien, Steve, Russell Knight, Andre Stechert, Rob Sherwood et Gregg Rabideau (1999), « Integrated Planning and Execution for Autonomous Spacecraft », *in* : *IEEE Aerospace Conference*, Aspen, Colorado, USA, doi : 10.1109/AERO.1999.794242 (cité pages 23, 78, 81).

Colledanchise, Michele et Lorenzo Natale (2018), « Improving the Parallel Execution of Behavior Trees », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018)*, Madrid, Spain (cité page 71).

Colledanchise, Michele et Petter Ögren (2017), « How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture, and Decision Trees », *in* : *IEEE Transactions on Robotics* 33.*2*, p. 372–389, doi : 10.1109/TRO.2016.2633567 (cité page 71).

Costelha, H. et P. Lima (2007), « Modelling, analysis and execution of robotic tasks using petri nets », *in* : *International Conference on Intelligent Robots and Systems (IROS)*, San Diego, California, USA, doi : 10.1109/IROS.2007.4399365 (cité page 82).

Crestani, Didier, Karen Godary-Dejean et Lionel Lapierre (2015), « Enhancing fault tolerance of autonomous mobile robots », *in* : *Robotics and Autonomous Systems* 68, p. 140–155, doi : 10.1016/j.robot.2014.12.015 (cité pages 39, 68, 74).

Cucu-Grosjean, Liliana, Luca Santinelli, Michael Houston, Code Lo, Tullio Vardanega, Leonidas Kosmidis, Jaume Abella, Enrico Mezzeti, Eduardo Quinones et Francisco Cazorla (2012), « Measurement-Based Probabilistic Timing Analysis for Multi-path Programs », *in* : *Euromicro Conference on Real-Time Systems (ECRTS 2012)*, Pisa, Italy, doi : 10.1109/ECRTS.2012.31 (cité page 45).

Das, Jnaneshwar, Frédéric Py, Thom Maughan, Tom O'Reilly, Monique Messié, John Ryan, Gaurav Sukhatme et Kanna Rajan (2012), « Coordinated sampling of dynamic oceanographic features with underwater vehicles and drifters », *in* : *The International Journal of Robotics Research (IJRR)* 31.*5*, p. 626–646, doi : 10.1177/0278364912440736 (cité page 78).

De Cubber, Geert, Daniela Doroftei, Daniel Serrano, Keshav Chintamani, Rui Sabino et Stephane Ourevitch (2013), « The EU-ICARUS project : Developing assistive robotic tools for search and rescue operations », *in* : *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR 2013)*, Linköping, Sweden, doi : 10.1109/SSRR.2013.6719323 (cité page 51).

de Silva, Lavindra, Amit Kumar Pandey et Rachid Alami (2013), « An interface for interleaved symbolic-geometric planning and backtracking », *in* : *IEEE/RSJ Interna-*

tional Conference on Intelligent Robots and Systems (IROS 2013), Tokyo, Japan, DOI : 10.1109/IROS.2013.6696358 (cité page 77).

DECHTER, Rina, Itay MEIRI et Judea PEARL (1991), « Temporal Constraint Networks », *in* : *Artificial Intelligence* 49.*1-3*, p. 61–95, DOI : 10.1016/0004-3702(91)90006-6 (cité page 60).

DI ROCCO, Maurizio, Federico PECORA et Alessandro SAFFIOTTI (2013), « When robots are late : Configuration planning for multiple robots with dynamic goals », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*, Tokyo, Japan, DOI : 10.1109/IROS.2013.6697214 (cité pages 54, 77).

DIAS, Bernardine, Robert ZLOT, Nidhi KALRA et Anthony STENTZ (2006), « Market-Based Multirobot Coordination : A Survey and Analysis », *in* : *Proceedings of the IEEE* 94.*7*, p. 1257–1270, DOI : 10.1109/JPROC.2006.876939 (cité page 52).

DOOSE, David, Christophe GRAND et Charles LESIRE (2017), « MAUVE Runtime : a component-based middleware to reconfigure software architectures in real-time », *in* : *Journal on Software Engineering for Robotics (JOSER)* 8.*1*, p. 128–140, URL : http://joser.unibg.it/index.php/joser/article/view/121 (cité page 48).

DVORÁK, Filip, Roman BARTÁK, Arthur BIT-MONNOT, Félix INGRAND et Malik GHALLAB (2014), « Planning and Acting with Temporal and Hierarchical Decomposition Models », *in* : *IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2014)*, Limassol, Cyprus, DOI : 10.1109/ICTAI.2014.27 (cité page 58).

EDWARDS, George, Joshua GARCIA, Hossein TAJALLI, Daniel POPESCU, Nenad MEDVIDOVIC, Gaurav SUKHATME et Brad PETRUS (2009), « Architecture-driven self-adaptation and self-management in robotics systems », *in* : *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2009)*, Vancouver, British Columbia, Canada, DOI : 10.1109/SEAMS.2009.5069083 (cité page 68).

EMERSON, E. Allen (1990), « Temporal and Modal Logic », *in* : *Handbook of Theoretical Computer Science, Volume B : Formal Models and Sematics (B)*, sous la dir. de Jan van LEEUWEN, Cambridge, MA, USA : MIT Press, p. 995–1072, URL : http://www.cs.utexas.edu/users/emerson/Pubs/handbook3.ps (cité page 75).

