Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace

Présentée et soutenue par :
Sébastien PIEDADE

le mercredi 5 mai 2021

Titre :

Synthèse de plans conditionnels pour la décision dans l'incertain

---

École doctorale et discipline ou spécialité :
EDSYS : Informatique & Robotique

Unité de recherche :
Équipe d'accueil ISAE-ONERA ACDC

Directeur(s) de Thèse :

M. Charles LESIRE-CABANIOLS (directeur de thèse)
M. Guillaume INFANTES (co-directeur de thèse)

Jury :
M. François CHARPILLET Directeur de recherche INRIA Nancy - Président
M. Guillaume INFANTES Ingénieur Jolibrain - Co-directeur de thèse
M. Charles LESIRE-CABANIOLS Maître de recherche ONERA - Directeur de thèse
M. Andrea ORLANDINI Chercheur Institute of Cognitive Sciences and Technologies Rome - Rapporteur
M. Damien PELLIER Maître de conférences Université Grenoble Alpes - Rapporteur
Mme Sylvie THIÉBAUX Professeure Australian National University - Examinatrice

Remerciements

Dans un premier temps, j'aimerais remercier mes encadrants Charles Lesire et Guillaume Infantes pour m'avoir permis de réaliser cette thèse. J'aimerais particulièrement souligner l'investissement de Charles, dans les bons comme dans les mauvais moments. Deux autres personnes ont activement contribué à la réalisation de cette thèse : Alban Grastien et Alexandre Albore. Alban, je te remercie pour ton aide à la compréhension et l'utilisation de CPCES, et tes précieuses idées ayant permis le développement des différentes versions du planificateur présenté dans cette thèse. Je te remercie également pour ton accueil chaleureux à Canberra pendant cette inoubliable mobilité. Alexandre, je te remercie d'avoir rejoint mon équipe d'encadrement et de t'être autant impliqué dans mon travail. Tes conseils et le temps que tu m'a accordé ont grandement contribué au résultat final de cette thèse.

Je remercie les membres du jury pour le temps qu'ils m'ont accordé et leurs précieux conseils et remarques : tout d'abord le président du jury François Charpillet, les rapporteurs Andrea Orlandini et Damien Pellier, et enfin Sylvie Thiébaux. Je remercie particulièrement Sylvie de m'avoir invité à venir travailler avec Alban à Canberra dans le cadre de ma mobilité.

Une thèse n'est pas qu'une aventure scientifique mais aussi une aventure humaine, c'est pour cela que je tiens à remercier tous les collègues doctorants, post-docs et stagiaires de l'ONERA pour toutes ces pauses café et ces soirées mémorables. Valentin, je te remercie doublement d'avoir partagé mon bureau ainsi que toutes ces aventures épiques.

Un grand merci aux poussins Nico et Floflo pour tous ces bons moments partagés depuis notre enfance, particulièrement pendant nos vacances estivales idéales pour s'aérer l'esprit pendant ce travail de thèse et se remplir la panse avec excès.

Je remercie évidemment Chaton qui m'a soutenu et motivé pendant tout le processus de rédaction de ce manuscrit, et continue de me supporter au quotidien.

Pour finir, je tiens à remercier l'ensemble de ma famille pour leur encouragement constant pendant toutes mes années d'études.

Table des matières

Liste des acronymes

DNF_{ct} Contingent Disjunctive Normal Form.

BCTCF Backtracking based ConTingent planner using a ConFormant planner.
BDD Binary Decision Diagram.

CLG Closed-Loop Greedy planner.
Conformant-FF Conformant Fast-Forward.
Contingent-FF Contingent Fast-Forward.
CPCES Conformant Planner via Counter Example and Sampling.
CPOR Contingent Planner using Online Replanning.
CTCF ConTingent planner using a ConFormant planner.
CTCF+ ConTingent planner using a ConFormant planner +.
CTCFE ConTingent planner using a ConFormant planner with the Earliest counter-example.
CTCFL ConTingent planner using a ConFormant planner for large problems.

DNF Disjunctive Normal Form.

FF Fast-Forward.
FOND Fully-Observable Non-deterministic problem.

HSP Heuristic Search Planner.

MBP Model Based Planner.
MDP Markov Decision Process.
MPSR Multi-Path, Sampling, Replanner.

PDDL Planning Domain Definition Language.
POMDP Partially Observable Markov Decision Process.

SAT Satisfiability problem.
SDR Sample, Determinize, Replan.
SMT Satisfiability Modulo Theories.
ssi Si et seulement si.
STRIPS STanford Research Institute Problem Solver.

Notations

A Ensemble des actions.

F Ensemble de fluents.

P Problème de planification classique.

S Ensemble d'états possibles.

T Fonction de transition.

Ω Ensemble des observations.

Σ Système de transition d'états.

γ État de contre-exemple.

$\mathcal{B}$ État de croyance.

$\mathcal{P}_\Omega$ Problème de planification contingente.

$\mathcal{P}$ Problème de planification conformante.

$\nu(p)$ Valeur de vérité de la proposition p.

π Plan.

σ Observation.

ε Plan vide.

φ Contrainte sur le résultat d'une observation.

a Action.

g But d'un problème de planification.

p Proposition.

s_0 État initial.

s État.

Introduction

De nos jours, les missions d'exploration ou encore de sauvetage sont de plus en plus confiées à des robots autonomes. Lors de ces missions, les robots sont confrontés à des environnements complexes et incertains, ce qui nécessite la planification des différentes tâches devant être accomplies pour mener à bien la mission. La planification est une discipline qui consiste à choisir selon un critère donné quelles tâches doivent être réalisées parmi un ensemble de tâches et dans quel ordre elles doivent être effectuées afin d'atteindre un but précis à partir des informations initiales de l'environnement dont nous disposons.

Dans ce type de mission, l'environnement est décrit à l'aide de nombreuses variables et contraintes, ce qui rend le problème de planification des tâches très complexe à réaliser par un humain, qui mettrait plusieurs heures à résoudre les problèmes les plus simples ne comportant aucune incertitude. Pour résoudre ce type de problème de manière rapide et efficace, il est donc nécessaire de réaliser une planification des tâches automatique grâce à une machine. La planification classique peut être considérée comme le socle commun des différentes approches de planification automatique, celle-ci décrivant une grande partie des formalismes utiles au développement des autres cadres de planification. La planification classique consiste à générer un plan sous forme d'une séquence de tâches menant de l'état initial d'un problème à son état but. Ce type de plan est adapté aux problèmes déterministes mais il échoue lorsqu'un aléa apparaît lors de son exécution, celui-ci étant incapable de s'adapter aux incertitudes du problème. Plusieurs méthodes de planification peuvent être choisies pour traiter l'incertitude selon la manière dont celle-ci est modélisée. On peut prendre deux exemples de problèmes dans lesquels l'incertitude est modélisée de deux manières différentes.

Considérons tout d'abord un premier problème de planification pour satellite d'observation de la Terre devant réaliser des acquisitions d'image dans plusieurs endroits de la Terre. Chaque lieu ne peut être photographié que pendant une fenêtre de temps définie et dans ce problème l'incertitude concerne le pourcentage de couverture nuageuse d'un lieu d'acquisition. Cette incertitude est modélisée de manière probabiliste, chaque lieu à visiter par le satellite comporte un pourcentage moyen de couverture nuageuse et nous avons décidé de modéliser l'incertitude comme une gaussienne centrée sur cette valeur. Le but de ce problème est de maximiser la qualité des images acquises par le satellite, cette qualité étant influencée par le taux de couverture nuageuse incertain de chaque lieu d'acquisition.

Afin de résoudre ce problème, il existe des méthodes de replanification en ligne (Kuter et collab., 2008) dans lesquelles un premier plan est calculé sans prendre en compte les incertitudes et un plan est recalculé en cours de mission si le plan calculé hors ligne (avant le départ de la mission) échoue. Cependant, dans ce type de mission, il est généralement préférable d'opter pour des méthodes permettant de traiter l'incertitude avant le départ de la mission afin d'éviter des calculs de plan coûteux supplémentaires pendant la mission. C'est particulièrement le cas pour les exemples de missions traités dans ce manuscrit, dans lesquels les décisions en ligne doivent être réalisées rapidement. La planification contingente (Albore et collab., 2009) est une de ces méthodes hors-ligne permettant de calculer un plan conditionnel traitant l'incertitude du problème tout en laissant la possibilité d'effectuer des décisions en ligne rapides et conditionnées par des observations de l'environnement.

La figure 1 illustre un plan conditionnel d'une mission de satellite d'observation dans laquelle une observation de la couverture nuageuse de Milan est décidée et un branchement alternatif est intégré, permettant de choisir un autre chemin d'acquisitions si la présence de nuage observée à Milan est bien plus importante que prévue.

Afin d'évaluer l'intérêt de la planification contingente, nous avons décidé de réaliser un travail préliminaire consistant à résoudre ce problème de satellite d'observation en développant une première approche de planification contingente. Dans un premier temps, l'idée de cette approche est de calculer

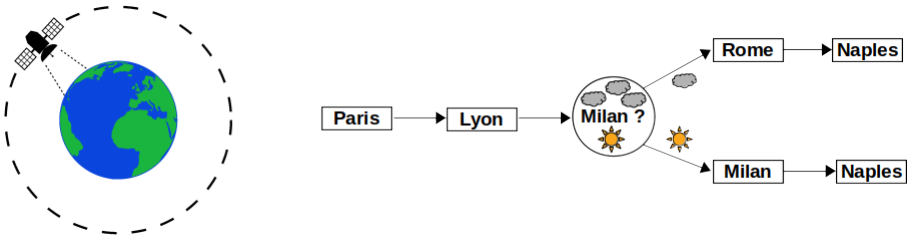

FIGURE 1 – Plan conditionnel d'une mission de satellite d'observation

un premier plan ne prenant pas en compte l'incertitude du problème, c'est à dire évaluant la qualité potentielle de chaque acquisition seulement par son pourcentage moyen de couverture nuageuse et ignorant l'incertitude gaussienne centrée sur ce pourcentage.

Ce plan est ensuite parcouru afin de vérifier pour chaque acquisition s'il est possible d'ajouter des branchements permettant d'améliorer la qualité du plan dans le cas où le pourcentage de couverture nuageuse est le pire possible.

Dans cette approche, un branchement est ajouté au premier plan si le gain maximal, minimal, ou espéré de la branche alternative est supérieur à celui du premier plan compte tenu de l'incertitude sur la couverture nuageuse.

Ce travail préliminaire nous a permis de nous rendre compte de l'intérêt de la planification contingente de par la lisibilité et la compacité des plans générés. Plus important encore, ce travail nous a permis de nous rendre compte de l'importance de calculer un plan contenant un nombre d'observations limité afin de réduire la taille du plan et son temps de calcul, l'ajout d'une observation signifiant le calcul coûteux d'une nouvelle partie du plan. Ce travail a été publié dans le workshop Planning and Learning de la conférence IJCAI 2018 (Piedade et collab., 2018).

Un autre exemple de problème contenant de l'incertitude est le problème de recherche et sauvetage dans lequel un robot doit rechercher et secourir des victimes dans un environnement incertain et comportant des obstacles. Dans ce type de problème, l'incertitude porte souvent sur la localisation inconnue des obstacles et des objectifs. Dans ce contexte, la précision du modèle de l'environnement disponible ne permet pas toujours de pouvoir quantifier les incertitudes de localisation sous forme de probabilités. En revanche, il est plus facile de la modéliser comme un ensemble fini de scénarios possibles, c'est à dire un ensemble d'états initiaux possibles de l'environnement.

La figure 2 est un exemple d'ensemble d'états initiaux possibles pour une mission de recherche et sauvetage. Ce type de mission consiste par exemple à localiser une personne dans les décombres d'un immeuble effondré à l'aide d'un robot mobile. Dans la mission illustrée par la figure 2, la position d'un bâtiment effondré et la position d'une victime sont incertaines, laissant quatre scénarios possibles.

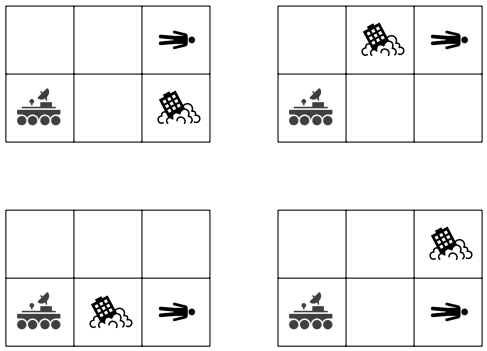

FIGURE 2 – Exemple d'ensemble d'états initiaux possibles

Les méthodes de planification symboliques sont les plus utilisées pour traiter l'incertitude modélisée de cette manière. Parmi ces méthodes, on peut compter la planification conformante (Albore et collab., 2011), dans laquelle on tente de calculer un plan sous forme de séquence d'actions valide pour chaque état initial possible sans réaliser d'observation, et la planification contingente. La planification conformante

est particulièrement intéressante compte tenu de sa complexité plus faible que celle de la planification contingente, néanmoins, certains problèmes ne sont pas traitables sans réaliser d'observation. Comme nous l'avons vu dans le travail préliminaire développé dans le contexte du satellite d'observation, la planification contingente est particulièrement adaptée aux missions de robotique autonome de par la facilité d'embarquabilité des plans contingents et leur compacité mais elle présente néanmoins une complexité élevée. Une des intuitions inspirées par ce travail préliminaire est que le temps de calcul d'un plan contingent peut être réduit en limitant le nombre d'observations. De plus, la réalisation d'une observation en cours de mission peut être coûteuse pour l'agent devant la réaliser. Parmi les différents coûts imaginables que peuvent engendrer les observations, on peut citer un long temps de chauffe ou d'allumage pour un capteur, ou bien devoir se déplacer dans un lieu éloigné de l'objectif de la mission afin de réaliser l'observation. Au vu du coût potentiel que représente une observation, et des intuitions inspirées par notre travail préliminaire, nous avons décidé de développer un planificateur contingent limitant le nombre d'observations incorporées au plan grâce à l'utilisation d'un planificateur conformant.

Cette thèse consiste donc à développer un planificateur contingent traitant des missions comportant de l'incertitude sous forme d'un ensemble d'états initiaux possibles en limitant le nombre d'observations par l'utilisation d'un planificateur conformant pour calculer chaque branche du plan.

Dans la première partie de ce manuscrit, nous allons introduire les principales notions et cadres de modélisation existants en planification classique et planification sous incertitude.

La démarche de cette thèse consiste tout d'abord à développer un planificateur contingent limitant le nombre d'observations (Chapitre 5). Pour cela l'idée est de tenter de résoudre le problème en calculant un plan conformant par l'appel d'un planificateur conformant. Par la suite, l'objectif est de déterminer quelle observation rajouter au plan si le planificateur conformant échoue à calculer un plan conformant. Cette observation est déterminée en utilisant les informations retournées par le planificateur conformant, à savoir un plan conformant pour seulement une partie des états initiaux et un état contre-exemple pour lequel ce plan n'est pas valide. L'idée est ensuite de calculer chaque branche du plan menant et partant de cette observation en relançant le processus contingent. Cette étude a été publiée dans (Piedade et collab., 2020).

Les limites de cette première démarche nous ont tout d'abord poussé à exploiter les caractéristiques du planificateur conformant choisi afin de réduire le temps de calcul du plan contingent (Chapitre 6). Pour cela, l'idée a été de modifier directement le planificateur conformant afin de sélectionner le contre-exemple faisant échouer le plan défaillant au plus tôt. Cette modification permet d'obtenir un début de plan conformant pouvant être utilisé pour éviter le calcul d'un plan menant à l'observation choisie, et donc réduire le temps de calcul.

Afin de pouvoir traiter un plus grand nombre de problèmes, l'objectif a ensuite été de modifier les contraintes de but du calcul du plan menant à l'observation en ne considérant plus le but comme un état de croyance mais plutôt comme un ensemble de propositions représentant les préconditions de l'observation à atteindre. (Chapitre 7).

Afin de traiter des problèmes de plus grande taille, l'objectif a ensuite été de ne plus représenter les états de croyance de manière explicite, mais plutôt comme l'association de l'état de croyance initial et de la séquence d'actions/observations menant à l'état de croyance courant. L'objectif à ensuite été de modifier les communications avec le planificateur conformant afin que celui-ci puisse prendre en entrée cette nouvelle représentation de l'état de croyance (Chapitre 8).

Afin de rendre complète la dernière version du planificateur contingent développé (Chapitre 8), nous avons eu l'idée d'intégrer un backtracking sur les observations candidates, ainsi qu'une fonction permettant de trouver une observation discriminant directement l'état de contre-exemple des autres états de l'état de croyance courant sans se servir des informations retournées par le planificateur conformant en cas d'échec (Chapitre 9).

Finalement, dans le chapitre 10 de ce manuscrit, nous allons élaborer une conclusion de ce travail de thèse et nous discuterons des perspectives pouvant être envisagées dans le chapitre 11.

Première partie

État de l'art

Chapitre 1

Introduction

La planification est une discipline qui consiste à choisir selon un critère donné quelles actions doivent être réalisées parmi un ensemble d'actions et dans quel ordre elles doivent être effectuées afin d'atteindre un but précis. Plus le problème à résoudre est complexe et plus la tâche de planification est coûteuse en ressources et temps de calcul. Ce coût est une des motivations de la planification automatique, qui a pour but de décharger l'humain de la tâche de planification au profit d'un système de calcul permettant de traiter des problèmes complexes automatiquement en un temps largement réduit.

Exemple: Problème de planification de robotique autonome

Dans ce manuscrit, nous utiliserons un exemple de problème de robotique autonome afin d'illustrer les différentes notions et algorithmes présentés. Ce problème est une simplification du problème de recherche et sauvetage présenté dans l'introduction de ce manuscrit. Dans ce problème, un robot autonome doit récolter un ou plusieurs rochers en évitant des obstacles statiques présents dans l'environnement. L'environnement est modélisé comme une grille 2D dans laquelle chaque case peut soit être vide, soit comporter un robot, un rocher ou un obstacle. Les actions du robot sont les déplacements, la collecte de rocher et l'observation d'une case adjacente au robot. Le robot ne peut se déplacer que dans une case adjacente et ne peut récolter le rocher qu'en étant dans la même case.

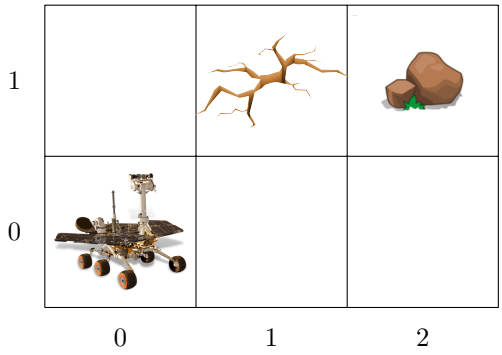

FIGURE 1.1 – Représentation d'un état de l'environnement

La figure 1.1 illustre un exemple très simple dans lequel le robot doit récolter un rocher dans un environnement restreint ne comportant qu'un seul obstacle. Le robot en position (0,0) est représenté par un robot d'exploration, l'obstacle en (1,1) est représenté par un trou dans le sol et enfin le rocher à récolter se trouve dans la case (2,1) de la grille.

Dans le domaine de la planification, plusieurs notions sont importantes, parmi lesquelles on peut citer les notions d'observabilité et de déterminisme. La notion d'observabilité traite de l'ensemble de variables pour lesquelles on peut connaître la valeur de vérité. On dit qu'un système a une **observabilité totale** si on peut connaître la valeur de vérité de toutes ses variables à chaque instant. Un système a une

observabilité partielle si on ne peut connaître la valeur de vérité que d'une partie seulement de ses variables à chaque instant. Enfin, un système a une **observabilité nulle** si on ne peut connaître la valeur de vérité d'aucune de ses variables à n'importe quel instant. Le déterminisme d'une action évoque les connaissances à priori de ses effets. Une action déterministe est une action dont les effets sont invariants et connus à chaque instant. Par contradiction, une action non déterministe désigne une action dont les effets peuvent être variants ou inconnus selon l'instant et l'état du monde dans lequel l'action est effectuée.
La planification automatique peut se découper en plusieurs cadres de planification permettant de traiter différents types de problème. Nous avons vu dans l'introduction de ce manuscrit que l'incertitude d'une mission peut être modélisée de différentes manières. Dans cette partie, nous allons nous intéresser aux différents cadres de planification existants dans la littérature pouvant traiter les problèmes comportant ces différents types d'incertitude. Dans un premier temps nous allons détailler le formalisme et les techniques employées en planification classique pour résoudre un problème de planification sans incertitude (Chapitre 2). On peut noter que la planification classique en elle-même ne permet pas le traitement des problèmes sous incertitude, mais introduit des notions importantes et indispensables servant de socle à la planification sous incertitude. Dans un deuxième temps, nous nous intéresserons plus particulièrement à la planification dans l'incertain (Chapitre 3) en commençant par détailler les notions et les méthodes employées en planification probabiliste permettant de résoudre les problèmes dans lesquels l'incertitude est modélisée sous forme de probabilité (Section 3.1). Nous allons ensuite nous attarder plus longuement sur les notions et méthodes de planification symbolique permettant de traiter les problèmes dans lesquels l'incertitude est modélisée comme un ensemble d'états initiaux possibles (Section 3.2). Nous mettrons plus particulièrement l'accent sur la planification conformante (Sous-section 3.2.1) et la planification contingente (Sous-section 3.2.2), celles-ci étant au centre de notre approche, et enfin nous conclurons sur les différents cadres de planification exposés dans cet état de l'art.

Chapitre 2

Planification classique

Dans un premier temps nous allons détailler une représentation inspirée du formalisme STRIPS (STanford Research Institute Problem Solver) (Fikes et Nilsson, 1971), qui tire son nom du premier planificateur dédié à la robotique. Les définitions 1 à 6 détaillées dans ce chapitre sont adaptées du livre de Ghallab et collab. (2004).

La planification classique est le domaine de la planification dans lequel l'environnement est entièrement observable et dans lequel les actions sont déterministes.

Dans ce manuscrit, comme dans le formalisme STRIPS, nous représentons les états de manière factorisée : l'ensemble des composantes du monde sont représentées par un ensemble de $fluents$ F prenant des valeurs booléènnes. Si on prend comme exemple l'environnement décrit dans la figure 1.1 on peut représenter par (at_rock $cell21$) le fait que le rocher se trouve en position (2,1).

À partir de ces $fluents$ décrivant l'environnement, on peut définir la notion d'état du monde.

Définition 1: États

Un **état** s est un ensemble de fluents de F avec $s \subseteq F$. On nomme S l'ensemble fini des états possibles.

Dans cette thèse, nous nous plaçons sous l'hypothèse de *Closed World Assumption*, ce qui signifie que les fluents qui ne sont pas définis comme étant vrais dans un état, sont faux. Un fluent f sera donc vrai dans un état s si et seulement si $f \in s$.

Définition 2: Actions

On définit une action comme un couple $a = (Pre(a), Eff(a))$ avec :
- $Pre(a)$ un ensemble de préconditions
- $Eff(a)$ un ensemble d'effets.

Chaque effet $e \in Eff$ est défini comme un couple $Add(e) \subseteq F$, $Del(e) \subseteq F$ correspondants respectivement à un ensemble d'additions et un ensemble d'effacements.

Une action a est applicable dans un état s ssi $Pre(a) \subseteq s$. Le résultat de l'application de a dans s est un nouvel état défini par la fonction de transition T :

$$T(s, a) = (s - Del(a)) \cup Add(a) \tag{2.1}$$

On peut définir le domaine de planification comme l'ensemble des propriétés de l'environnement partagées par toutes les missions effectuées dans un même environnement.

Définition 3: Domaine

Un domaine de planification classique dans F est un système de transition d'états $\Sigma = (S, A, T)$ tel que :
- S est l'ensemble des états possibles avec $S \subseteq 2^F$
- A est l'ensemble des actions
- T est la fonction de transition avec $T(s,a) = (s - Del(a)) \cup Add(a)$ si $a \in A$ est applicable dans $s \in S$, dans le cas contraire $T(s,a)$ est indéfinie
- Si $s \in S$ alors pour chaque action $a \in A$ applicable à s, $T(s,a) \in S$

On définit les caractéristiques spécifiques à chaque mission comme un problème de planification comportant un domaine, un état initial du monde, et un but.

Définition 4: Problème

Un problème de planification classique est un triplet $P = (\Sigma, s_0, g)$ dans lequel :
- Σ est un domaine de planification classique
- $s_0 \in S$ est l'état initial
- g est un ensemble de fluents dans F représentant le but

On note S_g l'ensemble des états $s \in S$ satisfaisant g. Le but de la planification classique est de calculer une séquence d'actions (un plan) menant de l'état initial du problème à un état but $s \in S_g$.

Définition 5: Plan

On définit une séquence d'actions $\pi =< a_1, ..., a_k >$ avec $k \geq 0$. La taille d'une séquence d'actions est $|\pi| = k$, et correspond au nombre d'actions de la séquence. On note ε la séquence d'actions vide.
On peut généraliser l'application d'une action dans un état pour définir l'application d'une séquence d'action π dans un état s. L'état produit par l'application de π à un état s est l'état produit par l'application des actions de π dans l'ordre donné. On peut étendre la fonction de transition d'état T de la manière suivante :

$$T(s,\pi) = \begin{cases} s & si\ k = 0 \\ T(T(s,a_1), < a2, ..., a_k >) & si\ k > 0 \text{ et } a_1 \text{ est applicable à } s \\ \text{indéfinie} & \text{autrement} \end{cases} \quad (2.2)$$

On dit qu'une séquence d'action $\pi =< a_1, ..., a_k >$ est valide ssi $\forall a_i \in \pi$, $Pre(a_i) \subseteq T(s, < a_1, ..., a_{i-1} >)$. Un plan est une séquence d'action valide.

Définition 6: Solution

Soit $P = (\Sigma, s_0, g)$ un problème de planification. Un plan π est une solution pour P si $g \subseteq T(s_0, \pi)$.

Exemple: Représentation de notre exemple en planification classique

Considérons notre exemple de mission de récolte de rocher présenté dans la figure 1.1 comme état initial du système.
On peut représenter l'ensemble F des fluents représentant les variables possibles décrivant l'environnement de la manière suivante :
- Les positions possibles du robot dans la grille :
  ($at\_robot\ cell00$) ($at\_robot\ cell10$) ($at\_robot\ cell20$)
  ($at\_robot\ cell01$) ($at\_robot\ cell11$) ($at\_robot\ cell21$)
- Les positions possibles du rocher dans la grille :
  ($at\_rock\ cell00$) ($at\_rock\ cell10$) ($at\_rock\ cell20$)

(*at_rock cell*01) (*at_rock cell*11) (*at_rock cell*21)

— Les positions possibles de l'obstacle :
(*is_obstacle cell*00) (*is_obstacle cell*10) (*is_obstacle cell*20)
(*is_obstacle cell*01) (*is_obstacle cell*11) (*is_obstacle cell*21)

— Les possibilités d'adjacence des cases :
(*adjacent cell*00 *cell*01) (*adjacent cell*00 *cell*10) (*adjacent cell*10 *cell*11)
(*adjacent cell*11 *cell*21) ... (*adjacent cell*21 *cell*20)

— Le fait que le rocher ait été collecté :
(*collected*)

L'état initial s_0 correspond aux fluents suivants :
{(*is_obstacle cell*11) (*at_robot cell*00) (*at_rock cell*21) (*adjacent cell*00 *cell*01) (*adjacent cell*00 *cell*10) (*adjacent cell*10 *cell*00) (*adjacent cell*10 *cell*11) (*adjacent cell*10 *cell*20) (*adjacent cell*20 *cell*21) (*adjacent cell*20 *cell*10) (*adjacent cell*01 *cell*00) (*adjacent cell*01 *cell*11) (*adjacent cell*11 *cell*01) (*adjacent cell*11 *cell*21) (*adjacent cell*11 *cell*10) (*adjacent cell*21 *cell*11) (*adjacent cell*21 *cell*20)}

L'ensemble des actions A est composé de deux actions, *move* et *collect* correspondant aux déplacements du robot et à la collecte du rocher. L'application de l'action *move* dans un état s supprime le fluent de position du robot dans s pour le remplacer par le fluent représentant sa nouvelle position. La fonction de transition $T(s_0,< move\ cell10 >)$ représentant l'application de l'action de déplacement du robot en case (1,0) est la suivante :

$T(s_0,< move\ cell10 >) = \{(is\_obstacle\ cell11)\ (at\_robot\ cell10)\ (at\_rock\ cell21)\ (adjacent\ cell00\ cell01)\ ...\ (adjacent\ cell21\ cell20)\}$

On peut représenter le but du problème g comme un fluent (*collected*) représentant le fait que le rocher ait été collecté.

Un des plans solutions de la mission peut être la séquence d'actions $\pi =< move\ cell10, move\ cell20, move\ cell21, collect >$ dont l'application sur l'état initial est illustré par la figure 2.1

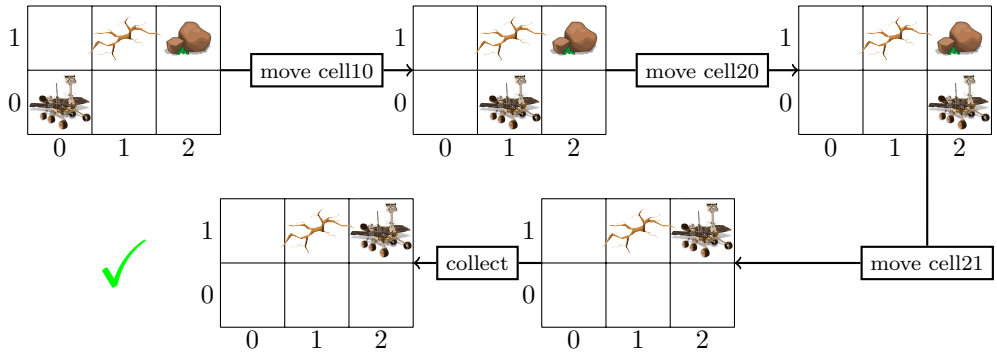

FIGURE 2.1 – Illustration de l'application de π

2.1 PDDL

Planning Domain Definition Language (PDDL) (McDermott et collab., 1998) est un langage de spécification de problèmes de planification dont le but est de représenter les domaines et problèmes de planification de manière santardisée afin d'être utilisé comme langage d'entrée par de nombreux planificateurs.

En PDDL, la description de l'environnement de planification est séparée en deux fichiers distincts : un fichier de domaine contenant les variables et descriptions partagées par tous les problèmes d'un même environnement, et un fichier problème spécifiant les caractéristiques de la mission. Le fichier de domaine va réunir les spécifications des différents types d'objets représentant le monde, les différents prédicats modélisant les différents fluents, et les actions pouvant être réalisées. Le fichier de problème regroupe les instances d'objets présents dans l'environnement, les différents prédicats représentant l'état initial de l'environnement, et le but du problème sous forme de conjonction ou disjonction de prédicats.

Exemple: Implémentation PDDL

Afin de décrire les caractéristiques et syntaxes de PDDL, on peut construire une modélisation PDDL de notre exemple.
On s'intéresse tout d'abord au domaine PDDL. Dans celui-ci, nous pouvons modéliser les types d'objets de notre mission de collecte comme cela :

```
(:types robot position rock)
```

Les différents fluents du monde sont représentés par les prédicats suivants :

```
(:predicates (at_robot ?x - robot ?y - position)
       (at_rock  ?y - position)
       (is_obstacle ?y - position)
       (collected ?x - robot ?r -rock)
       (adjacent ?y - position ?z - position)
)
```

Une action de déplacement du robot peut être modélisée de la manière suivante :

```
(:action move
:parameters (?x - robot ?y - position ?z - position)
:precondition (and (adjacent ?y ?z) (not(is_obstacle ?z)) (at_robot ?x ?
    y))
:effect (and (not (at_robot ?x ?y)) (at_robot ?x ?z)))
)
```

Les paramètres de l'action sont les prédicats représentant le robot devant se déplacer, ainsi que les positions de départ et d'arrivée du robot. Les préconditions de l'action sont les prédicats nécessaires à sa réalisation, à savoir que le robot doit être à sa position de départ, les deux positions de départ et d'arrivée doivent être adjacentes, et la position d'arrivée ne doit pas contenir d'obstacle. Les effets de l'action sont le déplacement du robot de la position de départ à la position d'arrivée.
Le fichier problème PDDL permet d'instancier les objets du monde de la manière suivante :

```
(:objects
    rob - robot
    cell00 cell10 cell20 cell01 cell11 cell21 - position
  roc - rock
)
```

L'ensemble des prédicats permettant d'instancier l'état initial de l'exemple est spécifié dans une balise (: *init* :

```
(:init
(at_robot rob cell00)
(at_rock roc cell21)
(is_obstacle cell11)
(adjacent cell00 cell01) (adjacent cell00 cell10)
(adjacent cell10 cell00) (adjacent cell10 cell11)
(adjacent cell10 cell20) (adjacent cell20 cell21)
(adjacent cell20 cell10) (adjacent cell01 cell00)
(adjacent cell01 cell11) (adjacent cell11 cell01)
(adjacent cell11 cell21) (adjacent cell11 cell10)
(adjacent cell21 cell11) (adjacent cell21 cell20)
)
```

Enfin le but de l'exemple étant la collecte du rocher par le robot, voici comment le modéliser en PDDL :

```
(:goal
(collected rob roc)
)
```

2.2 Classes de complexité

L'évaluation de la complexité de calcul des algorithmes de planification repose sur la notion de classes de complexité. Une classe de complexité est un ensemble de problèmes de calcul reliés par la quantité de ressources nécessaire à leur résolution. Dans la plupart des cas, les ressources évaluées sont le temps et l'espace mémoire nécessaires à la résolution du problème.

De manière plus formelle, la définition d'une classe de complexité consiste en trois choses : un type de problème de calcul, un modèle de calcul et une ressource de calcul limitée. La plupart du temps, le problème de calcul est un problème de décision et le modèle de calcul est une machine de Turing avec des ressources de temps ou d'espace mémoire limités. Une machine de Turing (Turing, 1936) est un modèle mathématique d'une machine de calcul générale. Ce modèle est le plus utilisé en théorie de la complexité de par sa facilité d'analyse et le fait que d'après la thèse de Church-Turing (Church, 1936), s'il existe un algorithme résolvant un problème particulier, alors il existe aussi une machine de Turing résolvant ce même problème. Mécaniquement, une machine de Turing manipule des symboles contenus dans une bande infinie sur laquelle elle peut lire et écrire.

Les machines de Turing permettent une évaluation intuitive du temps et de l'espace mémoire. La complexité du temps de calcul correspond au nombre d'étapes élémentaires devant être réalisées par la machine de Turing. La complexité de l'espace mémoire correspond au nombre de cellules de sa bande devant être utilisées pour résoudre le problème.

Il existe différentes classes de complexité en temps et en espace mémoire. Les principales classes de complexité en temps sont les suivantes :

— **P** est la classe de problèmes que l'on peut résoudre par une machine de Turing déterministe dans un temps polynomial.
— **NP** est la classe de problèmes que l'on peut résoudre par une machine de Turing non déterministe dans un temps polynomial.
— **EXPTIME** est la classe de problèmes que l'on peut résoudre par une machine de Turing déterministe en temps exponentiel.
— **NEXPTIME** est la classe de problèmes que l'on peut résoudre par une machine de Turing non déterministe en temps exponentiel.

Les principales classes de complexité en espace de stockage sont les suivantes :

— **L** est la classe de problèmes que l'on peut résoudre par une machine de Turing déterministe dans un espace logarithmique.
— **NL** est la classe de problèmes que l'on peut résoudre par une machine de Turing non déterministe dans un espace logarithmique.
— **PSPACE** est la classe de problèmes que l'on peut résoudre par une machine de Turing déterministe dans un espace polynomial.
— **NPSPACE** est la classe de problèmes que l'on peut résoudre par une machine de Turing non déterministe dans un espace polynomial.
— **EXPSPACE** est la classe de problèmes que l'on peut résoudre par une machine de Turing déterministe dans un espace exponentiel.
— **NEXPSPACE** est la classe de problèmes que l'on peut résoudre par une machine de Turing non déterministe dans un espace exponentiel.

Il existe une relation d'inclusion entre les classes de temps et d'espace :

$$L \subseteq NL \subseteq NC \subseteq P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME \subseteq NEXPTIME \subseteq EXPSPACE = NEXPSPACE \tag{2.3}$$

La plupart des classes peuvent être définies en utilisant le concept de réduction. Une réduction est la transformation d'un problème en un autre problème. Cette transformation introduit l'idée qu'un problème est au moins aussi difficile qu'un autre. Si on considère un problème $P1$ pouvant être résolu par un algorithme utilisé pour le problème $P2$, alors $P1$ n'est pas plus difficile que $P2$, et on dit que $P1$ se réduit à $P2$. Cette idée motive le concept qu'un problème puisse être difficile pour une classe de complexité. Un problème P est difficile pour une classe de problèmes C si chaque problème dans C peut être réduit à P. On dit alors que P est C-difficile. Si un problème P est dans C et est C-difficile, alors P est dit complet pour C, signifiant que P est le problème le plus difficile dans C. On dit que P est C-complet.

Dans son article sur la complexité de la planification STRIPS (Bylander, 1994), Bylander indique que la complexité d'un problème de planification classique est PSPACE-complet.

2.3 Principes de résolution

Afin de calculer un plan menant de l'état initial du problème à son état but, il existe plusieurs manières de considérer le problème dans lesquelles différentes méthodes peuvent être mises en place. Dans un premier temps, nous allons étudier la recherche en avant dans un espace d'états, puis nous détaillerons les principes de la recherche en avant dans un espace de graphe. Nous ne parlerons pas dans ce manuscrit des méthodes de recherche dans un espace de plan, les notions présentent dans ces méthodes n'introduisant pas les notions utilisées par la majorité des méthodes de planification conformante et contingente.

2.3.1 Recherche en avant dans un espace d'états

Une première manière de considérer le problème de recherche de plan solution est de représenter le problème comme un problème de recherche en avant dans un espace d'états. Dans cette méthode de planification classique, l'espace de recherche est un sous ensemble de l'espace d'états représenté comme un arbre dans lequel chaque nœud de la recherche correspond à un état du monde et chaque arc correspond à une transition d'état. Le plan solution correspond au chemin suivant la succession des transitions d'états menant au but du problème.

La recherche en avant dans un espace d'états consiste à démarrer de l'état initial du système et à appliquer les actions applicables dans l'état courant itérativement jusqu'à ce qu'on arrive dans un état but. L'algorithme générique de cette méthode est décrit dans l'algorithme 1.

Algorithm 1 Forward search Algorithm

```
1: s := s_0
2: π := ε
3: loop
4:    if s satisfies g then
5:       return π
6:    choose an action a applicable in s
7:    if no such action exists then
8:       return failure
9:    s = T(s, a)
10:   π :=< π, a >
```

L'originalité des différentes approches de recherche en avant dans l'espace d'états est souvent la manière dont ces approches choisissent les nœuds de recherche à développer, c'est à dire les opérateurs à appliquer dans l'état courant. Ce choix a en effet un impact majeur sur la performance de l'algorithme de recherche. Afin d'améliorer les performances de la recherche, les algorithmes font souvent appel à

une heuristique permettant d'estimer la distance au but du problème. Une heuristique est une méthode approximée permettant de calculer rapidement une solution réalisable, pour un problème d'optimisation complexe. On dit d'une heuristique qu'elle est admissible si elle ne surestime jamais le coût pour atteindre l'objectif. On peut noter que la solution retournée par une méthode heuristique n'est pas forcément optimale.

Une manière commune de dériver la fonction heuristique est de "relaxer" le problème en un problème plus simple pour lequel la solution optimale peut être calculée efficacement. Le coût optimal pour calculer ce problème simplifié peut ensuite être utilisé comme heuristique pour résoudre le problème initial. En planification classique, un problème est souvent relaxé en supprimant les effets d'effacements de propositions des actions (Hoffmann, 2001). Pour n'importe quel état initial s du problème, le coût optimal $h'(s)$ pour atteindre un but dans le problème relaxé est une borne inférieure du coût optimal $h^*(s)$ pour atteindre un but dans le problème original. Utiliser $h'(s)$ comme fonction heuristique permet d'obtenir une heuristique informative et admissible, mais c'est au prix d'une complexité NP-difficile. C'est pourquoi des solutions heuristiques sub-optimales sont généralement utilisées, celles-ci étant moins précises mais plus rapides.

L'algorithme Heuristic Search Planner (HSP) (Bonnet et Geffner, 1998) est un planificateur classique basé sur cette idée de recherche heuristique. La fonction heuristique de HSP est dérivée d'une représentation plus haut niveau des actions et des buts. L'originalité de HSP est qu'au lieu de considérer $h'(s)$ comme fonction d'heuristique, une approximation des valeurs optimales est utilisée rendant la fonction heuristique non admissible, mais très informative et nécessitant un temps de calcul réduit.

L'algorithme Fast-Forward (FF) (Hoffmann, 2001) peut être considéré comme un successeur de HSP. FF diffère de HSP dans un nombre important de détails. Premièrement, FF a une méthode d'évaluation de l'heuristique prenant en compte les interactions positives entre les faits contrairement à HSP. Deuxièmement, FF utilise un genre différent de stratégie de recherche locale, utilisant la recherche systématique pour échapper aux plateaux et minima locaux. Enfin, FF utilise une méthode d'identification du successeur d'un noeud de recherche semblant le plus susceptible de mener au but. Ces différences rendent FF bien plus performant que HSP, faisant de FF une des méthodes références en planification classique.

2.3.2 Recherche en avant dans un espace de graphe

Une autre manière de résoudre le problème de planification classique est de réaliser une recherche en avant dans un espace de graphe. Les définitions 7 à 10 présentées ici sont directement tirées du livre de Ghallab et collab. (2004). Les approches basées sur un espace de graphe reposent sur deux idées : l'analyse d'atteignabilité et l'affinage disjonctif. L'analyse d'atteignabilité répond au problème de savoir si un état est atteignable depuis un état donné. L'affinage disjonctif consiste à aborder un ou plusieurs défauts à travers une disjonction de résolveurs.

La structure de graphe de planification permet d'estimer efficacement quel ensemble de propositions est atteignable depuis l'état initial du problème et avec quelles actions.

Définition 7: Atteignabilité d'un état

Étant donné un ensemble d'actions A, un état s est atteignable depuis un état initial s_0 s'il existe une séquence d'action dans A définissant un chemin de s_0 à s.

L'analyse d'atteignabilité consiste à analyser quels états peuvent être atteints à partir de s_0, en combien d'étapes et grâce à quelle séquence d'actions. Il est possible de calculer l'atteignabilité avec un arbre d'atteignabilité, ou bien de l'approximer avec un graphe de planification. Nous ne détaillerons pas la notion d'arbre d'atteignabilité, cette notion n'étant pas utile à la compréhension des approches de planification conformante et contingente au coeur de ce manuscrit.

La relaxation de l'analyse d'atteignabilité a été introduite par le planificateur Graphplan (Blum et Furst, 1997). Graphplan est une procédure proche de l'approfondissement itératif, qui découvre une nouvelle partie de l'espace de recherche à chaque itération. Graphplan permet de définir une condition d'atteignabilité incomplète à travers un graphe de planification de taille polynomiale et pouvant être construit en un temps polynomial dans la taille du problème d'entrée.

Le principe du graphe de planification est de considérer à chaque niveau de la structure non pas des états individuels mais l'union des ensembles de propositions de plusieurs états. Le graphe de planification considère une disjonction inclusive des actions depuis un noeud à un autre qui contient tous les effets de ces actions. Chaque noeud du graphe de planification contient les propositions présentes à un instant donné.

Un graphe de planification est un graphe orienté multi-couche : les arcs sont permis uniquement d'une couche à l'autre. Les noeuds de niveau 0 correspondent à l'ensemble de propositions P_0 définissant l'état initial s_0 d'un problème de planification. Le niveau 1 contient deux couches : un niveau d'action A_1 et un niveau de propositions P_1. A_1 est l'ensemble des actions dont les préconditions sont des noeuds dans P_0. P_1 est l'union de P_0 et des ensembles d'effets positifs des actions dans A_1.

Un noeud action dans A_1 est connecté avec les arcs *préconditions* entrants depuis ses préconditions dans P_0, et avec les arcs sortants reliés à ses effets positifs et négatifs dans P_1.

Ici, un plan est une séquence d'ensembles d'actions $\Pi =< \pi_1, \pi_2, ..., \pi_k >$. Cette séquence correspond à un plan multi-couche organisé en couches correspondant à celles du graphe de planification avec $\pi_i \subseteq A_i$.

Définition 8: Indépendance des actions

Deux actions (a, b) sont indépendantes si :
- $Del(a) \cap [Pre(b) \cup Add(b)] = \emptyset$ et
- $Del(b) \cap [Pre(a) \cup Add(a)] = \emptyset$

Inversement, deux actions a et b sont dépendantes si :
- a supprime une précondition de b
- a supprime un effet positif de b
- symétriquement pour les effets négatifs de b respectivement à a : b supprime une précondition ou un effet positif de a.

Définition 9: Plan multi-couche

Un plan multi-couche est une séquence d'ensemble d'actions. Le plan multi-couche $\Pi =< \pi_1, \pi_2, ..., \pi_n >$ est une solution d'un problème (Σ, s_0, g) ssi chaque ensemble $\pi_i \in \Pi$ est indépendant et l'ensemble π_1 est applicable à s_0, π_2 est applicable à $T(s_0, \pi_1)$, etc., et $g \subseteq T(...T(T(s_0, \pi_1), \pi_2)...\pi_n)$.

Définition 10: Mutex

Deux actions a et b d'un niveau A_i sont mutex si a et b sont dépendants ou si une précondition de a est mutex avec une précondition de b. Deux propositions p et q dans P_i sont mutex si chaque action dans A_i contenant p comme effet positif est mutex avec chaque action produisant q, et il n'existe pas d'action dans A_i produisant à la fois p et q.

Graphplan développe itérativement le graphe de planification d'un niveau, puis effectue une recherche en arrière d'une solution depuis le dernier niveau de ce graphe. La boucle itérative d'expansion du graphe et la recherche se poursuivent jusqu'à ce qu'un plan soit trouvé ou qu'une condition de terminaison d'échec soit rencontrée. Graphplan ne change pas la complexité intrinsèque de la planification qui est PSPACE-complet dans la représentation de la théorie des ensembles. En revanche l'expansion du graphe de planification est réalisé en un temps polynomial, ce qui signifie que la partie coûteuse de l'algorithme est dans la recherche du graphe de planification. De plus, le besoin en mémoire de la structure de données du graphe de planification peut être un facteur limitant.

2.4 Conclusion

La planification classique est le socle des méthodes de planification automatique. Trouver la solution d'un problème de planification classique consiste à calculer un plan sous forme d'une séquence d'actions à partir de l'état initial du problème. De nombreuses méthodes de planification classique existent comme par exemple la recherche en avant dans un espace d'état (Bonnet et Geffner, 1998; Hoffmann,

2001), la recherche en avant dans un espace de plan (Ghallab et collab., 2004), ou encore la recherche en avant dans l'espace des graphes (Blum et Furst, 1997). Toutes ces méthodes ne fonctionnent que dans un environnement dans lequel l'observabilité est totale, on a une connaissance complète des variables d'états du système et les actions ont des effets déterministes. Ces hypothèses conviennent parfaitement pour des problèmes dans lesquels aucune incertitude n'entre en compte. En revanche, on peut difficilement imaginer appliquer la planification classique dans le monde réel où l'observabilité n'est souvent que partielle et les actions peuvent être non déterministes. Si l'on se place dans le contexte des missions de robotique autonome, comme une mission d'exploration de l'environnement par exemple, les hypothèses de la planification classique sont très rapidement mises à mal. Dans ce type de mission, il peut exister une incertitude environnementale pouvant avoir un impact désastreux sur l'exécution du plan généré par la planification de la mission. L'incertitude de l'environnement peut par exemple concerner les caractéristiques ou les emplacements des éléments de l'environnement, la connaissance certaine de ces informations n'étant pas toujours accessible.

Afin de traiter cette incertitude, des méthodes évoluant hors du processus de planification existent. Parmi ces méthodes, on peut citer la replanification, qui consiste à calculer un plan sans considérer l'incertitude de l'environnement, puis à replanifier en cas d'aléa lors de la mission (Kuter et collab., 2008). Cependant, le calcul de plan en ligne nécessite de monopoliser une partie de la capacité de calcul, du temps, et de l'énergie à l'agent durant la mission. Ne pas considérer les incertitudes revient en général à relaxer le problème incertain en problème de planification classique dans lequel les actions sont déterministes et les variables totalement observables. Il existe plusieurs méthodes de replanification basées sur cette relaxation, dont nous pouvons citer FF-Replan (Yoon et collab., 2007), qui se place dans un contexte probabiliste, ainsi que Robust-FF (RFF) (Teichteil-Koenigsbuch et collab., 2008).

Dans cette thèse, nous nous intéressons plutôt aux méthodes de planification capables de traiter directement l'incertitude de l'environnement durant le processus de calcul d'un plan avant la mission : les méthodes de planification dans l'incertain.

Chapitre 3

Planification dans l'incertain

L'incertitude présente dans l'environnement peut généralement être modélisée de deux manières différentes. La première modélisation possible de l'incertitude est sous forme de distributions de probabilités sur les effets des actions et les transitions d'états. La deuxième manière de modéliser l'incertitude est sous forme d'un ensemble d'états initiaux possibles, représentant les différents scénarios envisageables. Les méthodes de planification dans l'incertain peuvent être réparties selon ces deux représentations de l'incertitude. Dans un premier temps, nous nous intéressons aux méthodes de planification probabiliste traitant l'incertitude modélisée sous forme de probabilités, puis dans un deuxième temps nous nous intéresserons aux méthodes de planification symbolique permettant de traiter l'incertitude modélisée comme un ensemble d'états initiaux possibles.

3.1 Planification probabiliste

En planification probabiliste l'incertitude est modélisée par des distributions de probabilités sur les effets des actions et les transitions d'états. Deux cadres s'imposent en planification probabiliste, la planification probabiliste sous observabilité totale, et la planification probabiliste sous observabilité partielle.

3.1.1 Planification probabiliste sous observabilité totale

Dans un premier temps, nous allons définir les notions de planification probabiliste dans un contexte d'observabilité totale. Le cadre généralement utilisé et présenté dans cette section est celui des Processus de Décision Markoviens (MDPs) (Puterman, 2014). Les définitions 11 à 13 sont directement traduites du livre de Ghallab et collab. (2004).

Définition 11: Domaine stochastique

Un domaine stochastique est un système de transition d'état non déterministe avec des distributions de probabilités sur chaque transition d'état. C'est un triplet $\Sigma = (S, A, P)$ dans lequel :

- S est un ensemble fini d'états
- A est un ensemble fini d'actions
- $P_a(s'|s)$ est une distribution de probabilités, où $a \in A$, s et $s' \in S$. Pour chaque $s \in S$, s'il existe $a \in A$ et $s' \in S$ tel que $P_a(s'|s) \neq 0$, on a $\Sigma_{s' \in S} P(s, a, s') = 1$.

$P_a(s'|s)$ est la probabilité que l'exécution de l'action a dans l'état s mène à l'état s'.

Définition 12: Politique

Un plan spécifie les actions qu'un agent doit exécuter dans un état donné. Un plan peut être représenté comme une politique π, une fonction associant les états aux actions :

$$\pi : S \to A \tag{3.1}$$

Les exécutions de politique correspondent à des séquences infinies d'états appelées *historiques*.

Contrairement à la planification classique dans laquelle les buts sont des ensembles d'états à atteindre, en planification probabiliste, les buts sont généralement des fonctions d'utilité.

Afin de définir une fonction d'utilité, on utilise des critères de coûts et de récompenses.

Définition 13: Utilité

Soit $C : S \times A \to \Re$ une fonction de coût et $R : S \to \Re$ une fonction de récompense pour un système stochastique Σ. On peut définir l'utilité de réaliser une action a dans un état s comme $V(s,a) = R(s) - C(s,a)$ et l'utilité d'une politique π dans un état s comme $V(s|\pi) = R(s) - C(s,\pi(s))$. Cette définition se généralise aux historiques. Soit $h =< s_0, s_1, ... >$ un historique. L'utilité d'un historique h induit par une politique π est définie comme

$$V(h|\pi) = \sum_{i \geq 0} \gamma^i (R(s_i) - C(s_i, \pi(s_i))) \tag{3.2}$$

avec γ un facteur d'actualisation avec $0 < \gamma < 1$ permettant à l'utilité de converger.
Étant donné une fonction d'utilité, on peut définir l'utilité espérée d'une politique en prenant en compte la probabilité de l'ensemble d'historiques H induits par la politique de la manière suivante :

$$E(\pi) = \sum_{h \in H} P(h|\pi) V(h|\pi) \tag{3.3}$$

Une politique π^* est une politique optimale pour un système stochastique Σ si $E(\pi^*) \geq E(\pi)$ pour chaque politique π de Σ, c'est à dire si π^* a une utilité espérée maximale.

On définit un problème de planification comme un problème d'optimisation dans lequel la solution du problème est une politique optimale. Une des grandes différences entre la planification probabiliste et la planification classique (ainsi que la plupart des autres approches de planification dans l'incertain), est que l'on considère qu'une politique qui n'est pas optimale n'est pas une solution du problème.

Complexité

La plupart des travaux et résultats sur la complexité de la planification basés sur les Markov Decision Process (MDP) sont décrits dans l'article de Mundhenk et collab. (Mundhenk et collab., 2000). Dans cet article, nous retrouvons les différents résultats présentés dans cette sous-section ainsi que bien d'autres résultats dont nous ne parlerons pas ici. Parmi les évaluations de complexité possibles, le problème d'existence d'une politique est celui qui se rapproche le plus de notre problèmatique. Le problème d'existence d'une politique stationnaire, c'est à dire l'assignation d'une action à un état de manière indépendante du temps pour les MDP est P-difficile et dans NP. Lorsque la politique est dépendante d'un historique ou du temps, le problèmes d'existence de politique pour les MDP compressés sont PSPACE-complet.

