Vers le couplage sémantique de planifications de tâches et de trajectoires pour la validation de tâches complexes sous fortes contraintes spatiales

Florent Léoty

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (Toulouse INP)

Discipline ou spécialité :

Robotique et Informatique

Présentée et soutenue par :

M. FLORENT LEOTY

le mercredi 20 décembre 2023

Titre :

Vers le couplage sémantique de planifications de tâches et de trajectoires pour la validation de tâches complexes sous fortes contraintes spatiales

---

Ecole doctorale :

Systèmes (EDSYS)

Unité de recherche :

Laboratoire Génie de Production de l'ENIT (E.N.I.T-L.G.P.)

Directeur(s) de Thèse :

M. BERNARD ARCHIMEDE

M. PHILIPPE FILLATREAU

Rapporteurs :

M. ALBERTO IZAGUIRRE ALTUNA, UNIVERSITE DE MONDRAGON

M. FAROUK BELKADI, ECOLE CENTRALE DE NANTES

Membre(s) du jury :

M. MOHAMED HEDI KARRAY, ECOLE NATIONALE D'INGENIEUR DE TARBES, Président

M. BERNARD ARCHIMEDE, ECOLE NATIONALE D'INGENIEUR DE TARBES, Membre

MME KHOULOUD BOUKADI, FACULTE DES SCIENCES DE SFAX, Membre

M. PHILIPPE FILLATREAU, ECOLE NATIONALE D'INGENIEUR DE TARBES, Membre

Remerciements

Je souhaite en premier lieu remercier mes directeurs de thèse, Monsieur Bernard Archimède et Monsieur Philippe Fillatreau. Je les remercie de m'avoir offert la possibilité d'effectuer cette thèse sous leur direction. Je les remercie pour leur disponibilité durant ces années de travail. Leur expérience, leurs connaissances et surtout leurs qualités humaines ont grandement contribué à ce que cette aventure se déroule parfaitement, et soit enrichissante tant au niveau professionnel et scientifique que personnel.

Je tiens également à remercier les membres de mon jury pour le temps qu'ils m'ont consacré ainsi que pour leurs retours : Monsieur Hedi Karray pour avoir présidé le jury, Monsieur Farouk Belkadi et Monsieur Alberto Izaguirre Altuna pour avoir évalué mon manuscrit, et à Madame Khouloud Boukadi pour avoir assisté à ma soutenance en qualité d'examinatrice.

Je remercie également les personnes que j'ai rencontré au Laboratoire Génie de Production durant cette période, et plus particulièrement Benjamin Mauzé qui a été d'un grand soutien pour la préparation de ma soutenance, ce toujours dans la bonne humeur. Je remercie également Raphaël Marczak pour ses encouragements et sa bienveillance à mon égard. Merci aussi à Benjamin Paul, sans qui je n'aurais peut-être pas commencé cette aventure.

J'aimerais également remercier mes proches qui m'ont soutenu et encouragé tout au long de ce voyage, spécialement Sophie Djordjis, Clément Ara, Alexandre Goyheneix et Thomas Sanchez.
Enfin, je tiens à remercier du fond du coeur mes parents, qui ont toujours soutenu mes choix et qui ont toujours été présents.

Table des matières

1 Introduction **1**

2 Contexte et état de l'art **6**
2.1 Contexte Industriel . . . 6
2.1.1 Industrie et PLM . . . 6
2.1.2 Prototypes virtuels . . . 7
2.1.3 Utiliser la RV dans le processus du PLM . . . 8
2.1.3.1 Définition de la RV . . . 9
2.1.3.2 RV et PLM . . . 9
2.1.4 Synthèse . . . 11
2.2 Contexte scientifique . . . 11
2.3 État de l'art . . . 12
2.3.1 Planification de trajectoires . . . 13
2.3.1.1 Planification de trajectoires basée sur des modèles purement géométriques . . . 13
2.3.1.2 Planification de trajectoires utilisant des informations de haut niveaux d'abstraction . . . 17
2.3.1.2.1 Carte sémantique 3D des objets . . . 17
2.3.1.2.2 Modélisation multi-niveaux de l'environnement . . . 17
2.3.1.3 Synthèse . . . 22
2.3.2 Planification de tâches . . . 22
2.3.2.1 Planification classique . . . 23
2.3.2.1.1 Méthode de recherche . . . 24
2.3.2.1.2 Limites de la planification classique . . . 25
2.3.2.2 Planification temporelle . . . 25
2.3.2.3 Planification sous incertitudes . . . 27
2.3.2.3.1 Planification conditionnelle . . . 27
2.3.2.3.2 Planification probabiliste . . . 27
2.3.2.3.3 Limites de la planification sous incertitude . . . 28
2.3.2.4 Synthèse sur la planification de tâches . . . 28
2.3.3 Planification conjointe de tâches et de trajectoires : TAMP . . . 28
2.3.3.1 Planification TAMP régulières . . . 28
2.3.3.1.1 Méthodes de planification TAMP . . . 29
2.3.3.1.2 Optimisation de la planification TAMP . . . 30
2.3.3.1.3 Synthèse . . . 31
2.3.3.2 Planification TAMP en ligne . . . 31

2.3.3.2.1 Planification des comportements . . . 32
2.3.3.2.2 Planification TAMP basée sur le contrôle . . . 32
2.3.3.2.3 Synthèse . . . 33
2.3.3.3 Planification TAMP et apprentissage . . . 34
2.3.3.3.1 Apprentissage pour la planification symbolique . . . 34
2.3.3.3.2 Apprentissage pour la planification de trajectoires contraint par la tâche . . . 34
2.3.3.3.3 Synthèse . . . 35
2.3.3.4 Stratégies de planification TAMP . . . 35
2.3.3.5 Synthèse sur la planification TAMP . . . 36
2.3.4 Les ontologies comme modèle de connaissance . . . 36
2.3.4.1 Définition de l'interopérabilité . . . 37
2.3.4.2 Définition et structure d'une ontologie . . . 38
2.3.4.2.1 Définition . . . 38
2.3.4.2.2 Structure . . . 39
2.3.4.3 Les niveaux d'abstractions . . . 39
2.3.4.3.1 Ontologies de haut niveau . . . 40
2.3.4.3.2 Ontologies de domaines liées à nos recherches . . . 43
2.3.4.4 Méthodologie pour construire une ontologie . . . 46
2.3.4.5 Synthèse . . . 49
2.3.5 Synthèse de l'état de l'art et positionnement de nos travaux . . . 50

3 Proposition d'ontologies pour le TAMP **52**
3.1 Introduction . . . 52
3.2 Proposition de l'ontologie ENVOn-2 . . . 54
3.2.1 Présentation de ENVOn . . . 54
3.2.2 Construction de ENVOn-2 . . . 56
3.2.2.1 Phase de spécification . . . 56
3.2.2.2 Phase de conceptualisation, de formalisation et d'intégration . . . 57
3.2.2.3 Phase d'implémentation . . . 59
3.2.2.3.1 Architecture générale . . . 59
3.2.2.3.2 Description du niveau Méta . . . 61
3.2.2.3.3 Description du niveau Domaine . . . 61
3.2.2.3.4 Description du niveau Instance . . . 64
3.2.2.4 Phase d'évaluation . . . 64
3.3 Proposition de l'ontologie TAMPO . . . 64
3.3.1 Phase de spécification . . . 65
3.3.1.1 Objectif de l'ontologie . . . 65
3.3.1.2 Les exigences de TAMPO . . . 65
3.3.1.3 Questions de compétences . . . 65
3.3.2 Phase d'intégration . . . 66
3.3.3 Phase de conceptualisation et de formalisation . . . 66
3.3.4 Phase d'implémentation . . . 68
3.3.4.1 Architecture générale . . . 68
3.3.4.2 Description du niveau Méta . . . 70
3.3.4.3 Description du niveau Domaine . . . 70
3.3.4.4 Description du niveau Instance . . . 71

3.3.5 Phase d'évaluation . . . 71
3.4 Synthèse sur la construction des ontologies . . . 71

4 Proposition d'une méthodologie pour le couplage sémantique de planifications de tâches et de trajectoires 73
4.1 Introduction . . . 73
4.2 Approche globale . . . 73
4.3 Contrôle sémantique sur la planification de trajectoires . . . 78
4.3.1 Introduction . . . 78
4.3.2 Réduction de l'espace de recherche d'un algorithme probabiliste . . . 79
4.3.3 Utilisation d'une stratégie multi-planificateurs . . . 80
4.3.4 Stratégies de planification de trajectoires proposées . . . 81
4.3.5 Synthèse sur la planification de trajectoires par contrôle sémantique . . . 82
4.4 Contrôle de la planification de tâches . . . 83
4.4.1 Introduction . . . 83
4.4.2 Concepts généraux . . . 83
4.4.2.1 Définition du PDDL . . . 83
4.4.2.2 Compétition Internationale de Planification . . . 84
4.4.3 Définition de nos besoins . . . 85
4.4.4 Évaluation de différents plans de tâches . . . 86
4.4.5 Description de la fonction de coût utilisée . . . 87
4.4.6 Synthèse sur le contrôle de la planification de tâche . . . 88
4.5 Synthèse générale . . . 88

5 Implémentation et résultats expérimentaux 89
5.1 Introduction . . . 89
5.2 Logiciels utilisés . . . 90
5.2.1 Virtools . . . 90
5.2.2 Protégé . . . 91
5.3 Description des cas d'utilisation utilisés . . . 92
5.3.1 Premier cas d'utilisation : Insertion d'un stylo dans une boîte . . . 93
5.3.2 Deuxième cas d'utilisation : Jeu d'insertion de formes . . . 94
5.3.3 Troisième cas d'utilisation : Insertion d'une pompe . . . 95
5.3.4 Quatrième cas d'utilisation : Changement de piles . . . 96
5.3.5 Synthèse . . . 97
5.4 Résultats : Amélioration du contrôle sémantique sur la planification de trajectoires pour une tâche primitive . . . 97
5.4.1 Résultats expérimentaux des stratégies de planification de trajectoires . . . 98
5.4.1.1 Premier cas d'utilisation : insertion d'un stylo dans une boîte . . . 98
5.4.1.2 Deuxième cas d'utilisation : Jeu d'insertion de formes . . . 107
5.4.1.3 Troisième cas d'utilisation : Insertion d'une pompe . . . 115
5.4.2 Synthèse des résultats sur la planification de trajectoires . . . 119
5.5 Validation de notre approche pour le couplage sémantique de la planification de tâches et de trajectoires basée ontologies . . . 121
5.5.1 Appel du planificateur de tâches . . . 121
5.5.2 Instanciation des ontologies . . . 123
5.5.3 Évaluation des différents plans et choix . . . 126

5.5.4 Simulation et validation sur un cas d'utilisation : changement de piles 127
5.5.5 Synthèse des résultats de notre approche collaborative 129
5.6 Synthèse 131

6 Conclusion et perspectives **132**
6.1 Synthèse des contributions 132
6.2 Perspectives 133
6.2.1 Utiliser la méréotopologie pour générer les contraintes spatiales 134
6.2.2 Intégrer le rebouclage entre les planificateurs en cas d'échec 134
6.2.3 Développer de nouvelles modalités d'interaction entre un opérateur humain et le module d'assistance TAMP 135
6.2.4 Remise en cause du plan de tâches par un opérateur humain 135

Table des figures

1.1 Représentation de notre méthodologie de recherche 4

2.1 Modèle de cycle en V pour le développement d'un produit 7
2.2 Modèle de cycle en V pour le développement d'un produit 8
2.3 Interfaçage entre le monde réel et le monde virtuel [Ladeveze, 2010] 9
2.4 Performances des algorithmes P-RRT* et RRT* ([Qureshi and Ayaz, 2016]) 16
2.5 Modèle de l'environnement multi-niveaux ([Cailhol et al., 2015]) 18
2.6 Modèle de l'environnement multi-niveaux et planification de trajectoires en deux étapes ([Cailhol et al., 2015]) 20
2.7 Stratégie G 21
2.8 Startégie GT 21
2.9 Stratégie GTO 31
2.10 Un cadre TAMP classique basé sur le contrôle ([Castaman et al., 2021]) 33
2.11 Cadre d'interopérabilité de l'entreprise 38
2.12 Niveau d'abstraction d'une ontologie 40
2.13 Ontologie BFO 41
2.14 Ontologie DOLCE 42
2.15 Ontologie SUMO 42

3.1 Positionnement selon le cadre d'interopérabilité de [Chen, 2006] 53
3.2 Architecture générale de l'ontologie ENVOn 55
3.3 Structure de l'ontologie ENVOn-2 60
3.4 Structure de l'ontologie ASK 68
3.5 Structure de l'ontologie TAMPO 69

4.1 Processus de construction d'un PAS et de génération d'une PPQD correspondante . . . 74
4.2 Processus du couplage sémantique TAMP 76
4.3 Diagramme de séquence présentant notre approche globale 77
4.4 Utilisation de l'algorithme A* sur un quadtree 80
4.5 Diagramme de séquence de la planification de trajectoire fine (MG*TO) 81
4.6 MGTO 82

5.1 Environnement Virtools 91
5.2 Script Virtools 92
5.3 Environnement de Protégé 93
5.4 Cas d'utilisation n°1 : insertion d'un stylo 94
5.5 Cas d'utilisation n°2 : jeu d'insertion de formes 95
5.6 Cas d'utilisaton n°3 : insertion d'une pompe dans un coffre de traction 95

5.7 Cas d'utilisation n°4 : Changement de piles 96
5.8 Instanciation de l'environnement du cas d'utilisation n n°1 dans ENVOn-2 100
5.9 Questions de compétences sur l'environnement 101
5.10 Questions de compétences sur les contraintes spatiales et géométriques 102
5.11 Instanciation des informations relatives au TAMP pour le cas d'utilisation n°1 dans TAMPO 103
5.12 Représentation du graphe topologique du premier cas d'utilisation 104
5.13 Résultats sur le cas d'utilisation n°1 sans réduction de l'espace de recherche 106
5.14 Résultats sur le cas d'utilisation n°1 avec réduction de l'espace de recherche 106
5.15 Trajectoires obtenues pour le premier cas d'utilisation 107
5.16 Instanciation de l'environnement du cas d'utilisation n°2 dans ENVOn-2 109
5.17 Représentation des vecteurs d'une contrainte géométrique 110
5.18 Instanciation des informations relatives au TAMP pour le cas d'utilisation n°2 dans TAMPO111
5.19 QC : Dans quel trou est-il possible d'insérer la pièce cylindrique ? 113
5.20 QC : Quel est le trou à favoriser pour insérer Triangle1 ? 114
5.21 Trajectoires obtenues pour le deuxième cas d'utilisation 115
5.22 Résultats sur le cas d'utilisation n°3 sans réduction de l'espace de recherche 118
5.23 Résultats sur le cas d'utilisation n°3 avec réduction de l'espace de recherche 118
5.24 Trajectoires obtenues pour le troisième cas d'utilisation 119
5.25 Description du domaine et du problème en langage PDDL 122
5.26 Instanciation des informations relatives au TAMP pour le cas d'utilisation n°4 dans TAMPO 124
5.27 Instanciation de l'environnement du cas d'utilisation n°4 dans ENVOn-2 125
5.28 Évaluation des plans de tâches à l'aide de requête SPARQL 127
5.29 Graphe topologique pour une tâche de retrait 129
5.30 Trajectoires obtenues pour les différentes tâches primitives du plan de tâches n°1 (Stratégie GT et MG*TO) 130
5.31 Trajectoires obtenues pour les différentes tâches primitives du plan de tâches n°2 (Stratégie GT et MG*TO) 130

Liste des tableaux

2.1 Stratégies de planification de trajectoires 13
2.2 Relations utilisées dans les ontologies 39

3.1 Questions de compétences de l'ontologie ENVOn-2 57
3.2 Classes principales de l'ontologie SUMO 58
3.3 Questions de compétences de l'ontologie TAMPO 66
3.4 Principaux concepts de l'ontologie ASK 67
3.5 Classes du niveau Méta de l'ontologie TAMPO 70

5.1 Synthèse des résultats de planification de trajectoires pour le premier cas d'utilisation . 107
5.2 Nombre de tirages aléatoires pour les différentes stratégies - deuxième cas d'utilisation 115
5.3 Temps de calcul pour les différentes stratégies - deuxième cas d'utilisation 116
5.4 Longueur des trajectoires obtenues pour les différentes stratégies - deuxième cas d'utilisation 116
5.5 Tableau récapitulatif des résultats pour l'ensemble des stratégies déployées sur le cas d'utilisation n°3 119
5.6 Temps de calcul nécessaire pour résoudre l'ensemble des plans de tâches 128
5.7 Temps de calcul pour résoudre chacune des actions primitives des différents plans de tâches 129

Liste des acronymes

ASK Action Specification Knowledge.

BB Building Blocks.
BFO Basic Formal Ontology.
BIM Building Information Modeling.
B-Rep Boundary representation.

CAO Conception Assistée par Ordinateur.
CG Contraintes Géométriques.
CIP Concours International de Planification.
CoMPNetX Constrained Motion Planning Networks X.
CS Contraintes Spatiales.
CSG Géométrie de Construction des Solides.
CSP Problèmes de Satisfaction de Contraintes.

DOLCE Descriptive Ontology for Linguistic and Cognitive Engineering.
DUL DOLCE+DnS Ultralite.

FF Fast-Forward.

HTN Hierarchical Task Network.

IA Intelligence Artificelle.
IDTMP Iteratively Deepened Task and Motion Planning.
IEEE Institute of Electrical and Electronics Engineers.
IRI Internationalized Resource Identifier.

LCGP Least Committed GraphPlan.
LGP Laboratoire Génie de Production.

MAMO Manipulation Among Movable Obstacles.

NAMO Navigation Among Movable Obstacles.

ONIONS ONtologic Integration Of Naive Sources.

OWL Web Ontology Language.

PA Protocole d'Application.
P-ALS Planning-Acting Loop System.
PAS Primitive Action Specification.
PDDL Langage de Description du Domaine de la Planification.
PLM Product Lifecycle Management.
PPQD Path Planning Query Description.
PRM Probabilistic Roadmap.

QC Questions de Compétences.

RA Réalité Augmentée.
RH-TAMP Receding Horizon Task and Motion Planning.
ROS Robot Operating System.
RRT Rapidly-exploring Random Tree.
RV Réalité Virtuelle.

SDK Software Development Kit.
SMT Satisfiabilité Modulo des Théories.
SPARQL Sparql Protocole and RDF Query Language.
STEP STandard for the Exchange of Product model.
STRIPS Stanford Research Institute Problem Solver.
SUMO Suggested Upper Merged Ontology.
SUO Standard Upper Ontology.
SWRL Semantic Web Rule Language.

TAMP Task And Motion Planning.
TLO Top-Level Ontology.
TOVE Toronto Virtual Enterprise.
TR Temps Réel.
TSP Travelling Salesman Problem.

VRPN Virtual-Reality Peripheral Network.

WFOL WonderWeb Fundational Ontologies Library.

Résumé

Pour rester compétitifs, les industriels doivent réduire de plus en plus les coûts et les temps de développement de leurs nouveaux produits. Ceux-ci sont aujourd'hui de plus en plus intégrés, plus petits, plus légers et moins gourmands en énergie. Ils sont plus difficiles à concevoir et doivent être assemblés, maintenus et désassemblés sous de très fortes contraintes géométriques. Traditionnellement, en phase de conception, on établit le modèle Conception Assistée par Ordinateur (CAO) du produit, puis on fabrique les différentes parties physiques de celui-ci pour s'apercevoir trop souvent ensuite, que tout ou partie des tâches associées au cycle de vie du produit sont difficiles ou impossibles à réaliser. Une détection tardive de ces problèmes nécessite alors de remettre en cause la conception du produit.

Les travaux de cette thèse s'intéressent à valider, dès la phase de conception et par simulation numérique avant la fabrication des prototypes physiques, l'ensemble des tâches associées au Product Lifecycle Management (PLM), ce qui permettrait de réduire les temps et coûts de développement et de viser des procédés de fabrication plus respectueux de l'environnement en réduisant le nombre de prototypes physiques fabriqués.

Une étape clef dans la validation par simulation des tâches du PLM consiste à trouver une trajectoire réalisable et sans collision afin de prouver leur faisabilité. La communauté robotique a, depuis les années 80, mis en œuvre des méthodes de planification automatiques de trajectoires pour résoudre cette problématique. Toutefois, ces méthodes ont des limites, principalement liées à la complexité des modèles de l'environnement, traditionnellement purement géométriques. Dans des environnements très complexes, les planificateurs de trajectoires peuvent proposer des trajectoires peu pertinentes, dans des temps pouvant être très longs, voire échouer. Pour répondre à ces limites, des travaux ont considéré des approches collaboratives homme - planificateur mais qui ne permettent que rarement une interaction continue.

Par ailleurs, les techniques de Réalité Virtuelle (RV) permettent la simulation avec un opérateur humain dans la boucle, en immersion dans l'environnement virtuel et en interaction avec celui-ci.
Une approche originale liant planification automatique de trajectoires et RV a ainsi été développée au Laboratoire Génie de Production (LGP) permettant de profiter de la puissance de calcul des ordinateurs et des capacités cognitives d'un opérateur humain. Toutefois, dans cette approche, l'assistance proposée à l'opérateur n'est pas orientée vers le métier et la tâche à réaliser. Pour pouvoir raisonner au niveau de la tâche à réaliser il faut considérer conjointement planification de tâches et planification de trajectoires et s'intéresser à la capacité de modéliser des informations relatives à cette tâche et de raisonner sur celles-ci ; les ontologies sont un outil prometteur.

L'objectif de cette thèse concerne l'élaboration d'une méthodologie pour le couplage sémantique des planificateurs de trajectoires et de tâches pour l'assistance à la manipulation en RV ou la robotique. Dans ce cadre, nous proposons deux contributions principales :
La première contribution de ce travail propose deux ontologies originales. La première, ENVOn-2, concerne la modélisation de l'environnement dans lequel se déroule une tâche. La seconde, TAMPO, est une ontologie développée pour le planification conjointe de tâches et de trajectoires.
La seconde contribution porte sur l'élaboration d'une méthodologie pour le couplage sémantique des planificateurs de tâches et de trajectoires. Cette méthodologie, par l'utilisation conjointe des deux ontologies, permet d'améliorer la planification de trajectoires d'une action primitive tout en proposant un plan (ou des plans) de tâche (s) pertinent(s) pour la manipulation effectuée.

Ces développements ont ensuite été validés à l'aide de scénarios variés et de complexités croissantes. Les résultats obtenus montrent la pertinence de l'approche.

Mots clefs : Couplage sémantique de planification de tâches et de trajectoires, Connaissance liée à la planification conjointe de tâches et de trajectoires, Ontologies, Contraintes spatiales et géométriques, Raisonnement, Assistance à la validation de scénarios

Abstract

To remain competitive, manufacturers need to reduce the costs and development times of their new products. Current products are increasingly integrated, smaller, lighter and more energy-efficient. They are more difficult to design and have to be assembled, maintained and disassembled under very high geometric constraints. Traditionally, during the design phase, the CAD model of the product is established, then the physical parts of the product are manufactured, to discover all too often that some or all of the tasks associated with the product's life cycle are difficult or impossible to carry out. If these problems are detected too late, the product design has to be reconsidered.

The aim of this thesis is to validate, at the design stage, all the tasks associated to the PLM using digital simulation before the physical prototypes are manufactured. This would make it possible to reduce development times and costs and to aim for more environmentally-friendly manufacturing processes by reducing the number of physical prototypes manufactured.

A key step in the simulation-based validation of PLM tasks is to find a feasible collision-free trajectory in order to prove their feasibility. Since the 1980s, the robotics community has been using automatic path planning methods to solve this problem. However, these methods have limitations, mainly linked to the complexity of the environment models, which are traditionally purely geometric. In very complex environments, path planners can propose trajectories that are not very relevant, in times that can be very long, or even fail. To overcome these limitations, some works has considered collaborative human-planner approaches, but these rarely enable continuous interaction.

On the other hand, VR techniques allow simulation with a human operator in the loop, immersed in the virtual environment and interacting with it.

An original approach linking automatic path planning and VR has been developed at LGP, taking advantage of the computing power of computers and the cognitive abilities of a human operator. However, in this approach, the assistance offered to the operator is not oriented towards the task to be carried out. In order to be able to reason at the level of the task to be carried out, task planning and path planning must be considered together, and attention must be paid to the ability to model information relating to the task and to reason about these information ; ontologies are a promising tool.

The aim of this thesis is to develop a common framework for the semantic coupling of path and task planners for manipulation assistance in VR or robotics. Within this framework, we propose two main

contributions :
The first contribution of this work proposes two original ontologies. The first, ENVOn-2, concerns the modelling of the environment in which a manipulation task takes place. The second, TAMPO, is an ontology developed for jointly use path and task planning.
The second contribution concerns the development of a methodology for the semantic coupling of task and trajectory planners. This methodology, through the joint use of the two ontologies, makes it possible to improve the path planning of a primitive action while proposing a task plan (or plans) that is (are) relevant to the manipulation being carried out.

These developments were then validated using a variety of scenarios of increasing complexity. The results obtained demonstrate the relevance of the approach.

Keywords : Semantic coupling of task and motion planning, Knowledge related to joint task and motion planning, Ontologies, Spatial and geometric constraints, Reasoning, Assistance in validating scenarios

Chapitre 1

Introduction

Les travaux de cette thèse s'inscrivent dans la lignée des travaux développés depuis plus de 15 ans au LGP. Ceux-ci répondent à des problématiques liées à l'Industrie 4.0. La compétitivité économique étant de plus en plus forte, les industriels doivent réduire les coûts de production tant au niveau temporel que financier, tout en améliorant la qualité de leurs produits. Dans le même temps, les produits fabriqués sont de plus en plus intégrés, plus difficiles à concevoir, à réaliser et à valider, et doivent être assemblés, maintenus et désassemblés sous de très fortes contraintes géométriques.

Les méthodes de développement d'un produit suivent l'approche historique suivante, en phase de conception, le modèle CAO du produit est établi, puis les différentes parties physiques de celui-ci sont fabriquées pour s'apercevoir trop souvent ensuite, que tout ou partie des tâches associées au cycle de vie du produit sont difficiles ou impossibles à réaliser. Une détection tardive de ces problèmes nécessite alors de remettre en cause la conception du produit.

Ainsi, les industriels ont exprimé le besoin de valider l'ensemble des tâches du cycle de vie du produit (assemblage, maintenance, désassemblage) dès la phase de conception par des simulations sur des maquettes numériques ce qui permettrait de réduire les temps et coûts de développement et de viser des procédés de fabrication plus respectueux de l'environnement en réduisant le nombre de prototypes physiques fabriqués.

Pour valider, à partir d'un modèle numérique, les tâches du cycle de vie du produit, une étape clef est de trouver une trajectoire pour montrer la faisabilité des scénarios simulés. Cette question a donné lieu, à partir des années 80, au développement par la communauté robotique de méthodes de planification automatique de trajectoires. Toutefois, les techniques explorées connaissent certaines limites, principalement liées à la complexité de l'environnement mis en œuvre (ni les modèles de l'environ-

nement ni les méthodes d'exploration de ces modèles ne s'appuient sur des informations relatives à la tâche à réaliser). Ainsi, dans de tels environnements, les planificateurs de trajectoires peuvent proposer des solutions peu pertinentes au regard de la tâche à réaliser, possiblement dans des temps très longs, voire échouer.

En parallèle, la RV a émergé ces dernières années et permet d'envisager la simulation interactive et immersive. Dans ces simulations, un opérateur humain est dans la boucle et a la possibilité d'agir en temps réel sur la scène virtuelle, pour déplacer des objets, modifier le point de vue ou naviguer dans celle-ci.

Le LGP a alors développé une démarche originale qui associe des techniques de planification automatique de trajectoires et la RV permettant la planification interactive et immersive de trajectoires avec guidage visuo-haptique de l'opérateur humain dans la boucle. La démarche développée a permis d'aborder les questions de la planification de trajectoires collaborative et du partage d'autorité entre l'humain et les algorithmes de planification de trajectoires mis en jeu. Les résultats obtenus ont permis de démontrer la pertinence de l'approche, la collaboration planificateur - humain développée permettant de valider les simulations plus rapidement que si l'opérateur humain ou les algorithmes de planification de trajectoires sont chargés seuls de cette validation.

Pour améliorer cette démarche, le LGP a ensuite proposé une architecture originale multi-niveaux pour la modélisation d'environnement 3D et la planification de trajectoires en considérant des données de plus haut niveau d'abstraction (topologiques et sémantiques) que les données purement géométriques traditionnellement utilisées. Cette approche a permis d'améliorer la pertinence des trajectoires proposées tout en réduisant significativement les temps de calcul. Cependant, le niveau sémantique utilisé reste embryonnaire. De plus, la planification de trajectoires, seule, n'est pas suffisante pour valider une tâche du cycle de vie du produit.

Pour pouvoir raisonner au niveau de la tâche à réaliser il faut :

— Considérer conjointement planification de tâches et planification de trajectoires. Les techniques de planification de tâches sont issues de la communauté scientifique de l'Intelligence Artificelle (IA) alors que les techniques de planification de trajectoires proviennent de celle de la Robotique, ce qui explique que ces communautés n'ont commencé à étudier la question difficile de leur mise en œuvre conjointe que récemment. L'étude de la problématique du Task And Motion Planning (TAMP) est aujourd'hui au cœur des préoccupations d'une communauté bien identifiée de chercheurs en Robotique et en IA.

— Se doter de capacités de représentation de la connaissance et de raisonnement relatifs à la tâche

à réaliser. Les ontologies ([Gruber, 1993]) issues des sciences cognitives et de l'IA sont un outil prometteur ; l'étude de l'utilisation des ontologies en Robotique suscite un intérêt croissant de la part de communauté scientifique, très souvent d'ailleurs pour coupler sémantiquement planification de tâches et de trajectoires.

Ces deux verrous sont au cœur de nos travaux de thèse qui s'intéressent au couplage sémantique basé ontologies des planifications de tâches et de trajectoires.

Dans ce contexte, nos contributions sont :

- La proposition de deux ontologies originales.
  - La première, ENVOn-2, est une ontologie pour la représentation des environnements 3D dans lesquels s'effectuent les tâches à valider.
  - La deuxième, TAMPO, permet de modéliser les différents concepts relatifs au TAMP.
- L'élaboration d'une méthodologie pour le couplage sémantique des planifications de tâches et de trajectoires. Cette méthodologie utilisant conjointement les deux ontologies (ENVOn-2 et TAMPO) permet d'améliorer la planification de trajectoires d'une action primitive. La démarche proposée s'appuie sur a) une nouvelle stratégie de planification de trajectoires qui utilise une combinaison de plusieurs algorithmes pour une requête globale basée sur nos ontologies, et b) une utilisation conjointe de ces ontologies permettant d'évaluer différents plans de tâches et de proposer le plus pertinent au regard de la tâche à réaliser, et enfin c) des capacités de représentation et de raisonnement relatives à des contraintes spatiales exprimées au niveau de la tâche à valider et géométriques exprimées au niveau de la planification de trajectoires.

Ces contributions ont donné lieu à 2 conférences internationales :

- F. LEOTY, P. FILLATREAU and B. ARCHIMEDE, "Path planning control using high abstraction level environment model and industrial task-oriented knowledge", IEEE International Workshop of Electronics, Control, Measurement, Signals and their application to Mechatronics, ECMSM 2021, pp.1-6, doi : 10.1109/ECMSM51310.2021.9468850.
- F. LEOTY and J. THAI and B. ARCHIMEDE and P. FILLATREAU and M. GRUNINGER, "Using Mereotopology for Automated Spatial Inference in Task and Motion Planning", The 4th RobOntics Workshop, 32nd IEEE International Conference on Robot and Human Interactive Communication, RO-MAN 2023, Busan, South Korea, August 28th-31st 2023.

  Ce mémoire de thèse est organisé comme suit : le chapitre 2 présente le contexte industriel et scientifique de cette thèse. Dans la partie 2.1, nous présentons le contexte industriel dans lequel s'intègrent nos travaux. Plus précisément, nous nous intéressons au PLM, puis à l'utilisation des maquettes numériques dans l'industrie et enfin, à ce qu'apporte la RV vis-à-vis du PLM. La partie 2.2 présente quant à elle le contexte scientifique dans lequel nos travaux prennent place.

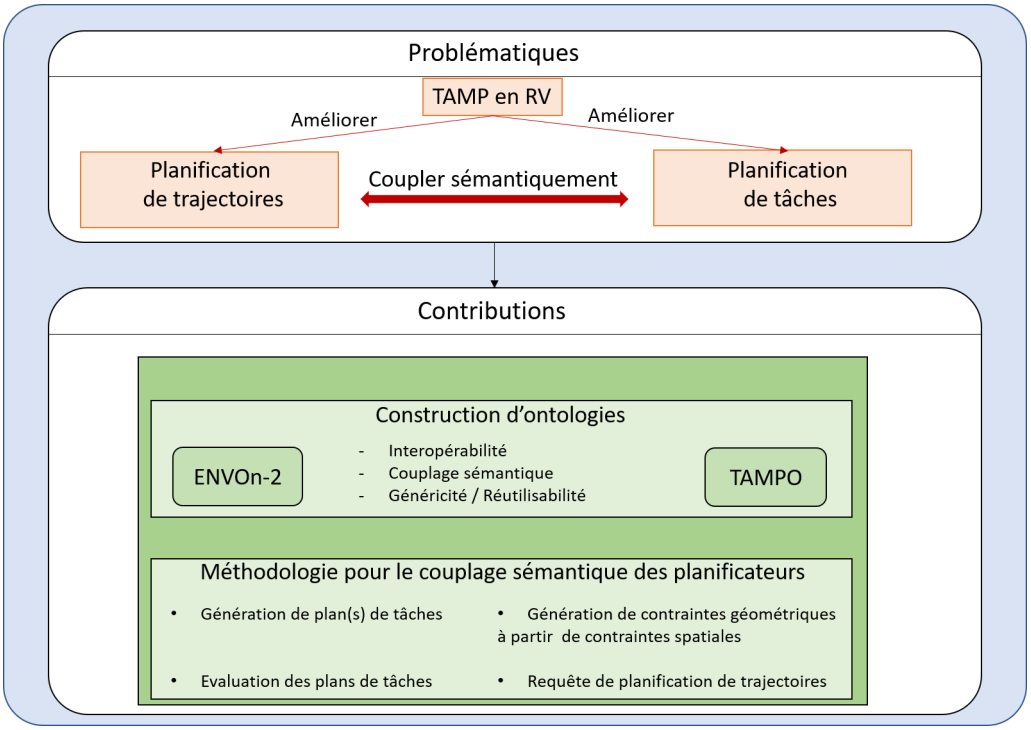


FIGURE 1.1 – Représentation de notre méthodologie de recherche

La partie 2.3 présente l'état de l'art des domaines qui nous intéressent. Dans un premier temps, nous présentons la planification de trajectoires (2.3.1). Dans une deuxième partie, nous traitons de la planification de tâches (2.3.2). Dans la troisième partie (2.3.3), nous discutons de la mise en relation de ces deux planificateurs dans ce qui est appelé le TAMP. Enfin, la quatrième partie présente un modèle de connaissance – les ontologies -, utilisé pour modéliser l'information nécessaire à la planification de trajectoires et/ou de tâches (2.3.4).

Les chapitres 3 et 4 présentent les contributions de ce travail. Plus précisément, le chapitre 3 présente les deux ontologies originales proposées dans cette thèse. Dans la partie 3.2, nous présentons ENVOn-2, une ontologie permettant de représenter un modèle de l'environnement. Elle suit les travaux commençaient au LGP par [Zhao, 2019].
Dans la partie 3.3, nous présentons notre deuxième ontologie, TAMPO, qui est une ontologie de domaine pour le TAMP. Elle permet la modélisation des informations liées à la planification de tâches et de trajectoires.

Le chapitre 4 présente une méthodologie pour le couplage sémantique des planificateurs de tâches et de trajectoires. La partie 4.2 présente la démarche globale. La partie 4.3 décrit comment a été amélioré le contrôle sémantique sur la planification de trajectoires, quant à la partie 4.4, elle décrit comment le plan de tâche est obtenu.

Le chapitre 5 présente l'ensemble des résultats expérimentaux obtenus pour valider notre approche. Nous commençons par présenter les outils et logiciels utilisés dans la partie 5.2. Nous présentons ensuite les différentes applications utilisées dans la partie 5.3. La partie 5.4 présente les résultats de la planification de trajectoire obtenus après avoir amélioré le contrôle sémantique sur celle-ci. Enfin, la partie 5.5 présente les résultats de notre approche complète.

Pour finir, le chapitre 6 présente la conclusion, résumant nos contributions ainsi que les résultats obtenus, et propose de potentielles futures directions pour nos travaux.

Chapitre 2

Contexte et état de l'art

Nos travaux de thèse s'intéressent au couplage sémantique basé ontologies des planificateurs de tâches et de trajectoires.

Dans ce chapitre, nous rappelons le contexte industriel de nos travaux dans le paragraphe 2.1, puis le contexte scientifique dans le paragraphe 2.2. Le paragraphe 2.3 présente les états de l'art des travaux relatifs aux différents domaines liés à nos travaux : la planification de trajectoires (paragraphe 2.3.1), la planification de tâches (paragraphe 2.3.2), la planification conjointe de tâches et de trajectoires (paragraphe 2.3.3), les ontologies relatives aux connaissances de l'information liée aux tâches à valider (paragraphe 2.3.4), et une synthèse de l'état de l'art et le positionnement de nos travaux par rapport à celui-ci (paragraphe 2.3.5).

2.1 Contexte Industriel

Le contexte actuel, avec ses problématiques économiques, environnementales et sociétales, oblige les entreprises à revoir leur système de production. Pour rester compétitives, les entreprises doivent maîtriser parfaitement l'ensemble des tâches du cycle de vie de leur produits et la traçabilité tout au long de celui-ci.

2.1.1 Industrie et PLM

Le PLM est défini comme une activité permettant la gestion des produits créés par une entreprise tout au long de leur cycle de vie (« du berceau à la tombe ») (Stark, 2015).
Le développement d'un produit commence par l'expression d'un besoin du client et se conclut par la réception du produit par celui-ci. Le cycle en V est un des outils couramment utilisés (figure 2.1). Dans un premier temps, les étapes de spécification et de conception sont effectuées

dans la « phase descendante ». Puis l'implémentation est réalisée. Enfin, dans la « phase montante », des tests sont effectués pour valider la conception, puis les spécifications du produit (étape de réception par le client). L'utilisation d'une telle démarche permet de détecter d'éventuelles anomalies le plus tôt possible.
Au vu de la complexité toujours plus importante des produits développés, le nombre de validations à faire augmente fortement. Ainsi, un grand nombre de prototypes sont nécessaires pour valider entièrement un nouveau produit. Or, ceux-ci sont coûteux d'un point de vue financier et temporel. Une alternative aux prototypes physiques est alors envisagée : le prototype virtuel.

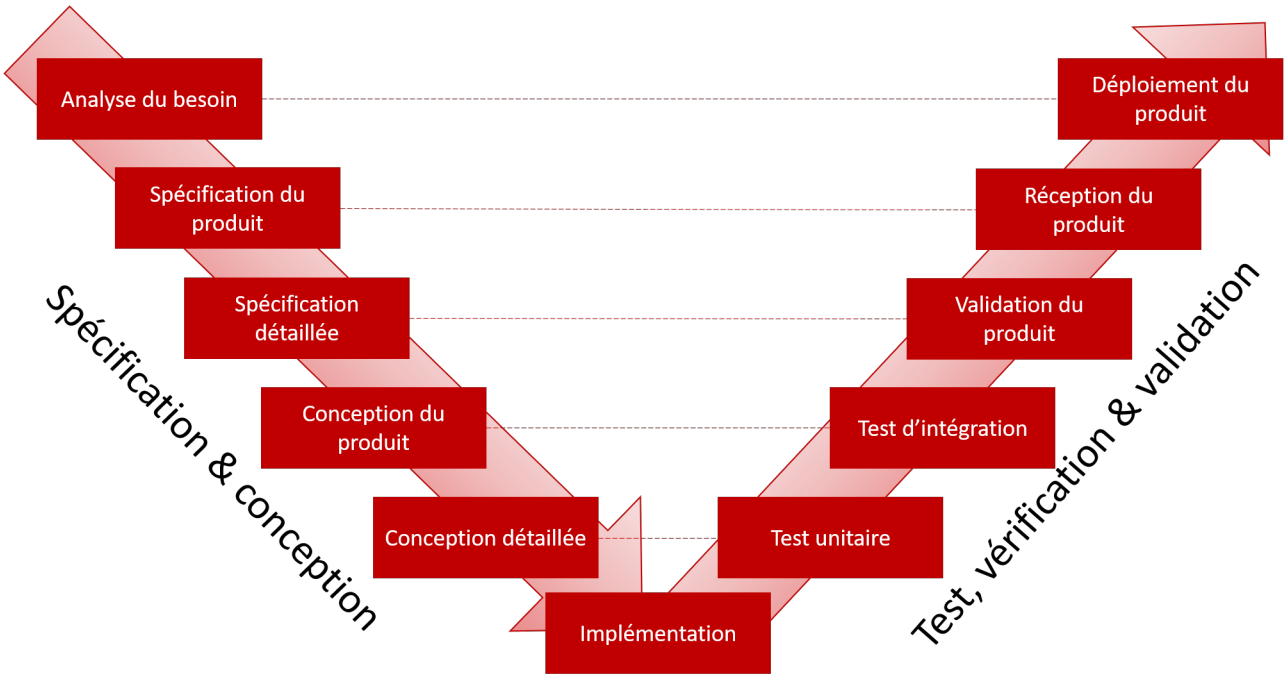


FIGURE 2.1 – Modèle de cycle en V pour le développement d'un produit

2.1.2 Prototypes virtuels

Les modèles numériques des produits développés sont aujourd'hui de plus en plus utilisés pour la validation par simulation des fonctionnalités ou pour l'évaluation des performances ([Xiao and Zhang, 2017]). Ils peuvent permettre de diagnostiquer, entre autres, des anomalies dès la phase de conception, avant toute fabrication de prototypes physiques, permettant de réduire les temps et coûts de développement. Cependant, tous les tests ne peuvent être effectués sur les prototypes virtuels et ceux-ci peuvent comporter des imprécisions. Par ailleurs, les tests finaux doivent être effectués sur des prototypes physiques. Les prototypes virtuels ne les remplacent donc pas ; en revanche, ils permettent de réduire leur utilisation durant l'ensemble du processus de fabrication.
Le cycle en V est alors transformé par l'ajout d'une phase de validation sur prototypes virtuels

qui, une fois approuvée, est suivie par les validations et la réception des prototypes physiques (figure 2.2).

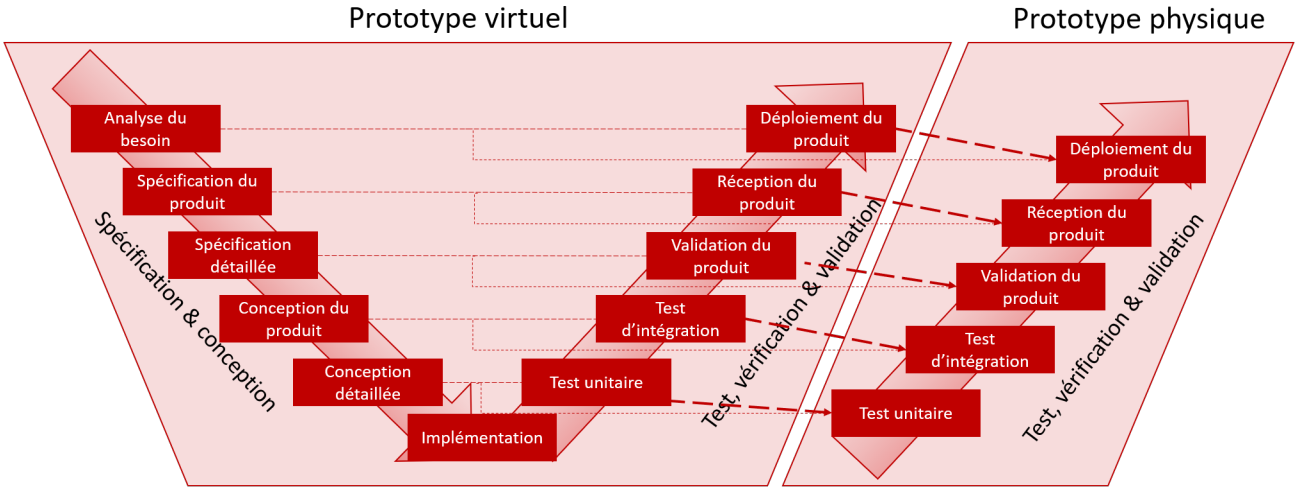


FIGURE 2.2 – Modèle de cycle en V pour le développement d'un produit

Avec la généralisation de l'utilisation des maquettes numériques, de nombreux logiciels ont vu le jour permettant de simuler les comportements multiphysiques d'un produit, son assemblage ou le fonctionnement général d'une ligne de production par exemple. Pour ne citer que quelques exemples :

— Certains outils se concentrent sur les aspects cinématiques (analyse cinématique et animation du système). Nous pouvons citer DMU Kinematics pour CATIA® [1] ou Motion Module pour SolidWorks® [2].

— AUTODESK® SIMULATION [3] permet la simulation d'un nombre étendu de comportements physiques (simulation thermiques, mécaniques des fluides, simulation et analyse de structures, simulations mécaniques ...).

— Certains outils comme Delmia® [4] ou FlexSim® [5] permettent d'aller jusqu'à des simulations de comportement d'usines.

2.1.3 Utiliser la RV dans le processus du PLM

En parallèle, la RV s'est développée et elle permet d'immerger un opérateur humain dans un environnement virtuel, en interaction avec celui-ci.

1. https ://www.3ds.com/fr/produits-et-services/catia/
2. https ://www.3ds.com/fr/products/solidworks
3. https ://www.autodesk.fr/solutions/simulation/overview
4. https ://www.3ds.com/fr/produits-et-services/delmia/
5. https ://www.flexsim.com/fr/

2.1.3.1 Définition de la RV

Dans « Le traité de la réalité virtuelle », la RV est définie comme « un domaine scientifique et technique exploitant l'informatique et des interfaces comportementales en vue de simuler dans un monde virtuel le comportement d'entités 3D, qui sont en interaction en temps réel entre elles et avec un ou des utilisateurs en immersion pseudo-naturelle par l'intermédiaire de canaux sensori-moteurs » ([Fuchs and Moreau, 2003]).

La RV est un outil de simulation intégrant un opérateur humain « dans la boucle » et permettant de tirer profit de ses capacités cognitives. Ce dernier peut percevoir le monde virtuel au travers d'interfaces dites sensorielles. Puis, à l'aide de ses capacités cognitives, peut prendre des décisions pour effectuer des actions, par le biais d'interfaces dites motrices (figure 2.3). Un autre type d'interface, permettant à la fois de ressentir et d'agir, est appelée interface sensori-motrice.

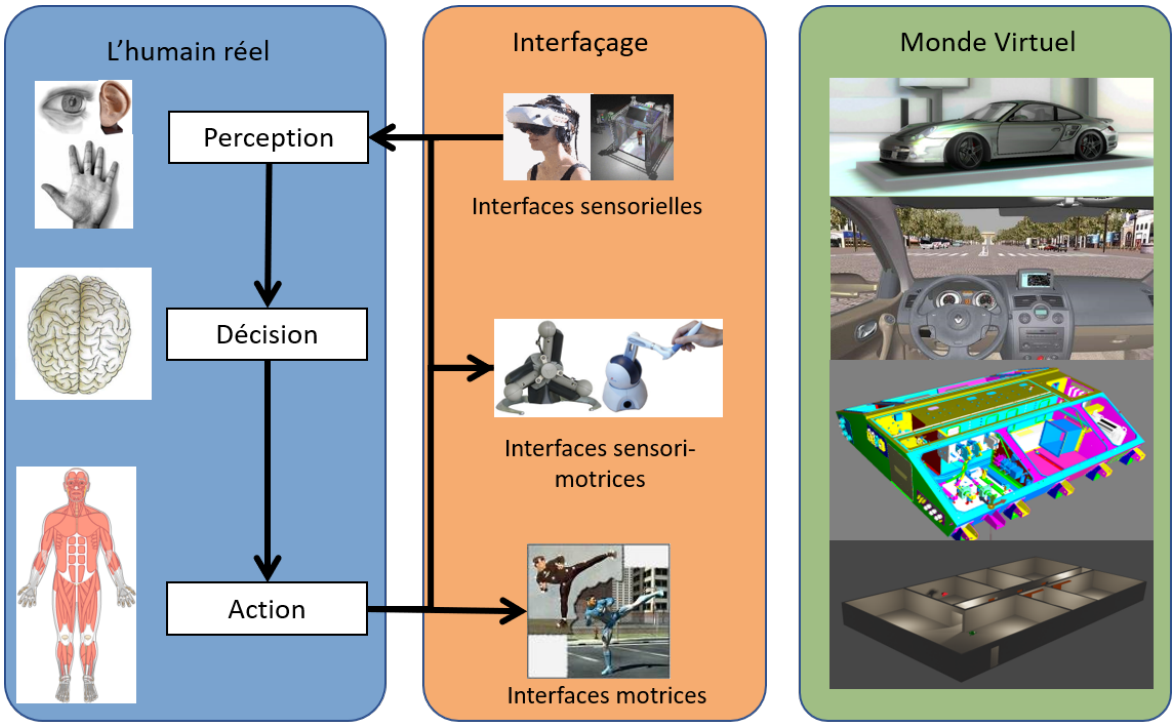


FIGURE 2.3 – Interfaçage entre le monde réel et le monde virtuel [Ladeveze, 2010]

2.1.3.2 RV et PLM

La RV est une technique utilisable pour la simulation de l'ensemble des tâches du cycle de vie du produit. Celle-ci permet d'utiliser des modèles numériques offrant les avantages discutés dans le paragraphe 2.1.2 tout en permettant à un opérateur humain d'interagir avec ces modèles en immersion. Ceci permet d'envisager le test de tâches, de modifier le design rapidement et

facilement des différents modèles utilisés. Différents travaux ont permis de développer des applications de RV destinées à l'industrie. Ces travaux concernent notamment :

— La conception mécanique : l'objectif est de permettre à un utilisateur de modifier la géométrie d'objets virtuels. Virtual Reality Aided Design (VRAD) ([Bourdot et al., 2010]) est un démonstrateur pour la CAO en RV permettant de manipuler des éléments de CAO tout en les enrichissant. Cet outil est destiné à être utilisé par des opérateurs habitués à la CAO. Une autre approche, destiné à des opérateurs non-experts ([Meyrueis, 2011], utilise un contrôleur (une Wiimote) pour déformer localement des objets.

— Les processus de production : La réalité mixte, qui est l'utilisation mixte de la RV et de la Réalité Augmentée (RA), permet de tester l'intégration de nouveaux systèmes dans une ligne de production. ([Lee et al., 2011]) proposent des éléments virtuels qui sont positionnées dans des prises de vues du système réel à l'aide de reconnaissance et à la localisation de marqueurs 2D.

— L'analyse ergonomique : l'ergonomie d'un système de production peut être évalué à l'aide de la RV. [Bennes et al., 2012] propose de visualiser et de tester un poste de travail ainsi que les différentes opérations qui y seront menées. Ces tests permettent de valider l'ergonomie avant l'implémentation réelle du poste de travail.
De plus, la RV permet également de valider l'ergonomie du produit. Une application d'essai d'un fauteuil roulant motorisé équipé d'un bras robotisé pour tester différentes alternatives pour le contrôle du fauteuil et du bras, et ainsi à définir celles à implanter sur le produit réel a été proposée par [Di Gironimo et al., 2013].

— Le BIM : la RV permet de réduire le nombre des modifications de conception dans le Building Information Modeling (BIM). La solution proposée par la RV pour les modifications de conception améliore la collaboration et la communication entre les parties prenantes d'un projet de construction ([Panya et al., 2023]).

— L'assemblage, la manipulation : la RV permet l'entraînement aux tâches de montage et de démontage pour la maintenance de systèmes industriels ([Li and Okamura, 2003]). Ceci peut être effectué à l'aide d'interfaces de visualisation et de manipulation.
De plus, la simulation peut être enrichie par l'utilisation d'outils assistant l'utilisateur. Il existe des outils pouvant définir des contraintes d'assemblage ([Iacob et al., 2012]). Ces outils permettent de déterminer les contraintes d'assemblage et les degrés de liberté des différents composants.
Une assistance plus poussée peut proposer une trajectoire d'assemblage à l'aide d'algorithmes de planification de trajectoires ([Ladeveze, 2010]). La trajectoire proposée est utilisée pour guider un opérateur humain à l'aide d'une interface haptique. La trajectoire proposée a ensuite été améliorée en utilisant des données de plus haut niveau d'abstraction que des données purement géométriques (informations topologiques et séman-

tiques) [Cailhol et al., 2019]), puis [Zhao, 2019] a utilisé des informations relatives à la tâche
.

2.1.4 Synthèse

Nous avons vu dans cette partie que les industriels optaient de plus en plus pour l'utilisation de prototypes virtuels à la place de prototypes physiques dans l'optique de réduire de façon conséquente les temps et coûts de développement de leurs nouveaux produits. En effet, les prototypes virtuels permettent d'effectuer des nombreux tests pour vérifier les fonctionnalités et les performances des produits dès la phase de conception.
Par ailleurs, le recours à de nouvelles technologies comme la RV leur permet d'ajouter un opérateur en immersion dans un monde virtuel. Celui-ci peut interagir directement avec les prototypes virtuels dans l'optique de simuler différentes tâches, notamment des tâches d'assemblages, de maintenances ou de désassemblages, nécessaires dans le processus du PLM.