EROL, Kutluhan, James HENDLER et Dana NAU (1994), « HTN Planning : Complexity and Expressivity », *in* : *National Conference on Artificial Intelligence (AAAI 1994)*, Seattle, Washington, USA, URL : http://www.aaai.org/Library/AAAI/1994/aaai94-173.php (cité page 80).

FARGES, Jean-Loup, Guillaume INFANTES, Charles LESIRE et Augustin MANECY (2018), « Using POMDP with raw observations for detecting and recognizing objects of interest », *in* : *ICRA Workshop on Informative Path Planning and Adaptive Sampling (WIPPAS)*, Brisbane, Australia, URL : http://robotics.usc.edu/~wippas/papers/jean-loup.pdf (cité page 30).

Fazil Ayan, Necip, Ugur Kuter, Fusun Yaman et Robert Goldman (2007), « HOTRiDE : hierarchical ordered task replanning in dynamic environments », *in* : *ICAPS Workshop on Planning and Plan Execution for Real-World Systems*, Providence, Rhode Island, USA, URL : http://rpgoldman.goldman-tribe.org/papers/icaps07-hotride.pdf (cité page 54).

Foughali, Mohammed, Bernard Berthomieu, Silvano Dal Zilio, Félix Ingrand et Anthony Mallet (2016), « Model Checking Real-Time Properties on the Functional Layer of Autonomous Robots », *in* : *International Conference on Formal Engineering Methods (ICFEM 2016)*, Tokyo, Japan, DOI : 0.1007/978-3-319-47846-3_24 (cité page 41).

Fratini, Simone, Federico Pecora et Amedeo Cesta (2008), « Unifying planning and scheduling as timelines in a component-based perspective », *in* : *Archives of Control Science* 18.*2*, p. 231–271, URL : http://www.diva-portal.org/smash/record.jsf?pid=diva2:540638 (cité page 79).

Gancet, Jeremy, Gautier Hattenberger, Rached Alami et Simon Lacroix (2005), « Task Planning and control for a multi-AUV system : architecture and algorithms », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, Edmonton, Alberta, Canada, DOI : 10.1109/IROS.2005.1545217 (cité pages 54, 55).

Gat, Erann (1997), « ESL : A Language for Supporting Robust Plan Execution in Embedded Autonomous Agents », *in* : *IEEE Aerospace Conference*, Aspen, Colorado, USA, DOI : 10.1109/AERO.1997.574422 (cité page 69).

Gateau, Thibault, Charles Lesire et Magali Barbier (2013), « HiDDeN : Cooperative plan execution for heterogeneous robots in dynamic environments », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*, Tokyo, Japan, DOI : 10.1109/IROS.2013.6697047 (cité page 57).

Georgas, John et Richard Taylor (2008), « Policy-based Self-adaptive Architectures : A Feasibility Study in the Robotics Domain », *in* : *International Workshop on Software Engineering for Adaptive and Self-managing Systems (SEAMS 2008)*, Leipzig, Germany, DOI : 10.1145/1370018.1370038 (cité page 68).

Georgeff, Michael et Félix Ingrand (1989), « Decision-Making in an Embedded Reasoning System », *in* : *International Joint Conference on Artificial Intelligence (IJCAI 1989)*, Detroit, Michigan, USA, URL : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.127.7047&rep=rep1&type=pdf (cité page 69).

Ghallab, Malik, Dana Nau et Paolo Traverso (2014), « The actor's view of automated planning and acting : A position paper », *in* : *Artificial Intelligence* 208, p. 1–17, DOI : 10.1016/j.artint.2013.11.002 (cité pages 66, 79).

Gherardi, Luca et Davide Brugali (2014), « Modeling and Reusing Robotic Software Architectures : the HyperFlex Toolchain », *in* : *IEEE International Conference on*

Robotics and Automation (ICRA 2014), Hong Kong, China, DOI : 10.1109/ICRA.2014.6907806 (cité pages 68, 73).

GOBILLOT, Nicolas, David DOOSE, Charles LESIRE et Luca SANTINELLI (2015), « Periodic state-machine aware real-time analysis », *in* : *IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2015)*, Luxembourg, Luxembourg, DOI : 10.1109/ETFA.2015.7301403 (cité pages 41, 45).

GOBILLOT, Nicolas, Fabrice GUET, David DOOSE, Christophe GRAND, Charles LESIRE et Luca SANTINELLI (2016), « Measurement-Based Real-Time Analysis of Robotic Software Architectures », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, DOI : 10.1109/IROS.2016.7759509 (cité pages 45, 46).

GOBILLOT, Nicolas, Charles LESIRE et David DOOSE (2014), « A Modeling Framework for Software Architecture Specification and Validation », *in* : *International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR 2014)*, Bergamo, Italy, DOI : 10.1007/978-3-319-11900-7_26 (cité pages 41, 42).

GOBILLOT, Nicolas, Charles LESIRE et David DOOSE (2019), « A design and analysis methodology for component-based real-time architectures of autonomous systems », *in* : *Journal of Intelligent Robots and Systems (JINT)*, DOI : 10.1007/s10846-018-0967-5 (cité pages 40–44).