Principes de résolution

Dans cette section on s'intéresse à deux algorithmes de planification permettant de résoudre des problèmes basés sur les MDP, à savoir l'itération de valeur et l'itération de politique (Kolobov, 2012). Pour simplifier les équations, on va considérer que les fonctions d'utilité sont définies par des fonctions de coût.

On définit $Q(s,a)$ le coût espéré de l'exécution de l'action a dans l'état s :

$$Q(s,a) = C(s,a) + \gamma \sum_{s' \in s} P_a(s'|s) E(s') \tag{3.4}$$

À partir de l'équation de Bellman suivante :

$$E(s) = \min_{a \in A} Q(s,a) \tag{3.5}$$

On peut définir $E_{\pi*}(s)$ le coût optimal dans l'état s :

$$E_{\pi*}(s) = \min_a \{C(s,a) + \gamma \sum_{s' \in s} P_a(s'|s) E_{\pi*}(s')\} \tag{3.6}$$

À partir de la formule 3.6, il existe deux algorithmes permettant de calculer $\pi*$.

Itération de politique Le premier algorithme est appelé itération de politique, dont le but est de résoudre le système d'équation suivant à partir d'une politique initiale π arbitraire :

$$E_\pi(s) = C(s, \pi(s)) + \gamma \sum_{s' \in S} P_{\pi(s)}(s'|s) E_\pi(s') \tag{3.7}$$

L'itération de politique trouve ensuite des actions qui diminuent la valeur de $E_\pi(s)$ et met à jour la politique avec ces actions jusqu'à ce que cela converge vers une politique optimale.

Itération de valeur Le deuxième algorithme permettant de calculer $\pi*$ s'appelle l'itération de valeur. À partir d'une valeur initiale arbitraire pour chaque $E(s)$, l'algorithme calcule itérativement la valeur

$$E_k(s) = \min_{a \in A} \{(s,a) + \gamma \sum P_a(s'|s) E_{k-1}(s')\} \tag{3.8}$$

en augmentant k à chaque étape jusqu'à ce que $E_k(s)$ converge vers le coût optimal et la politique correspondante vers la politique optimale.

3.1.2 Planification probabiliste sous observabilité partielle

Dans cette section on s'intéresse maintenant à une situation d'observabilité partielle du système. Le cadre généralement utilisé et présenté ici est celui des Processus de décision Markoviens partiellement observables (POMDP) (Kaelbling et collab., 1998). L'ensemble des définitions de cette section sont traduites du livre de Ghallab et collab. (2004).

Définition 14: Système stochastique partiellement observable

Un système stochastique partiellement observable est défini par :
- Un système stochastique $\Sigma = (S, A, P)$, tel que défini dans la définition 11.
- Un ensemble fini d'observations O avec des probabilités $P_a(o|s)$, pour chaque $a \in A$, $s \in S$ et $o \in O$. $P_a(o|s)$ représente la probabilité d'observer o dans un état s après avoir exécuté l'action a. Les probabilités sont définies pour chaque état $s \in S$ et action $a \in A$ avec $\sum_{o \in O} P_a(o|s) = 1$.

Les observations sont introduites afin de modéliser l'observabilité partielle. Le seul moyen d'obtenir des informations sur l'état du système est d'utiliser des observations.

Dans les POMDP, on modélise une observation par une distribution de probabilités sur les états du système. Les distributions de probabilités sur les états sont appelées les états de croyance.

Définition 15: États de croyance

On note b un état de croyance et B l'ensemble des états de croyance. $b(s)$ désigne la probabilité assignée à l'état s par l'état de croyance b. On a $0 \leq b(s) \leq 1$ pour tout $s \in S$ et $\sum_{s \in S} b(s) = 1$. Étant donné un état de croyance b, l'exécution d'une action a résulte en un nouvel état de croyance b_a. Pour chaque état $s \in S$, la probabilité $b_a(s)$ peut être calculée comme la somme de la distribution de probabilités déterminée par b pondérée par la probabilité que l'action a mène de s' à s :

$$b_a(s) = \sum_{s' \in S} P_a(s|s') b(s') \tag{3.9}$$

On peut calculer la probabilité d'observer $o \in O$ après l'exécution de l'action $a \in A$ comme ceci :

$$b_a(o) = \sum_{s \in S} P_a(o|s)b(s) \tag{3.10}$$

Enfin, on peut calculer la probabilité que l'état soit s après l'exécution de l'action a dans l'état de croyance b en observant o :

$$b_a^o(s) = \frac{P_a(o|s)b_a(s)}{b_a(o)} \tag{3.11}$$

Plans comme politiques sur les états de croyance

Dans les POMDP, une politique est une fonction associant les états de croyance aux actions. Soit B un ensemble d'états de croyance. Une politique est une fonction $\pi : B \to A$.

Problèmes de planification comme problèmes d'optimisations

Un problème de planification dans le cadre des POMDP peut être défini comme un problème d'optimisation dans lequel la solution est une politique optimale $\pi^* : B \to A$.

Cette politique peut être générée en considérant le problème comme un problème de MDP totalement observable sur l'ensemble infini des états de croyance. L'équation sur les états de croyance correspondant à l'équation de Bellman s'écrit de la manière suivante :

$$E(b) = \min_{a \in A} C(b, a) + \gamma \sum_{o \in O} b_a(o)E(b_a^o) \tag{3.12}$$

dans laquelle

$$C(b, a) = \sum_{s \in S} C(s, a)b(s) \tag{3.13}$$

Complexité

Comme dans la section 3.1.1 de ce manuscrit décrivant quelques résultats de complexité pour les MDP, les résultats présentés ici pour les POMDP sont tirés de l'article de Mundhenk et collab. (2000). Parmi ces résultats, nous nous intéressons particulièrement au problème d'existence de politique, qui est le plus proche de notre problématique. Le problème d'existence d'une politique stationnaire pour les POMDP est NP-complet. Les problèmes d'existence d'une politique dépendante de l'historique pour les POMDP sont PSPACE-complets.

Principes de résolution

Une façon de résoudre les POMDP est d'utiliser les algorithmes fonctionnant pour les MDP complètement observables sur les états de croyance. Étant donné que l'état de croyance est infini et continu, le problème des POMDP est très difficile à résoudre. Cependant, quelques algorithmes fonctionnent très bien sur des problèmes de taille relativement petite. Une des manières de contourner la difficulté de calcul de la solution est de calculer des approximations de politique optimale. Les techniques de recherche en avant dans l'espace des états de croyance guidés par une fonction heuristique sur les états de croyance peuvent aussi être utilisés pour calculer un plan dont l'optimalité n'est pas certaine.

3.1.3 Conclusion

Les méthodes de planification probabiliste permettent de traiter l'incertitude modélisée sous forme de probabilités sur les effets des actions et les transitions d'états. Dans le cas d'une observabilité totale, les processus de décision markoviens sont généralement utilisés, et peuvent être résolus par des méthodes d'itération de politique ou de valeur. Dans le cas d'une observabilité partielle, les processus

de décision markoviens partiellement observables sont généralement utilisés, et peuvent être résolus par exemple en utilisant les algorithmes fonctionnant pour les MDP totalement observables sur les états de croyance. Cependant, l'incertitude d'une mission ne peut pas toujours être exprimée de manière probabiliste à cause du manque d'information nécessaire à la quantification de ces probabilités. Dans ce cas, l'incertitude peut être modélisée comme un ensemble fini d'états initiaux possibles représentant les différents scénarios possibles compte tenu des informations que nous possédons sur l'environnement. Dans la suite de cet état de l'art, nous allons justement nous intéresser aux méthodes de planification symbolique permettant de traiter l'incertitude modélisée sous la forme d'un ensemble fini d'états initiaux possibles.

3.2 Planification symbolique

Dans la section précédente, nous avons détaillé le formalisme de la planification probabiliste dans laquelle l'incertitude est modélisée sous forme de probabilités sur les transitions d'états et les effets des actions. Cette modélisation de l'incertitude sous forme de probabilités n'est pas toujours possible, notamment dans des missions de robotique autonome dans des environnements incertains. Nous allons maintenant nous intéresser à la planification symbolique dans laquelle l'incertitude n'est généralement pas modélisée sous forme de probabilités mais plutôt sous la forme d'un ensemble d'états initiaux possibles. C'est dans ce cadre en particulier que notre approche se place.

Deux méthodes de planification symbolique se distinguent particulièrement : la planification conformante et la planification contingente. Nous allons tout d'abord détailler le principe de la planification conformante puis dans un deuxième temps nous allons nous intéresser à la planification contingente.

3.2.1 Planification conformante

En planification conformante (Albore et collab., 2011), on se place sous l'hypothèse d'une observabilité nulle du système. On ne va donc avoir aucun moyen de lever l'incertitude sur les variables de l'environnement, les observations n'étant pas possibles. La planification conformante a pour objectif de réussir à calculer un plan menant au but du problème quel que soit l'état initial possible et sans réaliser d'observation.

Contrairement à la planification probabiliste, en planification conformante, on va définir un état de croyance comme l'ensemble des états possibles. On notera $\mathcal{B}_0$ l'état de croyance initial et $\mathcal{B}$ l'état de croyance courant. L'introduction de cet état de croyance modifie la manière dont le problème et le domaine de planification doivent être modélisés par rapport à la planification classique étant donné qu'il n'y a pas d'état initial unique mais un ensemble d'états initiaux.

Exemple: Ensemble d'états initiaux possibles

La figure 3.1 donne un exemple d'un ensemble d'états initiaux possibles, et donc de l'état de croyance $\mathcal{B}_0$ initial de la mission de récolte de rocher. Ici, on ne sait pas si l'obstacle est en position (2,0) ou (0,1) dans la grille. Il va donc falloir calculer un plan qui fonctionne quel que soit la position de cet obstacle.

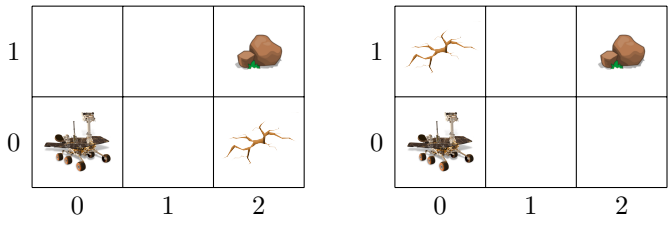

FIGURE 3.1 – Exemple d'ensemble d'états initiaux possibles

PDDL

Le langage PDDL permet de modéliser l'incertitude sous forme d'ensemble d'états initiaux. Pour cela, il suffit d'introduire des balises *oneof* ou *or* dans la balise (: *init* représentant l'état initial du problème PDDL. La balise *oneof* permet de représenter un opérateur logique *xor* sur des propositions de l'état initial.

Exemple: Modélisation PDDL de l'incertitude de l'exemple

On peut utiliser un opérateur *oneof* pour décrire l'ensemble d'états initiaux possibles de notre exemple de la figure 3.1 de la manière suivante :

```
(oneof
(is_obstacle cell01)
(is_obstacle cell20)
)
```

On peut noter que le *oneof* ne prend en compte que les variables incertaines de l'environnement. Dans notre cas, il s'agit uniquement de la variable *is_obstacle* décrivant la position de l'obstacle dans l'environnement.

On peut aussi utiliser un opérateur *or* pour décrire une relation logique *or* entre des variables de l'état initial du problème PDDL, ce qui ne nous est pas utile dans notre exemple.

Les définitions des états et des actions restent les mêmes qu'en planification classique (Définition 1). En revanche, la définition de la fonction de transition d'états T diffère, celle-ci ne considérant plus un seul état mais un ensemble d'états.

Définition 16: Actions

Tout comme en planification classique, on définit une action comme un couple $a = (Pre(a), Eff(a))$ avec :
- $Pre(a)$ un ensemble de préconditions
- $Eff(a)$ un ensemble d'effets.

En revanche, la prise en compte de l'incertitude dans le problème de planification induit le fait que l'exécution de certains effets des actions puisse être conditionné par un ensemble de propositions représentant l'état dans lequel l'effet sera réalisé. Chaque effet $e \in Eff$ est défini comme un triplet $Cond(e) \subseteq F$, $Add(e) \subseteq F$, $Del(e) \subseteq F$ correspondant respectivement aux conditions nécessaires à l'exécution de l'effet, à un ensemble d'additions et un ensemble d'effacements.

Une action a est applicable dans un état s ssi $Pre(a) \subseteq s$. Le résultat de l'application de a dans s est un nouvel état défini par la fonction de transition T :

$$T(s,a) = s - \bigcup_{e \in Eff(a) \text{ s.t. } Con(e) \subseteq s} Del(e) \cup \bigcup_{e \in Eff(a) \text{ s.t. } Con(e) \subseteq s} Add(e) \tag{3.14}$$

Une action a est applicable dans un état de croyance $\mathcal{B}$ ssi $\forall s \in \mathcal{B}\ Pre(a) \subseteq s$. Le résultat de l'application de a dans $\mathcal{B}$ est un nouvel état de croyance défini par la fonction de transition $T(\mathcal{B}, a)$:

$$T(\mathcal{B}, a) = \{T(s,a), \forall s \in \mathcal{B}\} \tag{3.15}$$

où $T(s,a)$ est la fonction de transition d'état définie dans la définition 2.

Un domaine de planification conformante est différent d'un domaine classique (Définition 3) par la considération de la fonction de transition définie par la définition 16.

Définition 17: Domaine conformant

Un domaine de planification conformante dans F est un système de transition d'états $\Sigma = (S, A, T)$ tel que :
- S est l'ensemble des états possibles avec $S \subseteq 2^F$

— A est l'ensemble des actions
— T est la fonction de transition définie par l'équation 5.2.

Un problème de planification conformante diffère de la planification classique par la considération d'un ensemble d'états initiaux possibles.

Définition 18: Problème conformant

Un problème de planification conformante est un triplet $\mathcal{P} = (\Sigma, \mathcal{B}_0, g)$ dans lequel :
— Σ est un domaine comme décrit dans la définition 17
— $\mathcal{B}_0 \subseteq S$ est l'ensemble des états initiaux possibles, l'ensemble d'état de croyance initial
— g est un ensemble de fluents dans F représentant le but

En planification conformante, un plan est une séquence d'actions pouvant être défini tel que dans la définition 5. La différence avec la planification classique se situe dans la définition d'un plan solution, qui ne doit pas mener au but seulement depuis un état initial, mais depuis l'ensemble des états initiaux possibles.

Définition 19: Solution conformante

Soit $\mathcal{P} = (\Sigma, \mathcal{B}_0, g)$ un problème de planification. Un plan π est une solution pour $\mathcal{P}$ si $\forall s \in \mathcal{B}_0$, $g \subseteq T(s, \pi)$.
On dit qu'un plan conformant $\pi =< a_1, ..., a_k >$ est valide ssi $\forall\ a_i \in \pi$, $\forall\ s \in \mathcal{B}_0$, $Pre(a_i) \subseteq T(s, < a_1, ..., a_{i-1} >)$.

Exemple: Plan solution

On peut illustrer un plan solution d'un problème de planification conformante avec un des plans solution de notre exemple de la figure 3.1. Un des plans solutions de la mission peut être la séquence d'actions $\pi =< move\ cell10, move\ cell11, move\ cell21, collect >$ dont l'application sur l'ensemble d'états initiaux de la figure 3.1 est illustrée par la figure 3.2 ci-dessous.

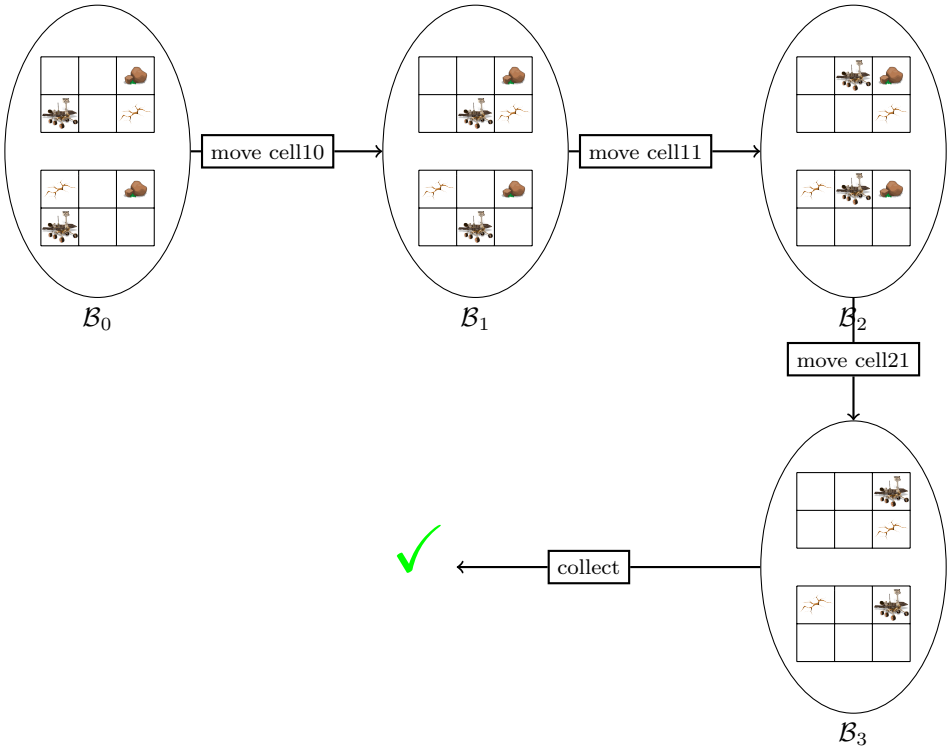

FIGURE 3.2 – Application du plan conformant π

Complexité

Dans l'article sur la complexité de la planification avec des informations incomplètes, Haslum et Jonsson (1999) indiquent que les problèmes de planification conformante sont EXPSPACE-complets.

Principes de résolution

Il existe différents types d'approches permettant de résoudre un problème de planification conformante. Certaines approches permettant de résoudre les problèmes de planification conformante sont basées sur une traduction du problème conformant en problème classique.

Dans leur article, Palacios et Geffner (2006) introduisent un nouveau schéma général de traduction de problèmes conformants en problèmes classiques polynomial mais pas toujours complet. Ce schéma dépend de deux paramètres : un ensemble d'étiquettes faisant référence aux contextes locaux dans les états initiaux, et un ensemble de fusions représentant des ensembles exhaustifs d'étiquettes. En réalisant ce planificateur, Palacios et Geffner ont aussi introduit une mesure de complexité en planification conformante appelée largeur conformante, ainsi qu'un schéma de traduction $K_i(P)$ impliquant seulement les étiquettes de taille i et qui est complet pour les problèmes de largeur $\leq i$. Palacios et Geffner ont aussi montré que la plupart des benchmarks conformants ont une largeur de 1. Le planificateur conformant T0, lui aussi introduit dans (Palacios et Geffner, 2006) est basé sur la traduction $K_i(P)$ et utilise le planificateur classique FF. Ce planificateur conformant a prouvé son efficacité en gagnant la cinquième compétition internationale de planification IPC5.

Le planificateur conformant T1 (Albore et collab., 2011) allie la flexibilité des planificateurs qui recherchent explicitement dans l'espace de croyance avec les heuristiques et états de croyance qui résultent des traductions. Afin de contourner cette limitation des approches basées sur la traduction, T1 formule une nouvelle traduction traitable et complète mais pas toujours saine. Cette nouvelle traduction ne peut pas être utilisée pour traiter des problèmes conformants avec un planificateur classique mais elle est efficace pour dériver des heuristiques et calculer des états de croyance dans un planificateur basé

sur les états de croyance. L'idée principale est d'échantillonner et utiliser un nombre polynomial d'états initiaux, même si leur nombre est exponentiel. T1 exploite la flexibilité des planificateurs basés sur l'espace de croyance de deux manières, premièrement en incorporant une seconde heuristique dérivée des *invariants oneof* du problème qui est reliée à des heuristiques de cardinalité et de point de repère, et deuxièmement en utilisant un algorithme de recherche meilleur d'abord multi-queue façonné à partir des planificateurs classiques FD (Helmert, 2006) et LAMA (Richter et collab., 2008).

Conformant Planner via Counter Example and Sampling (CPCES) (Grastien et Scala, 2017) est un planificateur conformant sain et complet basé sur la traduction du problème de planification conformante en problème de planification classique de (Palacios et Geffner, 2009). Cette traduction est une nouvelle version de la traduction de (Palacios et Geffner, 2006), dans laquelle Palacios et Geffner introduisent la notion de *base*, un sous-ensemble d'états initiaux garanti d'avoir le même ensemble de plans que l'état de croyance initial complet. Une méthode de planification conformante saine et complète consiste dans un premier temps à calculer une base, puis à calculer une solution à partir de la base à l'aide d'une réduction en planification classique. Cependant, cette méthode n'est réalisable que si la base est petite. La base est garantie de représenter le même ensemble de plans que l'ensemble d'états initiaux complet. À la place, Grastien et Scala ont seulement besoin d'un échantillon d'états initiaux tels que le planificateur retourne un plan valide pour l'ensemble d'états initiaux, ce qui représente une condition nécessaire plus faible qu'une base. À partir de cet échantillonnage, l'état de croyance généré sera généralement plus petit qu'en se basant sur une base. La particularité de CPCES résulte dans le fait que celui-ci calcule un plan conformant en utilisant seulement un sous-ensemble des états initiaux possibles. CPCES effectue premièrement une réduction du problème en problème de planification classique pour un petit échantillon de l'état de croyance initial, et utilise FF pour obtenir un premier plan candidat. La validité de ce plan candidat sur l'espace de croyance complet est ensuite vérifié par une méthode de régression et une transformation du problème en problème SAT traité par le solveur Z3 (De Moura et Bjørner, 2008). Si le plan n'est pas valide sur un des états de l'état de croyance, alors cet état est extrait et constitue un contre-exemple prouvant que le plan en question n'est pas conformant pour l'ensemble des états initiaux. Cet état contre-exemple est ensuite ajouté à l'ensemble des états initiaux considérés, élargissant l'espace de recherche précédemment exploré. Cette procédure permet à CPCES de réaliser un échantillonnage intelligent de l'espace de croyance, chacun des contre-exemples étant pertinent en contredisant au moins un des plans précédemment calculés. Cette procédure est réalisée de manière itérative, démarrant avec un seul état de l'ensemble des états initiaux, jusqu'à ce qu'un plan conformant soit trouvé, ou bien que FF ne trouve aucun plan pour le sous-ensemble de recherche, auquel cas le contre-exemple et le plan précédemment calculé par FF sont retournés.

La limitation principale des approches basées sur la traduction est que les traductions complètes ne sont pas réalisables pour des problèmes de grande taille et les traductions incomplètes peuvent très bien traduire un problème résolvable en un problème sans solution. Pour contourner ces difficultés, certaines méthodes de planification conformante ne sont pas basées sur la traduction.

Disjunctive Normal Form (DNF) (To et collab., 2009) est un planificateur conformant complet basé sur la progression et la recherche best-first. Ce planificateur repose sur une représentation complète des états de croyance par une formule de forme normale disjonctive. Cette représentation permet un calcul des états de croyance successeurs ne demandant pas une consommation de mémoire très importante, ce qui permet des calculs rapides et la résolution de problèmes de plus grande taille. Cependant, cette représentation connaît des limites sur des problèmes suffisamment larges pour que la taille de la formule disjonctive pose problème.

Conformant-FF (Hoffmann et Brafman, 2006) propose une nouvelle approche de modélisation des états de croyance dans laquelle le but et les préconditions des actions sont restreints à de simples conjonctions de propositions, tout comme en STRIPS. Dans Conformant-FF, une séquence d'actions est un plan conformant seulement si celle-ci mène à un état de croyance dans lequel l'intersection des mondes contient toutes les propositions buts. Ici, un état de croyance $\mathcal{B}$ est représenté seulement par l'état de croyance initial auquel on associe une séquence d'actions *act* menant à $\mathcal{B}$. Afin de vérifier l'accomplissement d'un but ou les préconditions d'une action, Conformant-FF teste si chaque proposition p est contenue dans l'intersection des mondes dans $\mathcal{B}$. Ces propositions sont dites connues dans $\mathcal{B}$.

Réaliser ce test est en général difficile et co-NP complet. Cette représentation des états de croyance permet de réduire la place en mémoire occupée par la représentation des états de croyance en ne conservant pas une connaissance entière des états de croyance au prix d'un temps de vérification des propositions plus élevé. Ce temps de vérification reste tout de même assez réduit grâce à l'utilisation d'un solveur SAT. Afin de ne pas traverser l'espace de recherche de manière aveugle, Conformant-FF a adapté la relaxation du problème et la fonction heuristique de FF au contexte conformant.

Conclusion

La planification conformante est une méthode de planification symbolique consistant à traiter l'incertitude sous forme d'un ensemble d'états initiaux en calculant un plan sous forme d'une séquence d'actions menant au but du problème quel que soit l'état initial possible sans réaliser d'action d'observation de l'environnement. Parmi ces méthodes, certaines se basent sur une traduction du problème de planification conformante en problème de planification classique ((Palacios et Geffner, 2006),(Albore et collab., 2011), (Grastien et Scala, 2017)). Le planificateur DNF (To et collab., 2009) se base sur une représentation complète des états de croyance sous forme normale disjonctive, tandis que dans Conformant-FF (Hoffmann et Brafman, 2006), l'état de croyance est représenté seulement par l'état de croyance initial auquel on associe une séquence d'action menant à l'état de croyance courant. Toutes ces méthodes parviennent à traiter un certain nombre de problèmes de manière efficace. Cependant, il existe tout de même un grand nombre de problèmes pour lesquels l'incertitude n'est traitable que par l'ajout d'observations de l'environnement. Dans ce cas, une autre méthode de planification symbolique peut être utilisée : la planification contingente. Dans la section suivante, nous allons justement nous intéresser à la planification contingente.

3.2.2 Planification contingente

La planification conformante n'est pas toujours capable de trouver un plan solution quel que soit l'état initial du problème. En effet, il existe des problèmes que l'on ne peut résoudre sans lever l'incertitude sur une partie des états initiaux. La figure 3.3 illustre ce type de problème dans notre exemple de mission de collecte de rocher.

Exemple: Ensemble d'états initiaux possibles

On peut reprendre l'ensemble des états initiaux du problème de planification conformante exposé dans la figure 3.1 et y ajouter un état initial possible supplémentaire rendant le problème non traitable par un planificateur conformant. Le fait de ne pas savoir si l'obstacle se situe en position (2,0) ou en position (1,1) fait en sorte qu'aucun plan sous forme de séquence d'actions n'est valable pour ces deux positions d'obstacles, celles-ci rendant le déplacement du robot jusqu'au rocher impossible.

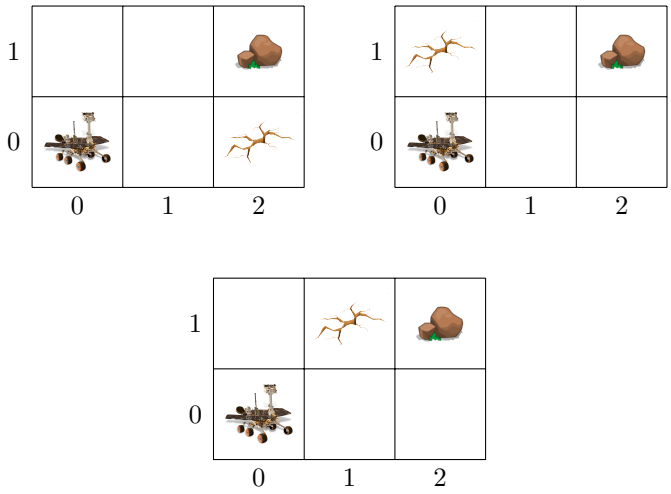

FIGURE 3.3 – Exemple d'ensemble d'états initiaux possibles

Ce qui différencie la planification contingente (Albore et collab., 2009) de la planification conformante est dans un premier temps la possibilité d'effectuer des actions d'observation de l'environnement permettant de lever les incertitudes.

Tout comme une action classique, une action d'observation correspond à un couple (Pre, Eff), cependant elle diffère d'une action classique par ses conditions d'application et ses effets.

Définition 20: Observation

Soit l'ensemble d'observations Ω, une action d'observation $\sigma \in \Omega$ est applicable dans un état de croyance $\mathcal{B}$ ssi $\forall s \in \mathcal{B}, Pre(\sigma) \subseteq s$. Soit $p(\sigma) \in F$ la proposition observée par l'action d'observation σ, $Eff(\sigma) = \nu(p)$ correspond à la valeur de vérité observée de la proposition $p(\sigma)$. L'application d'une action d'observation σ à un état de croyance $\mathcal{B}$ va éliminer de $\mathcal{B}$ les états ne contenant pas la valeur de vérité de la proposition observée $Eff(\sigma)$. Pour chaque $\sigma \in \Omega$, on a

$$T(\mathcal{B}, \sigma) = \{s,\ s.t\ s \in \mathcal{B} \wedge Eff(\sigma) \in s\} \tag{3.16}$$

La définition d'un domaine de planification contingente est la même que la définition 17 d'un domaine conformant à l'exception de l'ajout de l'ensemble des observations Ω et de la fonction de transition des actions d'observations.

Définition 21: Domaine contingent

Un domaine de planification contingente dans F est un système de transition d'états $\Sigma_\Omega = (S, A, \Omega, T)$ tel que :
- S est l'ensemble des états possibles avec $S \subseteq 2^F$
- A est l'ensemble des actions
- Ω est l'ensemble des actions d'observations
- $T(\mathcal{B}, a)$ est la fonction de transition définie par l'équation 5.2 avec $a \in A$
- $T(\mathcal{B}, \sigma)$ est la fonction de transition définie par l'équation 5.3 avec $\sigma \in \Omega$.

Définition 22: Problème contingent

Un problème de planification contingente est un triplet $\mathcal{P}_\Omega = (\Sigma_\Omega, \mathcal{B}_0, g)$ dans lequel :
- Σ_Ω est un domaine contingent comme décrit dans la définition 21
- $\mathcal{B}_0 \in S$ est l'ensemble des états initiaux possibles, l'ensemble d'état de croyance initial
- g est un ensemble de fluents dans F représentant le but.

L'effet de l'application d'une action d'observation est illustrée grâce à notre exemple de mission de collecte de rocher dans la figure 3.4

Exemple: Action d'observation

On peut définir l'action d'observation *obs_obst* permettant d'observer la valeur de la proposition *is_obstacle* d'une case adjacente au robot. Dans l'ensemble d'états initiaux de la figure 3.3, il est intéressant d'appliquer *obs_obst* afin de séparer l'ensemble des états initiaux en deux ensembles d'états possibles $\mathcal{B}^+$ et $\mathcal{B}^-$ permettant de savoir si l'obstacle se situe en position (1,1) ou non. Pour cela, le robot doit se trouver soit en position (0,1) potentiellement occupée par un obstacle, ou dans la position (1,0) libre quel que soit le scénario. C'est dans cette deuxième position qu'il est logique de déplacer le robot afin de réaliser l'observation.

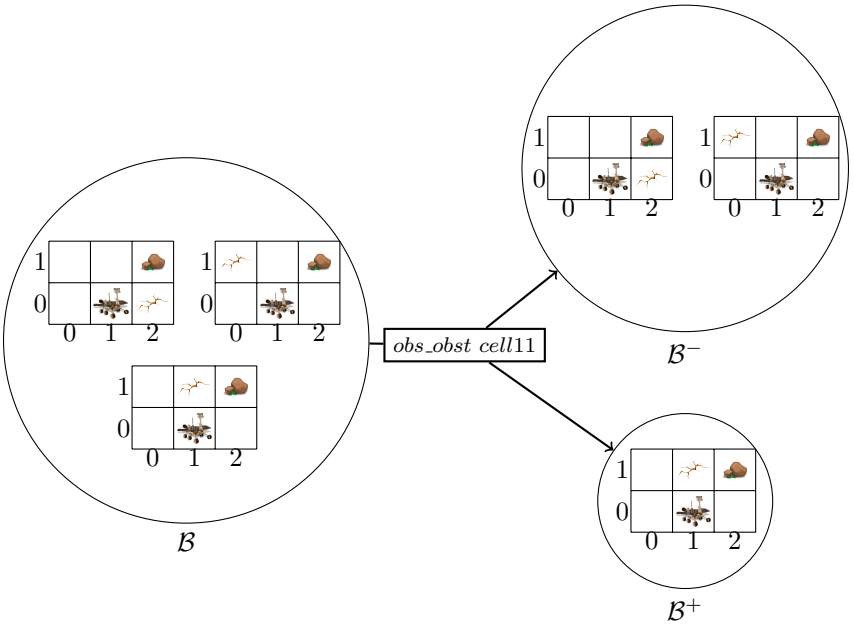

FIGURE 3.4 – Exemple d'application d'une observation

Modélisation PDDL d'une observation

En PDDL, la modélisation d'une action d'observation est presque similaire à celle d'une action classique. Ce qui diffère entre les deux modélisations est le remplacement de la balise : $effect$ d'une action par une balise : *observe* afin de caractériser quelle proposition sera observée par l'action d'observation.

Exemple: Exemple d'observation PDDL

On peut modéliser l'action d'observation *obs_obst* de notre exemple de la figure 3.4 de la manière suivante :

```
(:action obs_obst
 :parameters (?x - robot ?t - position ?z - position )
 :precondition (and (at_robot ?x ?t) (adjacent ?t ?z))
 :observe (is_obstacle ?z))
```

Contrairement à la planification conformante, un plan contingent n'est pas une séquence d'actions mais un graphe de décision dans lequel les branches (elles mêmes des graphes de décision) sont conditionnées par le résultat d'une observation de l'environnement.

Un plan contingent incorporant l'observation de l'exemple 3.4 sera composé d'une branche menant à l'état de croyance $\mathcal{B}$ puis de deux branches partant des états de croyance $\mathcal{B}^+$ et $\mathcal{B}^-$ et menant au but du problème.

Définition 23: Plan contingent solution

Un plan contingent est solution d'un problème $\mathcal{P}_\Omega = (\Sigma_\Omega, \mathcal{B}_0, g)$ s'il est exécutable dans $\mathcal{B}_0$ et si chaque chemin du plan mène au but g

Complexité

Dans son article sur la complexité en planification avec observabilité partielle, Rintanen (2004) expose des résultats de complexité de la planification contingente. En considérant que l'état courant du

monde peut être exactement observé pendant l'exécution du plan (observabilité totale), le problème d'existence d'un plan est EXPTIME-complet. Dans le cas d'une observabilité partielle, le problème d'existence d'un plan contingent devient 2-EXPTIME-difficile. Lorsque l'on se place dans un problème avec des opérateurs déterministes et une observabilité partielle, le problème devient EXPSPACE.

Principes de résolution

Pour des raisons d'efficacité, on peut vouloir privilégier des programmes plutôt que des plans pour la programmation agent haut niveau.

Le langage de programmation GOLOG (Levesque et collab., 1997; De Giacomo et collab., 1999) est un langage de programmation d'agent haut niveau développé pour modéliser le comportement d'un agent dans des mondes dynamiques à un haut niveau d'abstraction. Les seules différences entre le langage GOLOG et les langages de programmation structurés est que les effets ainsi que l'exécution du programme peuvent être déterminés par l'agent à travers une axiomatisation de calcul des situations (McCarthy et Hayes, 1981) (Le calcul des situations est un formalisme logique développé pour la représentation et le raisonnement à partir de domaines dynamiques dans lequel les scénarios changeant sont représentés comme un ensemble de formules logiques du premier ordre) et que les sémantiques d'exécution sont définies en logique. (Baier et Pinto, 2003) présente des algorithmes permettant de générer des programmes GOLOG contingents en étendant le calcul des situations avec de l'incertitude sur les effets des actions et une observabilité totale du monde. La procédure de planification démarre par le calcul d'un plan initial sous la forme d'une séquence d'actions permettant d'atteindre le but avec une certaine probabilité. L'algorithme identifie ensuite les états dans lesquels l'exécution de la séquence d'action n'aboutit pas au but puis ajoute une condition if-then-else au plan permettant de déterminer si l'agent se trouve dans l'état problématique par l'observation d'un fluent isolant cet état de tous les autres. L'algorithme calcule ensuite la branche alternative et poursuit son processus de vérification de la validité du plan. Il existe plusieurs limitations à cette approche, dans un premier temps on peut citer l'hypothèse d'observabilité totale qui limite son utilisation, mais aussi le fait que la recherche est complètement aveugle, augmentant la complexité des algorithmes utilisés.

Dans l'article (Pryor et Collins, 1996), Pryor et Collins présentent Cassandra, un planificateur contingent en ordre partiel dans lequel les décisions sont représentées comme des actions explicites permettant à l'agent exécutant le plan de décider quelle branche du plan celui-ci doit emprunter. Pryor et Collins définissent l'acquisition de connaissances sur le monde comme des sous-buts qui sont planifiés de la même manière que n'importe quel autre sous-but. Cassandra distingue donc la procédure d'acquisition d'information du processus de prise de décision, les sous-buts d'acquisition de connaissances étant les préconditions des actions de décision.

Cassandra n'essaie de calculer un plan contingent que lorsqu'une incertitude est rencontrée. Avant cela, Cassandra construit un plan en se basant sur la méthode développée pour le planificateur classique UCPOP (Weld et Penberthy, 1992) dans laquelle la planification se déroule à travers l'alternance de deux procédures : la résolution des conditions ouvertes et la protection des liens dangereux. Le planificateur tente de modifier son plan initial jusqu'à ce que plus aucune condition ouverte et plus aucun lien dangereux n'existe. L'exploration de l'espace des plans partiels est quant à elle guidée par un algorithme de recherche meilleur d'abord.

Les principales limitations de Cassandra résident dans un premier temps dans la qualité des plans générés, qui sont souvent plus complexes que nécessaire, et dans un deuxième temps dans le temps de calcul des plans permettant uniquement la résolution de problèmes simples.

Dans l'article (Bertoli et collab., 2001), Bertoli et collab. présentent une approche de planification contingente sous observabilité partielle basée sur un modèle représentant des observations automatiques (Tovey et Koenig, 2000) ainsi que des actions d'observation. L'algorithme contingent proposé par Bertoli et collab. est basé sur une exploration en avant de l'espace de recherche (un graphe AND/OR induit par le domaine) depuis l'état de croyance initial en écartant les plans cycliques. L'état de cette recherche est stocké en associant une étiquette à chaque état de croyance rencontré. L'algorithme génère des plans contingents acycliques permettant de résoudre l'incertitude sur les conditions initiales et les effets incertains des actions. L'algorithme présenté est intégré dans le planificateur Model Based Planner (MBP) (Cimatti et collab., 1998) capable de calculer une solution pour les domaines non déterministes sous observabilité totale. Ce planificateur est basé sur le model checking (McMillan, 1993) et sur les

diagrammes de décision binaires (BDD) (Bryant, 1992).

Dans l'article (Cimatti et collab., 2003), Cimatti et collab. définissent trois niveaux de robustesse d'une solution contingente :

— Les solutions *faibles* sont les plans qui peuvent accomplir le but mais ne sont pas garantis de l'atteindre à chaque fois, c'est à dire qu'au moins une des séquences possibles d'états générés par l'exécution du plan mène au but.
— Les solutions *fortes* sont les plans garantis d'atteindre le but quelque soit l'incertitude, toutes les séquences d'états générés par l'exécution du plan atteignent le but.
— Les solutions *fortes cycliques* qui sont des plans garantis d'atteindre le but sous une hypothèse *d'équité*, c'est à dire que leur exécution peut résulter en une séquence infinie d'états, mais que cet évènement n'apparaît que si certaines actions sont exécutées infiniment souvent dans un état donné et que ses effets menant à l'objectif ne s'exécutent jamais.

Cimatti et collab. présentent des algorithmes permettant de trouver des solutions *faibles*, *fortes* et *fortes cycliques* en générant des plans contingents qui observent le monde de manière répétée et exécutent différentes actions selon les résultats de celles-ci. Dans (Cimatti et collab., 2003) les plans sont décrits par des tables état-action associant à chaque état l'ensemble d'actions pouvant être exécutées dans cet état. Dans leur approche, Cimatti et collab. représentent le domaine de planification comme un automate avec un nombre fini d'états. La représentation et le parcours de cet automate sont réalisés grâce aux techniques standard du model-checking. Les ensembles d'états sont représentés par des formules propositionnelles et la recherche à travers l'espace d'état est réalisée comme un ensemble de transformations logiques sur ces formules. Tout comme (Bertoli et collab., 2001), Cimatti et collab. implémentent les algorithmes présentés dans le planificateur MBP (Cimatti et collab., 1998).

Certains planificateurs de la littérature se basent sur le principe de relaxation du problème contingent en problème conformant. Contingent-FF (Hoffmann et Brafman, 2005) est une de ces méthodes. Contingent-FF utilise la même méthode de représentation implicite des états de croyance que Conformant-FF (Hoffmann et Brafman, 2006) en utilisant une formule propositionnelle décrivant la séquence d'actions/observations utilisée pour atteindre l'état de croyance courant depuis l'état de croyance initial. Contingent-FF se base aussi sur la relaxation du problème de Conformant-FF. Afin d'adapter la relaxation du problème de Conformant-FF au contexte contingent, Contingent-FF introduit une relaxation additionnelle, relaxant le problème contingent en problème conformant. Le fait que Contingent-FF se base sur la relaxation de Conformant-FF a pour conséquence que son heuristique ne considère pas directement les observations, ce qui peut être un problème dans la résolution de problèmes contingents, les observations influant directement sur la compacité des plans.

Brafman et Shani (2012b) ont développé le planificateur SDR (Sample, Determinize, Replan) basé sur la traduction de (Palacios et Geffner, 2009). Celui-ci considère dans un premier temps un plan démarrant d'un état initial unique, réduisant le problème contingent en un problème de planification classique, pour ensuite incorporer des observations à ce plan permettant de lever les incertitudes sur les propositions dont la valeur est inconnue. Afin de pouvoir traiter des problèmes comportant un état de croyance de grande taille, SDR utilise un échantillonnage des états de l'état de croyance ainsi qu'une méthode de régression afin de vérifier la validité du plan calculé sur l'ensemble des états de l'état de croyance. Le problème de la méthode d'incorporation de SDR est que celle-ci peut résulter en la réalisation d'observations pouvant être inutiles. De plus, SDR n'est complet que pour des problèmes déterministes et sans deadends.

(Brafman et Shani, 2012a) ont proposé une autre méthode de traduction saine et complète appelée traduction multi-chemin. Cette méthode de compilation génère un problème de planification classique dont les solutions encodent un plan contingent considérant tous les chemins d'exécution. Le planificateur contingent découlant de cette méthode de compilation est nommée MPSR (Multi-Path, Sampling, Replanner). Celui-ci peut être utilisé hors ligne pour générer un plan contingent complet permettant d'atteindre un but quel que soit l'incertitude lorsqu'un tel plan existe. Malheureusement, la taille de ces plans, et la taille du problème classique généré peuvent être exponentielles dans le nombre de propositions du problème. Pour répondre à ce problème, Brafman et Shani ont développé une architecture de replanification qui utilise une version modifiée et incomplète de la traduction à chaque étape. Afin de contrôler la taille du problème classique généré et de maintenir l'information sur la

connaissance de l'agent en cours d'exécution, MPSR se base sur la méthode d'échantillonnage et la méthode de maintien de l'état de croyance courant définie dans SDR. Une autre spécificité de cette méthode est l'amélioration de l'ensemble des actions afin d'encoder le plan contingent. Pour cela plusieurs versions de chaque action sont introduites pour différents états de croyance.

Le planificateur Closed-Loop Greedy (CLG) (Albore et collab., 2009) est un planificateur contingent utilisant une traduction étendant l'approche de planification conformante basée sur la traduction introduite par Palacios et Geffner (2007) à la planification contingente. Dans le contexte conformant ou contingent, la traduction associe le problème de recherche dans l'espace des états de croyance en problème de recherche dans l'espace d'état avec des traductions complètes difficilement réalisables. CLG utilise une version modifiée de FF afin de calculer la séquence d'action π générée à partir de l'état initial s obtenu lors de la traduction. Contrairement à Contingent-FF et SDR, CLG intègre les observations via une relaxation du modèle heuristique utilisé pour sélectionner les prochaines actions, et en introduisant des M-littéraux spécifiques dans les préconditions permettant aux observations d'être considérées et incluses dans le plan relaxé lorsque celles-ci sont nécessaires pour déclencher une précondition. CLG a la particularité de pouvoir être utilisé hors ligne ou en ligne. Lorsqu'il est utilisé en ligne, les actions d'observation non déterministes sont appliquées en sélectionnant un des effets possibles de manière aléatoire. Au contraire, lorsqu'il est utilisé hors-ligne, les deux effets possibles sont considérés. Dans les deux cas, CLG est invoqué récursivement sur les états obtenus.

Muise et collab. (2014) ont développé le planificateur contingent PO-PRP en modifiant le planificateur PRP (Muise et collab., 2012) afin de calculer des plans représentés comme des graphes dirigés acycliques. PO-PRP utilise la méthode de compilation développée par Bonet et Geffner (2011) pour convertir un problème de planification contingente partiellement observable en un problème totalement observable non déterministe (FOND).

Dans l'article (Albore et collab., 2009), Albore et collab. définissent une mesure de complexité d'un problème contingent, appelée largeur contingente. Cette mesure est définie en terme de nombre maximum de variables dont les valeurs sont initialement inconnues, et dont l'incertitude doit nécessairement être levée afin de résoudre le problème. La plupart des problèmes contingents de la littérature sont de largeur 1. Cependant, il existe tout de même des problèmes nécessitant l'utilisation de plusieurs observations consécutives de l'environnement afin de trouver une solution. Les problèmes correspondant à cette catégorie ont une largeur supérieure à 1. Malgré l'efficacité de CLG et PO-PRP sur des problèmes contingents de largeur égale à 1, CLG n'est pas complet pour les problèmes contingents de largeur supérieur à 1, celui-ci ne trouve pas toujours de solution même s'il en existe une tandis que PO-PRP ne trouve pas toujours de solution valide pour ce type de problème.

Komarnitsky et Shani ont développé un planificateur contingent nommé CPOR (Contingent Planner using Online Replanning) n'étant pas limité par la largeur contingente d'un problème contrairement à CLG et PO-PRP. Le principe de CPOR est d'utiliser une méthode de replanification en ligne, SDR (Brafman et Shani, 2012b), pour calculer chaque branche du plan contingent. Komarnitsky et Shani utilisent la technique de représentation des états de croyance développée par (Brafman et Shani, 2014) et consistant à maintenir uniquement la formule représentant l'état de croyance initial et la séquence d'actions et d'observations menant à l'état de croyance courant. Afin de vérifier si les préconditions d'une action, ou le but sont présents dans l'état de croyance courant, la formule est régressée à travers la séquence d'action/observation jusqu'à l'état de croyance initial. La vérification de la validité de la formule obtenue est réalisée via une requête SAT. Étant donné que SDR (Brafman et Shani, 2012b) n'est complet que pour des problèmes comportant des actions déterministes et sans deadends, CPOR n'est complet que pour cette classe de problèmes.

DNF_{ct} (To, 2012) est un planificateur contingent encodant les états de croyance par des formules en forme normale disjonctive, une idée proposée par To et collab. pour la planification conformante et appliquée au planificateur conformant DNF (To et collab., 2009), pour la recherche de solutions dans l'espace des états de croyance. Dans DNF, une fonction de transition complète pour calculer les états de croyance successeurs en présence d'information incomplète à été définie. DNF_{ct} étend cette fonction afin de gérer les actions d'observations et les actions non-déterministes dans le paradigme de

recherche AND/OR pour la planification contingente. Cette fonction permet le calcul d'états de croyance successeurs efficacement, en temps polynomial sous des hypothèses raisonnables. DNF_{ct} introduit également une nouvelle variante d'algorithme de recherche AND/OR permettant au planificateur d'élaguer efficacement l'espace de recherche. Lorsqu'une solution est trouvée, grâce à cet élagage, le graphe de recherche restant est aussi un arbre solution du problème de planification contingente.

Conclusion

La planification contingente est une méthode de planification symbolique permettant de traiter l'incertitude modélisée sous forme d'un ensemble d'états initiaux possibles en calculant un plan représenté comme un graphe de décision dans lequel les branches sont conditionnées par le résultat d'une observation de l'environnement. De nombreuses méthodes contingentes existent et possèdent leur propre particularité. Parmi ces méthodes, certaines modélisent le plan comme un programme (Baier et Pinto, 2003), d'autres reposent sur des techniques de model checking (Bertoli et collab., 2001; Cimatti et collab., 2003), et d'autres sont basées sur différentes traductions (Brafman et Shani, 2012b,a; Albore et collab., 2009; Muise et collab., 2014). Le planificateur Cassandra (Pryor et Collins, 1996) distingue la procédure d'acquisition d'information du processus de prise de décision. CPOR (Komarnitsky et Shani, 2016), a lui la particularité d'utiliser une méthode de replanification en ligne pour le calcul des branches du plan. Tout comme CPOR, Contingent-FF (Hoffmann et Brafman, 2005) a la particularité de représenter l'état de croyance courant comme l'association de l'état de croyance initial et de la séquence d'actions/observations menant à l'état de croyance courant. Enfin, DNF_{ct} (To, 2012) se base sur la représentation des états de croyance définie dans le planificateur conformant DNF (To et collab., 2009) pour représenter efficacement les états de croyance sous la forme d'une formule normale disjonctive. Malgré toutes ces particularités, ces approches de planification contingente n'intègrent pas réellement de mécanisme permettant de limiter les observations contenues dans le plan calculé. Pourtant, ces observations ont un rôle dans la taille du plan final et son temps de calcul, leur présence indiquant nécessairement le calcul de nouvelles branches du plan. On peut facilement imaginer que réduire ces observations et la taille du plan serait avantageux, la taille des plans étant souvent un obstacle à la planification hors-ligne, les arbres de décision calculés prenant rapidement une dimension exponentielle. De plus, ces observations ont un coût non négligeable durant la mission, que ce soit à cause des contraintes d'énergie ou de temps que nécessite l'exécution d'une observation.

Chapitre 4

Conclusion

La planification classique pose les bases de la planification, en permettant de résoudre des problèmes évoluant dans des environnements déterministes et n'incorporant aucune incertitude. Cependant, la planification classique, à elle seule, est incapable de traiter des problèmes comportant de l'incertitude. La planification probabiliste permet de traiter les problèmes dans lesquels l'incertitude est modélisée sous forme de probabilités, cependant une telle modélisation de l'incertitude n'est pas possible pour tout type de problèmes, celle-ci étant particulièrement difficile dans le contexte de la robotique d'exploration de l'environnement à cause du manque de données sur l'environnement. Une manière de traiter l'incertitude modélisée sous forme de différents scénarios possibles hors ligne est d'utiliser des méthodes de planification conformante. Ces méthodes permettent le calcul efficace d'une séquence d'actions menant au but du problème quel que soit le scénario initial possible du monde sans réaliser d'observation. Par conséquent, les plans générés par un planificateur conformant sont robustes à l'incertitude, quoi qu'il arrive, le plan conformant est garanti d'atteindre le but du problème. Malheureusement, il existe des problèmes pour lesquels une telle séquence d'actions n'existe pas et pour lesquels l'incertitude n'est pas traitable sans l'ajout d'observation. La planification contingente permet de traiter les problèmes sous incertitude en calculant un graphe de planification incorporant des observations de l'environnement. Ce cadre de planification semble particulièrement adapté aux missions de robotique autonome de par la compacité et l'embarquabilité aisée des plans contingents. La planification contingente possède tout de même un problème de complexité élevée devant être limitée pour éviter un phénomène d'explosion combinatoire. De plus, les observations ont un coût non négligeable durant la mission, que ce soit à cause de la consommation d'énergie ou de temps que cela nécessite. Malgré cela, la plupart des approches de planification contingente n'intègrent pas de mécanisme permettant de limiter ces observations. C'est pour cela que nous avons décidé de développer dans cette thèse un planificateur contingent dont le but est de limiter le nombre d'observations incorporées dans le plan.

Deuxième partie

Développement d'un planificateur contingent limitant le nombre d'observation

Chapitre 5

Développement d'un planificateur contingent reposant sur un planificateur conformant afin de limiter le nombre d'observation dans le plan

5.1 Motivation

Dans cette thèse, nous nous plaçons dans le contexte des missions de robotique autonome dans lesquelles il est souvent plus facile de modéliser l'incertitude de l'environnement sous forme d'un ensemble de scénarios possibles plutôt qu'un ensemble de distributions de probabilités, le modèle de l'environnement ne permettant pas toujours de quantifier l'incertitude. Nous nous concentrons sur les problèmes comportant des opérateurs déterministes et dans lesquels l'incertitude est modélisée sous forme d'un ensemble d'états initiaux possibles. Nous avons vu dans l'état de l'art de ce manuscrit que lorsque l'incertitude est modélisée sous cette forme, il existe deux principales méthodes permettant de traiter l'incertitude avant le début de la mission : la planification conformante et la planification contingente. La planification conformante permet le calcul efficace d'une séquence d'actions menant au but du problème quel que soit le scénario initial possible du monde, et ce, sans incorporer d'observation dans le plan. Cette faculté est particulièrement intéressante dans le domaine de la robotique autonome. En effet, les observations sont coûteuses en énergie et en temps lors de leurs exécutions en cours de mission par le robot. Malheureusement, il existe des problèmes pour lesquels une telle séquence d'actions n'existe pas et pour lesquels l'incertitude n'est pas traitable sans l'ajout d'observations. La planification contingente permet justement de traiter les problèmes sous incertitude en calculant un graphe de planification incorporant des observations de l'environnement. Ce cadre de planification semble particulièrement adapté aux missions de robotique autonome de par l'embarquabilité aisée des plans contingents. Cependant, la planification contingente possède tout de même une complexité EXPSPACE lorsque les opérateurs sont déterministes et que l'observabilité est partielle, ce qui signifie qu'un plan contingent est exponentiel dans la taille des variables inconnues du système. Afin de rendre ces variables connues, il est nécessaire d'ajouter des observations, ce qui implique le calcul de nouvelles branches du plan augmentant la taille du plan contingent. Malgré cela et le coût potentiel de l'exécution d'une observation pendant la mission, il semble qu'aucun planificateur contingent de la littérature n'a pour but de limiter le nombre de ces observations. L'idée développée dans ce chapitre est de mélanger la planification conformante et la planification contingente afin que les avantages de l'une compensent les inconvénients de l'autre. Pour cela, nous avons développé une méthode contingente pouvant se résumer en trois étapes :

— Utiliser un planificateur conformant pour calculer un plan conformant si possible.
— Si le calcul d'un tel plan n'est pas possible, utiliser les informations retournées par le planificateur conformant pour décider quelle observation réaliser.
— Rappeler le planificateur conformant afin de calculer des branches conformantes résolvant les sous-problèmes générés par l'application de l'observation.