2.2 Contexte scientifique

La question de la validation de tâches devant être exécutées sous très fortes contraintes géométriques a donné lieu au développement de méthodes de planification automatique de trajectoires par la communauté robotique, à partir des années 1980. Néanmoins, ces techniques connaissent certaines limites, principalement liées à la complexité de l'environnement mis en œuvre (ni les modèles de l'environnement ni les méthodes d'exploration de ces modèles ne s'appuient sur des informations relatives à la tâche à réaliser). Ainsi, dans de tels environnements, les planificateurs de trajectoires peuvent proposer des solutions peu pertinentes vis-à-vis de la tâche à réaliser, possiblement dans des temps très longs, voire échouer.

En parallèle, l'émergence de la RV ces dernières années a permis d'envisager la simulation interactive et immersive. Dans ces simulations, un opérateur humain est dans la boucle et a la possibilité d'agir en temps réel sur la scène virtuelle, pour déplacer des objets, modifier le point de vue ou naviguer dans la scène.

Une démarche originale a alors été développée au LGP, qui associe des techniques de planification automatique de trajectoires et la RV permettant la planification interactive et immersive de trajectoires avec guidage visuo-haptique de l'opérateur humain dans la boucle. La démarche développée a permis d'aborder les questions de la planification de trajectoires collaborative et du partage d'autorité entre l'humain et les algorithmes de planification de trajectoires mis en

jeu. Les résultats obtenus ont permis de démontrer la pertinence de l'approche, la collaboration planificateur-humain développée permettant de valider les simulations plus rapidement que si l'opérateur humain ou les algorithmes de planification de trajectoires sont chargés seuls de cette validation.

Pour améliorer cette démarche, le LGP a ensuite proposé une architecture multi-niveaux originale pour la modélisation d'environnement 3D et la planification de trajectoires en considérant des données de plus haut niveau d'abstraction (topologiques et sémantiques) que les données purement géométriques traditionnellement utilisées. Cette approche a permis d'améliorer la pertinence des trajectoires proposées tout en réduisant significativement les temps de calcul associés. Cependant, le niveau sémantique utilisé reste embryonnaire. De plus, la planification de trajectoires, seule, n'est pas suffisante pour valider une tâche du cycle de vie du produit.

Pour pouvoir raisonner au niveau de la tâche à réaliser il faut :

— Considérer conjointement planification de tâches et planification de trajectoires. Les techniques de planification de tâches sont issues de la communauté scientifique de l'Intelligence Artificielle alors que les techniques de planification de trajectoires proviennent de celle de la Robotique, ce qui explique que ces communautés n'ont commencé à étudier la question difficile de leur mise en œuvre conjointe que récemment. L'étude de la problématique du TAMP est aujourd'hui au cœur des préoccupations d'une communauté bien identifiée de chercheurs en Robotique et en IA.

— Se doter de capacités de représentation de la connaissance et de raisonnement relatifs à la tâche à réaliser. Les ontologies issues des sciences cognitives et de l'IA sont un outil prometteur ; l'étude de l'utilisation des ontologies en Robotique suscite un intérêt croissant de la part de communauté scientifique, très souvent d'ailleurs pour coupler sémantiquement planification de tâches et de trajectoires.

Ces deux verrous sont au cœur de nos travaux de thèse qui s'intéressent au couplage sémantique basé ontologies des planificateurs de tâches et de trajectoires.

2.3 État de l'art

Ce paragraphe présente les états de l'art des travaux relatifs aux différents domaines liés à nos travaux : la planification de trajectoires (paragraphe 2.3.1), la planification de tâches (paragraphe 2.3.2), la planification conjointe de tâches et de trajectoires (paragraphe 2.3.3), les ontologies relatives aux connaissances de l'information liée aux tâches à valider (paragraphe 2.3.4), et une synthèse de l'état de l'art et le positionnement de nos travaux par rapport à celui-ci (paragraphe 2.3.5).

2.3.1 Planification de trajectoires

Pour valider des tâches sous de très fortes contraintes géométriques, une étape clef est de montrer la faisabilité du mouvement. Des techniques de planification de trajectoires ont ainsi été développées pour répondre à cette problématique.
La planification de trajectoires a pour objectif de trouver une trajectoire sans collision dans un espace donné, entre une configuration initiale (S) et une configuration finale (G). L'espace considéré ainsi que les configurations initiales et finales considèrent une requête de planification de trajectoires.
En robotique, une planification de trajectoires a lieu dans un espace des configurations C, défini comme $C = C_{libre} \cup C_{obstacle}$. Dans ce travail, nous allons uniquement parler de planification de trajectoires dans l'espace libre.
Dans le paragraphe 2.3.1.1 sont présentées les méthodes de planification de trajectoires traditionnelles, n'utilisant que des informations géométriques. Les stratégies utilisant des informations de plus haut niveau d'abstraction que des informations purement géométriques sont, elles, présentées dans le paragraphe 2.3.1.2.

2.3.1.1 Planification de trajectoires basée sur des modèles purement géométriques

Historiquement, les techniques de planification de trajectoires ont été développées à partir des années 1980 ([Choset et al., 2005], [Lozano-Perez, 1983], [Latombe, 1991]), par la communauté robotique. [Cailhol et al., 2015] a classé ces techniques selon leur façon d'appréhender l'environnement (approche globale ou locale) et de la méthode mise en œuvre pour obtenir la trajectoire (méthode déterministe ou probabiliste). Cette classification est présentée par le tableau 2.1.

	Approche globale	Approche locale
Méthodes déterministes	- Décomposition en cellule - Feuille de route	- Champ de potentiel
Méthodes probabilistes	- Feuille de route probabiliste	- Rapidly-exploring Random Tree (RRT) / RRT*

TABLE 2.1 – Stratégies de planification de trajectoires

— Les techniques avec approche globale se basent sur un modèle de l'environnement qui peut être créé par la décomposition de l'espace libre en cellules géométriques ou par la saisie de la connectivité de l'espace libre dans d'un graphe (feuille de route ou *Roadmap*). Le modèle de l'environnement, synthétisé dans un graphe, est ensuite exploré afin de

trouver un chemin. Ces approches sont favorisées lorsque plusieurs requêtes de planification peuvent être appliquées dans un même modèle d'environnement préalablement construit ;

— Les techniques avec approche locale se basent, elles, sur une progression autour du point courant. Celles-ci sont principalement utilisées quand la modélisation de l'environnement n'est pas souhaitée, car trop coûteuse (requête de planification unique, environnements dynamiques...) ;

— Les méthodes déterministes sont basées sur des calculs déterministes et fournissent toujours le même résultat pour une même requête ;

— Les méthodes probabilistes sont basées sur une description aléatoire de l'espace des configurations. Ces méthodes garantissent de trouver un résultat s'il existe (on parle de complétude probabiliste), mais ne garantissent pas de trouver cette solution dans un temps fini. Cependant, pour la même requête, elles pourront donner des trajectoires différentes.

1. Les techniques de planification de trajectoires déterministes avec approche globale regroupent :
   (a) La décomposition en cellules de l'espace libre : cette technique décompose l'environnement en cellules géométriques élémentaires. Une cellule peut être libre ou occupée selon la présence d'obstacles. Il est ainsi possible d'utiliser la connexion des cellules libres pour trouver un chemin. Les décompositions proposées varient selon que les cellules sont de tailles identiques ou non, et selon la forme géométrique des cellules élémentaires (par exemple cellules triangulaires [Brooks and Lozano-Perez, 1985]).
   Une autre approche couramment utilisée est la décomposition de l'espace libre dans un arbre déséquilibré (quadtree en 2D [Samet, 1984], octree en 3D [Meagher, 1982]).
   (b) L'utilisation de feuilles de route : cette technique identifie des points d'intérêt de l'environnement dans le but de les relier dans un graphe pour synthétiser la connectivité de l'espace libre. Cela peut être fait à l'aide d'un diagramme de Voronoï qui construit un graphe avec le maximum de distance par rapport aux obstacles ([Voronoi, 1908]) ou d'un graphe de visibilité qui relie les sommets des obstacles pouvant être connectés deux à deux sans rencontrer d'obstacles ([Alt and Welzl, 1988]). On obtient alors un graphe composé d'arcs et de nœuds. Ce dernier est ensuite exploré par un algorithme de parcours de graphe (comme Dijkstra ([Dijkstra, 1959]) ou A* ([Hart et al., 1968])) afin de définir le chemin.

L'un des principaux avantages des techniques déterministes avec approche globale est qu'il s'agit de techniques complètes si les modèles utilisés sont exacts. Sinon, - ce qui est

généralement le cas -, ces techniques sont dites complètes en résolution. Un second avantage est que le modèle peut être calculé hors ligne dans le cas d'environnement statique, et peut donc être réutilisé pour plusieurs requêtes de planification.

2. Les techniques probabilistes avec approche globale :
   Comme pour les techniques déterministes avec approche globale, ces techniques se basent sur une feuille de route pour trouver un chemin. Cependant, la construction de la feuille de route diffère et se fait par tirage aléatoire dans l'environnement libre. Il s'agit de *Probabilistic Roadmap (PRM)*, qui a été développée dans les années 90 ([Overmars, 1992], [Kavraki et al., 1996], [Amato and Dale, 1999], [Wilmarth et al., 1999]). Les techniques basées sur les PRM ont conduit à des travaux visant à optimiser le tirage aléatoire des configurations afin de synthétiser plus rapidement la feuille de route. Pour arriver à ces fins, [Amato and Wu, 1996] proposent des méthodes construisant la feuille de route en partant de configurations qui sont en contact avec des obstacles. Une autre technique consiste à « étendre » l'espace libre en tolérant la pénétration au travers d'obstacles ([Hsu et al., 1998]). La tolérance est ensuite réduite (jusqu'à être nulle) et la feuille de route est partiellement reconstruite jusqu'à ce qu'elle soit satisfaisante. Qu'importe la technique utilisée, après l'obtention de la PRM, un algorithme de parcours de graphe est utilisé pour trouver une trajectoire. Ces techniques sont complètes (complétude probabiliste).

3. Les techniques déterministes avec approche locale :
   Ces techniques s'appuient sur les champs de potentiels. Il s'agit de superposer un champ attractif à l'objectif et un champ répulsif aux obstacles ([Khatib, 1985]). Le robot suit le gradient maximal des potentiels jusqu'à atteindre l'objectif. Cependant, si les temps de calcul peuvent être plus courts qu'avec les autres approches présentées ici, plusieurs problèmes peuvent survenir avec ces techniques. En effet, quatre principaux problèmes ont été identifiés : a) « être piégé » dans des minima locaux, b) ne pas percevoir un passage entre deux obstacles proches, c) « osciller » en présence d'obstacles et d) « osciller » dans des passages contraints ([Koren and Borenstein, 1991]).
   Des solutions existent pour surmonter ces problèmes. Certaines cherchent à supprimer les minima locaux, par exemple en utilisant des fonctions de navigations ([Koditschek, 1987]) ou les équations de Laplace, afin de générer des champs de potentiel ne faisant apparaître qu'un minimum sur la configuration finale. Cela permet d'être certain de trouver une solution, si une existe, mais au prix de lourds calculs et donc de temps de calcul élevé. D'autres méthodes, moins coûteuses, consistent à éviter les minimaux locaux en ajoutant des points de passages intermédiaires ([Zou and Zhu, 2003]).

4. Les techniques probabilistes avec approche locale :
   Ces techniques explorent aléatoirement l'environnement à partir d'une configuration don-

née, a priori la configuration de départ. Il s'agit d'un arbre évolutif tels les algorithmes RRT ([Lavalle, 1998]) ou Bi-RRT ([Kuffner and LaValle, 2000]). Le RRT déploie un arbre à partir de la configuration de départ, puis tire aléatoirement des configurations dans l'ensemble de l'espace libre. Si cela est possible, la configuration est reliée à l'arbre déployé. Ce processus est répété jusqu'à ce que la configuration finale soit atteinte. Le Bi-RRT déploie lui deux arbres, un à partir de la configuration initiale et l'autre à partir de la configuration finale. L'algorithme s'arrête quand les deux arbres peuvent se rejoindre. Le Bi-RRT est plus rapide que le RRT, mais les deux algorithmes présentent un même inconvénient : ils ne sont pas optimaux, l'exploration s'arrêtant à la première solution trouvée. Le RRT* ([Karaman and Frazzoli, 2010]) a été développé pour pallier ce manque d'optimalité. Un premier chemin est trouvé comme le ferait le RRT mais les itérations se poursuivent afin de trouver le chemin optimal (en termes de distance euclidienne). La contrepartie de cette recherche est le temps de calcul plus élevé que le RRT. Pour limiter ce problème, [Qureshi and Ayaz, 2016] ont associés le RRT* aux champs de potentiels afin d'améliorer le temps de calcul dans ce qui est appelé le P-RRT*. La figure 2.4 montre les résultats obtenus par les algorithmes P-RRT* et RRT* en environnement contraint. La figure 2.4 (a) montre le chemin initial trouvé par le P-RRT* quand l'image 2.4 (b) montre le chemin optimal pour le même algorithme. L'image 2.4 (c) montre quant à elle le chemin initial pour l'algorithme RRT* qui est ensuite affiné sur l'image 2.4 (d). Cela démontre que le P-RRT* est le plus performant.

Dans le même temps, [Xinyu et al., 2019] ont développé l'algorithme P-RRT*-connect qui crée deux arbres (à partir de la position de départ et d'arrivée) et les connecte pour trouver un chemin. Ceci aide à diminuer le temps de calcul et facilite donc la recherche en temps réel.

Ces techniques garantissent de trouver un chemin si un existe, mais ne garantissent pas de le trouver dans un temps fini. On parle de complétude probabiliste.

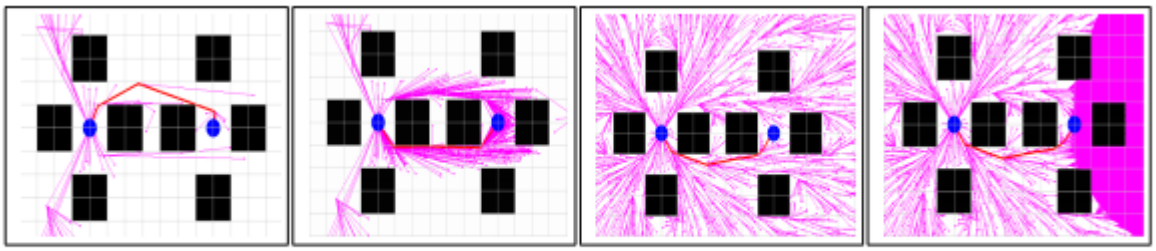


FIGURE 2.4 – Performances des algorithmes P-RRT* et RRT* ([Qureshi and Ayaz, 2016])

En conclusion de cette section 2.3.1.1, nous pouvons dire que de nombreuses techniques de planification de trajectoires existent, mais toutes présentent des limites importantes. En effet,

seules des données purement géométriques sont considérées. Dans un environnement complexe et très contraint, les modèles sont complexes, et les techniques de planification de trajectoires peuvent nécessiter des temps de calcul très élevés, échouer, ou proposer des solutions peu pertinentes au regard de la tâche à exécuter. L'utilisation d'information de plus haut niveau d'abstraction que les données purement géométriques peut alors être envisagée afin d'améliorer les performances de la planification de trajectoires.

2.3.1.2 Planification de trajectoires utilisant des informations de haut niveaux d'abstraction

En observant le comportement des humains, des chercheurs se sont aperçus que les tâches qu'ils accomplissent étaient souvent réalisables rapidement ([Ahmadi-Pajouh et al., 2007]). Ceci peut s'expliquer par le fait qu'ils sont dotés de capacités cognitives élevées ([Cailhol et al., 2019]). Lorsqu'un humain réalise une tâche dans un environnement donné, il ne le considère pas seulement comme un ensemble d'objets géométriques, mais s'appuie également sur ses connaissances et sur un haut niveau d'abstraction pour l'interpréter au mieux. De nombreux travaux ont montré les bénéfices d'utiliser des informations de plus haut degré d'abstraction que les données purement géométriques traditionnellement utilisées (informations topologiques et sémantiques) pour la navigation d'un robot mobile ([Devy et al., 1995]), la modélisation de l'environnement ([Bastianelli et al., 2013], [Belsky et al., 2016]), ou la préhension (*grasping* en anglais) ([Vahrenkamp et al., 2016], [Tosello et al., 2015]). Les parties 2.3.1.2.1 et 2.3.1.2.2 vont présenter les principales approches décrites dans la littérature.

2.3.1.2.1 Carte sémantique 3D des objets

[Rusu, 2010] a présenté une carte sémantique 3D des objets pour annoter l'ensemble des objets présents dans un environnement ainsi que leur surface avec des étiquettes. Cette carte sert de ressources sémantiques pour déterminer la position finale d'une manipulation.
[Günther et al., 2017] ont également créé une carte sémantique d'un environnement intérieur à l'aide de points capturés par une caméra sur un robot mobile. Il associe les nuages de points à des modèles de CAO afin de reconstituer l'environnement le plus précisément possible.

2.3.1.2.2 Modélisation multi-niveaux de l'environnement

Récemment, des chercheurs du LGP ont proposé une modélisation multi-niveaux de l'environnement et une nouvelle architecture pour la planification de trajectoires ([Cailhol et al., 2019]) pour considérer des informations de niveau d'abstraction plus élevé. Cela remplace l'utilisation des données purement géométriques, traditionnellement utilisées pour la planification de trajectoires. Ce modèle sert de support pour une planification de trajectoires en deux étapes.

Modèle multi-niveaux de l'environnement

L'environnement de simulation est considéré comme une partie fermée de l'espace cartésien, qui peut être encombré d'obstacles (statiques ou mobiles), considérés comme des corps rigides. Le modèle de l'environnement proposé se compose d'un modèle d'espace libre et d'un modèle de corps rigides comme présenté sur la figure 2.5. Cet environnement est composé de quatre objets rigides, un mobile (en gris) et trois statiques (en noir).

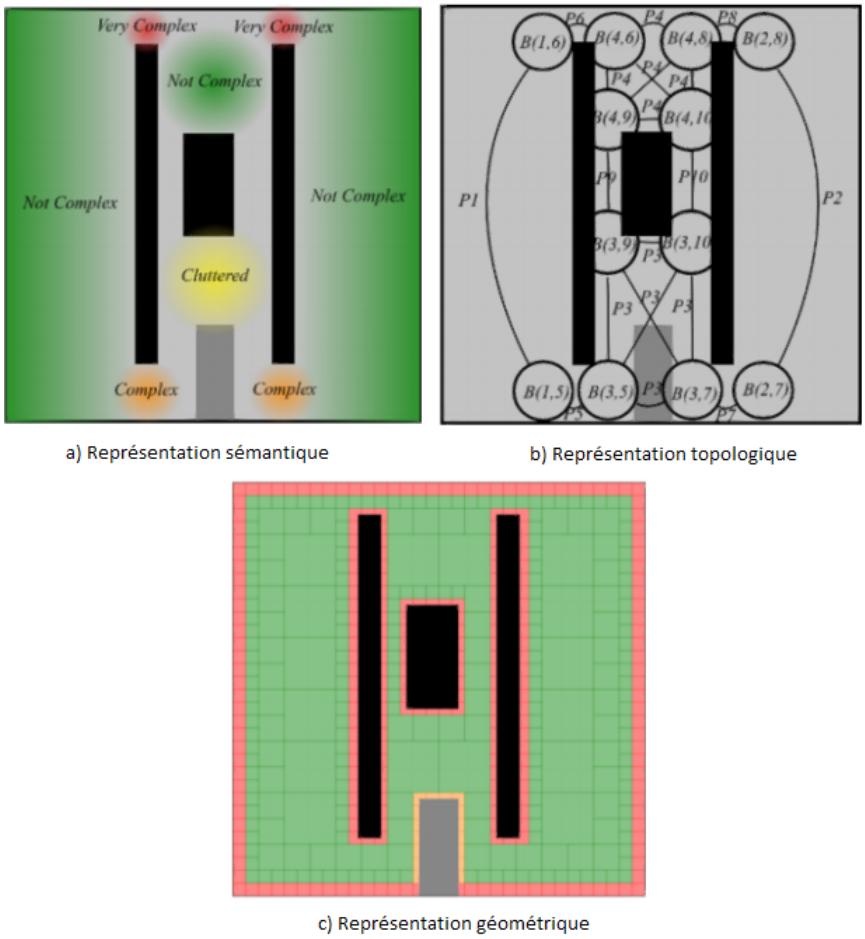


FIGURE 2.5 – Modèle de l'environnement multi-niveaux ([Cailhol et al., 2015])

A/ Modèle de l'espace libre

— Niveau géométrique : Une décomposition cellulaire basée sur un arbre déséquilibré (quadtrees en 2D, octrees en 3D) décrit la géométrie de l'espace libre. (figure 2.5 -c). Cette décomposition distingue trois types de cellules : celles qui n'intersectent aucun cors rigide, celles qui n'intersectent que des corps rigides mobiles et celles qui intersectent au moins

un corps rigide statique ;

— Niveau topologique : un graphe topologique est utilisé pour décrire la connectivité entre des lieux et des frontières (figure 2.5 -b). Ici, un arc représente un lieu à traverser et un nœud représente une frontière à atteindre ;

— Niveau sémantique : la sémantique d'un modèle d'espace libre est faite d'attributs attachés aux lieux et aux frontières. Il s'agit par exemple du niveau de complexité pour traverser un lieu, ou de la présence d'un obstacle mobile (figure 2.5 -a).

B/ Modèle des corps rigides

— Niveau géométrique : la géométrie d'un modèle de corps rigide est représentée par un modèle polygonal (sommets, bords et faces dû à la triangulation de Delaunay) ;

— Niveau sémantique : ce niveau permet de définir des attributs pour décrire des connaissances orientées vers les tâches/métiers de haut degré d'abstraction. Par exemple, les attributs sémantiques associés à un corps rigide donné peuvent décrire sa forme, sa fonction ou le fait qu'il soit statique ou potentiellement mobile (figure 2.5-a).

Stratégie de planification en deux étapes
La planification de trajectoires est composée de deux étapes, et s'appuie sur le modèle multi-niveaux ([Cailhol et al., 2019]), comme présenté sur la figure 3.2 :

— Un premier processus de planification grossière : dans un premier temps, des poids sont assignés à chaque arc du graphe topologique permettant par la suite de déterminer le plus court chemin. Par défaut, le poids est une image de la longueur de l'arc, mais peut être modifié selon la sémantique qui est assignée au lieu correspondant (par exemple, plus un lieu est complexe à traverser, plus le poids sera élevé). Puis, à l'aide d'un algorithme de recherche de chemin (A*, Dijkstra etc ...), le chemin le moins coûteux est obtenu. Ce chemin se décompose en étapes topologiques successives, chacune étant composée d'un lieu à traverser et d'une frontière à atteindre ;

— Un second processus de planification fine : un jalon, ayant une position et une orientation n'entrant pas en collision avec l'environnement, est aléatoirement tiré au niveau de chaque frontière (afin d'avoir une configuration intermédiaire, pouvant jouer le rôle de configuration initiale ou finale d'une étape topologique). Puis, le processus de planification fine appelle, pour chaque étape topologique, l'algorithme Bi-RRT pour trouver une trajectoire.

Stratégie de planification de trajectoires proposées Trois stratégies de planification de trajectoires ont été proposées :

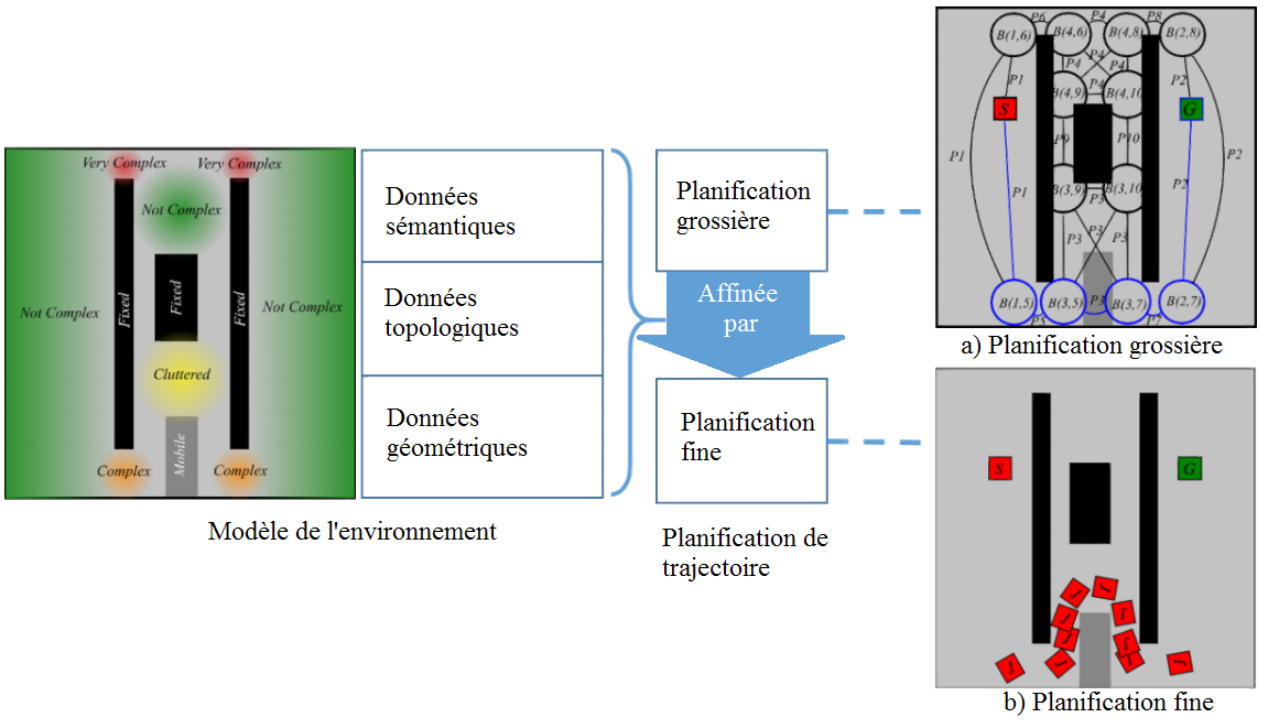


FIGURE 2.6 – Modèle de l'environnement multi-niveaux et planification de trajectoires en deux étapes ([Cailhol et al., 2015])

— Stratégie G : Cette stratégie ne considère que les données purement géométriques de l'environnement. Celle-ci est implémentée par un algorithme Bi-RRT. Le fonctionnement de cet algorithme est représenté sur la figure 2.7 dans un exemple 2D. Toutefois, le fonctionnement est identique dans un exemple 3D.
Cet algorithme construit deux arbres, à partir des configurations initiales et finales (q_{init} et q_{goal}). À chaque itération, une configuration aléatoire est tirée dans l'espace libre (q_1,q_2, q_3 et q_4). Puis chaque arbre est étendu dans la direction de la configuration tirée. Ce processus est ensuite répété jusqu'à ce que les deux arbres puissent se rejoindre. La trajectoire obtenue est finalement une suite d'arcs et de nœuds reliant q_{init} et q_{goal}.

— Stratégie GT : Cette stratégie permet d'explorer un environnement modélisé par des informations géométriques et topologiques. Celle-ci est basée sur la planification en deux étapes (planification grossière puis fine). Lors de la construction du graphe topologique, des coûts sont associés aux arcs. Ceux-ci sont dépendants de la distance entre les deux frontières connectées par un arc. Un algorithme de recherche de chemin à travers un arc (type Dijkstra ou A*) est appelé pour trouver un chemin topologique. Ensuite, un jalon géométrique est tiré aléatoirement dans chacune des frontières du chemin topologique. Celui-ci sert ensuite de configuration intermédiaire à atteindre. Enfin, pour chaque étape du chemin topologique (qui correspond à un lieu à atteindre et une frontière à traverser)

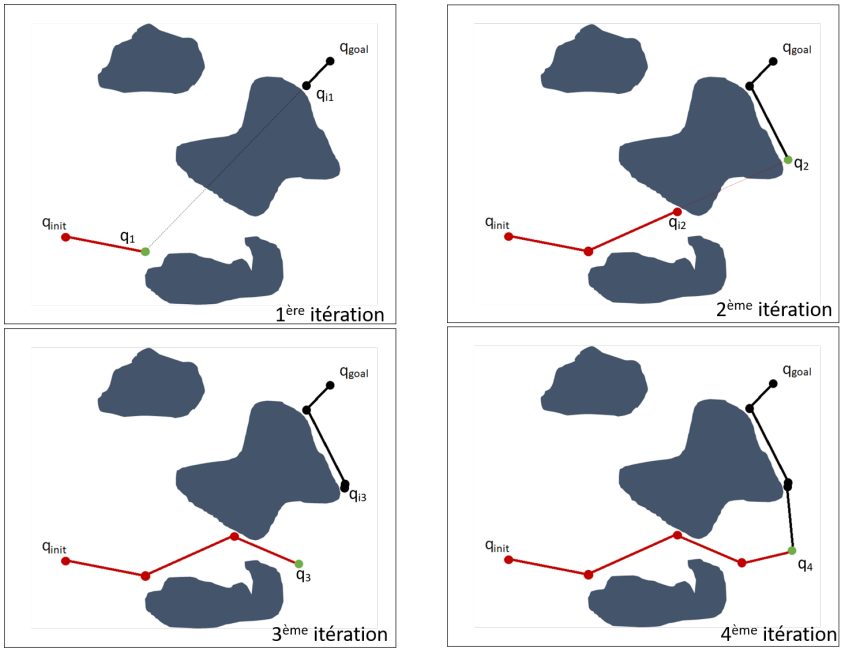
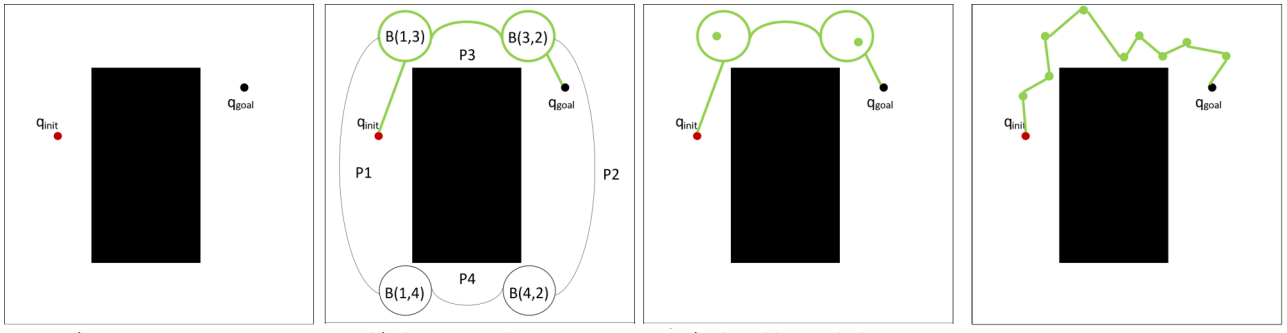


FIGURE 2.7 – Stratégie G

l'algorithme Bi-RRT est appelé afin de trouver une trajectoire réalisable.



a) Environnement 2D b) Chemin topologique c) Jalon aléatoire à chaque frontière d) Chemin géométrique (Bi-RRT)

FIGURE 2.8 – Startégie GT

— Stratégie GTS : Cette stratégie reprend le fonctionnement de la stratégie GT. Toutefois, le coût associés aux arcs n'est plus dépendant de la distances entre deux frontières mais est fonction de la sémantique. Plus précisément, le coût est dépendant de la complexité du lieu à traverser pour favoriser les endroits les plus faciles à traverser.

Finalement, ce travail a été validé sur l'application présentée sur la figure 2.5. Les résultats ont démontré qu'en utilisant des informations de plus haut niveau d'abstraction, la qualité du chemin trouvé était meilleure :

— D'un point de vue qualitatif, le chemin calculé est plus pertinent ;

— D'un point de vue quantitatif, le nombre d'échantillons tirés afin de trouver une trajectoire a été fortement réduit. Ainsi, les temps de calcul associés ont aussi été abaissés, d'un facteur dix environ.

Cependant, les informations sémantiques utilisées restent embryonnaires. De plus, la planification de trajectoires, seule, n'est pas suffisante pour valider une tâche du cycle de vie du produit.

2.3.1.3 Synthèse

Un point clef de la simulation de tâche sous de fortes contraintes géométriques est le fait de trouver une trajectoire sans collision afin de prouver la faisabilité du mouvement. Il existe pour ce faire de nombreuses méthodes de planification automatique de trajectoires. Traditionnellement, elles utilisent uniquement des données géométriques et souffrent, de ce fait, de limitations importantes. Il est alors intéressant d'utiliser des informations de plus haut niveau d'abstraction.
De plus, la planification automatique de trajectoires ne suffit pas, à elle seule, à la validation d'une tâche car ni les modèles de l'environnement ni les méthodes d'exploration de ces modèles ne s'appuient sur des informations relatives à la tâche à réaliser. Il faut donc considérer, en parallèle, la planification de tâches.

2.3.2 Planification de tâches

Afin de pouvoir proposer une assistance orientée métier, il faut pouvoir raisonner au niveau de la tâche à réaliser. Pour cela, il faut considérer, en plus de la planification de trajectoires, la planification de tâches. La planification de tâches consiste à choisir et à organiser les actions permettant d'atteindre un objectif donné ([Ghallab et al., 2016]). Elle se résout par l'utilisation d'un planificateur qui va trouver une séquence de tâches permettant d'atteindre un but. Cette solution est appelée un plan de tâches. Celui-ci est considéré comme valide si tous les buts désirés sont satisfaits et il est dit optimal (vis-à-vis d'un critère à optimiser) si la séquence trouvée est celle qui satisfait au mieux le but à atteindre.

Avant de présenter les approches développées pour la planification de tâches, il est important de définir quelques concepts importants :

— Relation : Liste de prédicats décrivant comment deux (ou plus) choses sont liées entre elles ;

— Agent : Il s'agit d'un déterminant actif animé ou inanimé d'un processus ;

— Environnement : Espace dans lequel se déroule une tâche ;

— État de l'environnement : liste de propositions décrivant la position spatiale et/ou temporelle des objets de l'environnement (où sont positionnés les objets, comment ils sont liés les uns aux autres ...) ;

— État initial : définit le premier état de l'environnement ;

— État final : définit l'état de l'environnement à atteindre ;

— Primitive : tâche pouvant être réalisée directement ([Erol et al., 1994]), par un changement d'état. Celui-ci dépend d'une ou plusieurs conditions (à remplir pour déclencher le changement d'état) et d'un ou plusieurs effets (décrivant l'état de transition après l'exécution de la primitive d'action) ([Srivastava et al., 2014]).

Nous allons maintenant présenter les approches développées dans la planification de tâches. Dans le paragraphe 2.3.2.1, la planification classique est présentée. Il s'agit d'une forme restreinte de la planification, dont la compréhension est primordiale pour la suite. Cette partie introduit également certains planificateurs / langages essentiels pour la suite. Le paragraphe 2.3.2.2 présente la planification temporelle (ainsi que certains planificateurs temporels), qui introduit la notion de temps explicite et d'ordonnancement. Le paragraphe 2.3.2.3 discute de la planification sous incertitude. Celle-ci permet de prendre en compte un environnement ou des évènements incertains. Ce paragraphe présente également certains planificateurs de tâches sous incertitudes.

2.3.2.1 Planification classique

L'idée de la planification de tâches est de générer de manière automatique un résultat sous la forme d'un plan de tâche. La planification classique impose trois hypothèses restrictives qui sont, d'après [Ghallab et al., 2016] :

— Être dans un environnement statique, fini : les ensembles d'états et d'actions doivent être finis. De plus, les changements ne se produisent qu'en réponse à des actions : si l'acteur n'agit pas, l'état actuel de l'environnement reste inchangé ;

— Pas de temps explicite : il n'y a pas de modèle explicite du temps (par exemple, quand commencer à effectuer une action, combien de temps une action doit durer, ou comment effectuer d'autres actions simultanément). Il n'y a qu'une séquence discrète d'états et d'actions ;

— Ne pas avoir d'incertitude, déterminisme du modèle : les actions produisent un résultat qui est connu. Cela exclut ainsi les résultats non déterministes comme ceux d'un lancer de dé.

Plusieurs planificateurs classiques ont été conçus :

— Stanford Research Institute Problem Solver (STRIPS) ([Fikes and Nilsson, 1971]) : conçu par Richard Fikes et Nils John Nilsson en 1971, il s'agit de l'un des premiers planificateurs. Ce nom désigne à la fois l'algorithme développé et le langage de représentation des données qui est utilisé par ce dernier. Il définit des actions sous la forme d'opérateurs auxquels sont attachés des préconditions et des effets. Cet ensemble (préconditions - effet) forme une proposition pouvant être vraie ou fausse ;

— GraphPlan ([Blum and Furst, 1997]) : développé par Avrim Blum et Merrick Furst en 1995, cet algorithme prend en entrée un problème exprimé en langage STRIPS et génère des plans complets STRIPS tout en autorisant une certaine parallélisation des actions. Cette méthode a permis d'augmenter les performances des planifications STRIPS permettant dorénavant d'envisager des applications réelles à moyen terme ;

— D'autres planificateurs ont été conçus en se basant principalement sur STRIPS ou GraphPlan. Nous pouvons citer par exemple ProPlan ([Fourman, 2000]) ou Least Committed GraphPlan (LCGP) ([Cayrol et al., 2001]) ;

— HTN ([Erol et al., 1994]) : la planification hiérarchique essaie de décomposer une tâche complexe en des tâches de plus bas niveau, jusqu'à obtenir uniquement des tâches primitives. L'algorithme cherche donc à décomposer une tâche en tâches primitives.

— FF [Hoffmann, 2001] : Développé par Jörg Hoffmann en 2001, cet algorithme utilise une approche de recherche dans l'espace des états pour trouver un plan pour un ensemble de conditions initiales et un objectif donné. Il utilise une heuristique basée sur l'analyse de causalité pour guider la recherche vers une solution optimale.

2.3.2.1.1 Méthode de recherche

La planification de tâches consiste à trouver un plan de tâches passant de l'état initial à l'état final (et donc remplir toutes les préconditions de tous les opérateurs). Il est dit cohérent s'il n'y a aucune contradiction dans l'ordonnancement des opérateurs.
Les résultats obtenus permettent de fournir un plan de tâches qui pourra par la suite être utilisé. Toutefois, il est important de réduire les temps de calcul des algorithmes afin d'obtenir des résultats efficacement. Il existe deux principales stratégies de recherche : la recherche dans l'espace des états ([Farreny, 1997]) et la recherche dans l'espace des plans ([Weld, 1994]).

Recherche dans l'espace des états Un état est une position spatiale et/ou temporelle de l'environnement. Un problème de planification peut être représenté par un graphe dans lequel les nœuds représentent les états possibles et les arcs représentent les actions à effectuer pour passer d'un état à un autre. Ensuite, un algorithme détermine un chemin pour passer de l'état initial à l'état final. La recherche se termine quand l'état courant contient l'ensemble des buts à

atteindre du problème. Toutefois, les planificateurs classiques n'offrent pas de garantie sur l'optimalité du nombre d'actions dans les plans.

Recherche dans l'espace des plans Dans la planification par recherche dans l'espace des plans, les nœuds représentent des plans partiels et les arcs représentent les opérations d'extensions. Le planificateur étend un plan initial pour obtenir une solution. La planification se termine lorsque les préconditions de toutes les actions du plan courant sont produites par d'autres actions du plan et que ce dernier ne comporte plus aucune action qui puisse interférer avec une autre. Cette approche permet d'obtenir des plans possédant un haut degré de flexibilité, ce qui permet de s'adapter efficacement à des changements imprévus.

2.3.2.1.2 Limites de la planification classique

La planification classique a permis l'émergence de méthodes efficaces pour résoudre des problèmes qui répondent à des hypothèses strictes (environnement statique, pas de temps explicite, pas d'incertitude). Il faut donc avoir une connaissance parfaite sur l'environnement et que celui-ci ne soit soumis à aucun changement. Il s'agit donc de planifier dans un monde très simplifié vis-à-vis du monde réel. Les planificateurs classiques ne sont pas capables :

— De traiter des actions avec des durées ;

— De permettre aux actions de s'exécuter simultanément ;

— De considérer des incertitudes sur l'environnement ou sur les tâches.

De plus, ces planificateurs limitent l'évaluation des plans à une fonction binaire : le plan de tâches retourne un résultat ou non. Or, dans certaines situations une évaluation plus large (par exemple le coût, le temps d'exécution, la probabilité de réussite ...) pourrait être intéressante.

2.3.2.2 Planification temporelle

La planification classique représente souvent les actions comme étant discrètes. Cela signifie que leurs préconditions sont vraies jusqu'au début de l'exécution et leurs effets deviennent vrais quand l'exécution se termine. La transition est considérée instantanée. Or, cela est une hypothèse restrictive lorsque des contraintes d'ordonnancement entrent en jeu. L'ordonnancement est le fait d'organiser, en fonction du temps, le séquencement des différentes tâches. Il peut également intervenir après le processus de planification en considérant un plan et en proposant une stratégie d'exécution en temps réel en attribuant des dates d'échéances. Il faut ici prendre en compte les potentielles contraintes temporelles comme les délais ou la disponibilité des ressources. Des planificateurs spécifiques ont alors été développés : les planificateurs

temporels.

DEVISER DEVISER ([Vere, 1983]) est un des premiers planificateurs pouvant générer des plans afin d'atteindre des objectifs qui sont contraints par le temps. Il considère des actes, des évènements et des inférences en leur assignant des spécifications temporelles. Les actes, évènements et inférences sont modélisés en accord avec STRIPS. Bien que les transitions soient instantanées, il est possible d'y associer des contraintes temporelles de durées (pouvant être définies à l'avance ou être calculées dynamiquement) ou des fenêtres d'activation (intervalles de temps pendant lequel on peut exécuter l'action).
DEVISER peut également prendre en compte des évènements qui se produisent à une date déterminée. Les contraintes qu'ils engendrent sont ensuite prises en compte dans le processus de génération de plan pour satisfaire les objectifs à atteindre.

CPT CPT ([Vidal and Geffner, 2006]) est un planificateur temporel optimal et indépendant du domaine. Il est fondé sur la programmation par contraintes qui intègre des heuristiques avec une nouvelle représentation et des règles de propagation qui parviennent à élaguer l'espace de recherche. Ce planificateur est capable de raisonner sur les relations de précédences et les liens causaux grâce à des actions qui n'appartiennent pas encore à un plan partiel. L'objectif de CPT est de trouver des plans ayant une durée d'exécution minimale. Il a été récompensé par un deuxième prix dans la catégorie de planification optimale lors de la compétition internationale de planification en 2004 ([Loukil et al., 2006].

Sapa Le planificateur Sapa ([Do and Kambhampati, 2011]) peut gérer la durée des actions et les échéances. Il s'appuie sur des méthodes basées sur le graphe de planification pour dériver des heuristiques qui sont sensibles à la fois au coût et au délai de réalisation, sur des techniques pour affiner les estimations heuristiques et sur des techniques de post-traitement pour améliorer la flexibilité de la solution. Lors de la troisième compétition internationale de planification en 2002 [6], il s'est classé parmi les meilleurs planificateurs considérant les contraintes temporelles.

Limites

Les planificateurs temporels produisent un plan qui respecte les contraintes temporelles et qui peut être exécuté en accord des délais prévus. Toutefois, ces planificateurs ne sont utilisables qu'en environnement statique et ne prennent pas en compte la possible incertitude des actions ni celles sur les durées.

6. https ://ipc02.icaps-conference.org/

2.3.2.3 Planification sous incertitudes

Des évènements imprévus peuvent avoir lieu lors de l'exécution du plan. L'incertitude est un concept qui doit être considéré lors de problèmes de planification réels. Il faut alors utiliser des techniques de planification sous incertitude. Les paragraphes suivants présentent les différents types de planification sous incertitudes.

2.3.2.3.1 Planification conditionnelle

Ce type de planification génère un plan qui peut changer en fonction des observations faites par l'agent durant l'exécution du plan. Les incertitudes peuvent venir de l'état de l'environnement, des effets des actions ou de la durée de celles-ci. La méthode est la suivante : un plan contenant toutes les possibilités est généré sous la forme d'un arbre. Chaque branche représente les observations faites lors de l'exécution. Toutefois, cet arbre peut croître très rapidement si le scénario est complexe. Ainsi, la planification conditionnelle est difficilement utilisable.

2.3.2.3.2 Planification probabiliste

La planification probabiliste génère un plan dont la probabilité de succès est très forte ou est supérieure à un seuil. Cela permet d'utiliser un planificateur dans un monde qui n'est pas parfaitement connu ou avec des actions dont les effets ne sont pas garantis. Les paragraphes suivants présentent des planificateurs couramment utilisés par la communauté.

BURIDAN BURIDAN ([Kushmerick et al., 1995]) est un planificateur probabiliste basé sur la représentation STRIPS. Les agents sont représentés par des arbres dans lesquels les feuilles correspondent aux conséquences possibles de l'action qui est à la racine de l'arbre. Les nœuds représentent l'état du monde permettant d'obtenir l'effet défini dans une feuille située au bout de sa branche et une probabilité est associée à chaque transition vers une feuille.
Les différents états sont représentés par des distributions de probabilités sur les valeurs des variables qui les définissent. Le planificateur doit ensuite générer une séquence d'actions qui permet de passer d'un état initial vers un état dans lequel le but à atteindre à de fortes chances de succès. On vérifie d'abord si la probabilité que l'objectif soit atteint est supérieure au seuil au-delà duquel un plan est acceptable. Si oui, la planification est terminée, sinon, le plan est affiné en ajoutant des agents ou en résolvant des conflits dans le but d'augmenter la probabilité de succès du plan.

PGraphPlan et TGraphPlan Le GraphPlan étant un algorithme très utilisé dans la planification, plusieurs extensions ont été faites pour pouvoir gérer les incertitudes.

PGraphPlan ([Blum and Langford, 2000]) construit le graphe de la même manière que le GraphPlan sauf que chaque action contient un ensemble de possibilités et une probabilité est associée à chacune d'elles. Ce planificateur produit un plan optimal, mais l'ajout des effets probabilistes a diminué ses performances temporelles.
Le TGraphPlan ([Blum and Langford, 2000]), lui, garde les mêmes performances que le GraphPlan mais ne fournit pas le plan optimal. À chaque étape, il choisit l'action ayant la plus grande probabilité de réussite et la probabilité totale d'un plan se calcule en multipliant cette probabilité à celle du plan partiel déjà construit.

2.3.2.3.3 Limites de la planification sous incertitude

Bien que ces planificateurs permettent de travailler dans des environnements incertains, ils rencontrent certaines limites. Parmi elles, la modélisation des incertitudes peut être difficile, ce qui peut rendre les plans inefficaces. Par ailleurs, la planification en temps réel peut être rendue difficile, car les plans élaborés sous incertitudes sont souvent conçus pour être robustes face aux incertitudes et donc, ne peuvent pas réagir efficacement aux changements.

2.3.2.4 Synthèse sur la planification de tâches

Il existe plusieurs stratégies pour la planification de tâches. Cependant, seule, elle ne peut vérifier géométriquement la faisabilité des primitives de mouvement d'un plan de tâches. Cette planification assure seulement que l'ordre des tâches est correct, sans prendre en compte la faisabilité du mouvement. Il est alors nécessaire d'associer conjointement planification de trajectoires et planification de tâches.

2.3.3 Planification conjointe de tâches et de trajectoires : TAMP

Dans cette partie, nous nous intéressons uniquement aux études combinant planification de tâches et de trajectoires, sans considération pour les types de planificateurs utilisés. Ainsi, nous allons présenter dans un premier temps comment peut être construite une approche TAMP régulière (paragraphe 2.3.3.1). Par la suite, nous allons décrire la planification TAMP en ligne, permettant de résoudre des problèmes plus complexes (paragraphe 2.3.3.2). Nous allons également discuter de la planification TAMP et de l'apprentissage, afin d'améliorer les performances de la planification (paragraphe 2.3.3.3). Enfin, nous allons décrire les deux principales stratégies utilisées dans la littérature de planification pour le TAMP (paragraphe 2.3.3.4).

2.3.3.1 Planification TAMP régulières

Le TAMP est définit comme une méthode de planification traitant les contraintes de la planification de trajectoires de bas niveau comme des facteurs de planification de tâches de haut niveau

[Guo et al., 2023]. Il faut donc étudier comment s'intègre la planification de tâches à la planification de trajectoires (2.3.3.1.1) et comment le rendre efficace en termes de calcul 2.3.3.1.2.

2.3.3.1.1 Méthodes de planification TAMP

Nous pouvons classer les méthodes de TAMP selon la façon dont elles sont construites. Ces méthodes peuvent être construites selon une architecture hiérarchique ou construite selon des relations de contraintes dans le TAMP.

Planification TAMP basée sur la hiérarchie Ces approches décomposent les tâches de haut niveau en de nombreuses sous-tâches pouvant facilement être résolues. Une structure hiérarchique peut alléger l'effort de calcul nécessaire pour combiner le séquençage des tâches et la vérification de la faisabilité en résolvant des problèmes TAMP de taille plus petite.
[Kaelbling, 2011] propose une planification hiérarchique pour le TAMP. Il intègre des informations géométriques à la planification de tâches pour construire des choix appropriés pour les paramètres de l'opérateur en utilisant des « *suggesters* » géométriques. Cette méthode décompose continuellement les tâches de haut niveau en actions primitives concrètes par des procédures emboîtées.
Une autre stratégie qui décompose hiérarchiquement les tâches abstraites de haut niveau est proposée par [Agostini et al., 2020]. Cette méthode utilise une représentation des contraintes géométriques centrées sur l'objet pour générer des solutions cohérentes, ce qui permet au planificateur de trajectoires de ne pas faire des calculs inutiles. Les tâches sont décomposées en nœuds d'arbre et correspondent à des actions du robot. Le processus allant du planificateur de tâches à certaines actions est articulé au travers d'une nouvelle structure appelée contexte d'action. Celle-ci consiste en des informations sur l'action suivante et sur l'action qui a été exécutée avant.

Planification TAMP basée sur des contraintes Ces approches considèrent la combinaison du séquençage des tâches et des contraintes géométriques de bas niveau comme des Problèmes de Satisfaction de Contraintes (CSP). Ces derniers sont ensuite traités par des solveurs spéciaux.
[Lagriffoul and Andres, 2016] proposent la planification et l'évaluation des contraintes géométriques qui sont imbriquées dans le cadre du TAMP pour générer des séquences d'actions symboliques.
Un planificateur appelé Iteratively Deepened Task and Motion Planning (IDTMP) utilise un solveur de Satisfiabilité Modulo des Théories (SMT) pour vérifier la faisabilité du mouvement lors de la planification de tâches ([Dantam et al., 2016]). Cette méthode trouve un plan de tâches en incorporant les informations sur les contraintes géométriques venant de la vérification de la faisabilité des mouvements qui ont échoué, de manière incrémentale. Une version étendue

([Dantam et al., 2018]) améliore la communication entre les planificateurs de tâches et de trajectoires. Les contraintes peuvent être construites sous la forme d'un graphe de contraintes en unifiant la planification des tâches et la planification des trajectoires selon des règles de réarrangement ([Mirabel and Lamiraux, 2016]). Un algorithme, "Manipulation RRT" est proposé, basé sur le graphe de contraintes pour les problèmes Navigation Among Movable Obstacles (NAMO) et Manipulation Among Movable Obstacles (MAMO).

2.3.3.1.2 Optimisation de la planification TAMP

L'approche TAMP considère une planification de tâches de haut niveau qui est moins coûteuse que la planification de trajectoires de bas niveau. Cela est dû à l'espace de recherche discret qui est en jeu dans la planification de tâches. Il est important de chercher à améliorer l'efficacité des calculs qui entrent en jeu dans la planification.
[Zhang and Shah, 2016] proposent une approche TAMP basée sur l'optimisation hiérarchique. Les applications sont formulées comme des problèmes d'optimisation à plusieurs niveaux (au niveau de la planification des tâches, des actions et des trajectoires). L'approche détermine une séquence optimale de mesures de performances au niveau de la planification de tâches en utilisant un solveur Travelling Salesman Problem (TSP). Cela modélise le coût de migration d'une tâche à une autre. Concrètement, si un objet en bloque un autre, le coût de transition sera infini.
[Akbari et al., 2019] propose une méthode TAMP retardant la vérification de la faisabilité géométrique jusqu'à ce que l'action correspondante soit sélectionnée par l'heuristique. Cela réduit le coût de calcul du planificateur de trajectoires tout en réduisant les retours en arrière puisque le planificateur de tâches considère la plupart des contraintes géométriques (dont les configurations de prises et de poses), les solutions de cinématique inverse et le contrôle de collision.
D'autres méthodes utilisent une approche basée ontologie pour réduire les coûts de calcul de la planification de trajectoires. ([Zhao et al., 2018]) propose une stratégie pour le couplage sémantique de la planification de trajectoires et d'une action primitive (d'un plan de tâches) pour la simulation de tâches. Cette approche est basée sur deux ontologies (ENVOn et Action Specification Knowledge (ASK)), qui sont présentées dans le paragraphe 2.3.4.3.2). L'approche proposée permet de définir automatiquement des requêtes de planification de trajectoires pour une action primitve ainsi que des contraintes géométriques liées à la tâche sur ces requêtes. De plus, deux stratégies de planification de trajectoires sont proposées :

— Stratégie GO : Basée sur la stratégie G présentée dans le paragraphe 2.3.1.2.2, les configurations aléatoires sont contraintes selon les Contraintes Géométriques (CG) inférées à l'aide de l'ontologie ASK;

— Stratégie GTO : Basée sur la stratégie GTS présentée dans le paragraphe 2.3.1.2.2, lors de la planification fine, les tirages des configurations aléatoires s'effectuent sous des CG qui

sont inférées, à l'aide de l'ontologie ASK, à partir de Contraintes Spatiales (CS) définies préalablement par un opérateur humain. Les CG sont utilisées pour contrôler le tirage des configurations de l'algorithme Bi-RRT (figure 2.9). Ce contrôle s'effectue dans des sphères (pour un environnement 3D) centrées sur la position finale et sur les différentes frontières du chemin topologique retenu.