GOLDMAN, Claudia et Shlomo ZILBERSTEIN (2003), « Optimizing Information Exchange in Cooperative Multi-agent Systems », *in* : *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*, Melbourne, Australia, DOI : 10.1145/860575.860598 (cité page 53).

GOMBOLAY, Matthew, Ronald WILCOX et Julie SHAH (2018), « Fast Scheduling of Robot Teams Performing Tasks With Temporospatial Constraints », *in* : *IEEE Transactions on Robotics* 34.*1*, p. 220–239, DOI : 10.1109/TRO.2018.2795034 (cité page 54).

GOODFELLOW, Ian, Yoshua BENGIO et Aaron COURVILLE (2016), *Deep Learning*, MIT Press, URL : http://www.deeplearningbook.org (cité page 23).

GUET, Fabrice, Luca SANTINELLI et Jérôme MORIO (2016), « On the Reliability of the Probabilistic Worst-Case Execution Time Estimates », *in* : *Embedded Real-Time Software and Systems (ERTSS)*, Toulouse, France, URL : https://hal.archives-ouvertes.fr/hal-01289477/file/paper_16.pdf (cité page 45).

HANHEIDE, Marc, Moritz GÖBELBECKER, Graham HORN, Andrzej PRONOBIS, Kristoffer SJÖÖ, Alper AYDEMIR, Patric JENSFELT, Charles GRETTON, Richard DEARDEN, Miroslav JANICEK, Hendrik ZENDER, Geert-Jan KRUIJFF, Nick HAWES et Jeremy WYATT (2017), « Robot task planning and explanation in open and uncertain worlds », *in* : *Artificial Intelligence* 247, p. 119–150, DOI : 10.1016/j.artint.2015.08.008 (cité pages 78, 79).

HANSEN, Eric et Shlomo ZILBERSTEIN (2001), « LAO* : A heuristic search algorithm that finds solutions with loops », *in* : *Artificial Intelligence* 129.*1-2*, p. 35–62, DOI : 10.1016/S0004-3702(01)00106-0 (cité page 30).

HEINZEMANN, Christian et Ralph LANGE (2018), « vTSL-A Formally Verifiable DSL for Specifying Robot Tasks », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018)*, Madrid, Spain (cité page 71).

HSIAO, Kaijen, Leslie KAELBLING et Tomas LOZANO-PEREZ (2007), « Grasping POMDPs », *in* : *IEEE International Conference on Robotics and Automation (ICRA 2007)*, Roma, Italy, DOI : 10.1109/ROBOT.2007.364201 (cité page 30).

HUNSBERGER, Luke et Barbara GROSZ (2000), « A combinatorial auction for collaborative planning », *in* : *International Conference on MultiAgent Systems (ICMAS 2000)*, Boston, Massachussets, USA, DOI : 10.1109/ICMAS.2000.858447 (cité page 77).

INGHAM, Michel, Robert RAGNO et Brian WILLIAMS (2001), « A Reactive Model-based Programming Language for Robotic Space Explorers », *in* : *International Symposium on Artificial Intelligence, Robotics and Automation in Space (ISAIRAS 2001)*, Montréal, Canada, URL : http://www.ai.mit.edu/people/williams/papers/isairas01_rmpl.pdf (cité page 71).

INGRAND, Felix, Raja CHATILA, Rachid ALAMI et Frederic ROBERT (1996), « PRS : A high Level Supervision and Control Language for Autonomous Mobile Robots », *in* : *IEEE International Conference on Robotics and Automation (ICRA 1996)*, Minneapolis, Minnesota, USA, DOI : 10.1109/ROBOT.1996.503571 (cité page 69).

INGRAND, Félix et Malik GHALLAB (2017), « Deliberation for Autonomous Robots : A Survey », *in* : *Artificial Intelligence* 247, p. 10–44, URL : https://doi.org/10.1016/j.artint.2014.11.003 (cités pages 23, 24, 66).

JUNG, Min Yang, Marcin BALICKI, Russell TAYLOR et Peter KAZANZIDES (2013), « Lessons Learned from the Development of Component-Based Medical Robot Systems », *in* : *Journal of Software Engineering for Robotics (JOSER)* 5.*2*, p. 25–41, URL : http://joser.unibg.it/index.php/joser/article/view/75/28 (cité page 41).

KAELBLING, Leslie, Michael LITTMAN et Anthony CASSANDRA (1998), « Planning and acting in partially observable stochastic domains », *in* : *Artificial Intelligence* 101.*1*, p. 99–134, DOI : 10.1016/S0004-3702(98)00023-X (cités pages 23, 29).

KAELBLING, Leslie et Tomas LOZANO-PÉREZ (2011), « Hierarchical task and motion planning in the now », *in* : *IEEE International Conference on Robotics and Automation (ICRA 2011)*, Shangai, China, DOI : 10.1109/ICRA.2011.5980391 (cités pages 77, 78).

KAMBHAMPATI, Subbarao, Amol MALI et Biplav SRIVASTAVA (1998), « Hybrid planning for partially hierarchical domains », *in* : *AAAI/IAAI National Conference on Artificial Intelligence (AAAI 1998)*, Madison, Wisconsin, USA, URL : https://www.aaai.org/Papers/AAAI/1998/AAAI98-125.pdf (cité page 58).