5.2 Mise en oeuvre

La première étape de ce travail a été de définir le formalisme dans lequel se place notre approche. Nous avons décidé de nous inspirer grandement du formalisme habituellement utilisé en planification contingente et présenté dans la section 3.2.2 de ce manuscrit. Dans la section 5.2.1, nous allons rappeler les différentes définitions introduites dans les définitions du formalisme de la planification contingente et conformante. La deuxième étape de ce travail à été de définir un algorithme contingent générique détaillant l'utilisation d'un planificateur conformant pour le calcul des branches contingentes (section 5.2.2). La troisième étape a été de définir quel planificateur conformant utiliser pour notre processus. Dans la section 5.2.3 nous allons expliquer pourquoi nous avons choisi CPCES, puis nous détaillerons son fonctionnement dans la section 5.2.4. La dernière étape d'implémentation de l'approche a consisté à réaliser une procédure permettant d'extraire l'observation qu'il est nécessaire de réaliser à partir des informations retournées par CPCES en cas d'échec de calcul d'un plan conformant (section 5.2.5). Dans la section 5.2.6, nous illustrerons l'architecture de notre approche. Par la suite, nous évaluerons l'approche de manière théorique (section 5.3), puis nous détaillerons les résultats obtenus lors de l'implémentation de notre approche sur un ensemble de benchmarks que nous décrirons (section 5.4). Finalement, nous conclurons cette approche (sections 5.5 et 5.6).

5.2.1 Rappel du formalisme

Dans cette section, nous allons réunir les définitions présentées dans le chapitre I de ce manuscrit permettant de définir le cadre de la planification contingente utilisé dans cette thèse.

Définition: État

Un **état** s est un ensemble de fluents de F avec $s \subseteq F$. On nomme S l'ensemble fini des états possibles.

Définition: État de croyance

On va définir un état de croyance comme l'ensemble des états possibles. On notera $\mathcal{B}_0$ l'état de croyance initial et $\mathcal{B}$ l'état de croyance courant.

Définition: Action

On définit une action comme un couple $a = (Pre(a), Eff(a))$ avec :
- $Pre(a)$ un ensemble de préconditions
- $Eff(a)$ un ensemble d'effets.

Chaque effet $e \in Eff$ est défini comme un triplet $Cond(e) \subseteq F$, $Add(e) \subseteq F$, $Del(e) \subseteq F$ correspondants respectivement aux conditions nécessaires à l'exécution de l'effet, à un ensemble d'additions et un ensemble d'effacements.

Une action a est applicable dans un état s ssi $Pre(a) \subseteq s$. Le résultat de l'application de a dans s est un nouvel état défini par la fonction de transition T :

$$T(s,a) = s - \bigcup_{e \in Eff(a) \text{ s.t. } Con(e) \subseteq s} Del(e) \cup \bigcup_{e \in Eff(a) \text{ s.t. } Con(e) \subseteq s} Add(e) \tag{5.1}$$

Une action a est applicable dans un état de croyance $\mathcal{B}$ ssi $\forall s \in \mathcal{B}\ Pre(a) \subseteq s$. Le résultat de l'application de a dans $\mathcal{B}$ est un nouvel état de croyance défini par la fonction de transition $T(\mathcal{B}, a)$:

$$T(\mathcal{B}, a) = \{T(s,a), \forall s \in \mathcal{B}\} \tag{5.2}$$

Définition: Observation

Tout comme une action classique, une action d'observation correspond à un couple (Pre, Eff), cependant elle diffère d'une action classique par ses conditions d'application et ses effets.
Soit l'ensemble d'observations Ω, une action d'observation $\sigma \in \Omega$ est applicable dans un état de croyance $\mathcal{B}$ ssi $\forall s \in \mathcal{B}, Pre(\sigma) \subseteq s$. Soit $p(\sigma) \in F$ la proposition observée par l'action d'observation σ, $Eff(\sigma) = \nu(p)$ correspond à la valeur de vérité observée de la proposition $p(\sigma)$. L'application d'une action d'observation σ à un état de croyance $\mathcal{B}$ va éliminer de $\mathcal{B}$ les états ne contenant pas la valeur de vérité de la proposition observée $Eff(\sigma)$. Pour chaque $\sigma \in \Omega$, on a

$$T(\mathcal{B}, \sigma) = \{s,\ s.t\ s \in \mathcal{B} \wedge Eff(\sigma) \in s\} \tag{5.3}$$

Définition: Domaine

Un domaine de planification contingente dans F est un système de transition d'états $\Sigma_\Omega = (S, A, \Omega, T)$ tel que :
- S est l'ensemble des états possibles avec $S \subseteq 2^F$
- A est l'ensemble des actions
- Ω est l'ensemble des actions d'observation
- $T(\mathcal{B}, a)$ est la fonction de transition définie par l'équation 5.2 avec $a \in A$
- $T(\mathcal{B}, \sigma)$ est la fonction de transition définie par l'équation 5.3 avec $\sigma \in \Omega$.

Définition: Problème

Un problème de planification contingente est un triplet $\mathcal{P}_\Omega = (\Sigma_\Omega, \mathcal{B}_0, g)$ dans lequel :
- Σ_Ω est un domaine contingent
- $\mathcal{B}_0 \in S$ est l'ensemble des états initiaux possibles
- g est un ensemble de fluents dans F représentant le but

Définition: Plan contingent

Un plan contingent est un graphe de décision dans lequel les branches (des séquences d'actions) sont conditionnées par le résultat d'une observation de l'environnement.
Un plan contingent est solution d'un problème $\mathcal{P}_\Omega = (\Sigma_\Omega, \mathcal{B}_0, g)$ s'il est exécutable dans $\mathcal{B}_0$ et si chaque chemin du plan mène au but g

5.2.2 Définition du Processus Contingent

À partir de notre idée générale de processus contingent basé sur un planificateur conformant, on a pu définir un premier processus générique illustré dans l'algorithme 2. Dans la suite de ce manuscrit, nous nommerons cette approche ConTingent planner using a ConFormant planner (CTCF).

Algorithm 2 ContingentPlanning Procedure

Require: $\mathcal{P}_\Omega = (\Sigma_\Omega, \mathcal{B}_0, g)$
Ensure: π_c
1: $\mathcal{P} = (\Sigma, \mathcal{B}_0, g)$
2: $\pi, info :=$ conformantPlanner($\mathcal{P}$)
3: **if** π $conformant$ **then**
4: **return** π
5: $\mathcal{B}_\sigma, \sigma :=$ findObservation($\mathcal{P}_\Omega, info$)
6: $\pi_o :=$ ContingentPlanning($(\Sigma_\Omega, \mathcal{B}_0, \mathcal{B}_\sigma)$)
7: $\mathcal{B}^+ := T(\mathcal{B}_\sigma, \sigma)$ $with$ $eff(\sigma) = \top$
8: $\pi_p :=$ ContingentPlanning($(\Sigma_\Omega, \mathcal{B}^+, g)$)
9: $\mathcal{B}^- := T(\mathcal{B}_\sigma, \sigma)$ $with$ $eff(\sigma) = \bot$
10: $\pi_n :=$ ContingentPlanning($(\Sigma_\Omega, \mathcal{B}^-, g)$)
11: **return** (π_o ;**if** σ **then** π_p **else** π_n)

L'idée générale de notre processus contingent est tout d'abord de transformer le problème contingent initial $\mathcal{P}_\Omega$ en un problème conformant $\mathcal{P}$ (ligne 1), puis d'appeler un planificateur conformant à partir de ce problème afin de calculer un plan conformant (ligne 2). La transformation du problème initial contingent en un problème conformant est réalisée en pratique simplement en supprimant les observations du domaine et en ne considérant qu'un ensemble d'actions. Si un tel plan existe, alors on le retourne comme solution du problème. Dans le cas contraire, on utilise l'information retournée par le planificateur conformant afin de déterminer quelle observation est nécessaire et dans quel état de croyance la réaliser afin de lever l'incertitude posant problème au planificateur conformant (ligne 5). Le processus contingent est ensuite appelé récursivement afin de trouver un plan contingent ou conformant menant à l'état de croyance dans lequel on doit réaliser l'observation à partir de l'ensemble des états initiaux possibles. Pour cela, on modifie le but du problème d'entrée par l'état de croyance dans lequel l'observation peut être réalisée (ligne 6). L'observation est ensuite utilisée pour générer deux sous-problèmes dans lesquels l'ensemble d'états initiaux possibles représente une partie seulement des états initiaux possibles du problème initial, respectivement l'ensemble des états initiaux possibles contenant la proposition observée et dont la valeur est positive, et l'ensemble des états initiaux contenant la proposition observée et dont la valeur est négative (lignes 7 et 9). Enfin, le processus est ensuite relancé pour calculer deux plans partant de ces deux sous-problèmes et menant au but, permettant d'atteindre le but du problème quel que soit le résultat de l'observation (lignes 8 et 10). Ces trois plans et l'action d'observation sont ensuite assemblés et retournés pour former le plan contingent final solution du problème.

Étant donné que nous voulions développer une méthode permettant de traiter le plus de problèmes possibles, il nous est apparu naturel de choisir le langage PDDL comme langage de spécification des problèmes, celui-ci étant déjà utilisé pour implémenter de nombreux problèmes académiques dont des problèmes modélisant l'incertitude sous forme d'ensemble d'états initiaux possibles. Notre approche prend donc en entrée le fichier domaine et le fichier problème PDDL modélisant le problème à résoudre.

Exemple: Résolution d'un exemple par la procédure contingente

Reprenons l'ensemble d'états initiaux $\mathcal{B}_0$ de la figure 3.3 de la section 3.2.2 du chapitre 3 de l'état de l'art. Nous rappelons cet ensemble d'état initiaux dans la figure 5.1 ci-dessous.

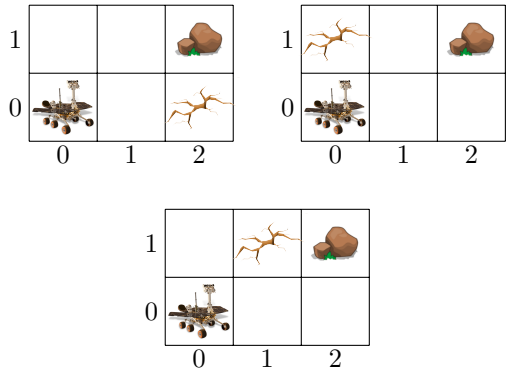

FIGURE 5.1 – Exemple d'ensemble d'états initiaux possibles

Nous pouvons nous rendre compte que dans l'état de croyance de la figure 5.1, l'incertitude sur la position de l'obstacle en position (2,0) ou en position (1,1) implique qu'aucun plan conformant n'est valable pour ces deux positions d'obstacles, celles-ci rendant le déplacement du robot jusqu'au rocher impossible avec une séquence d'actions.
La réalisation d'une observation de l'une de ces deux cases est nécessaire afin de séparer les états initiaux en deux problèmes potentiellement traitables par un planificateur conformant. Une observation candidate est celle présentée dans la figure 3.4 de la section 3.2.2 du chapitre de l'Etat de l'art de ce manuscrit : l'observation $< obs\_obst\ rob\ cell10\ cell11 >$. L'intégration de cette observation au plan solution du problème nécessite le calcul de trois sous-plans. Le premier sous-plan π_σ est celui menant de l'ensemble des états initiaux possibles $\mathcal{B}_0$ à l'état de croyance $\mathcal{B}_\sigma$ dans lequel l'observation est réalisable. Cet état de croyance $\mathcal{B}_\sigma$ et l'observation candidate σ (ici $< obs\_obst\ rob\ cell10\ cell11 >$) sont retournés par la procédure FINDOBSERVATION, qui sera décrite dans la suite de ce chapitre. Dans la figure 5.2, on peut se rendre compte assez facilement que le planificateur conformant utilisé peut calculer un plan conformant π_σ pour relier $\mathcal{B}_0$ et $\mathcal{B}_\sigma$.

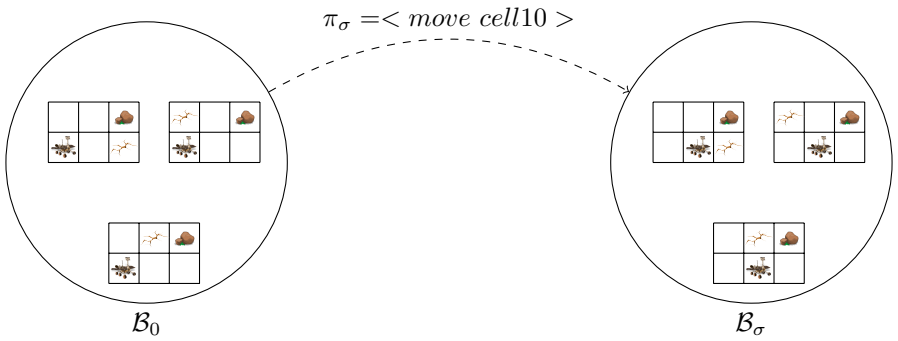

FIGURE 5.2 – Calcul du plan menant à $\mathcal{B}_\sigma$

Une fois que le planificateur conformant a calculé le plan π_σ menant à l'observation souhaitée, il ne reste plus qu'à appliquer l'observation pour séparer l'état de croyance courant $\mathcal{B}_\sigma$ en deux

états de croyance distincts $\mathcal{B}^+$ et $\mathcal{B}^-$. Ces deux états de croyance correspondent aux états de croyance obtenus considérant les deux résultats possibles de l'observation. L'application de l'observation est illustrée dans la figure 5.3.

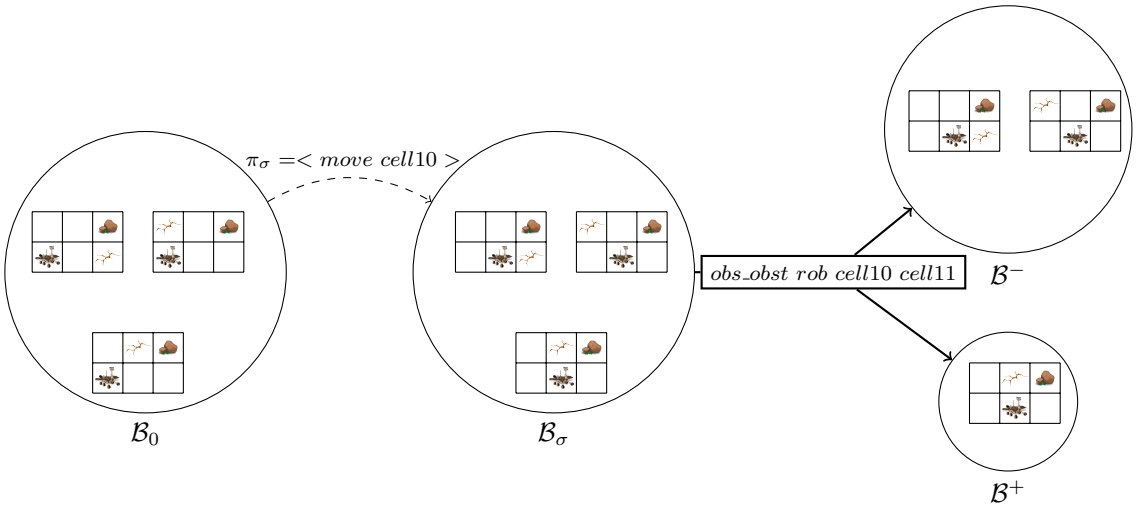

FIGURE 5.3 – Application de l'observation à $\mathcal{B}_\sigma$

Afin de résoudre le problème initial, il ne reste plus qu'à calculer un sous-plan menant de $\mathcal{B}^+$ à g et un sous-plan menant de $\mathcal{B}^-$ à g. Dans notre exemple, le planificateur conformant parvient à calculer un plan conformant π_+ menant de $\mathcal{B}^+$ à g et un plan π_- menant de $\mathcal{B}^-$ à g (figure 5.4).

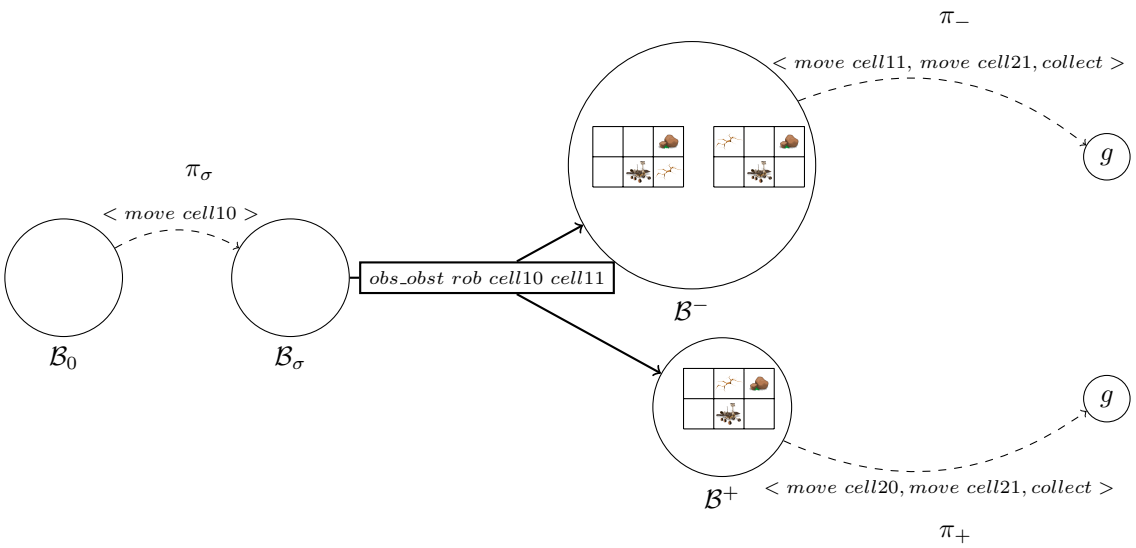

FIGURE 5.4 – Calcul des branches menant de $\mathcal{B}^+$ et $\mathcal{B}^-$ à g

5.2.3 Choix du planificateur conformant

Une étape essentielle de ce travail a consisté à trouver un planificateur conformant intégrant des caractéristiques permettant la communication externe d'informations suffisamment exploitables pour nous permettre de déterminer quelles observations nous devons ajouter au plan afin de réduire l'incertitude et permettre aux sous-problèmes générés d'être conformants.

Parmi les planificateurs conformants étudiés et introduits dans la section 3.2.1 de l'état de l'art de ce manuscrit (Palacios et Geffner, 2006; Albore et collab., 2011; Grastien et Scala, 2017; To et collab., 2009;

Hoffmann et Brafman, 2006), nous avons choisi d'utiliser le planificateur conformant CPCES (Grastien et Scala, 2017) et de travailler avec Alban Grastien pour le développement de notre planificateur contingent.

Plusieurs raisons nous ont poussés à choisir CPCES. Premièrement, CPCES a la particularité de calculer des plans généralement plus courts que les autres planificateurs conformants connus, ce qui, en utilisant CPCES pour calculer les branches d'un plan contingent, permettrait théoriquement d'obtenir des plans contingents plus compacts. Deuxièmement, les résultats de l'utilisation de CPCES sur les benchmarks académiques montrent que CPCES résout des problèmes complexes en un temps de calcul compétitif, la compétitivité de ce temps de calcul reposant particulièrement sur l'échantillonnage intelligent de l'ensemble d'états de croyance initial permettant de ne pas explorer l'intégralité des états initiaux possibles. CPCES a la particularité de pouvoir résoudre des problèmes de largeur conformante supérieure à 1, permettant le traitement de problèmes plus complexes. Enfin, le point le plus important pour nous est que CPCES est capable de retourner des informations potentiellement exploitables afin de déterminer quelles observations sont nécessaires pour diminuer l'incertitude du problème, ces informations étant un plan conformant sur une partie des états initiaux et un état de contre-exemple pour lequel ce plan précédemment calculé par CPCES n'est pas valide.

5.2.4 Fonctionnement de CPCES

CPCES est un planificateur conformant dont le but est de calculer un plan conformant (si un tel plan existe) menant de l'ensemble des états initiaux possibles $\mathcal{B}_0$ au but g du problème. CPCES prend en entrée un problème de planification conformante $\mathcal{P} = (\Sigma, \mathcal{B}_0, g)$ et retourne soit un plan conformant, soit un plan conformant seulement sur un sous-ensemble des états initiaux possibles ainsi qu'un état de contre-exemple γ pour lequel ce plan ne mène pas au but g. Le fonctionnement de CPCES est illustré par l'algorithme 3.

Algorithm 3 CPCES

Require: $\mathcal{P} = (\Sigma, \mathcal{B}_0, g)$
Ensure: π, γ
1: $\mathcal{B} := \emptyset$
2: $\pi := \varepsilon$
3: **loop**
4: **if** π is a solution for $\mathcal{P}$ **then**
5: **return** $\pi, \emptyset$
6: let γ be a counter-example $\in \mathcal{B}_0$
7: $\mathcal{B} := \mathcal{B} \cup \{\gamma\}$
8: compute a new plan π' for $\mathcal{P}' = (\Sigma, \mathcal{B}, g)$
9: **if** no such π' exists **then**
10: **return** π, γ
11: $\pi := \pi'$

CPCES calcule itérativement un plan π (sous la forme d'une séquence d'actions) menant au but g du problème depuis l'état de croyance courant $\mathcal{B}$. $\mathcal{B}$ est tout d'abord réduit à un ensemble vide (ligne 1). CPCES vérifie ensuite si le plan π calculé à partir de $\mathcal{B}$ (ou le plan vide ε dans un premier temps) mène à g considérant l'état de croyance initial complet $\mathcal{B}_0$ (ligne 4). Si c'est le cas, alors π est retourné comme solution au problème conformant (ligne 5). Dans le cas contraire, un état de contre-exemple $\gamma \in \mathcal{B}_0$ pour lequel π ne mène pas à g est ajouté à l'état de croyance courant $\mathcal{B}$ (ligne 7), et le processus redémarre à partir de ce nouvel état de croyance courant $\mathcal{B}$. Un nouveau plan est calculé à partir de ce nouvel état de croyance (ligne 8) jusqu'à ce qu'un plan conformant soit calculé, ou bien qu'aucune solution conformante ne puisse être calculée à partir de $\mathcal{B}$. Si c'est le cas, alors le dernier contre-exemple extrait est retourné avec le plan échouant sur ce contre-exemple (ligne 10). Le calcul de plan à partir de l'état de croyance courant est réalisé par le planificateur classique FF, et la vérification de la validité de ce plan est réalisée par le SAT-Solveur Z3 (De Moura et Bjørner, 2008).

La particularité de CPCES résulte dans le fait que celui-ci calcule un plan conformant en utilisant seulement un sous-ensemble des états initiaux possibles. La procédure de recherche de contre-exemple permet à CPCES de réaliser un échantillonnage intelligent de l'espace de croyance, chacun des contre-

exemples étant pertinent en contredisant au moins un des plans précédemment calculés. Cette particularité permet à CPCES de réduire la complexité de calcul du plan en réduisant le nombre d'états considérés pour ce calcul.

Etant donné que le langage d'entrée de CPCES est le PDDL, l'algorithme contingent 2 transmet le problème conformant $\mathcal{P}$ à CPCES sous la forme d'un fichier problème et un fichier domaine PDDL. On peut noter que l'on vérifie si le plan retourné par CPCES est conformant simplement en vérifiant que le contre-exemple retourné par celui-ci est vide.

Exemple: Fonctionnement de CPCES sur un exemple conformant et un exemple contingent

Prenons tout d'abord l'exemple d'états initiaux possibles conformant de la figure 3.1.

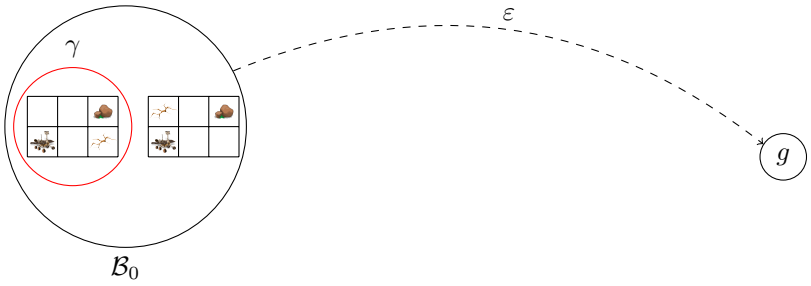

FIGURE 5.5 – Vérification du plan vide solution

Dans un premier temps, CPCES vérifie si le plan vide est solution de l'ensemble des états initiaux possibles $\mathcal{B}_0$ (figure 5.5). Ici, ce n'est pas le cas et un état de contre-exemple γ est sélectionné de manière aléatoire parmi l'ensemble des états initiaux pour lesquels le plan vide ne mène pas au but. Ce contre-exemple est ajouté à l'ensemble des états contre-exemples considérés $\mathcal{B}$ (figure 5.6).

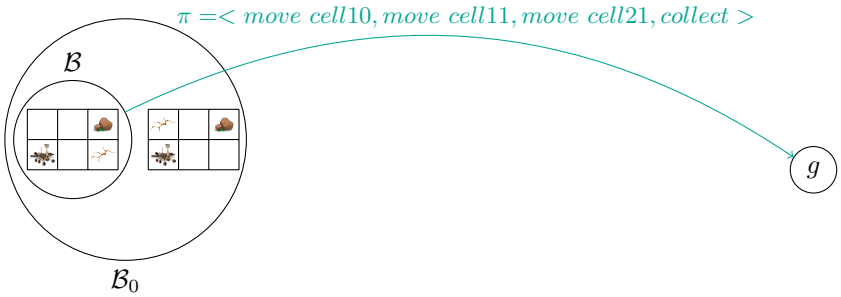

FIGURE 5.6 – Calcul d'un plan menant de $\mathcal{B}$ à g

CPCES tente ensuite de calculer un plan π menant de l'ensemble des états contre-exemples $\mathcal{B}$ au but g du problème (figure 5.6). Ici, CPCES réussit à calculer le plan $\pi =< move\ cell10, move\ cell11, move\ cell21, collect >$.

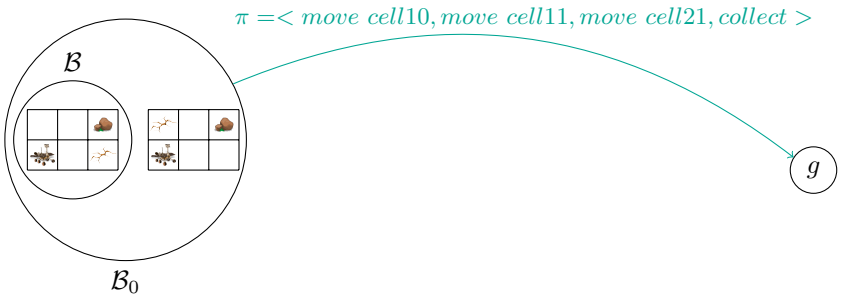

FIGURE 5.7 – Vérification du plan menant de $\mathcal{B}$ à g

La dernière étape de CPCES est de vérifier si le plan π calculé pour l'ensemble d'états contre-exemples $\mathcal{B}$ est valide pour l'ensemble des états initiaux possibles $\mathcal{B}_0$ (figure 5.7). Dans notre exemple, π est bien valide pour l'ensemble des états initiaux possibles, celui-ci est donc conformant et retourné par CPCES.

Intéressons nous maintenant au comportement de CPCES lorsqu'un plan conformant n'existe pas pour le problème à résoudre. Pour cela, reprenons l'ensemble d'états initiaux de la figure 3.3.

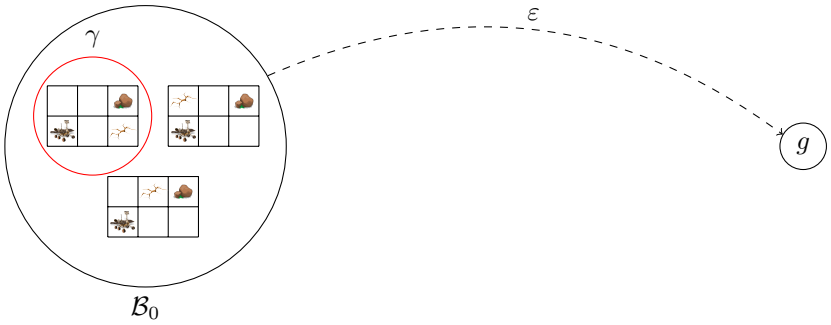

FIGURE 5.8 – Vérification du plan vide solution

Comme pour le problème conformant ci-dessus, CPCES vérifie si le plan vide mène au but du problème g quel que soit l'état initial possible de $\mathcal{B}_0$ (figure 5.8). Dans notre exemple ce n'est pas le cas, CPCES sélectionne aléatoirement un état initial γ de $\mathcal{B}_0$ pour lequel le plan vide ne mène pas à g. Ce contre-exemple est ensuite ajouté à un ensemble $\mathcal{B}$ à partir duquel CPCES tente de calculer un plan π menant à g.

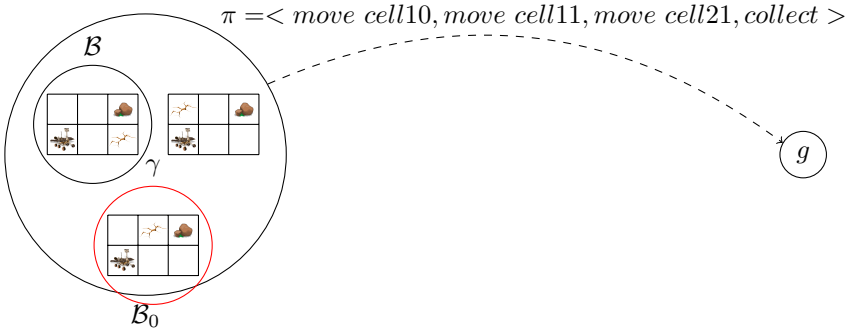

FIGURE 5.9 – Vérification du premier plan calculé à partir de $\mathcal{B}$

Une fois que CPCES a calculé un plan π menant de $\mathcal{B}$ à g, CPCES vérifie si π mène à g depuis n'importe quel état de $\mathcal{B}_0$ (figure 5.9). Ce n'est pas le cas ici, il existe un état de $\mathcal{B}_0$ pour lequel

π ne mène pas à g. CPCES ajoute cet état à l'ensemble d'états de contre-exemple $\mathcal{B}$ et tente de calculer un nouveau plan menant à g à partir de $\mathcal{B}$.

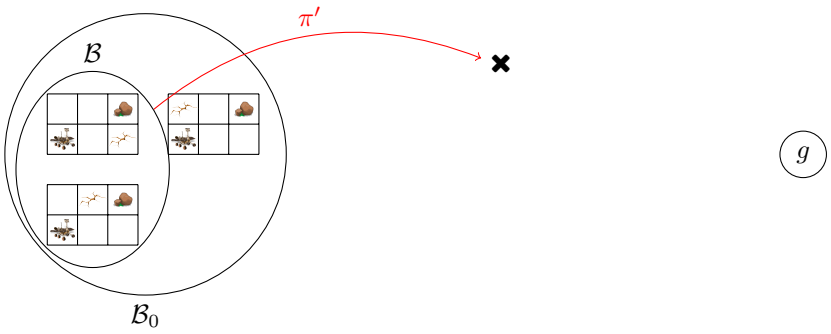

FIGURE 5.10 – Echec de calcul d'un plan à partir de $\mathcal{B}$

Dans le cas illustré par la figure 5.10, CPCES n'est pas en mesure de calculer un plan menant à g depuis l'ensemble d'états de contre-exemple $\mathcal{B}$ car aucun plan π' n'existe à partir des deux états initiaux de $\mathcal{B}$ pour ce problème. Afin de calculer un plan à partir de $\mathcal{B}$, une observation doit être ajoutée pour réduire l'incertitude, la solution du problème est donc contingente. Dans ce cas, CPCES retourne le dernier contre-exemple ajouté à $\mathcal{B}$ ainsi que le plan π calculé à l'étape précédente (illustrée par la figure 5.9).

5.2.5 Extraction de l'observation à partir des informations retournées par CPCES

Maintenant que nous avons vu comment notre processus contingent utilise CPCES pour calculer les branches du plan contingent, nous pouvons nous intéresser à la procédure de sélection de l'observation à réaliser, et de l'état de croyance dans lequel cette observation est réalisable.
Après avoir fait appel à CPCES, deux situations sont envisageables, soit CPCES retourne un plan conformant que nous pouvons retourner comme solution du problème en entrée du processus contingent, soit CPCES retourne un plan conformant pour seulement une partie de l'ensemble d'états initiaux possibles et un état contre-exemple contenu dans l'ensemble des états initiaux possibles pour lequel ce plan n'est pas applicable. On considère que le plan retourné par CPCES en cas d'échec a de grandes chances d'être proche du plan solution du problème étant donné que celui-ci est conformant pour une partie des états initiaux possibles. C'est en se basant sur cette hypothèse que nous avons décidé de déterminer quelle action du plan échoue dans l'état de contre-exemple, afin d'en observer les préconditions insatisfiables du fait de la valeur de vérité incertaine de ces préconditions. L'observation d'une des préconditions de l'action qui n'est pas applicable va permettre de discriminer l'incertitude sur ces préconditions, c'est à dire de savoir si l'agent se situe dans un état dans lequel cette proposition est vraie ou fausse.

La procédure de sélection de l'observation nécessaire et de son état de croyance d'exécution est la procédure FINDOBSERVATION illustrée par l'algorithme 4.

Dans un premier temps, on identifie quelle action du plan π n'est pas applicable dans l'état de contre-exemple γ. Pour cela, on exécute le plan π à partir de γ jusqu'à ce que l'action courante ne soit pas applicable (ligne 5). On applique aussi chaque action à l'état de croyance $\mathcal{B}$ courant (ligne 8) afin de garder une trace des différents états de croyance calculés par l'application de π à l'état de croyance initial $\mathcal{B}_0$ (ligne 9). Lorsque l'action non applicable est identifiée, on garde dans la variable $unsatPre$ les préconditions non satisfiables dans γ afin de déterminer lesquelles sont observables (ligne 11).

On sélectionne toutes les observations capables d'observer une des propositions contenues dans les préconditions (ligne 14). Pour chaque observation sélectionnée, on vérifie si elle est applicable dans le dernier état de croyance calculé (ligne 17). Si c'est le cas, on retourne l'observation et l'état de croyance dans lequel elle est applicable. Si ce n'est pas le cas, on vérifie si l'observation est applicable dans un autre des états de croyance calculés. Si l'observation n'est applicable dans aucun des états

Algorithm 4 findObservation

Require: $\mathcal{P}_\Omega, \pi, \gamma$
Ensure: $(\mathcal{B}_\sigma, \sigma)$
1: $beliefList := [\mathcal{B}_0]$
2: $\mathcal{B} := \mathcal{B}_0$
3: **for** a in π **do**
4: **if** a applicable in γ **then**
5: $\gamma := T(\gamma, a)$
6: **if** a applicable in $\mathcal{B}$ **then**
7: **for** $s \in \mathcal{B}$ **do**
8: $\mathcal{B}' := \{T(s, a) | s \in \mathcal{B}\}$
9: $beliefList := beliefList + \mathcal{B}'$
10: **else**
11: let $unsatPre$ be the unsatisfied preconditions of a
12: **break**
13: **for** p in $unsatPre$ **do**
14: let $\Omega_p \in \Omega$ be the set of observations for p
15: **for** $\sigma \in \Omega_p$ **do**
16: **for** $\mathcal{B}$ in $beliefList$ **do**
17: **if** σ applicable in $\mathcal{B}$ **then**
18: **return** $(\mathcal{B}, \sigma)$
19: **return** $(\mathcal{B}, None)$

de croyance calculés, alors on passe à l'observation suivante jusqu'à ce qu'une des observations soit applicable ou qu'aucune observation ne soit disponible, auquel cas nous considérons qu'il n'y a pas de solution contingente. Dans notre approche, afin de trouver une observation à réaliser, nous avons décidé de parcourir les états de croyance à reculons, du dernier calculé jusqu'à $\mathcal{B}_0$. Ce choix découle de l'hypothèse optimiste qu'une observation est en général applicable au plus près de la proposition à observer, et donc de l'échec du plan précédemment calculé par CPCES. Nous avons fait ce choix en nous basant sur des exemples de la réalité dans lesquels il est souvent nécessaire d'être à proximité d'un objet afin de l'observer.

Exemple: Extraction de l'observation dans l'exemple

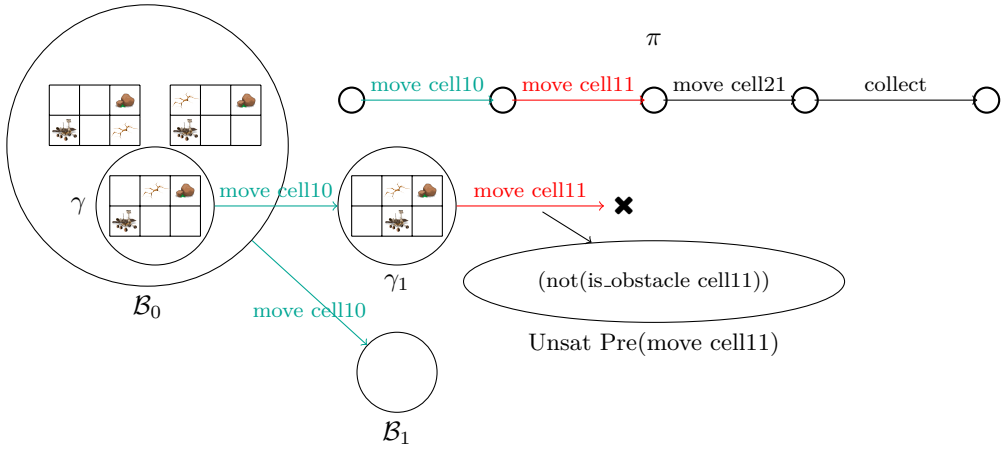

Figure 5.11 – Identification des prédicats de l'action du plan échouant

La figure 5.11 illustre les lignes 3 à 12 de l'algorithme 4. Cette partie de l'approche consiste à déterminer quelle action du plan π ne peut pas être appliquée au contre-exemple γ, puis à en extraire les préconditions ne pouvant pas être satisfaites dans γ. Dans cet exemple, on applique

les actions de π au contre-exemple et à l'état de croyance initial $\mathcal{B}_0$ jusqu'à l'action $< move\ cell11 >$ qui n'est pas applicable dans γ_1. Ici, la précondition de l'action $< move\ cell11 >$ n'étant pas satisfaite dans γ_1 est $(not(is\_obstacle\ cell11))$ puisque l'obstacle est bien présent en case (1,1) dans γ_1. L'ensemble des préconditions non satisfaites dans γ_1 ainsi que les états de croyance calculés $\mathcal{B}_0$ et $\mathcal{B}_1$ sont extraits pour la suite de la procédure.

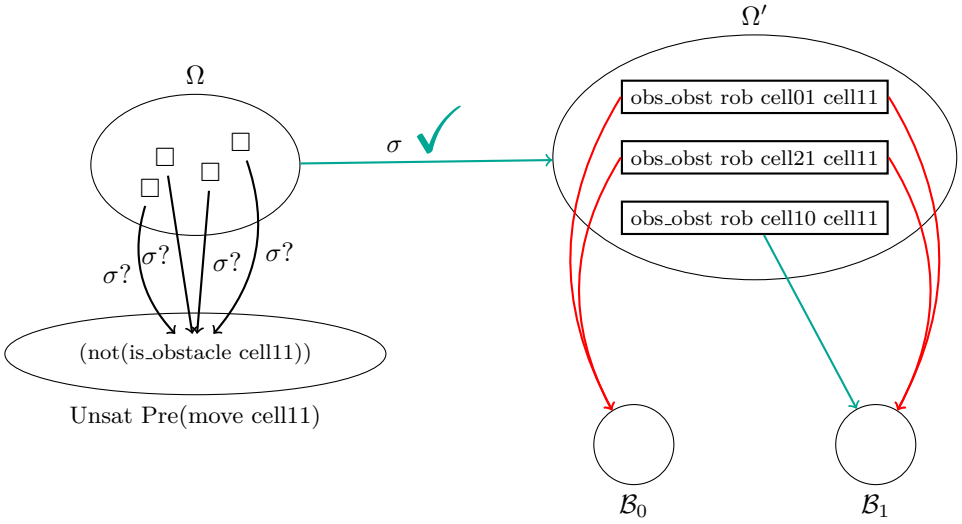

FIGURE 5.12 – Identification des observations pouvant observer les préconditions

La figure 5.12 illustre le processus de sélection de l'observation à réaliser (lignes 13 à 18 de l'algorithme 4). Dans un premier temps, cette partie de l'approche consiste à récupérer les observations de Ω capables d'observer les préconditions à observer. Dans un deuxième temps, on vérifie si une de ces observations extraites dans Ω' est applicable dans un des états de croyance calculés à l'étape précédente (illustrée par la figure 5.11). Dès que l'observation testée est applicable dans un des états de croyance, alors cet état de croyance et cette observation sont retournés. Dans notre exemple, trois observations sont capables d'observer la proposition $(not(is\_obstacle\ cell11))$. Il s'agit des observations $< obs\_obst\ rob\ cell10\ cell11 >$, $< obs\_obst\ rob\ cell01\ cell11 >$ et $< obs\_obst\ rob\ cell21\ cell11 >$ qui correspondent à des observations de la position de l'obstacle en considérant que le robot se trouve respectivement en case (1,0), (0,1) ou (2,1). Parmi ces observations, il existe une seule observation applicable dans $\mathcal{B}_1$, $< obs\_obst\ rob\ cell10\ cell11 >$ car dans $\mathcal{B}_1$, le robot se trouve en case (1,0). Cette observation est donc retournée ainsi que $\mathcal{B}_1$.

5.2.6 Architecture

On peut illustrer l'architecture générale de notre approche et les différentes informations échangées entre les trois composants principaux de notre approche, c'est à dire la fonction de recherche d'observation, CPCES, et le processus contingent par la figure 5.13.

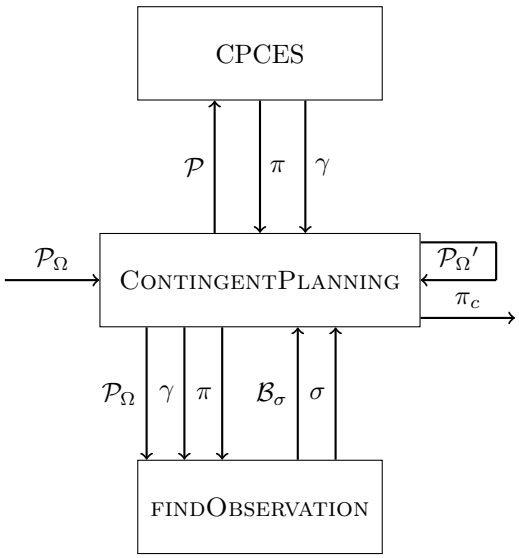

Figure 5.13 – Architecture du processus contingent

Dans cette illustration de l'architecture de l'approche, on identifie clairement les échanges entre procédures. La procédure contingente interagit avec CPCES en lui fournissant un problème conformant $\mathcal{P}$, puis CPCES lui retourne soit un plan conformant π et un contre-exemple vide, soit un plan π non conformant et un contre-exemple γ à ce plan. Si la procédure contingente reçoit un plan conformant de CPCES alors celui-ci le retourne, sinon la procédure contingente transmet le problème $\mathcal{P}_\Omega$, ainsi que le plan π et le contre-exemple γ retourné par CPCES à la procédure findObservation. Avec ces informations, la procédure findObservation retourne une observation σ et l'état de croyance $\mathcal{B}_\sigma$ dans lequel elle doit être réalisée à la procédure contingente. À partir de ces informations, la procédure contingente crée un nouveau problème $\mathcal{P}_\Omega{}'$ et est appelée récursivement à partir de ce nouveau problème jusqu'à trouver un plan contingent π_c solution du problème initial $\mathcal{P}_\Omega$.

5.3 Analyse théorique

5.3.1 Complétude

Un algorithme est dit complet si celui-ci retourne une solution dès lors qu'il en existe une. On peut remarquer en analysant les différents algorithmes présentés dans ce chapitre que notre approche n'est à ce stade pas complète. En effet si l'on regarde l'algorithme 4 on peut se rendre compte qu'en sélectionnant la première observation applicable, on rejette toutes les autres observations potentiellement utiles. Par conséquent, si finalement le plan calculé par la suite incorporant cette observation ne mène pas au but, ou si tout simplement l'état de croyance dans lequel nous devons réaliser cette observation n'est pas atteignable par un plan conformant ou contingent, alors nous n'avons aucun moyen de revenir sur notre choix d'observation et de choisir une autre observation utile, ce qui réduit le nombre de problèmes pouvant être traités par notre approche à ce stade. De plus, s'il existe des problèmes dans lesquels notre hypothèse selon laquelle le plan retourné par CPCES est proche de la solution finale est erronée, c'est à dire si le plan retourné par CPCES ne nous aide pas à discriminer le contre-exemple, et donc aucune des observations jugées pertinentes au regard du plan défaillant ne permettent finalement d'isoler le contre-exemple des autres états initiaux possibles, il n'existe aucun mécanisme de secours. Dans ces deux cas de figure, notre algorithme considère qu'il n'existe aucune solution, alors qu'en réalité une solution existe bel et bien. À cause de ce mécanisme, notre algorithme n'est pas complet.

5.3.2 Complexité

Dans cette section, nous allons nous intéresser à la complexité des algorithmes développés dans cette approche. Afin de simplifier les calculs, nous considérons l'application d'une action, la vérification et la sélection d'une observation, ou encore l'ajout d'un élément dans une liste comme une opération unitaire.

De plus, afin d'obtenir un ordre de grandeur, nous sur-approximons les équations en ne gardant que les variables possédant le plus grand ordre de grandeur.

Commençons d'abord par l'analyse de la fonction FINDOBSERVATION(algorithme 4). Nous négligeons les opérations unitaires du début de fonction permettant d'initialiser les variables.

Dans un premier temps, la fonction effectue une première boucle dans laquelle nous avons $|\pi|$ applications d'une action à un état de croyance. Étant donné que dans notre approche, nous avons codé les états de croyance de manière explicite, les actions doivent être appliquées dans le pire des cas à $2^{|F|}$ états de l'état de croyance. Nous avons donc $|\pi| \times 2^{|F|}$ opérations dans cette première boucle. Pour la deuxième partie de l'algorithme débutant ligne 13, nous avons $|unsatPre| \times |beliefList| \times |\Omega|$ opérations sur des états de croyance. Dans le pire cas, $|UnsatPre| = |F|$, $|beliefList| = |\pi|$ et l'état de croyance $\mathcal{B}$ (ligne 17) est constitué de $2^{|F|}$ états. On aura donc, dans le pire cas, un nombre d'opérations proportionnel à : $|\pi| \times 2^{|F|} + |\pi| \times 2^{|F|} \times |F| \times |\Omega| \propto |\pi| \times |\Omega| \times |F| \times 2^{|F|}$.

La complexité algorithmique est en notation de Landau :
$\mathcal{O}(|\pi| \times |\Omega| \times |F| \times 2^{|F|}) = \mathcal{O}\left(\max\left(|\pi|, |\Omega|, |F|, 2^{|F|}\right)\right) = \mathcal{O}(2^{|F|})$.

Notons $\mathcal{C}(n)$ la complexité de CPCES prenant n états en entrée, $\mathcal{F}(n)$ la complexité de FINDOBSERVATION prenant n états en entrée, et $\mathcal{T}(n)$ la complexité de la procédure contingente prenant en entrée n états. La complexité de la procédure contingente peut être exprimée par l'équation :

$$\mathcal{T}(n) = \mathcal{C}(n) + \mathcal{F}(n) + \mathcal{T}'(n) + 2\mathcal{T}(n/2) \tag{5.4}$$

avec $\mathcal{T}'(n)$ la complexité de la procédure contingente appelée sur le sous-problème de calcul d'un plan menant à une observation (ligne 6), et $2\mathcal{T}(n/2)$ la complexité des appels de la procédure contingente pour le calcul des sous-plans π_p et π_n (lignes 8 et 10) [1].

La complexité de CPCES ($\mathcal{C}(n)$) domine les autres éléments de la complexité de $\mathcal{T}(n)$, la complexité de l'approche est donc proportionnelle à $\mathcal{O}(\mathcal{C}(n))$ et donc a une complexité dans le pire des cas de $\mathcal{O}(2^{|F|})$.

5.4 Expérimentations

L'implémentation de notre approche a été développée en Python. Nous avons décidé de comparer notre approche avec plusieurs planificateurs contingents de la littérature, Contingent-FF (Hoffmann et Brafman, 2005), CLG (Albore et collab., 2009), et PO-PRP (Muise et collab., 2014) sur trois classes de benchmarks : des problèmes de la littérature dont la plupart sont fournis par les auteurs de Contingent-FF et qui découlent des différentes compétitions de planification de la conférence ICAPS (International Conference on Automated Planning and Scheduling), des problèmes dont la largeur contingente est supérieure à 1, et des problèmes *coûteux*. La largeur contingente d'un problème est définie en termes de nombre maximum de variables dont les valeurs sont initialement inconnues, et dont l'incertitude doit nécessairement être levée afin de résoudre le problème. Les problèmes dont la largeur contingente est supérieure à 1 constituent une classe de problèmes difficile, la nécessité de devoir réaliser plusieurs observations pour discriminer l'incertitude empêchant parfois certains planificateurs comme CLG (Albore et collab., 2009) ou PO-PRP (Muise et collab., 2014) de calculer une solution saine. Cette classe de problème est donc tout particulièrement intéressante pour comparer les performances de notre approche à celles des planificateurs de l'état de l'art.

La classe des problèmes *coûteux* découle de notre volonté d'évaluer notre approche sur des problèmes dans lesquels les observations ont un coût plus important que les actions classiques, afin de simuler le coût potentiel que celles-ci peuvent avoir lorsqu'elles sont exécutées en ligne. Étant donné que CPCES et FF, sur lesquels repose notre approche, ne peuvent considérer l'ajout d'un coût numérique aux actions PDDL, nous avons décidé de modéliser ce coût par l'ajout d'une action supplémentaire aux domaines devant être réalisée avant chaque observation, permettant d'influer de manière plus significative sur la taille et la profondeur du plan calculé.

Afin d'étudier plus facilement le comportement de notre approche lorsque l'observation discriminant l'incertitude n'est pas applicable dans le chemin d'exécution du plan défaillant retourné par CPCES, nous avons décidé de créer un domaine très simple et facilement analysable nommé *laboratory*. Dans ce domaine, un agent doit nécessairement réaliser un détours du chemin direct menant au but pour exécuter une observation permettant de réduire l'incertitude.

L'intégralité des benchmarks considérés sont décrits en annexe de ce manuscrit.

1. On considère en moyenne que les sous-problèmes possèdent un nombre d'états équitablement distribués

5.4.1 Conditions d'expérimentation

L'ensemble des résultats ont été obtenus sur un ordinateur Intel Xeon(R) W-2123 CPU 3.60GHz x 8 doté de 7,5 Go de mémoire RAM. Afin de limiter le temps de calcul des plans, nous avons fixé une limite de temps de 10 minutes au delà de laquelle le processus est stoppé.

La validité des plans générés est controlée par une fonction parcourant le plan et appliquant chaque action et observation à chaque état initial possible afin de vérifier que le plan mène au but du problème quel que soit l'état initial du problème.

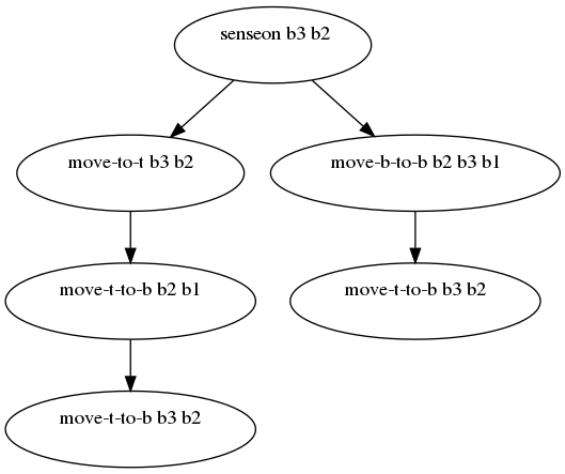

FIGURE 5.14 – Plan généré pour le problème p3 de blocks

Afin d'analyser les plans de manière plus visuelle, nous avons développé une fonction permettant de convertir le plan en image. La figure 5.14 donne un aperçu visuel de ce que donne un plan obtenu par cette fonction pour le problème *blocks/p3*.

5.4.2 Résultats

Dans un premier temps, nous allons nous intéresser aux problèmes de la littérature proposés lors des différentes compétitions de planification de la conférence ICAPS (International Conference on Automated Planning and Scheduling) et au problème *laboratory* développé pour cette thèse. Dans un deuxième temps, nous nous intéresserons aux problèmes de la littérature et aux problèmes modifiés possédant une largeur contingente supérieure à 1. Enfin, nous détaillerons les résultats sur des problèmes de la littérature modifiés afin que les observations de ces problèmes soient *coûteuses*.

Problèmes contingents de la littérature

L'ensemble détaillé des résultats de cette section se trouve dans le tableau B.1 situé en annexe de ce manuscrit. Dans la figure 5.15, nous pouvons distinguer quatre graphiques représentant respectivement le temps de calcul (en secondes), la taille (en nombre total d'actions et d'observations), la profondeur (taille de la plus longue séquence d'actions et d'observations), et le nombre total d'observations contenues dans le plan entier pour chaque problème. Dans ces graphiques, les problèmes sont triés de manière croissante selon un critère de difficulté que nous définissons comme le produit du nombre de proposition par état et du nombre d'états composant l'état de croyance initial.