Toutefois cette approche a des limites : une seule tâche primitive est considérée et non pas un plan de tâche complet, l'expression des CS et des CG n'est pas générique, et l'assignement des CS n'est pas automatique.

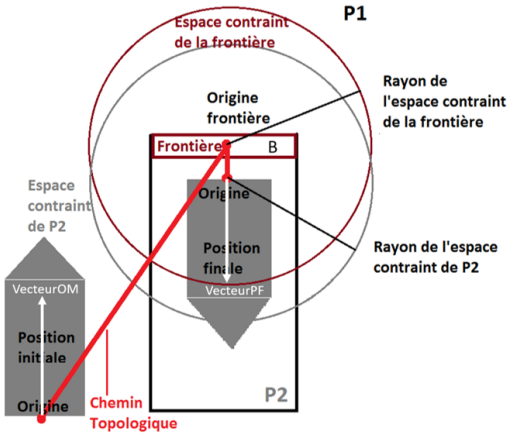


FIGURE 2.9 – Stratégie GTO

2.3.3.1.3 Synthèse

La planification TAMP est de plus en plus étudiée. Il s'agit de faire collaborer efficacement planification de tâches et de trajectoires. Bien qu'elle soit majoritairement utilisée pour des robots manipulateurs, de nouvelles applications sont demandeuses d'une telle approche comme la robotique mobile, l'exploration et la construction architecturale, ou encore la RV. Ces nouvelles applications arrivent avec de nouveaux besoins, comme la planification en Temps Réel (TR) ou la planification en environnement incertain ou dynamique. De nouvelles approches doivent alors être utilisées.

2.3.3.2 Planification TAMP en ligne

Les méthodes ordinaires de planification TAMP sont souvent associées à une ou plusieurs hypothèses telles que l'observabilité totale du monde, la planification en environnements statiques, une base de connaissances complète et des actions dont tous les effets sont connus. Toutefois, il est difficile d'exécuter une tâche à long terme dans un environnement réel. Ainsi, de nouvelles

méthodes prenant en compte les incertitudes ont été développées. Les approches de planification du comportement robustes et réactives liées à l'exécution d'actions sont présentées dans le paragraphe 2.3.3.2.1 et les approches TAMP basées sur le contrôle sont discutées dans le paragraphe 2.3.3.2.2.

2.3.3.2.1 Planification des comportements

Après avoir trouvé une séquence d'actions, l'étape d'après consiste à exécuter ces actions. La manière la plus simple d'implémenter des actions est d'envoyer directement des commandes aux planificateurs de mouvement. Toutefois, ces implémentations peuvent échouer dès lors qu'une perturbation intervient dans l'environnement. Ainsi, un exécuteur d'action spécial et un mécanisme de re-planification des actions sont nécessaires.

HTN
Les Hierarchical Task Network (HTN) servent à la résolution de tâches complexes. Ils peuvent servir comme planificateur de tâches mais peuvent aussi gérer l'exécution de chaque opérateur. [Weser et al., 2010] utilisent les HTN pour gérer des scénarios complexes en robotique sans hypothèse d'un monde fermé. Il intègre les HTN à un module de commande pour contrôler les actions primitives. Cette méthode génère un squelette de plan même si la représentation de l'environnement est incomplète tout en conservant les propriétés du planificateur HTN.

Arbres de comportement
Les arbres de comportement font également partie de la planification des comportements. Ils peuvent soient être fait manuellement par un opérateur humain, soient générés automatiquement par des planificateurs symboliques afin de résoudre des tâches complexes.
[Rovida et al., 2017] combinent les comportements d'exécution avec le domaine de planification pour générer un arbre de comportements étendus (eBT).

2.3.3.2.2 Planification TAMP basée sur le contrôle

Les approches TAMP basées sur le contrôle séparent généralement le système de planification en deux couches, une couche TAMP et une couche d'action appelée Planning-Acting Loop System (P-ALS). La couche de planification génère une séquence d'actions réalisables avec des conseils détaillés pour la couche d'action, puis celle-ci exécute ces actions. Des interactions efficaces et en temps réel entre les couches de planification et d'action ont une influence majeure sur les valeurs d'utilisation pratique du cadre TAMP. La figure 2.10 présente une structure classique de l'approche TAMP basée sur le contrôle combinant la couche de planification et la couche d'action. L'estimation de l'état est effectuée après l'exécution de chaque action. Si un

échec ou un changement de l'état inattendu est détecté, le processus réagit rapidement et génère un plan modifié sur la base du précédent.

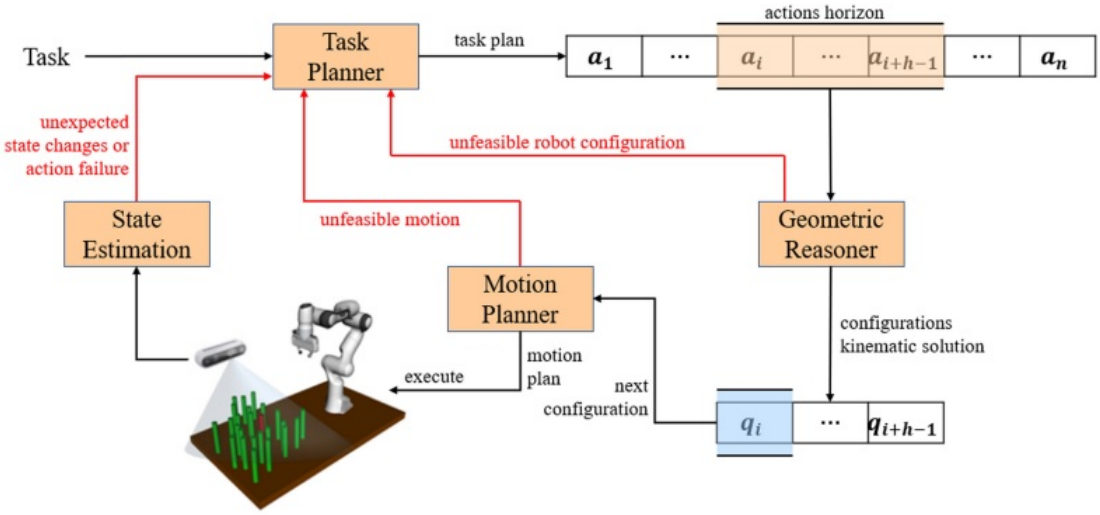


FIGURE 2.10 – Un cadre TAMP classique basé sur le contrôle ([Castaman et al., 2021])

[Dantam et al., 2018] ont développé un cadre open-source pour le TAMP nommé TMKit. Il s'agit d'un système pour la planification probabiliste complète et d'action instantanée. TMKit peut intégrer différentes méthodes de planification. La séquence d'action finale pour réaliser une tâche suit le processus d'exécution en interpolant des valeurs au planificateur de trajectoires et en exécutant les actions correspondantes. Un retour est utilisé pour contrôler l'erreur de positionnement pendant l'exécution.
Un algorithme Receding Horizon Task and Motion Planning (RH-TAMP), qui est un algorithme de planification en ligne à horizon glissant a été proposé par [Castaman et al., 2021]. Il est capable de gérer les perturbations qui surviennent dans un environnement. Cet algorithme ne se limite pas aux algorithmes à plan unique mais définit un horizon d'actions comme une séquence partielle du plan complet. Chaque action de l'horizon d'action est raffinée itérativement par le raisonneur géométrique. Lorsque la première action peut être exécutée, l'algorithme avance la fenêtre de l'horizon d'actions.

2.3.3.2.3 Synthèse

Les approches présentées dans cette partie permettent d'améliorer le TAMP, notamment dans des environnements dynamiques ou ayant des connaissances incomplètes sur celui-ci. Toutefois, ces approches pourraient également être améliorées en utilisant des approches d'apprentissages purs, comme l'apprentissage par renforcement.

2.3.3.3 Planification TAMP et apprentissage

Le TAMP a pour but de trouver une séquence d'actions pour planifier une tâche. Il est important de regarder s'il est possible de tirer parti d'algorithme d'apprentissage afin d'améliorer les performances du TAMP. L'apprentissage pour la planification symbolique est présentée dans le paragraphe 2.3.3.3.1 tandis que les approches contraintes par la tâche sont discutées dans le paragraphe 2.3.3.3.2

2.3.3.3.1 Apprentissage pour la planification symbolique

La plupart des méthodes régulières du TAMP que nous avons présentées auparavant n'ont pas la capacité d'apprendre à partir de procédures de planification similaires ou d'expériences passées, alors que cette capacité peut réduire considérablement l'effort de calcul. Cependant, les méthodes d'apprentissage pur sont souvent inefficaces en termes de données dans les problèmes à long terme tels que AlphaGo Zero ([Silver et al., 2017]) avec une fonction de valeur et une politique apprise à partir d'expériences passées. Une approche prometteuse pour équilibrer les méthodes régulières du TAMP et les méthodes d'apprentissage pur consiste à trouver des représentations appropriées pour les trajectoires et les plans afin de guider la planification. [Kim et al., 2018] proposent un algorithme d'apprentissage qui utilise un score pour évaluer la qualité d'un ensemble de contraintes correspondant à une instance de problème. La méthode apprend à prédire les contraintes en définissant un sous-ensemble de paramètres de décision sur l'espace de recherche. La méthode transfère les connaissances des expériences passées à l'instance actuelle pour en tenir compte.
Une autre méthode est l'apprentissage par mimétisme. [Diehl et al., 2021] proposent une méthode pour générer automatiquement des représentations du domaine à partir de démonstration humaine. Un opérateur est généré à partir des préconditions et des effets. Compte tenu d'un objectif défini par l'utilisateur, les opérateurs appris avec un critère d'optimisation sont utilisés par un planificateur symbolique pour rechercher des solutions réalisables.

2.3.3.3.2 Apprentissage pour la planification de trajectoires contraint par la tâche

Guider efficacement la planification de trajectoires à partir d'un apprentissage est également une question mérite d'être étudiée. Étant donné que le contrôle de faisabilité de bas niveau du TAMP est coûteux, les algorithmes d'apprentissage peuvent réduire les efforts de calcul des procédures d'échantillonnage ou d'optimisation de la trajectoire de manière efficace s'ils apprennent des expériences passées ou des démonstrations d'experts.
[Bowen and Alterovitz, 2014] utilisent un planificateur de trajectoires guidé par un modèle de tâches, en boucle fermée, qui échantillonne des trajectoires sans collision en détectant les obstacles de manière efficace. Le modèle de tâche est appris à partir d'un ensemble de démonstra-

tions d'experts et peut être généralisé à de nouveaux environnements pour la planification de trajectoires, même en présence d'obstacles mobiles.
Un cadre de planification appelée Constrained Motion Planning Networks X (CoMPNetX) est proposé par [Qureshi et al., 2022]. Il est composé d'un générateur neuronal conditionnel, d'un discriminateur, d'un opérateur de projection neuronale et d'échantillonneurs neuronaux. Le générateur et le discriminateur sont conditionnés par les représentations des observations de la tâche et de la scène. Ce cadre génère des configurations et travaille avec un algorithme de planification de trajectoires pour trouver des solutions.
[Loula et al., 2020] proposent un cadre pour l'apprentissage d'un modèle modèle TAMP basé sur les contraintes. Le modèle décompose une tâche en ensembles de contraintes pour un problème d'optimisation de la trajectoire de bas niveau. Chaque ensemble de contraintes correspond à un mode particulier, par exemple, un contact entre une pince et un objet. Les démonstrations observées au cours de la formation sont supposées présenter la même séquence de modes jusqu'à ce qu'un commutateur soit appelé. Le modèle est entraîné par descente de gradient pour minimiser la fonction de perte, ce qui correspond à la fonction paramétrée des contraintes de mode.

2.3.3.3.3 Synthèse

Les principaux objectifs de combiner planification TAMP et apprentissage sont la généralisation des approches dans le TAMP, la réduction des efforts de calcul et l'apprentissage autonome des modèles. Toutefois, une telle collaboration est encore nouvelle dans la communauté.

2.3.3.4 Stratégies de planification TAMP

Il existe ainsi de nombreuses façons d'appréhender la planification conjointe de tâches et de trajectoires. Toutefois, les stratégies pour lier les deux planificateurs peuvent être regroupées en deux catégories :

— Planification de tâches puis planification de trajectoires, de manière itérative :
Le point clef de cette stratégie est que la faisabilité des primitives de mouvement du plan de tâches est vérifiée après que l'ensemble du plan de tâches ait été construit. Ainsi, si toutes les tâches sont géométriquement faisables, le plan de tâches est validée, sinon le planificateur de tâches cherchera une alternative au plan de tâches qui a échoué.
Par exemple, ([Erdem et al., 2011]) cherche à trouver une trajectoire réalisable pour des robots manipulateurs ayant le même objectif, ces derniers pouvant effectuer des tâches en parallèle. Un tel problème ne peut se résoudre uniquement par de la planification de trajectoires, car l'ordre des actions est important. En revanche, cela peut être réalisable en calculant d'abord un plan de tâches qui est ensuite vérifié avec un planificateur de tra-

jectoires (ici l'algorithme RRT). Plus récemment, ([Srivastava et al., 2014]) utilise une approche qui corrige le plan de tâches quand la planification de trajectoires échoue. Quand une erreur est trouvée dans un plan de tâche, ce dernier est modifié là où l'erreur s'est produite, permettant de réduire le nombre d'itérations. Le processus est continu jusqu'à ce qu'un plan de tâches réalisable soit trouvé.
Un autre algorithme, IDTMP ([Dantam et al., 2016]), utilise un planificateur de tâches basé sur des contraintes pour trouver un plan de tâches candidat. Ensuite, un planificateur de trajectoires est utilisé pour vérifier sa faisabilité. Si le planificateur de trajectoires échoue, des contraintes sont ajoutées pour qu'un plan de tâches alternatif soit trouvé.

— Utilisation de la planification de trajectoires durant la planification de tâches :
Le point clef de cette stratégie est que la faisabilité des primitives de mouvement du plan de tâches est vérifiée durant la construction du plan de tâche. Ainsi, si la primitive de mouvement est faisable, le planificateur passe à l'action suivante dans la même branche, sinon la branche est abandonnée ([Caldiran et al., 2009], [Wolfe et al., 2010]).
([Kaelbling and Lozano-Pérez, 2011]) présente une approche qui décompose de manière régressive l'objectif en sous-objectif et forme un niveau hiérarchique d'abstraction. Le planificateur de tâches résout les sous-objectifs à chaque niveau et les regroupe dans un plan de tâches final.

2.3.3.5 Synthèse sur la planification TAMP

Indépendamment, la planification de tâches et la planification de trajectoires ont été bien étudiées. Les études pour les combiner sont encore récentes, mais permettent de générer des solutions détaillées et efficaces, notamment dans le cadre de la robotique de manipulation.
Cependant, cette collaboration peut être améliorée. En effet, de nouvelles applications sont demandeuses du TAMP, avec de nouveaux besoins, comme la navigation ou la RV. De plus, la planification multi-agents est relativement nouvelle, et des travaux sont nécessaires pour l'améliorer. Il est possible de citer comme exemple la coopération homme - robot pour effectuer des tâches collaboratives. Ici, la communication entre les deux entités est importante, et le robot doit pouvoir prédire les intentions de l'humain.
Dans ce cas précis, les deux entités collaborant sont très différentes et peuvent utiliser un vocabulaire qui leur est propre. Il est alors nécessaire d'utiliser un modèle de connaissances partagé pour que la collaboration puisse se faire efficacement.

2.3.4 Les ontologies comme modèle de connaissance

Il est important de pouvoir construire un modèle de connaissance évolutif de l'information liée aux tâches. Utiliser un modèle de connaissances permet, lors d'un changement d'application,

de pouvoir mettre à jour certaines informations. Ceci est important pour qu'un planificateur de trajectoires puisse fonctionner correctement et de manière efficace ([Vassev and Hinchey, 2012]). Les ontologies sont un outil prometteur pour représenter la connaissance et raisonner sur les tâches à réaliser.
Le paragraphe 2.3.4.1 présente ce qu'est l'interopérabilité et pourquoi cela est très important pour le TAMP. Le paragraphe 2.3.4.2 définit les ontologies et leurs structures. Le paragraphe 2.3.4.3 présente les différents niveaux d'abstraction que peut avoir une ontologie. Enfin, le paragraphe 2.3.4.4 décrit différentes méthodologies pour la construction d'une ontologie.

2.3.4.1 Définition de l'interopérabilité

L'interopérabilité peut se définir comme « la capacité pour deux, ou plus, composants logiciels à coopérer malgré les différences de langage, d'interface et de plate-forme d'exécution. » [Wegner, 1996]). Cela doit permettre à différents systèmes de pouvoir communiquer afin d'accomplir une tâche commune. [Chen, 2006] présente dans ses travaux les différentes barrières à l'interopérabilité auxquelles sont notamment confrontées les entreprises (figure 2.11). Celles-ci peuvent être classées en trois catégories : 1) les barrières conceptuelles (différences syntaxiques, sémantiques des informations à échanger), 2) les barrières technologiques (incompatibilités des technologies de l'information) et 3) les barrières organisationnelles (responsabilité et autorité, c'est-à-dire qui est responsable de quoi ? Qui peut faire quoi ?).
Ces barrières peuvent être retrouvées aux différents niveaux d'une entreprise :

— Au niveau des données : il s'agit de faire fonctionner ensemble des systèmes utilisant des données ou des langages différents. Cela consiste à trouver et à partager des informations venant de bases de données variées, potentiellement sur des machines différentes ;

— Au niveau des services : ce niveau permet d'identifier, de décomposer et de faire fonctionner ensemble différentes applications qui ont été conçues et développées de manière indépendante ;

— Au niveau des processus : l'objectif est de faire fonctionner différents processus ensemble ;

— Au niveau des entreprises : il s'agit de travailler en harmonie au niveau de l'organisation de l'entreprise malgré des divergences, par exemple, sur le mode de prise de décision, des méthodes de travail, des approches commerciales etc ... pour que différentes entreprises puissent travailler ensemble.

De plus, l'auteur propose différentes méthodes pour surmonter ces barrières. Il existe trois méthodes pour lier des entités entre elles. La première approche, dite « intégrée », est le cas où il existe un format commun pour tous les modèles considérés. L'approche « unifiée » est une approche dans laquelle il existe un format commun, mais uniquement à un niveau « méta » (de haut niveau). Enfin, la dernière approche, dite « fédérée », ne considère pas de format commun. Dans ce cas, les différentes entités doivent partager une ontologie commune. Celles-ci

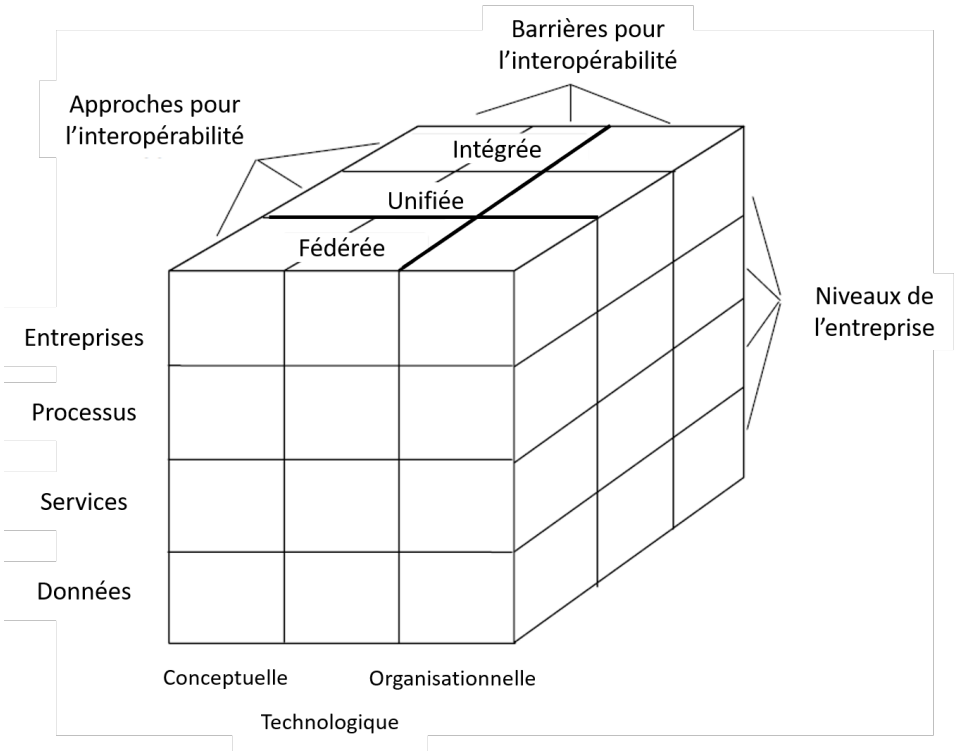


FIGURE 2.11 – Cadre d'interopérabilité de l'entreprise

peuvent être utilisées pour améliorer la communication entre humains et/ou ordinateurs en spécifiant la sémantique utilisée dans le processus de communication. Plus spécifiquement, [Uschold, 1999] a identifié trois utilisations majeures pour les ontologies : a) assister la communication entre êtres humains, b) réaliser l'interopérabilité entre les logiciels et c) améliorer le design et la qualité des logiciels.

2.3.4.2 Définition et structure d'une ontologie

Dans cette partie, nous allons définir ce qu'est une ontologie (paragraphe 2.3.4.2.1) ainsi que les différents éléments qui la composent (paragraphe 2.3.4.2.2).

2.3.4.2.1 Définition

Une ontologie peut être définie comme « une conceptualisation du monde avec un point de vue spécifique en fonction de ce que l'on souhaite représenter » ([Gruber, 1993]). Une ontologie définit des concepts ainsi que les relations qui les lient. Les ontologies sont exprimées dans un langage logique, de sorte que des distinctions précises, cohérentes et significatives peuvent être faites entre les classes, les instances, les propriétés, les attributs et les relations pour révéler les connaissances implicites et cachées afin de comprendre la signification des données.

2.3.4.2.2 Structure

La définition des principaux composants d'une ontologie est donnée en suivant :

— Les classes : elles représentent les concepts d'un domaine et leurs propriétés. Une classe peut représenter un objet physique (comme une table), un concept abstrait (comme la pensée) ou un ensemble d'action (comme un processus) ;

— Les relations : elles établissent une connexion binaire entre deux entités (concept ou instance). [Arp, 2015] a classifié trois relations binaires de base utilisées dans une ontologie qui sont présentées dans le tableau 2.2 :

Relation	Définition
Relation concept-concept	Relation généralement du type « est un sous-type de ». Il y a donc une hiérarchie générale enfant-parent. D'autres types sont cependant possibles tels que « est une partie de ».
Relation concept-instance	Instanciation du concept. Par exemple, Rox est un renard, Rouky est un chien.
Relation instance-instance	Relation permettant de décrire en détail une instance d'un concept.

TABLE 2.2 – Relations utilisées dans les ontologies

— Les axiomes : il s'agit de propositions qui représentent la connaissance qui est toujours vraie. Ils peuvent définir la signification de composants, les restrictions sur la valeur d'attributs ou encore vérifier la validité d'informations ;

— Les instances : elles sont des individus qui appartiennent à une classe.

2.3.4.3 Les niveaux d'abstractions

Il existe trois principaux niveaux d'abstractions pour les ontologies : les ontologies de haut niveau, (Top-Level Ontology (TLO)), les ontologies de domaine et les ontologies d'application ([Guarino, 1998]).
Premièrement, une ontologie de haut niveau peuvent être définies comme une ontologie qui « est utilisée pour faciliter l'intégration sémantique des ontologies de domaine et guider le développement de nouvelles ontologies. À cette fin, elles contiennent des catégories générales qui sont applicables à de multiples domaines. Les ontologies de haut niveau fournissent généralement des définitions et des axiomes riches pour leurs catégories. Différentes TLO fournissent différentes distinctions basées sur les types d'entités qu'elles incluent, les théories de l'espace et du temps ainsi que les relations des individus à l'espace et au temps » ([Hoehndorf, 2010]). Il

s'agit donc d'ontologies explicitant des termes généraux et qui servent de base pour construire des ontologies plus spécifiques, les ontologies de domaines. Celles-ci sont des représentations d'un domaine particulier. Enfin, les ontologies d'applications décrivent des concepts dépendant à la fois d'un domaine et d'une application spécifique. Elles correspondent souvent aux rôles joués par les entités du domaine lors de l'exécution d'une certaine activité, comme unité remplaçable ou composant de rechange ([Guarino, 1998]).

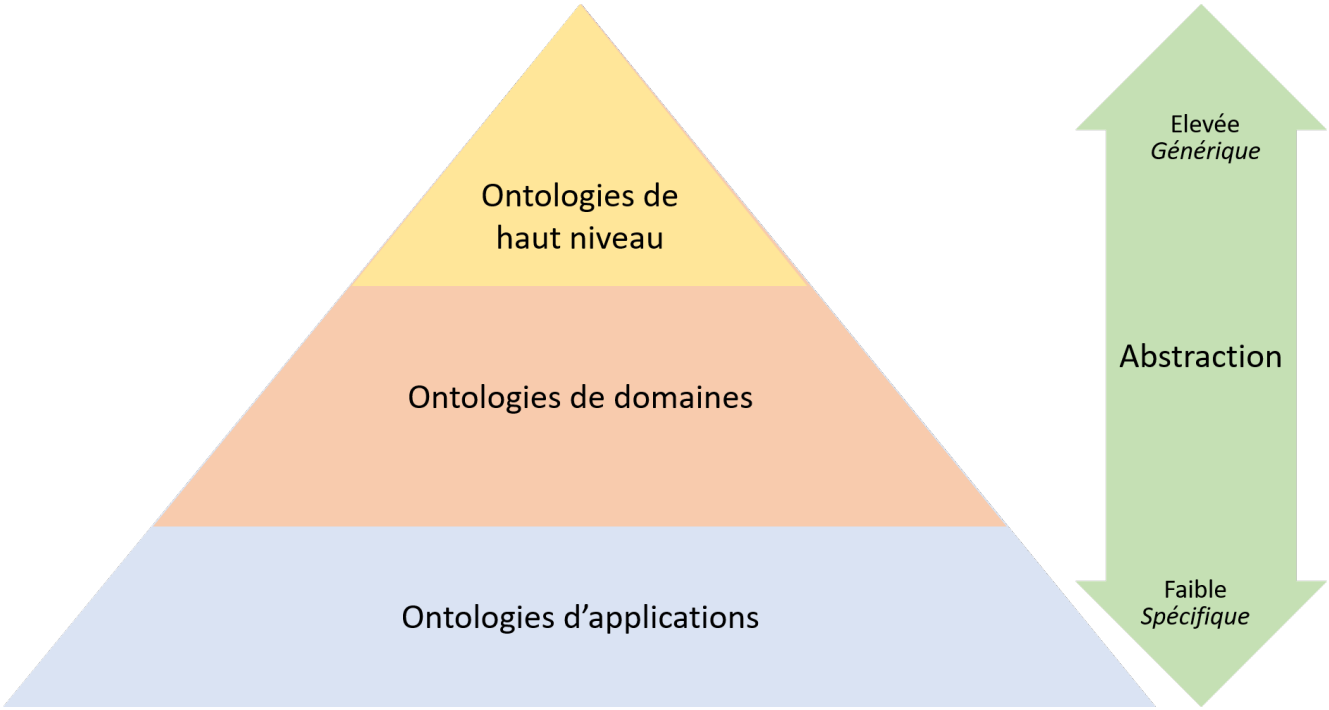


FIGURE 2.12 – Niveau d'abstraction d'une ontologie

Ainsi, dans la partie 2.3.4.3.1, certaines TLO particulièrement utilisées vont être présentées. Ensuite, des ontologies de domaines, intéressantes pour les travaux de cette thèse, vont être présentées dans la partie 2.3.4.3.2.

2.3.4.3.1 Ontologies de haut niveau

Une TLO est une ontologie qui a été créée pour représenter des catégories partagées par un éventail aussi large que possible de domaines. Elle décrit des concepts abstraits comme les objets, les processus ou encore les évènements.
De nombreuses ontologies de haut niveau ont été construites.

Basic Formal Ontology (BFO) ([Smith et al., 2005]) : projet lancé en 2002 et sponsorisé par la Fondation Volkswagen, il s'agit d'une ontologie destinée à être utilisée pour soutenir la recherche, l'analyse et l'intégration d'informations dans des domaines scientifiques et autres.

Elle ne contient pas de termes spécifiques tels que des termes physiques, chimiques ou biologiques par exemple, il s'agit d'une ontologie qui est neutre par rapport aux domaines. Elle a été conçue pour représenter à un très haut niveau de généralité les types d'entités qui existent dans le monde et les relations qui existent entre elles. Elle est également le point de départ de plus de 250 ontologies de domaines.

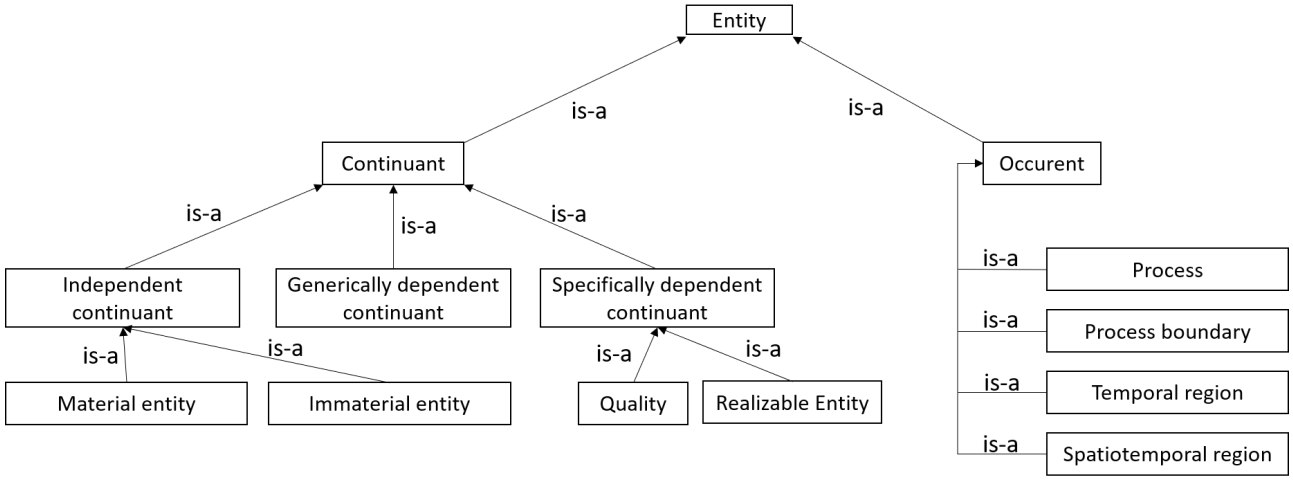


FIGURE 2.13 – Ontologie BFO

Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) ([Gangemi et al., 2002]) : développée et maintenue par ISTC-CNR Laboratory for Applied Ontology, elle était à l'origine développée dans le cadre WonderWeb [7] et a été conçue comme le premier module de WonderWeb Fundational Ontologies Library (WFOL). C'est une ontologie orientée vers la capture des catégories ontologiques qui sous-tendent le langage naturel et le sens commun humain. Elle n'a pas été conçue pour être la plus universelle ou standard possible. Les catégories qu'elle introduit sont plutôt considérées comme des artefacts cognitifs, qui dépendent finalement de la perception humaine, des empreintes culturelles et des conventions sociales.

Suggested Upper Merged Ontology (SUMO) ([Niles and Pease, 2001]) : développée en 2000 par le groupe de travail Standard Upper Ontology (SUO) qui est approuvé par l'Institute of Electrical and Electronics Engineers (IEEE) et qui est composé de chercheurs de différents domaines tels que l'ingénierie, la philosophie et les sciences de l'information, cette ontologie propose des définitions pour des termes d'usages généraux qui ont l'intention d'être étendus par des ontologies plus spécifiques (principalement par des ontologies de domaines). SUMO propose des concepts de haut niveau (qui ne sont pas spécifiques à un domaine en particulier). Un aperçu de

7. https ://cordis.europa.eu/project/id/IST-2001-33052

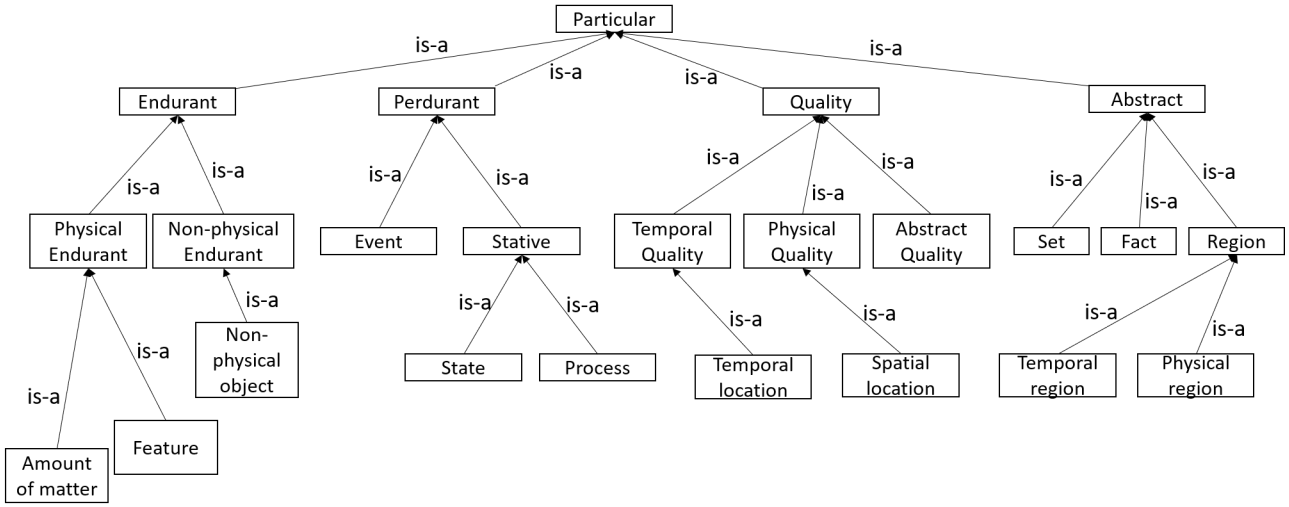


FIGURE 2.14 – Ontologie DOLCE

l'ontologie SUMO est montré dans la figure 2.15. SUMO et ses ontologies de domaines constituent la plus grande ontologie publique formelle existant aujourd'hui [8].

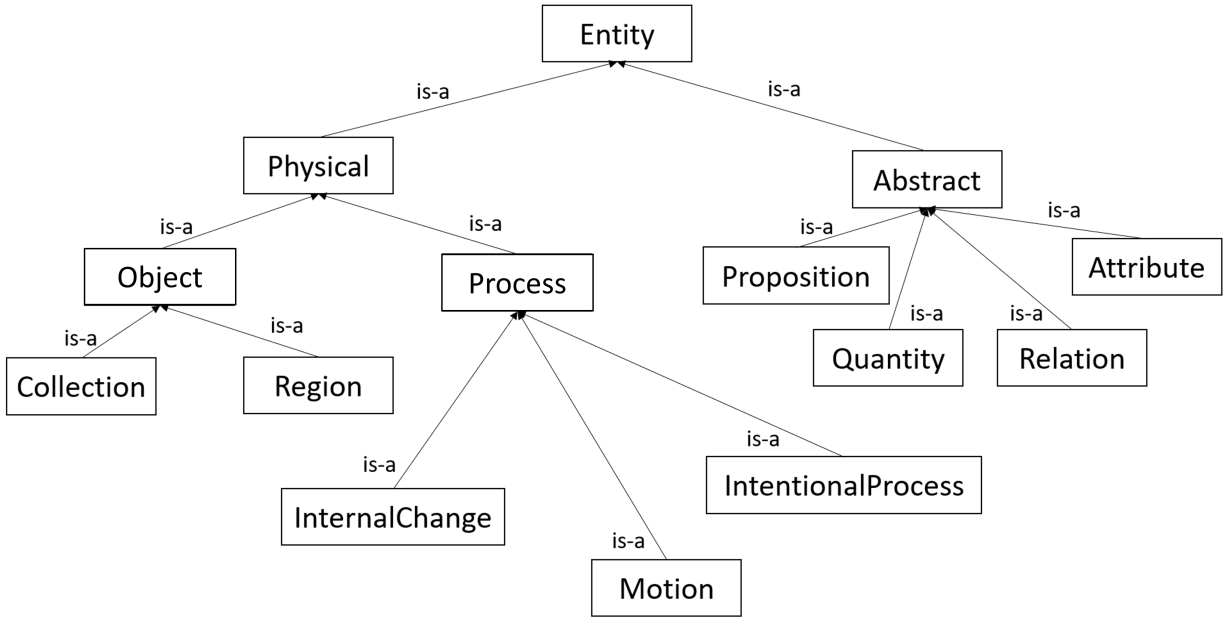


FIGURE 2.15 – Ontologie SUMO

Pour conclure cette section 2.3.4.3.1, nous pouvons dire que l'un des avantages de réutiliser une TLO est le fait de pouvoir profiter des nombreux concepts déjà définis et de la logique qui y est intégrée. De plus, cela permet de faciliter l'intégration de plusieurs ontologies de domaine. En d'autres mots, il est plus facile de faire collaborer plusieurs ontologies de domaines si elles

8. https ://www.ontologyportal.org/

sont toutes issues de la même TLO. De la même manière, concevoir une nouvelle ontologie de domaine est plus simple si elle étend une TLO. Enfin, commencer à partir d'une ontologie de haut niveau permet d'éviter d'avoir plusieurs ontologies de domaines incompatibles.

2.3.4.3.2 Ontologies de domaines liées à nos recherches

Une ontologie bien construite a pour but d'être facilement réutilisable et de pouvoir être étendue. Ainsi, il est important de regarder les ontologies liées à nos travaux, à savoir la modélisation de l'environnement et la planification de trajectoires et/ou de tâches. Au vu de l'importance des TLO, il est aussi nécessaire de regarder si ces ontologies ont été conçues à partir d'une TLO.

Ontologies de domaines liées à la modélisation de l'environnement Afin d'améliorer la planification de trajectoires, la partie 2.3.1.2 a démontré l'importance de considérer des informations de différents niveaux d'abstraction. En s'appuyant sur ce cadre proposé, il est alors intéressant de voir les ontologies modélisant ces différents niveaux d'informations, à savoir les informations géométriques, topologiques et sémantiques.

KnowRob ([Tenorth and Beetz, 2009]) : cette ontologie est utilisée pour conceptualiser le monde dans lequel le robot évolue. Elle a été modifiée récemment pour être alignée sur des concepts plus abstraits définis dans une ontologie fondamentale soigneusement conçue et dérivée de ces concepts : DOLCE+DnS Ultralite (DUL), qui est une version allégée de l'ontologie DOLCE.
KnowRob permet, entre autres, la modélisation d'informations sémantiques et topologiques. Pour les premières, l'ontologie s'appuie sur la connaissance encyclopédique (comme des livres, dictionnaires ...) pour décrire les types et les propriétés des objets et sur le développement du bon sens pour décrire à quoi peut servir un objet. Pour les secondes, l'ontologie définit le concept de lieux qui représentent une place pertinente dans l'environnement, mais aussi le concept de carte, représentant la carte topologique des lieux.

OMRKF ([Il Hong Suh et al., 2007]) : OMRKF est proposé pour mettre en œuvre l'intelligence du robot pour qu'il puisse être utile dans un environnement robotique. Cette ontologie permet, entre autres, de modéliser des informations sémantiques et topologiques. Le modèle se divise en trois niveaux :

- Niveau des caractéristiques de l'objet : inclut les parties de l'objet ou encore les caractéristiques visuelles ;
- Niveau objet : inclut le nom de l'objet et ses fonctionnalités ;
- Niveau de l'espace : inclut une carte métrique, une carte topologique et une carte sémantique.

OntoSTEP ([Pratt, 2001]) : cette ontologie permet de modéliser des informations géométriques. Le STandard for the Exchange of Product model (STEP) permet, en accord avec les besoins de l'industrie moderne, de faciliter la représentation et les échanges de données sur les produits ([Pratt, 2001]). Il est composé de plusieurs parties appelées Protocole d'Application (PA). Chaque PA est responsable de la modélisation d'un ou plusieurs aspects du produit. Une approche permettant de transformer le STEP en une ontologie formalisée en Web Ontology Language (OWL) est utilisée ([Barbau et al., 2012]).

OntoBREP ([Perzylo et al., 2015]) : des informations géométriques et topologiques sont modélisées à partir de cette ontologie. Cette ontologie introduit une approche exploitant les descriptions géométriques des modèles CAO au niveau des connaissances. Une ontologie est construite en définissant les Boundary representation (B-Rep) des objets. Cette ontologie consiste en une partie topologique qui illustre la connectivité et l'orientation des sommets, des arêtes et des faces, et en une partie géométrique qui décrit les primitives relatives à la partie topologique (points, courbes, surfaces).

ENVOn ([Zhao et al., 2023]) : Il s'agit d'une ontologie pour la représentation de l'environnement 3D dans lequel une tâche est exécutée, en modélisant des informations géométriques, topologiques et sémantiques. Pour ces dernières, elle fournit la description des corps rigides présents dans l'environnement, leurs propriétés (forme, couleur...) ou encore leur type (ouverture, trou...). Elle fournit également des informations sur les lieux et frontières du graphe topologique (comme la complexité à traverser un lieu). Les informations topologiques, elles, décrivent la connectivité des lieux dans l'environnement de simulation sous la forme d'un graphe. Enfin, la description géométrique ici faite regroupe les concepts et les relations liées aux corps rigides (B-Rep, Géométrie de Construction des Solides (CSG)) et à l'espace libre.

ASK ([Zhao, 2019]) : Il s'agit d'une ontologie permettant de formalisée des relations spatiales pour décrire comment deux objets sont relativement situés l'un par rapport à l'autre dans un environnement 2D/3D. Cette ontologie s'appuie sur ENVOn.

Parmi les différentes ontologies présentées, seule KnowRob s'appuie sur une TLO : l'ontologie DOLCE. Les ontologies présentées ici permettent de modéliser certaines informations sur un modèle de l'environnement. Cependant, ces informations ne sont pas suffisantes pour simuler des tâches, et d'autres informations relatives au TAMP doivent être considérées.

Ontologies de domaines liées au TAMP Une fois l'environnement modélisé, il faut également considérer la modélisation de concepts relatifs au TAMP. Plusieurs ontologies ont été construites dans cette optique, permettant de modéliser des informations relatives à la planifi-

cation de trajectoires ou de tâches.

ORO ([Lemaignan et al., 2010]) : « Open Robot Ontology » est un projet pour la mise en œuvre d'un cadre de représentation commun pour les robots autonomes considérant l'interaction homme-robot. Le cadre proposé vise à améliorer l'interaction du robot avec des environnements complexes et où un opérateur peut intervenir. Les robots doivent alors posséder des compétences cognitives avancées, telles que : la reconnaissance d'objets, l'interaction en langage naturel, la planification de tâches avec une éventuelle re-planification dynamique et la capacité à coopérer avec d'autres robots ou des humains.

RTPO ([Sun et al., 2019]) : « Robot Task Planning Ontology » présente une base de connaissance spécifique pour la planification de tâches d'un robot. Celle-ci est décomposée en trois parties :

— une partie décrivant le robot (notamment son type), le logiciel (Robot Operating System (ROS)) et le matériel équipé (composants et équipements du robot) ;

— une partie décrivant l'environnement dans lequel évolue le robot. Elle comporte une carte, les obstacles et tout autre objet pouvant être présents ;

— une partie décrivant les différentes tâches (les tâches de surveillance, de cartographie, de livraison et de charge).

OCRA ([Olivares-Alarcos et al., 2022]) : « Ontology for Collaborative Robotics and Adaptation » est une ontologie qui a été construite autour de deux notions principales : la collaboration et l'adaptation de plan. Cette ontologie assure une collaboration fiable entre un opérateur et un robot, puisque ce dernier est capable de formaliser et de raisonner sur leurs adaptations de plan de tâches et leurs collaborations dans des scénarios robotiques collaboratifs non structurés. Par ailleurs, OCRA améliore la réutilisation de la terminologie du domaine, permettant aux robots de représenter leurs connaissances sur les différentes situations de collaboration et d'adaptation.

PMK ([Diab et al., 2019]) : « Perception and Manipulation Knowledge » est une ontologie pour les robots autonomes de manipulation utilisés pour la planification de tâches et de trajectoires. Un module de perception peut être mis en place pour ajouter la sémantique de la scène, mais aussi les connaissances sur les objets et le raisonnement sur les tâches de manipulation.

Parmi ces ontologies, PMK s'appuie sur une ontologie de plus haut niveau, SUMO, et OCRA a été développé pour être compatible avec l'ontologie KnowRob (dont les concepts sont hérités de la TLO DOLCE).
Finalement, bien qu'il existe plusieurs ontologies intéressantes dans nos domaines de recherche,

rares sont celles s'appuyant sur une TLO. De plus, aucune de ces ontologies ne considèrent des informations pour le TAMP tout en prenant en compte des informations géométriques, topologiques et sémantiques pour la construction de l'environnement.
Il est alors nécessaire de construire nos propres ontologies et pour cela, il est important de voir quelles sont les méthodologies existantes pour cela.

2.3.4.4 Méthodologie pour construire une ontologie

Plusieurs méthodes ont été proposées pour construire correctement une ontologie. Il s'agit de définir clairement les éléments qui doivent apparaître dans une ontologie (son objectif, quels concepts doivent être définis...) et définir quelles sont les différentes phases, de la conception à la maintenance. Parmi ces méthodologies, il est possible de citer :

Toronto Virtual Enterprise (TOVE) ([Gruninger and Fox, 1994]) C'est une méthodologie qui a été proposée pour soutenir la modélisation des processus d'entreprise à l'Université de Toronto. Les différentes étapes sont :

- Scénarios de départ : le point de départ de la création d'une ontologie part d'un ensemble de problèmes rencontrés par une entreprise, qui est souvent présenté sous la forme de problèmes ou d'exemples ;
- Questions de compétences informelles : il s'agit des exigences de l'ontologie, il faut définir quelles sont les questions auxquelles doit répondre l'ontologie ;
- Spécification de la terminologie : c'est l'étape où les différents concepts, relations et attributs sont formellement définis ;
- Questions de compétences formelles : les exigences sont formalisées vis-à-vis de la terminologie exacte ;
- Spécification des axiomes : les axiomes sont donnés en logique du premier ordre, guidés par les questions de compétences formelles car les axiomes doivent être nécessaires et suffisants pour exprimer les questions de compétence et leurs solutions ;
- Théorèmes de complétude : une évaluation de l'ontologie en définissant les conditions dans lesquelles les solutions aux questions de compétence sont complètes.

Cette méthodologie est intéressante car elle fournit directement un moyen d'effectuer une évaluation grâce aux théorèmes de complétude. Ceux-ci sont utiles dans les tâches de maintenance ou pour fournir un point de référence pour les ontologies. Pour construire une ontologie, les étapes à suivre sont directement données et il suffit de les suivre.

Enterprise Model Approach Il s'agit d'une méthodologie de construction fortement basée sur un retour d'expérience, à savoir la construction de l'ontologie Enterprise ([Uschold, 1996a]). Ici, les différentes étapes sont :

— Identifier l'objectif de l'ontologie : cela permet de définir quel type d'ontologie construire et quel niveau d'abstraction considérer ;

— Identifier les champs d'application : une « Spécification » est faite. Cela permet de définir entièrement toutes les informations que l'ontologie doit caractériser. Cela peut se faire au travers de questions de compétences ou d'un système de « brainstorming et de tri » (lister des concepts potentiellement intéressants et supprimer ceux qui ne le sont pas) ;

— Formalisation : créer le « Code », les définitions formelles et les axiomes de la « Spécification ».

— Évaluation formelle : les critères d'évaluation peuvent être généraux ou spécifiques (vérification de l'objectif ou des questions de compétences).

METHONTOLOGY ([Fernández-López et al., 1997]) C'est l'une des méthodologies les plus utilisées. Il s'agit d'un ensemble d'activités permettant de construire des ontologies en partant de rien. Cela permet de construire des ontologies de domaines au niveau de la connaissance en suivant un cycle de vie basé sur l'implication de prototypes et de techniques.
METHONTOLOGY définit clairement les étapes à suivre lors de la construction d'une ontologie : 1) la spécification, 2) la conceptualisation, 3) la formalisation, 4) l'intégration, 5) l'implémentation, 6) l'évaluation et 7) la maintenance.
Plus concrètement, la phase de spécification décrit le but de l'ontologie, c'est-à-dire ses objectifs, ses utilisations prévues et ses utilisateurs finaux, tout en définissant des Questions de Compétences (QC). Ces QC sont un ensemble de questions pour lesquelles l'ontologie doit donner des réponses. Lors de la phase de conceptualisation, il convient d'organiser et de structurer la connaissance au travers d'un dictionnaire de vocabulaire ainsi que d'une taxonomie (qui est un modèle conceptuel utilisant des relations de types « est-un » entre les classes). La phase de formalisation consiste à transformer le modèle conceptuel en un format compatible en établissant les relations sémantiques entre les classes. Les ontologies étant construites pour être réutilisées, la phase d'intégration consiste intégrer le plus possibles des ontologies existantes, afin d'éviter la duplication des travaux. La phase d'implémentation permet d'avoir une ontologie dans un langage formel. Enfin, la phase d'évaluation a pour but de faire un jugement technique par rapport à un cadre de référence, afin de vérifier et de valider l'ontologie. L'étape finale est la maintenance. A tout moment, n'importe où, quelqu'un peut demander d'inclure ou de modifier des définitions dans l'ontologie. Des lignes directrices pour la maintenance des ontologies sont également nécessaires.
Comme pour la méthodologie TOVE, l'aspect le plus distinctif de METHONTOLOGY est la maintenance. La principale différence entre les deux est que dans METHONTOLOGY, l'accent est mis sur le traitement complet de la phase de maintenance du cycle de vie d'une ontologie, tandis que TOVE utilise des techniques plus formelles pour traiter un nombre plus limité de questions

de maintenance.

KBSI IDEF5 ([Benjamin Perakath, 1994]) Il s'agit d'une méthode conçue pour aider à la création, la modification et la maintenance des ontologies. Le point de vue de cette méthode est d'avoir une procédure générale avec un ensemble de lignes directrices. Il ne s'agit pas de suivre un protocole précis comme une recette de cuisine.

— Organisation et objectif : cela consiste à établir l'objectif, le point de vue de l'ontologie et le contexte de l'ontologie ;

— Collection des données : il faut réunir les données brutes nécessaires à la construction de l'ontologie ;

— Analyse des données : l'ontologie est extraite des résultats et de la collecte de données. Les objets d'intérêts sont listés puis sont identifiées par rapport aux limites de l'ontologie ;

— Développement initial de l'ontologie : une ontologie préliminaire, qui contient des proto-concepts est développée, c'est-à-dire les descriptions initiales des types, des relations et des propriétés ;

— Affinement et validation de l'ontologie : les proto-concepts sont affinés et testés de manière itérative.

Méthodes additionnelles En plus des méthodes décrites précédemment, il existe des approches qui se concentrent sur des points plus spécifiques.

Ontolingua L'un des principaux avantages à utiliser la bibliothèque d'ontologie Ontolingua est qu'elle offre une vaste collection d'ontologies préalablement définies ([Uschold, 1996b]). Cette bibliothèque est élargie dès qu'un utilisateur y ajoute une nouvelle ontologie, ce qui fait qu'elle est en perpétuelle évolution. Une ontologie créée à partir de cette bibliothèque peut donc être considérée comme un module. Une ontologie développée sous cette approche peut être réutilisée de différentes manières :

— Inclusion : une ontologie A est explicitement incluse dans une ontologie B. Le vocabulaire de A est traduit dans B. Cette traduction est ensuite appliquée aux axiomes de A et ceux-ci sont ajoutés dans B ;

— Affinement polymorphique : une définition venant d'une ontologie est incluse et affinée ;

— Restriction : une version restreinte d'une ontologie est incluse dans une autre ;

— Inclusion cyclique : l'inclusion d'ontologie est transitive. Ainsi est possible la situation suivante : l'ontologie A est incluse dans B, l'ontologie B est incluse dans C et l'ontologie C est incluse dans A.