KANTAROS, Yiannis et Michael ZAVLANOS (2018), « Distributed Intermittent Communication Control of Mobile Robot Networks under Time-Critical Dynamic Tasks », *in* : *IEEE International Conference on Robotics and Automation (ICRA 2018)*, Brisbane, Australia, DOI : 10.1109/ICRA.2018.8460570 (cité page 54).

KELLER, Thomas et Patrick EYERICH (2012), « PROST : Probabilistic Planning Based on UCT », *in* : *International Conference on Automated Planning and Scheduling (ICAPS 2012)*, Atibaia, Sao Paulo, Brazil, URL : http://gki.informatik.uni-freiburg.de/papers/keller-eyerich-icaps2012.pdf (cité page 31).

KEPHART, Jeffrey et David CHESS (2003), « The vision of autonomic computing », *in* : *Computer* 36.*1*, p. 41–50, DOI : 10.1109/MC.2003.1160055 (cité page 68).

KHALASTCHI, Eliahu et Meir KALECH (2018), « On Fault Detection and Diagnosis in Robotic Systems », *in* : *ACM Computing Surveys* 51.*1*, DOI : 10.1145/3146389 (cité page 39).

KIM, Dongsun et Sooyong PARK (2006), « Designing Dynamic Software Architecture for Home Service Robot Software », *in* : *International Conference on Embedded and Ubiquitous Computing (EUC 2006)*, Seoul, South Korea, URL : www.darkrsw.net/papers/EUC2006.pdf (cité page 68).

KITANO, Hiroaki et Satoshi TADOKORO (2001), « RoboCup Rescue : A Grand Challenge for Multiagent and Intelligent Systems », *in* : *AI Magazine* 22.*1*, DOI : 10.1609/aimag.v22i1.1542 (cité page 76).

KORF, Richard (1990), « Real-Time Heuristic Search », *in* : *Artificial Intelligence* 42.*2–3*, p. 189–211, DOI : 10.1016/0004-3702(90)90054-4 (cité page 31).

KURNIAWATI, Hanna, David HSU et Wee Sun LEE (2008), « SARSOP : Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces », *in* : *Conference on Robotics : Science and Systems (RSS 2008)*, Zurich, Switzerland, DOI : 10.15607/RSS.2008.IV.009 (cité page 30).

KWIATKOWSKA, Marta, Gethin NORMAN et David PARKER (2002), « Probabilistic Symbolic Model Checking with PRISM : A Hybrid Approach », *in* : *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2002)*, Grenoble, France, DOI : 10.1007/3-540-46002-0_5 (cité page 37).

LAGOUDAKIS, Michail, Evangelos MARKAKIS, David KEMPE, Pinar KESKINOCAK, Anton KLEYWEGT, Sven KOENIG, Craig TOVEY, Adam MEYERSON et Sonal JAIN (2005), « Auction-Based Multi-Robot Routing », *in* : *Conference on Robotics : Science and Systems (RSS 2005)*, Cambridge, Massachusetts, USA, URL : http://www.roboticsproceedings.org/rss01/p45.html (cité page 52).

LESIRE, Charles, David DOOSE et Hugues CASSÉ (2012), « Mauve : a Component-based Modeling Framework for Real-Time Analysis of Robotic Applications », *in* : *ICRA Workshop on Software Development and Integration for Robotics (SDIR)*, Saint-Paul,

Minnesota, USA, URL : https://hal.archives-ouvertes.fr/hal-01060327 (cité page 45).

Lesire, Charles, Guillaume Infantes, Thibault Gateau et Magali Barbier (2016), « A distributed architecture for supervision of autonomous multi-robot missions - Application to air-sea scenarios », *in* : *Autonomous Robots* 40.7, p. 1343–1362, DOI : 10.1007/s10514-016-9603-z (cité pages 56, 57).

Lesire, Charles et Franck Pommereau (2018), « ASPiC : an Acting system based on Skill Petri net Composition », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018)*, Madrid, Spain, DOI : 10.1109/IROS.2018.8594328 (cité pages 49, 72, 73, 76).

Lesire, Charles, Stéphanie Roussel, David Doose et Christophe Grand (2019), « Synthesis of Real-Time Observers from Past-Time Linear Temporal Logic and Timed Specification », *in* : *IEEE International Conference on Robotics and Automation (ICRA)*, Montreal, Canada (cité pages 49, 75).

Lima, Keila, Eduardo R. B. Marques, José Pinto et João B. Sousa (2018), « Dolphin : a task orchestration language for autonomous vehicle networks », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018)*, Madrid, Spain (cité page 71).

Lin, Christopher, Andrey Kolobov, Ece Kamar et Eric Horvitz (2015), « Metareasoning for Planning Under Uncertainty », *in* : *International Joint Conference on Artificial Intelligence (IJCAI 2015)*, Buenos Aires, Argentina, URL : http://christopherhlin.com/papers/ijcai2015.pdf (cité page 31).

Liu, Chang et James Layland (1973), « Scheduling Algorithms for Multiprogramming in a Hard- Real-Time Environment », *in* : *Journal of the Association for Computing Machinery (JACM)* 20.1, p. 46–61, DOI : 10.1145/321738.321743 (cité pages 44, 45).