Si on observe les graphiques dans leur ensemble, on peut se rendre compte que CTCF ne parvient pas à trouver une solution pour 13 des 49 problèmes étudiés dans le temps imparti. Pour les problèmes *erovers/p120* et *erovers/p500*, CTCF dépasse le temps imposé, tandis que pour les autres problèmes qu'il n'arrive pas à résoudre, celui-ci ne trouve tout simplement pas de solution. PO-PRP ne parvient pas à résoudre l'intégralité des problèmes de *logistics* et *colorballs*, pour lesquels celui-ci n'est pas sain et trouve des plans qui ne sont pas valides. PO-PRP ne parvient pas à trouver de solution pour les problèmes *erovers/p20, p40, p60 et p120*, tandis que pour les problèmes *ebtcs/p10 et p30*, celui-ci ne parvient pas à lire correctement le problème. Contingent-FF ne parvient pas à résoudre les problèmes

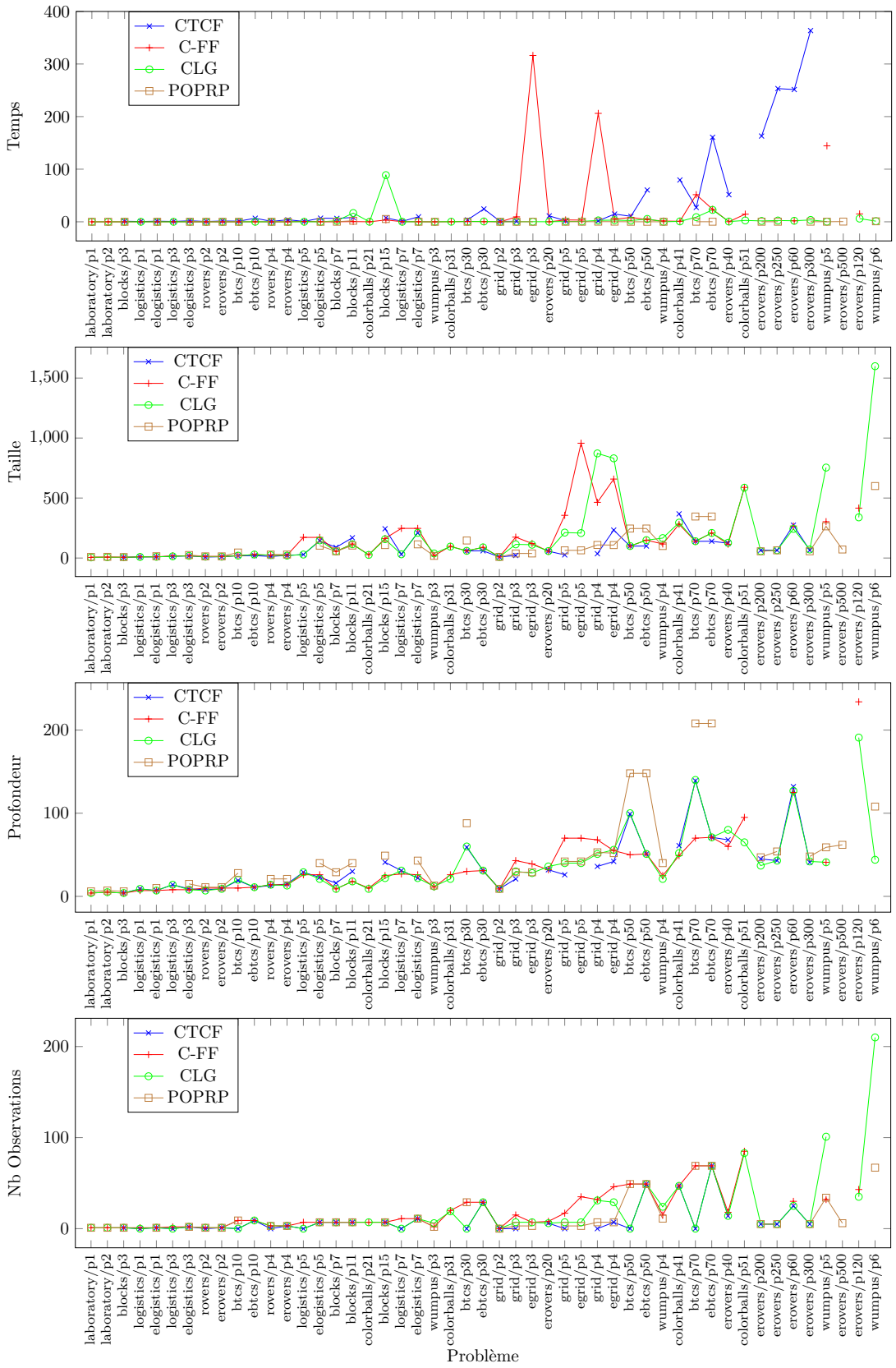

FIGURE 5.15 – Comparaison du temps de calcul, de la taille, de la profondeur, et du nombre d'observations du plan pour les problèmes de la littérature

erovers/p200, p250, p300 et *p500* pour lesquels la capacité mémoire est dépassée, tandis que CLG dépasse le temps limite sur le problème *erovers/p500*.

On peut s'intéresser plus particulièrement au premier graphique concernant le temps de calcul du plan. Nous pouvons déterminer que pour ce qui est des problèmes que CTCF parvient à résoudre, le temps de calcul des plans est généralement supérieur à ceux de Contingent-FF, CLG et PO-PRP, à l'exception des problèmes *grid/p3*, *grid/p4*, *grid/p5* et *btcs/p70* pour lesquels CTCF est meilleur que Contingent-FF, et *blocks/p15* pour lequel CTCF est meilleur que CLG. On peut aussi remarquer que le temps de calcul de CTCF semble grandir d'autant plus que la taille du problème augmente. Cette caractéristique semble rapidement devenir un problème si l'on veut pouvoir traiter des problèmes de grande taille. Si on s'intéresse au deuxième graphique de la figure 5.15 concernant la taille des plans générés, on peut se rendre compte que CTCF calcule des plans de plus petite taille que tous les autres planificateurs pour 10 problèmes sur les 36 pour lesquels celui-ci trouve une solution, à savoir *grid/p3*, *p4* et *p5*, *rovers/p2* et *p4*, l'ensemble des problèmes *ebtcs*, et *egrid/p4*. CTCF trouve des plans de moins bonne taille que tous les autres pour 6 des 36 problèmes qu'il peut résoudre, à savoir *blocks/p7*, *p11* et *p15*, *erovers/p60*, *p200* et *colorballs/p41*. Pour tous les autres problèmes, CTCF trouve des plans de taille équivalente à au moins un des autres planificateurs.

Le troisième graphique de la figure 5.15 détaille la profondeur les plans calculés. CTCF calcule des plans plus courts que tous les autres planificateurs pour 6 problèmes, à savoir *grid/p3*, *p4* et *p5*, *rovers/p4*, *egrid/p4* et *erovers/p300*. CTCF calcule des plans plus longs que les autres planificateurs sur seulement deux problèmes : *erovers/p60* et *colorballs/p41*. Pour tous les autres problèmes, CTCF calcule des plans aussi courts que les autres planificateurs ou bien des plans dont la profondeur se situe entre les valeurs de profondeur des plans des autres planificateurs.

Le quatrième graphique de la figure 5.15 décrit le nombre d'observations incorporées dans les plans calculés. En observant ce graphique, nous pouvons nous rendre compte que lorsqu'une solution conformante existe, CTCF retourne alors bien un plan conformant et n'incorpore pas d'observation dans le plan. C'est la cas pour 14 problèmes, à savoir l'ensemble des problèmes *btcs*, *rovers*, *grid* et *logistics*. Parmi ces problèmes conformants, Contingent-FF, CLG et PO-PRP ajoutent des observations aux problèmes *grid/p3* , *p4* et *p5*, *rovers/p2* et *p4* alors qu'elles ne sont pas nécessaires au calcul d'un plan solution. CLG n'incorpore tout de même pas d'observation pour les problèmes *btcs* et *logistics* contrairement à Contingent-FF et PO-PRP. En plus de ces problèmes conformants, CTCF incorpore moins d'observation que Contingent-FF sur les problèmes *erovers p20*, *p40* et *p60* et moins que Contingent-FF et CLG sur le problème *egrid/p4*. Au vu des résultats de ce graphique, on peut observer que CTCF ajoute toujours moins ou autant d'observations que Contingent-FF, CLG et PO-PRP, indiquant une limitation efficace du nombre d'observations dans le plan.

Problèmes de largeur contingente supérieure à 1

Maintenant que nous avons étudié les résultats sur les problèmes de la littérature, nous pouvons nous intéresser aux problèmes de largeur contingente supérieure à 1 dont les résultats sont contenus dans le tableau 5.1.

Les problèmes *wumpus** et *colorballs** sont des versions modifiées des problèmes wumpus et colorballs de la littérature afin d'augmenter leur largeur contingente. Ces modifications sont expliquées en détail dans l'annexe de ce manuscrit. Dans le tableau 5.1, nous pouvons nous rendre compte que CTCF ne parvient à résoudre que 5 des 18 problèmes de cette catégorie. CLG ne peut résoudre que 6 problèmes, ce qui peut s'expliquer par le fait que celui-ci repose sur une traduction qui n'est pas complète pour les problèmes de largeur supérieure à 1. Contingent-FF dépasse le temps limite sur les problèmes *doors/p9*, *p11* et *wumpus*/p6*, tandis que PO-PRP effectue un dépassement mémoire pour *doors/p11* et calcule des plans non valides pour les problèmes *colorballs**. Intéressons-nous tout de même aux problèmes pour lesquels CTCF trouve une solution. Si on regarde le temps de calcul, on peut se rendre compte que sur les 5 problèmes que CTCF parvient à résoudre, celui-ci met plus de temps à calculer un plan que Contingent-FF, CLG et PO-PRP. En revanche, pour ce qui est de la taille du plan calculé, CTCF obtient des plans de plus petite taille que Contingent-FF, CLG et PO-PRP sur 3 problèmes, à savoir *rovers/p6* , *p8* et *doors/p3*. Pour les 2 autres problèmes, CTCF obtient des plans dont la taille est comprise entre les tailles des plans calculés par Contingent-FF, CLG et PO-PRP. En ce qui concerne la profondeur du plan calculé, CTCF obtient des plans plus courts que Contingent-FF, CLG et PO-PRP pour 4 problèmes, à savoir *rovers/p6*, *p8*, *erovers/p8*, et *doors/p3*. Pour le problème *doors/p4*, CTCF trouve un plan dont la profondeur est situé entre les valeurs de profondeur des plans

TABLE 5.1 – Résultats sur les problèmes contingents de largeur > 1 de CTCF, Contingent-FF, CLG et PO-PRP. t représente le temps de calcul en secondes, s représente la taille du plan en nombre d'actions et d'observations, d représente la profondeur du plan en nombre d'actions et observations et o représente le nombre d'observations au total. NO signifie qu'aucune solution n'est trouvée, TO indique un dépassement du temps imposé, NV signifie que le plan calculé n'est pas valide et DQ signifie que le planificateur est disqualifié à cause d'un plan non valide dans ce benchmark.

Problèmes	CTCF				CT-FF				CLG				PO-PRP			
	t	s	d	o	t	s	d	o	t	s	d	o	t	s	d	o
rovers/p6	1.02	23	23	0	0.12	448	66	11	NO				0.04	110	39	11
rovers/p8	0.80	23	23	0	0.04	170	83	3	NO				0.04	39	28	3
erovers/p6	NO				0.12	346	48	11	NO				0.04	110	39	11
erovers/p8	3.35	62	21	3	0.03	95	36	3	NO				0.02	39	28	3
doors/p3	2.60	13	8	2	0.02	14	9	3	0.00	13	8	2	0.00	19	12	2
doors/p4	4.39	26	14	3	0.04	26	14	5	0.02	23	12	3	0.00	29	18	3
doors/p5	NO				0.37	182	37	39	0.14	144	24	24	0.04	152	44	24
doors/p6	TO				10.86	310	46	59	0.52	252	29	35	0.06	218	55	35
doors/p9	TO				TO				322.26	46024	95	6560	5.02	37016	168	6560
doors/p11	TO				TO				TO				MO			
wumpus*/p3	NO				0.04	19	11	2	NO				0.02	18	13	2
wumpus*/p4	NO				0.49	91	25	11	NO				0.04	93	30	11
wumpus*/p5	NO				71.7	411	39	40	NO				0.22	298	62	40
wumpus*/p6	NO				TO				0.78	921	42	122	0.62	526	89	70
colorballs*/p21	NO				0.07	45	15	13	NO				DQ			
colorballs*/p31	NO				0.26	133	26	32	NO				DQ			
colorballs*/p41	NO				155.86	405	56	79	NO				DQ			
colorballs*/p51	NO				208.75	1091	87	179	NO				NV			

retournés par Contingent-FF, CLG et PO-PRP. Enfin, le nombre d'observations incorporées au plan par CTCF est plus petit que Contingent-FF, CLG et PO-PRP sur les problèmes *rovers/p6* et *p8*. Ce nombre est équivalent à Contingent-FF, CLG et PO-PRP sur le problème *erovers/p8*, et plus petit que Contingent-FF seulement sur les problèmes *doors/p3* et *p4*.

Problèmes coûteux

La dernière classe de problèmes à étudier est la classe des problèmes coûteux. Pour réaliser les problèmes coûteux, nous avons modifié des problèmes de la littérature afin que les observations soient plus coûteuses en terme de nombre d'action en rendant obligatoire la réalisation d'une action supplémentaire pour l'exécution d'une observation. Les modifications permettant de constituer cette classe de problème sont décrites en annexe de ce manuscrit. Les résultats de CTCF, Contingent-FF, CLG et PO-PRP sur cette classe de problèmes sont réunis dans le tableau 5.2.

Dans un premier temps on peut remarquer que CTCF ne parvient à résoudre que les 8 problèmes possédant une solution conformante, à savoir *btcs+* et *erovers+*. Cela s'explique par le fait que les actions nécessaires à la réalisation d'une observation ne se trouvent pas dans le plan calculé par CPCES, ce qui rend impossible l'application d'une observation dans les états de croyance calculés par la fonction FINDOBSERVATION et dérivant de l'application du plan calculé par CPCES. CTCF ne trouve donc pas d'état de croyance dans lequel appliquer les observations candidates, et ne peut donc pas résoudre de problèmes contingents *coûteux* modélisés de cette manière, contrairement à Contingent-FF, CLG, et PO-PRP qui y parviennent. On peut cependant noter que CLG ne peut pas résoudre les problèmes *rovers+/p6*, *p8*, et *erovers+/p6* et *p8* étant donné que ces problèmes possèdent une largeur contingente supérieure à 1.

Intéressons-nous tout de même aux problèmes que CTCF parvient à résoudre. On peut remarquer dans le tableau 5.2 que CTCF a un temps de calcul supérieur à Contingent-FF, CLG et PO-PRP sur 6 des 8 problèmes. Pour les problèmes *btcs+/p50* et *p70*, CTCF est plus rapide que Contingent-FF mais reste bien plus lent que CLG et PO-PRP pour trouver une solution. En ce qui concerne la taille des plans en revanche, CTCF calcule des plans plus petits que Contingent-FF, CLG, et PO-PRP pour l'ensemble des 8 problèmes qu'il peut traiter (à égalité avec CLG pour les problèmes *btcs+*. CTCF calcule également des plans de plus petite profondeur que ceux calculés par Contingent-FF, CLG, et

TABLE 5.2 – Résultats sur les problèmes contingents *coûteux* de CTCF, Contingent-FF, CLG et PO-PRP. t représente le temps de calcul en secondes, s représente la taille du plan en nombre d'actions et d'observations, d représente la profondeur du plan en nombre d'actions et observations et o représente le nombre d'observations au total. NO signifie qu'aucune solution n'est trouvée.

Problèmes	CTCF				C-FF				CLG				PO-PRP			
	t	s	d	o	t	s	d	o	t	s	d	o	t	s	d	o
btcs+/p10	1.25	20	20	0	0.03	37	28	9	0.00	20	20	0	0.02	56	38	9
btcs+/p30	5.62	60	60	0	2.84	117	88	29	0.34	60	60	0	0.04	176	118	29
btcs+/p50	18.76	100	100	0	39.40	197	148	49	2.28	100	100	0	0.1	296	198	49
btcs+/p70	51.0	140	140	0	237.62	277	208	69	8.56	140	140	0	0.18	416	278	69
rovers+/p2	0.58	8	8	0	0.00	13	13	0	0.00	9	9	0	0.02	16	13	1
rovers+/p4	0.81	13	13	0	0.00	19	19	0	0.00	15	15	0	0.02	35	27	3
rovers+/p6	1.00	25	25	0	61.77	117	117	0	NO				0.2	45	39	2
rovers+/p8	0.79	23	23	0	0.10	89	89	0	NO				0.08	32	28	1
erovers+/p2	NO				0.02	32	30	1	0.00	15	13	1	0.02	16	13	1
erovers+/p4	NO				0.10	45	38	3	0.00	27	20	3	0.02	35	27	3
erovers+/p6	NO				4.89	310	65	11	NO				0.06	132	47	11
erovers+/p8	NO				3.23	87	51	3	NO				0.06	45	32	3
elogistics+/p1	NO				0.02	11	8	1	0.00	11	8	1	0.0	16	11	1
elogistics+/p3	NO				0.04	41	26	2	0.00	21	12	2	0.0	27	17	2
elogistics+/p5	NO				0.15	200	32	7	0.04	152	24	7	0.0	107	42	7
elogistics+/p7	NO				0.30	279	42	11	0.08	219	25	11	0.18	139	55	11
doors+/p3	NO				0.02	21	15	2	0.02	15	10	2	0.0	21	14	2
doors+/p4	NO				0.04	37	24	3	0.02	26	15	3	0.0	31	21	3
doors+/p5	NO				0.31	341	78	24	0.22	168	32	24	0.04	182	56	25
doors+/p6	NO				1.54	555	100	35	0.82	287	39	35	0.06	294	73	41

PO-PRP pour les 8 problèmes conformants (à égalité avec CLG pour les problèmes *btcs+*. Enfin, CTCF incorpore moins d'observations dans le plan que Contingent-FF et PO-PRP pour les problèmes *btcs+* et CTCF incorpore moins d'observations que PO-PRP pour les problèmes *rovers+*.

5.4.3 Conclusions expérimentales

Les résultats expérimentaux de la comparaison entre CTCF, Contingent-FF, CLG, et PO-PRP nous permettent de tirer plusieurs conclusions sur notre approche. Premièrement, nous pouvons identifier que CTCF manque de complétude, celui-ci ne parvenant pas à trouver une solution pour tous les problèmes. En ce qui concerne les problèmes *wumpus* et *wumpus**, l'observation de la présence d'un monstre ou d'un précipice dans une case se fait via une observation indirecte permettant d'observer l'odeur du monstre ou la brise du précipice dans une case adjacente. CTCF ne peut donc pas trouver d'observation directement à partir du plan que retourne CPCES, les observations du problème *wumpus* n'observant pas directement de propositions contenues dans les préconditions des actions. Pour les problèmes *laboratory*, CTCF ne peut pas trouver d'état de croyance dans lequel l'observation peut s'appliquer étant donné que l'observation doit être réalisée dans une case dans laquelle le robot ne passe pas dans le plan calculé par CPCES. Pour ce qui est des problèmes *coûteux* contingents, les actions nécessaires à la réalisation d'une observation ne se trouvent pas dans le plan calculé par CPCES, ce qui rend impossible l'application d'une observation dans les états de croyance calculés par la fonction FINDOBSERVATION et empêchant CTCF de trouver une solution. Pour tous les autres problèmes que CTCF ne peut pas résoudre, il se peut que l'observation nécessite une action qui ne peut pas être atteinte dans l'intégralité des états de croyance calculés, ce qui peut être le cas dans les problèmes *colorballs* et *colorballs** dans lesquels l'observation nécessaire peut concerner la couleur de la balle avant même que la position de la balle ne soit certaine. En ce qui concerne *doors* et *egrid*, CPCES peut retourner un plan dans lequel le contre-exemple demande d'observer l'ouverture d'une porte, et pour lequel le plan passe par plusieurs portes dont l'ouverture est incertaine. Cette incertitude ne permet pas l'application du plan à l'ensemble des états de croyance, et donc ne permet pas l'accès à l'observation.

Pour la plupart des problèmes que CTCF peut résoudre, CTCF met plus de temps à calculer une solution que Contingent-FF, CLG et PO-PRP. Même si ce temps de calcul reste acceptable étant donné que le plan est calculé hors ligne, ce temps de calcul est tout de même un inconvénient de l'approche.

Malgré ces inconvénients, CTCF semble limiter efficacement les observations incorporées au plan, rendant CTCF compétitif avec Contingent-FF, CLG et PO-PRP sur la taille et la profondeur des plans calculés. L'efficacité de la limitation des observations sur ces facteurs de taille et de profondeur semble prendre de l'ampleur lorsque l'on considère des problèmes coûteux dans lesquels le coût d'une observation est modélisée comme un coût en taille et en profondeur, forçant l'application d'une action avant de réaliser une observation. Cependant, CTCF ne résoud pas suffisamment de problèmes coûteux pour qu'une réelle conclusion puisse être réalisée sur cette classe de problème.

5.5 Discussion

En analysant nos résultats, on peut se rendre compte que l'un des inconvénients de CTCF est son temps de calcul des plans, qui est plus élevé que Contingent-FF, CLG et PO-PRP. De plus, CTCF ne fonctionne que lorsque l'observation candidate peut être appliquée sur les états de croyance formés par l'exécution de la partie applicable du plan retourné par CPCES. Dans le cas contraire, les états de croyance calculés ne peuvent atteindre les conditions d'exécution de l'observation et il est impossible pour notre approche de trouver un plan solution. Le fait de chercher un état de croyance bien précis dans lequel appliquer l'observation à partir du plan retourné par CPCES permet de représenter le plan sous la forme d'un graphe compact. Cependant, représenter le but du problème comme un état de croyance augmente grandement le nombre de propositions composant les contraintes de but, rendant plus complexe le calcul d'une solution satisfaisant ces contraintes. Lorsque l'on s'intéresse aux algorithmes de notre approche, on peut se demander si le fait de représenter et manipuler les états de croyance en interne ne rend pas l'approche inutilisable lorsque la taille du problème augmente énormément, les états de croyance prenant une place mémoire importante et leur génération à partir du problème PDDL prenant un temps beaucoup plus considérable. On peut supposer que traiter des centaines ou milliers d'états initiaux possibles de cette façon n'est pas envisageable dans un temps et une capacité de traitement raisonnable. Enfin, CTCF souffre d'un inconvénient majeur, son absence de complétude.

Une des pistes d'amélioration du temps de calcul est de pouvoir réutiliser au moins une partie du plan échouant retourné par CPCES comme plan menant à l'observation. Pour cela, on doit tout d'abord s'assurer que au moins une partie du plan retourné par CPCES est conformant et vérifier si l'observation peut être réalisée à la suite de ce bout de plan conformant. Le temps de calcul de CTCF sur les problèmes ayant une solution conformante peut principalement se résumer en une somme de deux temps distincts, le temps que met la procédure à générer l'état de croyance initial à partir du problème PDDL, et le temps que met CPCES à résoudre le problème étant donné que dans ce genre de problème, CPCES n'est appelé qu'une seule fois. Le temps que met CPCES à calculer un plan est malheureusement incompressible. En revanche, une piste d'amélioration de l'approche est la réduction du temps que met la procédure à générer et manipuler l'état de croyance en le représentant de manière plus compacte. De plus, une représentation plus compacte des états de croyance permettrait un traitement plus efficace des problèmes de plus grande taille. Une autre piste d'amélioration de l'approche peut être de réduire les contraintes de but du plan menant à l'observation afin de rendre le calcul du plan menant à l'observation plus simple, ce qui permettrait théoriquement de traiter un plus grand nombre de problèmes. Plutôt que d'être constitué d'un état de croyance complet, le but du problème menant à l'observation pourrait simplement être constitué des préconditions de l'observation en question. De plus, ne plus se baser sur les états de croyance générés par l'application du plan π échouant retourné par CPCES à l'état de croyance initial permettrait de résoudre des problèmes pour lesquels l'observation ne peut pas être réalisée sur le chemin d'exécution de π. Afin de rendre CTCF complet, une solution pourrait être de réaliser un retour sur le choix des observations à réaliser, permettant de sélectionner une autre observation si l'observation choisie ne peut pas être atteinte ou bien que sa réalisation ne mène finalement pas à un plan solution.

5.6 Conclusion

Dans ce chapitre, nous avons développé un algorithme de planification contingente limitant le nombre d'observations du plan par l'utilisation d'un planificateur conformant pour le calcul des branches. Afin d'implémenter cet algorithme, nous avons choisi de faire confiance à l'efficacité du planificateur conformant CPCES (Grastien et Scala, 2017).

Le principe de notre algorithme contingent consiste tout d'abord à appeler CPCES à partir du problème initial afin de calculer un plan conformant et de le retourner si un tel plan existe. Dans le cas contraire, le plan et l'état initial de contre-exemple retournés par CPCES sont utilisés afin de déterminer quelle observation est nécessaire et dans quel état de croyance la réaliser afin de discriminer le contre-exemple. Le processus contingent est ensuite redémarré afin de trouver un plan contingent ou conformant menant à l'état de croyance dans lequel on doit réaliser l'observation. Deux sous-problèmes prenant en compte les résultats potentiels de l'observation sont générés et utilisés en entrée du processus contingent pour calculer deux plans partant de ces deux sous-problèmes et menant au but, permettant d'atteindre le but du problème quel que soit le résultat de l'observation. Ces trois plans ainsi que l'action d'observation sont ensuite assemblés et retournés pour former le plan contingent final solution du problème.

Les résultats des expérimentations menées afin d'évaluer notre approche sont plutôt encourageants et compétitifs en comparaison d'autres planificateurs de la littérature mais révèlent tout de même que des améliorations sont possibles notamment en terme de rapidité de calcul et de complétude. Certaines de ces améliorations sont étudiées et implémentées dans les chapitres suivants.

Dans le prochain chapitre (Chapitre 6), nous allons nous intéresser à une de ces améliorations possible qui concerne l'amélioration de la rapidité du temps de calcul par une récupération plus intelligente du contre-exemple par CPCES permettant la réutilisation de la partie conformante du plan retourné par celui-ci comme plan menant à l'observation souhaitée.

Dans les chapitres suivants, nous étudierons les autres améliorations envisagées, en commençant par la réduction des contraintes de but pour le calcul du plan menant à l'observation (Chapitre 7). Nous poursuivrons par l'étude d'une représentation différente des états de croyance consistant à associer l'état de croyance initial et la séquence d'actions/observations menant à l'état de croyance courant (Chapitre 8). Finalement, nous terminerons cette thèse en rendant l'approche complète grace à un retour possible sur le choix des observations (Chapitre 9).

Chapitre 6

Méthode de réduction du temps de calcul par récupération intelligente de contre-exemple

6.1 Motivation

Les résultats de notre approche contingente présentée dans le chapitre précédent nous indiquent que l'une des améliorations possibles de cette approche est la diminution du temps de calcul d'une solution, qui est plus élevé (mais d'un ordre de grandeur tout de même raisonnable) que les planificateurs contingents de la littérature avec lesquels nous avons comparé l'approche. L'idée que nous avons développée afin de réduire ce temps de calcul consiste à ne plus rappeler CPCES pour calculer le plan menant à l'observation. À la place, on utilise directement le début de plan échouant retourné par CPCES. Ce processus diminue le nombre de branches calculées, influençant grandement le temps de calcul final du plan solution du problème.

6.2 Mise en oeuvre

Nous avons décidé de réduire ce nombre de branches calculées en proposant une modification du comportement de CPCES afin que celui-ci ne récupère plus le contre-exemple de manière aléatoire, mais au contraire que celui-ci soit sélectionné pour qu'il fasse échouer le plan le plus tôt possible.

Cette modification est théoriquement intéressante pour notre approche, puisque cela signifierait que le préfixe du plan (le début du plan valide sur le contre-exemple) serait conformant, et donc réutilisable comme plan menant à l'observation, si celle-ci est applicable à la suite de ce plan. Cette réutilisation du préfixe conformant du plan retourné par CPCES nous permet d'éviter le calcul du plan menant à l'observation, ce qui doit théoriquement diminuer le temps de calcul de la solution sur un bon nombre des problèmes. Dans la suite de ce manuscrit, nous appelerons cette version de l'approche ConTingent planner using a ConFormant planner and Early counter-examples (CTCFE). Dans un premier temps, nous allons décrire le nouveau processus de sélection de contre-exemples dans la section 6.2.1. Nous verrons ensuite dans la section 6.2.2 comment la procédure de sélection d'observation à été modifiée pour permettre la réutilisation du préfixe conformant du plan retourné par CPCES. La dernière étape d'implémentation de l'approche a été de modifier la procédure contingente afin d'utiliser directement le plan retourné par la fonction de sélection d'observations (section 6.2.3). Dans la section 6.2.4, nous détaillerons rapidement les modifications que CTCFE apporte à l'architecture globale de notre approche, puis nous analyserons l'approche de manière théorique dans la section 6.3. Finalement, nous évaluerons les résultats de l'implémentation de CTCFE dans la section 6.4 et enfin nous conclurons dans les sections 6.5 et 6.6.

Le travail de modification de CPCES et d'intégration au processus contingent a été partagé entre Alban Grastien et moi. Alban Grastien a pris en charge les modifications de CPCES tandis que je me suis occupé de la partie intégration de cette nouvelle version au processus contingent. Cette nouvelle version de l'approche fait partie des différents travaux menés en collaboration avec Alban Grastien lors de ma mobilité de trois mois à l'Australian National University de Canberra.

6.2.1 Sélection alternative de contre-exemple

Si on note Φ l'ensemble des états contre-exemples au plan $\pi =< a_0, ..., a_k >$, alors pour chaque état $s \in \Phi$, on peut associer l'action $a_i \in \pi$ échouant dans l'état s. La nouvelle version de CPCES que l'on nomme $CPCES_E$ dans ce manuscrit sélectionne l'état de contre-exemple $s \in \Phi$ pour lequel i est minimal, contrairement à la version classique de CPCES qui sélectionne un état aléatoirement dans Φ. Cette nouvelle façon de sélectionner le contre-exemple assure que le plan $\pi' =< a_0, ..., a_{i-1} >$ contenu dans le plan π est conformant étant donné qu'aucun contre-exemple n'existe avant l'action a_i.

Cette nouvelle sélection de contre-exemple nécessite la modification de CPCES. Cette modification est réalisée en transformant le processus de vérification du plan par le SAT-Solver en ajoutant une contrainte de minimisation afin de récupérer le contre-exemple de l'ensemble de contre-exemples existants minimisant l'étape du plan à partir de laquelle le plan courant échoue. Cette modification de l'algorithme 3 est illustrée par l'algorithme 5. Afin d'apporter de la visibilité aux modifications apportées, les éléments de l'algorithme 3 supprimés sont indiqués en rouge et les éléments apportés par l'algorithme 5 sont indiqués en cyan. Cette manière de visualiser les changements dans les algorithmes sera utilisée dans l'ensemble du manuscrit.

Algorithm 5 CPCES with earlier counter-example ($CPCES_E$)

Require: $\mathcal{P} = (\Sigma, \mathcal{B}_0, g)$
Ensure: π, γ
1: $\mathcal{B} := \emptyset$
2: $\pi := \varepsilon$
3: **loop**
4: **if** π is a solution for $\mathcal{P}$ **then**
5: **return** $\pi, \emptyset$
6: ~~let γ be a counter-example~~
7: let γ be the counter-example in which π fails the earlier
8: $\mathcal{B} := \mathcal{B} \cup \{\gamma\}$
9: compute a new plan π' for $\mathcal{P}' = (\Sigma, \mathcal{B}, g)$
10: **if** no such π' exists **then**
11: **return** π, γ
12: $\pi := \pi'$

Exemple: Récupération du contre-exemple faisant échouer le plan le plus tôt

Afin d'illustrer la nouvelle sélection du contre-exemple par $CPCES_E$ (algorithme 5), nous allons nous intéresser à un nouvel ensemble d'états initiaux possibles du problème de robot d'exploration.

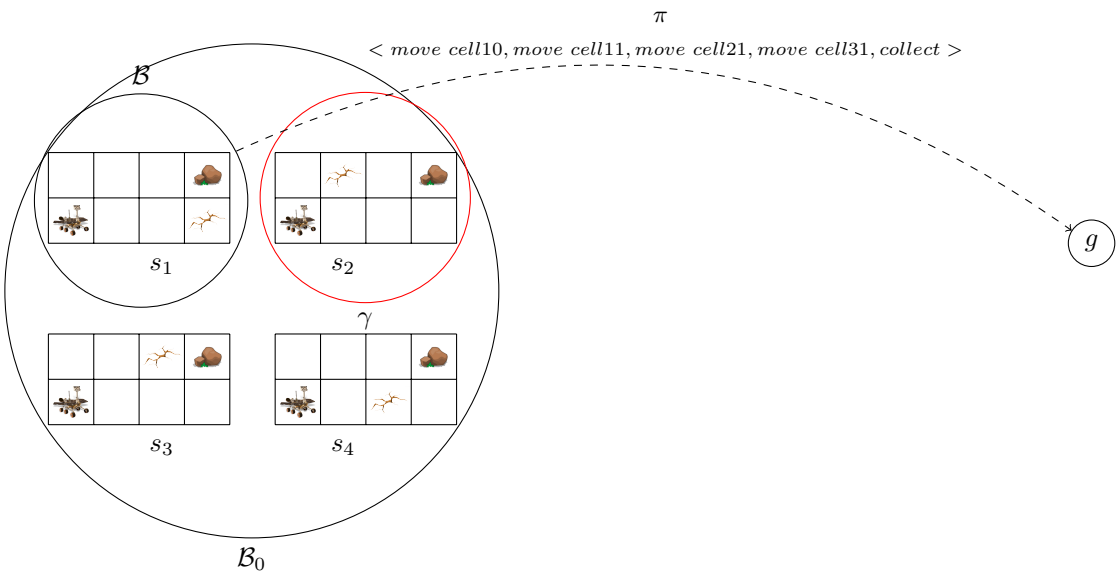

FIGURE 6.1 – Sélection d'un contre-exemple au premier plan calculé par $CPCES_E$

La figure 6.1 illustre le choix du contre-exemple sélectionné pour le premier plan calculé par $CPCES_E$. $CPCES_E$ sélectionne tout d'abord l'état s_1 comme contre-exemple au plan vide, puis l'intègre à l'ensemble de contre-exemples courant $\mathcal{B}$. $CPCES_E$ calcule ensuite le plan π menant de $\mathcal{B}$ à g. On peut remarquer que s_2 et s_3 sont deux états contre-exemples possible au plan π. Dans la version classique de CPCES, l'un de ces deux contre-exemples est sélectionné de manière aléatoire. En revanche, dans cette version, $CPCES_E$ sélectionne le contre-exemple qui fait échouer le plan le plus tôt, c'est à dire s_2, car s_2 fait échouer π à la deuxième action, tandis que s_3 le fait échouer à la troisième action.

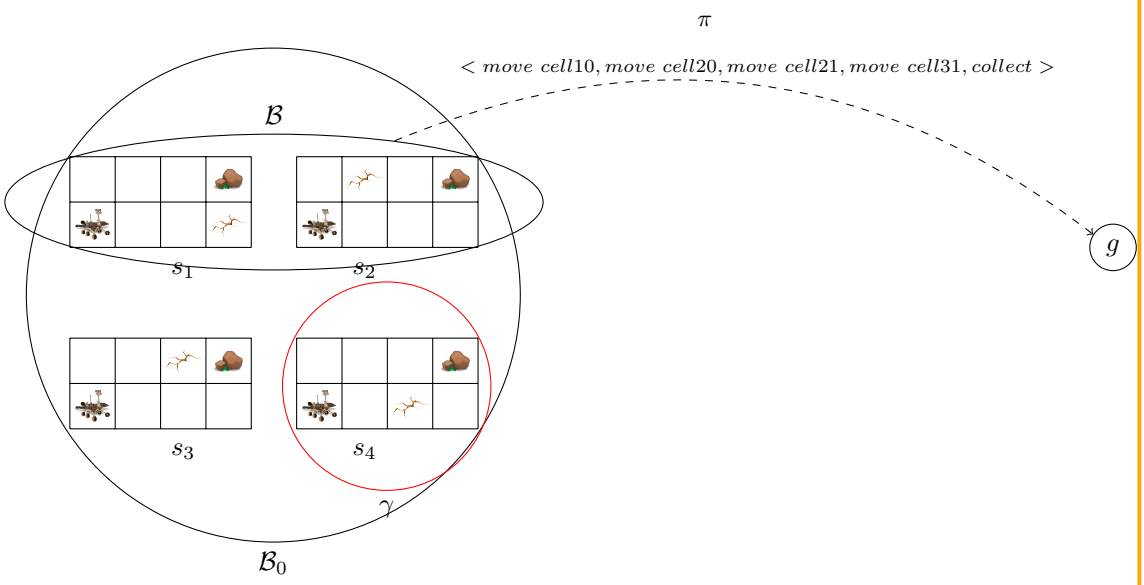

FIGURE 6.2 – Sélection du contre-exemple au deuxième plan calculé par $CPCES_E$

Une fois que $CPCES_E$ a intégré s_2 à l'ensemble des contre-exemples courant $\mathcal{B}$, $CPCES_E$ calcule un nouveau plan π menant de $\mathcal{B}$ à g. Dans la figure 6.2, on peut remarquer que s_3 et s_4 sont des contre-exemples au plan π. Encore une fois, $CPCES_E$ sélectionne le contre-exemple faisant échouer π le plus tôt, s_4, faisant échouer la deuxième action de π. On peut remarquer que l'ajout de s_4 à $\mathcal{B}$ rend impossible le calcul d'un plan conformant menant de $\mathcal{B}$ à g. $CPCES_E$ retourne alors le plan $\pi =< move\ cell10, move\ cell20, move\ cell21, move\ cell31, collect >$ ainsi que le contre-exemple s_4. Étant donné que $CPCES_E$ a sélectionné systématiquement le contre-exemple faisant échouer le plan le plus tôt, on peut remarquer que le préfixe du plan π valide pour $\mathcal{B}$ est conformant, le plan $< move\ cellcell10 >$ est donc un préfixe conformant de π.

6.2.2 Procédure d'extraction du préfixe conformant et de l'observation

Afin de réutiliser le début conformant du plan calculé par $CPCES_E$, la procédure de sélection de l'observation FINDOBSERVATION (algorithme 4) est modifié afin qu'il puisse retourner ce début de plan en plus de l'observation et de l'état de croyance dans lequel est réalisé l'observation. L'algorithme 6 inclut cette modification.

Pour cela, les lignes 3 et 8 sont ajoutées afin de récupérer seulement la partie conformante du plan π retourné par $CPCES_E$ en créant un plan π_σ vide (ligne 3) auquel on ajoute les actions applicables dans l'ensemble d'état de croyance courant (ligne 8). Afin de retourner uniquement la partie du plan à la suite de laquelle l'observation peut être ajoutée, on ajoute la ligne 22 qui permet de supprimer les dernières actions du plan π_σ si l'observation n'est pas applicable dans l'état de croyance courant correspondant à l'état de croyance après application de chaque action de π_σ.

6.2.3 Utilisation du préfixe conformant du plan retourné par $CPCES_E$ par la procédure contingente

Afin de pouvoir réutiliser la partie conformante du plan retourné par $CPCES_E$ et extraite par la fonction EARLYFINDOBSERVATION (algorithme 6), il est nécessaire de modifier la procédure contingente présentée dans CTCF (algorithme 2). Ce nouveau processus contingent est illustré par l'algorithme 7.

Dans ce nouveau processus contingent, la procédure de sélection d'observation FINDOBSERVATION et l'appel à la procédure contingente pour le calcul du plan menant à l'observation sont supprimés (lignes 5 et 6). À la place, la procédure EARLYFINDOBSERVATION est appelée afin de retourner directement l'observation à réaliser, le plan π_σ menant à l'observation, et l'état de croyance $\mathcal{B}_\sigma$ correspondant à l'exécution de π_σ sur $\mathcal{B}_0$ jusqu'à σ (ligne 7). π_σ est ensuite utilisé tel quel comme plan menant

Algorithm 6 EarlyfindObservation

Require: $\mathcal{P}_\Omega, \pi, \gamma$
Ensure: $(\mathcal{B}_\sigma, \sigma, \pi_\sigma)$
1: $beliefList := [\mathcal{B}_0]$
2: $\mathcal{B} := \mathcal{B}_0$
3: $\pi_\sigma := \varepsilon$
4: **for** a in π **do**
5: **if** a applicable in γ **then**
6: $\gamma := T(\gamma, a)$
7: **if** a applicable in $\mathcal{B}$ **then**
8: add a to π_σ
9: **for** $s \in \mathcal{B}$ **do**
10: $\mathcal{B}' := \{T(s, a) | s \in \mathcal{B}\}$
11: $beliefList := beliefList + \mathcal{B}$
12: **else**
13: let $unsatPre$ be the unsatisfied preconditions of a
14: **break**
15: **for** p in $unsatPre$ **do**
16: let $\Omega_p \in \Omega$ be the set of observations for p
17: **for** $\sigma \in \Omega_p$ **do**
18: **for** $\mathcal{B}_\sigma$ in $beliefList$ **do**
19: **if** σ applicable in $\mathcal{B}_\sigma$ **then**
20: ~~**return** $(\mathcal{B}_\sigma, \sigma)$~~
21: **return** $(\mathcal{B}_\sigma, \sigma, \pi_\sigma)$
22: remove last action of π_σ
23: ~~**return** $(\mathcal{B}, None)$~~
24: **return** $(\mathcal{B}, None, None)$

à l'observation (ligne 12). La suite de la procédure contingente est la même que pour CTCF, σ est appliquée à $\mathcal{B}_\sigma$ pour former $\mathcal{B}^+$ et $\mathcal{B}^-$, puis la procédure contingente est appelée pour calculer les branches menant de $\mathcal{B}^+$ et $\mathcal{B}^-$ au but du problème.

Algorithm 7 ContingentPlanning Procedure

Require: $\mathcal{P}_\Omega = (\mathcal{P}_\Omega, \mathcal{B}_0, g)$
Ensure: π_c
1: $\mathcal{P} = (S, A, T, \mathcal{B}_0, g)$
2: $\pi, \gamma := CPCES_E(\mathcal{P})$
3: **if** $\gamma = \emptyset$ **then**
4: **return** π
5: ~~$\mathcal{B}_\sigma, \sigma :=$ findObservation$(\mathcal{P}_\Omega, \pi, \gamma)$~~
6: ~~$\pi_\sigma :=$ ContingentPlanning$((\Sigma_\Omega, \mathcal{B}_0, \mathcal{B}_\sigma))$~~
7: $\mathcal{B}_\sigma, \sigma, \pi_\sigma :=$ EarlyfindObservation$(\mathcal{P}_\Omega, \pi, \gamma)$
8: $\mathcal{B}^+ := T(\mathcal{B}_\sigma, \sigma)$ *with* $eff(\sigma) = \top$
9: $\pi_p :=$ ContingentPlanning$((\Sigma_\Omega, \mathcal{B}^+, g))$
10: $\mathcal{B}^- := T(\mathcal{B}_\sigma, \sigma)$ *with* $eff(\sigma) = \bot$
11: $\pi_n :=$ ContingentPlanning$((\Sigma_\Omega, \mathcal{B}^-, g))$
12: **return** $(\pi_\sigma$;**if** σ **then** π_p **else** $\pi_n)$

6.2.4 Modification de l'architecture

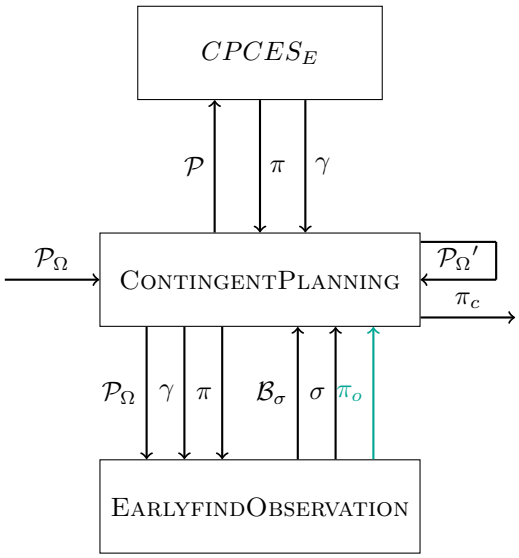

Figure 6.3 – Architecture du processus contingent

L'architecture de notre approche initiale (figure 5.13) ne subit qu'un seul changement dans cette version. On peut distinguer cette modification dans la figure 6.3 représentant l'architecture de CTCFE dans laquelle la procédure findObservation retourne un élément supplémentaire : un plan conformant π_σ menant à l'état de croyance $\mathcal{B}_\sigma$ dans lequel on peut réaliser l'observation σ choisie.

6.3 Analyse théorique

6.3.1 Complétude

Cette nouvelle approche ne résout pas les problèmes de complétude de CTCF. Même si on peut penser que le fait de réutiliser la partie conformante du plan retourné par $CPCES_E$ augmente les chances de directement obtenir un plan menant à l'observation souhaitée, il n'y a toujours aucun mécanisme de retour en arrière possible dans le choix des observations. De plus, CTCFE se basant toujours sur le plan π retourné par $CPCES_E$ pour déterminer quelle observation réaliser, les problèmes pour lesquels l'observation choisie ne peut pas être réalisée sur le chemin d'exécution de π ne peuvent pas être résolus.

6.3.2 Complexité

Dans cette section, nous allons déterminer quelle est la complexité de CTCFE à partir des algorithmes décrits dans ce chapitre. Comme pour le chapitre précédent, nous simplifions les calculs en considérant l'application d'une action ou la vérification et la sélection d'une observation comme une opération unitaire, et nous sur-approximons les équations en ne gardant que les variables du plus grand ordre de grandeur.

Étudions dans un premier temps l'algorithme EarlyfindObservation (algorithme 6). Nous négligeons les opérations unitaires du début de fonction permettant d'initialiser les variables.

Tout comme findObservation, la fonction effectue une boucle dans laquelle dans le pire des cas nous avons $|\pi|$ applications d'une action dans un état de croyance, et donc dans le pire des cas les $|\pi|$ actions sont appliquées à $2^{|F|}$ états. Nous avons donc $|\pi| \times 2^{|F|}$ opérations dans la première boucle.
La deuxième partie de l'algorithme EarlyfindObservation est aussi sensiblement équivalente à findObservation en terme de complexité, le fait de retirer une action à la ligne 22 ne change pas la complexité de la fonction, reposant sur les trois boucles imbriquées. On a donc toujours dans le pire cas une complexité de $|\pi| \times 2^{|F|} \times |F| \times |\Omega|$ pour cette deuxième partie de fonction. La complexité de la

procédure EarlyfindObservation est donc proportionnelle à $|\pi| \times 2^{|F|} + |\pi| \times 2^{|F|} \times |F| \times |\Omega| \propto |\pi| \times |\Omega| \times |F| \times 2^{|F|}$.

La complexité algorithmique en notation de Landau est de :
$\mathcal{O}(|\pi| \times |\Omega| \times |F| \times 2^{|F|}) = \mathcal{O}\left(\max\left(|\pi|, |\Omega|, |F|, 2^{|F|}\right)\right) = \mathcal{O}(2^{|F|})$.

Notons $\mathcal{C}(n)$ la complexité de CPCES prenant n états en entrée, $\mathcal{F}(n)$ la complexité de EarlyfindObservation prenant n états en entrée, et $\mathcal{T}(n)$ la complexité de la procédure contingente prenant en entrée n états. La complexité de la procédure contingente peut être exprimée par l'équation :

$$\mathcal{T}(n) = \mathcal{C}(n) + \mathcal{F}(n) + 2\mathcal{T}(n/2) \tag{6.1}$$

avec $2\mathcal{T}(n/2)$ la complexité des appels de la planification contingente pour le calcul des sous-plans π_p et π_n (lignes 9 et 11).

Comme CTCF, la complexité de CTCFE est dominée par la complexité de CPCES $\mathcal{C}(n)$, $\mathcal{T}(n)$ est proportionnel à $\mathcal{O}(\mathcal{C}(n))$ et donc a une complexité dans le pire cas de $\mathcal{O}(2^{|F|})$. Cependant, nous pouvons tout de même déterminer que l'appel de moins de la procédure contingente pour le calcul du plan menant à l'observation qui enlève l'élément $\mathcal{T}'(n)$ de la complexité de CTCFE est un gain important.

6.4 Expérimentations

Les expérimentations de l'implémentation de CTCFE se sont déroulées dans les mêmes conditions que pour CTCF. Comme pour CTCF, une limite de temps de 10 minutes maximum est imposée lors de la résolution des problèmes. Nous avons décidé de comparer cette nouvelle approche avec CTCF, Contingent-FF, CLG et PO-PRP sur les mêmes benchmarks : des problèmes de la littérature, des problèmes dont la largeur contingente est supérieure à 1, et des problèmes *coûteux*. L'intégralité des benchmarks considérés sont décrits en annexe de ce manuscrit.

6.4.1 Résultats

Dans un premier temps, nous allons nous intéresser aux problèmes conformants et contingents de la littérature et au problème *laboratory* développé dans cette thèse. Dans un deuxième temps, nous nous intéresserons aux problèmes de la littérature et aux problèmes modifiés possédant une largeur contingente supérieure à 1. Enfin, nous détaillerons les résultats sur des problèmes de la littérature modifiés afin que les observations de ces problèmes soient *coûteuses*.

Problèmes contingents de la littérature

L'ensemble détaillé des résultats de cette section se trouve dans les tableaux B.1 et B.2 situés en annexe de ce manuscrit. Dans la figure 6.4, nous pouvons distinguer quatre graphiques représentant respectivement le temps de calcul (en secondes), la taille (en nombre total d'actions et d'observations), la profondeur (taille de la plus longue séquence d'actions et d'observations), et le nombre total d'observations contenues dans le plan entier pour chaque problème. Dans ces graphiques, les problèmes sont triés de manière croissante selon un critère de difficulté que nous définissons comme le produit du nombre de propositions par état et du nombre d'états composant l'état de croyance initial.

Lorsque l'on observe ces courbes, on peut tout d'abord se rendre compte que comme CTCF, CTCFE ne résout pas tous les problèmes contingents de la littérature étudiés. Contrairement à CTCF, CTCFE ne parvient pas à résoudre les problèmes *egrid/p4* et *colorballs/p41*. Pour ce qui est du temps de calcul, nous pouvons observer dans le premier graphique de la figure 6.4 que CTCFE est plus rapide que CTCF sur 24 problèmes. Cet écart de temps est plus important pour les problèmes comportant des branches plus longues entre chaque observation, ce temps correspondant au temps de calcul du plan menant à l'observation évité par CTCFE. Néanmoins, cette amélioration du temps de calcul pour ces problèmes ne suffit pas à rejoindre les temps de calculs de Contingent-FF, CLG et PO-PRP. On peut remarquer que CTCFE met légèrement plus de temps à calculer des plans que CTCF pour 7 problèmes (*btcs/p30*, *p50* et *p70*, *grid/p3* et *p4*, *logistics/p3* et *p7*). Ces problèmes sont des problèmes conformants, la différence de temps de calcul entre les deux approches pour ce type de problème se situe nécessairement dans la modification de CPCES, qui met légèrement plus de temps à calculer un plan conformant que la version classique de CPCES.

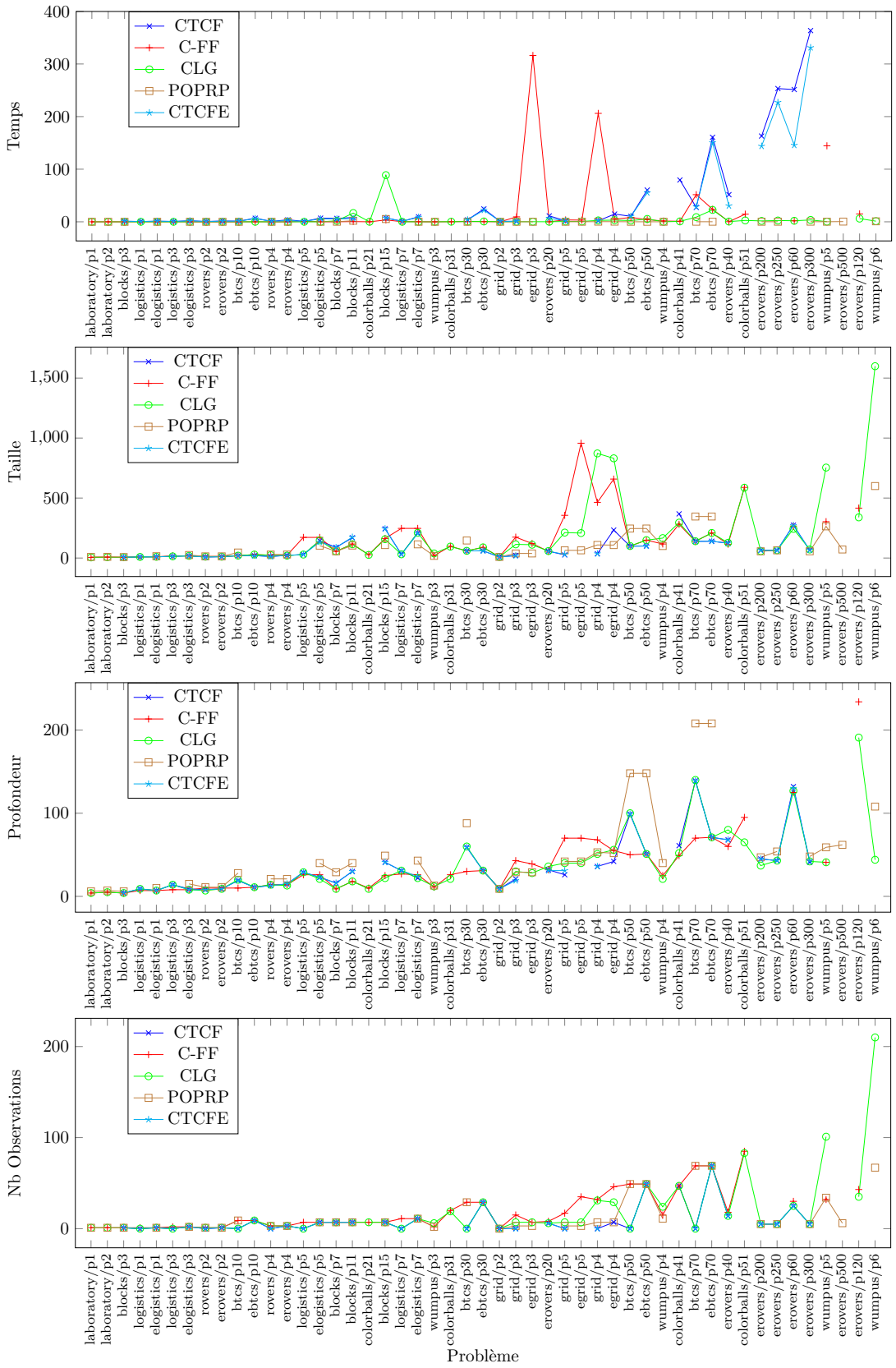

FIGURE 6.4 – Comparaison du temps de calcul, de la taille, de la profondeur, et du nombre d'observations du plan pour les problèmes de la littérature

En ce qui concerne la taille des plans, nous pouvons observer dans le deuxième graphique de la figure 6.4 que CTCFE trouve des plans de taille inférieure à CTCF pour 7 problèmes, c'est à dire les problèmes *grid/p3*, *blocks/p7*, *erovers/p20*, *p60*, *p250*, *elogistics/p5* et *p7*, mais trouve des plans de plus grande taille que CTCF pour 4 problèmes, c'est à dire les problèmes *grid/p5*, *blocks/p11*, *erovers/p40* et *p300*. Pour l'ensemble des autres problèmes, CTCFE trouve des plans de la même taille que CTCF. Les différences de valeurs de taille entre CTCF et CTCFE ne permettent pas de battre Contingent-FF, CLG et PO-PRP sur plus de problèmes.