ONtologic Integration Of Naive Sources (ONIONS) Cette méthodologie est motivée par des problèmes d'intégration de connaissances ([Gangemi et al., 2007]). Il s'agit d'un problème survenant lors de la création d'une ontologie lorsque des référentiels de connaissances existent. L'un des aspects distinctifs de l'approche ONIONS est la méthode d'acquisition de l'ontologie. Cela produit une ontologie non formelle, un compte rendu schématique de la conceptualisation du domaine. La solution utilisée prend en compte la pertinence de l'ontologie en fonction de la tâche prévue (même s'il n'existe pas de moyen de déterminer l'adéquation d'une ontologie par rapport à la tâche).

MENELAS Elle a été conçue dans le cadre d'un système de compréhension du langage naturel ([Bouaud et al., 1994]). Quatre principes liés au développement taxonomique sont décrits :

— Similarité : chaque sous-classe doit être du même type que son parent ;

— Spécificité : chaque sous-classe doit avoir une différence la distinguant de son parent. Cette différence forme des conditions nécessaires et suffisantes pour la définition de la sous-classe ;

— Opposition : chaque sous-classe d'un concept est incompatible avec les autres sous-classes ;

— Axe sémantique unique : les sous-classes d'un concept peuvent être contraintes de différer du parent par une propriété commune.

2.3.4.5 Synthèse

Les ontologies apparaissent comme un choix évident afin de modéliser la connaissance, d'autant plus qu'il est possible de facilement réutiliser des ontologies existantes.
Nous avons vu qu'il existe des ontologies représentant la connaissance sur la modélisation de l'environnement, et d'autres représentant des informations sur la planification de trajectoires et/ou de tâches. Toutefois, aucune ontologie ne prend en compte tous les types d'informations conjointement. En effet, aucune ontologie ne prend en compte la conceptualisation des informations de l'environnement (informations sémantiques, topologiques, géométriques) à la fois pour les corps rigides et les modèles d'espace libre dans une ontologie évolutive, tout en modélisant des informations liées au TAMP.
Comme aucune ontologie ne correspond à nos besoins, une possibilité est alors de construire les notres. Il existe plusieurs méthodologies dans la littérature et en suivre une semble essentiel car cela permet d'avoir une ontologie cohérente et bien construite.

2.3.5 Synthèse de l'état de l'art et positionnement de nos travaux

Cet état de l'art a permis de présenter différentes notions importantes pour la validation de tâches en environnement virtuel. Celles-ci peuvent être brièvement résumées :

— La planification de trajectoires : elle a pour objectif de trouver une trajectoire, un mouvement réalisable et sans collision. Valider le mouvement est en effet une étape clef pour la validation d'une tâche. Dans nos travaux, nous souhaitons valider des tâches sous de fortes contraintes géométriques, et avons ainsi choisi d'utiliser un algorithme probabiliste : le Bi-RRT. Toutefois, ses performances ne sont pas suffisantes et des stratégies sont développées pour les améliorer ;

— La planification de tâches : pour valider une tâche, le mouvement n'est pas suffisant. Il faut également considérer des informations relatives à l'action à effectuer, et pour cela, il faut considérer la planification de tâches ;

— Le TAMP : il est important de considérer conjointement planification de tâches et de trajectoires pour valider correctement une tâche. Cette approche est appelée TAMP ;

— Les ontologies : il s'agit d'un modèle de connaissance, capable d'assurer l'interopérabilité entre un planificateur de tâches et un planificateur de trajectoires. Cela permet également d'assurer une bonne collaboration entre ces planificateurs.

Néanmoins, cet état de l'art met aussi en exergue certaines faiblesses dans ces domaines.
En effet, individuellement, la planification de tâches et la planification de trajectoires ont été très bien étudiées depuis de nombreuses années. Toutefois, l'utilisation conjointe de celles-ci, dans ce qui est aujourd'hui appelé le TAMP, est plutôt nouvelle. Celle-ci, principalement utilisée pour des robots manipulateurs, commence à s'exporter à des domaines plus vastes comme la robotique mobile, l'exploration et la navigation, ou encore l'assistance à la validation de scénarios complexes tels que des tâches de manipulation sous fortes contraintes géométriques en RV.
Pour développer une démarche TAMP efficace, capable d'être utilisée dans différents domaines, l'utilisation d'ontologies comme modèle de connaissance peut être intéressante.
Toutefois, puisque le TAMP est une approche récente, il existe peu d'ontologies dans ce domaine. De plus, rares sont celles se basant sur une ontologie de haut niveau, permettant facilement leurs réutilisations. Finalement, il n'en existe pas pour la validation de tâches en environnement virtuel.

Pour faire face à ces limitations, les travaux de cette thèse portent sur le couplage sémantique basé ontologie de la planification de tâches et de trajectoires.
Nos contributions portent sur :

— La construction de deux ontologies, a) une ontologie, ENVOn-2, permettant de modéliser

l'environnement 3D dans lequel une tâche se déroule et b) une seconde ontologie, TAMPO, permettant de modéliser les concepts liés à la planification de tâches et de trajectoires. Cette contribution est présentée dans le chapitre 3 ;

— La définition d'une méthodologie pour coupler sémantiquement la planification de tâches et de trajectoires. Cette méthodologie s'appuie sur les deux ontologies présentées et permet a) d'évaluer les stratégies de planifications de trajectoires sur une action primitive d'un plan de tâches et b) d'évaluer les plans de tâches proposées par un planificateur de tâches ou un opérateur humain. Cette contribution est présentée dans le chapitre 4.

Chapitre 3

Proposition d'ontologies pour le TAMP

3.1 Introduction

Afin de pouvoir simuler des tâches en environnement virtuel, plusieurs étapes clefs sont à considérer. La première consiste en la modélisation de l'environnement dans lequel se déroulent les tâches.
La deuxième concerne la modélisation des connaissances sur la tâche en elle-même. Ceci peut être le plan de tâche, les différentes tâches primitives à exécuter, les contraintes géométriques auxquelles sont soumises les différentes tâches etc.
Par ailleurs, pour la validation de scénarios simulant des tâches, deux planificateurs entrent en jeu : le planificateur de tâches et le planificateur de trajectoires, et ceux-ci ont besoin d'un vocabulaire commun pour qu'ils puissent collaborer efficacement.

Le TAMP est une approche récente qui a peu été étudiée en environnement numérique. Nous proposons ainsi une approche basée ontologie pour le couplage sémantique de la planification de tâches et de trajectoires.
Dans ce travail, nous avons conceptualisé et formalisé deux ontologies de domaines liées au TAMP, appelées ENVOn-2 et TAMPO, qui ont pour but respectif de modéliser les connaissances de l'environnement 3D dans lequel se situent les tâches à réaliser et de modéliser les connaissances qui sont spécifiques aux tâches. Ces ontologies, correctement construites, assurent l'interopérabilité des deux planificateurs.

Pour arriver à ces fins, nous nous sommes positionnés par rapport aux travaux de [Chen, 2006], décrits dans le paragraphe 2.3.4.1 du chapitre 2. Dans nos travaux, les barrières pour l'interopérabilité sont au niveau technologique car venant de l'utilisation de deux planificateurs différents. Ceux-ci utilisent des données différentes, et nous proposons donc d'utiliser une approche fédérée (donc basée ontologie)(figure 3.1).

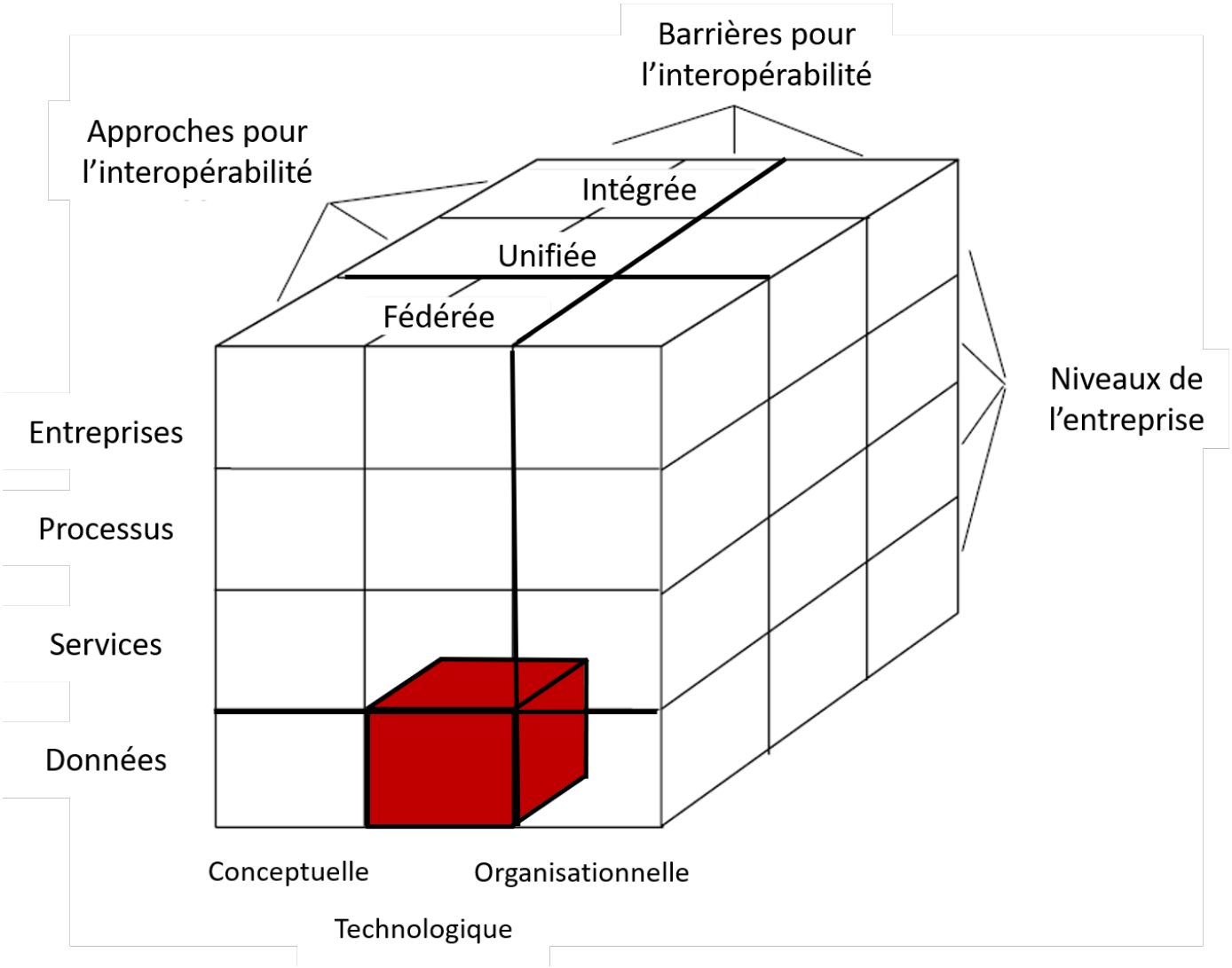


FIGURE 3.1 – Positionnement selon le cadre d'interopérabilité de [Chen, 2006]

Notre approche, basée ontologie, pour le couplage sémantique entre les planificateurs de tâches et les planificateurs de trajectoires s'appuie sur deux nouvelles ontologies de domaines. Afin de construire correctement ces deux ontologies, et parmi les différentes méthodologies décrites au paragraphe 2.3.4.4, nous avons retenu la méthode METHONTOLOGY du fait de sa bonne structuration et de la facilité de compréhension de ses méthodes ([Gaševic et al., 2009]). De plus, nos ontologies de domaines doivent avoir un point de départ commun. Celui-ci, est l'ontologie de haut niveau SUMO (paragraphe 2.3.4.3.1, chapitre 2.3). Nous avons fait ce choix car :

— Il s'agit d'une ontologie de haut niveau très complète et bien organisée ;

— Le développement et la maintenance de SUMO est toujours d'actualité, elle est régulièrement mise à jour ;

— Elle est le point de départ d'ontologies de domaine dont la réutilisation pourrait être intéressante pour de futurs travaux (CORA ([Neto et al., 2019]) par exemple).

Ce chapitre présente nos deux propositions d'ontologies pour le TAMP. Une ontologie pour la modélisation de l'environnement 3D dans lequel une tâche a lieu, est présentée dans la partie 3.2. Cette dernière modélise l'environnement selon plusieurs niveaux d'abstraction (géométriques, topologiques et sémantiques). Puis, une seconde ontologie modélisant les différents concepts liés au TAMP qui entrent en jeu au cours d'une tâche est présentée dans la partie 3.3.

3.2 Proposition de l'ontologie ENVOn-2

ENVOn-2 (ENVironment Ontology 2) est une ontologie qui a pour but de modéliser les informations de l'environnement 3D dans lequel se déroulent les tâches. Elle s'appuie sur l'ontologie ENVOn développé au LGP ([Zhao et al., 2023]).

3.2.1 Présentation de ENVOn

Dans cette partie, nous allons présenter ses principales fonctionnalités.

ENVOn permet de modéliser l'environnement selon plusieurs informations : sémantiques, topologiques et géométriques. Elle a pour objectif de :

— Rechercher rapidement toutes informations sur l'environnement (Où est l'objet A ? Est-ce que l'objet A est dans le lieu B ? ...)

— Inférer des informations implicites telles que «L'objet A convient au lieu B », « Dans quel trou devrait être inséré l'objet A ? » ...

ENVOn doit pouvoir capturer les notions clefs et les relations liées à un environnement 3D dans lequel s'effectue des tâches. Les informations liées à l'environnement sont regroupées en trois catégories (les informations géométriques, les informations topologiques et les informations sémantiques) et décrivent à la fois l'espace libre et les objets rigides qui composent l'environnement. Cette ontologie réutilise la structure qui a été définie par [Cailhol et al., 2015].

L'architecture générale d'ENVOn est présentée dans la figure 3.2. Celle-ci est composée de trois modules :

— Le module de description de la géométrie : ce module regroupe les concepts et leurs relations en lien avec la géométrie de l'espace libre ou des objets rigides.

— Le module de description de la topologie : ce module décrit les lieux et les frontières identifiés dans l'environnement 3D.

— Le module de description de la sémantique : ce module fournit une description sémantique des différents composants de l'environnement (objets rigides, lieux, frontières ...)

Synthèse sur ENVOn

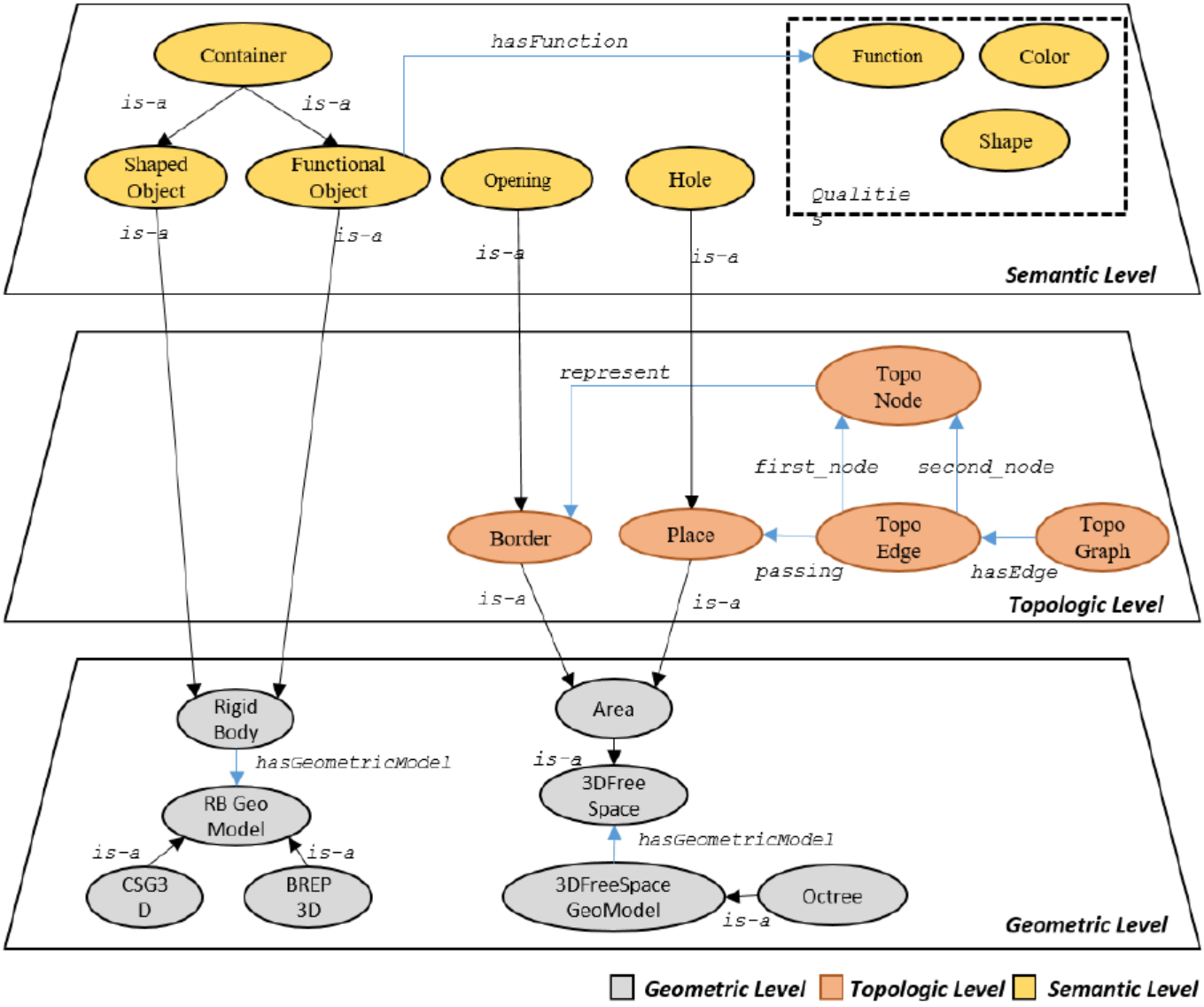


FIGURE 3.2 – Architecture générale de l'ontologie ENVOn

L'ontologie intègre les informations de l'environnement 3D (corps rigides et espace libre), en considérant plusieurs niveaux d'informations (géométriques, topologiques et sémantiques). Les modules peuvent ainsi être considéré individuellement. Pour chacun, il est alors plus simple de mettre à jour ou modifier sa description sémantique pour l'adapter à l'application ciblée.

Bien que les résultats apportés par ENVOn soient encourageants, cette ontologie a des limites. Aucune méthodologie n'a été utilisée pour la construire. De plus, cette ontologie a été conçu de zéro, et ne s'appuie pas sur une ontologie de haut niveau. Or, une des bonnes pratiques est d'utiliser une ontologie neutre au domaine de plus haut niveau [Arp, 2015]. Une ontologie de domaine est dite bien fondée si elle est basée sur une ontologie fondatrice. Utiliser une ontolo-

gie de haut niveau permet également d'assurer l'interopérabilité avec toutes les autres ontologies de domaines utilisant cette ontologie de haut niveau. Enfin, certains concepts importants pour la modélisation de l'environnement sont manquants.

3.2.2 Construction de ENVOn-2

Après avoir présenté l'ontologie ENVOn, ses objectifs, ses caractéristiques ainsi que ses limites, nous allons dans cette partie décrire comment l'ontologie ENVOn-2 a été construite afin de surmonter les limitations précédentes tout en répondant aux mêmes besoins.
Pour construire correctement notre ontologie, il est nécessaire de :

— Suivre les différentes étapes de METHONTOLOGY présentée dans le paragraphe 2.3.4.4 du chapitre 2.3 ;

— Se définir clairement comme une ontologie de domaine ;

— S'appuyer sur une ontologie de plus haut niveau.

3.2.2.1 Phase de spécification

Lors de la phase de spécification, METHONTOLOGY prévoit de spécifier le but de l'ontologie, c'est-à-dire son utilisation, ses scénarios d'utilisation, ses utilisateurs finaux etc. Il faut aussi spécifier le champ d'application, qui comprend l'ensemble des termes à représenter, ses caractéristiques et sa granularité.

L'ontologie ENVOn répond à ces spécifications. Ainsi, nous pouvons nous appuyer sur le paragraphe 3.2.1 pour définir l'ensemble des spécifications de notre nouvelle ontologie.

ENVOn-2 a pour objectif, comme ENVOn, de modéliser l'environnement 3D dans lequel se déroule une tâche, en modélisant les informations géométriques, topologiques et sémantiques des objets rigides et de l'espace libre. Cette ontologie doit permettre de rapidement fournir tout type d'information sur l'environnement (Quel type d'objet y a-t-il ? Est-ce que l'objet (X) est manipulable ? Quelle est la direction d'ouverture de la place (Y) ? Dans quel lieu (Y) se trouve l'objet (X) ? ...) tout en pouvant inférer des informations pratiques (Est-ce que l'objet (X) peut être inséré dans la place (P) ? ...). Ces différentes questions font partie des QC auxquelles l'ontologie doit répondre, et qui sont présentées par le tableau 3.1.
Dans ce tableau, (X) représente n'importe quel objet de l'environnement et (Y) représente n'importe quel lieu.

Recherche de détails géométriques	QC1	Quel est l'axe central de (X) ou (Y) ?
	QC2	Quel est le sens d'ouverture de (Y) ?
	QC3	Quelle est la direction de pointage de (X) ?
	QC4	Quel est le volume de (X) ou (Y) ?
	QC5	Quelle est l'origine de (X) ou (Y) ?
Localisation	QC6	Où se trouve (X) dans l'environnement 3D ?
	QC7	Quels sont les lieux (Y) à l'intérieur d'un corps rigide (X) ?
Réalisation de la tâche	QC8	Est-ce que l'objet (X) peut être inséré dans le lieu (Y)
	QC9	Quel est le meilleur lieu (Y) où insérer l'objet (X) ?

TABLE 3.1 – Questions de compétences de l'ontologie ENVOn-2

3.2.2.2 Phase de conceptualisation, de formalisation et d'intégration

Ces différentes phases ont pour objectifs d'acquérir les différentes connaissances nécessaires pour notre ontologie et de formaliser ces connaissances. Ces différentes connaissances peuvent venir de livres, de définitions d'experts ou de dictionnaires ou de figures par exemple. Elles peuvent également venir d'autres ontologies, ce qui correspond à la phase d'intégration de METHONTOLOGY.
Pour notre ontologie ENVOn-2, un premier dictionnaire est considéré, qui est celui défini dans ENVOn (paragraphe 3.2.1). De plus, nous nous appuyons sur une ontologie de plus haut niveau - SUMO -, afin d'avoir une généricité dans nos concepts. Ainsi, un deuxième dictionnaire vient s'ajouter.

La classe supérieure définie dans SUMO, qui est le point de départ de toute notre ontologie, est la classe *Entity*, qui est définie comme « la classe universelle des individus. Il s'agit de la racine de l'ontologie ». Cette classe est divisée en deux grandes classes : *Physical* et *Abstract*. La première correspond aux entités ayant un emplacement dans l'espace-temps. La seconde correspond aux propriétés ou aux qualités des objets. Les principales classes sont présentées dans le tableau 3.2.

Classe	Définition
Entity	Classe universelle des individus. Il s'agit de la racine de l'ontologie.
Physical	Entité qui a un emplacement dans l'espace-temps. Notez que les lieux sont eux-mêmes compris comme ayant une localisation dans l'espace-temps.
Abstract	Propriétés ou qualités. Les instances de cette classe existent dans le même sens physiques que les objets mathématiques (i.e. ensembles ou relations), mais ne peuvent exister sans incarnation physique.
Object	Correspond à la classe des objets ordinaires. Il s'agit par exemple des objets physiques normaux ou de régions géographiques.
Process	La classe des choses qui se produisent et qui ont des parties ou des étapes temporelles. La définition formelle est la suivante : tout ce qui se produit dans le temps mais qui n'est pas un objet.
Collection	Les collections ont des membres, comme les classes, mais ont une position dans l'espace-temps et des membres peuvent s'ajouter ou se soustraire sans modifier l'identité de la collection (exemple : équipes de football, troupeaux de moutons...).
Region	Emplacement topographique. Une région englobe les surfaces des objets et des lieux imaginaires.
Agent	Il s'agit d'un déterminant actif animé ou inanimé d'un processus. (exemple : Eve est un agent dans la proposition : Eve a croqué une pomme)
Internal_Change	Processus qui impliquent la modification d'une propriété interne d'un objet (par exemple sa forme, sa couleur, sa structure ...).
Motion	Tout processus de mouvement.
Proposition	Les propositions sont des entités abstraites qui expriment une pensée complète.
Quantity	Toute spécification du nombre ou de la quantité de quelque chose. En conséquence, il existe deux sous-classes de *Quantity* : *Number* et *Physical_Quantity*.
Relation	Il y a deux types de relations : les prédicats et les fonctions. Ce sont tous deux des ensembles de n-tuples ordonnés.
Attribute	Des qualités que nous ne pouvons ou ne voulons pas réifier en sous-classes.

TABLE 3.2 – Classes principales de l'ontologie SUMO

Après avoir défini les spécifications de notre ontologie, et après avoir conceptualisé et formalisé l'ensemble des classes à considérer, nous pouvons maintenant décrire comment l'ontologie ENVOn-2 a été construite.

3.2.2.3 Phase d'implémentation

Cette section décrit en détail l'implémentation de l'ontologie ENVOn-2. L'architecture générale est présentée dans le paragraphe 3.2.2.3.1. Celle-ci se décompose en trois niveaux, un niveau Méta qui est présenté dans le paragraphe 3.2.2.3.2, un niveau Domaine présenté dans le paragraphe 3.2.2.3.3, et un niveau Instance présenté dans le paragraphe 3.2.2.3.4.

3.2.2.3.1 Architecture générale

Cette nouvelle ontologie a une architecture en trois niveaux, classant les concepts en fonction de leur généricité. La figure 3.3 montre l'architecture générale de notre ontologie, avec les principaux concepts de chaque niveau.

— le niveau « Méta » : Dans ce niveau, les concepts généraux, non spécifiques à un domaine en particulier, sont décrits. Ils apparaissent en bleu clair et en gris.

— le niveau « Domaine » : Ce niveau permet de décrire l'ensemble des concepts nécessaires à la description de l'environnement 3D dans lequel se déroule une tâche, pour une application précise. Ces informations sont regroupés en trois familles : les informations topologiques en bleu foncé, les informations sémantiques en jaune et les informations géométriques en orange.

— le niveau « Instance » : À ce niveau sont définies les différentes instances, spécifiques à l'application. Ils ne sont pas colorés.

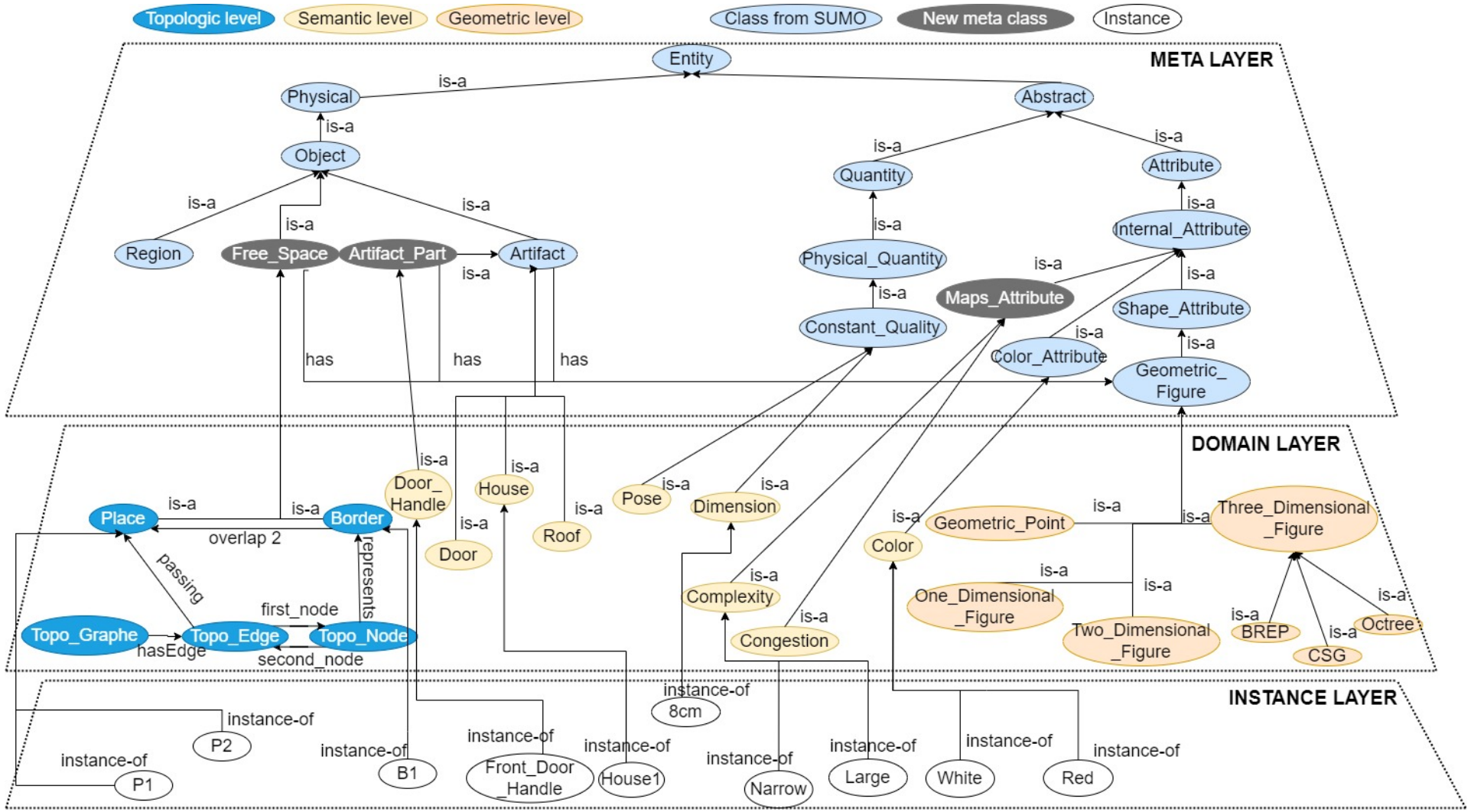


FIGURE 3.3 – Structure de l'ontologie ENVOn-2

3.2.2.3.2 Description du niveau Méta

Le niveau « Méta » permet de représenter des informations génériques, telles que le concept d'objet physique ou certaines notions abstraites comme les quantités. Les concepts définis ici sont des concepts généraux, indépendants du domaine d'application. Il s'agit pour la plupart de concepts venant de SUMO (en bleu sur la figure 3.3), bien que certains concepts, colorés en gris, aient été ajoutés car ils sont nécessaire dans notre approche. Les principaux concepts de SUMO sont définis dans le paragraphe 2.3.4.3.1, les autres peuvent être définis comme suit :

— *Free_Space* : Espace dans lequel il n'y a pas d'obstacles ;

— *Artifact* : Un objet (*Object)* qui est le produit d'une fabrication ;

— *Artifact_Part* : Composant/pièce d'un artéfact (*Artifact*) ;

— *Physical_Quantity* : C'est une mesure d'un aspect quantifiable du monde modélisé (exemple : le diamètre de la Terre) ;

— *Constant_Quantity* : C'est une quantité physique qui a une valeur constante ;

— *Internal_Attribute* : Tout attribut qui est une propriété interne d'une entité (*Entity*), comme sa forme, sa couleur… ;

— *Place_Attribute* : Tous les attributs relatifs à la représentation des lieux (*Places*) ;

— *Color_Attribute* : Tous les attributs relatifs à la couleur des objets ;

— *Function_Attribute* : Tous les attributs relatifs aux fonctions des objets ;

— *Shape_Attribute* : Tous les attributs relatifs à la forme d'un objet ;

— *Geometric_Figure* : Correspond à la classe de toutes les figures géométriques.

Des concepts plus précis et plus spécifiques au domaine de la modélisation de l'environnement sont définis à partir des classes de ce niveau.

3.2.2.3.3 Description du niveau Domaine

Les concepts du niveau « Domaine » sont des concepts spécifiques de la modélisation de l'environnement. Ce niveau peut également être décomposé en trois sous-parties (visible sur la figure 3.3) qui représentent chacun un niveau d'abstraction particulier de l'information : le niveau géométrique, le niveau topologique et le niveau sémantique.

Les informations géométriques, peuvent être relatives aux objets (*Objects*) ou à l'espace libre (*Free_Space*).
Dans le cadre d'une simulation numérique, les objets (*Objects*) sont représentés par des modèles CAO. Il existe deux techniques principales pour la représentation géométrique des objets (*Objects*) : 1) CSG et 2) B-Rep.

La première représentation, CSG, décrit un objet (*Object*) par son volume. Ce dernier est construit à partir de primitives standard et d'opération booléennes. La seconde représentation, B-Rep, décrit un objet (*Object*) par sa surface.
Un artéfact (*Artifact*) a une forme géométrique (*Geometric_Figure*) qui est une forme géométrique 3D (*Three_Dimensional_Figure*), soit de type B-Rep soit de type CSG. Il s'agit d'objets (*Object*) physiques non déformable. Un tel objet a certaines propriétés telles qu'une origine, définie par un point géométrique (*Geometric_Point*) ou un système d'axes définis par des formes géométriques 2D (*Two_Dimensional_Figure*).

Après avoir vu comment était représenté géométriquement un objet (*Object*), nous nous intéressons à la représentation de l'espace libre (*Free_Space*). Nous proposons de décomposer l'espace libre en cellules géométriques. Cette technique, appelée décomposition en octree, permet de représenter de l'espace libre.
De la même manière qu'un objet (*Object*) a certaines propriétés qui lui sont propres, l'espace libre (*Free_Space*) en a également :

— *Free_Space* : Cela représente tout l'espace qui n'est pas occupé par un objet (*Object*) ;
— *Octree* : Il s'agit d'une méthode de décomposition de l'espace libre (*Free_Space*) bien connu qui subdivise un cube en cubes de plus petites tailles jusqu'à ce que la taille seuil (qui a été préalablement définie) soit atteinte. Il s'agit de la taille minimale qu'un cube d'octree peut atteindre. Les cubes obtenus peuvent être de différentes tailles car seuls ceux contenant un obstacle sont subdivisés. L'octree forme ainsi un arbre déséquilibré ;
— *Octree_Node* : Un noeud d'octree (*Octree_Node*) correspond à une cellule de l'octree (qui est donc sous la forme d'un cube).

Les informations topologiques, les lieux (*Places*) représentent des emplacements importants de l'environnement qui doivent être identifiés. Ces lieux (*Places*) sont ensuite connectés par des frontières (*Borders*). Celles-ci sont des zones de l'espace libre qui chevauchent deux lieux (*Places*). La connectivité formée par cet ensemble de lieux (*Places*) et de frontières (*Borders*) est appelé un graphe topologique (*Topo_Graph*) et permet de représenter facilement et de manière simple l'environnement dans lequel se déroule la tâche.

Les concepts topologiques sont définis comme suit :

— *Place* : Un lieu (*Place*) représente une localisation spatiale dans l'environnement. Il s'agit de localisation pertinente d'un environnement, qui peut être caractérisée par sa fonctionnalité (chambre, cuisine) ;
— *Border* : Une frontière (*Border*) est le chevauchement de deux lieux (*Places*). Dans un bâtiment, il s'agirait par exemple des portes entre deux pièces ;

— *Topo_Graph* : Il s'agit de la représentation de la connectivité de l'espace libre ;

— *Topo_Node* : Il s'agit d'un élément primitif du graphe topologique *Topo_Graph* et qui représente un lieu (*Place*) ;

— *Topo_Edge* : À l'inverse d'un nœud topologique (*Topo_Node*), il s'agit de l'élément du graphe topologique *Topo_Graph* qui représente les frontières (*Border*).

Les informations sémantiques apportent des informations contextuelles sur les lieux (*Places*) et les frontières (*Borders*). En effet, lorsqu'un humain effectue une tâche, celui-ci ne raisonne pas qu'avec des informations géométriques ou topologiques. Il associe un contexte, une signification, aux objets qu'il manipule ou aux lieux qu'il voit. Par exemple, dans une maison, il retrouvera des objets tels que des tables, des chaises, un lave-vaisselle, un four… dans une pièce qu'il reconnaîtra alors comme une cuisine. Il en va de même pour l'ensemble des objets et des pièces, et il sera facile pour lui de se repérer. Ainsi, ce niveau fournit des informations contextuelles à des objets, à des lieux et à des frontières.

[Xu, 2017] fait une différenciation entre ontologie et contexte. D'après lui, une ontologie est un modèle d'un domaine dans lequel est représentée une vision commune de l'ensemble de ces composants. Un contexte est, quant à lui, un modèle local spécifiant le point de vue d'un domaine particulier. En accord avec cette définition, le niveau sémantique représente deux types d'informations : la sémantique indépendante de l'application (ontologie), et la sémantique spécifique au contexte.

1) sémantique indépendante de l'application
Il s'agit d'informations caractérisant des propriétés générales qui sont présentes dans tous les scénarios.
Cela peut être pour décrire une entité (*Entity*), ce à quoi elle ressemble, mais aussi sa fonction. Ces informations peuvent aussi servir à décrire l'état de l'espace libre (*Free_Space*) (complexité, encombrement ...).

— *Shape* : Décrit la forme géométrique d'une entité (*Entity*). Plusieurs sous-classes héritent de celle-ci, telles que *Irregular_Shape_Quality* ou *Regular_Shape_Quality*.

— *Color* : Décrit la couleur d'une entité.

— *Complexity* : Décrit la complexité à traverser un lieu. Si un lieu contient plusieurs obstacles, dynamiques de surcroît, sa complexité sera élevée.

— *Congestion* : Décrit l'encombrement d'un lieu . Il s'agit d'un état à un instant donné et dépendant de la taille de l'objet manipulé en fonction du lieu considérée.

— *Object_Mobility* : Un objet peut être mobile, pouvant être manipulé ou bien être un obstacle mobile. Un objet peut également être fixe, il s'agit par exemple de murs ou d'objets qui ne doivent pas bouger.

2) Sémantique dépendante au contexte
Il s'agit d'informations relatives à un scénario ou une application particulière. Dans le cadre qui nous intéresse, nous retrouvons des concepts qui sont spécifiques à la simulation de tâches. Il s'agit donc de spécifier certaines informations utiles à des tâches d'assemblages, de désassemblages ou autres. Dans ce contexte, il peut être intéressant de savoir si un objet possède un trou (*Hole*), pouvant indiquer qu'il est possible d'insérer un objet à l'intérieur. Toutefois, dès lors que l'application change, de nouveaux concepts peuvent être ajoutés ou supprimés, si l'environnement est modifié.

3.2.2.3.4 Description du niveau Instance

Ce niveau décrit les instances qui entrent en jeu pour une application spécifique. Cette partie de l'ontologie est alors explicitée précisément dans la partie 5.4 pour chacune des applications décrites.

3.2.2.4 Phase d'évaluation

Cette partie consiste à vérifier que notre ontologie est construite correctement. Il s'agit de vérifier que l'ontologie est cohérente, que les classes définies sont satisfaisantes et que les différentes inférences correspondent aux souhaits demandés. Pour cela, nous commençons par vérifier la cohérence de notre ontologie. Nous utilisons le raisonneur Pellet qui est intégré dans Protégé ([Noy et al., 2003]). Le raisonneur nous retourne l'information que notre ontologie est cohérente et consistante.

L'étape suivante est de vérifier que notre ontologie peut correctement répondre aux spécifications (c'est-à-dire aux questions de compétences). Cette étape de la validation sera présentée dans la partie 5.4 et 5.5, quand nous présenterons l'instanciation de différents cas d'utilisation dans notre ontologie.

3.3 Proposition de l'ontologie TAMPO

Nos travaux propose une deuxième ontologie originale, TAMPO, qui a été construite pour assurer le couplage sémantique entre la planification de tâches et la planification de trajectoires.

Pour la construction de TAMPO, la même méthodologie que pour ENVOn-2 a été suivie (METHONTOLOGY) et la même ontologie de haut niveau (SUMO) sert de point de départ (afin de s'assurer que les deux ontologies ont un vocabulaire commun pour pouvoir collaborer).

3.3.1 Phase de spécification

En suivant la même méthodologie, il faut spécifier l'objectif de l'ontologie, ses exigences et définir des QC.

3.3.1.1 Objectif de l'ontologie

TAMPO est une ontologie de domaine destinée à résoudre des problématiques liées à la coopération entre les planificateurs de tâches et de trajectoires. Pour cela, elle peut utiliser les informations de l'environnement modélisées dans ENVOn-2. Il faut donc que ces deux ontologies puissent communiquer facilement. TAMPO doit :

— Modéliser les concepts relatifs au TAMP ;

— Inférer, à partir de contraintes spatiales définies au niveau tâche, des contraintes géométriques permettant de mieux contrôler la planification de trajectoires ;

— Évaluer différents plans de tâches, pour proposer le plus pertinent.

3.3.1.2 Les exigences de TAMPO

TAMPO doit pouvoir modéliser les notions clefs de la planification de tâches et de trajectoires. De plus, une tâche étant réalisée dans un environnement particulier, cette ontologie doit pouvoir communiquer avec ENVOn-2.

3.3.1.3 Questions de compétences

Afin de pouvoir répondre correctement aux objectifs de l'ontologie, un ensemble de question de compétences ont été définies et sont présentées dans le tableau 3.3 :

Plan de tâches	QC1	Quel est la longueur du plan (P) ?
	QC2	Quelle est la complexité du plan (P) ?
	QC3	Quel est le plan (P) à privilégier ?
Contraintes spatiales	QC4	Quelles sont les contraintes spatiales de la tâche primitive (T) ?
Contraintes géométriques	QC5	Quelles sont les contraintes géométriques inférées pour la tâche primitive (T) ?

TABLE 3.3 – Questions de compétences de l'ontologie TAMPO

3.3.2 Phase d'intégration

Une ontologie, ASK ([Zhao, 2019]) a été développé au LGP permettant d'inférer des contraintes géométriques à partir de contraintes spatiales. Nous allons réutiliser une partie des concepts définis dans cette ontologie. Les concepts réutilisés de cette ontologie sont définies dans le tableau 3.4. Sa structure est présentée sur la figure 3.4.

3.3.3 Phase de conceptualisation et de formalisation

Afin de pouvoir correctement communiquer avec l'ontologie ENVOn-2, nous nous sommes appuyé sur la même ontologie de haut niveau, à savoir SUMO. Les principaux concepts de cette ontologie de haut niveau sont définis dans le paragraphe 3.2.2.2.

L'ontologie TAMPO modélise des concepts du domaine du TAMP. Il faut donc maintenant enrichir les classes de SUMO avec des concepts propres au TAMP. Ces concepts sont liées à la planification de trajectoires, à la planification de tâches et à la formalisation des relations spatiales. Les principaux concepts n'étant pas définis dans l'ontologie SUMO ou dans l'ontologie ASK sont :

— *Task_Plan* : Séquence d'action primitives (*Primitive_Action*) résolvant une tâche ;

— *Primitive_Action* : Tâches pouvant être réaliser directement ([Erol et al., 1994]) ;

— *Algorithms* : Classe des algorithmes de planification de tâches et de trajectoires.

Classe	Définition
Primitive Action Specification (PAS)	Classe constituée de quatre parties : - l'identité de l'action primitive - l'objet manipulé - la géométrie de référence - les contraintes spatiales associés à l'action primitive
Path Planning Query Description (PPQD)	Classe constituée de trois parties : - l'objet manipulé - la configuration finale - les contraintes géométriques à appliquer à la planification de trajectoires
Spatial Constraint	Contraintes qui restreignent une relation spatiale entre deux éléments. Elles sont compréhensibles pour un humain que les contraintes géométriques.
Geometric Constraint	Contraintes formulées comme les fonctions mathématiques d'égalité ou d'inégalité
Spatial Relation	Relations décrivant comment deux objets sont localisés l'un par rapport à l'autre.
Orientation Relation	Décrit comment deux géométries ou deux éléments géométriques sont placées l'un par rapport à l'autre (exemple : au-dessus, à droite etc...).
Topological Relation	Décrit comment les frontières ou les intérieurs de deux géométries sont liées (exemple : à l'intérieur de, égal, etc...).
Geometric Relation	Décrit comment deux éléments géométriques sont liés l'un à l'autre (exemple : coplanaire, perpendiculaire etc...).

TABLE 3.4 – Principaux concepts de l'ontologie ASK

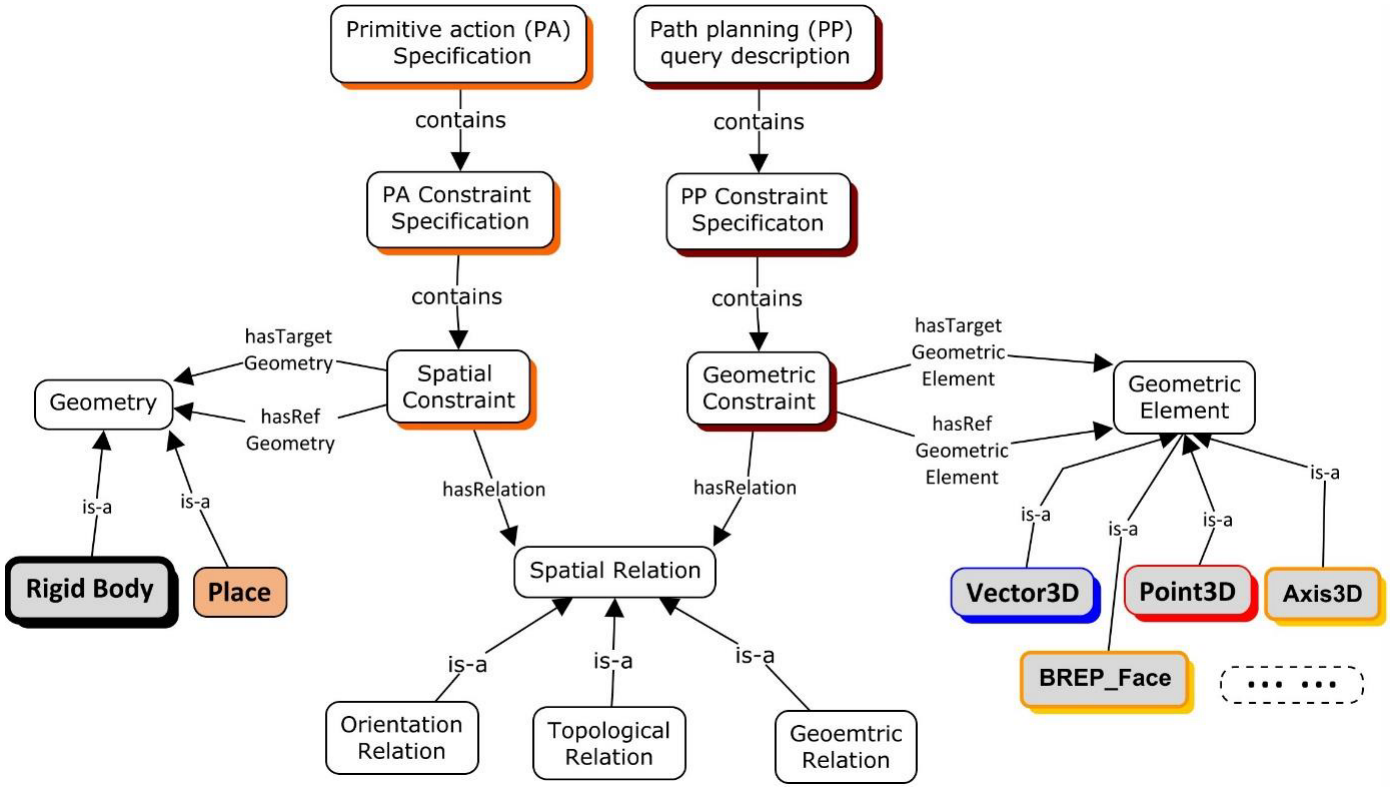


FIGURE 3.4 – Structure de l'ontologie ASK

3.3.4 Phase d'implémentation

Cette section décrit en détail l'implémentation de l'ontologie TAMPO. L'architecture générale est présentée dans le paragraphe 3.3.4.1. Celle-ci se décompose en trois niveaux, un niveau « Méta » qui est présenté dans le paragraphe 3.3.4.2, un niveau « Domaine » présenté dans le paragraphe 3.3.4.3, et un niveau « Instance » présenté dans le paragraphe 3.3.4.4. L'ontologie TAMPO est présentée sur la figure 3.5.

3.3.4.1 Architecture générale

TAMPO a été construite de manière similaire à ENVOn-2. Ainsi, celle-ci comporte trois niveaux :

— le niveau « Méta » : Ce niveau décrit les concepts généraux utilisés par notre ontologie ;

— le niveau « Domaine » : Il s'agit du niveau où sont décrits les concepts liés au domaine d'application, à savoir le TAMP ;

— le niveau « Instance » : Dans ce niveau sont décrits les différentes instances utilisées dans nos scénarios.

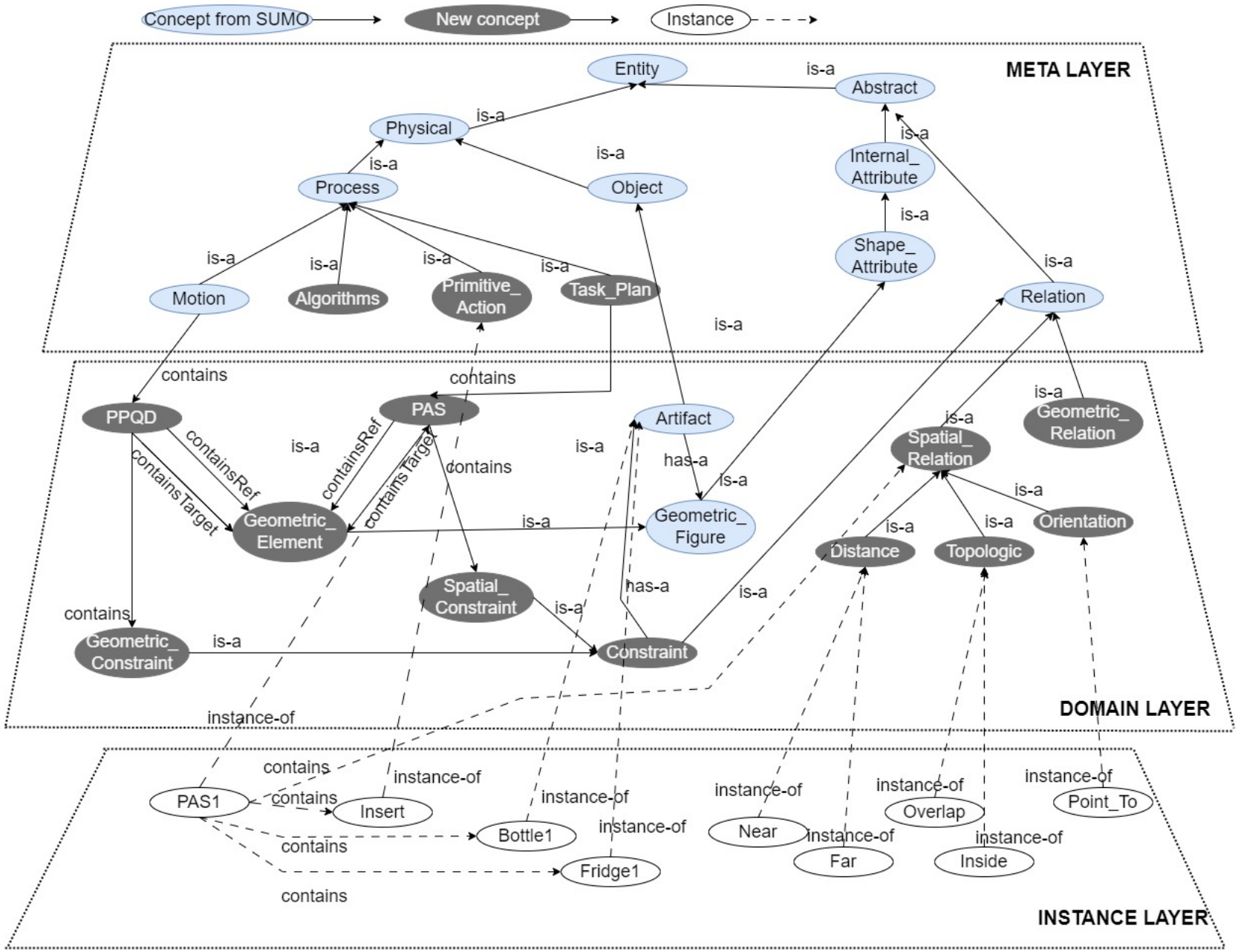


FIGURE 3.5 – Structure de l'ontologie TAMPO

3.3.4.2 Description du niveau Méta

Il s'agit du niveau qui permet de représenter les informations génériques, qui ne sont pas dépendantes du domaine. Il s'agit ici de concepts tels que *Object*, *Process* ou *Attribute* ... La définition de ces concepts est donnée dans le tableau 3.5 :

Classe	Définition
Entity	La classe universelle des individus. C'est la classe mère de l'ontologie.
Physical	Entité ayant une localisation dans l'espace-temps.
Abstract	Propriétés ou qualités distinctes de toute incarnation dans un support physique
Object	Classe des objets ordinaires (objets physiques, régions géographiques ...)
Process	Chose qui se produit et ayant des parties/étapes temporelles
Attribute	Qualité ne pouvant être réifiée en sous-classe

TABLE 3.5 – Classes du niveau Méta de l'ontologie TAMPO

3.3.4.3 Description du niveau Domaine

Les concepts du niveau « Domaine » sont des concepts spécifiques au TAMP. Ces concepts modélisent les informations pour la planification de trajectoires, pour la planification de tâches et également pour la modélisation des relations spatiales.