Lotz, Alex, Juan Inglés-Romero, Cristina Vicente-Chicote et Christian Schlegel (2013), « Managing Run-Time Variability in Robotics Software by Modeling Functional and Non-functional Behavior », *in* : *International Conference on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD 2013)*, Valencia, Spain, DOI : 10.1007/978-3-642-38484-4_31 (cité pages 69, 73).

Makarenko, Alexei, Stefan Williams, Frederic Bourgault et Hugh Durrant-Whyte (2002), « An experiment in integrated exploration », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*, Lausanne, Switzerland, DOI : 10.1109/IRDS.2002.1041445 (cité page 29).

Mallet, Anthony, Cédric Pasteur et Matthieu Herrb (2010), « GenoM3 : Building middleware-independent robotic components », *in* : *IEEE International Conference on Robotics and Automation (ICRA 2010)*, Anchorage, Alaska, USA, DOI : 10.1109/ROBOT.2010.5509539 (cité page 41).

Manso, Luis, Pilar Bachiller, Pablo Bustos, Pedro Nunez, Ramon Cintas et Luis Calderita (2010), « RoboComp : A Tool-Based Robotics Framework », *in* : *International Conference on Simulation, Modelling and Programming for Autonomous Robots (SIMPAR 2010)*, Darmstadt, Germany, doi : 10.1007/978-3-642-17319-6_25 (cité page 41).

Marconi, Lorenzo, Claudio Melchiorri, Michael Beetz, Dejan Pangercic, Roland Siegwart, Stefan Leutenegger, Raffaella Carloni, Stefano Stramigioli, Herman Bruyninckx, Patrick Doherty, Alexander Kleiner, Vincenzo Lippiello, Alberto Finzi, Bruno Siciliano, Andrea Sala et Nicola Tomatis (2012), « The SHERPA project : smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments », *in* : *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, College Station, Texas, USA, doi : 10.1109/SSRR.2012.6523905 (cité page 51).

Marthi, Bhaskara, Stuart Russell et Jason Wolfe (2007), « Angelic Semantics for High-Level Actions », *in* : *International Conference on Automated Planning and Scheduling (ICAPS 2007)*, Providence, Rhode Island, USA, url : https://people.eecs.berkeley.edu/~russell/papers/icaps07-hla.pdf (cité page 81).

Martínez, Jesús, Adrián Romero-Garcés, Luis Manso et Pablo Bustos (2010), « Improving a robotics framework with real-time and high-performance features », *in* : *International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR 2010)*, Darmstadt, Germany, doi : 10.1007/978-3-642-17319-6_26 (cité page 41).

McGann, Conor, Frederic Py, Kanna Rajan, Hans Thomas, Richard Henthorn et Robert McEwen (2008), « A deliberative architecture for AUV control », *in* : *IEEE International Conference on Robotics and Automation (ICRA 2008)*, Pasadena, California, USA, doi : 10.1109/ROBOT.2008.4543343 (cité pages 52, 78, 81).

Melo, Francisco et Manuela Veloso (2011), « Decentralized MDPs with sparse interactions », *in* : *Artificial Intelligence* 175.*11*, p. 1757–1789, doi : 10.1016/j.artint.2011.05.001 (cité page 53).

Murata, Takeshi (1989), « Petri nets : Properties, analysis and applications », *in* : *Proceedings of the IEEE* 77.*4*, p. 541–580, doi : 10.1109/5.24143 (cité page 72).

Nau, Dana, Tsz-Chiu Au, Okhtay Ilhami, Ugur Kuter, William Murdock, Dan Wu et Fusun Yaman (2003), « SHOP2 : an HTN planning system », *in* : *Journal of Artificial Intelligence Research (JAIR)* 20.*1*, p. 379–404, url : https://arxiv.org/pdf/1106.4869.pdf (cité pages 54, 56, 80).

Nau, Dana, Yue Cao, Amnon Lotem et Hector Munoz-Avila (1999), « SHOP : Simple hierarchical ordered planner », *in* : *International Joint Conference on Artificial Intelligence (IJCAI 1999)*, Stockholm, Sweden, url : http://dl.acm.org/citation.cfm?id=1624312.1624357 (cité pages 54, 56, 80).

Nau, Dana, Malik Ghallab et Paolo Traverso (2015), « Blended Planning and Acting : Preliminary Approach, Research Challenges », *in* : *AAAI Conference on Artificial Intelligence (AAAI 2015)*, Austin, Texas, USA, URL : http://www.cs.umd.edu/~nau/papers/nau2015blended.pdf (cité page 31).

Navet, Nicolas et Loïc Fejoz (2016), « CPAL : High-level Abstractions for Safe Embedded Systems », *in* : *International Workshop on Domain-Specific Modeling (DSM 2016)*, New York City, New York, USA, DOI : 10.1145/3023147.3023153 (cité page 41).

Nilsson, Nils (1969), « A Mobile Automaton : An Application of Artificial Intelligence Techniques », *in* : *International Joint Conference on Artificial Intelligence (IJCAI 1969)*, Washington, DC, USA, URL : http://ijcai.org/Proceedings/69/Papers/046.pdf (cité page 23).

Noreils, Fabrice (1990), « Integrating error recovery in a mobile robot control system », *in* : *IEEE International Conference on Robotics and Automation (ICRA 1990)*, Cincinnati, Ohio, USA, DOI : 10.1109/ROBOT.1990.126008 (cité page 69).