En ce qui concerne la profondeur des plans, nous pouvons observer dans le troisième graphique de la figure 6.4 que CTCFE trouve des plans de plus petite profondeur que CTCF pour 4 problèmes, c'est à dire *grid/p3*, *erovers/p20* et *p60* et *elogistics/p3*. Ces nouvelles valeurs de profondeur de plan permettent à CTCFE de calculer des plans de plus petite profondeur que Contingent-FF, CLG et PO-PRP pour le problème *erovers/p20*. CTCFE calcule un plan de la même profondeur que Contingent-FF et CLG sur le problème *elogistics/p3*. En revanche, CTCFE trouve des plans plus longs que CTCF pour 3 problèmes, c'est à dire les problèmes *grid/p5*, *erovers/p300* et *elogistics/p7*. Ces valeurs de profondeurs rendent CTCFE moins performant que CLG en terme de profondeur de plan pour les problèmes *erovers/p300* et *elogistics/p7*.

Enfin, nous pouvons nous rendre compte dans le dernier graphique de la figure 6.4 que CTCFE calcule des plans comportant le même nombre d'observations que CTCF pour l'ensemble des problèmes.

Problèmes de largeur contingente supérieure à 1

Nous pouvons maintenant nous intéresser aux problèmes de largeur contingente supérieure à 1. Les résultats de CTCFE sur ce type de problème sont contenus dans le tableau B.4 situé en annexe de ce manuscrit. En comparant les résultats du tableau B.4 et du tableau B.3, nous pouvons nous rendre compte que tout comme CTCF, CTCFE ne résout pas l'ensemble des problèmes contingents de largeur supérieure à 1. Cependant, CTCFE trouve une solution pour les problèmes *erovers/p6*, *doors/p5* et *colorballs*/21* contrairement à CTCF.

Si nous nous concentrons sur le temps de calcul des plans, nous pouvons déterminer que CTCFE calcule des plans plus rapidement que CTCF pour les problèmes *erovers/p8* et *doors/p3* et *p4*. Malheureusement, cette amélioration du temps de calcul sur ces problèmes ne suffit pas à rejoindre les temps de calcul bien plus petits de Contingent-FF, CLG et PO-PRP sur les problèmes de cette classe. De plus, CTCFE calcule des plans moins rapidement que CTCF sur les problèmes *rovers/p6* et *p8*.

CTCFE calcule des plans plus petits que CTCF sur les problèmes *erovers/p8* et *doors/p4*, battant Contingent-FF sur *erovers/p8* et *doors/p4* ainsi que PO-PRP sur *doors/p4*. De plus, CTCFE calcule un plan plus court que Contingent-FF sur *doors/p5* et *erovers/p6*. Pour les autres problèmes, les tailles de plan sont les mêmes que CTCF.

CTCFE calcule un plan plus court que CTCF pour le problème *doors/p4*, battant Contingent-FF et PO-PRP sur ce problème. CTCFE calcule un plan moins profond que Contingent-FF pour le problème *colorballs*/p21* et *doors/p5*. CTCFE calcule également un plan moins profond que PO-PRP pour le problème *doors/p5*. Pour les autres problèmes, les profondeurs des plans calculés par CTCFE sont les mêmes que CTCF.

CTCFE incorpore le même nombre d'observations que CTCF sur l'ensemble des problèmes. CTCFE incorpore moins d'observations que Contingent-FF et PO-PRP pour le problème doors/p5. CTCFE incorpore autant d'observations que Contingent-FF pour le problème *colorballs*/p21*, et autant que Contingent-FF et PO-PRP pour le problème *erovers/p6*.

Problèmes coûteux

Intéressons-nous à présent à la dernière classe de problèmes à calculer : les problèmes coûteux. Les résultats de CTCFE sur ce type de problème sont contenus dans le tableau B.6 situé en annexe de ce manuscrit. En comparant les résultats du tableau B.6 avec les résultats de CTCF, Contingent-FF, CLG et PO-PRP du tableau B.5, nous pouvons nous rendre compte dans un premier temps que CTCFE ne résout pas plus de problèmes que CTCF dans cette classe de problème. Tout comme CTCF, CTCFE ne peut résoudre que les problèmes *coûteux* conformants (*btcs+* et *rovers+*) compte tenu de la méthode de recherche d'observation, cette méthode n'étant basée que sur le plan retourné par $CPCES_E$. Ce plan ne contient pas l'action préalable nécessaire à l'exécution d'une observation dans ce type de problème, ce qui rend impossible la sélection d'observation.

En observant les temps de calcul des plans pour les 8 problèmes que CTCFE et CTCF peuvent résoudre, on peut se rendre compte que CTCFE calcule des plans plus rapidement que CTCF pour les 4 problèmes *btcs+*, tandis que pour le problème *rovers+/p6*, CTCFE met plus de temps que CTCF à calculer une solution. Pour les trois autres problèmes *rovers+*, CTCF et CTCFE ont un temps de calcul similaire. L'écart entre les temps de calcul de CTCF et CTCFE sur ces problèmes étant petits, les conclusions de la comparaison de temps de calcul entre CTCFE, Contingent-FF, CLG et PO-PRP sont les mêmes que pour CTCF.

Pour ce qui est de la taille des plans calculés, CTCFE calcule un plan plus petit que CTCF pour le problème *rovers+/p6*, et calcule des plans de même taille que CTCF pour les autres problèmes. CTCFE calcule aussi un plan plus court que CTCF pour *rovers+/p6*.

Enfin, CTCFE incorpore le même nombre d'observations que CTCF pour l'ensemble des problèmes.

Conclusions expérimentales

Après avoir analysé la comparaison entre CTCFE et CTCF sur des problèmes contingents de la littérature, les problèmes de largeur contingente supérieure à 1 et les problèmes *coûteux*, nous pouvons effectuer une conclusion en plusieurs points de ces résultats expérimentaux. Dans un premier temps, on peut remarquer que CTCFE souffre du même problème de complétude que CTCF, celui-ci ne parvenant pas à résoudre beaucoup plus de problèmes pour les mêmes raisons que CTCF, à savoir l'impossibilité de réaliser des observations indirectes, ou bien encore l'impossibilité de réaliser une observation sur le chemin que forme l'exécution du plan retourné par $CPCES_E$. On peut aussi conclure que dans la plupart des problèmes contingents, CTCFE est légèrement plus rapide que CTCF pour trouver une solution. Cependant, cette différence de temps de calcul n'est pas suffisante pour se rapprocher des temps de calculs de Contingent-FF, CLG et PO-PRP. De plus, cette légère amélioration du temps de calcul ne suffit pas à trouver une solution dans le temps imparti pour les problèmes de grande taille. Pour ce qui est des problèmes comportant une solution conformante, CTCFE calcule parfois des plans conformants légèrement plus lentement que CTCF, ce qui peut s'expliquer par une légère perte de performance en temps de calcul de $CPCES_E$ lors de la sélection du contre-exemple.

Dans la globalité, la taille et la profondeur des plans générés par CTCFE n'est que légèrement meilleure que CTCF sur l'ensemble des problèmes évalués. La plupart du temps les tailles et profondeurs des plans sont équivalents, et se retrouvent parfois moins bonnes que CTCF pour un petit nombre de problèmes.

En ce qui concerne l'ajout d'observations, on peut noter que CTCFE ajoute le même nombre d'observations que CTCF dans l'ensemble des problèmes, gardant la même efficacité pour limiter le nombre d'observations total du plan.

6.5 Discussion

Cette nouvelle version de l'approche est un peu plus dépendante de $CPCES_E$ que CTCF de par la modification de la manière dont CPCES retourne les contre-exemples. Cette nouvelle version est donc plus difficilement compatible avec l'utilisation d'autres planificateurs conformants que CPCES, ces planificateurs devant retourner un contre-exemple assurant qu'il n'en existe aucun autre faisant échouer le plan plus tôt.

CTCFE souffre des mêmes problèmes de complétude que CTCF, étant donné que la réussite de la recherche d'observations est toujours autant conditionnée par le fait que le plan retourné par CPCES soit proche de la solution finale, et que les observations soient applicables sur le chemin d'exécution de ce plan. De plus, CTCFE n'implémente toujours pas de retour possible sur le choix des observations.

Contrairement à CTCF, CTCFE est grandement dépendante de CPCES. Les avantages de temps de calcul de CTCFE par rapport à CTCF n'étant pas assez importants pour contrebalancer cette dépendance, nous ne pouvons pas considérer CTCFE comme une amélioration majeure de CTCF, lui faisant perdre en généralité sans gagner un grand avantage en temps de calcul.

Nous devons donc nous pencher sur les autres pistes d'amélioration de CTCF, parmi lesquelles nous pouvons citer la réduction des contraintes de but du problème menant à l'observation, la représentation plus compacte des états de croyance, ou encore l'amélioration de la complétude de l'approche par un retour possible sur le choix des observations.

6.6 Conclusion

Dans ce chapitre, nous avons détaillé une nouvelle version de CTCF, nommée CTCFE, dans laquelle le contre-exemple n'est plus sélectionné par CPCES de manière aléatoire parmi l'ensemble des contre-exemples. À la place, $CPCES_E$ sélectionne le contre-exemple dans lequel le plan précédemment calculé échoue le plus tôt. Ce changement permet d'obtenir un début de plan conformant correspondant à la séquence d'actions s'arrêtant juste avant l'action échouant dans le contre-exemple. Ce début de plan conformant est ensuite utilisé afin de déterminer si une observation est applicable à la suite d'une de ses actions et auquel cas ce plan est utilisé comme plan menant à l'observation, évitant le calcul d'un nouveau plan.

Les différents résultats expérimentaux montrent que cette nouvelle version de l'approche est légèrement plus efficace en terme de temps de calcul que CTCF pour les problèmes dotés d'une solution contingente. Cette différence de temps de calcul entre CTCF et CTCFE semble grandir en fonction de la taille du plan final et du nombre de branches à calculer. Néanmoins, ce temps de calcul n'est toujours pas assez efficace pour pouvoir résoudre des problèmes de grande taille dans le temps imparti. CTCFE n'est pas plus efficace pour les problèmes dont la solution peut être conformante, la modification de CPCES impactant légèrement le temps de calcul de $CPCES_E$. CTCFE calcule généralement des plans de taille et de profondeur équivalente à CTCF et n'incorpore pas plus d'observations dans le plan. Cependant, CTCFE souffre des mêmes problèmes de complétude que CTCF, et ne peut pas trouver de solution pour l'ensemble des problèmes. De plus, CTCFE est dépendant de $CPCES_E$, celui-ci reposant entièrement sur l'hypothèse que le planificateur conformant retourne un contre-exemple faisant échouer le plan le plus tôt. Cet inconvénient n'étant pas suffisamment compensé par le gain de temps de calcul des plans, nous devons explorer les autres pistes d'améliorations de CTCF.

Dans le prochain chapitre, nous allons nous intéresser à l'une des pistes d'amélioration de CTCF. Pour cela, nous allons étudier une des possibilités évoquées dans la section 5.5 du chapitre 5 de ce manuscrit, à savoir la modification des contraintes de but lors de la modélisation du problème de calcul du plan menant à l'observation. Dans cette version, on ne considère plus le but comme un état de croyance précis mais plutôt comme un ensemble restreint de propositions correspondants aux préconditions de l'observation à réaliser.

Dans les chapitres suivants, nous étudierons les autres améliorations envisagées, en commençant par l'étude d'une représentation différente des états de croyance consistant à associer l'état de croyance initial et la séquence d'actions/observations menant à l'état de croyance courant (Chapitre 8). Finalement, nous terminerons cette thèse en rendant l'approche complète grace à un retour possible sur le choix des observations (Chapitre 9).

Chapitre 7

Réduction des contraintes de but menant à l'observation de CTCF

7.1 Motivation

Dans les deux précédents chapitres, nous avons détaillé deux approches, CTCF et CTCFE. Ces deux versions précédentes de l'approche ne parviennent pas à résoudre un grand nombre de problèmes. Ceci est lié au fait que CTCF et CTCFE ne parviennent pas toujours à trouver une observation applicable dans l'ensemble des états de croyances calculés lors de la recherche d'observation, ou bien ne parviennent pas à calculer un plan menant à l'état de croyance bien précis dans lequel nous voulons réaliser l'observation. Afin de trouver un plan menant à une observation, dans CTCF et CTCFE on tente d'identifier un état de croyance parmi les états de croyance calculés par l'application du plan défaillant de CPCES dans lequel l'observation utile serait applicable. Un des problèmes de cette manière de raisonner, est qu'elle repose sur l'hypothèse que les observations nécessaires peuvent être réalisées sur le chemin d'exécution formé par le plan défaillant retourné par CPCES, or cette hypothèse ne se vérifie pas pour un grand nombre de problèmes. Cette hypothèse entraîne un problème de complétude, un nombre restreint de solutions étant considérées. Parmi les problèmes ne pouvant pas être résolus par CTCF et CTCFE, on peut compter l'ensemble des problèmes *coûteux* présentés dans ce manuscrit, dans lesquels une des préconditions des observations est une proposition obtenue seulement par l'application d'une action ne servant qu'à permettre l'exécution d'une observation, et qui n'est donc pas présente dans le plan non conformant retourné par CPCES. On peut aussi citer des problèmes de la littérature comme *colorballs* ou *doors*. La méthode de sélection du contre-exemple de CPCES étant aléatoire, CTCF ne peut pas résoudre *colorballs* si le contre-exemple du plan retourné par CPCES concerne la couleur de la balle avant que l'incertitude de sa position ne soit levée, la partie conformante du plan de CPCES ne comportant pas l'action de ramasser la balle, rendant impossible l'application de l'observation de la couleur de la balle. De même, pour *doors*, le contre-exemple retourné par CPCES concerne l'observation d'une porte en particulier, cependant le chemin menant à cette porte peut comporter des portes dont l'ouverture est incertaine, rendant impossible l'accès à l'observation voulue.

Nous avons discuté dans la section 5.5 du chapitre 5 de ce manuscrit de la possibilité de modifier les contraintes de but définies lors de la modélisation du problème de calcul du plan menant à l'observation, en ne considérant plus le but comme un état de croyance précis mais plutôt comme un ensemble de propositions correspondants aux préconditions possibles de l'observation à réaliser. Cette modification doit en théorie rendre le plan menant à l'observation plus facile à calculer, étant donné que les contraintes de but sont bien moins exigeantes, et ne reposent pas intégralement sur le plan précédemment calculé par CPCES. Cette amélioration du but doit permettre à cette version de l'approche de s'approcher de la complétude en permettant la résolution d'un plus grand nombre de problèmes. Dans ce chapitre, nous allons détailler la mise en oeuvre et l'application d'une nouvelle approche implémentant cette modification, que l'on nommera CTCF+.

7.2 Mise en oeuvre

Contrairement à CTCFE, le développement de cette nouvelle version de l'approche n'implique pas de réaliser des modifications du planificateur conformant utilisé. Les modifications se concentrent donc sur le processus contingent et la procédure de recherche d'observations. Afin de restreindre le nombre de propositions contenues dans le but du problème, mais aussi de permettre de considérer des observations non atteignables dans le chemin d'exécution du plan retourné par CPCES, nous ne devons plus sélectionner une observation bien précise et un état de croyance dans lequel la réaliser. Au contraire, nous décidons de sélectionner toutes les observations permettant de réaliser l'observation d'une proposition souhaitée indépendamment du fait qu'elles soient applicables ou non dans un des états de croyance formés par l'application du plan π retourné par CPCES à l'état de croyance initial. Cette sélection d'observation est décrite dans la section 7.2.1 de ce chapitre. Une fois que les observations candidates ont été sélectionnées, le but du problème est représenté par une disjonction des préconditions des observations candidates, afin de permettre le calcul d'un plan rejoignant un état de croyance contenant les préconditions d'une de ces observations, au lieu d'un état de croyance complet précis et déterminé par nos soins. Cette nouvelle représentation du but du problème menant à l'observation candidate et sa gestion par la procédure contingente est décrite dans la section 7.2.3 de ce chapitre. Dans la section 7.2.4, nous verrons comment ces modifications impactent l'architecture générale de notre approche. Dans la section 7.3, nous évaluerons notre approche de manière théorique, puis nous évaluerons son implémentation dans la section 7.4. Finalement, nous conclurons dans les sections 7.5 et

7.6 de ce chapitre.

7.2.1 Sélection d'un ensemble d'observations candidates

Dans cette nouvelle version de l'approche, nous souhaitons modifier le processus de sélection de l'observation permettant de connaître les valeurs de vérité des propositions empêchant CPCES de trouver un plan conformant, ce processus étant restreint à la sélection d'une seule observation dont les paramètres sont définis, limitant la possibilité de trouver un plan menant à cette observation. De plus, le fait de déterminer nous-même dans quel état de croyance nous voulons appliquer l'observation en nous basant sur le plan π défaillant de CPCES limite la possibilité de trouver une observation applicable, et donc une solution au problème. À la place, nous décidons de sélectionner non plus une, mais l'ensemble des observations paramétrées (dont les paramètres sont définis) $\Omega_{\pi,\gamma}$ de l'ensemble des observations Ω capables de révéler les valeurs de vérité de la précondition qui nous est nécessaire. Contrairement à CTCF, nous ne contrôlons pas l'applicabilité de ces observations dans un état de croyance, et nous laissons à CPCES la liberté de trouver un plan rejoignant les préconditions d'une de ces observations, permettant une plus grande liberté de calcul du plan pour CPCES. Cette représentation différente du but du problème de calcul d'un plan menant à l'observation est décrite dans la section 7.2.2 de ce chapitre.

Nous avons implémenté ce processus de sélection des observations dans la fonction FINDOBSERVATION+ illustrée par l'algorithme 8 dénotant les différences entre FINDOBSERVATION+ et la fonction FINDOBSERVATION (algorithme 4). Dans l'algorithme 8, on ne s'intéresse plus à l'application des actions et des observations à un ensemble d'états de croyance calculés mais seulement à l'application des actions du plan π défaillant retourné par CPCES sur l'état de contre-exemple γ. Dans un premier temps, tout comme dans CTCF, on vérifie quelle action de π n'est pas applicable dans γ et on récupère ses préconditions (lignes 3 à 11). Contrairement à CTCF, on ne garde plus en mémoire les états de croyance calculés par l'application des actions du plan à l'état de croyance initial. On récupère ensuite l'ensemble des observations de Ω permettant l'observation d'une des préconditions de l'action échouant (lignes 14 à 16), puis on retourne cet ensemble (ligne 21).

Algorithm 8 FINDOBSERVATION+

Require: $\mathcal{P}_\Omega, \pi, \gamma$
Ensure: $(\mathcal{B}_\sigma, \sigma)$
1: ~~$beliefList := [\mathcal{B}_0]$~~
2: ~~$\mathcal{B} := \mathcal{B}_0$~~
3: **for** a in π **do**
4: **if** a applicable in γ **then**
5: $\gamma := T(\gamma, a)$
6: **if** ~~a applicable in $\mathcal{B}$~~ **then**
7: **for** ~~$s \in \mathcal{B}$~~ **do**
8: ~~$\mathcal{B}' := \{T(s,a) | s \in \mathcal{B}\}$~~
9: ~~$beliefList := beliefList + \mathcal{B}'$~~
10: **else**
11: let $unsatPre$ be the unsatisfied preconditions of a
12: **break**
13: $\Omega_{\pi,\gamma} = \emptyset$
14: **for** p in $unsatPre$ **do**
15: let $\Omega_p \in \Omega$ be a set of observation for p
16: $\Omega_{\pi,\gamma} = \Omega_{\pi,\gamma} + \Omega_p$
17: **for** ~~$\sigma \in \Omega_p$~~ **do**
18: ~~**for** $\mathcal{B}_\sigma$ in $beliefList$ **do**~~
19: ~~**if** σ applicable in $\mathcal{B}_\sigma$ **do**~~
20: ~~**return** $(\mathcal{B}_\sigma, \sigma)$~~
21: **return** $(\Omega_{\pi,\gamma})$
22: ~~**return** $(\mathcal{B}, None)$~~

7.2.2 But du problème de planification menant à l'observation

Dans CTCF, le but g_σ du problème de calcul de plan menant à l'observation σ sélectionnée est modélisé comme la disjonction des états de l'état de croyance $\mathcal{B}_\sigma$ dans lequel l'observation σ peut être appliquée. Si $\mathcal{B}_\sigma = \{s_0, ..., s_n\}$ alors on a $g_\sigma = \{s_0 \vee s_1 \vee ... \vee s_n\}$. Cette représentation permet d'être certain que les branches du plan menant à l'observation atteindront un seul et même état de croyance, donnant une structure de graphe au plan calculé. Cependant, plus la taille d'un état de croyance est grand, plus cette représentation est lourde en place mémoire, et plus les contraintes de but sont nombreuses et difficiles à atteindre pour CPCES. De plus, représenter le but de cette manière implique de ne pas laisser la liberté à CPCES de choisir quelle observation parmi la liste des observations potentielles peut être plus facilement atteinte. Au contraire, une observation candidate est sélectionnée directement par la fonction FINDOBSERVATION sans aucune liberté de choix.

C'est justement pour laisser une plus grande liberté de calcul à CPCES et réduire les manipulations d'états de croyance, que nous avons décidé de représenter le but du problème de calcul de plan menant à l'observation comme la disjonction des préconditions des différentes observations candidates de l'ensemble $\Omega_{\pi,\gamma}$ retourné par la procédure FINDOBSERVATION+. Nous définissons alors pour $\Omega_{\pi,\gamma} = \{\sigma_0, ..., \sigma_m\}$ un nouveau but $g_\sigma = \{Pre(\sigma_0) \vee Pre(\sigma_1) \vee ... \vee Pre(\sigma_m)\}$.

Exemple: Modélisation du but de CTCF+ sur un exemple

Reprenons l'exemple de l'observation *obs_obst rob cell*10 *cell*11 de la figure 5.3. Dans CTCF, le but du problème menant à l'observation est modélisé comme une disjonction des différents états composants l'état de croyance $\mathcal{B}_\sigma$ dans lequel nous voulons réaliser l'observation. Cet état de croyance est rappelé dans la figure 7.1 ci-dessous.

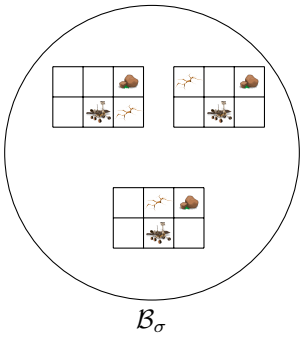

FIGURE 7.1 – Rappel état de croyance $\mathcal{B}_\sigma$ de l'exemple

Dans CTCF, nous avons donc le but g_σ du problème de recherche de plan menant à l'observation suivant :
$g_\sigma = \{(at\_robot\ rob\ cell10)(is\_obstacle\ cell20)(at\_rock\ cell21)(adjacent\ cell00\ cell10)...\}$
$\vee\ \{(at\_robot\ rob\ cell10)(is\_obstacle\ cell01)(at\_rock\ cell21)(adjacent\ cell00\ cell10)...\}$
$\vee\ \{(at\_robot\ rob\ cell10)(is\_obstacle\ cell11)(at\_rock\ cell21)(adjacent\ cell00\ cell10)...\}$
La taille de g_σ est donc de trois états de 17 propositions (un état complet est détaillé dans le chapitre 2 de l'Etat de l'art). g_σ a donc une taille totale de 51 propositions.

Au contraire, dans CTCF+ le but n'est pas modélisé comme une disjonction d'états, mais comme une disjonction d'un ensemble plus restreint de propositions correspondants aux préconditions des différentes observations utiles retournées par la nouvelle procédure FINDOBSERVATION+.

Reprenons l'exemple de l'ensemble d'observations *obs_obst* possibles $\Omega_{\pi,\gamma}$ de la figure 5.12. Considérons $\Omega_{\pi,\gamma} = \{< obs\_obst\ rob\ cell01\ cell11 >, < obs\_obst\ rob\ cell21\ cell11 >, < obs\_obst\ rob\ cell10\ cell11 >\}$ l'ensemble d'observations retourné par la nouvelle fonction FINDOBSERVATION+. Dans ce cas, le but du problème de calcul d'un plan menant à l'une des observations de $\Omega_{\pi,\gamma}$ devient $g_\sigma = \{Pre(< obs\_obst\ rob\ cell01\ cell11 >)\ or\ Pre(< obs\_obst\ rob\ cell21\ cell11 >)$

or $Pre(< obs\_obst\ rob\ cell10\ cell11 >)$}. Si on reprend l'exemple d'observation PDDL de la section 3.2.2 du chapitre 3.2 de l'Etat de l'art, chaque observation comporte deux propositions en préconditions. La taille de g_σ dans CTCF+ sera donc dans notre exemple de 6 propositions.

7.2.3 Adaptation de la procédure contingente à la nouvelle sélection des observations

Nous allons à présent nous intéresser aux différentes modifications de la procédure contingente permettant de traiter cette nouvelle sélection d'observations et cette représentation différente du but du problème de calcul de plan menant à l'observation. Ces modifications sont représentées dans l'algorithme 9.

Dans un premier temps, tout comme les versions précédentes de l'approche, CPCES est appelé afin de calculer un plan conformant solution du problème initial, ou bien un plan conformant pour une partie des états de l'état de croyance, et un état de contre-exemple dans lequel ce plan n'est pas entièrement applicable. La première modification notable se situe dans la procédure FINDOBSERVATION+ (ligne 5), qui ne retourne plus un état de croyance et une observation, mais simplement l'ensemble des observations candidates $\Omega_{\pi,\gamma}$ permettant d'observer une des préconditions incertaines faisant échouer le plan retourné par CPCES dans γ. La nouvelle procédure FINDOBSERVATION+ est décrite dans l'algorithme 8.

Ligne 7, le but g_σ est défini comme une disjonction des préconditions de chaque observation σ de l'ensemble $\Omega_{\pi,\gamma}$ retourné par FINDOBSERVATION+. La procédure contingente est ensuite appelée pour calculer un plan menant à au moins une des observations $\sigma \in \Omega_{\pi,\gamma}$ à partir du problème initial dans lequel le but a été remplacé par le but g_σ (ligne 9).

Étant donné que contrairement à CTCF, le but du problème menant à l'observation n'est plus un état de croyance bien défini mais seulement un petit ensemble de propositions, chaque branche du plan calculé π_σ ne mène pas forcément à un même état de croyance, et donc pas forcément à la même observation σ de $\Omega_{\pi,\gamma}$.

Dans CTCF, chaque branche de π_σ arrive dans un état de croyance complet $\mathcal{B}_\sigma$ totalement défini par l'ensemble de ses états. En revanche, dans CTCF+, étant donné que le but est un ensemble restreint de propositions n'établissant pas de contraintes sur l'intégralité de l'état de croyance d'arrivée de chaque branche, l'état de croyance d'arrivée n'est que partiellement défini par g_σ. Par conséquent, il se peut que l'état de croyance d'arrivée de chaque branche de π_σ satisfaisant les contraintes de g_σ ne soit pas le même pour chaque branche.

Cette particularité implique qu'un nouveau plan menant au but du problème initial doit être calculé pour chaque branche de π_σ (ligne 10). Pour cela, on calcule l'état de croyance courant $\mathcal{B}_\sigma$ en appliquant chaque action de la branche π de π_σ à l'état de croyance initial $\mathcal{B}_0$ (ligne 11). La première observation σ de $\Omega_{\pi,\gamma}$ applicable dans $\mathcal{B}_\sigma$ est ensuite sélectionnée (ligne 12). Tout comme CTCF, les états de croyance $\mathcal{B}^+$ et $\mathcal{B}^-$ correspondant à un résultat positif ou négatif de l'observation sont calculés à partir de cet état de croyance $\mathcal{B}_\sigma$ en supprimant les états de $\mathcal{B}_\sigma$ incompatibles avec le résultat de l'observation σ (lignes 13 et 15). La procédure contingente est appelée à nouveau pour calculer les branches positives et négatives menant au but du problème initial à partir de $\mathcal{B}^+$ et $\mathcal{B}^-$ (lignes 14 et 16). Finalement, chaque branche est connectée à σ puis à la branche π courante (ligne 17) et le processus recommence pour chaque branche de π_σ. Une fois que toutes les branches de π_σ mènent au but du problème initial, alors π_σ est retourné comme solution du problème (ligne 18).

Algorithm 9 ContingentPlanning Procedure

Require: $\mathcal{P}_\Omega = (\Sigma_\Omega, \mathcal{B}_0, g)$
Ensure: π_c
1: $\mathcal{P} = (S, A, T, \mathcal{B}_0, g)$
2: $\pi, \gamma :=$ CPCES$(\mathcal{P})$
3: **if** $\gamma = \emptyset$ **then**
4: **return** π
5: ~~$\mathcal{B}_\sigma, \sigma :=$ findObservation$(\mathcal{P}_\Omega, \pi, \gamma)$~~
6: $\Omega_{\pi,\gamma} :=$ findObservation+$(\mathcal{P}_\Omega, \pi, \gamma)$
7: let g_σ be the disjunction of each Pre(σ) in $\Omega_{\pi,\gamma}$
8: ~~$\pi_\sigma :=$ ContingentPlanning$((\Sigma_\Omega, \mathcal{B}_0, \mathcal{B}_\sigma))$~~
9: $\pi_\sigma :=$ ContingentPlanning$((\Sigma_\Omega, \mathcal{B}_0, g_\sigma))$
10: **for** each branch π of π_σ **do**
11: $\mathcal{B}_\sigma := T(\mathcal{B}_0, \pi)$
12: let σ be the observation of $\Omega_{\pi,\gamma}$ applicable in $\mathcal{B}_\sigma$
13: $\mathcal{B}^+ := T(\mathcal{B}_\sigma, \sigma)$ *with* $eff(\sigma) = \top$
14: $\pi_p :=$ ContingentPlanning$((\Sigma_\Omega, \mathcal{B}^+, g))$
15: $\mathcal{B}^- := T(\mathcal{B}_\sigma, \sigma)$ *with* $eff(\sigma) = \bot$
16: $\pi_n :=$ ContingentPlanning$((\Sigma_\Omega, \mathcal{B}^-, g))$
17: $\pi :=$ (π;**if** σ **then** π_p **else** π_n)
18: **return** π_σ

7.2.4 Modification de l'architecture

L'architecture de CTCF+ n'est pas très différente de celle de CTCF. Cette nouvelle architecture est illustrée par la figure 7.2 dans laquelle on peut voir que la seule différence entre les architectures de CTCF et de CTCF+ réside dans la communication entre la procédure findObservation+ et la procédure contingente. On peut remarquer que CTCF+ réduit le nombre d'intéractions entre ces deux procédures, la nouvelle procédure findObservation+ ne transmettant plus un état de croyance et une observation candidate à la procédure contingente, mais un ensemble d'observations candidates.

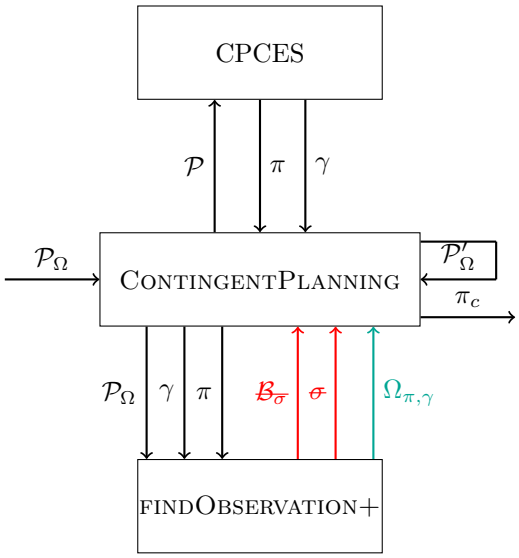

Figure 7.2 – Architecture du processus contingent

7.3 Analyse théorique

7.3.1 Complétude

Le fait de sélectionner un ensemble d'observations possibles au lieu d'une seule dans CTCF, et de représenter le but du problème de calcul du plan menant à une de ces observations comme la disjonction de leurs préconditions et non comme un état de croyance unique, permet de résoudre des problèmes que CTCF ne pouvait pas résoudre. Dans ces problèmes, l'observation nécessaire à lever l'incertitude n'est pas applicable dans le chemin d'exécution du plan retourné par CPCES. Même si CTCF+ peut résoudre plus de problèmes que CTCF, celui-ci n'est toujours pas complet, n'étant pas capable de résoudre des problèmes dans lesquels les préconditions de l'action échouant ne peuvent pas être observées directement, rendant impossible le traitement des problèmes *wumpus* et *wumpus**. De plus, CTCF+ n'implémente pas de retour sur le choix de l'observation réalisée au cas où on ne puisse finalement pas construire de plan après avoir réalisé cette observation.

7.3.2 Complexité

Dans cette section, nous allons déterminer quelle est la complexité de CTCF+ en analysant les différents algorithmes présentés dans ce chapitre. Comme pour les chapitres précédents, nous simplifions les calculs en considérant l'application d'une action ou la vérification et la sélection d'une observation comme une opération unitaire, et nous sur-approximons les équations en ne gardant que les variables de plus grand ordre de grandeur.

Étudions tout d'abord l'algorithme FINDOBSERVATION+ (algorithme 8). Dans un premier temps, la procédure de sélection des observations effectue une boucle dans laquelle dans le pire cas, $|\pi|$ actions sont appliquées à un seul état. Nous avons donc $|\pi|$ opérations pour cette première boucle.

Pour la deuxième partie de l'algorithme, nous avons une seule boucle qui parcourt les propositions de $UnsatPre$. Nous avons donc $|UnsatPre|$ opérations, qui dans le pire cas est égal à $|F|$. Le nombre d'opérations de la procédure EARLYFINDOBSERVATION est donc proportionnel à : $|\pi| + |F|$. La complexité algorithmique est donc en notation de Landau : $\mathcal{O}(|\pi| + |F|)$.

Notons $\mathcal{C}(n)$ la complexité de CPCES prenant n états en entrée, $\mathcal{F}(n)$ la complexité de FINDOBSERVATION+ prenant n états en entrée, et $\mathcal{T}(n)$ la complexité de la procédure contingente prenant en entrée n états. La complexité dans le pire des cas de la procédure contingente (algorithme 9) peut être exprimée par l'équation :

$$\mathcal{T}(n) = \mathcal{C}(n) + \mathcal{F}(n) + \mathcal{T}'(n) + k \times (n \times |\pi| + 2\mathcal{T}(n/2)) \tag{7.1}$$

Dans l'équation 7.1, on retrouve l'élément $\mathcal{T}'(n)$ de l'équation 5.4 qui correspond à l'appel de la procédure contingente pour calculer un plan menant à une des observations de $\Omega_{\pi,\gamma}$ (ligne 9), k correspond au nombre de branches de π_σ impliquées par la boucle de la ligne 10, $n \times |\pi|$ correspond à l'application de la branche π à l'état de croyance initial $\mathcal{B}_0$ afin de former l'état de croyance courant (ligne 11), et $2\mathcal{T}(n/2))$ correspond aux deux appels de la procédure contingente pour résoudre les 2 sous-problèmes de calcul d'un plan menant au but pour chaque branche du plan (lignes 14 et 16).

On a $\mathcal{O}(k \times n \times |\pi|) = \mathcal{O}(max(k, n, |\pi|))$. Dans le pire cas on a $n = 2^{|F|}$, donc $\mathcal{O}(max(k, n, |\pi|) = \mathcal{O}(n)$. On a donc une complexité en $\mathcal{O}(\mathcal{C}(n) + \mathcal{T}'(n) + \mathcal{O}(n))$, qui est dominée encore une fois par $\mathcal{C}(n)$, donnant une complexité dans le pire cas de $\mathcal{T}(n) = \mathcal{O}(\mathcal{C}(n))$.

Dans le pire cas, la complexité de CTCF+ est la même que CTCF. Cependant, on peut supposer que le fait d'ajouter un facteur k correspondant au parcours des branches de π_σ pour le calcul des sous-problèmes rend le temps de calcul de CTCF+ légèrement plus grand que CTCF à cause de cette boucle.

7.4 Expérimentations

Les expérimentations de l'implémentation de CTCF+ se sont déroulées dans les mêmes conditions que pour CTCF et CTCFE. Une limite de temps de 10 minutes maximum est imposée lors de la résolution des problèmes. Nous avons décidé de comparer cette nouvelle approche avec CTCF, CTCFE, Contingent-FF, CLG et PO-PRP sur les mêmes benchmarks : des problèmes de la littérature, des problèmes dont la largeur contingente est supérieure à 1, et des problèmes *coûteux*. L'intégralité des benchmarks considérés sont décrits en annexe de ce manuscrit.

7.4.1 Résultats

Nous allons tout d'abord comparer CTCF+ avec CTCF, CTCFE, Contingent-FF, CLG et PO-PRP sur les problèmes de la littérature étudiés. Nous étudierons par la suite les problèmes de largeur contingente supérieure à 1 et enfin nous étudierons les problèmes *coûteux*. Les résultats des problèmes de la littérature sont illustrés dans la figure 7.3 et les résultats des problèmes *coûteux* sont illustrés par la figure 7.4. Ces deux figures sont découpées en quatre graphiques représentant pour chaque problème le temps de calcul du plan en secondes, la taille du plan en nombre total d'actions et d'observations, la profondeur du plan (taille de la plus longue séquence d'actions et d'observations, et enfin le nombre total d'observations du plan. Dans ces graphiques, les problèmes sont triés de manière croissante selon un critère de difficulté choisi comme le produit du nombre de propositions par état et du nombre d'états composant l'état de croyance initial.

Problèmes de la littérature

Les résultats détaillés de l'évaluation de CTCF+ sur les problèmes de la littérature sont décrits dans les tableaux B.2 et B.1 situés en annexe de ce manuscrit. Dans la figure 7.3, on peut observer que CTCF+ résout plus de problèmes que CTCF et CTCFE. CTCF+ résout 7 problèmes de plus que CTCF et 9 problèmes de plus que CTCFE. Les seuls problèmes que CTCF+ ne peut pas résoudre sont les problèmes *wumpus* du fait de la nécessité de réaliser des observations indirectes, et *erovers/p120* et *p500* pour lesquels CTCF+ dépasse le temps limite imparti pour calculer un plan.

Dans le premier graphique de la figure 7.3 et dans le tableau B.2, on peut observer que le temps de calcul de CTCF+ est inférieur à CTCFE et CTCF pour 14 problèmes : *btcs/p10*, *p50*, *p70*, l'ensemble des problèmes *grid* et *rovers*, *logistics/p1*, *p3* et *p7*, *egrid/p4* et *colorballs/p41*. Parmi ces problèmes, CTCF+ calcule des plans plus rapidement que Contingent-FF sur les problèmes *btcs/p70*, *grid/p3* et *p5*. De plus, CTCF+ calcule un plan plus rapidement que Contingent-FF et CLG pour le problème *grid/p4*. On peut noter que CTCF+ calcule des plans moins rapidement que CTCFE et CTCF pour 11 problèmes : les problèmes *blocks/p3* et *p15*, *ebtcs/p10*, *p30*, *p50*, *erovers/p2*, *p4* et l'ensemble des problèmes *elogistics*. Pour les problèmes que CTCF+ peut résoudre et non CTCFE et CTCF, CTCF+ est plus rapide que Contingent-FF pour le problème *egrid/p3*, et moins rapide que Contingent-FF, CLG et PO-PRP sur l'ensemble des autres problèmes.

Dans le deuxième graphique de la figure 7.3 et dans le tableau B.2, nous pouvons constater que CTCF+ calcule des plans de plus petite taille que CTCFE et CTCF pour le problème *blocks/p11*, de plus petite taille que CTCFE pour *erovers/p40* et de plus petite taille que CTCF pour *colorballs/p41*. Cependant, CTCF+ calcule des plans de plus grande taille que CTCFE pour *grid/p3*, *blocks/p7*, *erovers/p20*, *p60*, *p250* et *elogistics/p7* (pour lesquels CTCF+ a la même performance que CTCF), de plus grande taille que CTCF pour *egrid/p4*, et de plus grande taille que CTCFE et CTCF pour *elogistics/p5*. Parmi les problèmes que CTCF+ peut résoudre et non CTCFE et CTCF, CTCF+ calcule des plans plus petits que Contingent-FF et CLG pour les problèmes *egrid/p3* et *p5*, *colorballs/p31*, *p41* et *p51*, plus petits que Contingent-FF pour *colorballs/p21*, et plus petit que PO-PRP pour *laboratory*.

Dans le troisième graphique de la figure 7.3 et dans le tableau B.2, nous pouvons observer que CTCF+ calcule des plans plus courts que CTCFE et CTCF pour 8 problèmes : les problèmes *blocks/p11*, *erovers/p2*, *p4*, *p200*, *p250* et *p300*, *elogistics/p5* et *colorballs/p41*. CTCF+ calcule des plans plus courts que CTCFE pour *grid/p5* et *elogistics/p7*. Cependant, CTCF+ calcule des plans plus longs que CTCFE pour *grid/p3*, *erovers/p20*, *p60* et *elogistics/p3*. CTCF+ calcule aussi des plans plus longs que CTCF pour le problème *egrid/p4*. Parmi ces problèmes, CTCF+ calcule des plans plus courts que Contingent-FF, CLG et PO-PRP pour les problèmes *erovers/p2*, *p4*, *p250* et *p300* et *colorballs/p41*. CTCF+ calcule un plan plus court que Contingent-FF et PO-PRP pour le problème *elogistics/p5*, et CTCF+ calcule des plans plus courts que PO-PRP pour les problèmes *blocks/p11* et *erovers/p200*. Parmi les problèmes pour lesquels CTCF et CTCFE ne trouvent pas de solution, CTCF+ trouve des plans plus courts que Contingent-FF, CLG, et PO-PRP pour les problèmes *egrid/p3*, *egrid/p5* et *colorballs/p31*. CTCF+ calcule aussi des plans plus courts que Contingent-FF pour le problème *colorballs/p21* et *p51*. Enfin, CTCF+ calcule des plans plus courts que PO-PRP pour les problèmes *laboratory*.

Dans le dernier graphique de la figure 7.3 et dans le tableau B.2, nous pouvons remarquer que CTCF+ ajoute autant d'observations dans le plan que CTCF et CTCFE pour l'ensemble des problèmes que CTCF et CTCFE peuvent résoudre. Les comparaisons entre CTCF, Contingent-FF, CLG et

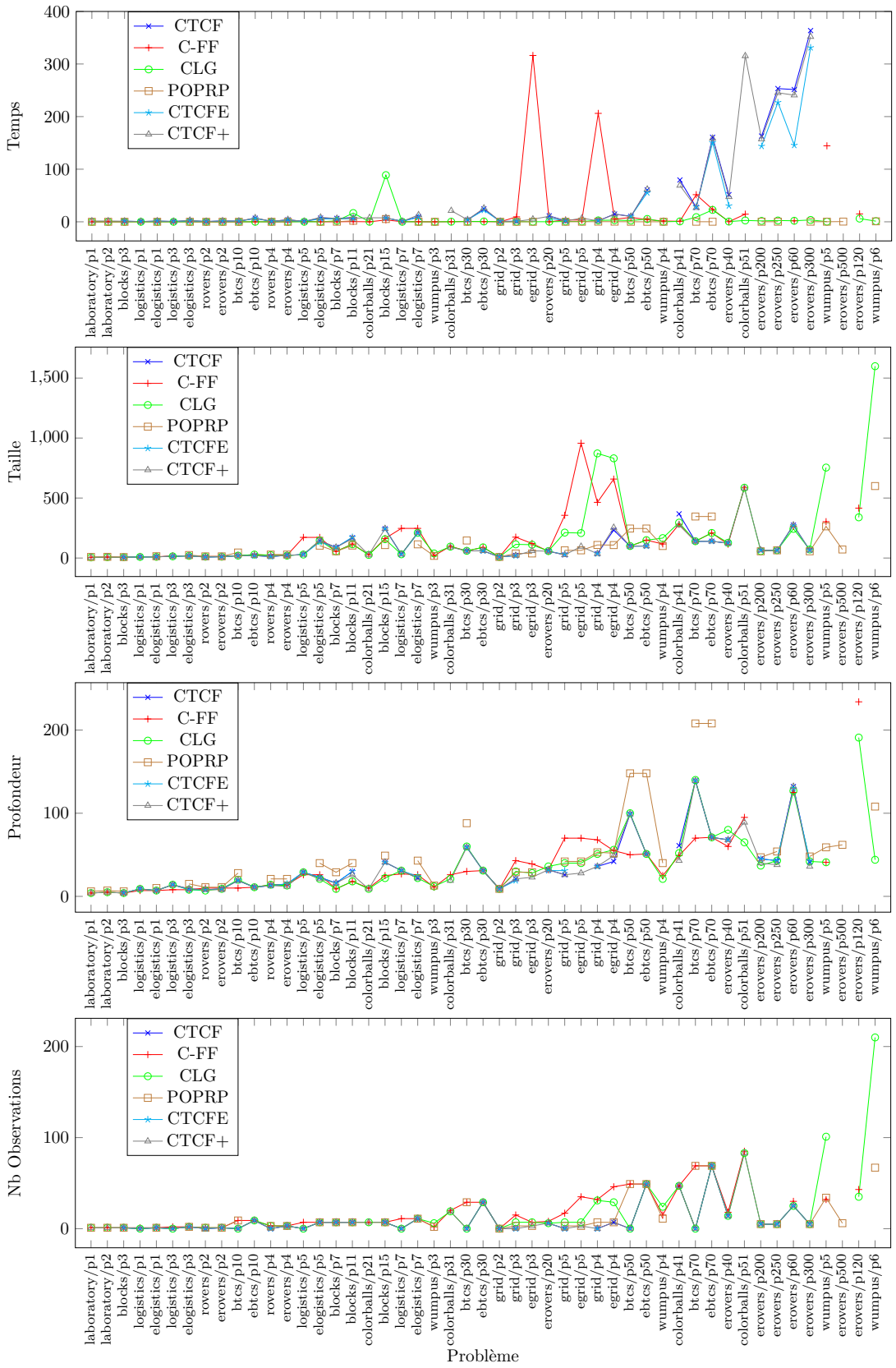

FIGURE 7.3 – Comparaison du temps de calcul, de la taille, de la profondeur, et du nombre d'observations du plan pour les problèmes de la littérature

PO-PRP sont donc les mêmes pour CTCF+ sur ces problèmes. En ce qui concerne les problèmes que CTCF et CTCFE ne peuvent pas résoudre, CTCF+ incorpore moins d'observations que Contingent-FF et CLG pour les problèmes *egrid/p3* et *p5*. CTCF+ incorpore moins d'observation que Contingent-FF sur les problèmes *colorballs/p31* et *p51*.

Problèmes de largeur contingente supérieure à 1

Concentrons-nous à présent sur les problèmes de largeur contingente supérieure à 1. Les résultats de l'implémentation de CTCF+ sur ces problèmes sont détaillés dans le tableau B.4. En comparant les résultats du tableau B.4 et du tableau B.3, nous pouvons remarquer que CTCF+ résout un problème de plus que CTCFE (*colorballs*/p31*) et 4 de plus que CTCF (*erovers/p6*, *doors/p5*, *colorballs*/p21* et *p31*).

Nous pouvons déterminer que CTCF+ calcule des plans plus rapidement que CTCFE et CTCF pour les problèmes *rovers/p6* et *p8*. Cependant, cette différence de temps de calcul ne suffit pas à battre Contingent-FF, CLG et PO-PRP sur ces deux problèmes. De plus, nous pouvons constater que CTCF+ calcule des plans plus lentement que CTCFE et CTCF pour les problèmes *erovers/p8*, *doors/p3* et *p4*. CTCF+ calcule aussi des plans plus lentement que CTCFE pour les problèmes *erovers/p6*, *doors/p5* et *colorballs*/p21*. Pour ce qui est du problème *colorballs*/p31*, CTCF+ calcule un plan plus lentement que Contingent-FF, tandis que CLG et PO-PRP ne peuvent pas résoudre ce problème.

En ce qui concerne la taille des plans calculés, CTCF+ calcule des plans plus petits que CTCFE pour *colorballs*/p21*, permettant de battre Contingent-FF sur ce problème. En revanche, CTCF+ calcule des plans plus grands que CTCFE pour *erovers/p6*, *doors/p4* et *p5* (même performance que CTCF). CTCF+ calcule aussi des plans plus grands que CTCFE et CTCF pour le problème *erovers/p8*. CTCF+ calcule un plan plus petit que Contingent-FF sur le problème *colorballs*/p31*.

Si on se concentre sur la profondeur des plans, on peut noter que CTCF+ calcule des plans plus courts que CTCFE et CTCF pour les problèmes *erovers/p6*, *erovers/p8* et *colorballs*/p21*. Pour ces problèmes, ainsi que pour le problème *colorballs*/p31*, CTCF+ calcule aussi des plans plus courts que Contingent-FF et PO-PRP. Cependant, CTCF+ calcule des plans plus longs que CTCFE pour les problèmes *doors/p4* et *p5*.

Enfin, étudions le nombre d'observations incorporées au plan. Nous pouvons remarquer que CTCF+ incorpore moins d'observations que CTCFE pour le problème *colorballs*/p21*. De plus, CTCF+ incorpore moins d'observations que Contingent-FF pour ce problème et le problème *colorballs*/p31*. Comme CTCF et CTCFE, CTCF+ incorpore moins d'observations que Contingent-FF sur les problèmes *rovers*, *doors/p3*, *p4* et *p5*. Pour l'ensemble des autres problèmes, CTCF+ incorpore autant d'observations que CTCF et CTCFE.

Problèmes coûteux

Étudions maintenant les résultats de CTCF+ sur la dernière catégorie de problèmes, les problèmes *coûteux*. Les résultats de CTCF+ sur les problèmes coûteux sont illustrés par la figure 7.4 et détaillés dans les tableaux B.6 et B.5. Dans un premier temps, nous pouvons remarquer que CTCF+ parvient à résoudre 11 problèmes coûteux de plus que CTCF et CTCFE : les problèmes *erovers+*, *elogistics+* et *doors+/p3*, *p4* et *p5*. Le seul problème que CTCF+ ne peut pas résoudre est *doors/p6*, pour lequel CTCF+ dépasse le temps limite imposé.

Si on s'intéresse au temps de calcul des plans, on peut constater que CTCF+ calcule des plans plus rapidement que CTCFE et CTCF pour les problèmes *btcs+/p70*, *rovers+/p6* et *p8* tandis que CTCF+ calcule des plans plus lentement que CTCFE et CTCF pour les problèmes *btcs+/p10*, *p30* et *p50*. CTCF+ calcule des plans plus rapidement que Contingent-FF pour les problèmes *btcs+/p50* et *p70*, *rovers+/p6*. En revanche, pour l'ensemble des problèmes, CTCF+ calcule des plans moins rapidement que CLG et PO-PRP.