Concepts liés à la planification de trajectoires Nous retrouvons des concepts tels que :

- les algorithmes pour la planification de trajectoires ;
- la description de la requête de planification de trajectoires qui contient :
  - l'identité de l'objet manipulé ;
  - la position de la configuration finale à atteindre ;
  - les CG à appliquer.
- l'ensemble des CG générées.

Concepts liés à la planification de tâches Nous retrouvons les concepts liés à la tâche à réaliser comme :

- l'algorithme de planification de tâches ;
- les plans de tâches décomposant la tâche à réaliser en séquence de tâches primitives ;

— les différentes tâches primitives des plans ainsi que leurs spécifications :
  — L'identité de la tâche (insérer, déplacer ...) ;
  — l'objet manipulé ;
  — les CS qui ont été manuellement assignées.
  — une référence (*Object* ou *Place*) permettant d'appliquer les CS.
— l'ensemble des contraintes spatiales.

Modélisation des relations spatiales Cette partie permet, à l'aide de règles de raisonnement, d'inférer les contraintes géométriques à partir des contraintes spatiales. Les relations spatiales utilisées sont les relations géométriques (*Against*, *Colinear*, *Parallel* ...), topologiques (*Disjoint*, *Inside*, *Touch* ...), orientation (*Back*, *Right*, *PointTo* ...) et la distance (*Far*, *Near* ...).

3.3.4.4 Description du niveau Instance

Comme pour l'ontologie ENVOn-2, ce niveau est présenté plus tard, dans la partie 5.4 et 5.5, quand les différentes applications seront instanciées dans l'ontologie.

3.3.5 Phase d'évaluation

Dans cette partie, nous allons évaluer notre ontologie, pour voir si elle a été construite correctement. Il faut vérifier que notre ontologie est cohérente, que les classes définies soient en accord avec les besoins définis dans le paragraphe 3.3.1. L'utilisation du raisonneur Pellet de *Protégé* nous retourne l'information que notre ontologie est cohérente.

Il faut également vérifier les réponses obtenues à nos QC. Toutefois, cette partie est validée dans les parties 5.4 et 5.5, quand notre ontologie est instanciée. Cela est fait sur plusieurs scénarios pour être le plus général possible.

3.4 Synthèse sur la construction des ontologies

Dans ce chapitre, nous avons décrit la construction de deux ontologies de domaine, qui s'appuient sur SUMO, l'une pour la modélisation de l'environnement - ENVOn-2 -, et l'autre pour la modélisation des concepts liés au TAMP - TAMPO -.
La première ontologie permet la modélisation de l'environnement 3D dans lequel se déroule

une tâche. L'environnement est décomposé en trois niveaux d'abstraction distincts (sémantiques, topologiques et géométriques). Ces différents niveaux permettent, par la suite, l'utilisation de stratégies de planifications particulières.

La seconde ontologie modélise les informations liées à la planification de tâches et de trajectoires. Elle permet de proposer un plan de tâches pertinent pour la tâche à réaliser tout en générant de manière automatique des contraintes géométriques pour la planification de trajectoires.

Chapitre 4

Proposition d'une méthodologie pour le couplage sémantique de planifications de tâches et de trajectoires

4.1 Introduction

Dans le chapitre 3, nous avons présenté deux ontologies. La première, ENVOn-2, permettant de modéliser les informations relatives à l'environnement 3D dans lequel se déroule la tâche à réaliser. La seconde ontologie, TAMPO, permettant quant à elle de modéliser les informations relatives au TAMP.
Dans ce chapitre, nous présentons une méthodologie pour le couplage sémantique de la planification de trajectoires et de tâches basée ontologie. Dans la partie 4.2, nous décrivons l'approche globale que nous avons développée. Dans la partie 4.3, nous présentons comment cette méthodologie permet d'améliorer la planification de trajectoires. Puis, dans la partie 4.4, nous présentons comment notre méthodologie permet de contrôler la planification de tâches afin de proposer un plan de tâches pertinent au regard de la tâche à réaliser, pour que la planification de trajectoires utilisées par la suite soit la plus efficace possible.

4.2 Approche globale

Dans cette partie, nous présentons notre méthodologie originale pour le couplage sémantique de la planification de tâches et de trajectoires, basée ontologies.

Cette méthodologie s'appuie sur les travaux de [Zhao, 2019], dans lesquels une première étape vers le couplage sémantique, entre une unique tâche primitive et un planificateur de trajec-

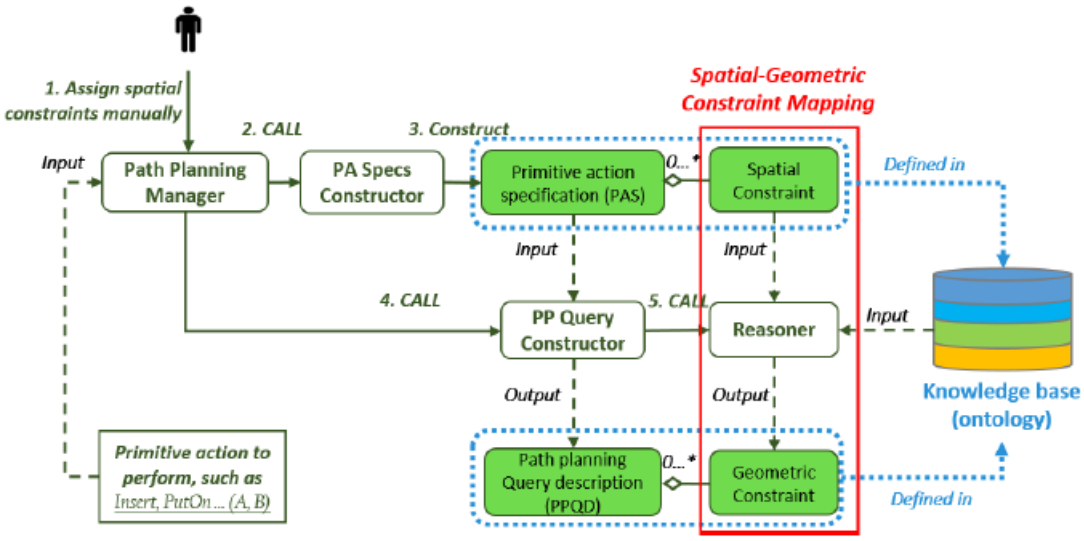


FIGURE 4.1 – Processus de construction d'un PAS et de génération d'une PPQD correspondante

toires, a été proposée. Ceux-ci permettent d'inférer automatiquement une requête de planification de trajectoires pour une tâche primitive d'un plan de tâches. Deux principales classes sont mises en oeuvre :

— La PAS (Primitive Action Specification) qui spécifie l'identité de l'action primitive, l'objet manipulé et les contraintes spatiales exprimées au niveau de la tâche ;

— La PPQD (Path Planning Query Description) qui spécifie l'objet manipulé, les configurations initiale et finale et les contraintes géométriques de la requête de planification de trajectoire.

La figure 4.1 décrit le fonctionnement général des travaux de [Zhao, 2019]. Un opérateur humain assigne, manuellement, un ensemble de contraintes spatiales relatives à la tâche primitive à effectuer. Puis, une PAS est construite. Ensuite, le constructeur de requête de planification de trajectoires construit automatiquement une PPQD. Cette dernière spécifie l'objet manipulé (qui est celui spécifié dans la PAS), les configurations initiale et finale ainsi que les contraintes géométriques permettant de réduire l'espace de recherche de l'algorithme de planification de trajectoires mis en oeuvre (ici, le Bi-RRT). Ces contraintes géométriques, sont inférées, à partir des contraintes spatiales, par un raisonneur qui utilise des règles Semantic Web Rule Language (SWRL). Ces règles sont prédéfinies dans les ontologies ENVOn et ASK, présentées dans la partie 2.3.4.3.2, du chapitre 2.

Ces travaux permettent d'améliorer les performances de la planification de trajectoires (temps de calcul, pertinence de la trajectoire proposée) grâce à un meilleur contrôle sémantique. Cela permet également d'apporter plus de flexibilité sur la définition des contraintes géométriques, permettant à l'approche de s'adapter à différents cas d'utilisation facilement.

Toutefois, les travaux de [Zhao, 2019] ne concernent qu'une tâche primitive, et non pas un plan de tâches complet. Notre contribution permet d'améliorer ces travaux en les étendant à un plan de tâches complet et en permettant également de comparer des plans alternatifs. De plus, dans l'optique d'une démarche utilisable en Réalité Virtuelle, nous offrons à un opérateur humain le choix de faire une proposition de plan de tâches, qui est également comparée. Enfin, cette proposition s'appuie sur ENVOn-2 et TAMPO, nos deux ontologies proposées dans le chapitre 3.

La figure 4.2 présente notre proposition pour le couplage sémantique de la planification de trajectoires et de la planification de tâches. Celle-ci porte sur les points suivants :

— Dans un premier temps, un planificateur de tâches propose un plan de tâches. Un opérateur humain peut également en proposer un autre, auquel cas, les différents plans sont évalués par un raisonneur, à l'aide de l'ontologie TAMPO présentée dans le chapitre 3, et le meilleur plan de tâches est retenu ;

— Puis, des contraintes spatiales, exprimées au niveau tâche, sont assignées manuellement par un opérateur pour chacune des tâches primitives ;

— Puis, une PAS est construite spécifiant les informations relatives à la tâche primitive à effectuer. Un raisonneur infère, ensuite, automatiquement les contraintes géométriques au niveau de la planification de mouvements à partir des contraintes spatiales exprimées au niveau de la tâche primitive ;

— Enfin, une PPQD est construite automatiquement pour chaque tâche primitive. La planification de trajectoires est contrôlée par les contraintes géométriques précédemment inférées.

Le diagramme de séquence présenté par la figure 4.3 décrit en détail comment se déroule notre approche. La partie en jaune présente les étapes liées au contrôle de la planification de tâches. La partie 4.4 décrit plus précisément cette planification de tâches. La partie en vert présente quant à elle les étapes liées à la planification de trajectoires. Celle-ci est présentée en détail dans la partie 4.3.

Un *opérateur humain* initie le lancement en appelant le *Gestionnaire des planificateurs*. Ce dernier appelle le *Planificateur de tâches* qui décompose la tâche à réaliser en une séquence d'actions primitives, appelée plan de tâches. Ce plan de tâches vient enrichir l'ontologie TAMPO puis est proposé à l'opérateur humain qui peut l'accepter ou en construire un autre. Si un autre plan de tâches est construit, il enrichit également l'ontologie TAMPO.

Si plusieurs plans de tâches ont été instanciés dans TAMPO, le raisonneur Pellet (qui est natif à *Protégé*) est appelé pour les évaluer. Cette étape est décrite dans le paragraphe 4.4.4. Le meilleur plan de tâches est alors retourné à l'*opérateur humain*.

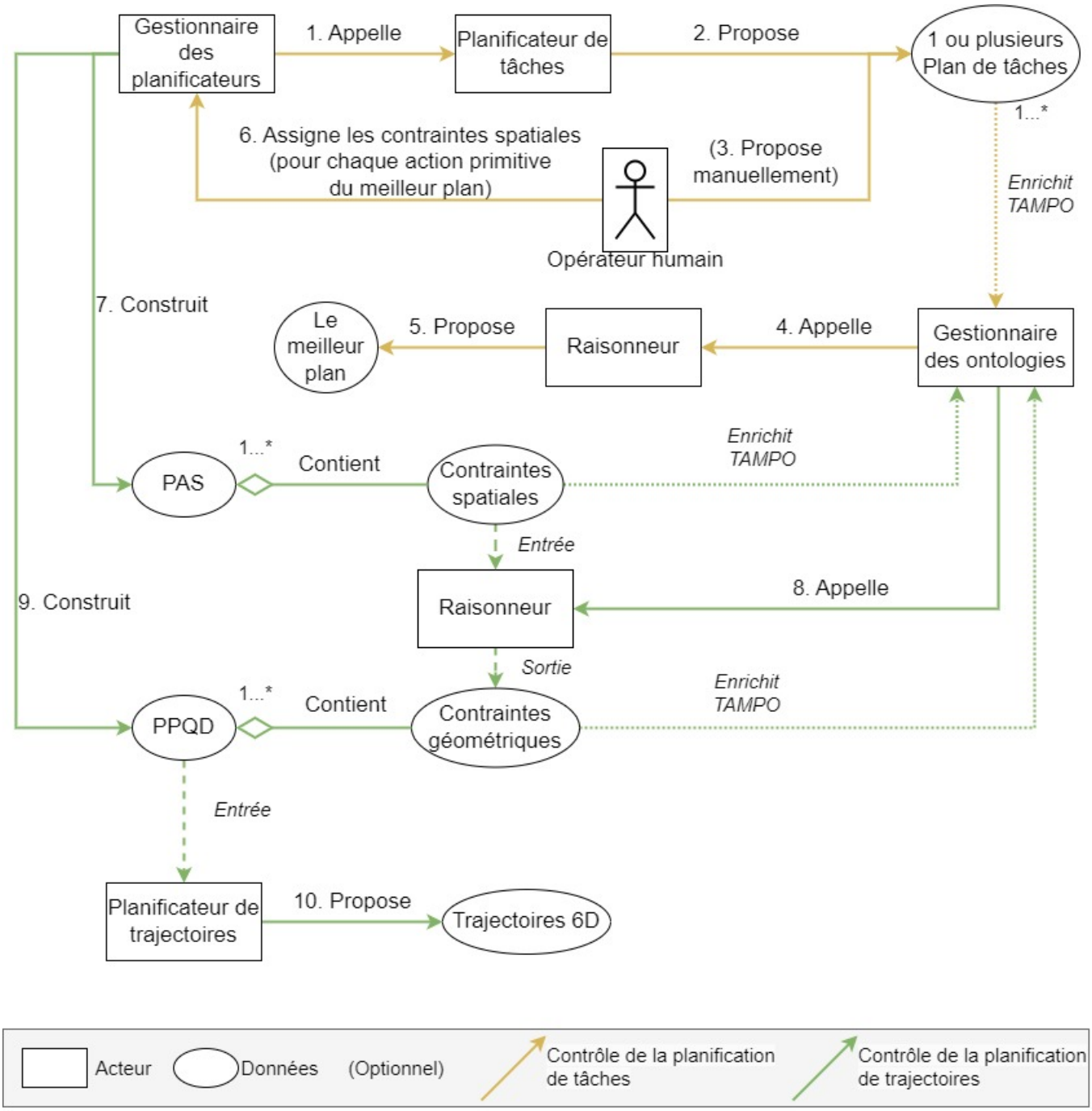


FIGURE 4.2 – Processus du couplage sémantique TAMP

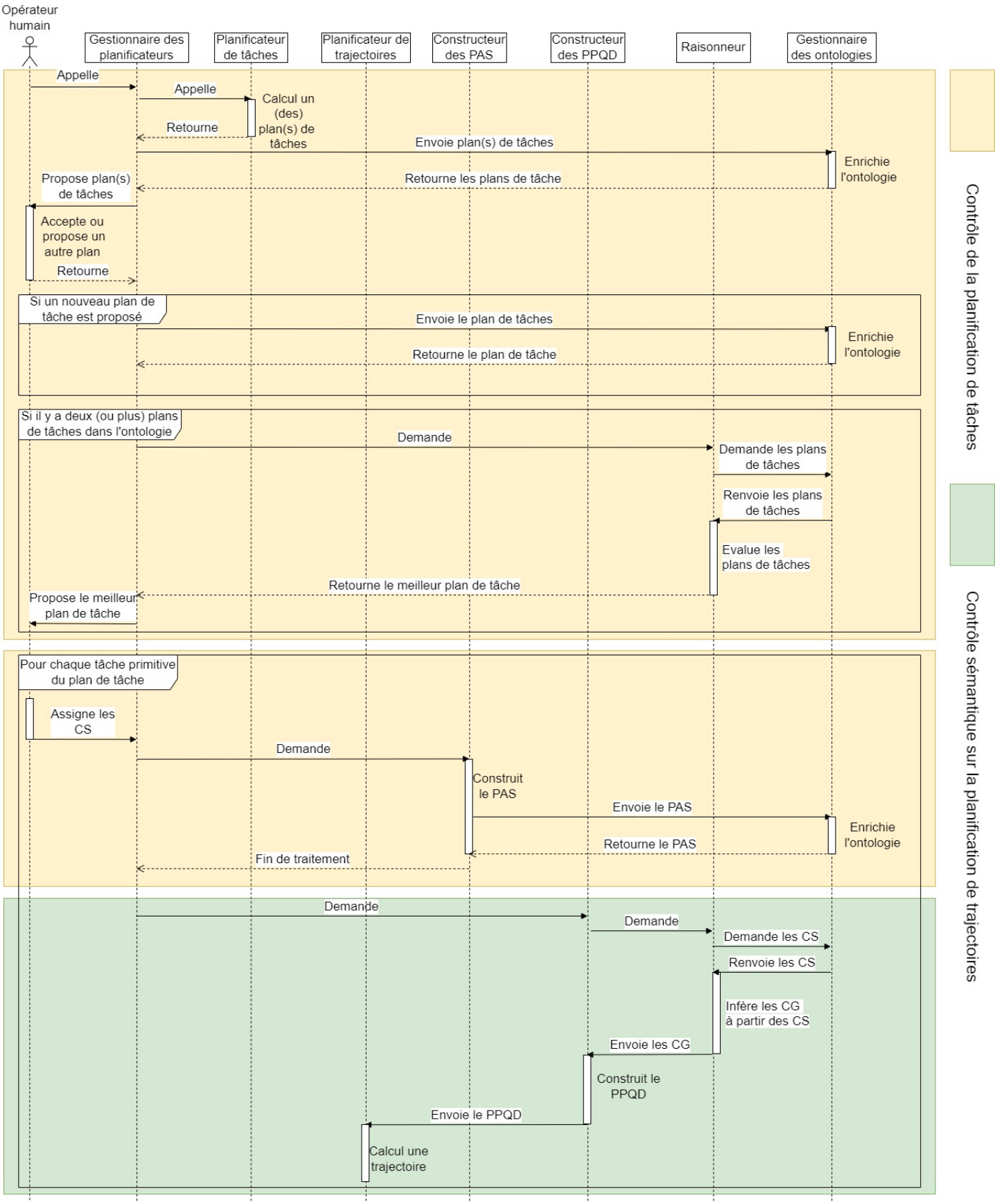


FIGURE 4.3 – Diagramme de séquence présentant notre approche globale

Puis pour chaque tâche primitive du plan de tâches (processus décrit par le cadre vert), l'*Opérateur humain* assigne ensuite des contraintes spatiales. Le *Gestionnaire des planificateurs* appelle ensuite le *Constructeur des PAS*, qui construit la PAS. La PAS, décrit dans le paragraphe 3.3.2 du chapitre 3, contient l'identité de la tâche primitive, l'objet manipulé, la géométrie de référence et les CS (Contraintes Spatiales) associées à la tâche primitive. Une fois la PAS construite, elle enrichie l'ontologie TAMPO.
Le *Gestionnaire des planificateurs* appelle ensuite le *Constructeur des PPQD*, qui à son tour appelle le raisonneur Pellet. Ce dernier, à partir des CS qui peuplent TAMPO, infère des CG (Contraintes Géométriques). Ces CG sont ensuite retournées au *Constructeur des PPQD*, qui peut maintenant construire la PPQD relative à la tâche primitive courante. La PPQD est également décrite dans le paragraphe 3.3.2 du chapitre 3. Elle est constituée de l'objet manipulé, de la configuration finale à atteindre ainsi que des CG qui viennent d'être inférées. Cette PPQD est finalement envoyée au *Planificateur de trajectoires* qui, à l'aide des informations relatives à l'environnement décrites dans ENVOn-2, utilise un algorithme (ou une combinaison d'algorithmes) pour trouver une trajectoire réalisable. Cette partie est décrite dans le paragraphe 4.3.

Notre méthodologie permet de coupler sémantiquement la planification de tâches et la planification de trajectoires, ce qui permet :

— 1) D'améliorer le contrôle sémantique sur la planification de trajectoires pour améliorer les performances de notre planificateur de trajectoires pour une tâche primitive d'un plan de tâches. Ceci est détaillé dans la partie 4.3 ;

— 2) D'améliorer le contrôle de la planification de tâches en proposant un plan de tâches complet et pertinent vis-à-vis de la tâche à réaliser (c'est-à-dire réalisable le plus simplement possible par un planificateur de trajectoires), ce qui est décrit dans la partie 4.4.

4.3 Contrôle sémantique sur la planification de trajectoires

4.3.1 Introduction

Afin de valider les différentes tâches du PLM dans un environnement virtuel, une étape clef consiste à trouver une trajectoire réalisable prouvant la faisabilité du scénario. Pour cela, la planification automatique de trajectoires a commencé à être étudiée à partir des années 1980 par la communauté robotique. Toutefois, ces techniques utilisent principalement des informations géométriques, ce qui engendre des limites car elles peuvent produire des résultats peu pertinents, avoir des temps de calcul longs, voire échouer.
En s'inspirant du comportement et de la façon de réfléchir des humains, de nouvelles approches, considérant des informations de plus haut niveau d'abstraction, ont vu le jour. Ces nouvelles

techniques utilisent principalement des informations topologiques et sémantiques. Malgré des améliorations montrées dans le paragraphe 2.3.1.2 du chapitre 2.3, les performances pourraient encore être améliorées.
Ainsi, pour améliorer les performances de planification de trajectoires, pour une tâche primitive unique, nous proposons, dans le paragraphe 4.3.2, d'utiliser une méthode réduisant l'espace de recherche d'un algorithme probabiliste, dans le paragraphe 4.3.3, d'utiliser une méthode permettant de combiner différents planificateurs au sein d'un même requête de planification et enfin, dans le paragraphe 4.3.4, nous mettons en œuvre différentes stratégies pour améliorer les performances de la planification de trajectoires.

4.3.2 Réduction de l'espace de recherche d'un algorithme probabiliste

Valider la faisabilité d'une tâche par simulation dans un environnement virtuel peut être compliqué si ce dernier est fortement contraint géométriquement. Dans ces cas, l'utilisation d'un algorithme probabiliste est très intéressant car cela permet d'être sûr de trouver une solution, si une existe. Cependant, de tels algorithmes sont très coûteux (temps de calcul, nombre d'itérations ...) et ne fournissent pas nécessairement une trajectoire pertinente. Cela s'explique car il s'agit d'algorithmes dits « aveugle », tirant des échantillons dans tout l'espace libre, sans considération sur la pertinence du tirage.
Dans nos travaux, nous avons choisi d'utiliser un algorithme Bi-RRT qui est décrit dans la partie 2.3.1.1. Toutefois, les trajectoires obtenues ainsi que les performances obtenues lors des précédents travaux ([Zhao, 2019]), doivent être améliorées et l'utilisation de l'algorithme Bi-RRT doit être contrôlée. Nous proposons de réduire la zone de tirage de cet algorithme afin d'éviter des zones peu ou pas pertinentes. Cela permet de réduire le nombre de tirages nécessaire tout en améliorant la pertinence de la trajectoire puisqu'aucune configuration n'est tirée dans des zones non souhaitées.

Pour cela, et en considérant le modèle multi-niveaux de [Cailhol et al., 2019], nous allons utiliser un algorithme A* sur l'octree.
L'octree permet d'avoir un modèle de l'espace libre décomposé en un ensemble de cubes de différentes tailles. L'algorithme A*, qui est un algorithme de recherche de chemin, va parcourir ces différents cubes jusqu'à trouver un chemin et progresse en minimisant une fonction de coût tel que $f(n) = g(n) + h(n)$, où $g(n)$ est la distance parcourue depuis la position initiale (distance prise au centre du cube d'indice n) et $h(n)$ la distance euclidienne entre le cube d'indice n jusqu'à la position finale.
Le résultat est un tunnel 3D d'espace libre qui pourra alors être considéré comme un nouvel sous espace. C'est dans ce dernier que le tirage des configurations de l'algorithme bi-RRT seront alors tirés.

Le principe de cette approche est expliqué dans la figure 4.4. Il s'agit d'un exemple 2D simple pour illustrer le fonctionnement de cette contribution, mais cela fonctionne de la même manière en 3D. Sur la figure 4.4a), l'espace libre est représenté en vert. Un algorithme probabiliste va effectuer des tirages dans tout cet espace. Sur la figure 4.4b), la décomposition de l'espace en quadtree est représenté (le quadtree est en 2D ce qu'est l'octree en 3D). En appliquant l'algorithme A* sur le quadtree, on obtient la figure 4.4c), où le chemin jaune représente le nouvel sous espace. Les tirages sont effectués dans ce dernier, évitant les zones non pertinentes.

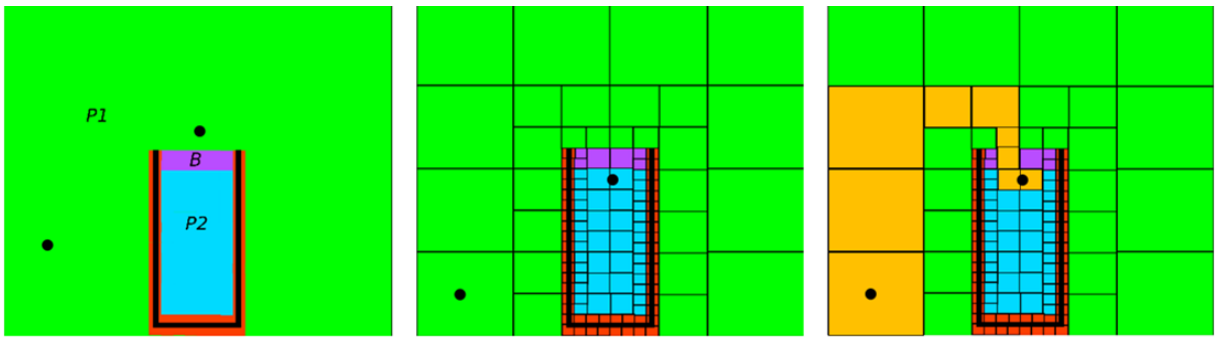


FIGURE 4.4 – Utilisation de l'algorithme A* sur un quadtree

4.3.3 Utilisation d'une stratégie multi-planificateurs

Dans la littérature, quand une trajectoire est recherchée, on fait appel à un unique planificateur de trajectoires. Nous pensons que l'utilisation d'un unique planificateur de trajectoires n'est pas pertinent et qu'il serait intéressant d'appeler différents planificateurs au sein de la même requête, en fonction de la complexité des lieux à traverser. Afin de valider cette idée, deux planificateurs peuvent être appelés en fonction de la sémantique locale : une interpolation linéaire si l'environnement n'est pas contraint, un algorithme Bi-RRT sinon.
Pour cela, nous nous appuyons toujours sur le modèle multi-niveaux de [Cailhol et al., 2019], principalement sur les niveaux topologiques et sémantiques. En effet, dans un premier temps, il faut commencer par découper le chemin pour déterminer sur quelles portions nous allons pouvoir appeler tel ou tel planificateur. Pour cela, nous considérons les étapes topologiques. Ensuite, comme de la sémantique est associé à une étape topologique, nous nous servons de celle-ci pour choisir le planificateur de trajectoires.

Le processus pour le choix du planificateur est décrit par le diagramme de séquence présenté sur la figure 4.5. Pour chaque étape topologique, le *Gestionnaire des planificateurs* étudie le ni-

veau de complexité du lieu à traverser. Ainsi, s'il n'y a pas d'obstacles entre la configuration initiale et la configuration finale (qui peut dans notre cas être une configuration intermédiaire), le planificateur effectue une interpolation linéaire. Sinon, l'algorithme Bi-RRT est appelé pour calculer une trajectoire. Ce dernier peut être précédé par une réduction de l'espace de recherche, comme décrit dans le paragraphe 4.3.2.

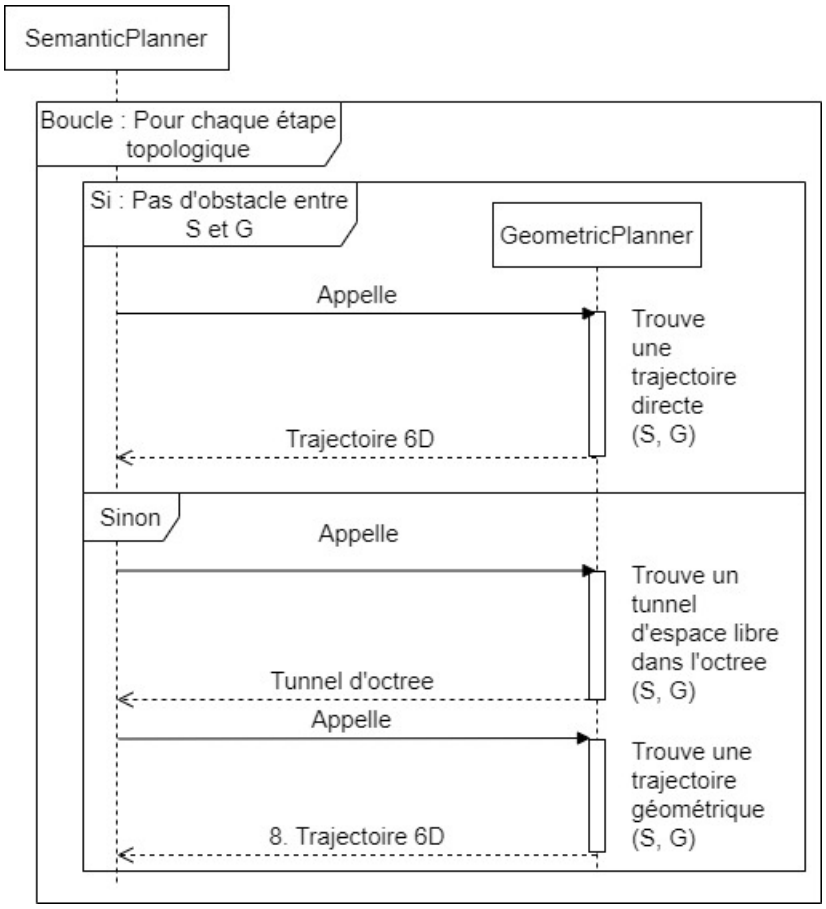


FIGURE 4.5 – Diagramme de séquence de la planification de trajectoire fine (MG*TO)

4.3.4 Stratégies de planification de trajectoires proposées

Basés sur les deux contributions précédemment présentées et sur les stratégies G, GT (présentées dans le paragraphe 2.3.1.2.2) du chapitre 2.3) et GTO (présentée dans le paragraphe 2.3.3.1.2 du chapitre 2.3) plusieurs stratégies ont été développées :

— la stratégie *MGTO*. Cette stratégie multi-planificateur est basée sur la stratégie *GTO*. Il s'agit d'une stratégie ayant deux phases de planification. La planification grossière est identique à celle de *GTO*. En revanche, lors de la planification fine, cette stratégie offre la possibilité d'utiliser un autre algorithme pour résoudre la recherche d'une trajectoire d'une étape topologique.

Cette stratégie utilise des informations sémantiques sur l'environnement présentes dans l'ontologie ENVOn-2, concernant le niveau de difficulté à traverser un lieu. Ainsi, si un lieu n'est pas ou peu contraint, l'utilisation d'un algorithme complexe tel que le Bi-RRT n'est pas nécessaire et dans ce cas, une interpolation linéaire entre la configuration initiale et finale est suffisante. Dans le cas contraire, l'algorithme Bi-RRT est appelé.

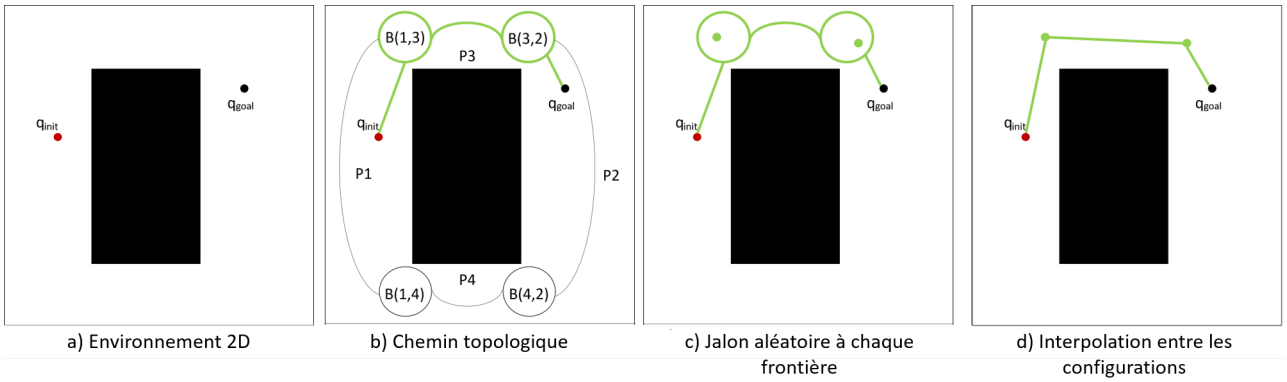


FIGURE 4.6 – MGTO

Sur l'exemple décrit sur la figure 4.6, où l'objectif est d'aller de la configuration q_{init} à la configuration q_{goal}, il n'y a pas de contraintes géométriques fortes sur l'environnement, une interpolation linéaire sur les différentes étapes topologiques est alors possible, rendant le chemin obtenu plus pertinent que lorsque la stratégie *GT* est appliquée (résultat présentée par la figure 2.8 dans le chapitre 2.3).

— La stratégie G* qui est une stratégie implémentant le paragraphe 4.3.2. Il s'agit de réduire l'espace de recherche quand un algorithme probabiliste (comme le bi-RRT) est appelé. Cette stratégie est étendue aux stratégies GT, GTO et MGTO, et sont respectivement appelées G*T, G*TO et MG*TO.

— La stratégie MG*TO est une combinaison des approches présentées dans les paragraphes 4.3.2 et 4.3.3 et de la stratégie *GTO*. Son fonctionnement est présentée par le diagramme de séquence de la figure 4.5.

4.3.5 Synthèse sur la planification de trajectoires par contrôle sémantique

Nous avons dans cette partie envisagé l'utilisation conjointe de 1) l'utilisation de différents algorithmes de planification de trajectoires ou des combinaisons de ceux-ci; 2) l'utilisation d'informations d'un niveau d'abstraction plus élevé que les données purement géométriques; 3) l'utilisation d'ontologies pour représenter les connaissances liées à la tâche à accomplir.
Toutefois, nous nous intéressons dans cette partie uniquement à l'amélioration de la planification de trajectoires. Il faut, en plus, considérer la planification de tâches.

4.4 Contrôle de la planification de tâches

4.4.1 Introduction

Pour la validation des tâches du PLM, valider seulement le mouvement n'est pas suffisant. Il est important de pouvoir raisonner au niveau de la tâche à réaliser et donc, considérer la planification de tâches.
L'objectif de la planification de tâches est de décomposer une tâche complexe en une séquence de tâches primitives. Il existe plusieurs critères spécifiques afin d'optimiser la solution, chacun ayant ses avantages et ses limites. Dans une première partie, nous allons présenter des concepts généraux sur la planification de tâches (paragraphe 4.4.2). Différents critères pour la planification de tâches sont présentés dans le paragraphe 4.4.3, et nous définissons les critères qui doivent être considérés pour la planification de tâches (en ayant connaissance qu'il doit être étudié pour une future planification de trajectoires). Nous proposons ensuite une méthode pour évaluer plusieurs plans de tâches dans le paragraphe 4.4.4, et définissons une fonction de coût pour cela (paragraphe 4.4.5).

4.4.2 Concepts généraux

Les différentes approches de planification de tâches ont été présentées dans la partie 2.3.2. Toutefois, il est possible de compléter ces informations en présentant le Langage de Description du Domaine de la Planification (PDDL), qui a permis, par une description formelle des problèmes de planification, d'établir un cadre pour comparer les différentes solutions proposées (4.4.2.1). Ce langage a également permis la création du Concours International de Planification (CIP), dont un bref historique est présenté dans la partie 4.4.2.2.

4.4.2.1 Définition du PDDL

En 1998, Drew McDermott a publié la première version du PDDL ([McDermott et al., 1998]). Il s'agit d'un langage formel de description de problèmes de planification qui est largement utilisé dans la recherche en planification automatisée. Le PDDL permet de décrire formellement les états possibles du monde, les actions qui peuvent être effectuées pour modifier ces états, les objectifs que l'on souhaite atteindre et les contraintes spécifiques liées au problème de planification. Ce langage est utilisé pour représenter de nombreux types de problèmes de la planification comme la planification classique, la planification temporelle ou encore la planification sous incertitudes, notamment présentées dans le chapitre 2.3. Ce langage a permis d'importantes avancées dans la recherche en matière de planification grâce aux nombreux avantages qu'il propose. Tout d'abord, le PDDL fournit un format standardisé pour représenter les pro-

blèmes de planification, ce qui facilite la comparaison des approches de planification et la réutilisation de code entre différents systèmes de planification. En outre, le PDDL est un langage formel expressif, qui permet aux utilisateurs de représenter des problèmes de planification complexes de manière compacte et intuitive. Le PDDL offre également une grande flexibilité pour la représentation de différents types de problèmes de planification, ce qui en fait un outil puissant.

Le PDDL est maintenu par la communauté de la planification automatisée, qui travaille activement à son développement et à son amélioration. Dans ce sens, plusieurs évolutions ont été faites :

— PDDL 2.1 : le PDDL était très limité dans sa capacité à représenter les problèmes de planification. Le PDDL2.1, publiée en 2003, a introduit plusieurs extensions pour améliorer la capacité du PDDL à représenter les problèmes de planification, notamment la possibilité de définir des préconditions négatives et des effets conditionnels pour les actions.

— PDDL 3.0 : publiée en 2005, le PDDL3.0 a introduit plusieurs nouvelles fonctionnalités pour améliorer la capacité du PDDL à représenter les problèmes de planification, notamment la possibilité de définir des domaines et des problèmes avec des durées, des états initiaux probabilistes et des coûts pour les actions.

— PDDL+ : le PDDL+ est une extension du PDDL qui a été proposée en 2006 pour permettre la représentation de problèmes de planification plus complexes, tels que la planification multi-agent, la planification avec des contraintes de ressources et la planification avec des coûts non-linéaires.

— PDDL 4.0 : publiée en 2014, le PDDL4.0 a introduit plusieurs nouvelles fonctionnalités comme la possibilité de définir des domaines et des problèmes avec des états initiaux partiellement observables, des durées stochastiques et des préférences pour les plans.

Ces nombreuses évolutions montrent l'importance du PDDL comme outil pour la planification.

4.4.2.2 Compétition Internationale de Planification

En parallèle du développement de ce langage, il y a eu la création des CIP en 1998. Afin de pouvoir comparer les différents participants, l'utilisation du PDDL s'est imposée et est ainsi devenu un standard dans la recherche en planification.
Ces compétitions prennent depuis place tous les deux ans et attirent de nombreux chercheurs et offrent une très grosse visibilité pour présenter ses recherches. En effet, cela permet d'évaluer et de comparer les performances des derniers planificateurs / algorithmes développés.
La première édition ne proposait qu'une épreuve de planification classique mais cela a rapidement évolué pour inclure des épreuves plus complexes, telles que la planification temporelle, la

planification probabiliste ou la planification multi-agents.
Les résultats des différents planificateurs obtenus lors de ces compétitions permettent ainsi à la communauté de pouvoir se positionner sur les planificateurs à utiliser, selon leurs besoins.

4.4.3 Définition de nos besoins

Pour valider des tâches numériquement, il est important d'avoir des plans de tâches cohérents et pertinents. Il existe plusieurs critères pour évaluer un plan de tâches. Parmi eux, nous pouvons citer :

— La flexibilité : cela correspond à sa capacité à s'adapter à des changements imprévus, comme des contraintes ou des objectifs modifiés. Un plan de tâches flexible est capable de trouver des solutions alternatives en cas d'imprévus, sans compromettre les objectifs principaux;

— La robustesse : cela correspond à sa capacité à résister à des perturbations. Un plan de tâches robuste est capable de continuer à fonctionner même en présence d'incertitudes ou de variations dans un environnement;

— La stabilité : cela correspond à sa capacité à maintenir une solution dans le temps. Un plan de tâches stable est capable de minimiser les perturbations, les retards ou les coûts, même en présence de variations dans l'environnement ou d'évolution des objectifs;

— La simplicité : cela peut inclure des critères tels que la complexité du plan, le nombre d'étapes nécessaires pour accomplir les tâches, la clarté des instructions du plan, etc;

— L'efficacité : cela peut inclure des critères comme le temps nécessaire pour calculer ou pour exécuter un plan de tâches. Cela peut également inclure le nombre de ressources utilisées;

— La résilience : cela correspond à la capacité du planificateur à réagir lorsque le plan n'est plus valide.

Dans notre cas, la planification de tâches fournit un plan pour qu'un planificateur de trajectoires puisse être exécuté et trouver une trajectoire réalisable avec les meilleures performances possibles. Cela signifie que certains critères spécifiques sont à considérer lors de l'élaboration du plan de tâches.

L'état de l'art de la planification de trajectoires montre que les performances de celle-ci diminuent très fortement si elle est effectuée dans un environnement ayant de très fortes contraintes géométriques. Or dans notre cas, nous cherchons à simuler des tâches sous de fortes contraintes géométriques pour valider la conception de produits de plus en plus intégrés. Ainsi, pour obtenir un plan de tâches qui permet d'avoir de bonnes performances sur la planification de trajectoires, il faut tenir compte :

— De la longueur du plan de tâches. En effet, chaque action primitive engendre une requête de planification de trajectoires. Or, plus il y a de requêtes différentes, plus long sera le temps d'exécution du planificateur de trajectoires. Cela permet également de réduire la possibilité d'un échec.

— Des contraintes géométriques que génèrent les différentes tâches primitives. Un plan générant moins de contraintes sera plus facile à résoudre pour un planificateur de trajectoires et donc, sera plus performant (moins de risque d'échec, temps de calcul plus rapide...)

Maintenant que nos besoins ont été définis, nous pouvons choisir quel planificateur de tâches utiliser. Notre choix s'est porté sur l'algorithme Fast-Forward (FF) ([Hoffmann, 2001]).
Il s'agit d'un algorithme de recherche heuristique créé par Jörg Hoffmann. Il a été introduit en 2001. L'algorithme FF utilise une approche de recherche dans l'espace des états avant pour trouver un plan pour un ensemble de conditions initiales et un objectif donné. Il utilise une heuristique basée sur l'analyse de causalité pour guider la recherche vers une solution optimale. La fonction heuristique estime la distance jusqu'à l'état d'objectif en calculant la distance de plan relaxée, qui est le nombre minimum d'actions nécessaires pour atteindre l'état d'objectif si toutes les préconditions des actions étaient ignorées. La distance de plan relaxée est calculée à l'aide d'un graphe de causalité qui représente les liens de causalité entre les sous-objectifs du problème.
Les avantages de l'algorithme FF sont sa rapidité, sa flexibilité et sa capacité à résoudre une large gamme de problèmes de planification.
De plus, cet algorithme a participé plusieurs fois avec succès au CIP, remportant souvent la première place. En particulier, lors du CIP de 2008, FF a remporté la première place dans la catégorie de planification temporelle et a été classé deuxième dans la catégorie de planification classique. Les performances de FF ont été attribuées à la qualité de sa fonction heuristique et à sa capacité à exploiter efficacement la structure du problème pour trouver des plans de haute qualité.

4.4.4 Évaluation de différents plans de tâches

La plupart des planificateurs proposent un unique plan de tâches quand un est trouvé. Cela car le premier plan généré est souvent suffisamment bon pour atteindre l'objectif de la tâche à réaliser. De plus, la recherche du plan optimal peut être très coûteuse en termes de temps de calcul et de ressources informatiques, l'option du premier plan trouvé est souvent favorisée. Toutefois, selon le/les critères définis pour optimiser celui-ci, le plan le plus pertinent n'est pas toujours celui choisi.

Dans le contexte de l'utilisation du TAMP pour la validation de tâches en environnement numérique et afin de réaliser une planification pertinente, il est important d'avoir un plan de tâches pertinent. Il est possible de générer plusieurs plans de tâches afin d'évaluer, selon les contraintes géométriques qu'ils génèrent, leur pertinence. Traditionnellement, cela n'est pas couramment fait car avoir plusieurs plans peut compliquer la sélection du plan à choisir.
Toutefois, cela peut avoir plusieurs avantages, notamment dans des environnements complexes ou quand la tâche est incertaine :

— La redondance : en ayant plusieurs plans de tâches, si l'un d'entre eux échoue ou rencontre des problèmes imprévus, il est possible de basculer vers un autre plan de tâches pour poursuivre la mission. Cela permet d'augmenter la robustesse et la fiabilité du système.

— L'adaptabilité : différents plans de tâches peuvent être conçus pour répondre à des scénarios différents ou à des objectifs différents. Par exemple, un plan de tâches peut être conçu pour minimiser le temps de réalisation, tandis qu'un autre peut être conçu pour minimiser les coûts. Avoir plusieurs plans de tâches permet donc de répondre à des besoins différents.

— L'optimisation : en générant plusieurs plans de tâches, il est possible de comparer et d'évaluer leur performance pour choisir celui qui convient le mieux en fonction des critères de qualité définis. Cette évaluation permet de choisir le plan de tâches le plus performant en termes d'efficacité, de fiabilité, de coût, etc.

Plusieurs plans de tâches peuvent être proposés soit par un planificateur soit par un opérateur humain en fonction de son expérience et ses connaissances.
Si nous avons plusieurs plans de tâches, il faut également définir une démarche permettant de comparer et d'évaluer ces différents plans.

4.4.5 Description de la fonction de coût utilisée

Afin d'évaluer les différents plans de tâches que nous avons, nous utilisons une fonction de coût définie par F() = A + B + C + D où :

— A : représente le niveau de complexité géométrique autour de la configuration initiale ;

— B : représente le niveau de complexité géométrique autour de la configuration finale ;

— C : représente le niveau de complexité géométrique maximale sur la trajectoire ;

— D : le type d'action primitive à réaliser.

Les valeurs de A et B peuvent être extraites à partir de l'ontologie ENVOn-2, dépendant des niveaux de complexités des lieux (*Places*) où se trouvent les configurations initiales et finales.

La valeur de C peut également être extraites à partir de ENVOn-2 en considérant le lieu (*Place*) à traverser ayant le plus haut niveau de complexité.
Enfin, la valeur de D peut être extraite de l'ontologie TAMPO, en fonction du type de l'action primitive à exécuter.

4.4.6 Synthèse sur le contrôle de la planification de tâche

Ce paragraphe montre l'approche qui a été développée pour l'intégration de la planification de tâches dans nos recherches. Celle-ci, par l'utilisation de l'algorithme FF, permet de proposer un plan. Par ailleurs, il est possible à un opérateur humain d'également proposer un plan de tâches si, d'après ses connaissances et son expérience, celui proposé par l'algorithme ne lui convient pas.
Ensuite, les différents plans de tâches sont instanciés dans une ontologie pour le TAMP, TAMPO. Ceux-ci sont ensuite évalués à l'aide des différentes informations contenues dans les ontologies ENVOn-2 et TAMPO. Grâce à cela, le plan le plus pertinent, - celui générant le moins de contraintes géométriques -, peut être choisi pour simplifier par la suite la planification de trajectoires.

4.5 Synthèse générale

Dans cette partie, nous avons proposé une approche basée ontologies pour le couplage sémantique de la planification de tâches et de trajectoires. Cette approche permet d'inférer automatiquement des contraintes géométriques au niveau de la planification de trajectoires, à partir de contraintes spatiales exprimées au niveau tâche et définies par un opérateur humain, pour restreindre l'espace de recherche de la planification de trajectoires. Les contributions présentées dans ce chapitre permettent également de prendre en compte des plans de tâches alternatifs pouvant, en particulier, être proposés par un opérateur humain. Ces alternatives sont ensuite évaluées et comparées pour considérer le plan de tâches le plus pertinent au regard de la tâche à résoudre. Pour ce faire, ces contributions s'appuient sur les deux ontologies, ENVOn-2 et TAMPO, proposées dans le chapitre 3.

Chapitre 5

Implémentation et résultats expérimentaux

5.1 Introduction

Dans ce chapitre, nous allons évaluer notre approche basée ontologie pour la collaboration de planificateurs de tâches et de trajectoires présentée dans le chapitre 4. L'évaluation de nos travaux porte sur :

- La vérification et la validation de nos ontologie ENVOn-2 et TAMPO, appliquées à différents cas d'utilisation, et à différents environnements. Cela consistera à vérifier les réponses retournées à l'ensemble des questions de compétences qui ont été définies dans le chapitre 3 ;
- Valider notre méthodologie basée ontologies pour coupler sémantiquement les planificateurs de tâches et de trajectoires en :
  - Évaluant les stratégies de planifications de trajectoires proposées dans ces travaux pour une action primitive d'un plan de tâches ;
  - Évaluant les plans de tâches proposés par la planification de tâches.

Quatre cas d'utilisation ont été utilisés pour valider l'ensemble de nos besoins. Le premier correspond à l'insertion d'un stylo, mine vers le bas, dans une boîte. Le deuxième correspond à l'insertion de différentes formes géométriques dans des trous de formes correspondantes. Le troisième est un cas plus industriel qui simule l'insertion d'une pompe dans un module d'électronique de puissance. Enfin, le dernier correspond au changement de piles dans un compartiment.

Ce chapitre est organisé de la façon suivante : le paragraphe 5.2 présente les différents outils et logiciels utilisés pour mettre en œuvre nos simulations, le paragraphe 5.3 décrit les différents cas d'utilisation qui sont utilisés et leurs intérêts. La partie 5.4 présente les résultats obtenus par nos stratégies de planification de trajectoires, pour une tâche primitive d'un plan de tâche. Enfin, le

paragraphe 5.5 présente notre méthodologie générale, allant de la planification de tâches (et du choix du plan de tâches) jusqu'à la planification de trajectoires de chacune des tâches primitives du plan retenu.

5.2 Logiciels utilisés

Afin d'effectuer la simulation de nos tâches, deux logiciels ont été utilisés. Le premier est *Virtools*®, un environnement 3D pour la simulation, et le second est *Protégé*, un éditeur d'ontologies permettant de créer et de modifier des ontologies.

5.2.1 Virtools

La plateforme de RV du LGP utilise Virtools Dev 4.1. Ce logiciel a été développé par Virtools SA qui a publié la première version de Virtools en 1997. Cette société a ensuite été rachetée par Dassault Systèmes en 2005 qui a continué de développer le produit. Initialement, ce logiciel est utilisé pour la création d'applications 3D. Toutefois, l'intérêt pour des applications de RV s'est montré de plus en plus fort et un module, le *VR Pack*, a été développé pour répondre aux nouvelles demandes des utilisateurs. Ce module facilite, entre autres, la visualisation 3D et intègre le protocole Virtual-Reality Peripheral Network (VRPN), qui est une bibliothèque fournissant une interface transparente entre une application de RV (tournant sur un ordinateur) et divers dispositifs matériels et logiciels (comme un système de capture de mouvement ou des systèmes haptiques) connectés à un réseau.

L'interface de Virtools Dev 4.1 est présentée par la figure 5.1. Elle est constituée comme suit :

— Un volet haut gauche qui permet la visualisation de la scène 3D et qui permet d'importer, ajouter, sélectionner des éléments ;

— Un volet haut droit qui regroupe les différentes ressources disponibles. Celles-ci peuvent être des éléments graphiques, des modèles d'objets, des Building Blocks (BB) (qui sont décrits dans le paragraphe suivant).

— Un volet bas qui permet de construire des scripts en définissant les comportements des objets virtuels à l'aide de BB ou de modifier les éléments de la scène (position, taille, texture ...).

Les scripts sont donc composés de BB. Ceux-ci correspondent à des fonctions élémentaires. Chaque BB comporte des entrées, des sorties et des paramètres. Les entrées et sorties servent à transmettre un message d'activation entre les différents blocs et les paramètres servent à transférer des données entre les BB.

Virtools propose nativement un certain nombre de BB pré-programmés qui permettent de réaliser des actions simples sur les objets d'une scène. Toutefois, ces fonctions sont limitées et il est nécessaire qu'un développeur puisse programmer ses propres BB. Dans notre laboratoire, les BB sont codés en C++ grâce au logiciel Visual Studio. De plus, il existe au sein même de Virtools un Software Development Kit (SDK), qui permet de développer des BB directement dans le logiciel.