Ong, Sylvie, Shao Wei Png, David Hsu et Wee Sun Lee (2010), « Planning under Uncertainty for Robotic Tasks with Mixed Observability », *in* : *The International Journal of Robotics Research (IJRR)* 29.*8*, p. 1053–1068, DOI : 10.1177/0278364910369861 (cité page 30).

Otte, Michael, Michael Kuhlman et Donald Sofge (2017), « Multi-Robot Task Allocation with Auctions in Harsh Communication Environments », *in* : *Internatioal Symposium on Multi-Robot and Multi-Agent Systems (MRS 2017)*, Los Angeles, California, USA, DOI : 10.1109/MRS.2017.8250928 (cité page 52).

Paolucci, Massimo, Onn Shehory et Katia Sycara (2000), « Interleaving planning and execution in a multiagent team planning environment », *in* : *Linköping Electronic Articles in Computer and Information Sciences* 5.*18*, URL : https://www.cs.cmu.edu/~softagents/papers/etai.ps.gz (cité pages 54, 55).

Parker, James, Ernesto Nunes, Julio Godoy et Maria Gini (2016), « Exploiting Spatial Locality and Heterogeneity of Agents for Search and Rescue Teamwork », *in* : *Journal of Field Robotics* 33.*7*, p. 877–900, DOI : 10.1002/rob.21601 (cité page 77).

Parker, Lynne (2008), « Distributed Intelligence : Overview of the Field and its Application in Multi-Robot Systems », *in* : *Journal of Physical Agents* 2.*2*, p. 5–14, URL : http://www.jopha.net/index.php/jopha/article/view/18/14 (cité page 51).

Penberthy, Scott et Daniel Weld (1992), « UCPOP : A sound, complete, partial order planner for ADL », *in* : *International Conference on Principles of Knowledge Representation and Reasoning (KR 1992)*, Cambridge, Massachusetts, USA, URL : http://ai.unibo.it/system/files/u11/ucpop-kr92.pdf (cité page 57).

Petri, Carl Adam (1962), *Kommunikation mit Automaten*, Dissertation, Schriften des IIM 2, Bonn : Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn (cité page 72).

Pineau, Joelle, Geoffrey Gordon et Sebastian Thrun (2006), « Anytime Point-Based Approximations for Large POMDPs », *in* : *Jounal of Artificial Intelligence Research (JAIR)* 27, p. 335–380, doi : 10.1613/jair.2078 (cité pages 30, 31).

Ponzoni Carvalho Chanel, Caroline, Alexandre Albore, Jorrit T'Hooft, Charles Lesire et Florent Teichteil-Königsbuch (2019), « AMPLE : an anytime planning and execution framework for dynamic and uncertain problems in robotics », *in* : *Autonomous Robots* 43.*1*, p. 37–62, doi : 10.1007/s10514-018-9703-z (cité pages 32, 35, 37).

Ponzoni Carvalho Chanel, Caroline, Charles Lesire et Florent Teichteil-Königsbuch (2014), « A Robotic Execution Framework for Online Probabilistic (Re)Planning », *in* : *International Conference on Automated Planning and Scheduling (ICAPS 2014)*, Portsmouth, New Hampshire, USA, url : http://www.aaai.org/ocs/index.php/ICAPS/ICAPS14/paper/view/7928 (cité page 35).

Ponzoni Carvalho Chanel, Caroline, Florent Teichteil-Königsbuch et Charles Lesire (2013), « Multi-Target Detection and Recognition by UAVs Using Online POMDPs », *in* : *AAAI Conference on Artificial Intelligence (AAAI 2013)*, Bellevue, Washington, USA, url : http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6435 (cité page 33).

Popescu, Mihai-Ioan, Hervé Rivano et Olivier Simonin (2016), « Multi-robot Patrolling in Wireless Sensor Networks using Bounded Cycle Coverage », *in* : *IEEE Internatioal Conference on Tools with Artificial Intelligence (ICTAI)*, San Jose, CA, USA, doi : 10.1109/ICTAI.2016.32 (cité page 55).

Pralet, Cédric et Charles Lesire (2014), « Deployment of Mobile Wireless Sensor Networks for Crisis Management : A Constraint-Based Local Search Approach », *in* : *International Conference on Principles and Practice of Constraint Programming (CP 2004)*, Lyon, France, doi : 10.1007/978-3-319-10428-7_62 (cité page 51).

Puterman, Martin (1994), *Markov Decision Processes : Discrete Stochastic Dynamic Programming*, New York, NY, USA : John Wiley & Sons, Inc., doi : 10.1002/9780470316887 (cité page 29).

Quigley, Morgan, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler et Andrew Mg (2009), « ROS : an open-source Robot Operating System », *in* : *ICRA Workshop on Open Source Software*, Kobe, Japan, url : www.willowgarage.com/sites/default/files/icraoss09-ROS.pdf (cité pages 39, 41).

Rock (p.d.), *Rock, the Robot Construction Kit*, url : http://www.rock-robotics.org/ (cité page 41).

Ross, Stéphane et Brahim Chaib-Draa (2007), « AEMS : An anytime online search algorithm for approximate policy refinement in large POMDPs », *in* : *International*

Joint Conference on Artificial Intelligence (IJCAI 2007), Hyderabad, India, URL : https://www.ijcai.org/Proceedings/07/Papers/417.pdf (cité pages 30, 33).