Dans un deuxième temps, nous pouvons étudier la taille des plans calculés. CTCF+ calcule des plans de même taille que CTCF pour l'ensemble des problèmes que CTCF peut résoudre, c'est à dire les problèmes *btcs+* et *rovers+*. En revanche, CTCF+ calcule un plan de plus grande taille que CTCFE pour le problème *rovers+/p6*. CTCF+ calcule des plans plus petits que Contingent-FF, CLG et PO-PRP pour l'ensemble des problèmes *rovers+*, les problèmes *erovers+/p2* et *elogistics+/p3*. CTCF+ calcule des plans plus petits que Contingent-FF et CLG pour les problèmes *elogistics+/p7*. CTCF+ calcule des plans plus petits que Contingent-FF et PO-PRP pour l'ensemble des problèmes *btcs+*, les

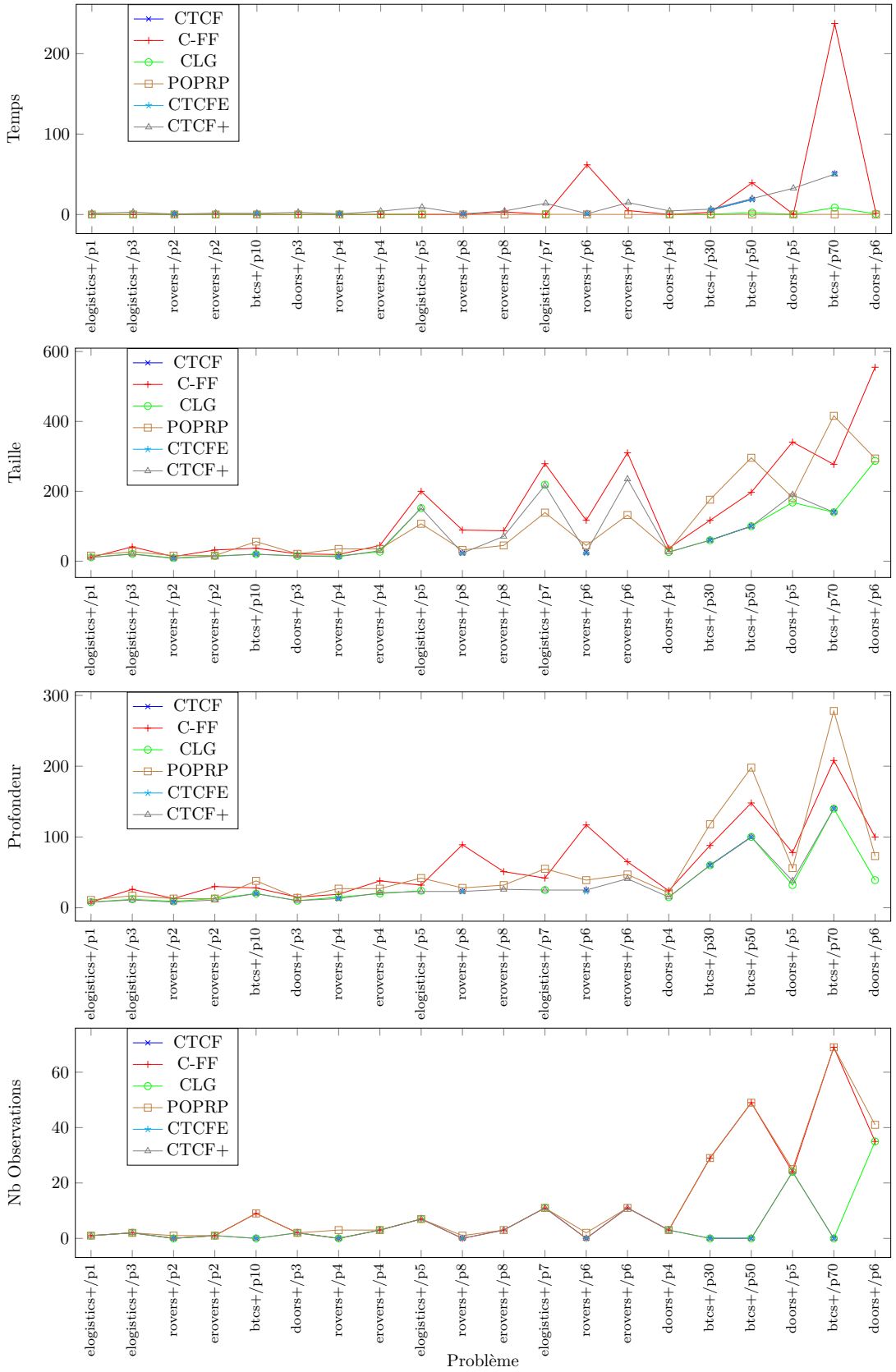

FIGURE 7.4 – Comparaison du temps de calcul, de la taille, de la profondeur, et du nombre d'observations du plan pour les problèmes coûteux

problèmes *erovers+/p4*, *doors+/p3* et *p4*. CTCF+ calcule des plans plus petits que Contingent-FF pour les problèmes *erovers+/p6* et *p8*, *elogistics+/p5*, *doors+/p5*, et plus petits que PO-PRP pour le problème *elogistics/p1*.

Nous pouvons maintenant analyser la profondeur des plans calculés. CTCF+ calcule des plans de même profondeur que CTCF pour l'ensemble des problèmes *btcs+* et *rovers+*. Cependant CTCF+ calcule un plan plus long que CTCFE pour le problème *rovers+/p6*. CTCF+ calcule des plans plus courts que Contingent-FF, CLG et PO-PRP pour l'ensemble des problèmes *rovers+*, les problèmes *erovers+/p2*, *p6* et *p8*, *elogistics/p3*, *p5*. CTCF+ calcule aussi des plans plus courts que Contingent-FF et PO-PRP pour l'ensemble des problèmes *btcs+*, *erovers+/p4*, *elogistics+/p7*, *doors+/p3*, *p4* et *p5*, et des plans plus courts que PO-PRP pour le problème *elogistics/p1*.

Enfin, nous pouvons nous rendre compte que CTCF+ ajoute autant d'observations dans le plan que CTCF et CTCFE pour l'ensemble des problèmes *btcs+* et *rovers+*. CTCF+ ajoute autant d'observations que CLG pour l'ensemble des problèmes que CTCF+ et CLG peuvent résoudre. CTCF+ ajoute moins d'observations que Contingent-FF et PO-PRP pour les problèmes *btcs+*, et moins d'observations que PO-PRP pour les problèmes *rovers+* et *doors+/p5*.

Conclusion des résultats

À partir des résultats expérimentaux, nous pouvons effectuer des conclusions sur l'approche. Dans un premier temps, on peut remarquer que CTCF+ résout 21 problèmes de plus que CTCFE et 22 de plus que CTCF (soit 71 problèmes résolus sur 87 problèmes).

Cette amélioration réside dans le fait que la fonction de sélection des observations candidates permet à présent d'identifier des observations n'étant pas applicables dans le chemin d'exécution du plan défaillant retourné par CPCES. Cela permet à CTCF+ de résoudre la plupart des problèmes coûteux, contrairement à CTCF et CTCFE. Cependant, CTCF+ ne peut toujours pas résoudre les problèmes *wumpus* et *wumpus**, la procédure de sélection des observations ne prenant pas en compte les observations indirectes.

En ce qui concerne le temps de calcul des plans, la plupart des problèmes pour lesquels CTCF+ calcule des plans plus rapidement que CTCF et CTCFE sont des problèmes conformants, révélant uniquement une implémentation légèrement plus efficace. Dans la majorité des cas, CTCF+ a un temps de calcul compris entre les temps de CTCF et CTCFE. Ce temps de calcul souffre des mêmes problèmes que CTCF et CTCFE pour les problèmes de grande taille, pour lesquels CTCF+ dépasse le temps limite. Ce dépassement de temps réside dans la plupart des problèmes au temps de chargement de l'état de croyance initial grandissant en fonction de la taille des problèmes.

CTCF+ calcule généralement des plans de tailles et profondeurs équivalentes à CTCF et CTCFE, et qui sont en général compétitives avec Contingent-FF, CLG et PO-PRP. CTCF+ ajoute généralement autant d'observations que CTCF et CTCFE dans les plans, à l'exception du problème *colorballs*/p21* pour lequel CTCF+ incorpore moins d'observations que CTCFE, révélant que CTCFE incorpore tout de même des observations non nécessaires dans certains problèmes. Dans l'ensemble des problèmes que peut traiter CTCF+, le nombre d'observations incorporées n'est jamais supérieur à Contingent-FF, CLG et PO-PRP, révélant une limitation du nombre d'observations dans le plan aussi efficace que CTCF. Aussi, dans plusieurs problèmes, ce nombre d'observations est plus petit que Contingent-FF, CLG et PO-PRP.

7.5 Discussion

L'amélioration du processus de sélection des observations et de la modélisation du but du problème de calcul du plan menant à l'observation permet à cette nouvelle approche de résoudre bien plus de problèmes que CTCF et CTCFE sans modifier grandement la qualité des plans générés, ni le temps de calcul des plans. De plus, cette version de l'approche n'est pas aussi dépendante de CPCES que CTCFE, les améliorations apportées n'incluant pas de modification de CPCES. Cependant, même si l'augmentation du nombre de problèmes traités est importante, CTCF+ n'est toujours pas capable de résoudre les problèmes dans lesquels les observations sont indirectes. CTCF+ n'implémente pas non plus de retour possible sur le choix des observations en cas d'échec de calcul d'une solution après l'incorporation d'une observation. Au delà du problème de complétude que contient CTCF+, nous pouvons aussi noter que comme CTCF et CTCFE, le temps de calcul d'une solution devient

rapidement un problème lorsque la taille des problèmes grandit, empêchant l'approche de résoudre certains problèmes dans le temps imparti. Une grande partie de ce problème de temps de calcul est lié à la représentation explicite de l'état de croyance devant être manipulé dans la procédure contingente. Afin d'éviter ce coût de modifications internes des états de croyance, ainsi qu'une occupation mémoire importante, il semble nécessaire de changer de représentation des états de croyance. Dans le chapitre suivant, nous allons étudier l'implémentation d'une nouvelle version de l'approche dans laquelle on ne représente plus explicitement les états de croyance, ceux-ci étant plutôt représentés comme l'association de l'état de croyance initial et de la séquence d'actions et d'observations menant de l'état de croyance initial à l'état de croyance courant.

7.6 Conclusion

Dans ce chapitre nous avons présenté une nouvelle version de l'approche nommée CTCF+ dans laquelle le processus de sélection des observations est modifié afin de ne plus se reposer sur les états de croyance calculés par l'application du plan π défaillant retourné par CPCES à l'état de croyance initial $\mathcal{B}_0$. À la place, toutes les observations permettant d'observer les préconditions de l'action échouant dans le contre-exemple sont sélectionnées indépendamment du fait qu'elles soient applicables ou non dans un des états de croyance formés par l'application du plan π à $\mathcal{B}_0$. Le but du problème de calcul du plan menant à l'observation est ensuite modélisé comme une disjonction des préconditions de ces observations, et non plus comme un état de croyance précis. Ces modifications permettent à CTCF+ de résoudre un plus grand nombre de problèmes que CTCF et CTCFE, et plus particulièrement les problèmes dans lesquels les observations ne sont pas forcément applicables dans le chemin d'exécution formé par l'application du plan retourné par CPCES à l'état de croyance initial. De plus, ces modifications n'influent pas grandement sur la taille, la profondeur, et le temps de calcul des plans calculés par rapport à CTCF et CTCFE. Malgré cette amélioration du nombre de problèmes résolus, il semble que CTCF+ souffre toujours d'un problème de temps de calcul trop élevé pour les problèmes de grande taille causé par la représentation explicite des états de croyance. De plus, cette représentation explicite des états de croyance peut entraîner des dépassements mémoire pour des problèmes de grande taille. CTCF+ souffre du même problème de complétude que CTCF et CTCFE, les problèmes avec observations indirectes (les problèmes dont les préconditions de l'action échouant ne peuvent pas être observées directement) ne pouvant toujours pas être résolus, et aucun retour sur le choix des observations n'étant implémenté en cas d'échec de calcul d'un plan après ajout d'une observation.

Dans le chapitre suivant (Chapitre 8), nous allons nous intéresser à une nouvelle approche dans laquelle les états de croyance ne sont plus représentés de manière explicite, mais plutôt comme l'association de l'état de croyance initial et de la séquence d'action et d'observations menant de l'état de croyance initial à l'état de croyance courant.

Enfin, dans le chapitre 9, nous étudierons l'implémentation d'une procédure de sélection d'observation supplémentaire permettant d'identifier les observations indirectes, ainsi qu'une procédure de backtracking sur le choix des observations en cas d'échec de calcul d'un plan incorporant l'observation choisie permettant de rendre l'approche complète.

Chapitre 8

Une représentation compacte des états de croyance pour le passage à l'échelle

8.1 Motivation

En étudiant la complexité des versions précédentes de l'approche, nous avons pu nous rendre compte que le temps de calcul dépend directement du nombre d'états initiaux possibles du problème. Ces résultats sont d'autant plus visibles du fait que les versions précédentes de l'approche représentent les états de croyance de manière explicite pour les communiquer à CPCES. Cette représentation explicite des états de croyance peut entrainer une explosion combinatoire, entrainant une augmentation des temps de calcul du plan et de l'occupation mémoire proportionnellement au nombre d'états initiaux possibles. Dans ce chapitre, nous allons appliquer une approche dans laquelle les états de croyance ne sont plus représentés de manière explicite mais plutôt comme l'association de l'état de croyance initial et de la séquence d'actions et d'observations menant à l'état de croyance courant depuis cet état de croyance initial. Cette représentation plus compacte a pour but de pouvoir traiter des problèmes de plus grande taille, grâce aux gains de temps et de mémoire obtenus en ne manipulant plus les états de croyance en interne.

8.2 Mise en oeuvre

Tout comme CTCFE, cette version du planificateur que l'on nommera CTCFL (ConTingent planner using a ConFormant planner for Large problems) a été développée en grande partie lors de ma mobilité à l'Australian National University de Canberra en collaboration avec Alban Grastien. Le nouveau processus contingent est détaillé dans la section 8.2.1. Afin que le planificateur passe à l'échelle, nous définissons les états de croyance comme l'association de l'état de croyance initial et de la séquence d'actions et d'observations menant à l'état de croyance courant. Cette procédure est détaillée dans la section 8.2.2. Cette représentation nécessite la transformation des plans sous la forme de séquences d'actions et d'observations avant de pouvoir être transmis au planificateur conformant (section 8.2.3). La section 8.2.4 détaille les modifications apportées par Alban Grastien à CPCES afin que celui-ci puisse se servir de la nouvelle représentation des états de croyance. Cette représentation de l'état de croyance implique d'adapter les procédures et les communications avec CPCES afin que celui-ci puisse calculer l'ensemble d'états courant à partir de ces nouvelles informations. Afin de forcer CPCES à démarrer son calcul de plan par la séquence d'actions et d'observations voulue, nous avons décidé d'encoder directement cette séquence dans le problème fourni à CPCES (section 8.2.5). Le processus de sélection des observations est évoqué dans la section 8.2.6. Le but du problème de calcul de plan menant à l'observation est déterminé à partir des observations sélectionnées (section 8.2.7). Les répercussions de la nouvelle représentation des états de croyance sur l'architecture générale de l'approche sont détaillées dans la section 8.2.8. Une analyse théorique de l'approche est effectuée dans la section 8.3, tandis qu'une analyse expérimentale de l'approche est effectuée dans la section 8.4. Finalement, les sections 8.5 et 8.6 de ce chapitre concluront cette approche.

8.2.1 Processus contingent

La représentation explicite de l'état de croyance est un des éléments de CTCF, CTCFE et CTCF+ jouant un rôle majeur dans le temps de calcul d'un plan, la complexité de l'approche dépendant du nombre d'états initiaux possibles. On considère qu'un problème a une grande taille lorsque celui-ci possède soit un grand nombre de propositions dans chaque état, soit un grand nombre d'états dans l'état de croyance. De plus, la représentation explicite de l'état de croyance peut mener à des dépassements mémoire lorsque la taille du problème devient très grande. Afin de contourner ces problèmes, nous ne représentons plus l'état de croyance de manière explicite, mais plutôt de manière implicite. Pour cela, comme dans Contingent-FF (Hoffmann et Brafman, 2005), nous représentons l'état de croyance courant $\mathcal{B}$ d'une manière plus compacte en associant l'état de croyance initial $\mathcal{B}_0$ avec la séquence d'actions et d'observations π menant à $\mathcal{B}$ étant donné que $\mathcal{B} = T(\mathcal{B}_0, \pi)$. Comme nous utilisons un planificateur conformant, qui par définition ne prend pas en compte les observations, nous encodons les informations concernant les résultats des observations de π par le biais d'un ensemble de contraintes φ. Les détails sur la constitution de ces contraintes se situent dans la section 8.2.2 de ce chapitre.

Le processus contingent de CTCFL doit s'adapter à la nouvelle représentation des états de croyance. L'algorithme 10 illustre ce nouveau processus contingent.

Algorithm 10 ContingentPlanning Procedure

Require: $\mathcal{P}_\Omega = (\Sigma, \mathcal{B}_0, g), \pi_i, \varphi_i$
Ensure: π_c
1: $\mathcal{P} = (S, A, T, \mathcal{B}_0, g)$
2: $\pi, \gamma :=$ CPCES*$(\mathcal{P}, \pi_i, \varphi_i)$
3: **if** $\gamma = \emptyset$ **then**
4: **return** π
5: $\Omega_{\pi,\gamma} :=$ findObservation+$(\mathcal{P}_\Omega, \pi, \gamma)$
6: let g_σ be an artificial proposition for each $\sigma \in \Omega_{\pi,\gamma}$
7: $A_\Omega := \{(Pre(\sigma), g_\sigma, \emptyset) | \sigma \in \Omega_{\pi,\gamma}\}$
8: let g_Ω be the disjunction of each g_σ
9: $\mathcal{P}'_\Omega = (S, A \cup A_\Omega, \Omega, T, \mathcal{B}_0, g_\Omega)$
10: $\pi_\sigma :=$ ContingentPlanning$(\mathcal{P}'_\Omega, \pi_i, \varphi_i)$
11: $paths :=$ Separate(π_σ)
12: **for** each couple π, φ_π of $paths$ **do**
13: $\varphi_+ := \varphi_\pi \oplus (v(p) == \top)$
14: $\pi_+ :=$ ContingentPlanning$(\mathcal{P}_\Omega, \pi, \varphi_+)$
15: $\varphi_- := \varphi_\pi \oplus (v(p) == \bot)$
16: $\pi_- :=$ ContingentPlanning$(\mathcal{P}_\Omega, \pi, \varphi_-)$
17: $\pi := (\pi$;**if** σ **then** π_+ **else** $\pi_-)$
18: **return** π_σ

Premièrement, par rapport à CTCF et CTCF+, la procédure contingente de CTCFL prend en entrée deux nouveaux paramètres : l'ensemble de contraintes ϕ et un plan initial π_i, tous deux vides lors de la première exécution de la procédure. La nouvelle version de CPCES que nous nommons CPCES* est appelée à partir du problème conformant $\mathcal{P}$, de la séquence d'actions et d'observations initiale π_i, et de l'ensemble de contraintes φ_i. Tout comme les versions précédentes de l'approche, CPCES* retourne soit un plan conformant, soit un plan valide pour une partie des états initiaux seulement et un état de contre-exemple γ (ligne 2).

À la ligne 5 de l'algorithme 10, la procédure findObservation+, qui est la même que le chapitre précédent (section 7.2.1 du chapitre 7), retourne un ensemble d'observations $\Omega_{\pi,\gamma}$ permettant de discriminer le contre-exemple γ.

Afin de calculer un plan menant de l'état de croyance $\mathcal{B}_0$ à une des observations σ de $\Omega_{\pi,\gamma}$, on crée une proposition artificielle g_σ correspondant à chaque observation σ de $\Omega_{\pi,\gamma}$ (ligne 6), puis on crée un ensemble d'actions artificielles $A_\Omega = \{(Pre(\sigma), g_\sigma, \emptyset) | \sigma \in \Omega_{\pi,\gamma}\}$ dans laquelle chaque action artificielle a_σ comporte les mêmes préconditions que l'observation σ et a pour unique effet la proposition g_σ correspondante (ligne 7). On définit l'ensemble de propositions buts g_Ω comme la disjonction de l'ensemble des propositions artificielles g_σ (ligne 8). Ceci est traduit dans un nouveau problème contingent $\mathcal{P}'_\Omega = (S, A \cup A_\Omega, \Omega, \mathcal{B}_0, g_\Omega)$ qui, à partir du problème original $\mathcal{P}$, intègre l'ensemble d'actions artificielles A_Ω et le but g_Ω (ligne 9). Enfin, la procédure ContingentPlanning est appelée à partir de $\mathcal{P}'_\Omega$, π_i et φ_i afin de calculer un plan π_σ dont les branches mènent à l'une des propositions g_σ et donc par substitution, à l'observation σ correspondante (ligne 10).

Chaque branche du plan π_σ ne termine pas forcément au même état de croyance, ni à la même observation. Il faut donc calculer un nouveau plan menant au but du problème pour chaque branche de π_σ. Le plan π_σ est découpé afin de récupérer chaque branche π du plan et les contraintes φ_π associées grâce à la procédure Separate (ligne 11), qui est expliquée dans la section 8.2.3.

Pour chaque branche π, on définit les ensembles de contraintes φ_+ et φ_- comme l'ensemble de contraintes φ_π auquel on ajoute la contrainte représentant le fait que la valeur observée par l'observation σ située à la fin de π est respectivement vraie ou fausse (lignes 13 et 15). Le détail de cette opération est expliquée dans la section 8.2.2.

Le processus contingent est ensuite appelé de nouveau pour calculer la branche positive π_+ et la branche négative π_- suivant la dernière observation de π et menant au but du problème. Cette procédure prend comme entrée la branche courante π comme plan initial π_i, accompagnée de l'ensemble de contrainte correspondant φ_+ ou φ_- selon que l'on calcule la branche positive ou négative, ainsi que du problème contingent initial $\mathcal{P}_\Omega$ (lignes 14 et 16). Dès que ces branches sont calculées, on connecte

ces branches à l'observation σ à la fin de la branche courante π de π_σ (ligne 17) et on passe à la branche suivante de π_σ ne menant pas encore au but. Une fois que toutes les branches mènent au but du problème, alors on retourne le plan contingent π_σ calculé (ligne 18).

8.2.2 Représentation implicite de l'état de croyance

Afin de forcer CPCES* à démarrer le plan qu'il doit calculer par la branche courante π_i du plan menant à l'observation, il est nécessaire d'introduire dans le problème un ensemble d'actions artificielles A_{π_i} correspondant aux actions et observations présentes dans cette branche, et défini tel que :

$$A_{\pi_i} = \big\{\big(Pre(a_k) \cup \{c_k\}, Add(a_k) \cup \{c_{k+1}\}, Del(a_k)\big) |\ a_k \in \pi_i\big\} \tag{8.1}$$

Les propositions compteurs c_k et c_{k+1} sont introduites aux préconditions et aux effets de chaque nouvelle action pour forcer la séquence d'action à être exécutée dans l'ordre précis des actions et observations de π_i. On peut noter que les observations sont transformées de la même manière que les actions, à l'exception de leur effet d'observation, qui est ignoré. On introduit de la même manière une proposition compteur $c_{|\pi_i|+1}$ aux préconditions de chaque action de A pour s'assurer que ces actions seront planifiées après les actions de π_i, obtenant l'ensemble d'actions A' tel que :

$$A' = \{(Pre(a_j) \cup \{c_{|\pi_i|+1}\}, Add(a_j), Del(a_j)) | a_j \in A\} \tag{8.2}$$

Le problème conformant fourni à CPCES (ligne 2 de l'algorithme 10) est donc finalement $\mathcal{P} = \{S, A' \cup A_{\pi_i}, T, \mathcal{B}_0, g\}$.

Afin de représenter les contraintes φ_i encodant les résultats des observations de π_i, nous nous sommes adaptés à la manière dont le planificateur conformant pouvait considérer ces contraintes. Étant donné que CPCES raisonne à partir de contraintes logiques sous la forme de formules SMT, nous avons par conséquent décidé d'encoder les contraintes sur les valeurs des propositions observées par le biais de formules SMT. Si on considère une séquence d'actions et d'observations π_i, alors l'ensemble de contraintes d'assertions SMT φ_i de π_i est $\varphi_i = \bigcup_{\sigma_j \in \pi_i}((assert\ (=\ p\text{-}j\ \nu(p)))$ avec $\nu(p)$ la valeur de vérité de la proposition p observée par σ, et j l'étape du plan π_i à laquelle se trouve l'observation σ. La contrainte d'assertion $(assert\ (=\ p\text{-}j\ \nu(p))$ représente le fait que la proposition p observée par l'observation σ a nécessairement la valeur $\nu(p)$ à l'étape j du plan π_i.

L'opérateur $\oplus$ des lignes 13 et 15 de l'algorithme 10 peut désormais être décrit comme l'ajout d'une contrainte SMT représentant la valeur de vérité de la proposition observée par σ selon que l'on calcule une branche positive ou négative à l'ensemble de contraintes SMT φ_i représentant les résultats des observations de π_i. On a donc :

$$\varphi_i \oplus (\nu(p) == \top) \quad = \quad \varphi_i \ \wedge\ (assert(=\ p\text{-}|\pi_i|\ true)) \tag{8.3}$$

$$\varphi_i \oplus (\nu(p) == \bot) \quad = \quad \varphi_i \ \wedge\ (assert(=\ p\text{-}|\pi_i|\ false)) \tag{8.4}$$

La reconstitution de l'état de croyance courant est laissée aux soins de CPCES qui utilise le SAT-Solver Z3 sur les formules SMT, l'état de croyance initial et la séquence d'actions et d'observations de π_i.

8.2.3 Séparation d'un plan en séquences d'actions et contraintes d'observations

Afin d'illustrer la séparation d'un plan en séquences d'actions et en contraintes d'observations, nous nous concentrons sur la séparation d'un plan π_σ menant à une observation. Dans ce cas, il y a deux possibilités : soit π_σ est conformant, soit il est contingent. Les différentes branches de ce plan π_σ doivent êtres transmises une à une à CPCES sous forme de séquences d'actions afin que celui-ci puisse déterminer par quel plan démarrer son calcul de plan menant au but du problème initial à partir de l'observation.

Lorsque π_σ est conformant, nous avons seulement besoin de fournir la contrainte d'observation φ de l'observation finale σ de π_σ étant donné que le reste du plan est déja une séquence d'actions unique. Prenons l'exemple de plan contingent de la figure 8.1.

Exemple: Plan contingent

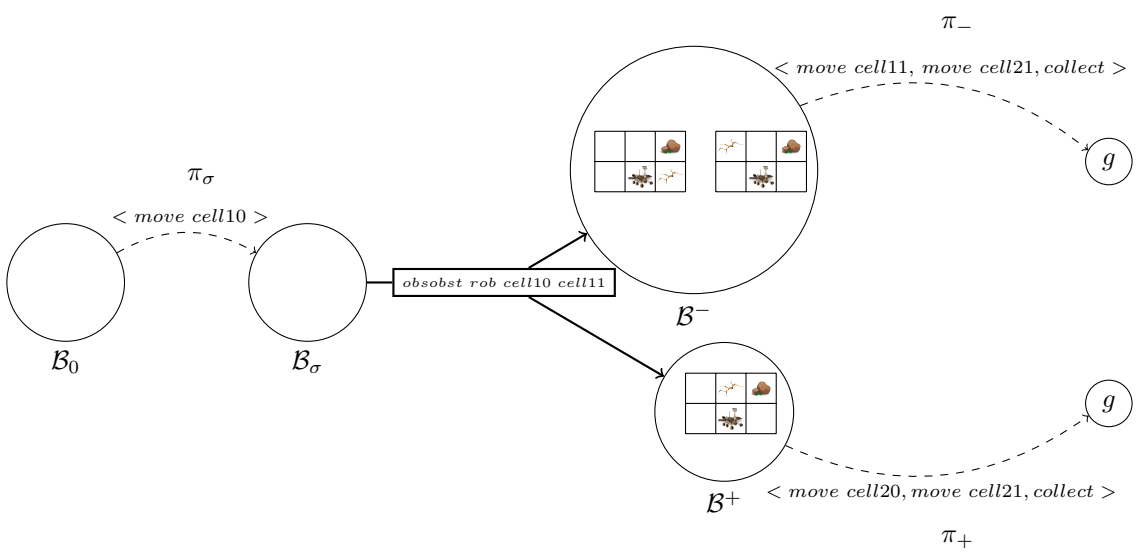

FIGURE 8.1 – Plan contingent

Afin de calculer les branches π_- et π_+ du plan décrit dans la figure 8.1, CTCFL va transmettre le plan $\pi_\sigma =< move\ cell10, obsobst\ rob\ cell10\ cell11 >$ qui sera directement transmis à CPCES ainsi qu'une contrainte φ indiquant si l'on se trouve dans le calcul de la branche considérant que l'observation $< obsobst\ rob\ cell10\ cell11 >$ est positive ou négative.

Lorsque π_σ est contingent, nous devons séparer le plan en plusieurs séquences d'actions correspondant aux différents chemins possibles dans le plan, c'est à dire aux différentes séquences d'actions et d'observations démarrant par la première action du plan et terminant à la fin des branches de π_σ.

L'algorithme 11 décrit ce processus de séparation.

Algorithm 11 SEPARATE Algorithm

Require: π_σ
Ensure: *paths*
1: let a be the first action of π_σ
2: $paths := []$
3: **if** a is at the end of π_σ **then**
4: **return** $[< a >, \{\}]$
5: **if** a is an observation **then**
6: let π_+ be the positive branch following a
7: $\varphi_+ := \{\nu(p) == \top\}$
8: $\varphi_- := \{\nu(p) == \bot\}$
9: let π_- be the negative branch following a
10: $paths := paths + [< a >, \{\varphi_+\} + \text{SEPARATE}(\pi_+)]$
11: $paths := paths + [< a >, \{\varphi_-\} + \text{SEPARATE}(\pi_-)]$
12: **else**
13: let π be the branch following a
14: $paths := paths + [< a >, \{\} + \text{SEPARATE}(\pi)]$
15: **return** *paths*

L'algorithme 11 prend en entrée un plan π_σ et retourne la liste des différents chemins π du plan auxquels sont associés l'ensemble des contraintes φ_π correspondantes aux résultats des observations présentes dans chaque chemin π. Dans un premier temps, on considère l'action ou l'observation courante a comme la première action du plan π_σ, et on initialise la liste de chemins et de contraintes *paths* comme une liste vide (lignes 1 et 2). Si a se situe à la fin de la branche courante du plan π_σ (ligne 3),

alors il est certain que celle-ci est une observation, le plan fourni à CPCES terminant systématiquement par une observation. Cette observation est retournée accompagnée d'une contrainte φ vide (ligne 4), qui sera renseignée plus tard dans le processus contingent pour indiquer quelle branche suivant l'observation σ nous voulons calculer (lignes 13 et 15 de l'algorithme 10).

Lorsque a n'est pas à la fin de la branche courante, deux cas de figure existent : soit a est une action, soit c'est une observation. Si a est une observation (lignes 5 à 11), alors les contraintes φ_+ et φ_- représentant les résultats positifs et négatifs de cette observation sont créées (lignes 7 et 8). Afin de générer les chemins et les contraintes correspondant à la branche positive π_+ suivant l'observation, la procédure SEPARATE est appelée à partir de π_+. On ajoute à la liste de chemins *paths* l'observation a, la contrainte φ_+, et la liste de chemins et de contraintes retournés par cet appel de la procédure SEPARATE (ligne 10). La même procédure est utilisée pour générer la liste de chemins et contraintes de la branche négative π_- (ligne 11). Si a est une action (lignes 13 et 14), alors la procédure SEPARATE est appelée à partir de la branche π suivant a afin de calculer les chemins et contraintes de π. Ces chemins et contraintes, l'action a, et un ensemble de contraintes vide sont ajoutés à la liste de chemins et contraintes *paths* (ligne 14).

Finalement, la liste de chemins et contraintes *paths* est retournée (ligne 15).

L'exemple ci-dessous illustre le découpage d'un plan contingent en séquences d'actions et d'observations à transmettre à CPCES accompagnées des contraintes SMT correspondantes.

Exemple: Séparation d'un plan contingent en séquences d'actions

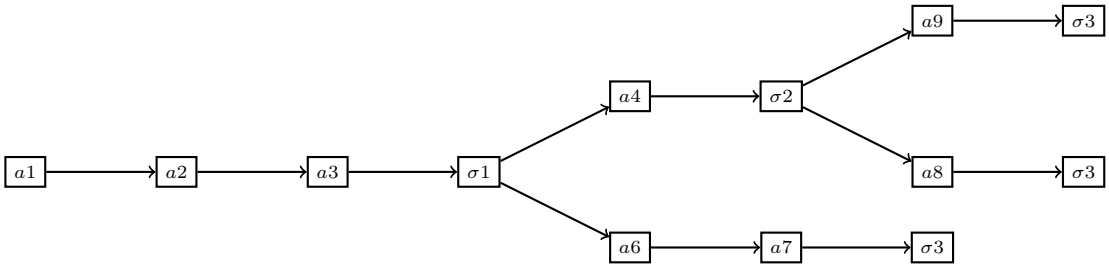

FIGURE 8.2 – Exemple de plan contingent menant à une observation

Prenons l'exemple du plan contingent décrit dans la figure 8.2 et menant à l'observation $\sigma 3$. Une fois que ce plan a été calculé, il reste à calculer des branches menant de chaque observation $\sigma 3$ au but du problème. Pour cela, chaque chemin du plan menant à l'observation $\sigma 3$ doit être transmis à CPCES accompagné d'un ensemble de contraintes SMT précisant le résultat de chaque observation du chemin à CPCES. On peut découper le plan de la figure 8.2 en trois chemins. Le premier chemin est $< a1, a2, a3, \sigma 1, a4, \sigma 2, a9, \sigma 3 >$. Les contraintes SMT ($assert$ ($= \sigma 1\_prop\text{-}3\ true$)) et ($assert$ ($= \sigma 2\_prop\text{-}5\ true$)) sont transmises avec ce chemin à CPCES pour lui indiquer que les résultats des observations $\sigma 1$ et $\sigma 2$ sont tous les deux positifs, ainsi qu'une contrainte ($assert$ ($= \sigma 3\_prop\text{-}7\ true$)) ou une contrainte ($assert$ ($= \sigma 3\_prop\text{-}7\ false$)) selon que l'on souhaite calculer la branche positive ou négative après $\sigma 3$. Le deuxième chemin est $< a1, a2, a3, \sigma 1, a4, \sigma 2, a8, \sigma 3 >$. Les contraintes SMT associées sont ($assert$ ($= \sigma 1\_prop\text{-}3\ true$)) et ($assert$ ($= \sigma 2\_prop\text{-}5\ false$)) indiquant que le résultat de $\sigma 1$ est positif et le résultat de $\sigma 2$ est négatif, ainsi qu'une contrainte ($assert$ ($= \sigma 3\_prop\text{-}7\ true$)) ou une contrainte ($assert$ ($= \sigma 3\_prop\text{-}7\ false$)) selon que l'on souhaite calculer la branche positive ou négative après $\sigma 3$. Enfin, le dernier chemin est $< a1, a2, a3, \sigma 1, a6, a7, \sigma 3 >$. La contrainte SMT associée est ($assert$ ($= \sigma 1\_prop\text{-}3\ false$)) indiquant que le résultat de l'observation $\sigma 1$ est négatif, ainsi qu'une contrainte ($assert$ ($= \sigma 3\_prop\text{-}7\ true$)) ou une contrainte ($assert$ ($= \sigma 3\_prop\text{-}7\ false$)) selon que l'on souhaite calculer la branche positive ou négative après $\sigma 3$.

8.2.4 Modification de CPCES

La modification de CPCES a été entièrement réalisée par Alban Grastien. Cette modification consiste à permettre à CPCES* de prendre en entrée un plan initial π_i ainsi qu'un ensemble de contraintes

SMT φ_i. Le plan π_i en entrée est une séquence d'actions et d'observations par lequel le plan solution retourné par CPCES* devra démarrer. L'ensemble de contraintes SMT permet à CPCES* de connaître les résultats de toutes les observations présentes dans π_i afin de les transmettre au SAT-Solver Z3, permettant la vérification de la validité du plan π_i. Si l'appel à CPCES* est le premier du processus contingent, alors π_i est un plan vide et l'ensemble de contraintes SMT est vide lui aussi. L'algorithme 12 représente cette nouvelle version CPCES*.

Algorithm 12 CPCES* Algorithm

Require: $\mathcal{P} = (\Sigma, \mathcal{B}_0, g)$,$\pi_i$,$\varphi_i$
Ensure: π, γ
1: $\mathcal{B} := \emptyset$
2: $\pi := \varepsilon$
3: **loop**
4: **if** $< \pi_i, \pi >$ is a solution for $\mathcal{P}$ considering φ_i **then**
5: **return** $< \pi_i, \pi >, \emptyset$
6: let γ be a counter-example
7: $\mathcal{B} := \mathcal{B} \cup \{\gamma\}$
8: compute a new plan $< \pi_i, \pi' >$ for $\mathcal{P}' = (\Sigma, \mathcal{B}, g)$ considering φ_i
9: **if** no such $< \pi_i, \pi' >$ exists **then**
10: **return** $< \pi_i, \pi >, \gamma$
11: $\pi := \pi'$

On peut remarquer dans la ligne 4 de l'algorithme 12 que ce n'est plus le nouveau plan π calculé par CPCES* seulement qui est vérifié en tant que solution mais bien le plan composé de la séquence $< \pi_i, \pi >$ en considérant les différentes contraintes de φ_i assurant la valeur des propositions observées dans π_i. Si le plan $< \pi_i, \pi >$ est solution alors il est retourné, sinon CPCES* trouve un contre-exemple et l'ajoute à l'état de croyance considéré $\mathcal{B}$. À partir de ce nouvel état de croyance CPCES* tente de calculer un nouveau plan $< \pi_i, \pi' >$, toujours en considérant les contraintes de φ_i (ligne 8). En pratique, afin que CPCES* démarre par le plan initial lors de son calcul de plan, il est nécessaire de le "forcer" en modifiant directement le fichier de domaine et problème PDDL qu'il prend en entrée pour imposer la séquence d'actions présente dans π_i (détaillé dans les sections 8.2.2 et 8.2.5). Si CPCES* échoue à calculer un nouveau plan, alors celui-ci retourne le plan $< \pi_i, \pi >$ précédemment calculé à l'étape précédente ainsi que le contre-exemple de ce plan γ, sinon le plan $< \pi_i, \pi' >$ est vérifié et le processus continue tant qu'une solution n'est pas trouvée.

8.2.5 Modification du domaine PDDL pour le calcul des branches menant au but du problème à partir de l'observation

Afin de forcer CPCES* à commencer son calcul de plan par le plan initial π_i, nous introduisons un ensemble d'actions artificielles représentant les actions de π_i et comportant des propositions compteurs dans leurs préconditions et leurs effets (détaillé dans la section 8.2.2). Le problème de planification conformante est fourni en entrée de CPCES sous la forme d'un fichier domaine et d'un fichier problème PDDL, par conséquent, nous devons introduire ces actions artificielles et ces compteurs directement dans le fichier domaine PDDL.

Exemple: Codage utilisant des actions artificielles représentant les actions d'un plan

Reprenons l'exemple de π_σ de la figure 5.4 du chapitre 5. CTCFL restitue un plan $\pi_\sigma =< move\ rob\ cell00\ cell10, obsobst\ rob\ cell10\ cell11 >$. Par conséquent, si l'on veut que CPCES démarre son calcul de branche par la séquence $< move\ rob\ cell00\ cell10, obsobst\ rob\ cell10\ cell11 >$, alors les actions artificielles $< art\_move\_rob\_cell00\_cell10 >$ et $< art\_obsobst\_rob\_cell10\_cell11 >$ sont introduites de la manière suivante dans le domaine PDDL :

```
(:constants rob - robot
      cell00 cell10 cell11 - position
)
(:predicates
(adjacent ?x ?y)
...
(compteur1)
(compteur2)

)

(:action art_move_rob_cell00_cell10
:precondition (and (adjacent cell00 cell10) (not(is_obstacle cell00)) (
    at_robot rob cell00))
:effect (and (not (at_robot rob cell00)) (at_robot rob cell10) (
    compteur1))
)

(:action art_obs_obst_rob_cell10_cell11
 :precondition (and (at_robot rob cell10) (adjacent cell10 cell11) (
    compteur1))
 :effect (compteur2))
)

(:action move
:parameters (?x - robot ?y - position ?z - position)
:precondition (and (adjacent ?y ?z) (not(is_obstacle ?z)) (at_robot ?x ?
    y) (compteur2))
:effect (and (not (at_robot ?x ?y)) (at_robot ?x ?z)))
)

...
```

FIGURE 8.3 – Modification du domaine PDDL

On peut remarquer que la proposition compteur ($compteur1$) est introduite dans les préconditions et les effets des actions artificielles assurant que $< art\_move\_rob\_cell00\_cell10 >$ soit exécutée avant $< art\_obs\_obst\_rob\_cell10\_cell11 >$. La proposition ($compteur2$) est insérée dans les effets de la dernière action de la séquence d'action artificielle $< art\_obs\_obst\_rob\_cell10\_cell11 >$ et dans les préconditions de l'ensemble des autres actions $a \in A$ permettant de s'assurer que la séquence d'actions artificielles soit entièrement exécutée avant de réaliser d'autres actions. L'exemple de l'action $move \in A$ est donné dans la figure 8.3.

8.2.6 Processus de sélection des observations

La procédure de recherche d'observations est similaire à celle du processus de sélection d'observations FINDOBSERVATION+ de CTCF+. Ce processus est illustré par l'algorithme 8. Dans un premier temps, on applique les actions du plan π à l'état de contre-exemple pour récupérer les préconditions de l'action qui n'est pas applicable. Dans un deuxième temps, on sélectionne l'ensemble des observations de Ω permettant d'observer les préconditions de l'action échouant, et on les retourne.

8.2.7 Représentation du but du problème de calcul de plan menant à une observation

Dans la section 8.2.1, nous avons vu que nous créons des actions artificielles A_Ω correspondant aux observations retournées par la procédure FINDOBSERVATION+. Ces actions artificielles ont comme seul

effet une proposition artificielle g_σ ajoutée au but du problème de calcul de plan menant à l'observation et n'ayant pas d'effet sur les propositions déja présentes dans l'environnement. Nous avons donc un ensemble d'actions artificielles $A_\Omega = \{(Pre(\sigma), g_\sigma, \emptyset) | \sigma \in \Omega_{\pi,\gamma}\}$. Cette création d'actions artificielles permet de ne plus considérer le but comme une disjonction de préconditions pouvant être longue, mais plutôt comme une disjonction de propositions uniques présentes dans le but des actions artificielles, réduisant la taille de la formule représentant le but, donnant $g_\Omega = \{g_{\sigma 1} \vee g_{\sigma 2} ... \vee g_{\sigma w}\}$, avec w le nombre d'observations de $\Omega_{\pi,\gamma}$.

Afin de préciser à CPCES* que l'on veut réaliser une des observations de $\Omega_{\pi,\gamma}$, on modifie dans un premier temps le domaine PDDL afin d'ajouter les actions artificielles correspondant aux observations candidates, ainsi que des constantes correspondant aux paramètres des observations. Une fois que ces actions ont été introduites dans le domaine, on modifie le but du problème afin que celui-ci soit une disjonction des différentes propositions présentes dans les effets des actions artificielles. De cette manière, on s'assure qu'une des actions artificielles termine le plan calculé par CPCES*. Il nous suffit ensuite de faire la correspondance entre cette action artificielle et l'observation correspondante pour obtenir un plan menant à celle-ci. Afin de réduire encore la taille de la formule de but g_σ, nous profitons du langage PDDL en fournissant à l'implémentation PDDL des actions artificielles contenant seulement les paramètres communs des observations devant être réalisées. Ces paramètres correspondent à ceux utilisés par la proposition à observer, laissant à CPCES* le soin de déterminer les préconditions complètes des observations.

Exemple: Modification de domaine et problème PDDL avec des actions artificielles

En reprenant l'exemple de différentes observations possibles de la figure 5.12 du chapitre 5, on a $\Omega_{\pi,\gamma} = \{< obs\_obst\ rob\ cell01\ cell11 >, < obs\_obst\ rob\ cell21\ cell11 >, < obs\_obst\ rob\ cell10\ cell11 >\}$. La figure 8.4 illustre les modifications apportées au domaine pour demander à CPCES* de calculer un plan menant à une de ces observations.

```
(:constants cell11)

(:predicates
(adjacent ?x ?y)
...
(goalobs1)
)

(:action art_obs_obst_cell11
 :parameters (?x - robot ?t - position)
 :precondition (and (at_robot ?x ?t) (adjacent ?t cell11))
 :effect (goal_obs1))
```

FIGURE 8.4 – Modifications du domaine PDDL pour le calcul de plan menant à une observation

Dans la figure 8.4, on peut distinguer l'action artificielle *art_obs_obst_cell11* ajoutée au domaine et représentant les observations $< obsobst\ rob\ cell10\ cell11 >, < obsobst\ rob\ cell21\ cell11 >, < obsobst\ rob\ cell01\ cell11 >$. On peut remarquer que les paramètres abstraits des actions artificielles contenus dans la proposition à observer sont supprimés (ici seulement *cell11*) et des constantes sont introduites pour forcer les paramètres des actions artificielles à correspondre à ceux des observations candidates. La proposition but *goal_obs1* est ajoutée aux prédicats et aux effets de ces actions artificielles afin de définir le but du problème g_Ω comme la disjonction de ces propositions (figure 8.5), avec ici une seule proposition but (*goal_obs1*) correspondant à l'effet de l'unique action artificielle.

```
(:goal
(goal_obs1)
)
```

FIGURE 8.5 – Modification du problème PDDL pour le calcul de plan menant à une observation

8.2.8 Modification de l'architecture

La figure 8.6 illustre les modifications apportées à l'architecture de CTCF+ par la version CTCFL. On peut noter que dans CTCFL, il y a autant d'éléments échangés entre la fonction de recherche d'observation et le processus contingent, étant donné que la fonction de recherche d'observations FINDOBSERVATION+ est la même que la version CTCF+. En revanche, la procédure contingente transmet deux éléments supplémentaires à CPCES* : un plan initial π_i par lequel doit démarrer le calcul de plan de CPCES*, et un ensemble de contraintes φ_i précisant les résultats des différentes observations présentes dans le plan initial π_i. Ces deux éléments sont aussi ajoutés aux paramètres de la fonction contingente permettant leur transmission lors des appels récursifs de calcul de branches.

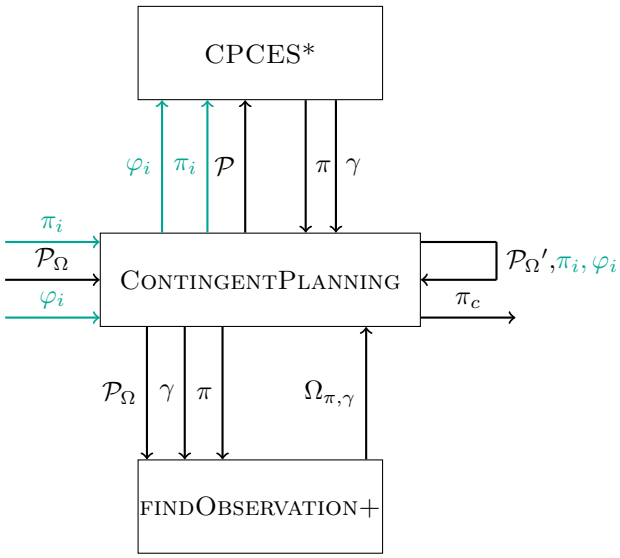

FIGURE 8.6 – Architecture du processus contingent

8.3 Analyse théorique

8.3.1 Complétude

Même si l'approche est maintenant capable de résoudre des problèmes de plus grande taille dans le temps imparti, cette amélioration ne change en rien la complétude de l'approche. L'approche n'est toujours pas capable de résoudre les problèmes dans lesquels les observations sont indirectes. De plus, il n'y a toujours aucune garantie que l'observation choisie nous permette de calculer un plan menant au but du problème initial après l'avoir réalisée. Pour cela, une procédure de retour en arrière sur le choix des observations doit être rajoutée à l'approche.

8.3.2 Complexité

Dans cette section, nous allons étudier la complexité de CTCFL en analysant les algorithmes présentés dans ce chapitre. Nous simplifions les calculs en considérant l'application d'une action ou la vérification et la sélection d'une observation comme une opération unitaire, et nous sur-approximons les équations en ne gardant que les variables de plus grand ordre de grandeur.

L'algorithme findObservation+ (algorithme 8) utilisé par CTCFL pour sélectionner un ensemble d'observations candidates est le même que CTCF+. Par conséquent, la complexité de findObservation+ est la même que le chapitre précédent en notation de Landau : $\mathcal{O}(|\pi| + |F|)$.

Intéressons-nous maintenant à la complexité de la procédure contingente (algorithme 10). On note $\mathcal{C}(n)$ la complexité de CPCES prenant n états en entrée, $\mathcal{F}(n)$ la complexité de findObservation+ prenant n états en entrée, $\mathcal{S}(n)$ la complexité de la procédure Separate prenant en compte n états, et $\mathcal{T}(n)$ la complexité de la procédure contingente prenant en entrée n états. On a :

$$\mathcal{T}(n) = \mathcal{C}(n) + \mathcal{F}(n) + \mathcal{T}'(n) + \mathcal{S}(n) + k \times 2\mathcal{T}(n/2) \tag{8.5}$$

La complexité de la procédure Separate (algorithme 11) ne dépend pas du nombre d'états n mais est plutôt fonction de la taille $|\pi|$ du plan à séparer. Cette procédure parcourt le plan π une fois seulement donnant une complexité en $\mathcal{O}(|\pi|)$. Le terme k correspond au nombre de branches du plan π_σ explorés par la boucle de la ligne 12 de l'algorithme 10. Enfin, $2\mathcal{T}(n/2)$ correspond à la complexité des appels de la procédure contingente pour le calcul des deux sous-plans menant au but du problème depuis l'observation. On a donc une complexité approximative $\mathcal{T}(n)$ de :

$$\mathcal{T}(n) = \mathcal{C}(n) + \mathcal{O}(|\pi| + |F|) + \mathcal{T}'(n) + \mathcal{O}(|\pi|) + k \times 2\mathcal{T}(n/2) \tag{8.6}$$

Finalement, on a $\mathcal{O}(\mathcal{T}(n)) = \mathcal{O}(max(\mathcal{C}(n), \mathcal{O}(|\pi| + |F|), \mathcal{T}'(n), \mathcal{O}(|\pi|), k \times \mathcal{O}(log_2(n)) = \mathcal{O}(\mathcal{C}(n))$.

On a donc $\mathcal{O}(\mathcal{T}(n)) = \mathcal{O}(\mathcal{C}(n))$. Dans le pire cas, CTCFL a la même complexité que CTCF+, c'est à dire $\mathcal{O}(2^{|F|})$. Cependant, on peut déterminer que la complexité de CTCFL peut être meilleure que les versions précédentes de l'algorithme lorsque la taille du problème augmente, le nombre de manipulations des états de croyance étant réduit, ce qui fait disparaître des éléments d'une complexité proportionnelle dans le pire cas à $2^{|F|}$ opérations.

8.4 Expérimentations

Les expérimentations de l'implémentation de CTCFL se sont déroulées dans les mêmes conditions que pour CTCF, CTCFE et CTCF+. Tout comme les précédentes approches, une limite de temps de 10 minutes maximum est imposée lors de la résolution des problèmes. Nous avons décidé de comparer cette nouvelle approche avec CTCF, CTCFE, CTCF+, Contingent-FF, CLG et PO-PRP sur les mêmes benchmarks : des problèmes de la littérature, des problèmes dont la largeur contingente est supérieure à 1, et des problèmes *coûteux*. L'intégralité de ces benchmarks sont décrits en détail en annexe de ce manuscrit.

8.4.1 Résultats

Dans un premier temps, nous allons étudier les résultats de l'exécution des différentes versions de l'approche, de Contingent-FF, de CLG, et de PO-PRP sur les benchmarks de la littérature considérés. Dans un deuxième temps, nous étudierons les résultats sur les problèmes dont la largeur contingente est supérieure à 1. Finalement, nous étudierons les résultats sur les problèmes *coûteux*. Les résultats des problèmes de la littérature sont illustrés dans la figure 8.7 et les résultats des problèmes *coûteux* sont illustrés par la figure 8.8. Tout comme les graphiques de résultats des chapitres précédents, ces deux figures sont découpées en quatre graphiques représentant pour chaque problème le temps de calcul du plan en secondes, la taille du plan en nombre total d'actions et d'observations, la profondeur du plan (taille de la plus longue séquence d'actions et d'observations), et enfin le nombre total d'observations du plan. Dans ces graphiques, les problèmes sont triés de manière croissante selon un critère de difficulté choisi comme le produit du nombre de propositions par état et du nombre d'états composant l'état de croyance initial.

Problèmes de la littérature

Les résultats de l'implémentation de CTCFL sur les problèmes de la littérature sont illustrés par la figure 8.7 et détaillés dans les tableaux B.1 et B.2 situés en annexe de ce manuscrit.

Dans un premier temps, en analysant le tableau B.2 et la figure 8.7, nous pouvons remarquer que CTCFL résout deux problèmes de plus que CTCF+, 11 de plus que CTCFE, et 9 de plus que CTCF

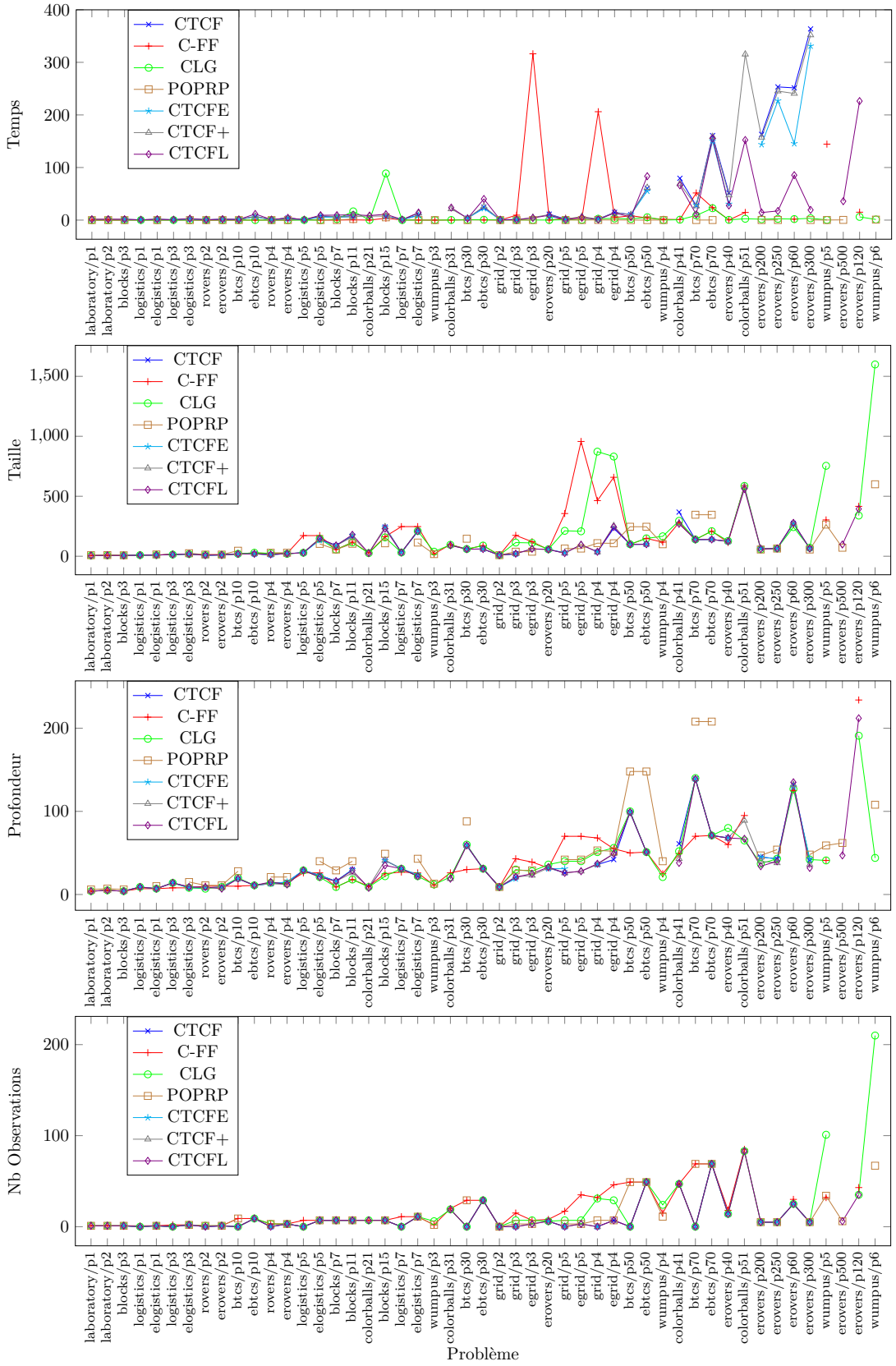

FIGURE 8.7 – Comparaison du temps de calcul, de la taille, de la profondeur, et du nombre d'observations du plan pour les problèmes de la littérature

dans le temps imparti. CTCFL résout plus de problèmes que Contingent-FF et PO-PRP mais ne parvient pas à résoudre les problèmes *wumpus*.