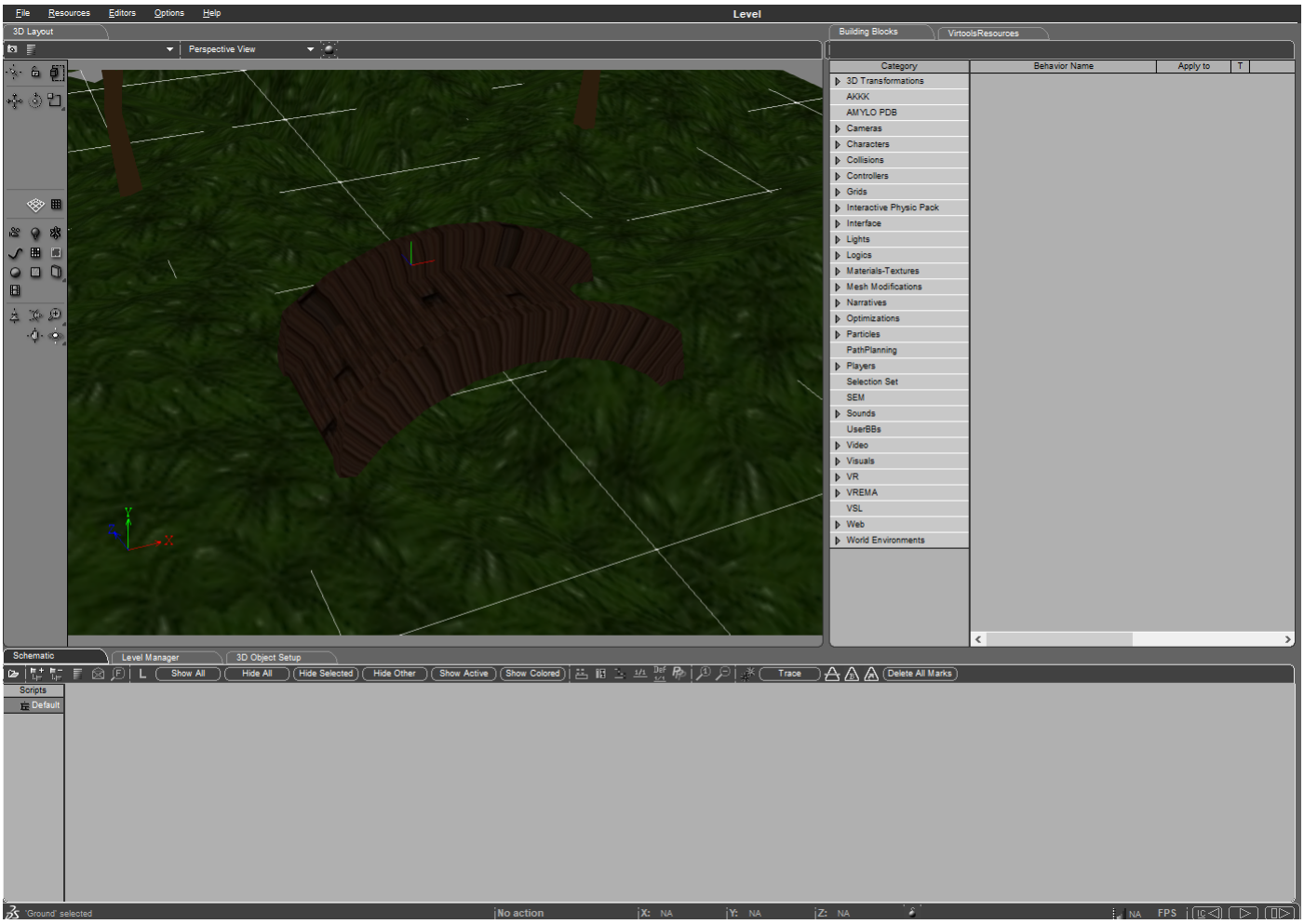


FIGURE 5.1 – Environnement Virtools

5.2.2 Protégé

Protégé est un logiciel libre qui aide à la création d'ontologies ([Noy et al., 2003]). Il a été développé par le "Centre de recherche en informatique biomédicale" à l'université de Stanford depuis 1999. Ce logiciel propose une interface graphique (figure 5.3) qui inclut de nombreux indicateurs sur l'ontologie comme des données sur les métriques (nombres d'axiomes, de classes, d'individus etc ...).

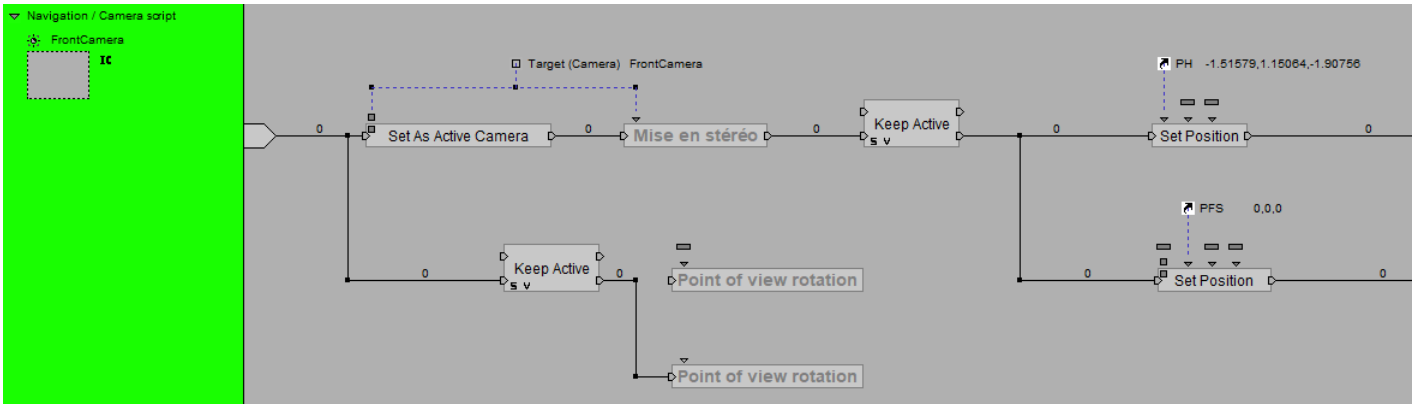


FIGURE 5.2 – Script Virtools

À partir de cette interface, il est également possible d'importer des ontologies. Cela permet d'utiliser des concepts, des classes ou des relations définies dans une autre ontologie. De plus, une Internationalized Resource Identifier (IRI) est générée. Celle-ci identifie une ontologie de manière unique permettant ainsi son partage.

De nombreux plugins sont installés nativement dans *Protégé* et ils permettent un grand nombre de fonctionnalités. Entre autres, ils permettent de gérer l'édition de concepts, de propriétés d'objets ou de données, d'acquérir des informations particulières via des requêtes Sparql Protocole and RDF Query Language (SPARQL) etc ... Cependant, d'autres plugins ont été développés par la communauté d'utilisateur pour enrichir le logiciel. Parmi eux, on retrouve la possibilité de créer, éditer, supprimer des règles SWRL, d'utiliser des mécanismes de visualisation alternatifs ou encore d'utiliser des moteurs d'inférences et des résolveurs de problèmes.

Protégé offre également la possibilité d'utiliser différents raisonneurs comme Pellet, HermiT, FaCT++ etc ... Un raisonneur permet l'inférence d'une ontologie pour vérifier la cohérence du modèle ou pour déduire de nouvelles connaissances à partir de celles définies dans le modèle. Dans nos travaux, nous avons choisi d'utiliser le raisonneur Pellet car il permet de vérifier la consistance et la cohérence d'une ontologie, de classer les concepts des taxonomies ou encore de raisonner sur les données OWL.

5.3 Description des cas d'utilisation utilisés

Dans cette partie, nous allons présenter les différents cas d'utilisation utilisés pour la validation de nos approches. Ceux-ci représentent de tâches dans des environnements 3D avec des contraintes géométriques fortes où nous comparons différentes stratégies de planification de trajectoires.

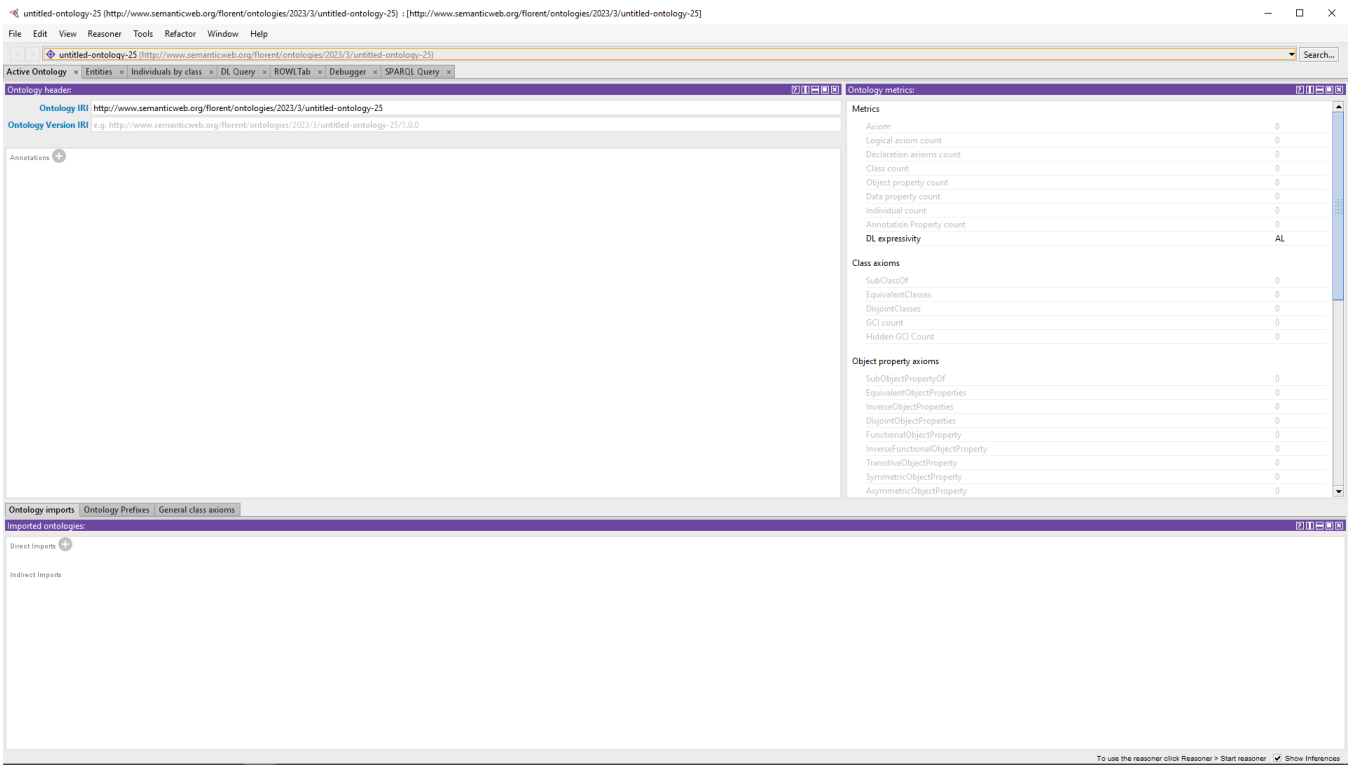


FIGURE 5.3 – Environnement de Protégé

Nos cas d'utilisation ont des complexités croissantes, montrant la généricité et l'efficacité de notre approche dans des situations variées. L'ensemble de ces cas d'utilisations ont a) des environnements avec de fortes contraintes géométriques dans lesquelles l'utilisation d'un algorithme probabiliste tel que le Bi-RRT est nécessaire, b) des objets manipulés qui ont des formes standards pouvant être assimilées à des objets industriels, c) des tâches qui s'assimilent à des tâches industrielles courantes.

Les trois premiers cas d'utilisation présentés servent à valider l'amélioration du contrôle sémantique de la planification de trajectoires. En effet, ces cas d'utilisation permettent la simulation d'une seule tâche primitive, facilitant l'étude de la planification de trajectoires. Le dernier cas d'utilisation présenté permet quant à lui de valider l'ensemble de nos contributions. En effet, la tâche à réaliser est plus complexe, nécessitant un plan de tâches complet. L'ensemble de ces cas d'utilisation permet donc de valider la totalité de notre approche pour le couplage sémantique de la planification de tâches et de trajectoires basée ontologies.

5.3.1 Premier cas d'utilisation : Insertion d'un stylo dans une boîte

Le premier cas d'utilisation correspond à l'insertion d'un stylo dans une boîte (figure 5.4). L'objectif ici est d'insérer le stylo (*Stylo1*), dans la boîte (*boîte1*) mine vers le bas. Bien que le cas

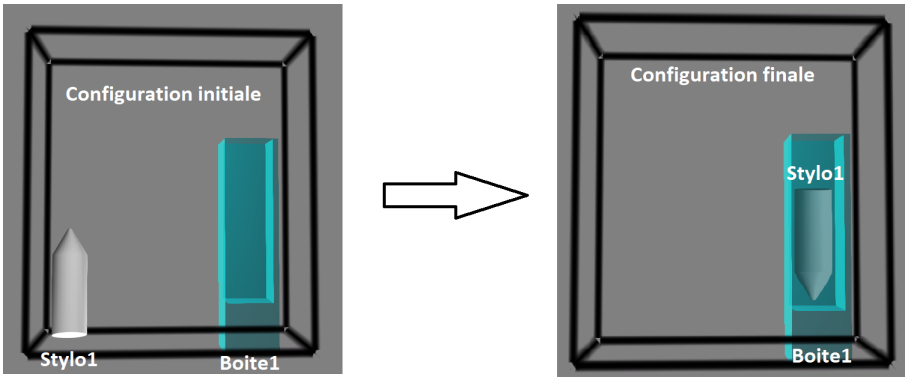


FIGURE 5.4 – Cas d'utilisation n°1 : insertion d'un stylo

d'utilisation soit relativement simple d'apparence, de fortes contraintes géométriques sont imposées par celui-ci :

— La marge entre le stylo et la boîte est faible, et la planification de trajectoires doit être effectuée sous de très fortes contraintes géométriques ;

— Des contraintes spécifiques sont associées à la tâche (insertion mine vers le bas) et permettent d'évaluer les bénéfices de l'utilisation d'informations relatives à la tâche.

5.3.2 Deuxième cas d'utilisation : Jeu d'insertion de formes

Le deuxième cas d'utilisation représente un jeu d'insertion de formes. L'environnement de ce cas d'utilisation est plus complexe que celui présenté dans le paragraphe précédent. Il est composé de quatre objets manipulables (un cube, un cylindre, un prisme triangulaire et un prisme pentagonal) ainsi que d'un coffre dans lequel se trouvent quatre trous dans lesquels les objets doivent être insérés (figure 5.5).

Plusieurs scénarios peuvent être imaginés sur ce cas d'utilisation. Par exemple, il est possible de vouloir insérer chaque pièce dans le trou de la bonne forme. Il est aussi possible de simplement vouloir insérer l'ensemble des pièces, sans considération pour les formes des trous d'insertion.

Ce cas d'utilisation a donc plusieurs avantages. Il permet de manipuler différentes pièces de forme standard, représentatifs d'objets manufacturés ou de pièces à assembler tout en réalisant des tâches standards (de type insertion) ce qui permet d'avoir une évaluation plus globale pour la planification de trajectoires. Enfin, plusieurs scénarios sont imaginables, rendant ce cas d'utilisation plus intéressant.

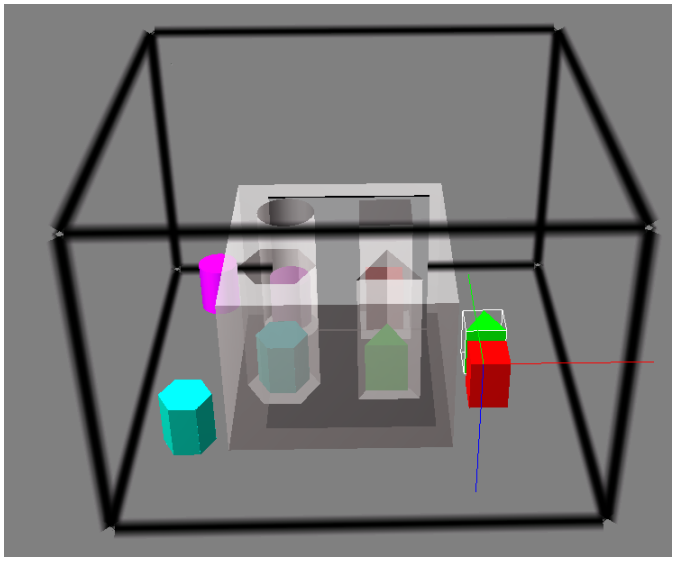


FIGURE 5.5 – Cas d'utilisation n°2 : jeu d'insertion de formes

5.3.3 Troisième cas d'utilisation : Insertion d'une pompe

Le troisième cas d'utilisation correspond à l'insertion d'une pompe dans un module d'électronique de puissance. Il s'agit d'un cas d'utilisation industriel. La scène est composée de nombreux éléments dont les principaux sont la pompe, le coffre, deux contacteurs ou encore des câbles (figure 5.6). Tous ces éléments rendent la scène fortement contrainte géométriquement. La tâche à effectuer consiste à insérer la pompe derrière les deux contacteurs présents dans le coffre de traction. Le principal intérêt de ce cas d'utilisation réside dans sa très forte complexité géométrique.

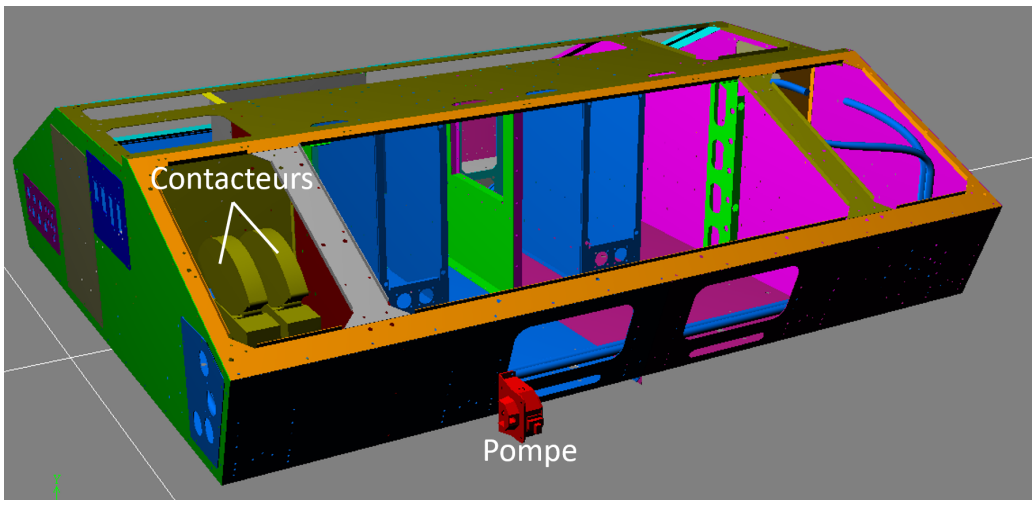


FIGURE 5.6 – Cas d'utilisaton n°3 : insertion d'une pompe dans un coffre de traction

5.3.4 Quatrième cas d'utilisation : Changement de piles

Le dernier cas d'utilisation simule un changement de piles. Dans celui-ci, nous considérons un petit appareil électronique qui est alimenté par deux piles identiques. Celles-ci sont positionnées côte à côte, mais dans des directions opposées, dans un compartiment possédant deux emplacements e^1 et e^2 (figure 5.7). Les piles usagées pu^3 et pu^4 doivent être remplacées par des piles neuves pn^1 et pn^2. L'espace de manipulation est fortement contraint et donc, les tâches de retraits et d'insertions des piles sont effectuées sous de fortes contraintes géométriques. De plus, pour que l'appareil puisse fonctionner correctement, les piles doivent être insérées dans un sens spécifique, afin de respecter la polarité.

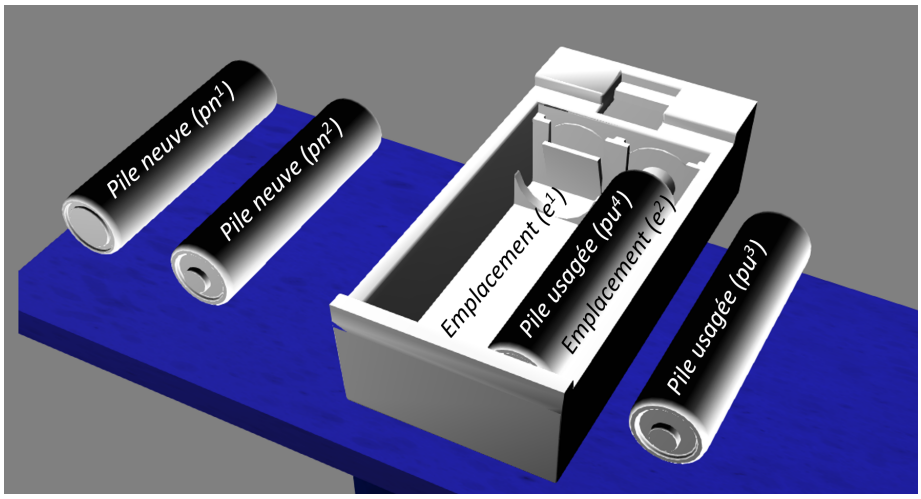


FIGURE 5.7 – Cas d'utilisation n°4 : Changement de piles

Intuitivement, plusieurs hypothèses peuvent être émises sur ce cas d'utilisation :

— Il est plus compliqué de retirer une pile quand il y en a deux dans le compartiment, que lorsqu'une seule pile est présente ;

— De même, il est plus compliqué d'insérer une pile lorsqu'une est déjà présente dans le compartiment.

Ces hypothèses seront validées par notre approche dans le paragraphe 5.5.4.

Plusieurs avantages à un tel cas d'utilisation peuvent être identifiés. Tout d'abord, il s'agit d'une manipulation que nous pouvons effectuer au quotidien. Nous avons tous une expérience du changement de pile et une manière de le réaliser. De plus, les contraintes imposées par ce cas d'utilisation (que ce soit le sens d'insertion ou le fait que l'environnement soit contraint géométriquement) le rendent pertinent pour la validation de notre approche pour coupler sémantiquement la planification de tâches et de trajectoires. Enfin, plusieurs plans de tâches peuvent permettre de réaliser cette tâche, en fonction de l'ordre de retrait et d'insertions des piles. Ceci ouvre la possibilité d'évaluer ces différents plans de tâches alternatifs.

5.3.5 Synthèse

Nous avons présenté les quatre cas d'utilisation qui ont été conçus et utilisés pour la validation de notre approche. Ces quatre cas d'utilisation sont de nature variées ce qui les rend intéressants pour montrer la généricité, la validité et l'évolutivité de notre approche de couplage sémantique TAMP et de nos ontologies.
Les deux premiers sont pertinentes car : a) ils se déroulent dans des environnements avec de fortes contraintes géométriques où l'utilisation d'un algorithme probabiliste est nécessaire, b) les objets qui sont manipulés ont des formes standards pouvant être assimilées à des objets industriels, c) déplacer les objets manipulés d'une configuration initiale à une configuration finale signifie les insérer dans des trous pertinents ou les déplacer dans des passages étroits, ce qui s'assimile à des tâches industrielles courantes, et d) ils ne sont pas spécifiques à une tâche industrielle particulière et sont suffisamment génériques pour valider objectivement les stratégies de planification du mouvement proposées.
Les cas d'utilisation 3 et 4 correspondent à des tâches industrielles concrètes. De plus, ceux-ci reprennent les intérêts a), b) et c). Enfin, le cas d'utilisation 4 permet la prise en compte de plan de tâches et, par conséquent, leur évaluation.

5.4 Résultats : Amélioration du contrôle sémantique sur la planification de trajectoires pour une tâche primitive

Dans l'optique d'offrir une collaboration efficace entre planificateur de trajectoires et planificateur de tâches, il est important, dans un premier temps, d'améliorer indépendamment la planification de trajectoires. Pour cela, nous nous sommes focalisés sur l'amélioration du contrôle sémantique de celle-ci.
Pour valider les stratégies proposées dans ces travaux, nous les avons comparées à différentes stratégies existantes. Ainsi nous comparons les stratégies *G*, *GT* et *GTO* (présentées respectivement dans les paragraphes 2.3.1.2.2 et 2.3.3.1.2 du chapitre 2.3) à notre stratégie *MGTO* décrite dans le paragraphe 4.3.4 du chapitre 4. A ces stratégies, nous appliquons également une réduction de l'espace de recherche lorsqu'un algorithme probabiliste (Bi-RRT ici) est appelé. Ces stratégies sont respectivement les stratégies *G**, *G*T*, *G*TO* et *MG*TO*.

Afin de pouvoir comparer au mieux nos différentes stratégies de planification de trajectoires, il est important de définir au préalable quels sont les critères que nous évaluons. Ceux-ci sont sont :

— le nombre de configurations aléatoires tirées. Il s'agit d'une variable commune à l'ensemble des stratégies considérées, il est donc pertinent d'utiliser cette métrique pour les

comparer.

— le temps de calcul nécessaire pour trouver une trajectoire sans collision réalisable. En effet, il est important d'abaisser au maximum les temps de calcul des planificateurs de trajectoires. Ceci afin de pouvoir faire de la collaboration en temps réel, mais également pour pouvoir effectuer un grand nombre de tests dans des temps raisonnables.

— la longueur de la trajectoire trouvée. Cela peut être un critère intéressant pour juger de la pertinence d'une trajectoire. En effet, en robotique mobile par exemple, plus court est un chemin et plus il peut être rapide à exécuter. Cela permet aussi d'économiser de l'énergie et ainsi augmenter la durée de vie du robot [Omar et al., 2015].

De plus, comme nos stratégies utilisent un algorithme probabiliste (le Bi-RRT), chaque stratégie a été exécutée cent fois, qui est un nombre suffisamment important pour lisser le côté aléatoire de l'algorithme utilisé, pour chacun des cas d'utilisation. Les résultats présentés tout au long de ce chapitre correspondent à la valeur moyenne des cent exécutions.

5.4.1 Résultats expérimentaux des stratégies de planification de trajectoires

L'ensemble de nos stratégies ont été comparées sur les trois premiers cas d'utilisation. Ceux-ci permettent de ne considérer que la partie planification de trajectoires, car elles ne mettent pas en œuvre des cas d'utilisation nécessitant une planification de tâches préalable. Elles correspondent à la simulation d'une tâche primitive simple (potentiellement d'un plan de tâches plus global) qui sera également présentée pour les besoins de certaines stratégies.

5.4.1.1 Premier cas d'utilisation : insertion d'un stylo dans une boîte

I. Étapes préalables

Afin de pouvoir appliquer l'ensemble des stratégies proposées à ce cas d'utilisation, certaines étapes doivent être réalisées en amont. Ces étapes sont 1) l'instanciation d'ENVOn-2 et, 2) la définition de la tâche primitive à réaliser, des contraintes spatiales associées et l'instanciation de ces informations dans l'ontologie TAMPO. Cela permettra également de vérifier et valider les réponses retournées à une partie des questions de compétences définies dans les paragraphes 3.2.2.1 et 3.3.1.3 du chapitre 3.

La première étape préalable est donc d'instancier l'environnement dans lequel va se dérouler la simulation dans notre ontologie ENVOn-2 et de vérifier si elle peut répondre à certaines questions de compétences. Ce cas d'utilisation est décrit dans le paragraphe 5.3.1. L'environnement est composé de deux artéfacts (*Artifact*), qui sont un stylo (*Pen*) et une boîte (*Penbox*). Chacun d'entre eux a une instance, respectivement *Pen1* et *Penbox1*. Ces différentes instances représentent les objets qui sont présents dans notre scène et chacune peut être caractérisée par

différentes propriétés comme son origine, son axe central, sa direction et son orientation (représentées par un vecteur), sa couleur, sa forme, etc. Des informations relatives aux lieux (*Places*) sont également définies comme sa direction d'ouverture s'il s'agit d'un lieu contenu dans un objet (ici le cas pour le lieu *P2*), sa forme globale, etc.

L'instanciation d'ENVOn-2 est présentée par la figure 5.8. Dans cet exemple, sont représentés en bleu clair les concepts du niveau Méta, en blanc les concepts du niveau Instance et en bleu foncé, jaune et orange les concepts du niveau Domaine. Ces trois dernières couleurs représentent respectivement les informations topologiques, sémantiques et géométriques.

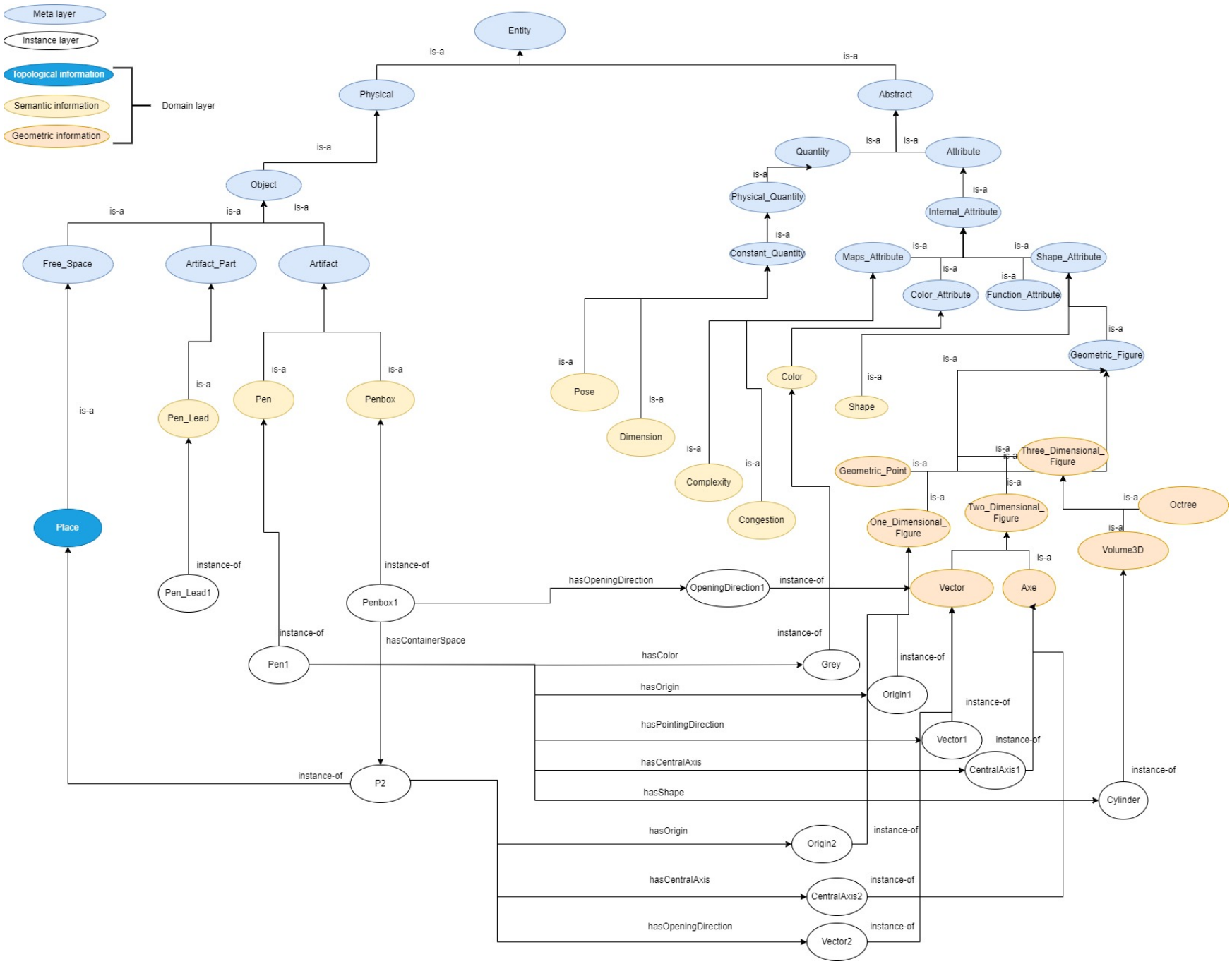


FIGURE 5.8 – Instanciation de l'environnement du cas d'utilisation n n°1 dans ENVOn-2

Après avoir instancié l'environnement dans l'ontologie, il faut préciser les questions de compétences auxquelles notre cas d'utilisation peut répondre. Celles-ci sont relatives aux informations de l'environnement :

— Quel est l'origine de *Pen1*?

— Quelle est la direction d'ouverture du lieu *P2*?

— Quel est le sens de *Pen1*?

Pour répondre à ces questions, nous utilisons des requêtes SPARQL comme le montre la figure 5.9.
Une requête SPARQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données RDF, qui sont des données de base dans la définition d'une ontologie.
Dans cet exemple, nous voulons savoir la direction dans laquelle pointe le stylo, quelles sont son origine et sa couleur...

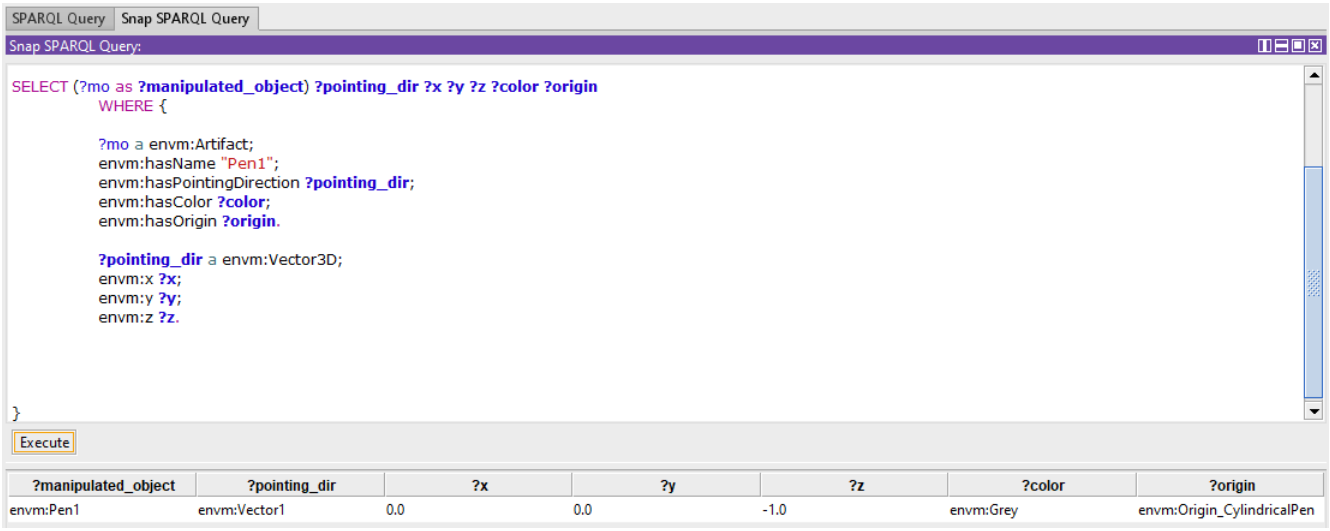


FIGURE 5.9 – Questions de compétences sur l'environnement

La deuxième étape préalable est maintenant d'instancier les différentes classes de notre ontologie TAMPO. Il faut donc commencer par définir l'action primitive à réaliser. Dans notre cas, il s'agit d'insérer le stylo dans la boîte (*Insert Pen1 into Penbox1*). De cette action, nous pouvons définir une contrainte spatiale correspondante qui est $Pen1\ Point\_To\ Penbox1$.
Ces différentes informations peuvent être instanciées dans l'ontologie TAMPO, comme représenté sur la figure 5.11. Ainsi, sur cette figure, nous avons représenté la tâche primitive "insérer" (*Insert*), les différents artéfacts (*Artifact*) qui entrent en jeu (*Pen* et *Penbox*), la contrainte spatiale (*Spatial_Constraint*) associée à la tâche primitive ainsi que la spécification de l'action primitive (*PAS*) qui contient l'identité de l'action primitive, les informations sur l'objet à manipuler, les contraintes spatiales définies pour cette action primitive et une référence (qui est un lieu (*Place*) ou un artéfact (*Artifact*)) sur laquelle s'applique les contraintes spatiales.

À l'aide d'un raisonneur (*Pellet*) et de règles SWRL, une contrainte géométrique est inférée à partir de la contrainte spatiale. Celle-ci est *Vector1 Against Vector2* .
Les règles SWRL permettent d'intégrer des règles d'inférences dans les ontologies. Une règle s'écrit sous la forme $condition(s) -> conséquence(s)$ comprenant donc une ou plusieurs conditions et une ou plusieurs conséquences.
Nous pouvons à l'aide d'une requête SPARQL demander quelles sont les contraintes géométriques qui ont été inférées à partir des contraintes spatiales (figure 5.10) dans ce cas d'utilisation.
Dans notre cas, *Vector1* est la direction dans laquelle pointe *Pen1* et *Vector2* est la direction d'ouverture de *P2*.

Snap SPARQL Query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX tampo: <http://www.semanticweb.org/florent/ontologies/2022/9/TPO-20#>

SELECT ?SpatialConstraint ?GeometricConstraint
WHERE {

?PAS a tampo:Primitive_Action_Specification;
            tampo:hasSpatialConstraint ?sc.
?SpaConstraint1 a tampo:Spatial_Constraint;
            tampo:hasSpatialRelation ?SpatialConstraint.


?PPQD a tampo:Path_Planning_Query_Description;
            tampo:hasGeometricConstraint ?gc.
?GeoConstraint1 a tampo:Geometric_Constraint;
            tampo:hasGeometricRelation ?GeometricConstraint.

}
```

Execute

?SpatialConstraint	?GeometricConstraint
tampo:PointTo	tampo:Against

FIGURE 5.10 – Questions de compétences sur les contraintes spatiales et géométriques

Nos deux ontologies sont instanciées, nous pouvons désormais simuler nos différentes stratégies afin de les comparer.

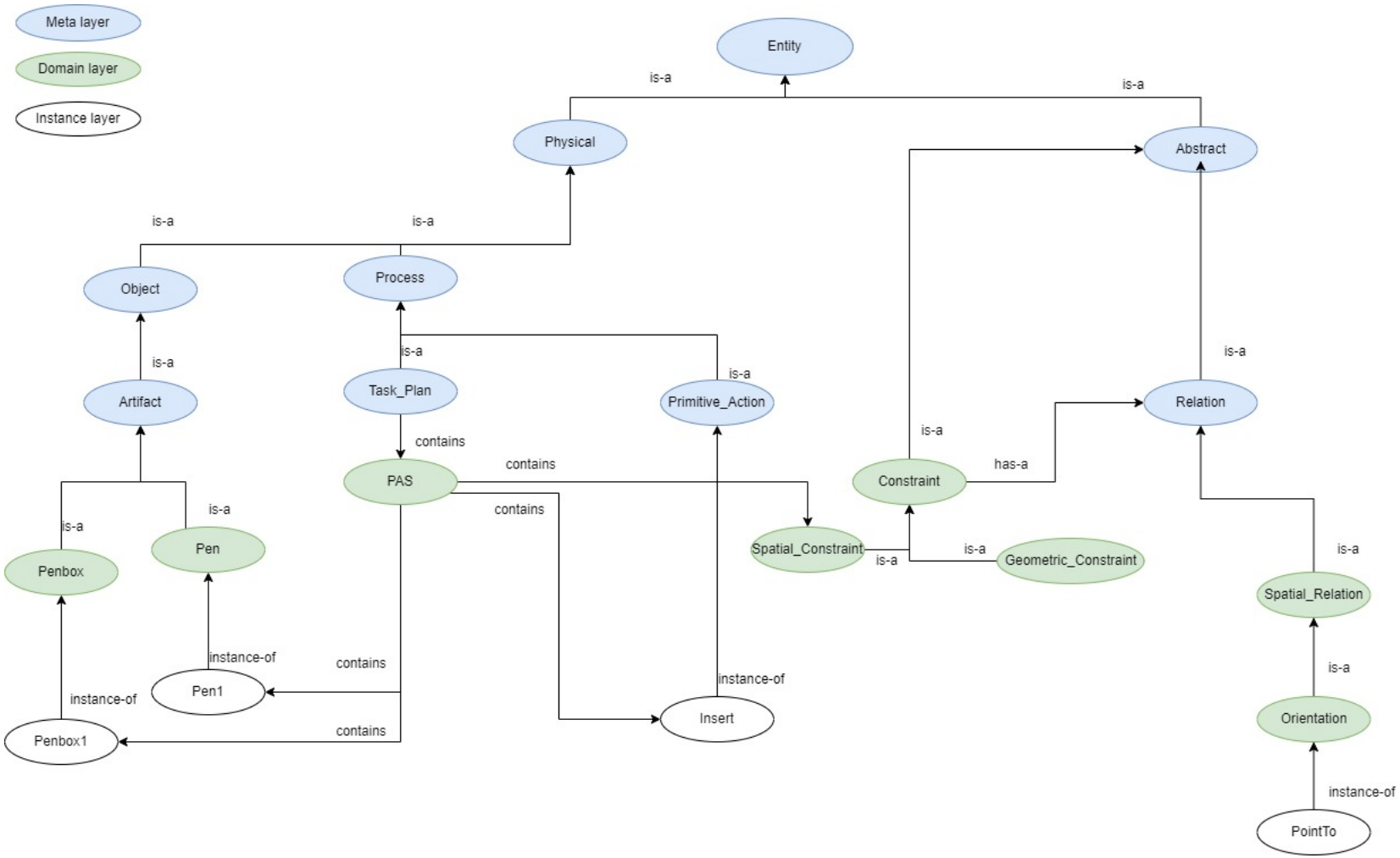


FIGURE 5.11 – Instanciation des informations relatives au TAMP pour le cas d'utilisation n°1 dans TAMPO

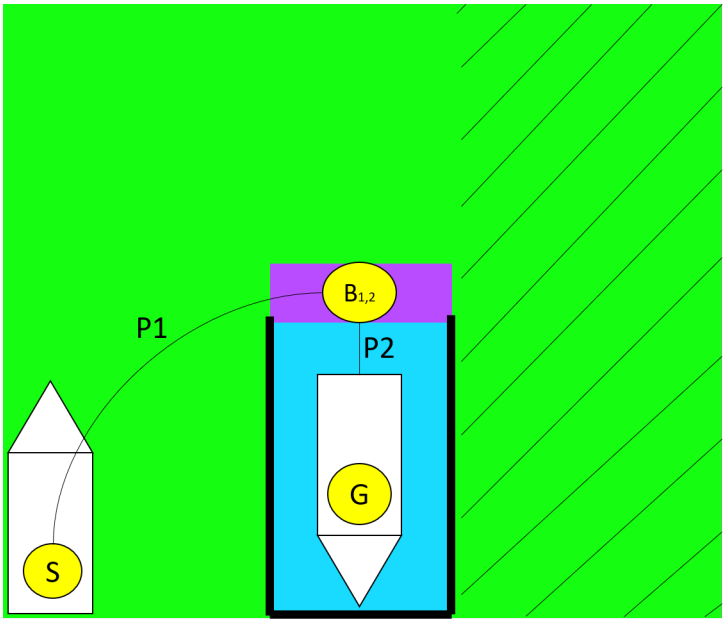


FIGURE 5.12 – Représentation du graphe topologique du premier cas d'utilisation

II. Déploiement des stratégies de planification de trajectoires

Les étapes préalables étant faites, nous pouvons nous intéresser pleinement aux performances de nos différentes stratégies de planification de trajectoires. Pour rappel, les différentes stratégies mises en œuvre sont les stratégies *G*, *GT*, *GTO* et *MGTO* (définies dans le chapitre 2.3) ainsi que leurs variantes *G**, *G*T*, *G*TO* et *MG*TO* (définies dans le paragraphe 4.3.4, du chapitre 4).

— Commençons par la stratégie *G*, qui est la stratégie ne considérant que les données purement géométriques et qui peut être considérée comme la stratégie de référence. Il faut en moyenne 83 secondes et 6 241 échantillons tirés pour obtenir une trajectoire réalisable.

— La figure 5.13 montre les résultats obtenus pour les stratégies *GT*, *GTO* et *MGTO*, comparés à la stratégie G. Nous pouvons dire que l'utilisation de données topologiques (stratégie *GT*), même seules, permet de réduire fortement le nombre de tirages aléatoires nécessaire et ainsi diminuer les temps de calcul. Cela peut s'expliquer par le fait que l'on cherche désormais à résoudre deux requêtes de planification de trajectoires différentes (car il y a deux étapes topologiques pour ce cas d'utilisation (figure 5.12)).
Par ailleurs, si d'autres informations sont considérées (stratégie *GTO*), notamment des informations liées à la tâche à réaliser, les performances du planificateur de trajectoires sont encore améliorées. En effet, grâce à l'application de contraintes géométriques au niveau tâche (cela est décrit par la figure 2.9 dans le paragraphe 2.3.3.1.2, du chapitre 2.3) au niveau de *P2* et de *B*, l'orientation du stylo est contrainte, ce qui rend son insertion plus simple. Cela permet de diminuer les temps de calcul de près de 50 fois en réduisant de 25

fois le nombre d'échantillons tirés nécessaires.

Enfin, la stratégie *MGTO* permet de ne pas appliquer un algorithme probabiliste en toute circonstance. Dans notre cas d'utilisation, approcher le stylo de la boîte (ce qui correspond à la première étape topologique) se déroule dans un environnement sans contraintes. Il n'est alors pas nécessaire d'appeler un algorithme coûteux (comme l'algorithme Bi-RRT). Avec cette stratégie, une simple interpolation linéaire entre la configuration initiale et la configuration finale est ici possible, ce qui permet de fortement réduire le nombre de tirages nécessaires et les temps de calcul. Ainsi, dans ce cas d'utilisation - qui comporte deux étapes topologiques -, les performances sont abaissées d'un facteur 2 par rapport à la stratégie *GTO*.

— La figure 5.14 montre les résultats obtenus pour les stratégies mettant en œuvre une réduction de l'espace de recherche lorsqu'un algorithme probabiliste est utilisé. La stratégie *G** permet un gain de l'ordre d'un facteur 3 pour le nombre d'échantillons et pour le temps de calcul, par rapport à la stratégie *G*. De plus, et respectivement par rapport aux stratégies *GT*, *GTO* et *MGTO*, les stratégies *G*T*, *G*TO* et *MG*TO* permettent de réduire de 50% le nombre d'échantillons aléatoires tout en diminuant de 10% les temps de calcul associés.

— Enfin, la dernière performance étudiée est la pertinence de la trajectoire proposée. Pour cela, nous étudions la longueur des chemins proposés (figure 5.15). La stratégie *G* autorise des tirages sur l'ensemble de l'environnement. Ainsi, le chemin proposé n'est pas pertinent puisque des configurations peuvent être tirées dans des zones inappropriées (représentées par la zone hachurée de la figure 5.12) et cela se reflète dans la longueur du chemin obtenu (31,39 unités). En réduisant l'espace de recherche de l'algorithme Bi-RRT, certaines zones sont désormais interdites et cela améliore la pertinence du chemin obtenu. Les longueurs des trajectoires trouvées sont approximativement réduites de 10% (tableau 5.1) par rapport aux stratégies originales sans réduction de l'espace de recherche (tableau 5.1). Enfin, l'utilisation d'une stratégie multi-planificateurs améliore la pertinence du chemin. Dans ce cas d'utilisation, la première étape topologique, correspondant à l'approche de la boîte, ne nécessite pas l'utilisation d'un algorithme probabiliste. Une interpolation linéaire est suffisante, et beaucoup moins coûteuse. En revanche, la tâche d'insertion, correspondant à la deuxième étape topologique, est beaucoup plus contrainte géométriquement, en raison de la faible marge entre le stylo et la boîte. L'utilisation d'un algorithme probabiliste est ici nécessaire. L'utilisation de différents planificateurs au cours de la même requête de planification améliore donc le chemin. Il y a un gain de 60% de la longueur du chemin entre la stratégie *MGTO* et la stratégie *GTO*. Le même gain est obtenu entre les stratégies *MG*TO* et *G*TO*.

En conclusion, notre stratégie *MG*TO* améliore grandement les performances du planificateur de trajectoires par rapport au planificateur géométrique (stratégie *G*). En effet, le nombre

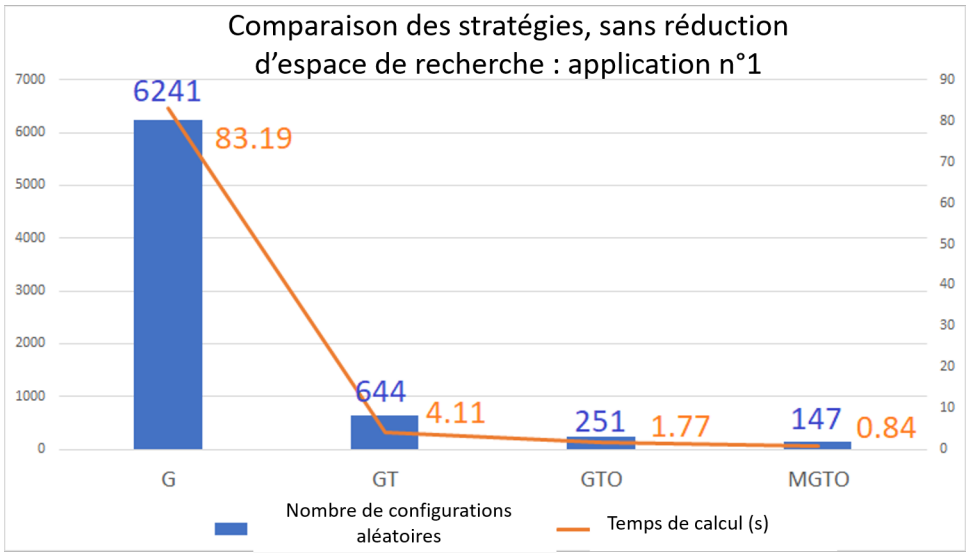


FIGURE 5.13 – Résultats sur le cas d'utilisation n°1 sans réduction de l'espace de recherche

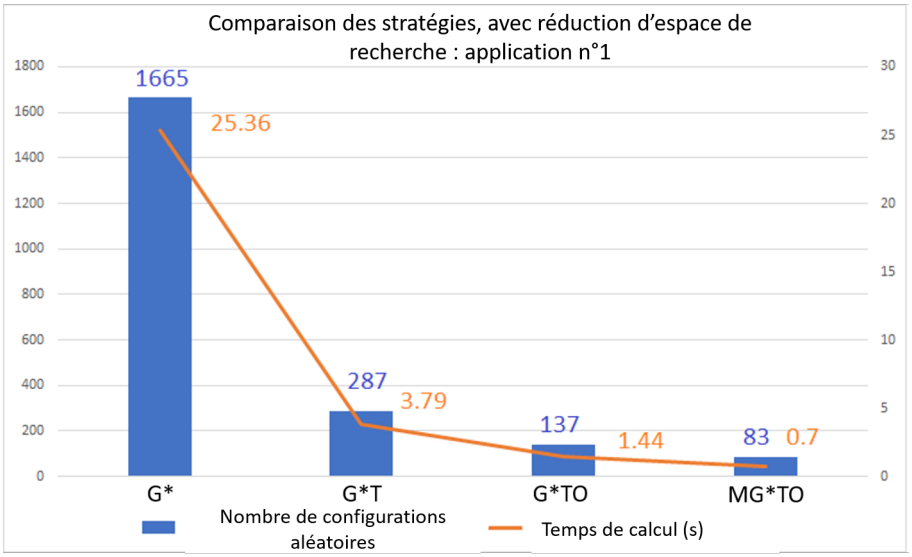


FIGURE 5.14 – Résultats sur le cas d'utilisation n°1 avec réduction de l'espace de recherche

d'échantillons tirés lors de l'utilisation de la stratégie *MG*TO* est divisé par 75 par rapport à la stratégie *G*. Le temps de calcul est quant à lui divisé en moyenne par 120. De plus, la stratégie *MG*TO* améliore la pertinence de la trajectoire proposée en réduisant la longueur du chemin par un facteur de 2,2.

Stratégies	Nombre de tirages aléatoires	Temps de calcul (s)	Longueur de la trajectoire (u)
G	6241	83.19	31.39
G*	1665	25.36	27.84
GT	644	4.11	26.28
G*T	287	3.79	24.28
GTO	251	1.77	24.22
G*TO	137	1.44	22.85
MGTO	147	0.83	15.20
MG*TO	83	0.7	14.27

TABLE 5.1 – Synthèse des résultats de planification de trajectoires pour le premier cas d'utilisation

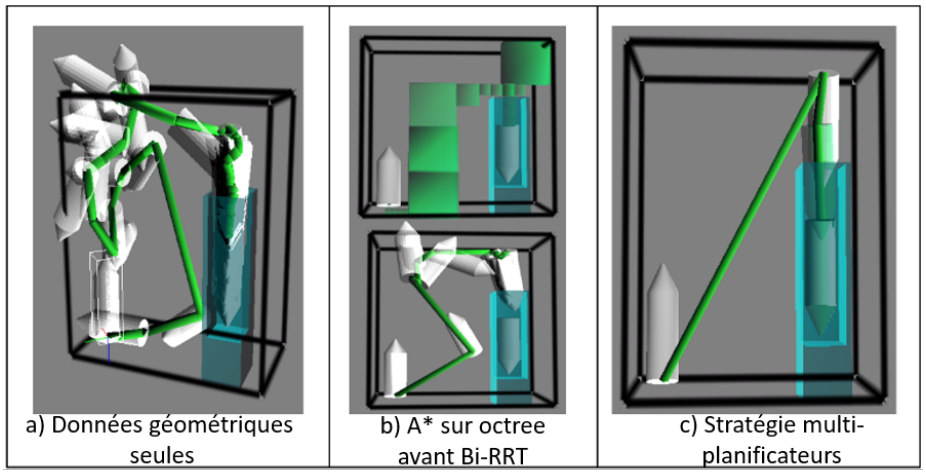


FIGURE 5.15 – Trajectoires obtenues pour le premier cas d'utilisation

5.4.1.2 Deuxième cas d'utilisation : Jeu d'insertion de formes

I. Étapes préalables

Comme pour le cas d'utilisation numéro 1, certaines étapes préalables sont à définir avant de regarder les performances de la planification de trajectoires. Ces étapes, pour rappel, sont : 1) l'instanciation d'ENVOn-2 et, 2) la définition des tâches primitives à réaliser, des contraintes spatiales associées et l'instanciation de l'ontologie TAMPO.

Bien que la façon d'instancier l'environnement et les tâches primitives dans les ontologies soit comparable au cas d'utilisation précédente, il y a quelques différences dûes à une complexité plus grande. Le fait d'avoir un cas d'utilisation plus complexe, qui a plusieurs objets manipulables et plusieurs zones d'insertions, permet de valider des questions de compétences que le cas d'utilisation n°1 ne permet pas.

L'environnement de ce cas d'utilisation est décrit dans le paragraphe 5.3.2. Il comprend cinq artéfacts (*Artifact*) qui sont un coffre (*Panel*) composé de quatre trous (de section carré, circulaire, pentagonal et triangulaire) ainsi que quatre objets manipulables qui sont un cube (*Cube*), un cylindre (*Cylinder*), un prisme triangulaire (*Triangular_Prism*) et un prisme pentagonal (*Pentagonal_Prism*). L'environnement est ainsi instancié dans l'ontologie ENVOn-2 comme présenté dans la figure 5.16. Les concepts en bleu clair représentent les concepts du niveau Méta en blanc les concepts du niveau Instance et en bleu foncé, jaune et orange les concepts du niveau Domaine. La figure 5.16 montre les informations relatives à l'instance *Cube1*. Toutefois, les informations des autres objets manipulables (*Pentagon1*, *Triangle1* et *Cylinder1*) ainsi que les informations liées aux différentes lieux (P1 à P5) sont instanciées de la même manière.

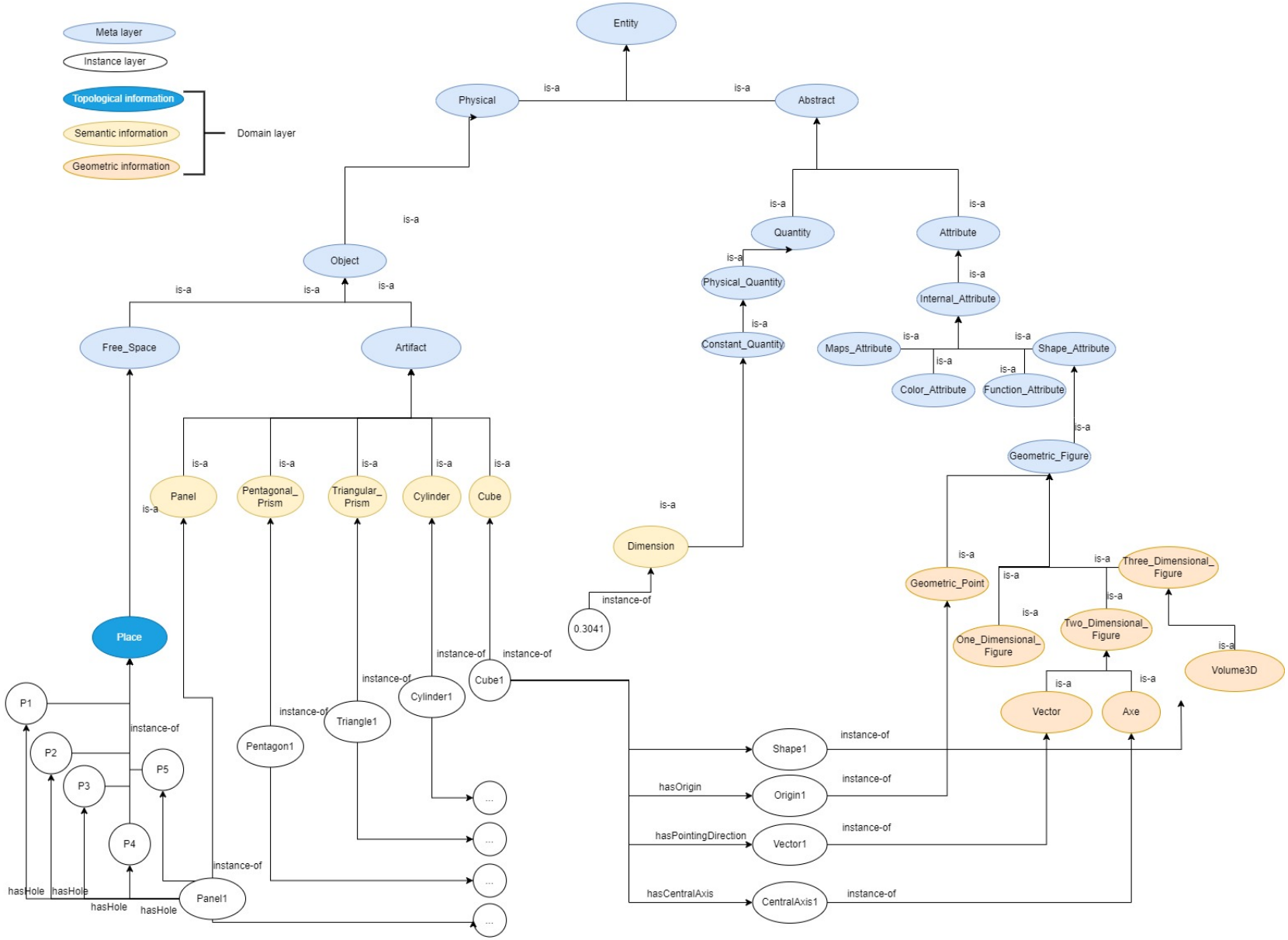


FIGURE 5.16 – Instanciation de l'environnement du cas d'utilisation n°2 dans ENVOn-2

Dans ce cas d'utilisation, nous considérons un scénario dans lequel quatre plans de tâches composés d'une seule action primitives sont considérés. Chaque action primitive correspond à l'insertion d'une forme dans le trou correspondant. Il s'agit donc d'*insérer le cube dans le trou cubique*, d'*insérer le prisme triangulaire dans le trou de section triangulaire* etc. Pour l'action primitive correspondant à l'insertion du cube par exemple, on peut assigner la contrainte spatiale de la forme : *Cube1 Algined* $Cubic\_Hole$. Un raisonneur peut ensuite inférer, à partir des informations de l'ontologie, une contrainte géométrique pour cette action primitive. La contrainte géométrique inférée est pour notre exemple : *Vector1 Against Vector2* (figure 5.17).
Toutes les informations nécessaires ont maintenant été définies pour opérer la planification de trajectoires.

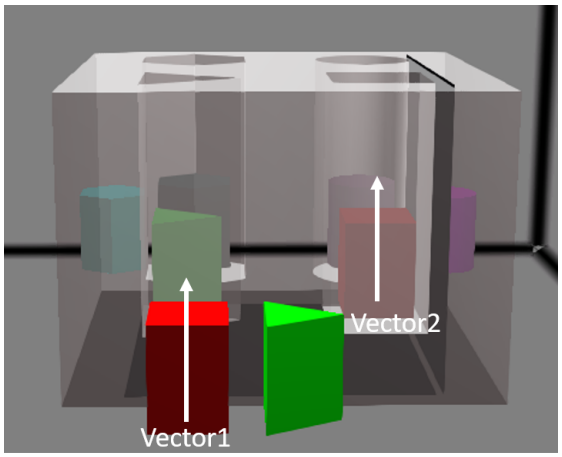


FIGURE 5.17 – Représentation des vecteurs d'une contrainte géométrique

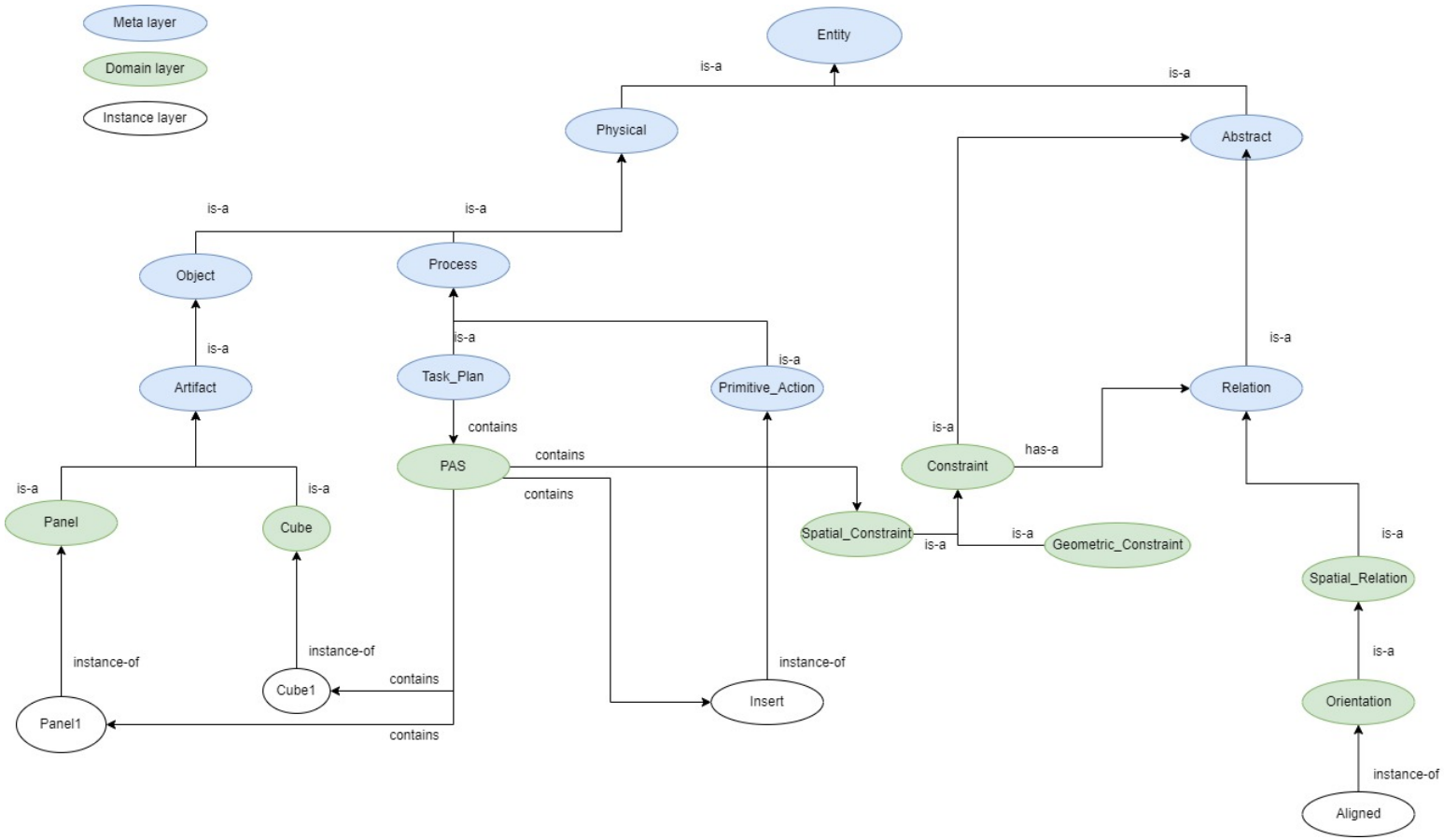


FIGURE 5.18 – Instanciation des informations relatives au TAMP pour le cas d'utilisation n°2 dans TAMPO

La figure 5.18 montre comment est instanciée l'ontologie TAMPO pour le deuxième cas d'utilisation. Dans cette représentation, seules les informations relatives à l'instance *Cube1* sont représentées, mais elles sont similaires pour les autres formes.

Ce cas d'utilisation permet de répondre à certaines des questions de compétences d'ENVOn-2 définies dans le paragraphe 3.2.2.1 du chapitre 2.3 :

— Quel est l'axe central de *Pentagon1* ?

— Quelle est la mesure de la diagonale de *Cube1* ?

— Quelle est la mesure de *P2* (correspondant au trou cylindrique) ?

Le cas d'utilisation n°1 permet déjà de répondre à ces questions de compétences (dans la partie 5.4.1.1). Nous nous focalisons alors, dans cette partie, sur des questions plus pratiques. Celles-ci peuvent par exemple être « Dans quel trou est-il possible d'insérer *Cylinder1* ? » ou encore « Quel est le trou à favoriser pour insérer *Triangle1* ? ».

La figure 5.19 présente la requête SPARQL effectuée afin d'identifier et de proposer les différentes lieux (*Places*) dans lesquelles l'objet *Cylinder1* peut être inséré. Pour cela, l'ontologie, par l'utilisation de règles SWRL, compare le rayon de l'objet *Cylinder1* à chacun des rayons des cercles inscrits des différents lieux. Puis, la requête SPARQL propose seulement les lieux (*Places*) dont le rayon du cercle inscrit est supérieur au rayon de l'objet manipulé, correspondant alors aux seules lieux (*Places*) capables de contenir l'objet *Cylinder1*.
Dans notre cas d'utilisation, les cercles inscrits des trous de section triangulaire, carré, pentagonal et circulaire sont respectivement de 0.086, 0.150, 0.1376 et 0.155 unités. Ainsi, dans le cas où l'objet manipulé à un rayon de 0.115 unité, il peut être inséré dans tous les trous sauf celui à section triangulaire (figure 5.19-a). En revanche, si son rayon est augmenté à 0.151 unité, il ne peut être inséré que dans le trou à section circulaire (figure 5.19-b).
L'ontologie, via des règles de raisonnement, est aussi capable d'indiquer quel est le trou à favoriser pour insérer une pièce. Dans notre cas d'utilisation, nous intuitons qu'il est plus simple d'insérer une pièce dans le trou ayant la forme correspondante. La figure 5.20 montre la requête SPARQL représentant notre demande ainsi que le résultat retourné. Ainsi, lorsque nous manipulons le prisme triangulaire, le résultat de la requête indique le trou d'insertion triangulaire.

II. Déploiement des stratégies de planification de trajectoires

Nous avons validé nos différentes stratégies de planification de trajectoires sur l'ensemble des formes manipulables. Dans ce scénario, nous cherchons simplement à insérer une forme dans le trou possédant la même forme. Les résultats obtenus sont présentés dans les tableaux 5.2, 5.3 et 5.4.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX envm: <http://www.semanticweb.org/florent/ontologies/2022/9/TPO-20#>

SELECT ?name_OM ?rayon_OM ?rayon_incercle ?name_hole
WHERE {

?artifact a envm:Artifact;
            envm:hasName "O2_CylindricalPrism";
            envm:hasName ?name_OM;
            envm:hasStandardForm ?SF;
            envm:hasRayon ?rayon_OM.

?reference a envm:Artifact;
            envm:hasName "Panel";
            envm:hasHole ?hole.

?hole a envm:Place;
            envm:hasName ?name_hole;
            envm:hasStandardForm ?SFP.

?SFP a envm:Volume3D;
            envm:value_insert ?rayon_incercle.

FILTER(?rayon_incercle >= ?rayon_OM)

}
```