Ross, Stéphane, Joelle Pineau, Sébastien Paquet et Brahim Chaib-Draa (2008), « Online planning algorithms for POMDPs », *in* : *Journal of Artificial Intelligence Research (JAIR)* 32.*1*, p. 663–704, URL : https://www.aaai.org/Papers/JAIR/Vol32/JAIR-3217.pdf (cité page 31).

Santos, Maria et Magnus Egerstedt (2018), « Coverage Control for Multi-Robot Teams with Heterogeneous Sensing Capabilities Using Limited Communications », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain (cité page 52).

Schattenberg, Bernd (2009), « Hybrid Planning And Scheduling », thèse de doct., Ulm University, Institute of Artificial Intelligence, DOI : 10.1007/s13218-015-0390-z (cité page 58).

Schillinger, Philipp, Mathias Bürger et Dimos Dimarogonas (2018a), « Auctioning over Probabilistic Options for Temporal Logic-Based Multi-Robot Cooperation under Uncertainty », *in* : *IEEE International Conference on Robotics and Automation (ICRA 2018)*, Brisbane, Australia, DOI : 10.1109/ICRA.2018.8462967 (cité page 55).

Schillinger, Philipp, Mathias Bürger et Dimos Dimarogonas (2018b), « Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems », *in* : *The International Journal of Robotics Research (IJRR)* 37.*7*, p. 818–838, DOI : 10.1177/0278364918774135 (cité page 54).

Schlegel, Christian (2006), « Communication Patterns as Key Towards Component-Based Robotics », *in* : *Journal of Advanced Robotic Systems* 3.*1*, p. 49–54, DOI : 10.5772/5759 (cité page 41).

Schlegel, Christian, Andreas Steck, Davide Brugali et Alois Knoll (2010), « Design Abstraction and Processes in Robotics : From Code-Driven to Model-Driven Engineering », *in* : *International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR 2010)*, Darmstadt, Germany, DOI : 10.1007/978-3-642-17319-6_31 (cité pages 41, 68).

Seegebarth, Bastian, Felix Müller, Bernd Schattenberg et Susanne Biundo (2012), « Making Hybrid Plans More Clear to Human Users — a Formal Approach for Generating Sound Explanations », *in* : *International Conference on Automated Planning and Scheduling (ICAPS 2012)*, Atibaia, São Paulo, Brazil, URL : https://pdfs.semanticscholar.org/552d/c4b225983c166e4a5c6e1250f399a9797fc2.pdf (cité page 81).

Simmons, Reid et David Apfelbaum (1998), « A Task Description Language for Robot Control », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 1998)*, Victoria, British Columbia, Canada, DOI : 10.1109/IROS.1998.724883 (cité page 69).

Simmons, Reid, J. Fernandez, Richard Goodwin, Sven Koenig et Joseph O'Sullivan (2000), « Lessons learned from Xavier », *in* : *Robotics and Automation Magazine* 7.*2*, p. 33–39, URL : https://doi.org/10.1109/100.848266 (cité page 23).

Smith, Reid (1980), « The Contract Net Protocol : high-level communication and control in a distributed problem solver », *in* : *IEEE Transactions on Computers* 29.*12*, p. 1104–1113, DOI : 10.1109/TC.1980.1675516 (cité page 52).

Soetens, Peter et Herman Bruyninckx (2005), « Realtime Hybrid Task-Based Control for Robots and Machine Tools », *in* : *IEEE International Conference on Robotics and Automation (ICRA 2005)*, Barcelona, Spain, DOI : 10.1109/ROBOT.2005.1570129 (cité pages 39, 41).

Steck, Andreas, Alex Lotz et Christian Schlegel (2011), « Model-driven engineering and run-time model-usage in service robotics », *in* : *International Conference on Generative Programming and Component Engineering (GPCE 2011)*, Portland, Oregon, USA, DOI : 10.1145/2047862.2047875 (cité page 71).

Steck, Andreas et Christian Schlegel (2010), « Towards quality of service and resource aware robotic systems through model-driven software development », *in* : *International Workshop on Domain-Specific Languages and models for Robotic systems (DSLRob 2010)*, Taipei, Taiwan, URL : http://arxiv.org/abs/1009.4877 (cité page 41).

Stigge, Martin, Pontus Ekberg, Nan Guan et Wang Yi (2011), « The Digraph Real-Time Task Model », *in* : *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2011)*, Chicago, Ilinois, USA, DOI : 10.1109/RTAS.2011.15 (cité page 44).

Stock, Sebastian, Masoumeh Mansouri, Federico Pecora et Joachim Hertzberg (2015), « Online task merging with a hierarchical hybrid task planner for mobile service robots », *in* : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015)*, Hamburg, Germany, DOI : 10.1109/IROS.2015.7354300 (cité page 58).

Sykes, Daniel, William Heaven, Jeff Magee et Jeff Kramer (2008), « From Goals to Components : A Combined Approach to Self-management », *in* : *International Workshop on Software Engineering for Adaptive and Self-managing Systems (SEAMS 2008)*, Leipzig, Germany, DOI : 10.1145/1370018.1370020 (cité page 68).