Dans le premier graphique de la figure 8.7 et le tableau B.2, nous pouvons remarquer que CTCFL calcule des plans plus rapidement que CTCF+, CTCFE et CTCF pour 20 problèmes : l'ensemble des problèmes *btcs*, *grid*, *egrid*, *erovers/p40*, *p60*, *p120*, *p200*, *p250*, *p300* et *p500*, *colorballs/p41* et *p51*. Parmi ces problèmes, CTCFL est plus rapide que Contingent-FF pour les problèmes *btcs/p50*, *p70*, *grid/p3*, *p5* et *egrid/p3*. CTCFL est aussi plus rapide que Contingent-FF et CLG pour le problème *grid/p4*. En revanche, CTCFL est plus lent que Contingent-FF, CLG et PO-PRP pour les autres problèmes que ces planificateurs peuvent résoudre. CTCFL calcule des plans plus lentement que CTCF+ pour les problèmes *rovers*, *logistics/p3* et *p7*, *colorballs/p21* et *p31*, et les problèmes *laboratory*. CTCFL calcule des plans plus lentement que CTCFE et CTCF+ pour le problème *logistics/p5*, plus lentement que CTCF+, CTCFE et CTCF pour les problèmes *logistics/p1*, l'ensemble des problèmes *blocks*, *ebtcs/p10*, *p30* et *p50*, *erovers/p4*, et l'ensemble des problèmes *elogistics*. Enfin, CTCFL calcule des plans plus lentement que CTCFE pour les problèmes *ebtcs/p70*, *erovers/p20*, et plus lentement que CTCFE et CTCF pour le problème *erovers/p2*.

Dans le deuxième graphique de la figure 8.7 et le tableau B.2, nous pouvons remarquer que CTCFL calcule des plans plus petits que CTCF+, CTCFE et CTCF pour les problèmes *blocks/p15*, *erovers/p200*, *p300*, *colorballs/p21*, *p41* et *p51*. CTCFL calcule des plans plus petits que CTCFE pour le problème *grid/p5*, plus petits que CTCF+ pour le problème *egrid/p4*, et plus petits que CTCF+ et CTCF pour le problème *erovers/p20*. Parmi ces problèmes, CTCFL calcule des plans plus petits que Contingent-FF, CLG et PO-PRP pour les problèmes *colorballs/p31*, *p41* et *p51* et des plans plus petits que CLG pour le problème *erovers/p300*. CTCFL calcule aussi des plans plus petits que Contingent-FF pour *erovers/p120* et *colorballs/p21*. En revanche, CTCFL calcule des plans plus grands que PO-PRP pour *erovers/p500*, et des plans légèrement plus grands que CTCF, CTCFE et CTCF+ pour les problèmes *grid/p4*, *rovers*, *blocks/p7* et *p11*, *erovers/p60* et *p250*. CTCFL calcule aussi des plans plus grands que CTCF et CTCF+ pour le problème *erovers/p40*, plus grands que CTCF et CTCFE pour le problème *elogistics/p5*, plus grands que CTCF+ pour le problème *egrid/p3* et plus grands que CTCFE pour les problèmes *grid/p3* et *elogistics/p7*. Les pertes de performance de CTCFL sur la taille des plans de ces problèmes étant légères, les conclusions de comparaison des tailles de plans de CTCFL sont les mêmes sur ces problèmes que les conclusions des comparaisons de CTCF, CTCFE et CTCF+ par rapport à Contingent-FF, CLG et PO-PRP.

Dans le troisième graphique de la figure 8.7 et le tableau B.2, nous pouvons remarquer que CTCFL calcule des plans plus courts que CTCF, CTCFE et CTCF+ pour 8 problèmes : les problèmes *blocks/p15*, *erovers/p2*, *erovers/p200*, *p250* et *p300*, *colorballs/p21*, *p41* et *p51*. Parmi ces problèmes, CTCFL calcule des plans plus courts que Contingent-FF, CLG et PO-PRP pour les problèmes *erovers/p2*, *p200*, *p250*, *p300*, *colorballs/p21*, *p31*, *p41*, plus courts que PO-PRP pour les problèmes *blocks/p15*, *erovers/p500*, et plus courts que Contingent-FF pour les problèmes *erovers/p120* et *colorballs/p51*. CTCFL calcule aussi des plans plus courts que CTCFE et CTCF pour les problèmes *blocks/p11*, *erovers/p4*, *elogistics/p5*, et plus courts que CTCFE pour les problèmes *grid/p5* et *elogistics/p7*. En revanche, CTCFL calcule des plans plus longs que CTCF, CTCFE et CTCF+ pour 5 problèmes : les problèmes *grid/p4*, *rovers*, *erovers/p20* et *p60*. CTCFL calcule également des plans plus longs que CTCFE pour les problèmes *grid/p3* et *elogistics/p3*, des plans plus longs que CTCF+ pour le problème *egrid/p3*, et plus longs que CTCF pour le problème *egrid/p4*. Parmi ces problèmes, CTCFL calcule des plans plus longs que Contingent-FF et CLG pour les problèmes *rovers/p4* et *erovers/p60*, des plans plus longs que CLG pour *rovers/p2*, et plus longs que Contingent-FF pour le problème *erovers/p20*.

Enfin, dans le dernier graphique de la figure 8.7 et le tableau B.2, nous pouvons remarquer que CTCFL inclut autant d'observations que CTCF+, CTCF et CTCFE dans l'ensemble des problèmes qu'ils peuvent résoudre. Les comparaisons entre CTCF, Contingent-FF, CLG et PO-PRP sont donc les mêmes pour CTCFL sur ces problèmes. En ce qui concerne les problèmes que CTCF, CTCFE et CTCF+ ne peuvent pas résoudre, à savoir *erovers/p120* et *p500*, CTCFL incorpore moins d'observations que Contingent-FF pour *erovers/p120*, et en incorpore autant que PO-PRP pour le problème *erovers/p500*.

Problèmes de largeur contingente supérieure à 1

Intéressons-nous maintenant aux problèmes de largeur contingente supérieure à 1. Les résultats de CTCFL sur les problèmes de cette catégorie sont détaillés dans le tableau B.4. En comparant les résultats du tableau B.4 et du tableau B.3, on peut remarquer que CTCFL résout deux problèmes de

plus que CTCF+, trois de plus que CTCFE et six de plus que CTCF.

Si on analyse le temps de calcul des plans, on peut voir que CTCFL calcule des plans plus rapidement que CTCFE, CTCF et CTCF+ pour les problèmes *rovers/p6*, *doors/p6*, *colorballs*/p31* et *p41*. Parmi ces problèmes, CTCFL est plus rapide que Contingent-FF pour le problème *colorballs*/p41*. On peut aussi noter que CTCFL calcule des plans plus rapidement que CTCFE et CTCF pour le problème *rovers/p8*, et plus rapidement que CTCF et CTCF+ pour le problème *doors/p4*. Cependant, CTCFL calcule des plans plus lentement que CTCFE et CTCF+ pour les problèmes *erovers/p6*, *doors/p5* et *colorballs*/p21*, et plus lentement que CTCF, CTCFE et CTCF+ pour les problèmes *erovers/p8*, *doors/p3*.

Si on regarde la taille des plans calculés, on peut se rendre compte que CTCFL calcule des plans plus petits que CTCF+, CTCFE et CTCF pour les problèmes *doors/p4* et *p5*, plus petits que CTCF+ pour les problèmes *erovers/p6*, plus petits que CTCFE pour le problème *colorballs*/p21*, et plus petits que Contingent-FF pour *doors/p6*. En revanche, CTCFL calcule des plans de plus grande taille que CTCF+ pour le problème *colorballs*/p31*, de plus grande taille que CTCF+, CTCFE et CTCF pour le problème *erovers/p8* et de plus grande taille que Contingent-FF pour le problème *colorballs*/p41*.

En ce qui concerne la profondeur des plans, CTCFL calcule des plans plus courts que CTCF+, CTCFE et CTCF pour les problèmes *erovers/p6* et *doors/p5*, plus courts que CTCF+ et CTCF pour le problème *doors/p4*, et plus court que CTCFE pour le problème *colorballs*/p21*. De plus, CTCFL calcule un plan plus court que Contingent-FF pour le problème *colorballs*/p41*, et plus court que Contingent-FF et PO-PRP pour le problème *doors/p6*. En revanche, CTCFL calcule un plan légèrement plus long que CTCF+, CTCFE et CTCF pour le problème *erovers/p8*.

Enfin, nous pouvons constater que CTCFL incorpore autant d'observation que les autres versions de l'approche à l'exception du problème *colorballs*/p21* pour lequel CTCFL incorpore moins d'observation que CTCFE. Comme CTCF, CTCFE et CTCF+, CTCFL incorpore moins d'observation que Contingent-FF pour les problèmes *rovers*, *doors/p3*, *p4* et *p5*. De plus, CTCFL incorpore moins d'observations que Contingent-FF pour les problèmes *doors/p6*, *colorballs*/p21*, *p31* et *p41*.

Problèmes coûteux

La dernière catégorie de problèmes que nous devons étudier sont les problèmes coûteux. Les résultats de l'implémentation de CTCFL sur les problèmes coûteux sont illustrés par la figure 8.8 et détaillés dans les tableaux B.6 et B.5. Nous pouvons remarquer dans un premier temps que CTCFL parvient à résoudre un problème de plus que CTCF+. CTCFL parvient donc à résoudre tous les problèmes coûteux contrairement à CTCF+, CTCFE et CTCF.

Dans le premier graphique de la figure 8.8 et dans le tableau B.6, on peut voir que CTCFL calcule des plans plus rapidement que CTCF+, CTCFE et CTCF pour les problèmes *btcs+/p30*, *p50* et *p70*, *rovers+/p6* et *p8*. On peut aussi remarquer que CTCFL calcule un plan plus rapidement que CTCF+ et CTCF pour le problème *btcs+/p10*, et plus rapidement que CTCF+ pour le problème *doors+/p6*. Parmi ces problèmes, CTCFL calcule des plans plus rapidement que Contingent-FF pour les problèmes *btcs+/p50*, *p70* et *rovers+/p6*. En revanche, CTCFL calcule des plans plus lentement que CTCF+ pour les problèmes *erovers+*, *elogistics+*, *doors+/p3* et *doors+/p5*, et plus lentement que CTCF+, CTCFE et CTCF pour les problèmes *rovers+/p2* et *p4*.

Dans le deuxième graphique de la figure 8.8 et dans le tableau B.6 nous pouvons remarquer que CTCFL calcule des plans plus petits que CTCF+ pour les problèmes *rovers+/p6*, *erovers+/p6*, *elogistics+/p5* et *doors+/p5*. CTCFL calcule des plans plus petits que Contingent-FF, CLG et PO-PRP pour le problème *elogistics+/p3*, plus petits que Contingent-FF et PO-PRP pour les problèmes *btcs+*, *rovers+*, *erovers+/p2* et *p4*, *doors+/p3*, *p4* et *p5*. De plus, CTCFL calcule des plans plus petits que Contingent-FF pour les problèmes *erovers+/p6* et *doors+/p6*, *erovers+/p6* et *p8*, plus petits que Contingent-FF et CLG pour les problèmes *elogistics+/p5* et *p7*, et plus petits que PO-PRP pour le problème *elogistics+/p1*. Cependant, CTCFL calcule des plans plus grands que CTCF, CTCFE et CTCF+ pour les problèmes *rovers+/p2* et *p4*, et plus grands que CTCF+ pour les problèmes *erovers+/p2*, *p4*, *p8* et *elogistics+/p7*.

Dans le troisième graphique de la figure 8.8 et dans le tableau B.6, nous pouvons remarquer que CTCFL calcule des plans de plus petite profondeur que CTCF+ et CTCF pour le problème *rovers+/p6*, et de plus petite profondeur que CTCF+ pour les problèmes *erovers+/p6* et *doors+/p5*. CTCFL calcule des plans de plus petite profondeur que Contingent-FF, CLG et PO-PRP pour les problèmes *erovers+/p2*, *elogistics+/p3* et *p5*, et de plus petite profondeur que Contingent-FF et

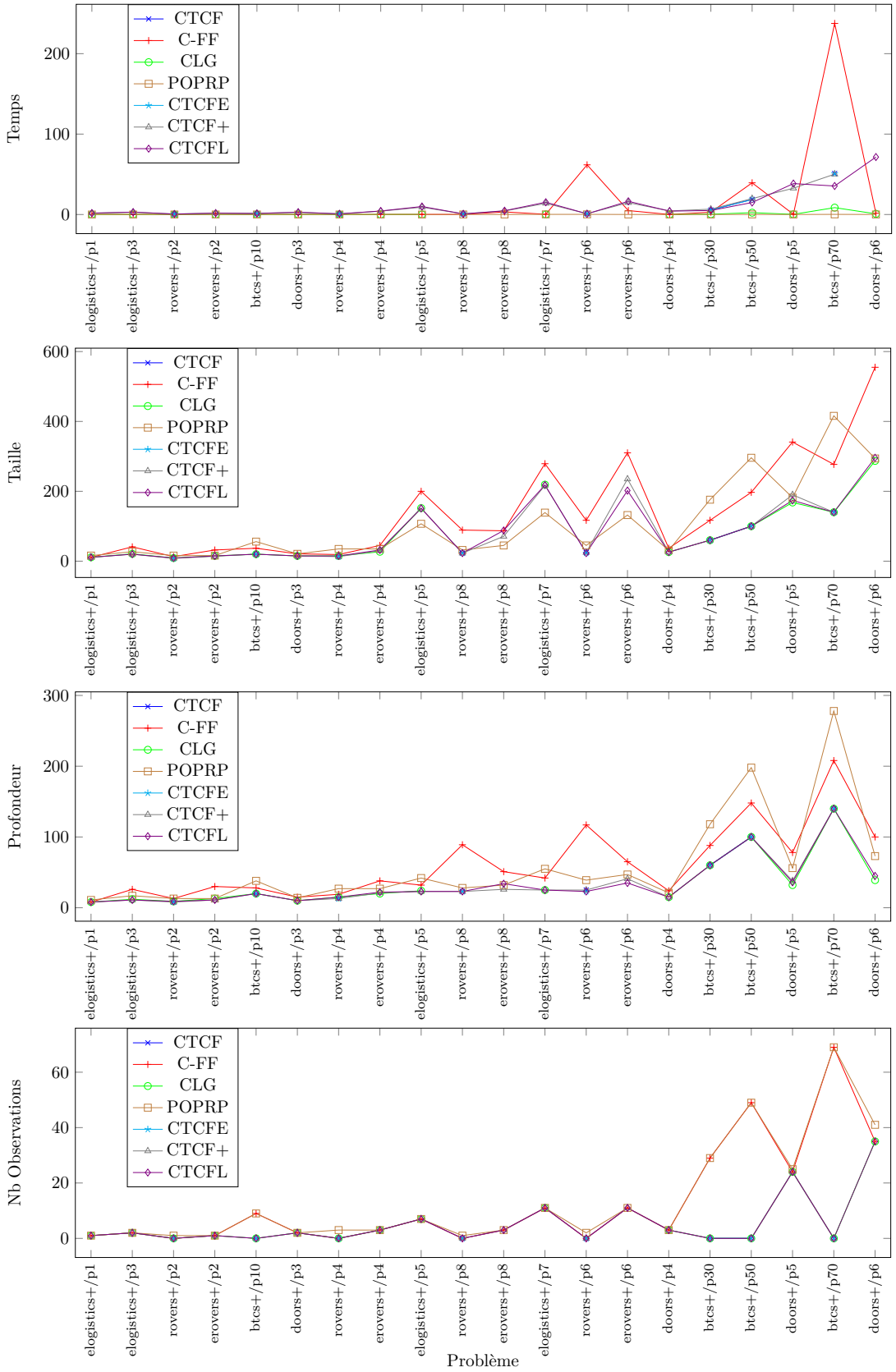

FIGURE 8.8 – Comparaison du temps de calcul, de la taille, de la profondeur, et du nombre d'observations du plan pour les problèmes coûteux

PO-PRP pour les problèmes *btcs+*, *rovers+*, *erovers+/p4* et *p6*,*elogistics+/p7* et *doors+*. Nous pouvons aussi remarquer que CTCFL calcule des plans de plus petite profondeur que Contingent-FF pour le problème *erovers+/p8*, et de plus petite profondeur que PO-PRP pour le problème *elogistics+/p1*. Cependant, CTCFL calcule des plans de plus grande profondeur que CTCF+, CTCFE et CTCF pour les problèmes *rovers+/p2*, *p4*, et de plus grande profondeur que CTCF+ pour les problèmes *erovers+/p4* et *p8*.

Enfin, dans le dernier graphique de la figure 8.8 et dans le tableau B.6, nous pouvons remarquer que CTCFL incorpore autant d'observations dans les plans calculés que CTCF+, CTCFE et CTCF pour l'ensemble des problèmes que ces versions de l'approche peuvent résoudre. Nous pouvons aussi noter que CTCFL incorpore moins d'observations que Contingent-FF et PO-PRP pour les problèmes *btcs+*, et moins d'observations que PO-PRP pour les problèmes *rovers+*, *doors+/p5* et *p6*.

Conclusions des résultats

À partir des résultats expérimentaux, nous pouvons déterminer dans un premier temps que CTCFL résout 5 problèmes de plus que CTCF+, 26 de plus que CTCFE et 27 de plus que CTCF (soit 76 problèmes résolus sur 87 problèmes). Cette amélioration réside dans le fait que la nouvelle représentation des états de croyance réduit considérablement le temps de calcul des plans pour les problèmes de grande taille, permettant de résoudre ces problèmes dans le temps imparti contrairement à CTCF, CTCFE et CTCF+. Nous pouvons remarquer une légère perte de performance de temps de calcul pour certains problèmes dont la taille est petite. Cependant cette légère perte de temps de calcul pour ces problèmes semble insignifiante si on la compare au gain de temps de calcul pour les problèmes de grande taille. CTCFL incorpore de manière générale autant d'observations que CTCF+, CTCFE et CTCF. En ce qui concerne la taille et la profondeur des plans, dans la plupart des cas CTCFL a des performances similaires à CTCF, CTCF+ et CTCFE à quelques actions près. Parmi ces exceptions, il existe plus de problèmes pour lesquels CTCFL calcule des plans plus petits ou plus courts que ces autres versions de l'approche, que de problèmes pour lesquels CTCFL calcule des plans plus grands ou plus longs. On peut aussi remarquer que dans la plupart des cas, et particulièrement pour les problèmes coûteux, lorsque CTCFL incorpore moins d'observations que Contingent-FF, CLG et PO-PRP, celui-ci calcule des plans plus petits et plus courts que ceux-ci. Cela indique que lorsque les observations sont bien considérées comme plus coûteuses qu'une action, leur incorporation semble bien avoir une influence non négligeable sur la qualité des plans calculés (ici en terme de taille et de profondeur des plans).

8.5 Discussion

La représentation des états de croyance de façon implicite permet de résoudre la plupart des problèmes de grande taille étudiés grâce à une réduction importante du temps de calcul du plan. Cependant, deux limites principales de l'approche restent non résolues. CTCFL n'est toujours pas en mesure de résoudre des problèmes dans lesquels les observations sont indirectes, à savoir les problèmes *wumpus* et *wumpus**. De plus, CTCFL ne comporte aucun mécanisme de retour sur le choix des observations candidates, ce qui peut empêcher l'approche de trouver une solution si le plan calculé après l'incorporation d'une observation ne mène finalement pas au but du problème. Pour réussir à résoudre les problèmes pour lesquels une observation indirecte est nécessaire, une idée peut être de développer un processus de sélection des observations différent de celui proposé dans les différentes versions de l'approche. Ce processus serait basé sur une discrimination directe de l'état de contre-exemple des autres états de l'état de croyance, en déterminant directement les propositions qui diffèrent entre le contre-exemple et les autres états de croyance sans se baser sur le plan retourné par CPCES*. Pour implémenter un retour sur le choix des observations, une solution peut être de garder en mémoire les différentes observations candidates afin de réaliser un backtracking sur ces observations en cas d'échec de calcul d'un plan.

8.6 Conclusion

Dans ce chapitre, nous avons développé une nouvelle version de l'approche dont la spécificité consiste à ne plus représenter les états de croyance de manière explicite, mais plutôt comme l'association de l'état de croyance initial et d'une séquence d'actions et d'observations menant à l'état de croyance

courant depuis l'état de croyance initial. Pour cela, le plan menant à l'observation sélectionnée est transformé en séquence d'actions et d'observations. Cette séquence d'actions et d'observation est ensuite transmise à CPCES* sous la forme d'une séquence d'actions et d'actions artificielles, et d'un ensemble de contraintes SMT permettant de préciser à CPCES* le résultat des différentes observations présentes dans la séquence d'actions et d'observation originelle afin que celui-ci puisse reconstituer l'état de croyance courant et calculer un plan menant au but du problème à partir de cet état de croyance. Pour forcer CPCES* à démarrer son calcul de plan par la bonne séquence d'actions et d'observations, cette séquence est directement imposée dans le domaine PDDL transmis à CPCES*. Contrairement à CTCF+, CTCFE et CTCF, cette représentation des états de croyance permet de résoudre la plupart des problèmes de grande taille étudiés grâce à une réduction importante du temps de calcul du plan. En ce qui concerne la taille et la profondeur des plans, dans la plupart des cas CTCFL a des performances similaires à CTCF, CTCF+ et CTCFE à quelques actions près. De plus, lorsque CTCFL incorpore moins d'observations que Contingent-FF, CLG et PO-PRP, celui-ci calcule généralement des plans plus petits et plus courts que ceux-ci, particulièrement pour les problèmes coûteux. Ceci indique que lorsque les observations sont bien considérées comme plus coûteuses qu'une action, leur incorporation semble bien avoir une influence non négligeable sur la qualité des plans calculés (ici en terme de taille et de profondeur des plans), soulignant l'efficacité de la limitation du nombre d'observations du plan. Malgré ces améliorations pour les problèmes de grande taille, CTCFL n'est pas complet, celui-ci ne pouvant pas résoudre des problèmes dans lesquels les observations sont indirectes, et celui-ci ne comportant aucun mécanisme de retour sur le choix des observations candidates.

Pour résoudre ces limites, nous allons étudier dans le prochain chapitre l'implémentation d'un processus de sélection des observations basé sur une discrimination directe de l'état de contre-exemple des autres états de l'état de croyance, en déterminant les propositions qui diffèrent entre le contre-exemple et les autres états de croyance sans se baser sur le plan retourné par CPCES*. Un processus de retour sur le choix des observations sera aussi implémenté afin de rendre l'approche complète.

Chapitre 9

Sélection alternative des observations et retour possible sur le choix des observations

9.1 Motivation

Du fait de sa représentation des états de croyances compacte, CTCFL parvient à résoudre plus de problèmes que les autres versions de l'approche dans le temps imparti. Grâce à ses caractéristiques, CTCFL semble être la version de l'approche la plus prometteuse. Cependant, CTCFL n'est pas capable de résoudre les problèmes dans lesquels les observations sont indirectes. De plus, CTCFL possède un problème de complétude, celui-ci n'implémentant aucune procédure de retour sur le choix des observations en cas d'échec de calcul d'un plan après l'ajout d'une observation.

Dans ce chapitre, nous allons justement étudier une dernière version de l'approche basée sur les mécanismes implémentés dans CTCFL implémentant dans un premier temps une sélection alternative des observations permettant la détection d'observations indirectes, et dans un deuxième temps, un retour sur les observations candidates en cas d'échec de calcul d'un plan contingent afin de régler le problème de complétude de CTCFL.

9.2 Mise en oeuvre

Nous nommons cette dernière version de l'approche Backtracking based ConTingent planner using a ConFormant planner (BCTCF). Afin de la simplifier, notre approche à été séparée en deux algorithmes majeurs : la procédure ContingentPlanning dans laquelle le planificateur conformant est appelé, et dans laquelle les procédures de recherche d'observations sont réalisées ; et la procédure Backtracking qui prend en charge le backtracking sur les observations candidates et formule les sous problèmes permettant le calcul des branches du plan solution.

Dans la section 9.2.1, nous détaillons la nouvelle procédure contingente de cette version de l'approche. Pour permettre de résoudre les problèmes dans lesquels l'observation incorporée au plan ne mène finalement pas au calcul d'une solution, nous définissons une procédure permettant d'effectuer un retour sur le choix de l'observation, la procédure Backtracking, permettant de sélectionner une autre observation candidate en cas d'échec (section 9.2.2). Afin de pouvoir résoudre des problèmes comportant des observations indirectes de l'environnement, nous définissons une nouvelle procédure de sélection des observations utilisée par la procédure ContingentPlanning en cas d'échec de la procédure findObservation+. Cette procédure est détaillée dans la section 9.2.3. Nous analysons les modifications apportées à l'architecture générale de notre approche dans la section 9.2.4. Une analyse théorique de l'approche est réalisée dans la section 9.3, et une analyse expérimentale de l'approche est développée dans la section 9.4. Enfin, les sections 9.5 et 9.6 concluent cette approche.

9.2.1 Procédure contingente

Intéressons-nous tout d'abord à la procédure contingente de cette dernière version de l'approche. Cette procédure est décrite par l'algorithme 13.

La procédure contingente prend en entrée un élément de plus que la procédure contingente de CTCFL (algorithme 10) : une liste Ω_{tabu} (initialement vide) contenant les observations déja explorées par le mécanisme de sélection des observations. Ω_{tabu} empêche la procédure Backtracking de réexplorer sans cesse les mêmes observations, et permet donc de faire converger la recherche d'observations.

Tout comme dans les procédures contingentes des versions précédentes, CPCES* est appelé pour calculer un plan conformant à partir du problème $\mathcal{P}$, du plan initial π_i et de l'ensemble de contraintes φ_i (ligne 2). Si un tel plan existe, alors on le retourne (ligne 4), sinon la procédure findObservation+ est appelée pour obtenir un ensemble d'observations candidates $\Omega_{\pi,\gamma}$ (ligne 5), et les observations de la liste Ω_{tabu} sont supprimées de $\Omega_{\pi,\gamma}$ (ligne 6).

Si un ensemble d'observations est obtenu avec succès par la procédure findObservation+, alors la procédure Backtracking (algorithme 14) explore les observations de $\Omega_{\pi,\gamma}$ et lance récursivement la procédure ContingentPlanning pour calculer les différentes branches du plan solution π_c. Cette procédure est détaillée dans la section 9.2.2. Si la procédure Backtracking parvient à calculer un plan π_c à partir de $\Omega_{\pi,\gamma}$, $\mathcal{P}_\Omega$, π_i, φ_i et Ω_{tabu}, alors celui-ci est retourné.

Si findObservation+ ne parvient pas à trouver d'observation candidate, ou bien si la procédure Backtracking ne parvient pas à trouver une solution en considérant l'ensemble d'observations $\Omega_{\pi,\gamma}$, alors on propose deux procédures alternatives de recherche d'un nouvel ensemble de propositions à observer : les procédures Props et Diff. Ces procédures sont expliquées en détail dans la section 9.2.3.

Algorithm 13 ContingentPlanning Procedure

Require: $\mathcal{P}_\Omega = (\Sigma, \mathcal{B}_0, g), \pi_i, \varphi_i, \Omega_{tabu}$
Ensure: π_c
1: $\mathcal{P} \leftarrow (S, A, T, \mathcal{B}_0, g)$
2: $\pi, \gamma \leftarrow \text{CPCES}(\mathcal{P}, \pi_i, \varphi_i)$
3: **if** $\gamma =$ null **then**
4: **return** π
5: $\Omega_{\pi,\gamma} \leftarrow$ findObservation+$(\mathcal{P}_\Omega, \pi, \gamma, \Omega_{tabu})$
6: $\Omega_{\pi,\gamma} := \Omega_{\pi,\gamma} / \Omega_{tabu}$
7: **if** $\Omega_{\pi,\gamma} \neq \emptyset$ **then**
8: $\pi_c \leftarrow$ Backtracking$(\Omega_{\pi,\gamma}, \mathcal{P}_\Omega, \pi_i, \varphi_i, \Omega_{tabu})$
9: **if** $\Omega_{\pi,\gamma} == \emptyset \ \vee \ \pi_c ==$ null **then**
10: $\mathcal{B}_i \leftarrow T(\mathcal{B}_0, \pi_i)$ considering φ_i
11: $P \leftarrow$ Props$(\gamma, \mathcal{B}_i)$
12: $\Omega_P \leftarrow \{\sigma \in \Omega / \Omega_{tabu} \mid \mathit{Eff}(\sigma) == \nu(p), \ p \in P \ \}$
13: **if** $\Omega_P \neq \emptyset$ **then**
14: $\pi_c \leftarrow$ Backtracking$(\Omega_P, \mathcal{P}_\Omega, \pi_i, \varphi_i, \Omega_{tabu})$
15: **if** $\pi_c \neq$ null **then**
16: **return** π_c
17: $D \leftarrow$ Diff$(\gamma, \mathcal{B}_i, P)$
18: **for** $P_{\gamma,s} \in D$ **do**
19: $\Omega_{P_{\gamma,s}} \leftarrow \{\sigma \in \Omega / \Omega_{tabu} \mid \mathit{Eff}(\sigma) == \nu(p), p \in P_{\gamma,s}\}$
20: $\pi_c \leftarrow$ Backtracking$(\Omega_{P_{\gamma,s}}, \mathcal{P}_\Omega, \pi_i, \varphi_i, \Omega_{tabu})$
21: **if** $\pi_c \neq$ null **then**
22: **return** π_c
23: **return** null
24: **return** π_c

Afin d'utiliser ces procédures, on doit construire l'état de croyance courant $\mathcal{B}_i$ en appliquant le plan initial π_i à l'état de croyance initial $\mathcal{B}_0$ (ligne 10). On récupère ensuite l'ensemble des propositions P permettant d'isoler complètement l'état γ des autres états de l'état de croyance $\mathcal{B}$ grâce à la procédure Props (ligne 11). On construit le nouvel ensemble d'observations Ω_P en sélectionnant dans Ω les observations capables d'observer les propositions de P, et on enlève de cet ensemble les observations déja explorées présentes dans la liste Ω_{tabu} (ligne 12). Si cet ensemble n'est pas vide, alors on tente de calculer un plan contingent solution du problème en appelant la procédure Backtracking à partir de cet ensemble d'observation Ω_P, du problème contingent $\mathcal{P}_\Omega$, du plan initial π_i, des contraintes φ_i et de la liste Ω_{tabu} (ligne 14). Si un plan π_c est calculé, alors celui-ci est retourné (ligne 16).

Si la procédure Backtracking ne parvient pas à calculer un plan en prenant en compte Ω_P, alors on récupère la liste D des ensembles de propositions $P_{\gamma,s}$ qui diffèrent entre le contre-exemple γ et chaque état s de l'état de croyance grâce à la procédure Diff (ligne 17). On parcourt ensuite D en sélectionnant chaque ensemble de propositions $p_{\gamma,s}$ dans D (ligne 18). On sélectionne les observations $\Omega_{P_{\gamma,s}}$ de Ω permettant d'observer les propositions $P_{\gamma,s}$ et on en supprime les observations déja étudiées et présentes dans la liste Ω_{tabu} (lignes 19). À partir de cet ensemble d'observations on essaie de calculer un plan grâce à la procédure Backtracking. Si c'est un succès, on retourne le plan calculé π_c (ligne 22), sinon on relance la procédure avec une autre liste de propositions dans D jusqu'à ce l'on trouve une solution, ou bien que la liste D soit vide, auquel cas il n'existe pas d'observation permettant de discriminer le contre-exemple γ, et donc il n'existe pas de solution contingente au problème $\mathcal{P}_\Omega$ (ligne 23).

9.2.2 Retour possible sur le choix des observations

La procédure Backtracking illustrée par l'algorithme 14 calcule un plan contingent π_c solution du problème $\mathcal{P}_\Omega$. Pour cela, la procédure explore chaque observation σ de Ω_σ et appelle la procédure ContingentPlanning pour calculer un sous-plan π_σ menant à σ depuis $\mathcal{B}_0$, et deux sous-plans π_+ et π_- menant au but du problème $\mathcal{P}_\Omega$ depuis chaque résultat possible de σ. Si un de ces sous-plans ne

peut pas être calculé par la procédure ContingentPlanning, alors une autre observation $\sigma \in \Omega_\sigma$ est considérée jusqu'à ce qu'une solution soit trouvée ou bien que l'ensemble des observations de Ω_σ soit exploré. On peut noter que contrairement à CTCFL, chaque observation de Ω_σ est étudiée une par une pour permettre un retour sur le choix des observations.

La procédure Backtracking prend en entrée l'ensemble d'observations candidates Ω_σ, le problème contingent $\mathcal{P}_\Omega$, le plan initial π_i, l'ensemble de contraintes φ_i et une liste tabou d'observations Ω_{tabu}.

Algorithm 14 Backtracking Procedure

Require: $\Omega_\sigma, \mathcal{P}_\Omega, \pi_i, \varphi_i, \Omega_{tabu}$
Ensure: π_c
1: **for** σ in Ω_σ **do**
2: $\Omega_{tabu} := \Omega_{tabu} \cup \{\sigma\}$
3: let g_σ be an artificial proposition
4: $\mathcal{P}'_\Omega = (S, A \cup \{(Pre(\sigma), g_\sigma, \emptyset)\}, \Omega, T, \mathcal{B}_0, g_\sigma)$
5: $\pi_\sigma =$ ContingentPlanning$(\mathcal{P}'_\Omega, \pi_i, \varphi_i, \Omega_{tabu})$
6: **if** $\pi_\sigma \neq Null$ **then**
7: $paths :=$ Separate(π_σ)
8: **for** each couple π,φ_π of $paths$ **do**
9: $\varphi_+ := \varphi_\pi \oplus (v(p) == \top)$
10: $\pi_+ :=$ ContingentPlanning$(\mathcal{P}_\Omega, \pi, \varphi_+, \Omega_{tabu})$
11: **if** $\pi_+ ==$ null **then**
12: **break and goto 1**
13: $\varphi_- := \varphi_\pi \oplus (v(p) == \bot)$
14: $\pi_- :=$ ContingentPlanning$(\mathcal{P}_\Omega, \pi, \varphi_-, \Omega_{tabu})$
15: **if** $\pi_- ==$ null **then**
16: **break and goto 1**
17: $\pi := (\pi$;**if** σ **then** π_+ **else** $\pi_-)$
18: **Return** π_σ
19: **Return** *null*

Dans un premier temps, l'algorithme 14 parcourt une par une les observations présentes dans l'ensemble d'observations candidates Ω_σ (ligne 1). Chaque observation σ étudiée est ajoutée à la liste tabou Ω_{tabu} afin de pouvoir transmettre à la procédure contingente quelles observations ont été étudiées, dans le but de ne pas étudier plusieurs fois la même observation lors des processus de sélection d'observations (ligne 2).

Pour calculer le plan π_σ menant à l'observation σ considérée depuis l'état de croyance initial, on crée une proposition artificielle g_σ indiquant le fait que l'observation σ ait été réalisée (ligne 3). Le planificateur conformant utilisé ne considérant pas les observations, on crée ensuite une action artificielle ayant les mêmes préconditions que σ, et ayant pour seul effet la proposition g_σ. On crée un nouveau problème contingent $\mathcal{P}'_\Omega$ dans lequel cette action est ajoutée à l'ensemble d'actions du problème initial $\mathcal{P}_\Omega$, et on définit comme but de ce problème la proposition g_σ (ligne 4). La procédure ContingentPlanning est ensuite appelée à partir de $\mathcal{P}'_\Omega$ pour calculer le plan menant de $\mathcal{B}_0$ à g_σ (ligne 5), ce qui assure que nous pouvons appliquer l'observation σ à la suite de π_σ.

Si la procédure parvient à calculer un plan π_σ, alors comme dans CTCFL, on sépare le plan π_σ en séquences d'actions et d'observations π auxquelles on associe les contraintes φ_π correspondantes (ligne 7). L'étape suivante de la procédure est de calculer un plan menant au but du problème $\mathcal{P}_\Omega$ depuis chaque branche π de π_σ (ligne 8). Pour chaque branche π, on tente de calculer deux sous-plans π_+ et π_- : un pour chaque résultat de σ. Pour chaque sous-problème, on définit l'ensemble de contraintes φ encodant les résultats des observations de π auquel on ajoute le résultat positif ou négatif de l'observation σ (lignes 9 et 13), et on appelle la procédure ContingentPlanning pour calculer chaque sous-plan (lignes 10 et 14). Si un des sous-plans π_+ ou π_- ne peut pas être calculé par la procédure ContingentPlanning, alors on stoppe le calcul de sous-plan et on sort de la boucle *for* courante (lignes 12 et 16) pour considérer une autre observation σ de Ω_σ (ligne 1).

Une fois que les branches π_+ et π_- ont été calculées, on les ajoute à la suite de l'observation σ à la fin de la branche courante π (ligne 17). Finalement, si chaque branche de π_σ mène au but g du problème $\mathcal{P}_\Omega$, on retourne π_σ comme solution de $\mathcal{P}_\Omega$ (ligne 18). En revanche, si l'ensemble d'observations Ω_σ

a été entièrement exploré et qu'aucune solution n'a été trouvée, alors *null* est retourné, signifiant qu'aucun plan contingent solution n'a été trouvé considérant Ω_σ (ligne 19). Dans ce cas, la procédure ContingentPlanning appelle de nouveau la procédure Backtracking à partir d'un autre ensemble d'observations candidates, jusqu'à ce que toutes les observations possibles soient explorées ou bien qu'une solution soit trouvée.

9.2.3 Procédure de sélection d'observations alternative

Dans cette procédure, nous détaillons les procédures de sélection de propositions Props et Diff utilisées dans la procédure ContingentPlanning (algorithme 13) Afin de pouvoir identifier les observations potentiellement utiles que la procédure findObservation+ ne peut pas trouver, comme les observations indirectes par exemple, on propose de discriminer directement le contre-exemple des autres états de l'état de croyance en identifiant les propositions qui diffèrent entre ces états. Dans ce processus, nous devons manipuler les états de croyance de manière explicite comme dans CTCF, CTCFE et CTCF+, ce qui a un coût non négligeable que l'on accepte de payer afin de rendre l'approche complète (ligne 10 de l'algorithme 13). Cependant, en raison de ce coût, cette procédure est utilisée uniquement en secours si la procédure findObservation+ ne trouve pas d'observation en premier lieu.

Dans un premier temps, on propose d'identifier les propositions P dont l'observation permettrait d'isoler totalement le contre-exemple γ des autres états de l'état de croyance courant $\mathcal{B}_i$. Cette isolation permet d'éviter, lorsque c'est possible, d'effectuer plusieurs observations afin de discriminer le contre-exemple alors qu'une seule peut suffire en observant les propositions de cet ensemble. En effet, isoler totalement γ grâce à l'observation d'une de ces propositions revient à créer un état de croyance contenant uniquement γ, et un état de croyance contenant l'ensemble des autres états de l'état de croyance excepté γ. Le calcul d'un plan à partir de ces deux états de croyance ne nécessiteront jamais l'ajout d'observation pour discriminer de nouveau γ d'un autre état. P est calculé par la procédure Props qui peut se résumer par l'équation 9.1 ci-dessous : les propositions p sont celles présentes uniquement dans γ, et les propositions p' sont celles présentes dans tous les états de $\mathcal{B}_i$ à l'exception de γ.

$$P = \{p \mid \gamma \models p \wedge \mathcal{B}_i/\gamma \not\models p\} \cup \{p' \mid \gamma \not\models p' \wedge \mathcal{B}_i/\gamma \models p'\} \tag{9.1}$$

On définit ensuite les observations Ω_P comme l'ensemble des observations permettant d'observer les propositions P, en ne considérant pas les observations déja étudiées dans l'ensemble d'observations Ω_{tabu} (ligne 12), et la procédure Backtracking est appelée à partir de cet ensemble d'observations (ligne 14).

Si un plan ne peut pas être calculé, ou bien si Ω_P est vide, alors on propose une deuxième méthode alternative de sélection de propositions permettant de discriminer γ des autres états de l'état de croyance $\mathcal{B}_i$. Cette méthode consiste à construire un ensemble D d'ensembles de propositions $P_{\gamma,s}$, dont l'observation permettrait de distinguer γ de chaque état $s \in \mathcal{B}_i$. L'ensemble D est construit par la procédure Diff (ligne 17), dans laquelle chaque $P_{\gamma,s}$ est défini par l'équation 9.2 suivante :

$$P_{\gamma,s} = \{p \mid \gamma \models p \wedge s \not\models p\} \cup \{p' \mid \gamma \not\models p' \wedge s \models p'\} - P \tag{9.2}$$

Pour chaque état $s \in \mathcal{B}_i$, $P_{\gamma,s}$ contient les propositions p présentes dans p et non dans s, et les propositions présentes dans s mais pas dans γ. Les propositions déjà étudiées dans l'ensemble P retourné précédemment par la procédure Props ne sont pas intégrées dans ces ensembles de propositions. Chaque ensemble $P_{\gamma,s}$ est considéré par la procédure ContingentPlanning et l'ensemble d'observations $\Omega_{P_{\gamma,s}}$ capables d'observer une proposition de $P_{\gamma,s}$ est définie (ligne 19), puis la procédure Backtracking est appelée à partir de $\Omega_{P_{\gamma,s}}$ jusqu'à ce qu'une solution soit trouvée, ou bien que chaque $P_{\gamma,s}$ soit exploré, auquel cas aucune solution contingente n'existe pour ce problème.

Exemple: Discrimination alternative du contre-exemple

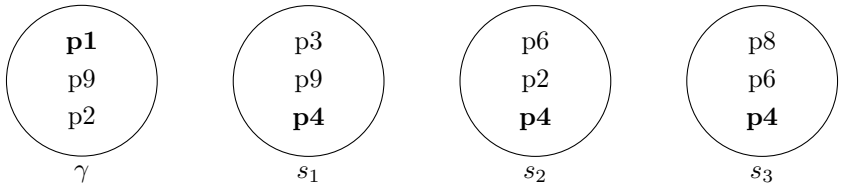

FIGURE 9.1 – Exemple d'état de croyance

Nous pouvons illustrer les procédures alternatives de sélection de propositions P et D en se basant sur l'exemple d'état de croyance de la figure 9.1. Dans ce cas, nous avons tout d'abord un ensemble de propositions $P = \{p1, p4\}$, étant donné que la proposition $p1$ est la seule présente uniquement dans γ, et $p4$ est l'unique proposition présente dans tous les états de croyance sauf γ.
Si on veut construire les ensembles de propositions $P_{\gamma,s}$ pour chaque état, on a :
$P_{\gamma,s_1} = \{p1, p2, p3, p4\}$, $P_{\gamma,s_2} = \{p1, p9, p6, p4\}$, $P_{\gamma,s_3} = \{p1, p2, p9, p8, p6, p4\}$.
Étant donné que l'on étudie l'ensemble de propositions P avant d'étudier l'ensemble D, on supprime de chaque $P_{\gamma,s}$ les propositions présentes dans P. On a donc :
$D = \{P_{\gamma,s_1} = \{p2, p3\}, P_{\gamma,s_2} = \{p9, p6\}, P_{\gamma,s_3} = \{p2, p9, p8, p6\}\}$.

9.2.4 Architecture

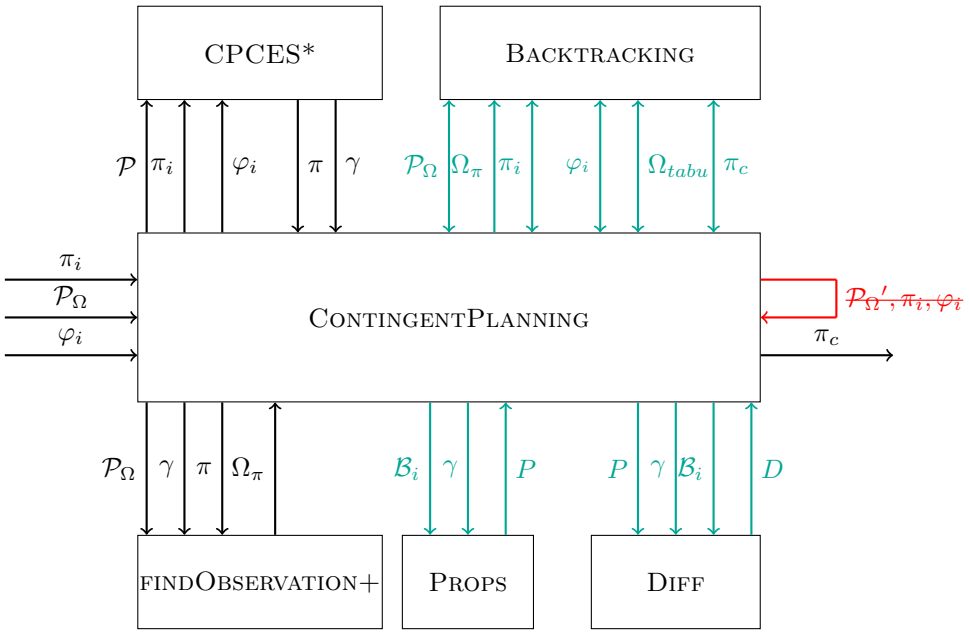

FIGURE 9.2 – Architecture du processus contingent

L'ajout de nouvelles procédures modifie l'architecture générale de l'approche présentée dans la section 8.2.8 du chapitre 8. Cette architecture illustrée par la figure 9.2 contient trois nouvelles procédures, la procédure BACKTRACKING, la procédure DIFF, et la procédure PROPS. Ces nouvelles procédures induisent de nouvelles communications entre les éléments de l'architecture. La procédure contingente transmet l'état de croyance courant $\mathcal{B}_i$ et le contre-exemple γ à la procédure PROPS afin que celle-ci retourne un ensemble de propositions P. De même, la procédure contingente transmet $\mathcal{B}_i$ et γ à la procédure DIFF, ainsi que l'ensemble P afin de retourner l'ensemble d'ensembles de propositions

D. Contrairement aux versions précédentes, la procédure contingente ne s'appelle pas elle même de manière récursive, mais appelle à la place la procédure BACKTRACKING en transmettant le problème contingent $\mathcal{P}_\Omega$, l'ensemble d'observations candidates Ω_π, le plan initial π_i, l'ensemble de contraintes φ_i et l'ensemble d'observations interdites Ω_{tabu} afin d'obtenir un plan contingent solution π_c. La procédure BACKTRACKING effectue plusieurs appels à la procédure contingente pour calculer les branches du plan solution π_c en transmettant à celle-ci le problème contingent $\mathcal{P}_\Omega$, la branche courante π_i, l'ensemble de contraintes φ_i et l'ensemble d'observations interdites Ω_{tabu}.

9.3 Analyse théorique

9.3.1 Complétude

Pour pouvoir déterminer que notre approche est complète, il faut vérifier que celle-ci trouve toujours une solution si au moins une solution existe. Dans l'article (Grastien et Scala, 2017), les auteurs indiquent que CPCES est sain et complet. Étant donné que CPCES est sain et complet, alors le processus de calcul des branches du plan est sain est complet (Brafman et Shani, 2012b).

Par conséquent, la complétude de l'approche repose sur la sélection des observations à incorporer au plan afin de former un plan contingent solution. Contrairement aux versions précédentes de l'approche, BCTCF inclut deux mécanismes permettant de rendre l'approche complète. Dans un premier temps, à chaque étape du calcul du plan, en cas d'échec de calcul d'un plan pour un ensemble d'observations courant, le mécanisme de recherche alternatif d'observations permet d'obtenir l'ensemble des propositions permettant de discriminer le contre-exemple des autres états de l'état de croyance courant. Grâce au mécanisme de retour sur le choix de l'observation, l'ensemble des observations permettant d'observer ces propositions est ensuite étudié jusqu'à ce qu'une observation permette de trouver une solution, ou bien que cet ensemble ait été entièrement parcouru. À chaque étape d'incorporation d'observations dans le plan, si une solution contingente existe, alors celle-ci contient au moins une des observations présentes dans cet ensemble d'observation, et sera donc étudiée et retournée en tant que solution. Si une solution n'est pas trouvée après avoir parcouru l'ensemble des observations pouvant permettre de discriminer le contre-exemple, alors aucune solution n'existe, puisqu'aucune observation ne peut être appliquée pour différencier cet état de contre-exemple des autres états, et qu'aucune solution conformante n'existe permettant de calculer un plan menant au but sans ajouter d'observations. Par conséquent, nous pouvons déterminer que notre approche est complète, celle-ci passant en revue toutes les observations possibles jusqu'à retourner une solution, ou bien qu'aucune solution contingente n'existe.

9.3.2 Complexité

Dans cette section, nous allons déterminer quelle est la complexité de BCTCF. Comme pour les chapitres précédents, nous simplifions les calculs en considérant l'application d'une action ou la vérification et la sélection d'une observation comme une opération unitaire, et nous sur-approximons les équations en ne gardant que les variables de plus grand ordre de grandeur.

Intéressons-nous tout d'abord à la procédure FINDOBSERVATION+. Cette procédure est la même que pour CTCFL et CTCF+. Par conséquent, sa complexité est dans le pire cas de $\mathcal{O}(|\pi| + |F|)$.

Essayons de déterminer la complexité de la procédure BACKTRACKING (algorithme 14). On note $\mathcal{T}'(n)$ la complexité de l'appel à la procédure contingente sur le sous-problème de calcul d'un plan menant à une observation (ligne 5), $\mathcal{S}(n)$ la complexité de la procédure SEPARATE, k le nombre de branches de π_σ impliqué dans la boucle de la ligne 8, $2\mathcal{T}(n/2)$ la complexité des deux appels à la procédure contingente pour le calcul des sous-plans partant de l'observation et menant au but du problème (lignes 10 et 14). On note $B(n)$ la complexité de la procédure BACKTRACKING avec dans le pire cas :

$$B(n) = |\Omega| \times (\mathcal{T}'(n) + \mathcal{S}(n) + k \times 2\mathcal{T}(n/2)) \tag{9.3}$$

On a $\mathcal{O}(\mathcal{S}(n)) = \mathcal{O}(|\pi|)$. $\mathcal{O}(\mathcal{B}(n))$ est dominé par $\mathcal{O}(T'(n))$. On a donc $\mathcal{O}(\mathcal{B}(n)) = \mathcal{O}(\mathcal{T}'(n))$.

Intéressons-nous à présent à la complexité de la procédure contingente (algorithme 13), que l'on note $\mathcal{T}(n)$. On note $\mathcal{C}(n)$ la complexité de CPCES prenant en entrée n états, $\mathcal{F}(n)$ la complexité de la procédure FINDOBSERVATION+, $P(n)$ la complexité de la procédure PROPS, et $D(n)$ la complexité de la procédure DIFF.

On a dans le pire cas :

$$\mathcal{T}(n) = \mathcal{C}(n) + \mathcal{F}(n) + B(n) + |\pi| \times 2^{|F|} + P(n) + B'(n) + D(n) + B''(n) \tag{9.4}$$

On a $\mathcal{O}(P(n)) = \mathcal{O}(D(n)) = \mathcal{O}(n)$. Dans l'équation 9.4, $|\pi| \times 2^{|F|}$ correspond à l'application du plan initial à l'état de croyance de la ligne 10, $B(n)$, $B'(n)$ et $B''(n)$ correspondent aux trois appels à la procédure Backtracking des lignes 8, 14 et 20 de l'algorithme 13.

La complexité de CPCES $\mathcal{C}(n)$ domine toujours les autres éléments de la complexité de $\mathcal{T}(n)$, la complexité de BCTCF est proportionnelle à $\mathcal{O}(\mathcal{C}(n))$ et donc a une complexité dans le pire des cas de $\mathcal{O}(2^{|F|})$. Cependant, l'ajout du facteur $|\Omega|$ de la procédure Backtracking (equation 9.3) joue un rôle important dans le temps de calcul d'une solution. À cause de ce facteur, on peut s'attendre à ce que l'approche soit plus lente que CTCFL pour les problèmes dans lesquels la première observation candidate n'est pas la bonne pour atteindre une solution, ceci entraînant le calcul de plans supplémentaires. Dans le cas où l'approche utilise la procédure alternative de sélection des observations, la manipulation explicite de l'état de croyance augmente l'occupation mémoire ($2^{|F|}$ états à stocker dans le pire cas) et le temps de calcul d'une solution, pouvant empêcher l'approche de résoudre des problèmes de très grande taille pour lesquels la procédure findObservation+ ne trouve pas de solution.