Execute

?name_OM	?rayon_OM	?rayon_incercle	?name_hole
O2_CylindricalPrism^^xsd:string	0.115	0.15	P3_SquareHole^^xsd:string
O2_CylindricalPrism^^xsd:string	0.115	0.13760000467300415	P4_PentagonalHole^^xsd:string
O2_CylindricalPrism^^xsd:string	0.115	0.155	P2_CylindricalHole^^xsd:string

a) Objet cylindrique de rayon r=0.115

?name_OM	?rayon_OM	?rayon_incercle	?name_hole
O2_CylindricalPrism^^xsd:string	0.151	0.155	P2_CylindricalHole^^xsd:string

b) Objet cylindrique de rayon r=0.151

FIGURE 5.19 – QC : Dans quel trou est-il possible d'insérer la pièce cylindrique ?

— Le premier tableau présente les nombres de tirages aléatoires nécessaires pour résoudre une requête de planification de trajectoires ;

— Le deuxième tableau présente les temps de calcul associés ;

— Le troisième présente la longueur des trajectoires trouvées.

À partir de ces résultats, nous pouvons tirer plusieurs conclusions.
Premièrement, utiliser des données de plus haut niveau d'abstraction que simplement les données géométriques permet d'améliorer considérablement les performances (nombre de tirages nécessaires et temps de calcul associés). En effet, par rapport à la stratégie *G*, la stratégie *GTO* permet de réduire le temps de calcul d'un facteur 20 en moyenne. Ces résultats sont plus disparates pour le nombre de tirages nécessaires. Selon la forme, la stratégie *GTO* a besoin d'entre 5

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX envm: <http://www.semanticweb.org/florent/ontologies/2022/9/TPO-20#>

SELECT ?artifact ?hole
WHERE {

?artifact a envm:Artifact;
          envm:hasName "O3";
          envm:hasShape ?shape.

?reference a envm:Artifact;
          envm:hasName "Panel";
          envm:hasHole ?hole.

?hole a envm:Place;
envm:hasShape ?place_shape.

FILTER(?shape = ?place_shape)

}
```

Execute

?artifact	?hole
envm:Object_TriangularPrism	envm:P5_TriangularHole

FIGURE 5.20 – QC : Quel est le trou à favoriser pour insérer Triangle1 ?

fois moins (prisme triangulaire) et 30 fois moins (cylindre) de tirages que la stratégie *G* pour résoudre la requête de planification. Les différences d'une forme à l'autre peuvent s'expliquer par le fait que les différentes formes manipulées engendrent des contraintes différentes. Les marges entre les objets manipulés et leur trou correspondant ne sont pas exactement identiques.
Deuxièmement, réduire l'espace de recherche de l'algorithme probabiliste est une solution très intéressante (paragraphe 4.3.2, chapitre 4). Pour l'ensemble des formes, cela permet de réduire en moyenne le temps de calcul de 40%. Cela permet également de réduire la longueur des trajectoires obtenues, car ces stratégies évitent de tirer des configurations dans des zones non pertinentes.
Enfin, nous pouvons voir que l'utilisation d'une stratégie multi-planificateurs (c'est-à-dire les stratégies *MGTO* et *MG*TO* décrites dans le paragraphe 4.3.4 du chapitre 4) améliore également les performances de la planification de trajectoires. Toutefois, en comparaison avec le cas d'utilisation numéro 1, les gains sont plus faibles. Ceci s'explique par le positionnement des objets dans l'environnement 3D. En effet, dans le cas du jeu d'insertion de formes, les objets sont placés proches du coffre. Ainsi, pour la première étape topologique qui correspond à approcher l'objet du trou d'insertion, il est impossible de faire une interpolation linéaire. Nous devons donc faire appel à l'algorithme Bi-RRT qui est plus coûteux. En ce qui concerne la seconde étape topologique, la marge est, dans ce cas d'utilisation, un peu plus grande que dans le cas d'utilisation numéro 1. Ainsi, il apparait des cas dans lequel le jalon aléatoirement tiré au niveau d'une frontière (expliqué dans le paragraphe 2.3.1.2.2) est placé de sorte qu'une interpolation linéaire

Stratégies	Nombre de tirages aléatoires			
	Prisme triangulaire	Cube	Cylindre	Prisme pentagonal
G	7401	9108	22042	4962
G*	3240	2042	18818	2016
GT	1804	2791	6282	1411
G*T	555	601	1955	412
GTO	1542	590	734	582
G*TO	496	501	186	248
MGTO	403	347	375	287
MG*TO	286	307	107	137

TABLE 5.2 – Nombre de tirages aléatoires pour les différentes stratégies - deuxième cas d'utilisation

soit possible entre ce jalon et la configuration finale. Cela montre qu'il faut améliorer le contrôle sur le tirage de ce jalon.

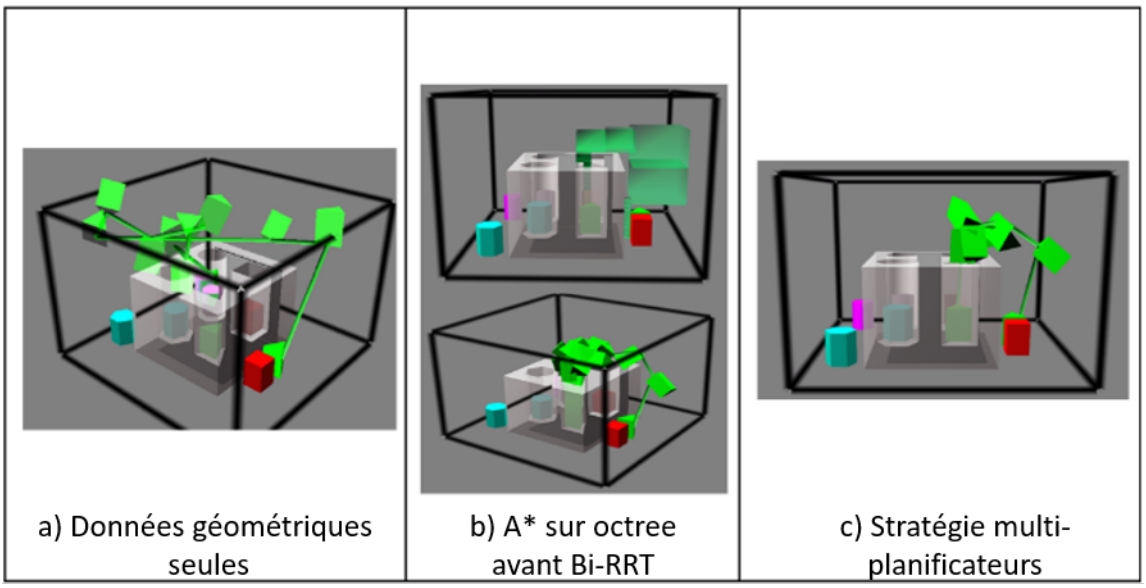


FIGURE 5.21 – Trajectoires obtenues pour le deuxième cas d'utilisation

5.4.1.3 Troisième cas d'utilisation : Insertion d'une pompe

I. Étapes préalables

De façon similaire aux deux premiers cas d'utilisations, il y a des étapes préalables à définir avant de faire la simulation de planification de trajectoires, qui correspondent à l'instanciation

Stratégies	Temps de calcul (s)			
	Prisme triangulaire	Cube	Cylindre	Prisme pentagonal
G	72.06	24.86	676.54	29.42
G*	42.91	22.08	287.72	16.5
GT	6.1	6.08	30.29	6.79
G*T	4.6	4.46	12.62	4.02
GTO	4.41	2.00	3.61	2.16
G*TO	3.58	1.92	1.72	1.64
MGTO	2.47	1.94	2.55	1.5
MG*TO	2.07	1.66	1.25	1.25

TABLE 5.3 – Temps de calcul pour les différentes stratégies - deuxième cas d'utilisation

Stratégies	Longueur de la trajectoire (u)			
	Prisme triangulaire	Cube	Cylindre	Prisme pentagonal
G	6.86	6.8	6.73	6.72
G*	6.29	6.27	6.34	6.29
GT	5.2	5.3	6.49	5.80
G*T	3.65	4.18	3.76	4.26
GTO	5.31	5.19	6.48	5.86
G*TO	3.77	4.11	3.72	4.27
MGTO	5.08	4.11	3.72	4.27
MG*TO	3.71	3.98	3.31	3.91

TABLE 5.4 – Longueur des trajectoires obtenues pour les différentes stratégies - deuxième cas d'utilisation

de nos deux ontologies, ENVOn-2 et TAMPO.
Ce cas d'utilisation est décrit dans le paragraphe 5.3.3. L'environnement de la scène est bien plus riche que les deux précédents, avec beaucoup de composant et donc, de contraintes géométriques. Les principaux artéfacts (*Artifact*) qui nous intéresse sont la pompe (*Pump*), les contacteurs (*Contactors*) et le coffre de traction (*Traction_Case*).
Dans notre cas d'utilisation, nous souhaitons insérer la pompe derrière les deux contacteurs qui se trouvent à l'intérieur du coffre, ce qui correspond à l'action primitive à effectuer (*Insert Pump into Traction_Case*).
L'instanciation des ontologies est similaire à celles des cas d'utilisations 1 et 2. Ainsi, nous n'allons pas la décrire et nous nous focalisons ici sur les résultats des différentes stratégies de planification de trajectoires.

II. Déploiement des stratégies de planification de trajectoires
Les résultats obtenus sont présentés par les figures 5.22 et 5.23.
Il s'agit d'un cas d'utilisation industriel ayant un modèle de l'environnement beaucoup plus complexe et volumineux que les deux cas d'utilisation précédents. Cela explique les résultats que nous obtenons. Ainsi, nous voyons que pour résoudre une requête de planification de trajectoires, un plus grand nombre d'échantillons tirés est nécessaire. Cela implique également un temps de calcul plus long.
À l'instar des deux premiers cas d'utilisation, nous pouvons conclure que :

— Du fait de la complexité très importante de l'environnement dans lequel se déroule la tâche, la stratégie G n'offre pas des performances satisfaisantes. En effet, il faut près d'une heure et demie pour résoudre une requête de planification de trajectoires. Ainsi, cela montre que pour des cas d'utilisation industriels, améliorer les performances des planificateurs de trajectoires est primordial.

— L'utilisation de données de haut niveau d'abstraction permet d'améliorer les performances de nos stratégies, que ce soit le nombre de tirages aléatoires, le temps de calcul ou la pertinence de la trajectoire proposée (figure 5.24).

— De plus, réduire l'espace de recherche d'un algorithme probabiliste dans un tel cas d'utilisation est important, car il existe de nombreuses zones qui ne sont pas pertinentes et qui sont coûteuses à explorer.

— Enfin, plus un modèle est grand, plus le nombre d'étapes topologiques est important et plus il est intéressant d'avoir différents algorithmes de planification pouvant être choisis en adéquation avec le contexte sémantique local du lieu à explorer.

Ce cas d'utilisation est particulièrement intéressant pour montrer la difficulté que peut avoir un planificateur de trajectoires dans un environnement très complexe et donc très contraint géométriquement. Cela montre aussi le grand intérêt de notre approche qui permet, grâce à

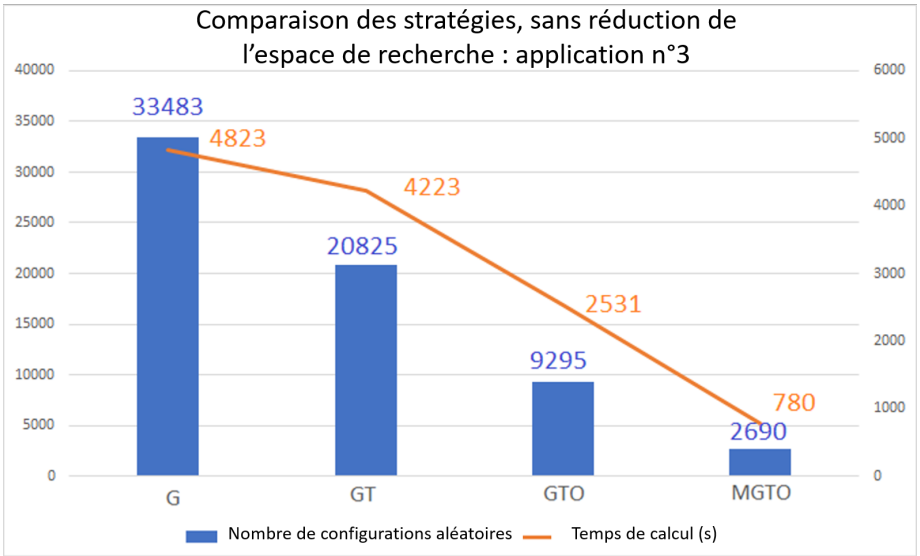


FIGURE 5.22 – Résultats sur le cas d'utilisation n°3 sans réduction de l'espace de recherche

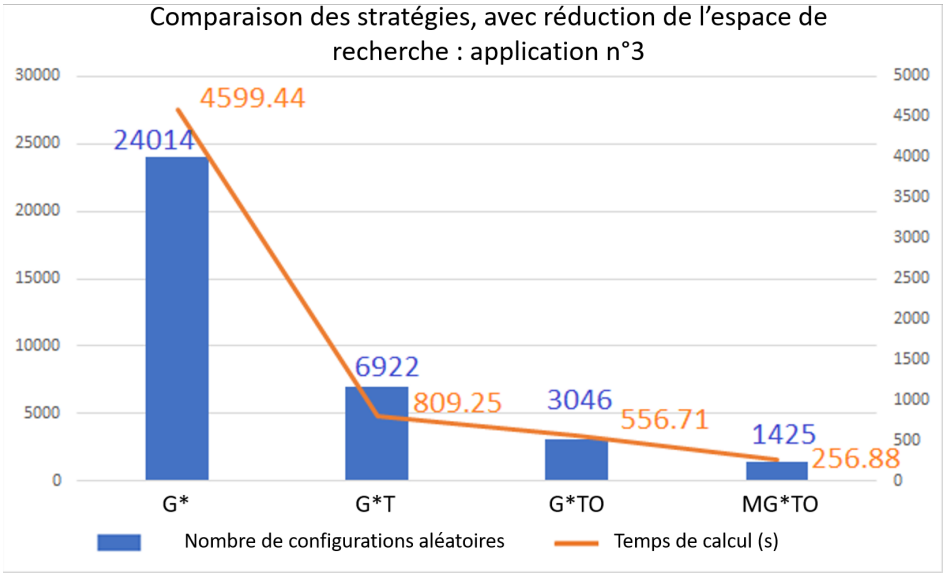


FIGURE 5.23 – Résultats sur le cas d'utilisation n°3 avec réduction de l'espace de recherche

l'utilisation d'informations de haut niveau d'abstraction et à des combinaisons d'algorithmes, d'améliorer grandement les performances de la planification de trajectoires. Le tableau 5.5 résume les résultats obtenus sur ce cas d'utilisation et montre que notre stratégie *MG*TO* permet d'obtenir une trajectoire en moins de 5 minutes, contre plus d'une heure et demie pour la stratégie *G*. La trajectoire obtenue est également deux fois plus courte avec notre stratégie *MG*TO*, en comparaison avec la stratégie *G*.

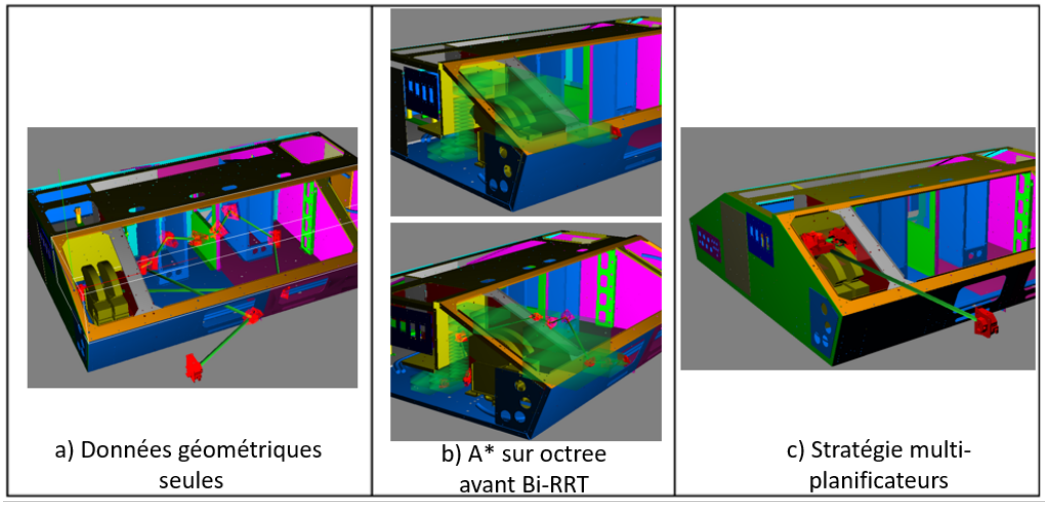


FIGURE 5.24 – Trajectoires obtenues pour le troisième cas d'utilisation

Stratégies	Nombre de tirages aléatoires	Temps de calcul (s)	Longueur de la trajectoire (u)
G	33483	4823.37	6.67
G*	24014	4599.44	5.59
GT	20825	4223.19	4.80
G*T	6922	809.25	3.52
GTO	9295	2531.87	4.24
G*TO	3046	556.71	3.20
MGTO	2690	780.47	3.91
MG*TO	1425	256.88	2.97

TABLE 5.5 – Tableau récapitulatif des résultats pour l'ensemble des stratégies déployées sur le cas d'utilisation n°3

5.4.2 Synthèse des résultats sur la planification de trajectoires

Nous avons appliqué nos stratégies de planification de trajectoires sur trois cas d'utilisation différents, de complexités croissantes, et permettant d'amener des conclusions variées et de valider plusieurs points importants.

De par l'utilisation de différents cas d'utilisation, nous avons pu instancier trois environnements dans notre ontologie ENVOn-2 et nos différentes tâches dans TAMPO. Nous avons ainsi pu valider une grande partie des questions de compétences définies dans le chapitre 3, pour nos deux ontologies. Particulièrement, nous avons pu répondre aux différentes questions de

compétences liées à ENVOn-2, pu spécifier quelles sont les contraintes spatiales et générer les contraintes géométriques pour chacun des cas d'applications.

De plus, nous avons pu valider nos différentes stratégies de planification de trajectoires. Ainsi, et de manière générale, les résultats obtenus pour l'ensemble des cas d'utilisation ont des tendances similaires. La stratégie *G*, qui ne prend en considération que les données géométriques de l'environnement, est la stratégie offrant les moins bonnes performances. Ne considérant que la géométrie de l'environnement, un algorithme probabiliste va tirer des configurations aléatoirement dans l'ensemble de l'environnement, sans considération pour la tâche à réaliser. Par conséquent, le temps de calcul d'une trajectoire sera long et la trajectoire obtenue pourrait ne pas être pertinente.

Par conséquent, utiliser des données d'un plus haut niveau d'abstraction pour la modélisation de l'environnement, typiquement des informations topologiques, est important. Cela permet d'améliorer les performances du planificateur, car la requête de planification est divisée requêtes sur les étapes successives du chemin topologique. Toutefois, ces stratégies ne considèrent que des informations sur l'environnement. Il est également nécessaire de considérer des informations relatives à la tâche à réaliser.

Dans cette optique, les stratégies *GTO*, *MGTO* ainsi que leurs variantes, *G*TO* et *MG*TO* ont été développées. Celles-ci, à l'aide d'ontologies, permettent d'inférer des contraintes géométriques sur le mouvement à partir de contraintes spatiales exprimées au niveau tâche. Ces stratégies ont permis d'abaisser fortement les temps de calcul tout en améliorant la pertinence des trajectoires obtenues. La différence entre *GTO* et *MGTO* (et également entre *G*TO* et *MG*TO*) est la possibilité pour les secondes d'utiliser différents planificateurs au sein d'une même requête. Cela permet d'éviter d'utiliser des algorithmes coûteux quand ce n'est pas nécessaire, et améliorer les performances de la planification.

Enfin, nous avons développé une approche, applicable sur toutes les stratégies permettant de réduire l'espace de recherche de notre algorithme probabiliste. Celles-ci sont les stratégies *G**, *G*T*, *G*TO*, *MG*TO*. Pour celles-ci, le tirage des configurations est restreint à des zones pertinentes, améliorant également les performances de la planification.

S'il est important d'améliorer les performances de la planification de trajectoires, il ne s'agit que d'une partie de notre approche globale. En effet, seule une tâche primitive a été considérée sur chacun de ces cas d'utilisation. Dès lors, nous ne considérons pas la planification de tâches et ne pouvons pas valider la globalité de notre approche. Il est donc nécessaire de considérer un

cas d'utilisation plus complexe, d'un point de vue tâches à réaliser, pour valider notre approche de couplage sémantique du TAMP.

5.5 Validation de notre approche pour le couplage sémantique de la planification de tâches et de trajectoires basée ontologies

Après avoir comparé et validé nos différentes stratégies de planification de trajectoires, il est temps de valider notre approche TAMP sur un cas d'utilisation nécessitant la collaboration des planificateurs de tâches et de trajectoires. Pour cela, nous avons défini un cas d'utilisation présenté dans le paragraphe 5.3.4, qui correspond au changement de deux piles dans un compartiment. Il s'agit d'un cas d'utilisation intéressant car à l'inverse des trois premiers, il nécessite la prise en compte d'un plan de tâches car il faut planifier l'ordre de retrait et d'insertion des piles. De plus, les actions d'insertion ou de retrait de piles se font sous des contraintes géométriques qui sont fortes et les piles doivent être insérées dans une direction imposée pour respecter la polarité, ce qui génère des contraintes à respecter.
Dans le paragraphe 5.5.1, il nous faut faire appel au planificateur de tâches pour trouver un plan de tâche réalisable. Ce plan de tâches ainsi que l'environnement dans lequel il doit se dérouler doivent ensuite être instanciés dans nos ontologies TAMPO et ENVOn-2. Cela est décrit dans le paragraphe 5.5.2. D'autres plans peuvent aussi être proposés par un opérateur humain. Enfin, dans le paragraphe 5.5.3, un raisonneur évalue les différents plans de tâches instanciés dans l'ontologie TAMPO, et propose le plan le plus pertinent. Cette approche est ensuite validée en simulant les différents plans de tâches dans le paragraphe 5.5.4.

5.5.1 Appel du planificateur de tâches

Notre approche globale pour le couplage sémantique de la planification de tâches et de trajectoires est décrite dans le paragraphe 4.2. Pour commencer, nous appelons le planificateur de tâches que nous avons choisi, pour rappel, le planificateur Fast-Forward qui est présenté dans le paragraphe 4.4.4. Celui-ci utilise le langage PDDL et une description PDDL se compose de deux parties. La première décrit le domaine en définissant les aspects « universels ». Il s'agit essentiellement des aspects qui ne changent pas, quel que soit le cas d'utilisation que nous essayons de résoudre. La deuxième partie décrit le problème spécifiquement.

Ces deux parties sont présentées par la figure 5.25. Dans la description du domaine (figure 5.25 a)), nous devons définir les types d'objets présents dans la scène. Ici, des piles et un compartiment. Nous devons également définir les prédicats, qui sont les faits qui nous intéressent (par exemple les propriétés des objets) et qui peuvent être vrais ou faux, ainsi que les différentes ac-

tions possibles.

Dans la description du problème (figure 5.25 b)), nous définissons les différentes instances qui sont dans la scène, et nous décrivons l'état initial et l'état final à atteindre.

```
(define (domain battery)
  (:types obj - object
          battery - obj
          compartment - obj)
(:predicates
             (have ?x)
             (in ?x ?y)
             (on ?x ?y)
             (out ?x ?y)
)


(:action insert
:parameters (?x - obj ?y - compartment)
:precondition (and (out ?x ?y))
:effect (and (in ?x ?y)
   (not (have ?x))))


(:action remove
:parameters (?x - obj ?y - compartment)
:precondition (and (in ?x ?y))
:effect (and (out ?x ?y)
   (not (have ?x))))


)
```

a) Description du domaine

```
(define (problem battery-1)
(:domain battery)
(:objects

battery_old1 battery_old2 battery_new1 battery_new2 - battery

compartment - compartment

)
(:init
(in battery_old1 compartment)
(in battery_old2 compartment)
(out battery_new1 compartment)
(out battery_new2 compartment)
)
(:goal
(and
(out battery_old1 compartment)
(in battery_new1 compartment)
(out battery_old2 compartment)
(in battery_new2 compartment)

)
)
)
```

b) Description du problème

FIGURE 5.25 – Description du domaine et du problème en langage PDDL

Nous pouvons désormais appeler notre planificateur de tâches qui nous retourne le plan suivant (qui sera appelé plan de tâche n°1) :

- 1- Remove battery_old1 compartment ;
- 2- Insert battery_new1 compartment ;
- 3- Remove battery_old2 compartment ;
- 4- Insert battery_new2 compartment.

Nous avons ainsi un premier plan de tâches pour notre cas d'utilisation. Cependant, un opérateur humain peut également proposer un autre plan de tâche à évaluer. Intuitivement, un humain ayant à effectuer cette action de changement de piles aura tendance à commencer par retirer les deux piles usagées en premier. Cela peut être pour des questions de simplicité, car il est plus facile d'enlever la seconde pile usagée quand il ne reste plus qu'elle dans le compartiment, ou pour des questions de logique, car en commençant par enlever les deux piles usagées, l'opérateur minimise les risques de mélanger les piles neuves et usagées. Ainsi, nous pouvons proposer un autre plan pour ce cas d'utilisation, qui est le suivant (et qui sera appelé plan de tâche n°2) :

— 1- Remove battery_old1 compartment ;

— 2- Remove battery_old2 compartment ;

— 3- Insert battery_new1 compartment ;

— 4- Insert battery_new2 compartment.

5.5.2 Instanciation des ontologies

Nous avons désormais deux plans de tâches possibles pour notre cas d'utilisation. Nous allons commencer par instancier nos deux plans de tâches dans l'ontologie TAMPO (figure 5.26). Dans cette figure, les concepts du niveau Méta sont représentés en bleu clair, les concepts en vert représentent ceux du niveau Domaine et les concepts du niveau Instance sont colorés en blanc.

Parmi les concepts importants liés au domaine du TAMP (présentés dans le paragraphe 3.3 du chapitre 4), nous retrouvons les différents artéfacts (*Artifacts*) entrant en jeu dans la planification de tâches, la notion de plan de tâches (*Task_Plan*) et d'actions primitives (*Primitive_Action*). Nous retrouvons également les contraintes (*Constraints*), les contraintes spatiales (*Spatial_Constraints*) et les contraintes géométriques (*Geometric_Constraints*).
Pour le niveau Instance, les différents plans de tâches (*Task_Plan*) sont décrits et instanciés à l'aide des spécifications de l'action primitives *PAS* qui décrivent chaque tâche primitive. Chaque *PAS* contient l'identité d'une tâche primitive, une contrainte spatiale (*Spatial_Constraint*), un objet à manipuler et un système de référence (qui peut être un artéfact (*Artifact*) ou un lieu (*Place*)). Les différentes action primitves (*Primitive_Action*) sont également définies et sont "Insérer" (*Insert*) et Retirer (*Remove*).

Ensuite, il faut instancier l'environnement dans lequel se déroule le cas d'utilisation dans l'ontologie ENVOn-2. Cela est montré dans la figure 5.27. Le processus d'instanciation est similaire à celui des cas d'utilisation 1, 2 et 3. Les concepts du niveau Méta sont colorés en bleu clair. Ceux du niveau Domaine sont colorés en bleu foncé (information topologique), jaune (information sémantique) et orange (information géométrique). Les concepts du niveau Instance ne sont pas colorés (blanc). Nous avons simplifié la représentation en ne modélisant que les informations relatives à *New_Battery* et *P2*, mais cela est identique pour les autres artéfacts (*Artifact*) et lieux (*Place*).

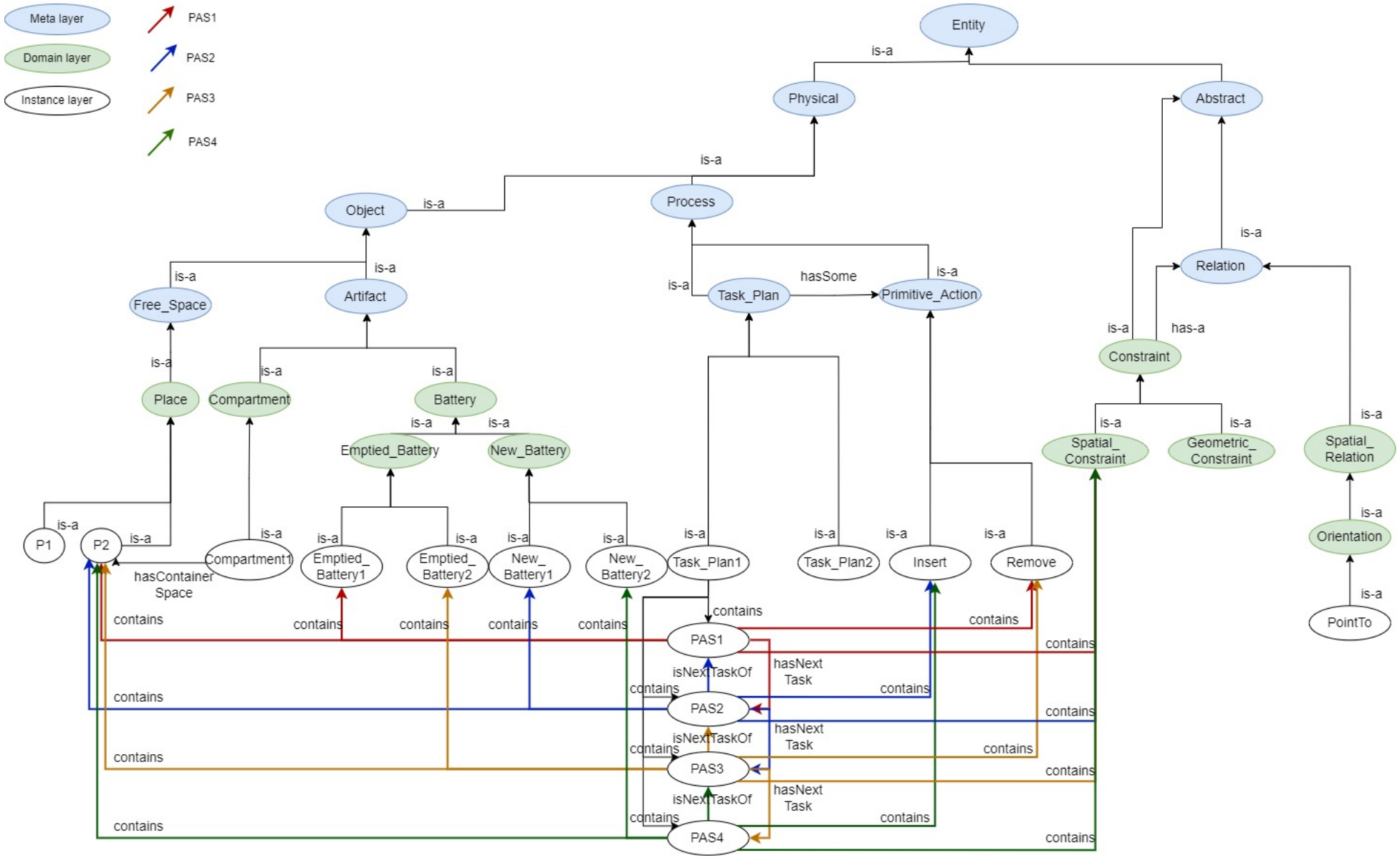


FIGURE 5.26 – Instanciation des informations relatives au TAMP pour le cas d'utilisation n°4 dans TAMPO

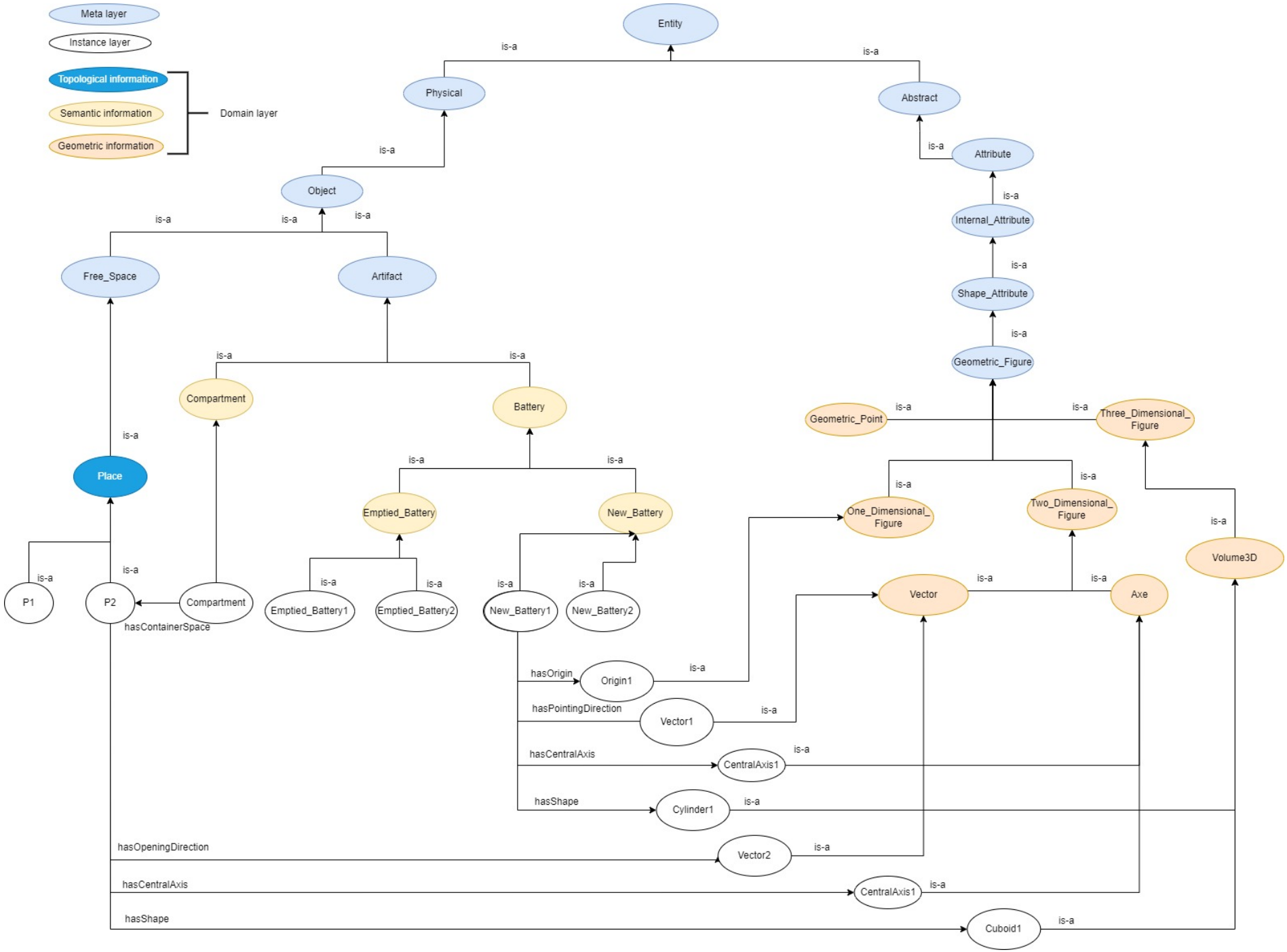


FIGURE 5.27 – Instanciation de l'environnement du cas d'utilisation n°4 dans ENVOn-2

5.5.3 Évaluation des différents plans et choix

L'environnement dans lequel se déroule notre cas d'utilisation ainsi que les deux plans de tâches à considérer sont instanciés dans nos ontologies, respectivement ENVOn-2 et TAMPO. Nous allons maintenant utiliser un raisonneur afin d'évaluer les différents plans, en fonction des contraintes géométriques qu'ils génèrent. Pour cela, nous allons appliquer la fonction de coût qui a été définie dans le paragraphe 4.4.5.

La fonction de coût est donc de la forme F() = A + B + C + D où :

— A : représente le niveau de complexité géométrique autour de la configuration initiale ;

— B : représente le niveau de complexité géométrique autour de la configuration finale ;

— C : représente le niveau de complexité géométrique maximale sur la trajectoire ;

— D : le type d'action primitive à réaliser.

Dans notre cas, le coût de D est constant, car nous intuitons que la tâche primitive « Enlever » est de la même complexité que la tâche primitive « Insérer ». Concernant le coût de C, il n'y a pas d'obstacles sur les trajectoires de l'ensemble des tâches primitives. Les seules contraintes géométriques liées à l'environnement sont autour des configurations initiales et finales. Ainsi, dans ce cas d'utilisation, seuls les coûts liés à A et B vont influencer le coût total du plan.

Les coûts de A et de B peuvent prendre comme valeurs 0, 1 ..., 6. Tous les objets peuvent être visualisés sous six angles différents (arrière, de face, de droite, de gauche, de dessus et de dessous). Nous considérons que si un obstacle est devant une de ces vues, le coût augmente de 1. Ainsi, si un objet est volant, le niveau de complexité autour de cette configuration sera de 0. En revanche, si un objet est enfermé dans une boîte, le niveau de complexité de cette configuration sera de 6.
En associant ainsi les différents coût (A, B, C et D) à chacune des tâches primitives, le raisonneur peut évaluer nos différents plans de tâches. Ainsi, à l'aide d'une requête SPARQL, nous obtenons les évaluations des plans de tâches qui sont présentées dans la figure 5.28.
Dans le paragraphe 5.3.4, nous avions émis l'hypothèse qu'il était plus simple d'insérer une pile neuve dans un compartiment vide ou de retirer une pile usagée si elle est seule dans le compartiment. Ces hypothèses semblent également être validées par notre approche. En effet, celle-ci évalue le plan de tâche n°1, avec un score de 16, comme générant plus de contraintes géométriques et donc, étant moins facilement solvable par la suite pour un planificateur de trajectoires. Le plan de tâche n°2 quant à lui, avec un score de 10, génère donc moins de contraintes géométriques. En effet, après avoir enlevé la première pile usagée, il y a désormais plus d'espace libre dans le compartiment pour enlever la seconde pile usagée. De même, quand la seconde pile usagée est enlevée, le compartiment est désormais vide pour l'insertion de la première pile

SPARQL Query | Snap SPARQL Query

Snap SPARQL Query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX tampo: <http://www.semanticweb.org/florent/ontologies/2022/9/TPO-20#>


SELECT ?TP1 ?PA1 ?PA2 ?PA3 ?PA4 ?c1 ?TP2 ?PA11 ?PA22 ?PA33 ?PA44 ?c2
WHERE {
            ?TP1 a tampo:Task_Plan;
            tampo:hasName "Task_Plan1";
            tampo:PA_Cost ?c1.

            ?PA1 a tampo:Primitive_Action;
            tampo:hasName "TP1_Remove_First_Battery".
            ?PA2 a tampo:Primitive_Action;
            tampo:hasName "TP1_Insert_First_Battery".
            ?PA3 a tampo:Primitive_Action;
            tampo:hasName "TP1_Remove_Second_Battery".
            ?PA4 a tampo:Primitive_Action;
            tampo:hasName "TP1_Insert_Second_Battery".

            ?TP2 a tampo:Task_Plan;
            tampo:hasName "Task_Plan2";
            tampo:PA_Cost ?c2.


            ?PA11 a tampo:Primitive_Action;
            tampo:hasName "TP2_Remove_First_Battery".
            ?PA22 a tampo:Primitive_Action;
            tampo:hasName "TP2_Remove_Second_Battery".
            ?PA33 a tampo:Primitive_Action;
            tampo:hasName "TP2_Insert_First_Battery".
            ?PA44 a tampo:Primitive_Action;
            tampo:hasName "TP2_Insert_Second_Battery".
}
```

?TP1	?PA1	?PA2	?PA3	?PA4	?c1
tampo:Task_Plan1	tampo:TP1_Remove_First_Battery	tampo:TP1_Insert_First_Battery	tampo:TP1_Remove_Second_Battery	tampo:TP1_Insert_Second_Battery	16
?TP2	**?PA11**	**?PA22**	**?PA33**	**?PA44**	**?c2**
tampo:Task_Plan2	tampo:TP2_Remove_First_Battery	tampo:TP2_Remove_Second_Battery	tampo:TP2_Insert_First_Battery	tampo:TP2_Insert_Second_Battery	10

FIGURE 5.28 – Évaluation des plans de tâches à l'aide de requête SPARQL

neuve. Le plan de tâches avec le coût le plus faible, donc ici le plan n°2, est à privilégier. A priori, il devrait être plus facilement solvable par un planificateur de trajectoires.