Tajalli, Hossein, Joshua Garcia, George Edwards et Nenad Medvidovic (2010), « PLASMA : A Plan-based Layered Architecture for Software Model-driven Adaptation », *in* : *IEEE/ACM International Conference on Automated Software Engineering (ASE 2010)*, Antwerp, Belgium, DOI : 10.1145/1858996.1859092 (cité page 68).

Tchidjo Moyo, Noel, Eric Nicollet, Frederic Lafaye et Christophe Moy (2010), « On Schedulability Analysis of Non-cyclic Generalized Multiframe Tasks », *in* : *Euromicro Conference on Real-Time Systems (ECRTS 2010)*, Brussels, Belgium, DOI : 10.1109/ECRTS.2010.24 (cité page 44).

TEICHTEIL-KÖNIGSBUCH, Florent, Ugur KUTER et Guillame INFANTES (2010), « Incremental plan aggregation for generating policies in MDPs », *in* : *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2010)*, Toronto, Canada, URL : https://pdfs.semanticscholar.org/7bba/4a9d596b0ee2db566a8e407ff9acb1261061.pdf (cité page 30).

TEICHTEIL-KÖNIGSBUCH, Florent, Charles LESIRE et Guillaume INFANTES (2011), « A generic framework for anytime execution-driven planning in robotics », *in* : *IEEE International Conference on Robotics and Automation (ICRA 2011)*, Shangai, China, DOI : 10.1109/ICRA.2011.5980289 (cité page 33).

T'HOOFT, Jorrit, Charles LESIRE et Caroline PONZONI CARVALHO CHANEL (2016), « Online Proactive Planning with Multiple Hypotheses », *in* : *European Starting AI Researchers Symposium (STAIRS 2016)*, The Hague, Netherlands, DOI : 10.3233/978-1-61499-682-8-123 (cité page 35).

UNHELKAR, Vaibhav et Julie SHAH (2016), « ConTaCT : Deciding to Communicate during Time-Critical Collaborative Tasks in Unknown, Deterministic Domains », *in* : *AAAI Conference on Artificial Intelligence (AAAI 16)*, Phoenix, Arizona, USA, URL : http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11913 (cité page 53).

VERMA, Vandi, Tara ESTLIN, Ari JONSSON, Corina PASAREANU, Reid SIMMONS et Kam TSO (2005), « Plan Execution Interchange Language (PLEXIL) for Executable Plans and Command Sequences », *in* : *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2005)*, Munich, Germany, URL : https://www.cs.cmu.edu/~reids/papers/exec_isairas05.pdf (cité page 69).

WILHELM, Reinhard, Jakob ENGBLOM, Andreas ERMEDAHL, Niklas HOLSTI, Stephan THESING, David WHALLEY, Guillem BERNAT, Christian FERDINAND, Reinhold HECKMANN, Tulika MITRA, Frank MUELLER, Isabelle PUAUT, Peter PUSCHNER, Jan STASCHULAT et Per STENSTRÖM (2008), « The worst-case execution-time problem - overview of methods and survey of tools », *in* : *ACM Transactions on Embedded Computing Systems (TECS)* 7.*3*, p. 1–53, DOI : 10.1145/1347375.1347389 (cité page 45).

WILKINS, David et Marie DESJARDINS (2001), « A Call for Knowledge-Based Planning », *in* : *AI Magazine* 22.*1*, p. 99–115, DOI : 10.1609/aimag.v22i1.1547 (cité pages 79, 81, 82).

WILLIAMS, Brian, Michel INGHAM, Seung CHUNG et Paul ELLIOTT (2003), « Model-based programming of intelligent embedded systems and robotic space explorers », *in* : *Proceedings of the IEEE* 91.*1*, p. 212–237, DOI : 10.1109/JPROC.2002.805828 (cité pages 71, 72, 76, 81).

WILLIAMSON, Simon, Enrico GERDING et Nicholas JENNINGS (2009), « Reward Shaping for Valuing Communications During Multi-agent Coordination », *in* : *International*

Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, URL : https://eprints.soton.ac.uk/267076/1/rewardShaping.pdf (cité page 53).

YOUNES, Håkan et Reid SIMMONS (2003), « VHPOP : Versatile Heuristic Partial Order Planner », *in* : *Journal of Artificial Intelligence Research (JAIR)* 20, p. 405–430, DOI : 10.1613/jair.1136 (cité page 57).

YU, Kevin, Ashish Kumar BUDHIRAJA et Pratap TOKEKAR (2018), « Algorithms for Routing of Unmanned Aerial Vehicles with Mobile Recharging Stations », *in* : *IEEE Internatioal Conference on Robotics and Automation (ICRA 2018)*, Brisbane, Australia, DOI : 10.1109/ICRA.2018.8460819 (cité page 54).

ZENG, Haibo et Marco DI NATALE (2012), « Schedulability Analysis of Periodic Tasks Implementing Synchronous Finite State Machines », *in* : *IEEE Euromicro Conference on Real-Time Systems (ECRTS 2012)*, Pisa, Italy, DOI : 10.1109/ECRTS.2012.30 (cité page 44).

ZLOT, Robert et Anthony STENTZ (2003), « Market-Based Multirobot Coordination Using Task Abstraction », *in* : *Field and Service Robotics (FSR 2003)*, Lake Yamanaka, Japan, DOI : 10.1007/10991459_17 (cité pages 52, 77).