5.5.4 Simulation et validation sur un cas d'utilisation : changement de piles

Après avoir instancié nos différentes ontologies, des outils de raisonnement ont pu proposer le plan générant le moins de contraintes géométriques et donc, le plus simple à résoudre pour un planificateur de trajectoires. Il faut toutefois vérifier ce résultat et pour cela, nous exécutons le processus de changement de piles selon les deux plans de tâches définis dans la partie 5.5.1. Pour cela, nous avons procédé comme suit :

— Nous avons exécuté les deux plans de tâches en considérant, comme critère de comparaison, le temps total pour effectuer l'ensemble des actions primitives ;

— Nous utilisons deux stratégies de planifications : la stratégie *GT*, qui est une solution qui a déjà montré son intérêt ([Cailhol et al., 2019]) et la stratégie *MG*TO* qui est la meilleure

stratégie présentée dans ces travaux (résultats présentées dans le paragraphe 5.4.1). Cela permettra ainsi de discuter des performances des différentes stratégies de planification.

Les résultats du tableau 5.6 montrent le temps de calcul nécessaire à la réalisation des plans de tâches n°1 et n°2 pour deux stratégies de planification différentes, *GT* et *MG*TO*.
Ces résultats montrent que le plan de tâche n°1, qui est le plus contraint géométriquement, est plus long à résoudre pour un planificateur de trajectoires que le plan de tâche 2. Cela, indépendamment de la stratégie de planification utilisée.

	Plan de tâche n°1	Plan de tâche n*2
Stratégie GT (s)	76.37	35.5
Stratégie MG*TO (s)	25.7	18.46

TABLE 5.6 – Temps de calcul nécessaire pour résoudre l'ensemble des plans de tâches

Nous pouvons détailler un peu plus ces résultats en regardant individuellement chacune des tâches primitives (tableau 5.7) :

— Nous pouvons voir que les tâches d'insertions sont plus coûteuses que les tâches de retrait. Le temps de calcul mis par le planificateur de trajectoires pour résoudre une tâche "Insérer" est plus long que pour une tâche "Enlever". Cela peut s'expliquer par le fonctionnement intrinsèque de l'algorithme Bi-RRT. En effet, il est plus simple de sortir d'une zone que d'y accéder. Ainsi, le coût D de la fonction de coût (paragraphe 5.5.3) est à ajuster. Une tâche d'insertion devrait avoir un coût plus élevé qu'une tâche de retrait.

— Comme dans la partie précédente 5.3, la stratégie *MG*TO* se montre bien plus performante que la stratégie *GT*. Cela est dû à la réduction de l'espace de recherche de l'algorithme Bi-RRT. Également, la stratégie multi-planificateurs arrive, selon la position du jalon tiré, à interpoler une trajectoire directe lors de la première étape topologique pour les tâches *« Enlever pile »* (figure 5.29). Ceci explique également pourquoi les tâches *« Enlever pile »* sont plus simples à résoudre que les tâches *« Insérer pile »* quand la stratégie MG*TO est utilisée. Enfin, le fait d'appliquer des contraintes géométriques sur le mouvement permet aussi d'améliorer les performances de la planification de trajectoires.

— Les tâches 1 et 4 des deux plans ont des performances comparables, car il s'agit de la même tâche à résoudre. En effet, les contraintes géométriques exercées sont les mêmes, elles diffèrent uniquement pour les tâches 2 et 3.

— Pour les tâches 2 et 3, les contraintes géométriques sont plus faibles et les performances sont donc meilleures.

Les figures 5.30 et 5.31 présentent les trajectoires obtenues pour les deux plans de tâches. Dans les deux cas, les trajectoires obtenues en utilisant la stratégie *MG*TO* sont plus pertinentes que

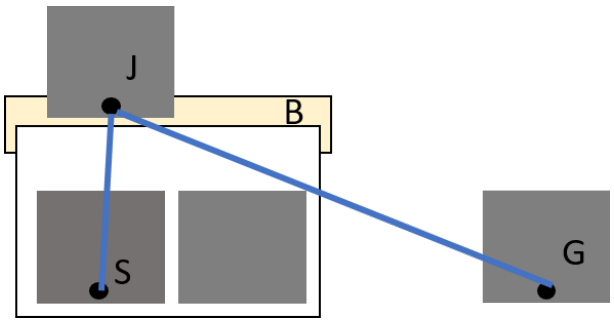


FIGURE 5.29 – Graphe topologique pour une tâche de retrait

	Plan de tâche 1	
	GT	MG*TO
Tâche Primitive 1 « Enlever pile usagée 1 »	8.51	1.28
Tâche Primitive 2 « Insérer pile neuve 1 »	39.87	8.54
Tâche Primitive 3 « Enlever pile usagée 2 »	6.13	1.64
Tâche Primitive 4 « Insérer pile neuve 2 »	21.86	14.24

	Plan de tâche 2	
	GT	MG*TO
Tâche Primitive 1 « Enlever pile usagée 1 »	8.49	1.29
Tâche Primitive 2 « Enlever pile usagée 2 »	1.30	1.23
Tâche Primitive 3 « Insérer pile neuve 1 »	3.85	1.67
Tâche Primitive 4 « Insérer pile neuve 2 »	21.86	14.27

TABLE 5.7 – Temps de calcul pour résoudre chacune des actions primitives des différents plans de tâches

celles obtenues avec la stratégie *GT*. En effet, l'espace de recherche est réduit, ce qui permet d'éviter les zones inutiles, ce qui n'est pas le cas avec la stratégie *GT* (visible sur la figure 5.30 d)).

5.5.5 Synthèse des résultats de notre approche collaborative

Nous avons validé notre approche collaborative sur un cas d'utilisation de changement de piles dans un compartiment. Ce cas d'utilisation permet à la fois de valider nos stratégies de planification de trajectoires (comme ce qui a été fait dans la partie 5.3) mais également de valider notre méthodologie basée ontologie pour le couplage sémantique de la planifications de tâches et de trajectoires. Cela permet, entre autres, de valider notre démarche dans le choix du plan de tâches. En effet, nous avons simulé deux plans de tâches différents et celui qui a été proposé par le raisonneur est en effet le plus rapide à résoudre par le planificateur de trajectoires.

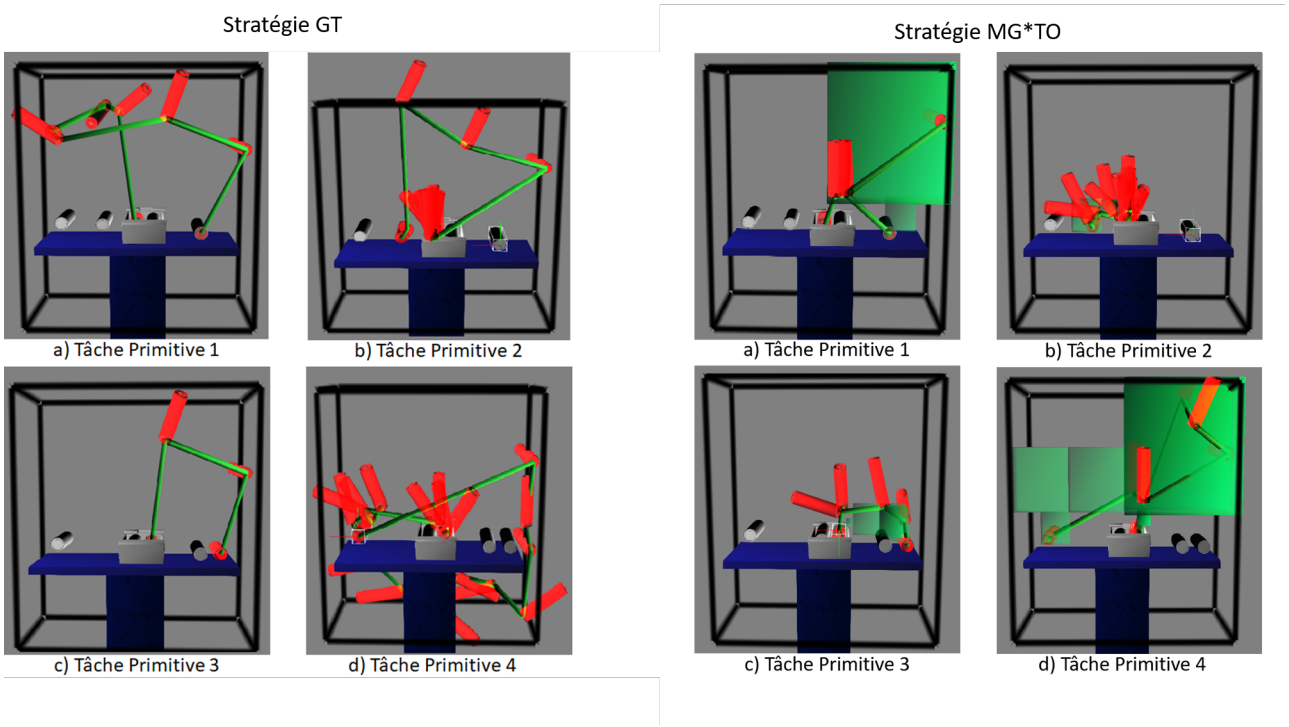


FIGURE 5.30 – Trajectoires obtenues pour les différentes tâches primitives du plan de tâches n°1 (Stratégie GT et MG*TO)

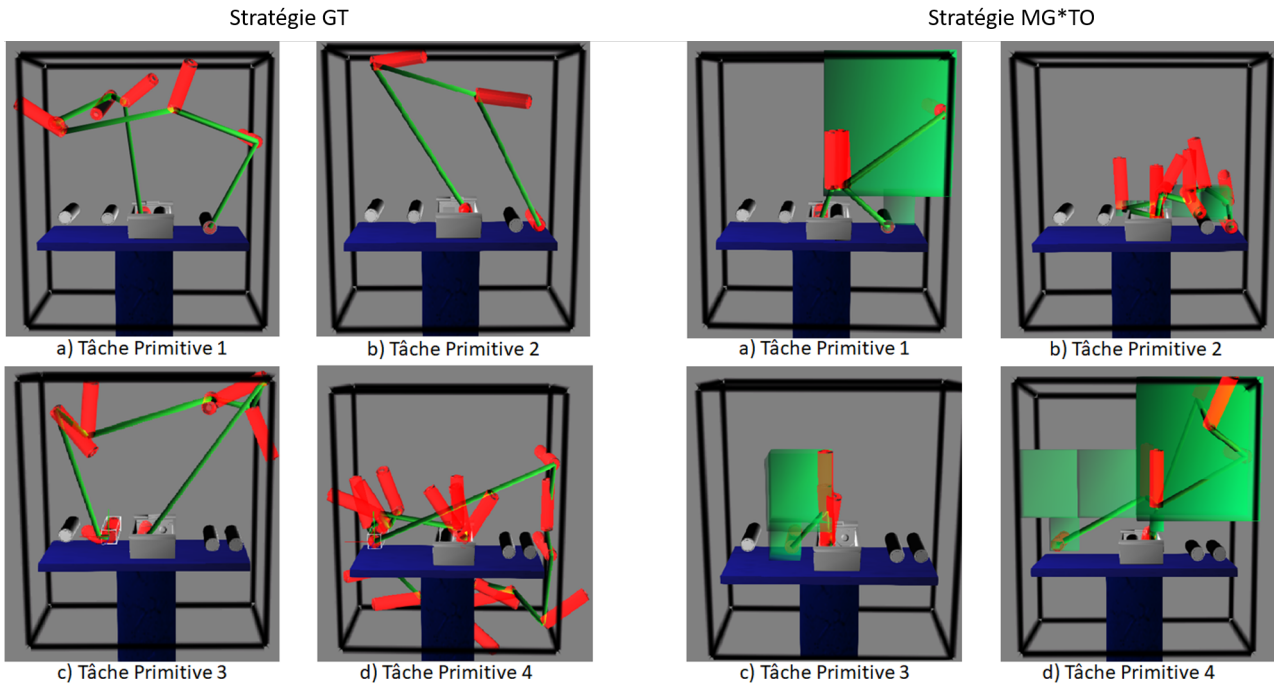


FIGURE 5.31 – Trajectoires obtenues pour les différentes tâches primitives du plan de tâches n°2 (Stratégie GT et MG*TO)

Par ailleurs, ce cas d'utilisation permet également de valider nos ontologies. En effet, nous avons instancié l'environnement de simulation dans notre ontologie ENVOn-2. Nous avons aussi instancié nos plans de tâches dans l'ontologie TAMPO et les avons évalués pour proposer le plan le plus pertinent à l'opérateur. Ceci permet de valider les questions de compétences liées aux plans de tâches.

5.6 Synthèse

Les différentes expérimentations qui ont été présentées dans ce chapitre permettent de valider nos démarches.

Dans un premier temps, nous avons validé nos stratégies de planification de trajectoires. En effet, avant de vouloir considérer conjointement les planifications de trajectoires et de tâches, il est important d'améliorer les performances de la planification de trajectoires.
Ces stratégies ont été mises en œuvre sur différents cas d'utilisation, de complexité différente, afin de pouvoir valider leur généricité. Cela a permis de montrer qu'il est important, d'autant plus dans des environnements très complexes, de réduire l'espace de recherche d'un algorithme probabiliste. Il est également intéressant de pouvoir limiter, quand cela est possible, l'utilisation d'un tel algorithme, car il est très coûteux.
De plus, nous ne nous contentons pas de considérer uniquement des informations relatives à l'environnement, mais également à la tâche en elle-même. Celles-ci permettent de générer des contraintes sur la planification de trajectoires pour restreindre les mouvements et limiter les tirages peu pertinents.
De plus, pour la validation de ces stratégies, nous avons instancié différents environnements dans notre ontologie ENVOn-2 et différentes tâches primitives dans TAMPO, ce qui nous a permis de les valider, en répondant à l'ensemble des questions de compétences définies dans le chapitre 3.
Après avoir validé nos concepts pour améliorer la planification de trajectoires, et à la représentation des informations qui lui sont liées, nous avons validé notre module d'assistance TAMP.
Pour cela, nous avons :

— Validé le choix du plan de tâches à effectuer à l'aide de processus de raisonnement et d'une ontologie (TAMPO) ;

— Vérifié et validé les résultats fournis par le raisonneur.

Nous avons ainsi déroulé notre approche TAMP sur les deux plans de tâches retenus pour vérifier la proposition faite par le raisonneur tout en comparant les performances de notre planificateur de trajectoires.

Chapitre 6

Conclusion et perspectives

6.1 Synthèse des contributions

Les travaux de cette thèse présentent une approche de couplage sémantique entre un planificateur de tâches et un planificateur de trajectoires qui améliorent les performances de cette collaboration. Pour cela, nous avons proposé :

— **Deux ontologies**. Ces deux ontologies, qui s'appuient sur une même ontologie de haut niveau, SUMO, afin d'être interopérables, sont :

  — ENVOn-2, qui permet de modéliser l'environnement selon trois niveaux d'abstraction de l'information (sémantiques, topologiques et géométriques). Cela permet de modéliser différents types d'informations à la fois sur les objets rigides, mais également sur l'environnement.

    — Les objets rigides sont représentés, au niveau géométrique, par leur représentation (volume (BREP) ou surface (CSG)). À ces représentations sont attachées des informations sémantiques pouvant décrire la forme de l'objet, s'il est mobile ou non...

    — L'espace libre est lui représenté au niveau géométrique par un octree. À partir de celui-ci, un graphe topologique est défini qui représente les lieux importants de l'environnement ainsi que les frontières entre ceux-ci. Des informations sémantiques sont aussi associées au graphe (décrivant la difficulté de passage causée par une marge faible entre l'objet et son emplacement, si un objet mobile peut obstruer le chemin etc ...)

  — TAMPO, qui s'appuie sur ENVOn-2, permet de modéliser les informations relatives à la planification de tâches et de trajectoires.
  Elle permet :

  - D'améliorer la pertinence du plan de tâches en proposant celui possédant le moins de contraintes géométriques afin de réduire la complexité de la planification de trajectoires ;
  - D'inférer des contraintes géométriques afin de contrôler la trajectoire à partir de contraintes spatiales définies par l'opérateur ;
  - De générer des requêtes de planification de trajectoires, pour chaque action primitive du plan de tâches proposé.

— **Une méthodologie pour le couplage sémantique du TAMP**. Elle doit permettre d'améliorer les performances de planification de trajectoires lors de simulation de tâches. Elle permet :

  - D'améliorer les stratégies de planifications de trajectoires pour une action primitive d'un plan de tâches, en réduisant les temps de calcul nécessaires pour obtenir un résultat tout en améliorant la pertinence de la trajectoire proposée.. Pour cela, nous proposons d'utiliser des combinaisons d'algorithmes pour 1) réduire l'espace de recherches d'un algorithme probabiliste traditionnellement utilisé (en fournissant une sous espace de recherche sous forme de tunnel dans l'octree) et 2) choisir un planificateur de trajectoires cohérent selon le contexte sémantique et géométrique local. En effet, à l'aide des informations de l'ontologie ENVOn-2, le système peut choisir d'utiliser un algorithme peu coûteux (qui correspond à une interpolation directe) si l'environnement est peu contraint géométriquement, ou un algorithme probabiliste (l'algorithme Bi-RRT dans notre cas) si l'environnement est contraint géométriquement.
  - De proposer un plan de tâches pertinent vis-à-vis de la tâche à réaliser. Différents plans de tâches peuvent être générés soit par le planificateurs de tâches soit par l'opérateur humain. Ces plans peuplent ensuite l'ontologie TAMPO et sont ensuite évalués à l'aide de règles d'inférences et d'un raisonneur, afin de les comparer et de ressortir le meilleur.
  - De contrôler la planification de trajectoires à l'aide de contraintes géométriques qui sont inférées, à l'aide de notre ontologie TAMPO, à partir de contraintes spatiales exprimées au niveau tâches par un opérateur humain.

6.2 Perspectives

Nous avons montré l'intérêt et la pertinence de notre approche pour le couplage sémantique de la planification de tâches et de trajectoires basé ontologies pour simuler des processus d'assemblages, de maintenances ou de désassemblages. De nombreuses perspec-

tives peuvent être envisagées pour améliorer cette approche, à court et à long terme.

A court terme, les contraintes spatiales exprimées au niveau tâche pourrait être inférées automatiquement (paragraphe 6.2.1). De plus, cette approche ne considère pas encore les possibles échecs provenant des différents planificateurs utilisés (paragraphe 6.2.2).
A long terme, cette approche pourrait être étendue à une utilisation en RV, dans laquelle un opérateur humain aurait la possibilité d'agir directement sur la simulation (paragraphes 6.2.3 et (6.2.4)).

6.2.1 Utiliser la méréotopologie pour générer les contraintes spatiales

Dans les travaux de cette thèse, les contraintes spatiales exprimées au niveau tâche sont manuellement renseignées par un opérateur. Une des améliorations possibles serait de les définir automatiquement. Toutefois, l'expression de ces contraintes spatiales n'est pas générique dans l'ensemble de l'état de l'art ([Borrmann et al., 2009], [Belouaer et al., 2011], [Touya et al., 2012]).
Une idée est alors d'utiliser la méréotopologie [Smith, 1996] pour générer automatiquement ces contraintes spatiales. La plupart des approches de la méréotopologie utilisent la Region Connection Calculus 8 (RCC8) ([Randell et al., 1992]), qui est un ensemble de huit relations binaires conjointement exhaustives et disjointes par paire représentant les relations méréotopologiques entre des paires d'individus. Puis, à partir de la méréotopologie, il est possible de générer automatiquement, des contraintes spatiales.
Des premières pistes ont été explorées ([Leoty et al., 2023]). Ces travaux démontrent l'intérêt d'utiliser la méréotopologie pour générer automatiquement des contraintes spatiales. L'exemple illustratif est celui du scénario de changement de piles présenté dans le paragraphe 5.3.3 du chapitre 5. Dans ces travaux, le scénario est défini comme un ensemble d'axiomes exprimés en logique du premier ordre. Puis en utilisant un prouveur de théorème automatisé, il est possible de générer les contraintes spatiales à appliquer.

6.2.2 Intégrer le rebouclage entre les planificateurs en cas d'échec

A ce stade, notre approche pour le couplage TAMP ne prend pas en compte les cas d'échecs des planificateurs. Il faudrait alors ajouter un rebouclage permettant, en cas d'échec, de trouver une alternative faisable.
Par exemple si pour une tâche primitive, le planificateur de trajectoires n'arrive pas à trouver une trajectoire, des stratégies originales devraient être développées pour que le *Gestionnaire des planificateurs* en tienne compte pour calculer de nouveaux plans.

6.2.3 Développer de nouvelles modalités d'interaction entre un opérateur humain et le module d'assistance TAMP

L'approche proposée a montré son efficacité dans un environnement virtuel. Toutefois, dans des environnements très contraints, les temps de calcul pour obtenir une trajectoire peuvent être très longs.
Les travaux menés au LGP depuis plus de quinze ans portent sur l'utilisation conjointe des techniques de planification et de la Réalité Virtuelle pour améliorer les simulations des tâches du PLM. Des premiers travaux ont permis à un opérateur humain d'interagir avec le planificateur de trajectoires, en lui permettant de prendre le contrôle ou de modifier la trajectoire proposée par le planificateur de trajectoires. Développer cette démarche pour rendre compatible l'ensemble du module d'assistance TAMP et la RV est alors une nouvelle piste à explorer. Parmi les améliorations possibles, il pourrait être avantageux qu'un opérateur humain puisse agir en temps réel sur le plan de tâches proposé par le module d'assistance TAMP.

6.2.4 Remise en cause du plan de tâches par un opérateur humain

S'il est important de rendre l'interaction possible entre un opérateur humain et le module d'assistance TAMP, il est également important d'améliorer les performances de cette collaboration, afin de la rendre plus fluide et plus réaliste. Pour cela, la détection d'intention, déjà prise en compte pour améliorer la planification de trajectoires, pourrait être utilisée lorsque l'opérateur humain souhaite modifier le plan de tâches proposé par le planificateur de tâches. Par exemple, si l'opérateur humain décide de commencer sa manipulation par une tâche primitive différente de celle proposée par le planificateur, ce dernier devra recalculer un plan de tâches commençant par la dite tâche. Cela permettrait à l'opérateur humain d'avoir un contrôle en temps réel sur l'ensemble de la planification TAMP.

Bibliographie

[Agostini et al., 2020] Agostini, A., Saveriano, M., Lee, D., and Piater, J. (2020). Manipulation Planning Using Object-Centered Predicates and Hierarchical Decomposition of Contextual Actions. *IEEE Robotics and Automation Letters*, 5(4) :5629–5636. Conference Name : IEEE Robotics and Automation Letters.

[Ahmadi-Pajouh et al., 2007] Ahmadi-Pajouh, M. A., Towhidkhah, F., Gharibzadeh, S., and Mashhadimalek, M. (2007). Path planning in the hippocampo-prefrontal cortex pathway : an adaptive model based receding horizon planner. *Medical Hypotheses*, 68(6) :1411–1415.

[Akbari et al., 2019] Akbari, A., Lagriffoul, F., and Rosell, J. (2019). Combined heuristic task and motion planning for bi-manual robots. *Autonomous Robots*, 43(6) :1575–1590.

[Alt and Welzl, 1988] Alt, H. and Welzl, E. (1988). Visibility graphs and obstacle-avoiding shortest paths. *Zeitschrift für Operations-Research*, 32(3) :145–164. Number : 3 Reporter : Zeitschrift für Operations-Research.

[Amato and Dale, 1999] Amato, N. and Dale, L. (1999). Probabilistic roadmap methods are embarrassingly parallel. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 1, pages 688–694 vol.1. ISSN : 1050-4729.

[Amato and Wu, 1996] Amato, N. and Wu, Y. (1996). A randomized roadmap method for path and manipulation planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 113–120 vol.1. ISSN : 1050-4729.

[Arp, 2015] Arp, T. M. (2015). Building Ontologies with Basic Formal Ontology. Reporter : The MIT Press.

[Barbau et al., 2012] Barbau, R., Krima, S., Rachuri, S., Narayanan, A., Fiorentini, X., Foufou, S., and Sriram, R. D. (2012). OntoSTEP : Enriching product model data using ontologies. *Computer-Aided Design*, 44(6) :575–590. Number : 6 Reporter : Computer-Aided Design.

[Bastianelli et al., 2013] Bastianelli, E., Bloisi, D., Capobianco, R., Gemignani, G., Iocchi, L., and Nardi, D. (2013). Knowledge Representation for Robots through Human-Robot Interaction. *arXiv :1307.7351 [cs]*. arXiv : 1307.7351.

[Belouaer et al., 2011] Belouaer, L., Bouzid, M., and Mouaddib, A.-I. (2011). Spatial knowledge in planning language. In *International Conference on Knowledge Engineering and Ontology Development*.

[Belsky et al., 2016] Belsky, M., Sacks, R., and Brilakis, I. (2016). Semantic Enrichment for Building Information Modeling. *Computer-Aided Civil and Infrastructure Engineering*, 31(4) :261–274. _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.12128.

[Benjamin Perakath, 1994] Benjamin Perakath, C. (1994). The idef5 ontology description capture method overview. *Texas : Knoledge Based Systems Incorporation*.

[Bennes et al., 2012] Bennes, L., Bazzaro, F., and Sagot, J.-C. (2012). Virtual reality as a support tool for ergonomic-style convergence : multidisciplinary interaction design methodology and case study. In *Proceedings of the 2012 Virtual Reality International Conference*, pages 1–10.

[Bidot et al., 2017] Bidot, J., Karlsson, L., Lagriffoul, F., and Saffiotti, A. (2017). Geometric backtracking for combined task and motion planning in robotic systems. *Artificial Intelligence*, 247 :229–265. Reporter : Artificial Intelligence.

[Biundo, 2000] Biundo, S. (2000). *Recent advances in AI planning : proceedings*. Number 1809 in Lecture notes in computer science Lecture notes in artificial intelligence. Springer, Berlin. OCLC : 248010185.

[Blum and Furst, 1997] Blum, A. L. and Furst, M. L. (1997). Fast planning through planning graph analysis. *Artificial intelligence*, 90(1-2) :281–300.

[Blum and Langford, 2000] Blum, A. L. and Langford, J. C. (2000). Probabilistic Planning in the Graphplan Framework. In Biundo, S. and Fox, M., editors, *Recent Advances in AI Planning*, Lecture Notes in Computer Science, pages 319–332, Berlin, Heidelberg. Springer.

[Borrmann et al., 2009] Borrmann, A., Hyvärinen, J., and Rank, E. (2009). Spatial constraints in collaborative design processes. *Proceedings of the International Conference on Intelligent Computing in Engineering*.

[Bouaud et al., 1994] Bouaud, J., Bachimont, B., Charlet, J., and Zweigenbaum, P. (1994). Acquisition And Structuring Of An Ontology Within Conceptual Graphs.

[Bourdot et al., 2010] Bourdot, P., Convard, T., Picon, F., Ammi, M., Touraine, D., and Vézien, J. M. (2010). VR–CAD integration : Multimodal immersive interaction and advanced haptic paradigms for implicit edition of CAD models. *Computer-Aided Design*, 42(5) :445–461.

[Bowen and Alterovitz, 2014] Bowen, C. and Alterovitz, R. (2014). Closed-loop global motion planning for reactive execution of learned tasks. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1754–1760. ISSN : 2153-0866.

[Brooks and Lozano-Perez, 1985] Brooks, R. A. and Lozano-Perez, T. (1985). A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics*.

[Cailhol et al., 2015] Cailhol, S., Fillatreau, P., Fourquet, J.-Y., and Zhao, Y. (2015). A hierarchic approach for path planning in virtual reality. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 9(4) :291–302. Number : 4 Reporter : International Journal on Interactive Design and Manufacturing (IJIDeM).

[Cailhol et al., 2019] Cailhol, S., Fillatreau, P., Zhao, Y., and Fourquet, J.-Y. (2019). Multilayer path planning control for the simulation of manipulation tasks : Involving semantics and topology. *Robotics and Computer-Integrated Manufacturing*, 57 :17–28. Reporter : Robotics and Computer-Integrated Manufacturing.

[Caldiran et al., 2009] Caldiran, O., Haspalamutgil, K., Ok, A., Palaz, C., Erdem, E., and Patoglu, V. (2009). Bridging the Gap between High-Level Reasoning and Low-Level Control. In Erdem, E., Lin, F., and Schaub, T., editors, *Logic Programming and Nonmonotonic Reasoning*, Lecture Notes in Computer Science, pages 342–354, Berlin, Heidelberg. Springer.

[Cantamessa, 2012] Cantamessa (2012). An empirical analysis of the PLM implementation effects in the aerospace industry | Marco Cantamessa | Request PDF.

[Castaman et al., 2021] Castaman, N., Pagello, E., Menegatti, E., and Pretto, A. (2021). Receding Horizon Task and Motion Planning in Changing Environments. *Robotics and Autonomous Systems*, 145 :103863.

[Cayrol et al., 2001] Cayrol, M., Régnier, P., and Vidal, V. (2001). Least commitment in Graphplan. *Artificial Intelligence*, 130(1) :85–118.

[Chen, 2006] Chen, D. (2006). Enterprise Interoperability Framework.

[Choset et al., 2005] Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., and Thrun, S. (2005). Principles of Robot Motion : Theory, Algorithms, and Implementation. page 150.

[Dandois and Ellis, 2013] Dandois, J. and Ellis, E. (2013). High spatial resolution three-dimensional mapping of vegetation spectral dynamics using computer vision. *Remote Sensing of Environment*, 136 :259–276. Reporter : Remote Sensing of Environment.

[Dantam et al., 2018] Dantam, N. T., Chaudhuri, S., and Kavraki, L. E. (2018). The Task-Motion Kit : An Open Source, General-Purpose Task and Motion-Planning Framework. *IEEE Robotics & Automation Magazine*, 25(3) :61–70. Conference Name : IEEE Robotics & Automation Magazine.

[Dantam et al., 2016] Dantam, N. T., Kingston, Z. K., Chaudhuri, S., and Kavraki, L. E. (2016). Incremental Task and Motion Planning : A Constraint-Based Approach. In *Robotics : Science and Systems XII*. Robotics : Science and Systems Foundation.

[Devy et al., 1995] Devy, M., Chatila, R., Fillatreau, P., Lacroix, S., and Nashashibi, F. (1995). On autonomous navigation in a natural environment. *Robotics and Autonomous Systems*, 16(1) :5–16.

[Di Gironimo et al., 2013] Di Gironimo, G., Matrone, G., Tarallo, A., Trotta, M., and Lanzotti, A. (2013). A virtual reality approach for usability assessment : case study on a wheelchair-mounted robot manipulator. *Engineering with Computers*, 29(3) :359–373.

[Diab et al., 2019] Diab, M., Akbari, A., Ud Din, M., and Rosell, J. (2019). PMK—A Knowledge Processing Framework for Autonomous Robotics Perception and Manipulation. *Sensors (Basel, Switzerland)*, 19(5).

[Diehl et al., 2021] Diehl, M., Paxton, C., and Ramirez-Amaro, K. (2021). Automated Generation of Robotic Planning Domains from Observations. arXiv :2105.13604 [cs].

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1) :269–271.

[Do and Kambhampati, 2011] Do, M. and Kambhampati, S. (2011). SAPA : A Multi-objective Metric Temporal Planner. Technical report. Publication Title : arXiv e-prints ADS Bibcode : 2011arXiv1106.5260D Type : article.

[Erdem et al., 2011] Erdem, E., Haspalamutgil, K., Palaz, C., Patoglu, V., and Uras, T. (2011). Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation. In *2011 IEEE International Conference on Robotics and Automation*, pages 4575–4581. ISSN : 1050-4729.

[Erol et al., 1994] Erol, K., Hendler, J., and Nau, D. (1994). HTN Planning : Complexity and Expressivity. *Proceedings of the National Conference on Artificial Intelligence*, 2.

[Farreny, 1997] Farreny, H. (1997). Recherche heuristiquement ordonnée : généralisations compatibles avec la complétude et l'admissibilité. *Technique et Science Informatiques, RAIRO*, 16(7) :25–953.

[Fernández-López et al., 1997] Fernández-López, M., Gómez-Pérez, A., and Juristo, N. (1997). Methontology : from ontological art towards ontological engineering.

[Fikes and Nilsson, 1971] Fikes, R. E. and Nilsson, N. J. (1971). Strips : A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3) :189–208.

[Fourman, 2000] Fourman, M. (2000). Propositional Planning.

[Fuchs and Moreau, 2003] Fuchs, P. and Moreau, G. (2003). Le Traité de la réalité virtuelle.

[Gangemi et al., 2002] Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., and Schneider, L. (2002). Sweetening Ontologies with DOLCE. In Gómez-Pérez, A. and Benjamins, V. R., editors, *Knowledge Engineering and Knowledge Management : Ontologies and the Semantic Web*, Lecture Notes in Computer Science, pages 166–181, Berlin, Heidelberg. Springer.

[Gangemi et al., 2007] Gangemi, A., Steve, G., and Giacomelli, F. (2007). ONIONS : An Ontological Methodology for Taxonomic Knowledge Integration.

[Gaševic et al., 2009] Gaševic, D., Djuric, D., and Devedžic, V. (2009). *Model driven engineering and ontology development*. Springer Science & Business Media.

[Ghallab et al., 2016] Ghallab, M., Nau, D., and Traverso, P. (2016). *Automated planning and acting*. Cambridge University Press.

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2) :199–220. Number : 2 Reporter : Knowledge Acquisition.

[Gruninger, 1995] Gruninger, M. (1995). Methodology for the design and evaluation of ontologies. In *Proc. IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing*.

[Gruninger and Fox, 1994] Gruninger, M. and Fox, M. S. (1994). The design and evaluation of ontologies for enterprise engineering. In *Workshop on Implemented Ontologies, European Conference on Artificial Intelligence (ECAI)*.

[Guarino, 1998] Guarino, N. (1998). Formal Ontology and Information Systems.

[Guitton, 2009] Guitton, J. (2009). Taking Into Account Geometric Constraints for Task-oriented Motion Planning. page 9.

[Guo et al., 2023] Guo, H., Wu, F., Qin, Y., Li, R., Li, K., and Li, K. (2023). Recent trends in task and motion planning for robotics : A survey. *ACM Computing Surveys*.

[Günther et al., 2017] Günther, M., Wiemann, T., Albrecht, S., and Hertzberg, J. (2017). Model-based furniture recognition for building semantic object maps. *Artificial Intelligence*, 247 :336–351.

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2) :100–107. Conference Name : IEEE Transactions on Systems Science and Cybernetics.

[Hoehndorf, 2010] Hoehndorf, R. (2010). What is an upper-level ontology?

[Hoffmann, 2001] Hoffmann, J. (2001). FF : The Fast-Forward Planning System. *AI Magazine*, 22(3) :57–57. Number : 3.

[Hsu et al., 1998] Hsu, D., Kavraki, L. E., Latombe, J.-C., Motwani, R., and Sorkin, S. (1998). On finding narrow passages with probabilistic roadmap planners. In *Proceedings of the third workshop on the algorithmic foundations of robotics on Robotics : the algorithmic perspective : the algorithmic perspective*, WAFR '98, pages 141–153, USA. A. K. Peters, Ltd.

[Iacob et al., 2012] Iacob, R., Popescu, D., and Mitrouchev, P. (2012). Assembly/disassembly analysis and modeling techniques : A review. *Strojniški vestnik-Journal of Mechanical Engineering*, 58(11) :653–664.

[Il Hong Suh et al., 2007] Il Hong Suh, Gi Hyun Lim, Wonil Hwang, Hyowon Suh, Jung-Hwa Choi, and Young-Tack Park (2007). Ontology-based multi-layered robot knowledge framework (OMRKF) for robot intelligence. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 429–436. Meeting Name : 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems Reporter : 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems.

[Kaelbling, 2011] Kaelbling, L. P. (2011). Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pages 1470–1477, Shanghai, China. IEEE. Meeting Name : 2011 IEEE International Conference on Robotics and Automation (ICRA) Reporter : 2011 IEEE International Conference on Robotics and Automation.

[Kaelbling and Lozano-Pérez, 2011] Kaelbling, L. P. and Lozano-Pérez, T. (2011). Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pages 1470–1477. ISSN : 1050-4729.

[Karaman and Frazzoli, 2010] Karaman, S. and Frazzoli, E. (2010). Incremental Sampling-based Algorithms for Optimal Motion Planning. *arXiv :1005.0416 [cs]*. arXiv : 1005.0416.

[Kavraki et al., 1996] Kavraki, L. E., Svestka, P., Latombe, J., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4) :566–580. Number : 4 Reporter : IEEE Transactions on Robotics and Automation.

[Khatib, 1985] Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *1985 IEEE International Conference on Robotics and Automation Proceedings*, volume 2, pages 500–505. Meeting Name : 1985 IEEE International Conference on Robotics and Automation Proceedings Reporter : 1985 IEEE International Conference on Robotics and Automation Proceedings.

[Kim et al., 2018] Kim, B., Wang, Z., Kaelbling, L. P., and Lozano-Perez, T. (2018). Learning to guide task and motion planning using score-space representation. arXiv :1807.09962 [cs].

[Koditschek, 1987] Koditschek, D. (1987). Exact robot navigation by means of potential functions : Some topological considerations. In *1987 IEEE International Conference on Robotics and Automation Proceedings*, volume 4, pages 1–6.

[Koren and Borenstein, 1991] Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *1991 IEEE International Conference on Robotics and Automation Proceedings*, pages 1398–1404 vol.2.

[Kuffner and LaValle, 2000] Kuffner, J. and LaValle, S. (2000). RRT-connect : An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001 vol.2. ISSN : 1050-4729.

[Kushmerick et al., 1995] Kushmerick, N., Hanks, S., and Weld, D. S. (1995). An algorithm for probabilistic planning. *Artificial Intelligence*, 76(1) :239–286.

[Ladeveze, 2010] Ladeveze, N. (2010). Apport des méthodes de planification automatique dans les simulations interactives d'industrialisation et de maintenance en réalité virtuelle.

[Lagriffoul and Andres, 2016] Lagriffoul, F. and Andres, B. (2016). Combining task and motion planning : A culprit detection problem. *The International Journal of Robotics Research*, 35(8) :890–927.

[Latombe, 1991] Latombe, J.-C. (1991). *Robot Motion Planning*. Springer US, Boston, MA.

[Lavalle, 1998] Lavalle, S. M. (1998). Rapidly-Exploring Random Trees : A New Tool for Path Planning. Technical report.

[Lee et al., 2011] Lee, J., Han, S., and Yang, J. (2011). Construction of a computer-simulated mixed reality environment for virtual factory layout planning. *Computers in Industry*, 62(1) :86–98.

[Lemaignan et al., 2010] Lemaignan, S., Ros, R., Mösenlechner, L., Alami, R., and Beetz, M. (2010). ORO, a knowledge management platform for cognitive architectures in robotics. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3548–3553. ISSN : 2153-0866.

[Leoty et al., 2021] Leoty, F., Fillatreau, P., and Archimède, B. (2021). *Path planning control using high abstraction level environment model and industrial task-oriented knowledge*. Pages : 6.

[Leoty et al., 2023] Leoty, F., Thai, J., Archimède, B., Fillatreau, P., and Grüninger, M. (2023). Using mereotopology for automated spatial inference in task and motion planning. *The 4th RobOntics Workshop, 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN 2023)*.

[Li and Okamura, 2003] Li, M. and Okamura, A. M. (2003). Recognition of operator motions for real-time assistance using virtual fixtures. In *11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings.*, pages 125–131. IEEE.

[Loukil et al., 2006] Loukil, Z., Hamadou, A. B., Marquis, P., and Vidal, V. (2006). Les ressources et la planification temporelle. In *INFORSID*, pages 515–529.

[Loula et al., 2020] Loula, J., Allen, K., Silver, T., and Tenenbaum, J. (2020). Learning constraint-based planning models from demonstrations. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5410–5416. ISSN : 2153-0866.

[Lozano-Perez, 1983] Lozano-Perez (1983). Spatial Planning : A Configuration Space Approach. *IEEE Transactions on Computers*, C-32(2) :108–120. Number : 2 Reporter : IEEE Transactions on Computers.

[McDermott et al., 1998] McDermott, D., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D., Barrett, A., Christianson, D., et al. (1998). Pddl| the planning domain definition language. *Technical Report, Tech. Rep.*

[Meagher, 1982] Meagher, D. (1982). Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2) :129–147. Number : 2 Reporter : Computer Graphics and Image Processing.

[Meyrueis, 2011] Meyrueis, V. (2011). *Modification interactive de formes en réalité virtuelle : application à la conception d'un produit*. PhD thesis, École Nationale Supérieure des Mines de Paris.

[Mirabel and Lamiraux, 2016] Mirabel, J. and Lamiraux, F. (2016). Constraint Graphs : Unifying task and motion planning for Navigation and Manipulation Among Movable Obstacles.

[Neto et al., 2019] Neto, A. B. d. O., Silva, J. A., and Barreto, M. E. (2019). Prototyping and Validating the CORA Ontology : Case Study on a Simulated Reconnaissance Mission. In *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, pages 341–345. ISSN : 2643-685X.

[Niles and Pease, 2001] Niles, I. and Pease, A. (2001). *Towards a Standard Upper Ontology*. Journal Abbreviation : Formal Ontology in Information Systems : Collected Papers from the Second International Conference Pages : 9 Publication Title : Formal Ontology in Information Systems : Collected Papers from the Second International Conference.

[Noreen et al., 2016] Noreen, I., Khan, A., and Habib, Z. (2016). A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms. *IJCSNS International Journal of Computer Science and Network Security, VOL.16 No.10*, page 8.

[Noy et al., 2003] Noy, N. F., Crubézy, M., Fergerson, R. W., Knublauch, H., Tu, S. W., Vendetti, J., and Musen, M. A. (2003). Protégé-2000 : an open-source ontology-development and knowledge-acquisition environment. In *AMIA... annual symposium proceedings. AMIA Symposium*, pages 953–953.

[Olivares-Alarcos et al., 2022] Olivares-Alarcos, A., Foix, S., Borgo, S., and Alenyà, G. (2022). OCRA – An ontology for collaborative robotics and adaptation. *Computers in Industry*, 138 :103627.

[Omar et al., 2015] Omar, R., CKNAH, C. K. M., and Sabudin, E. (2015). Performance comparison of path planning methods. *ARPN J Eng Appl Sci*.

[Overmars, 1992] Overmars, M. H. (1992). A random approach to motion planning.

[Panya et al., 2023] Panya, D. S., Kim, T., and Choo, S. (2023). An interactive design change methodology using a bim-based virtual reality and augmented reality. *Journal of Building Engineering*, 68 :106030.

[Perzylo et al., 2015] Perzylo, A., Somani, N., Rickert, M., and Knoll, A. (2015). An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4197–4203. Meeting Name : 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Reporter : 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

[Pratt, 2001] Pratt, M. J. (2001). Introduction to ISO 10303—the STEP Standard for Product Data Exchange. *Journal of Computing and Information Science in Engineering*, 1(1) :102. Number : 1 Reporter : Journal of Computing and Information Science in Engineering.

[Qureshi and Ayaz, 2016] Qureshi, A. H. and Ayaz, Y. (2016). Potential Functions based Sampling Heuristic For Optimal Path Planning. *Autonomous Robots*, 40(6) :1079–1093. arXiv : 1704.00264.

[Qureshi et al., 2022] Qureshi, A. H., Dong, J., Baig, A., and Yip, M. C. (2022). Constrained Motion Planning Networks X. *IEEE Transactions on Robotics*, 38(2) :868–886. Conference Name : IEEE Transactions on Robotics.

[Randell et al., 1992] Randell, D. A., Cui, Z., and Cohn, A. G. (1992). A spatial logic based on regions and connection. *KR*, 92 :165–176.

[Rosell, 2004] Rosell, J. (2004). Assembly and task planning using Petri nets : A survey. *Proceedings of the Institution of Mechanical Engineers, Part B : Journal of Engineering Manufacture*, 218(8) :987–994. Publisher : IMECHE.

[Rovida et al., 2017] Rovida, F., Grossmann, B., and Krüger, V. (2017). Extended behavior trees for quick definition of flexible robotic tasks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6793–6800. ISSN : 2153-0866.

[Rusu, 2010] Rusu, R. B. (2010). Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *KI - Künstliche Intelligenz*, 24(4) :345–348. Number : 4 Reporter : KI - Künstliche Intelligenz.

[Samet, 1984] Samet, H. (1984). The Quadtree and Related Hierarchical Data Structures. *ACM Computing Surveys*, 16(2) :187–260.

[Silver et al., 2017] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676) :354–359.

[Smith, 1996] Smith, B. (1996). Mereotopology : a theory of parts and boundaries. *Data & Knowledge Engineering*, 20(3) :287–303.

[Smith et al., 2005] Smith, B., Kumar, A., and Bittner, T. (2005). Basic Formal Ontology for bioinformatics.

[Srivastava et al., 2014] Srivastava, S., Fang, E., Riano, L., Chitnis, R., Russell, S., and Abbeel, P. (2014). Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 639–646. ISSN : 1050-4729.

[Stark, 2015] Stark, J. (2015). Product Lifecycle Management. In Stark, J., editor, *Product Lifecycle Management (Volume 1) : 21st Century Paradigm for Product Realisation*, Decision Engineering, pages 1–29. Springer International Publishing, Cham.

[Sun et al., 2019] Sun, X., Zhang, Y., and Chen, J. (2019). RTPO : A Domain Knowledge Base for Robot Task Planning. *Electronics*, 8(10) :1105. Number : 10 Publisher : Multidisciplinary Digital Publishing Institute.

[Tenorth and Beetz, 2009] Tenorth, M. and Beetz, M. (2009). KNOWROB — knowledge processing for autonomous personal robots. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4261–4266. Meeting Name : 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems Reporter : 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems.

[Tosello et al., 2015] Tosello, E., Fan, Z., and Pagello, E. (2015). A Semantic Knowledge Base for Cognitive Robotics Manipulation. *ResearchGate*.

[Touya et al., 2012] Touya, G., Balley, S., Duchêne, C., Jaara, K., Regnauld, N., and Gould, N. (2012). Towards an ontology of spatial relations and relational constraints. In *15th ICA workshop on generalisation and multiple representation*.

[Tzafestas, 2013] Tzafestas (2013). Introduction to Mobile Robot Control | Spyros G. Tzafestas | Request PDF. Reporter : ResearchGate.

[Uschold, 1996a] Uschold, M. (1996a). Building ontologies : Towards a uni ed methodology. In *Proceedings of 16th Annual Conference of the British Computer Society Specialists Group on Expert Systems*. Citeseer.

[Uschold, 1996b] Uschold, M. (1996b). Converting an informal ontology into ontolingua : Some experiences. *TECHNICAL REPORT-UNIVERSITY OF EDINBURGH ARTIFICIAL INTELLIGENCE APPLICATIONS INSTITUTE AIAI TR*.

[Uschold, 1999] Uschold, M. (1999). A Framework for Understanding and Classifying Ontology Applications.

[Vahrenkamp et al., 2016] Vahrenkamp, N., Westkamp, L., Yamanobe, N., Aksoy, E. E., and Asfour, T. (2016). Part-based grasp planning for familiar objects. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 919–925. ISSN : 2164-0580.

[Vassev and Hinchey, 2012] Vassev, E. and Hinchey, M. (2012). Knowledge Representation for Cognitive Robotic Systems. In *2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pages 156–163.

[Vere, 1983] Vere, S. A. (1983). Planning in Time : Windows and Durations for Activities and Goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(3) :246–267. Conference Name : IEEE Transactions on Pattern Analysis and Machine Intelligence.

[Vidal and Geffner, 2006] Vidal, V. and Geffner, H. (2006). Branching and pruning : An optimal temporal POCL planner based on constraint programming. *Artificial Intelligence*, 170(3) :298–335.

[Voronoi, 1908] Voronoi (1908). Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire. Sur quelques propriétés des formes quadratiques positives parfaites. *Journal für die reine und angewandte Mathematik (Crelle's Journal)*, 1908(133). Number : 133 Reporter : Journal für die reine und angewandte Mathematik (Crelle's Journal).

[Vásquez et al., 2017] Vásquez, A., Dapogny, A., Bailly, K., and Perdereau, V. (2017). Sequential recognition of in-hand object shape using a collection of neural forests. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3081–3086. Meeting Name : 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Reporter : 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

[Wegner, 1996] Wegner, P. (1996). Interoperability. *ACM Computing Surveys*, 28(1) :285–287.

[Weld, 1994] Weld, D. S. (1994). An Introduction to Least Commitment Planning. *AI Magazine*, 15(4) :27–27. Number : 4.

[Weser et al., 2010] Weser, M., Off, D., and Zhang, J. (2010). HTN robot planning in partially observable dynamic environments. In *2010 IEEE International Conference on Robotics and Automation*, pages 1505–1510. ISSN : 1050-4729.

[Wilmarth et al., 1999] Wilmarth, S., Amato, N., and Stiller, P. (1999). MAPRM : a probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 2, pages 1024–1031 vol.2. ISSN : 1050-4729.

[Winter, 2006] Winter, Maureen Donnelly, S. T. B. (2006). Ontology and Semantic Interoperability. In *Large-scale 3D Data Integration*. CRC Press. Num Pages : 22.

[Wolfe et al., 2010] Wolfe, J., Marthi, B., and Russell, S. (2010). Combined task and motion planning for mobile manipulation. In *Proceedings of the Twentieth International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'10, pages 254–257, Toronto, Ontario, Canada. AAAI Press.

[Xiao and Zhang, 2017] Xiao, L. and Zhang, S. (2017). Analysis and optimization of drum washing machine vibration isolation system based on rigid-flexible virtual prototype model. *Journal of Vibroengineering*, 19(3) :1653–1664. Number : 3 Publisher : JVE International Ltd.

[Xinyu et al., 2019] Xinyu, W., Xiaojuan, L., Yong, G., Jiadong, S., and Rui, W. (2019). Bidirectional Potential Guided RRT* for Motion Planning. *IEEE Access*, 7 :95046–95057. Conference Name : IEEE Access.

[Xu, 2017] Xu, D. (2017). *Contribution to the elaboration of a decision support system based on modular ontologies for ecological labelling*. PhD thesis.

[Zender et al., 2008] Zender, H., Martínez Mozos, O., Jensfelt, P., Kruijff, G. J. M., and Burgard, W. (2008). Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56(6) :493–502. Number : 6 Reporter : Robotics and Autonomous Systems.

[Zhang and Shah, 2016] Zhang, C. and Shah, J. A. (2016). Co-optimizing task and motion planning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4750–4756. ISSN : 2153-0866.

[Zhao, 2019] Zhao, Y. (2019). *An ontology-based approach towards coupling task and path planning for the simulation of manipulation tasks*. These de doctorat, Toulouse, INPT.

[Zhao et al., 2018] Zhao, Y., Fillatreau, P., Karray, M. H., and Archimède, B. (2018). An ontology-based approach towards coupling task and path planning for the simulation of manipulation tasks. In *2018 IEEE/ACS 15th International Conference on Computer*

Systems and Applications (AICCSA), pages 1–8. Meeting Name : 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA) Reporter : 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA).

[Zhao et al., 2023] Zhao, Y., Sarkar, A., Elmhadhbi, L., Karray, M. H., Fillatreau, P., and Archimede, B. (2023). An ontology of 3d environment where a simulated manipulation task takes place (envon).

[Zou and Zhu, 2003] Zou, X.-y. and Zhu, J. (2003). Virtual local target method for avoiding local minimum in potential field based robot navigation. *Journal of Zhejiang University-SCIENCE A*, 4(3) :264–269.