9.4 Expérimentations

Les expérimentations de l'implémentation de BCTCF se sont déroulées dans les mêmes conditions que pour CTCF, CTCFE, CTCF+ et CTCFL. Une limite de temps de 10 minutes maximum est imposée lors de la résolution des problèmes. Nous avons décidé de comparer cette nouvelle approche avec CTCF, CTCFE, CTCF+, Contingent-FF, CLG et PO-PRP sur les mêmes benchmarks : des problèmes de la littérature (auxquels on additionne le problème *laboratory*), des problèmes dont la largeur contingente est supérieure à 1, et des problèmes *coûteux*. L'intégralité de ces benchmarks sont décrits en détail en annexe de ce manuscrit.

9.4.1 Résultats

Dans un premier temps nous allons étudier les résultats de BCTCF en les comparant aux autres versions de l'approche, à Contingent-FF, CLG, et PO-PRP sur les benchmarks de la littérature. Dans un deuxième temps, nous étudierons les résultats sur les problèmes dont la largeur contingente est supérieure à 1. Finalement, nous étudierons les résultats sur les problèmes *coûteux*. Les résultats des problèmes de la littérature sont illustrés dans la figure 9.3 et les résultats des problèmes *coûteux* sont illustrés par la figure 9.4. Tout comme les graphiques de résultats des chapitres précédents, ces deux figures sont découpées en quatre graphiques représentant pour chaque problème le temps de calcul du plan en secondes, la taille du plan en nombre total d'actions et d'observations, la profondeur du plan (taille de la plus longue séquence d'actions et d'observations, et enfin le nombre total d'observations du plan. Dans ces graphiques, les problèmes sont triés de manière croissante selon un critère de difficulté choisi comme le produit du nombre de propositions par état et du nombre d'états composant l'état de croyance initial.

Problèmes de la littérature

Les résultats de l'implémentation de BCTCF sur les problèmes de la littérature sont illustrés par la figure 9.3 et détaillés dans les tableaux B.1 et B.2 en annexe de ce manuscrit.

Dans un premier temps, on peut voir que BCTCF parvient à résoudre tous les problèmes à l'exception des problèmes *wumpus/p5* et *wumpus/p6*, pour lesquels le planificateur dépasse le temps limite pour trouver une solution. Contrairement aux versions précédentes de l'approche, BCTCF parvient à résoudre les problèmes *wumpus/p3* et *p4* grâce à la procédure de sélection d'observations alternative. BCTCF résout donc plus de problèmes de la littérature que Contingent-FF et PO-PRP, mais résout un problème de moins que CLG dans le temps imparti.

En regardant le premier graphique de la figure 9.3 et le tableau B.2, nous pouvons analyser le temps de calcul d'une solution des différentes approches. Nous pouvons remarquer que BCTCF calcule des plans plus rapidement que les autres versions de l'approche pour 11 problèmes : les problèmes *grid*, *rovers*, *logistics/p1*, *p3*, *p7*, *erovers/p500* et *colorballs/p51*. BCTCF calcule des plans plus rapidement

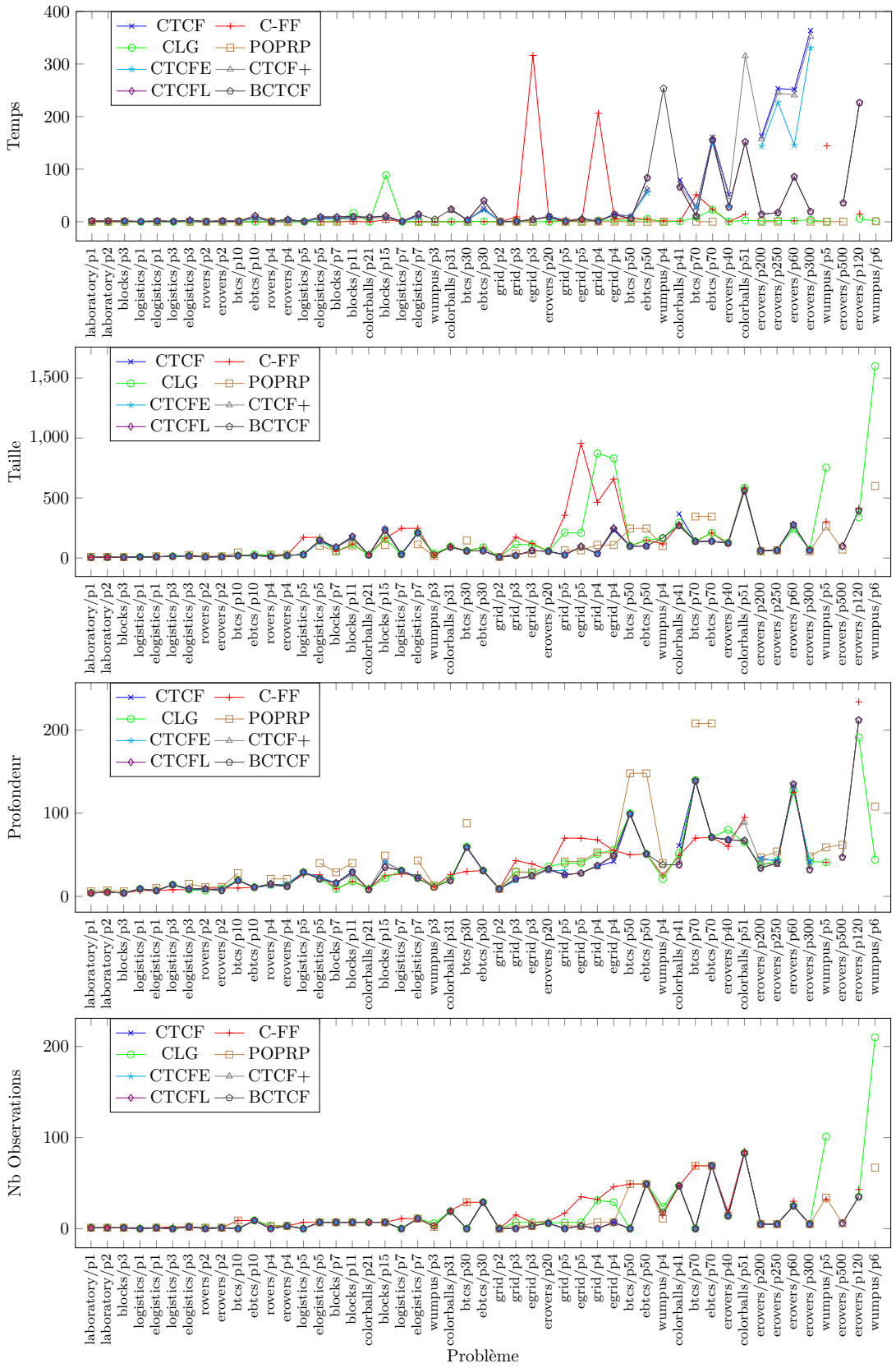

FIGURE 9.3 – Comparaison du temps de calcul, de la taille, de la profondeur, et du nombre d'observations du plan pour les problèmes de la littérature

que toutes les versions de l'approche sauf CTCFE pour le problème *logistics/p5*. BCTCF calcule aussi des plans plus rapidement que toutes les versions de l'approche sauf CTCFL pour les problèmes *btcs*, *egrid*, *erovers/p40*, *p60*, *p120*, *p200*, *p250* et *p300*, plus rapidement que CTCFL pour les problèmes *blocks*, *ebtcs/p10* et *p30*, *elogistics/p7*, *colorballs/p21*, *laboratory/p1* et *p2*. Enfin, BCTCF calcule des plans plus rapidement que CTCF et CTCF+ pour les problèmes *erovers/p20* et *colorballs/p41*. En revanche, BCTCF calcule des plans plus lentement que CTCF et CTCFE pour le problème *erovers/p2*, plus lentement que CTCF, CTCFE et CTCF+ pour les problèmes *erovers/p4* et *elogistics/p1*, légèrement plus lentement que CTCFL pour les problèmes *ebtcs/p50* et *p70*, plus lentement que CTCFL et CTCF+ pour les problèmes *colorballs/p31*. BCTCF calcule aussi des plans légèrement plus lentement que les autres versions de l'approche pour les problèmes *elogistics/p3* et *p5*. En ce qui concerne les planificateurs de la littérature, BCTCF calcule des plans plus rapidement que Contingent-FF et CLG pour le problème *grid/p4*, plus rapidement que Contingent-FF pour les problèmes *btcs/p50* et *p70*, *grid/p3* et *p5*, et *egrid/p3*. BCTCF est aussi plus rapide que CLG pour les problèmes *blocks/p11* et *p15*.

En regardant le deuxième graphique de la figure 9.3 et le tableau B.2, nous pouvons analyser la taille des plans calculés par les différentes approches. Nous pouvons remarquer que BCTCF calcule des plans de même taille que CTCFL. Par conséquent, les remarques sont les mêmes par rapport aux autres versions de l'approche : BCTCF calcule des plans plus petits que CTCF+, CTCFE et CTCF pour six problèmes : les problèmes *blocks/p15*, *erovers/p200*, *p300*, *colorballs/p21*, *p41* et *p51*. BCTCF calcule des plans plus petits que CTCFE pour le problème *grid/p5*, plus petits que CTCF+ pour le problème *egrid/p4*, et plus petits que CTCF+ et CTCF pour le problème *erovers/p20*. En revanche, BCTCF calcule des plans légèrement plus grands que CTCF, CTCFE et CTCF+ pour les problèmes *grid/p4*, *rovers*, *blocks/p7* et *p11*, *erovers/p60* et *p250*. BCTCF calcule aussi des plans plus grands que CTCF et CTCF+ pour le problème *erovers/p40*, plus grands que CTCF et CTCFE pour le problème *elogistics/p5*, plus grands que CTCF+ pour le problème *egrid/p3* et plus grands que CTCFE pour les problèmes *grid/p3* et *elogistics/p7*.

BCTCF calcule des plans plus petits que Contingent-FF, CLG et PO-PRP pour 12 problèmes : les problèmes *grid/p3*, *p4* et *p5*, *rovers*, *ebtcs*, *colorballs/p31*, *p41* et *p51*. BCTCF calcule aussi des plans plus petits que CLG et PO-PRP pour les problèmes *btcs*, plus petits que Contingent-FF et CLG pour les problèmes *egrid*, et plus petits que Contingent-FF pour les problèmes *logistics*, *erovers/p120*, *elogistics/p5*, *p7* et *colorballs/p21*. BCTCF calcule aussi des plans plus petits que CLG pour les problèmes *erovers/p20*, *p40*, *p300* et *wumpus/p3*, plus petits que Contingent-FF et PO-PRP pour les problèmes *erovers/p2* et *p4*, et des plans plus petits que PO-PRP pour les problèmes *elogistics/p1*, *p3* et *laboratory*. En revanche, BCTCF calcule des plans plus grands que PO-PRP pour *erovers/p250* et *p500*, plus grands que Contingent-FF et CLG pour les problèmes *erovers/p60*, et plus grands que Contingent-FF, CLG et PO-PRP pour le problème *wumpus/p4*.

En regardant le troisième graphique de la figure 9.3 et plus particulièrement le tableau B.2, nous pouvons analyser la profondeur des plans calculés par les différentes approches. Nous pouvons remarquer que BCTCF calcule des plans de même profondeur que CTCFL pour l'ensemble des problèmes que celui-ci peut résoudre. Par conséquent, les comparaisons que nous pouvons faire entre BCTCF et CTCF, CTCF+ et CTCFE sont les mêmes que pour CTCFL : BCTCF calcule des plans plus courts que CTCF, CTCFE et CTCF+ pour 8 problèmes : les problèmes *blocks/p15*, *erovers/p2*, *erovers/p200*, *p250* et *p300*, *colorballs/p21*, *p41* et *p51*. BCTCF calcule aussi des plans plus courts que CTCFE et CTCF pour les problèmes *blocks/p11*, *erovers/p4*, *elogistics/p5*, et plus courts que CTCFE pour les problèmes *grid/p5* et *elogistics/p7*. BCTCF calcule des plans plus courts que Contingent-FF, CLG et PO-PRP pour 14 problèmes : *grid/p3*, *p4*, *p5*, *egrid*, *erovers/p2*, *p4*, *p200*, *p250*, *p300*, *colorballs/p21*, *p31* et *p41*. De plus, BCTCF calcule des plans plus courts que CLG et PO-PRP pour les problèmes *btcs* et *wumpus/p3*, plus courts que Contingent-FF et PO-PRP pour les problèmes *rovers/p2*, *elogistics/p5* et *p7*, plus courts que PO-PRP pour les problèmes *rovers/p4*, *blocks*, *ebtcs/p50* et *p70*, *erovers/p500*, *elogistics/p1*, *p3*, *laboratory* et *wumpus/p4*. Enfin, BCTCF calcule des plans plus courts que CLG pour les problèmes *erovers/p20*, *p40*, et plus courts que Contingent-FF pour les problèmes *erovers/p120* et *colorballs/p51*. En revanche, BCTCF calcule des plans plus longs que CTCF, CTCFE et CTCF+ pour 5 problèmes : les problèmes *grid/p4*, *rovers*, *erovers/p20* et *p60*. BCTCF calcule également des plans plus longs que CTCFE pour les problèmes *grid/p3* et *elogistics/p3*, des plans plus longs que CTCF+ pour le problème *egrid/p3*, et plus longs que CTCF pour le problème *egrid/p4*. BCTCF calcule aussi des plans plus longs que Contingent-FF pour les problèmes *logistics*.

Nous pouvons maintenant nous intéresser au dernier graphique de la figure 9.3 et plus particulièrement

au tableau B.2. Nous pouvons analyser le nombre total d'observations ajoutées dans les plans calculés par les différentes approches. Nous pouvons remarquer que BCTCF calcule des plans comportant le même nombre d'observations que les précédentes versions de l'approche pour l'ensemble des problèmes que celles-ci peuvent résoudre. En ce qui concerne Contingent-FF, CLG et PO-PRP, nous pouvons nous rendre compte que BCTCF ajoute moins d'observations que Contingent-FF, CLG et PO-PRP pour 5 problèmes : *grid/p3*, *p4* et *p5* et *rovers*. BCTCF ajoute aussi moins d'observations que Contingent-FF et PO-PRP pour les problèmes *btcs*, moins d'observations que Contingent-FF et CLG pour les problèmes *egrid*, et moins d'observations que Contingent-FF pour les problèmes *logistics*, *erovers/p20*, *p40*, *p60*, *p120*, *colorballs/p31* et *p51*. En revanche, BCTCF ajoute plus d'observations que Contingent-FF et PO-PRP pour les problèmes *wumpus/p3* et *p4*.

Problèmes de largeur contingente supérieure à 1

Intéressons-nous maintenant aux problèmes de largeur contingente supérieure à 1. Les résultats de BCTCF sur les problèmes de cette catégorie sont détaillés dans le tableau B.4. En comparant les résultats du tableau B.4 et du tableau B.3, nous pouvons remarquer dans un premier temps que BCTCF parvient à résoudre deux problèmes de largeur contingente supérieure à 1 de plus que CTCFL : les problèmes *wumpus*/p3* et *wumpus*/p4*. BCTCF résout donc plus de problèmes de cette catégorie dans le temps imparti que CLG, et autant que PO-PRP, ces deux planificateurs n'étant pas complets pour ce type de problème. En revanche, BCTCF résout deux problèmes de moins que Contingent-FF dans le temps imparti.

Nous pouvons déterminer que BCTCF calcule des plans plus rapidement que les autres versions de l'approche pour les problèmes *rovers*, plus rapidement que les autres versions de l'approche à l'exception de CTCFE pour les problèmes *doors/p4* et *p5*, plus rapidement que CTCFL pour les problèmes *doors/p3*, *p6* et *colorballs*/p41*, et plus rapidement que CTCF+ pour les problèmes *colorballs*/p31*. En revanche, BCTCF calcule des plans plus lentement que l'ensemble des autres versions de l'approche pour le problème *erovers/p8*. BCTCF calcule aussi des plans plus lentement que CTCFE, CTCF+ et CTCFL pour les problèmes *erovers/p6* et *colorballs*/p21*. Nous pouvons tout de même noter que BCTCF calcule des plans plus rapidement que Contingent-FF pour le problème *colorballs*/p41*.

Si on s'intéresse à la taille des plans calculés, on peut se rendre compte que BCTCF calcule des plans de même taille que CTCFL pour les problèmes que celui-ci peut résoudre. BCTCF calcule des plans plus petits que CTCF+ pour le problème *erovers/p6*, plus petits que les autres versions de l'approche pour le problème *doors/p4* et *p5*, et plus petits que CTCFE pour le problème *colorballs*/p21*. En revanche, BCTCF calcule des plans plus grands que CTCF+, CTCFE et CTCF pour le problème *erovers/p8*, et plus grands que CTCF+ pour le problème *colorballs*/p31*. En ce qui concerne les planificateurs de la littérature, BCTCF calcule des plans plus petits que Contingent-FF et PO-PRP pour les problèmes *rovers*, *doors/p3*, *p4*, *p5* et *wumpus*/p3*, et des plans plus petits que Contingent-FF pour les problèmes *erovers*, *doors/p6* et *colorballs*/p21*.

Si on s'intéresse à la profondeur des plans calculés, on peut voir que BCTCF calcule des plans de même profondeur que CTCFL. BCTCF calcule des plans plus courts que les autres versions de l'approche pour les problèmes *erovers/p6*, *doors/p5*, plus courts que CTCF et CTCF+ pour le problème *doors/p4*, et plus courts que CTCFE pour le problème *colorballs*/p21*. En revanche, BCTCF calcule des plans légèrement plus longs que les autres versions de l'approche pour le problème *erovers/p8*. BCTCF calcule des plans plus courts que l'ensemble des planificateurs de la littérature étudiés pour les problèmes *rovers* et *erovers*, plus courts que Contingent-FF et PO-PRP pour les problèmes *doors/p3*, *p4*, *p5*, *p6* et *wumpus/p3*, et plus courts que Contingent-FF pour les problèmes *colorballs*/p21*, *p31* et *p41*.

En ce qui concerne le nombre d'observations ajoutées au plan calculé, BCTCF ajoute autant d'observations que les autres versions de l'approche pour l'ensemble des problèmes à l'exception du problème *colorballs*/p21* pour lequel BCTCF ajoute deux observations de moins que CTCFE. Comme CTCF, CTCFE, CTCF+ et CTCFL, BCTCF incorpore moins d'observation que Contingent-FF pour les problèmes *rovers*, *doors/p3*, *p4* et *p5*. De plus, BCTCF incorpore moins d'observations que Contingent-FF pour les problèmes *doors/p6*, *colorballs*/p21*, *p31* et *p41*. En revanche, on peut constater que BCTCF ajoute une observation de plus que Contingent-FF et PO-PRP pour le problème *wumpus*/p4*.

Problèmes coûteux

La dernière catégorie de problèmes que nous devons étudier sont les problèmes coûteux. Les résultats de l'implémentation de BCTCF sur les problèmes coûteux sont illustrés par la figure 9.4 et détaillés dans le tableau B.6. En comparant les résultats présents dans les tableaux B.5 et B.6, on peut se rendre compte que BCTCF, tout comme CTCFL, parvient à résoudre l'ensemble des problèmes coûteux contrairement aux autres versions de l'approche et à CLG.

En analysant le premier graphique de la figure 9.4 et les tableaux de résultats 5.2 et B.6, on peut se rendre compte que BCTCF calcule des plans plus rapidement que les autres versions de l'approche pour les problèmes *btcs+*, plus rapidement que CTCF, CTCFE et CTCF+ pour le problème *rovers+/p6*, plus rapidement que CTCF+ pour le problème *doors+/p4*, et plus rapidement que CTCFL pour les problèmes *doors+/p5* et *p6*. En revanche, BCTCF calcule des plans plus lentement que les autres versions de l'approche pour les problèmes *rovers+/p2*, *p4* et *p8*, plus lentement que CTCF+ pour les problèmes *doors+/p3*, et plus lentement que CTCF+ et CTCFL pour les problèmes *erovers+*, *elogistics+*. On peut tout de même noter que BCTCF calcule des plans plus rapidement que Contingent-FF pour les problèmes *btcs+/p50*, *p60* et *rovers+/p6*.

En analysant le deuxième graphique de la figure 9.4 et les tableaux de résultats 5.2 et B.6, on peut se rendre compte que BCTCF calcule des plans de même taille que CTCFL. Les comparaisons avec les autres versions de l'approche et les planificateurs de la littérature restent donc les mêmes : BCTCF calcule des plans plus petits que CTCF+ pour les problèmes *rovers+/p6*, *erovers+/p6*, *elogistics+/p5* et *doors+/p5*. BCTCF calcule des plans plus petits que Contingent-FF, CLG et PO-PRP pour le problème *elogistics+/p3*, plus petits que Contingent-FF et PO-PRP pour les problèmes *btcs+*, *rovers+*, *erovers+/p2* et *p4*, *doors+/p3*, *p4* et *p5*. De plus, BCTCF calcule des plans plus petits que Contingent-FF pour les problèmes *erovers+/p6* et *doors+/p6*, *erovers+/p6* et *p8*, plus petits que Contingent-FF et CLG pour les problèmes *elogistics+/p5* et *p7*, et plus petits que PO-PRP pour le problème *elogistics+/p1*. Cependant, BCTCF calcule des plans plus grands que CTCF, CTCFE et CTCF+ pour les problèmes *rovers+/p2* et *p4*, et plus grands que CTCF+ pour les problèmes *erovers+/p2*, *p4*, *p8* et *elogistics+/p7*.

En analysant le troisième graphique de la figure 9.4 et les tableaux de résultats 5.2 et B.6, nous pouvons remarquer que BCTCF calcule des plans de même profondeur que CTCFL. Par conséquent, les comparaisons de la profondeur des plans avec les autres versions de l'approche et les planificateurs de la littérature étudiés restent les mêmes : BCTCF calcule des plans de plus petite profondeur que CTCF+ et CTCF pour le problème *rovers+/p6*, et de plus petite profondeur que CTCF+ pour les problèmes *erovers+/p6* et *doors+/p5*. BCTCF calcule des plans de plus petite profondeur que Contingent-FF, CLG et PO-PRP pour les problèmes *erovers+/p2*, *elogistics+/p3* et *p5*, et de plus petite profondeur que Contingent-FF et PO-PRP pour les problèmes *btcs+*, *rovers+*, *erovers+/p4* et *p6*, *elogistics+/p7* et *doors+*. Nous pouvons aussi remarquer que BCTCF calcule des plans de plus petite profondeur que Contingent-FF pour le problème *erovers+/p8*, et de plus petite profondeur que PO-PRP pour le problème *elogistics+/p1*. Cependant, BCTCF calcule des plans de plus grande profondeur que CTCF+, CTCFE et CTCF pour les problèmes *rovers+/p2*, *p4*, et de plus grande profondeur que CTCF+ pour les problèmes *erovers+/p4* et *p8*.

Enfin, dans le dernier graphique de la figure 9.4 et dans les tableaux de résultats 5.2 et B.6, nous pouvons remarquer que BCTCF incorpore autant d'observations dans les plans calculés que les autres versions de l'approche pour l'ensemble des problèmes que ces versions de l'approche peuvent résoudre. Nous pouvons noter que BCTCF incorpore moins d'observations que Contingent-FF et PO-PRP pour les problèmes *btcs+*, et moins d'observations que PO-PRP pour les problèmes *rovers+*, *doors+/p5* et *p6*.

9.4.2 Conclusion des résultats

L'analyse des résultats de BCTCF sur les trois classes de problèmes étudiés nous permet de déterminer dans un premier temps que BCTCF résout 4 problèmes de plus que CTCFL, 9 de plus que CTCF+, 30 de plus que CTCFE, et 31 de plus que CTCF (soit 80 problèmes sur 87). De plus, BCTCF résout 1 problème de plus que Contingent-FF, 10 problèmes de plus que CLG et 12 problèmes de plus que PO-PRP. Le fait de résoudre plus de problèmes que les versions précédentes de l'approche dans le temps imparti réside dans la représentation compacte des états de croyance, mais aussi dans la capacité de pouvoir résoudre des problèmes dans lesquels les observations sont indirectes grâce à la procédure de sélection alternative des observations introduite dans BCTCF.

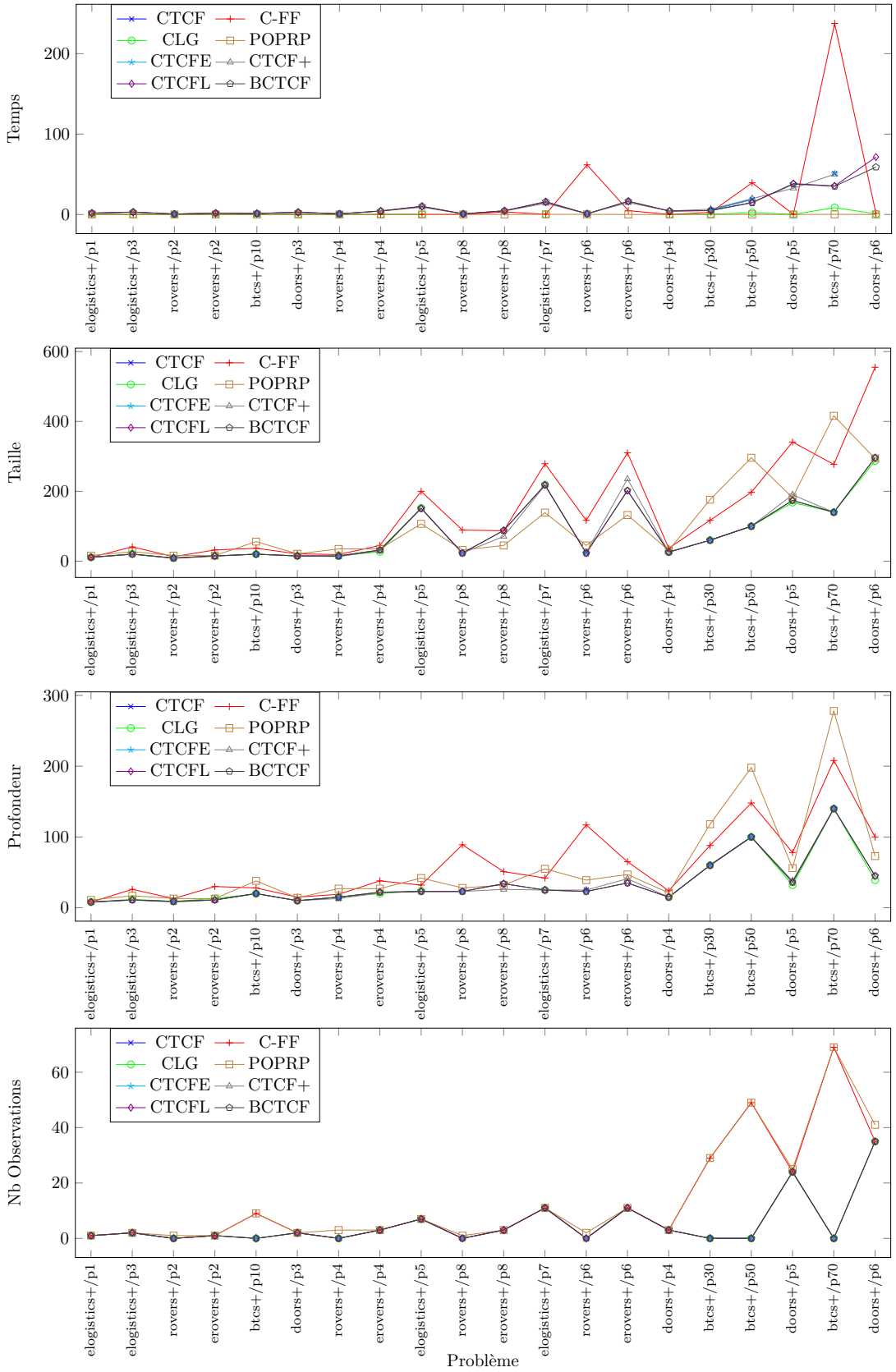

FIGURE 9.4 – Comparaison du temps de calcul, de la taille, de la profondeur, et du nombre d'observations du plan pour les problèmes coûteux

En ce qui concerne le temps de calcul d'une solution, on peut se rendre compte que de manière générale, le temps de calcul d'une solution de BCTCF est sensiblement équivalente à celle de CTCFL pour la majorité des problèmes. De ce fait, BCTCF calcule généralement des plans plus rapidement que les autres versions de l'approche, lui permettant de résoudre plus de problèmes de grande taille dans le temps imparti. Cependant, ce temps de calcul n'est pas suffisamment petit pour rivaliser avec les planificateurs de la littérature en terme de rapidité de calcul d'une solution pour la plupart des problèmes.

Si on considère la taille et la profondeur des plans, BCTCF a exactement les mêmes performances que CTCFL. BCTCF calcule donc des plans de tailles et de profondeurs équivalentes à CTCF, CTCF+ et CTCFE à quelques actions près. De même, BCTCF incorpore autant d'observations que les autres versions de l'approche sur l'ensemble des problèmes étudiés à une exception près. En ce qui concerne la comparaison avec Contingent-FF, CLG et PO-PRP, on peut remarquer que BCTCF incorpore généralement moins ou autant d'observations que les autres planificateurs de la littérature étudiés, à l'exception des problèmes *wumpus* et *wumpus**. Pour ces deux problèmes, la procédure findObservation+ ne peut pas trouver d'observations utiles, la procédure alternative de sélection d'observations intervient donc pour sélectionner les observations candidates. Ces résultats soulignent le fait que la procédure de sélection alternative des observations ne permet pas la limitation du nombre d'observations incorporées au plan, celle-ci n'étant pas guidée vers les observations particulièrement utiles contrairement à la procédure findObservation+ qui repose sur les informations du planificateur conformant. Cette capacité à incorporer moins d'observations que les planificateurs contingents de la littérature est particulièrement visible lorsque des solutions sont calculables sans ajouter d'observation (problèmes *btcs*, *rovers*, *grid*, *logistics*). On peut aussi remarquer que dans la plupart des cas, lorsque BCTCF incorpore autant d'observations que les autres planificateurs de la littérature, celui-ci est compétitif en terme de taille et de profondeur du plan calculé, et ce grâce notamment à la capacité de CPCES* à généralement calculer des branches de petite taille. Lorsque l'on s'intéresse aux problèmes dans lesquels BCTCF incorpore moins d'observations que les planificateurs de la littérature, nous pouvons observer que celui-ci calcule des plans généralement plus petits et plus courts que les planificateurs de la littérature, indiquant une réelle utilité à la limitation du nombre d'observations incorporées au plan. Cette utilité est particulièrement remarquable lorsque l'on résout des problèmes pour lesquels les observations sont modélisées comme plus coûteuses que les actions classiques. Dans notre cas, le coût additionnel des observations étant modélisé comme une action supplémentaire à réaliser pour chaque observation, on peut observer un impact non négligeable de l'ajout d'une observation sur la taille et la profondeur des plans générés pour ce type de problèmes. Cet impact indique que la limitation des observations de notre approche permet bien de se démarquer des planificateurs de la littérature incorporant plus d'observations que notre approche dans ce type de problème.

9.5 Discussion

Les conclusions que nous avons pu obtenir grâce aux résultats expérimentaux nous indiquent plusieurs éléments concernant cette dernière version de l'approche. Bien que notre approche soit complète et qu'elle ne soit pas limitée par la largeur contingente des problèmes à résoudre, celle-ci possède tout de même des limites. Parmi ces limites, nous pouvons citer le temps de calcul d'une solution, qui reste tout de même plus élevé que les planificateurs contingents de la littérature étudiés. De plus, bien que l'approche limite efficacement le nombre d'observations dans les plans calculés pour les problèmes dans lesquels les propositions présentes dans les préconditions des actions peuvent être directement observées, ce n'est pas le cas lorsque l'on tente de résoudre des problèmes dans lesquels les observations sont indirectes, comme le problème du *wumpus*. Ceci est explicable par le fait que le processus de sélection alternative des observations ne permet pas de limiter les observations, la recherche de proposition à observer n'étant pas guidée vers un ensemble de propositions dont l'observation est particulièrement utile contrairement à la procédure findObservation+ qui repose sur les informations du planificateur conformant pour déterminer quelle proposition observer. Dans l'algorithme 13, nous avons choisi de sélectionner l'ensemble de propositions $p_{\gamma,s}$ à observer de manière aléatoire car aucun critère ne nous permet de déterminer quel ensemble de proposition sera la plus prometteuse en terme de limitation du nombre d'observation. Les pistes d'amélioration de l'approche sont évoquées dans le chapitre 11.

9.6 Conclusion

Dans ce chapitre, nous avons développé la dernière version de l'approche nommée Backtracking based ConTingent planner using a ConFormant planner (BCTCF). La première particularité de cette version de l'approche est de permettre la résolution des problèmes dans lesquels les observations sont indirectes. Pour cela, une procédure de sélection d'observation est ajoutée dans laquelle les propositions à observer sont sélectionnées directement en récupérant les propositions permettant de différencier l'état de contre-exemple des autres états de l'état de croyance courant. La deuxième particularité de cette dernière version de l'approche consiste à effectuer un retour sur le choix de l'observation à incorporer au plan en cas d'échec de calcul d'une solution après l'incorporation d'une observation au plan. Pour cela, l'algorithme garde en mémoire les différentes observations candidates à chaque étape de calcul du plan et en sélectionne une différente si la précédente ne mène pas à une solution. Ces améliorations permettent à BCTCF de résoudre plus de problèmes étudiés que les autres versions de l'approche et que Contingent-FF, CLG et PO-PRP. De plus, l'approche est complète et n'est pas limitée aux problèmes de largeur contingente égale à 1, contrairement à CLG et PO-PRP.

De manière générale, le temps de calcul d'une solution de BCTCF est sensiblement équivalente à celle de CTCFL pour la majorité des problèmes. BCTCF calcule donc généralement des plans plus rapidement que les autres versions de l'approche, lui permettant de résoudre plus de problèmes de grande taille dans le temps imparti. BCTCF calcule des plans de tailles et de profondeurs équivalentes aux autres versions de l'approche à quelques actions près. De même, BCTCF incorpore autant d'observations que les autres versions de l'approche sur l'ensemble des problèmes étudiés sauf pour le problème *colorballs*/p21* pour lequel BCTCF ajoute deux observations de moins que CTCFE.

En ce qui concerne la comparaison avec Contingent-FF, CLG et PO-PRP, on peut remarquer que BCTCF met généralement plus de temps à trouver une solution que ces planificateurs, mais incorpore généralement moins ou autant d'observations que ceux-ci à l'exception des problèmes *wumpus* et *wumpus**. Ces résultats moins bons sur ces deux problèmes soulignent le fait que la procédure de sélection alternative des observations ne permet pas la limitation du nombre d'observations incorporées au plan. Néanmoins, cette capacité à incorporer moins d'observations que les planificateurs contingents de la littérature sur la majorité des problèmes est particulièrement visible lorsque des solutions sont calculables sans ajouter d'observation. De plus, lorsque BCTCF incorpore autant d'observations que les autres planificateurs de la littérature, celui-ci est compétitif en terme de taille et de profondeur du plan calculé. Lorsque BCTCF incorpore moins d'observations que les planificateurs de la littérature, celui-ci calcule des plans généralement plus petits et plus courts que les planificateurs de la littérature, soulignant l'utilité de la limitation du nombre d'observations, qui est d'autant plus visible pour les problèmes coûteux. Cet impact indique que la limitation des observations de notre approche permet bien de se démarquer des planificateurs de la littérature incorporant plus d'observations que notre approche dans ce type de problème.

Malgré l'efficacité de notre approche en ce qui concerne la limitation des observations du plan, celle-ci comporte tout de même deux points faibles qu'il est important de souligner : le temps de calcul et la perte d'efficacité de limitation des observations pour les problèmes avec observations indirectes. Dans le prochain chapitre (Chapitre 10) nous conclurons ce manuscrit, puis nous discuterons des différentes perspectives pouvant permettre d'améliorer les points faibles de l'approche dans le chapitre 11.

Troisième partie

Conclusion et perspectives

Chapitre 10

Conclusion

10.1 Contexte

Dans cette thèse, nous nous sommes placés dans le cadre des missions de robotique autonome dont l'environnement comporte de l'incertitude modélisée comme un ensemble d'états initiaux possibles. Dans ce type de problème, deux méthodes de planification sont particulièrement efficaces pour traiter l'incertitude : la planification conformante et la planification contingente. La planification conformante consiste à calculer un plan sous la forme d'une séquence d'action menant au but du problème quel que soit l'état initial, et sans réaliser d'observation. Ces méthodes conformantes sont particulièrement intéressantes dans le contexte de la robotique mobile autonome dans laquelle les observations sont souvent coûteuses en temps et en énergie lorsqu'elles sont exécutées lors de la mission. Cependant, un grand nombre de problèmes ne peuvent pas être résolus sans l'ajout d'observation. La planification contingente permet justement de traiter l'incertitude par le calcul de plans comportant des observations et prenant la forme d'un graphe de décision, mais la planification contingente comporte tout de même une complexité élevée et proportionnelle aux nombres de branches incluses dans le plan. Ce coût potentiel que peut représenter une observation nous a donné l'idée de développer un algorithme de planification contingente limitant le nombre d'observations du plan par l'utilisation d'un planificateur conformant pour le calcul des branches du plan.

10.2 Planificateur contingent limitant le nombre d'observations

Afin d'implémenter cet algorithme, nous avons choisi de faire confiance au planificateur conformant CPCES (Grastien et Scala, 2017). Ce planificateur conformant possède plusieurs caractéristiques intéressantes : celui-ci est capable de traiter des problèmes de largeur conformante supérieure à 1, ou encore de calculer des plans généralement plus courts que les autres planificateurs conformants. De plus, en cas d'échec de calcul d'un plan conformant, CPCES est capable de retourner des informations potentiellement exploitables pour déterminer quelles observations sont nécessaires pour diminuer l'incertitude du problème. Ces informations prennent la forme d'un plan conformant pour une partie des états initiaux possibles et d'un état de contre-exemple à partir duquel ce plan ne mène pas au but du problème.

Cette première approche contingente, nommée ConTingent planner using a ConFormant planner (CTCF), est détaillée dans le chapitre 5 de ce manuscrit. Celle-ci consiste tout d'abord à appeler CPCES à partir du problème initial afin de calculer un plan conformant et de le retourner si un tel plan existe. Dans le cas contraire, le plan et l'état initial de contre-exemple retournés par CPCES sont utilisés afin de déterminer quelle observation est nécessaire et dans quel état de croyance la réaliser afin de discriminer le contre-exemple. Le processus contingent est ensuite appelé afin de trouver un plan contingent ou conformant menant à l'état de croyance dans lequel l'observation peut être réalisée. Deux sous-problèmes prenant en compte les résultats potentiels de l'observation sont générés et utilisés en entrée du processus contingent pour calculer deux plans partant de ces deux sous-problèmes et menant au but du problème initial. Enfin, ces trois plans ainsi que l'action d'observation sont ensuite assemblés

et retournés pour former le plan contingent final permettant de résoudre le problème.

Afin d'évaluer expérimentalement cette première approche, nous avons décidé de la comparer avec des planificateurs contingents de la littérature sur des problèmes séparés en trois classes : des problèmes classiques de la littérature contingente, des problèmes de largeur contingente supérieure à 1, et des problèmes *coûteux*. La largeur contingente d'un problème est définie en termes de nombre maximum de variables dont les valeurs sont initialement inconnues, et dont l'incertitude doit nécessairement être levée afin de résoudre le problème. Les problèmes coûteux sont des problèmes dans lesquels nous avons ajouté une action devant être réalisée avant chaque observation pour rendre les observations plus coûteuses en taille et en profondeur. Le nombre de benchmarks de largeur contingente supérieure à 1 est limité dans la littérature. Par conséquent, nous avons décidé de modifier les problèmes *colorballs* et *wumpus* pour augmenter d'un niveau leur largeur contingente.

Les résultats de cette première approche sur ces ensembles de problèmes ont été publiés dans les conférences ICAS 2020 (Piedade et collab., 2020), JFPDA 2019 et JFPDA 2020. Une présentation de ce travail a aussi été réalisé pour la conférence RO&IA 2020. Ces résultats révèlent que l'approche limite efficacement le nombre d'observations, et calcule des plans de tailles et de profondeurs compétitives par rapport aux planificateurs de la littérature. Néanmoins, cette approche possède des limites : le temps de calcul d'une solution est généralement plus élevé que les planificateurs de la littérature, et l'approche ne peut pas traiter un grand nombre de problèmes. Afin de résoudre ces limites, nous avons développé différentes versions de l'approche résolvant ces limites incrémentalement.

10.3 Réutilisation du plan retourné par CPCES

Dans un premier temps, nous avons voulu améliorer le temps de calcul des solutions en développant une version de l'approche nommée Contingent Planner using a ConFormant planner with Early counter-examples (CTCFE). Celle-ci est détaillée dans le chapitre 6 de ce manuscrit. Dans cette version du planificateur, le plan retourné par CPCES en cas d'échec est réutilisé comme plan menant à l'observation candidate. Pour cela, le contre-exemple n'est plus sélectionné par CPCES de manière aléatoire parmi l'ensemble des contre-exemples. À la place, CPCES sélectionne le contre-exemple dans lequel le plan précédemment calculé échoue le plus tôt. Ce changement permet d'obtenir un début de plan conformant correspondant à la séquence d'actions s'arrêtant juste avant l'action échouant dans le contre-exemple. Ce début de plan conformant est ensuite utilisé afin de déterminer si une observation est applicable à la suite d'une de ses actions et auquel cas ce plan est utilisé comme plan menant à l'observation, évitant le calcul d'un nouveau plan. Les résultats de l'implémentation de CTCFE montrent que cette nouvelle version de l'approche est légèrement plus efficace pour ce qui est du temps de calcul que CTCF pour les problèmes dotés d'une solution contingente. Néanmoins, ce temps de calcul n'est toujours pas assez efficace pour pouvoir résoudre des problèmes de grande taille dans le temps imparti. De plus, cette version de l'approche ne permet pas de résoudre plus de problèmes que l'approche initiale.

10.4 Réduction des contraintes de but

Afin de permettre la résolution des problèmes contingents dans lesquels les observations ne sont pas toujours réalisables sur le chemin d'exécution du plan retourné par CPCES, nous avons développé une nouvelle version de l'approche nommée ConTingent Planner using a ConFormant planner + (CTCF+). Dans cette version présentée dans le chapitre 7, la sélection des observations est modifiée afin de ne plus se baser sur les états de croyance calculés par l'application du plan défaillant retourné par CPCES à l'état de croyance initial pour sélectionner l'observation candidate. À la place, toutes les observations permettant d'observer les préconditions de l'action échouant dans le contre-exemple sont sélectionnées indépendamment du fait qu'elles soient applicables ou non dans un des états de croyance formés par l'application du plan retourné par CPCES. Le but du problème de calcul du plan menant à l'observation est ensuite modélisé comme une disjonction des préconditions de ces observations, et non plus comme un état de croyance précis, réduisant dans la majorité des cas le nombre de contraintes à satisfaire pour atteindre l'observation.

Ces améliorations permettent à CTCF+ de résoudre bien plus de problèmes que les versions précédentes de l'approche, et ce sans perte d'efficacité concernant la limitation des observations, la taille, et la profondeur des plans. Cependant, CTCF+ possède toujours un problème de temps de calcul

empêchant l'approche de résoudre les problèmes de grande taille dans le temps imparti, en majorité à cause de la représentation explicite de l'état de croyance. De plus, comme les versions précédentes, CTCF+ n'est toujours pas capable de résoudre les problèmes dans lesquels les préconditions de l'action du plan échouant retourné par CPCES ne sont pas observables, comme dans le problème *wumpus*. CTCF+ n'est pas capable non plus de changer le choix de l'observation en cas d'échec de calcul d'un plan en considérant cette observation, ce qui empêche l'approche d'être complète.

10.5 Représentation implicite de l'état de croyance

Afin de résoudre le problème de temps de calcul excessif des versions précédentes de l'approche lors de la résolution de problèmes de grande taille, nous avons développé une nouvelle version de l'approche présentée dans le chapitre 8 et nommée ConTingent planner using a ConFormant planner for Large problems (CTCFL). La particularité de cette version de l'approche consiste à ne plus représenter les états de croyance de manière explicite. À la place, ceux-ci sont représentés grâce à l'association de l'état de croyance initial et d'une séquence d'actions et d'observations menant à l'état de croyance courant. Afin de calculer les branches du plan contingent, le plan menant à l'observation insérée dans le plan est transformé en séquence d'actions et d'observations. Les observations présentes dans cette séquence sont transformées en actions artificielles sans effet sur le monde afin d'être considérées par CPCES. L'information sur le résultat des différentes observations présentes dans la séquence d'actions et d'observations originelle est sauvegardée dans un ensemble de contraintes SMT. CPCES utilise cette séquence et cet ensemble de contraintes pour reconstituer l'état de croyance courant dans son processus de vérification de plan. Afin de forcer CPCES à démarrer son calcul de plan par la séquence d'actions et d'observations fournie en entrée, cette séquence est directement ajoutée dans le domaine PDDL transmis à CPCES. Des propositions compteurs sont ajoutées dans les préconditions et les effets des actions du domaine pour déclencher obligatoirement leurs réalisations dans l'ordre de la séquence d'actions avant tout ajout d'actions au plan.

Les résultats de l'implémentation de CTCFL démontrent que cette représentation plus compacte des états de croyance permet effectivement de résoudre la plupart des problèmes de grande taille étudiés grâce à une réduction du temps de manipulation de l'état de croyance. De plus, cette amélioration du temps de calcul pour les problèmes de grande taille ne modifie pas l'efficacité de la limitation du nombre d'observations. La taille et la profondeur des plans générés sont aussi similaires aux versions précédentes de l'approche à quelques actions près. Malgré ces améliorations, CTCFL ne peut pas résoudre les problèmes dans lesquels les préconditions de l'action non applicable du plan retourné par CPCES ne sont pas observables. De plus, CTCFL n'est toujours pas complet, celui-ci n'étant pas capable de changer le choix de l'observation sélectionnée en cas d'échec de calcul d'un plan.

10.6 Sélection alternative des observations et retour sur le choix des observations

Afin de rendre l'approche complète nous avons développé la dernière version de l'approche nommée Backtracking based ConTingent planner using a ConFormant planner (BCTCF). Cette dernière version du planificateur est détaillée dans le chapitre 9 de ce manuscrit. Dans un premier temps, une procédure de sélection d'observations est ajoutée dans laquelle les propositions à observer sont sélectionnées directement en récupérant les propositions permettant de différencier l'état de contre-exemple des autres états de l'état de croyance courant. Cet ajout permet la résolution des problèmes dans lesquels les préconditions de l'action non applicable du plan retourné par CPCES ne sont pas observables. Dans un deuxième temps, cette dernière version de l'approche implémente un retour sur le choix de l'observation à incorporer au plan en cas d'échec de calcul d'une solution. Pour cela, l'algorithme garde en mémoire les différentes observations candidates à chaque étape de calcul du plan et en sélectionne une différente si la précédente ne mène pas à une solution.

Les résultats indiquent que BCTCF résout plus de problèmes que les autres versions de l'approche et que Contingent-FF, CLG et PO-PRP : BCTCF résout 4 problèmes de plus que CTCFL, 9 de plus que CTCF+, 30 de plus que CTCFE, et 31 de plus que CTCF, 1 problème de plus que Contingent-FF, 10 problèmes de plus que CLG et 12 problèmes de plus que PO-PRP (soit 80 problèmes résolus sur

87). De plus, un point important à souligner est que l'approche est complète et n'est pas limitée aux problèmes de largeur contingente égale à 1, contrairement à CLG et PO-PRP.

BCTCF calcule des plans plus rapidement que l'ensemble des autres versions de l'approche pour 17 problèmes, calcule des plans de taille inférieure que CTCF, CTCF+ et CTCFE pour 14 problèmes, et de profondeur inférieure que CTCF, CTCF+ et CTCFE pour 13 problèmes. On peut noter que BCTCF repose sur les mêmes méthodes de calcul de plan que CTCFL, ce qui confère à BCTCF les mêmes performances que CTCFL concernant la taille et la profondeur de plans pour les problèmes que CTCFL peut résoudre. BCTCF incorpore autant d'observations que les autres versions de l'approche sur l'ensemble des problèmes étudiés à l'exception de *colorballs*/p21* pour lequel BCTCF ajoute moins d'observations que CTCFE.

Ces améliorations apportées par BCTCF n'empêchent cependant pas un temps de calcul généralement plus long que les planificateurs de la littérature, BCTCF calculant un plan plus rapidement que Contingent-FF pour 10 problèmes, et plus rapidement que CLG pour 3 problèmes seulement. Cette dernière version de l'approche incorpore moins d'observations dans le plan calculé que Contingent-FF pour 35 problèmes, moins que CLG pour 8 problèmes, et moins que PO-PRP pour 19 problèmes, ce qui indique une limitation efficace du nombre d'observations du plan calculé. Pour les autres problèmes, BCTCF incorpore autant d'observations que ceux-ci à l'exception des problèmes *wumpus* et *wumpus**, ce qui appuie le fait que la procédure de sélection alternative des observations ne permet pas une limitation efficace du nombre d'observations incorporées au plan pour ce genre de problèmes, la procédure ne bénéficiant pas des avantages de l'algorithme FINDOBSERVATION+. En ce qui concerne la taille des plans, BCTCF calcule des plans plus petits que Contingent-FF pour 38 problèmes, plus petits que CLG pour 15 problèmes, et plus petits que PO-PRP pour 32 problèmes. De plus, BCTCF calcule des plans plus courts que Contingent-FF pour 50 problèmes, plus courts que CLG pour 28 problèmes, et plus courts que PO-PRP pour 59 problèmes.

Pour finir, nous pouvons conclure que dans cette thèse, nous avons développé un planificateur complet, qui n'est pas limité aux problèmes de largeur contingente égale à 1, et qui contrebalance son temps de calcul d'une solution par la qualité des plans que celui-ci retourne (d'autant plus que le calcul est réalisé hors-ligne et a donc peu d'impact lorsque celui-ci entre dans le temps limite imposé). L'ensemble des résultats indiquent que le planificateur développé dans ce dernier chapitre limite efficacement le nombre d'observations du plan, ce qui a une conséquence positive sur la taille et la profondeur des plans calculés. Cet intérêt de limiter les observations est d'autant plus visible lorsque l'on s'intéresse aux problèmes dans lesquels les observations sont modélisées comme plus coûteuses que les actions (dans cette thèse concernant le nombre d'actions), ce coût ayant un impact sur la taille et la profondeur des plans.

10.7 Limitations

L'approche élaborée dans cette thèse comporte tout de même certaines limites faisant obstacle à une implémentation sur des cas d'applications réels. Un grand nombre de ces limites repose sur les hypothèses prises dans ce travail. Une de ces hypothèses est celle d'un monde statique, dans lequel aucun évènement imprévu ne peut arriver en cours de mission. De cette hypothèse forte découle l'incapacité de notre approche à anticiper des évènements comme l'arrivée d'un nouvel élément dans l'environnement, comme un objet mouvant. Ces éléments potentiellement présents dans un environnement réel dynamique doivent être pris en compte pour assurer la robustesse des plans générés par notre approche. Une autre hypothèse forte est que les observations sont déterministes et retournent systématiquement une réponse claire sur l'état du fluent observé dans le monde. À l'heure actuelle, aucun mécanisme n'est implémenté dans l'approche dans le cas où une observation ne retourne finalement pas l'un des deux résultats prévus, par exemple lors de pannes capteurs.

Dans le dernier chapitre de ce manuscrit (chapitre 11), nous évoquerons quelques pistes d'amélioration de l'approche, dont certaines consistent à élargir les hypothèses de travail pour permettre le traitement de certaines limites de l'approche.

Chapitre 11

Perspectives

Dans ce dernier chapitre, nous évoquerons dans un premier temps les pistes d'amélioration de l'approche, puis nous nous intéresserons à l'aspect visualisation et simulation de l'exécution du plan. Enfin, nous évoquerons l'extension possible des cas d'applications de l'approche.

Améliorations de l'approche

Temps de calcul

Une piste d'amélioration concernant le temps de calcul des plans peut être d'inclure directement un planificateur conformant dans le code de l'approche au lieu de l'appeler par l'intermédiaire d'écritures de fichiers. La grande majorité du temps de calcul final provient du temps de calcul pris par CPCES pour calculer l'ensemble des branches du plan, par conséquent, une manière de diminuer le temps de calcul serait d'utiliser un autre planificateur conformant que CPCES. Cependant, la plupart des planificateurs conformants de la littérature ont de moins bonnes performances concernant la taille des plans calculés. Remplacer CPCES par un autre planificateur conformant signifierait donc peut être un gain potentiel en temps de calcul, mais surtout une perte potentielle en qualité du plan généré, ce qui n'est pas négligeable.

Une autre piste d'amélioration du temps de calcul pourrait être de paralléliser certains calculs. On peut penser notamment aux calculs des sous-plans partant de l'observation et menant au but du problème. Ces calculs étant indépendants, une parallélisation de ceux-ci pourrait faire gagner un temps de calcul conséquent.

Processus de sélection des observations alternatif

Une piste d'amélioration du processus alternatif de sélection des observations, serait peut-être d'essayer de définir un critère de sélection des propositions pertinentes permettant de limiter les observations, mais cette tâche n'est pas aisée, surtout lorsque les propositions du problème observables et les préconditions de l'action qui échoue ne sont jamais liées, soit par leur présence dans les préconditions d'une action, ou dans ses effets.

Optimisation du plan solution

Une autre piste d'amélioration serait de calculer un plan contingent pour chaque proposition candidate à l'observation afin de retourner le plan qui contient le moins d'observations au lieu de retourner le premier plan contingent solution. Ce processus de sélection apporterait une limitation plus efficace des observations que le processus alternatif actuel ainsi qu'une composante d'optimisation de la qualité du plan supplémentaire, les plans calculés pouvant être sélectionnés par leur taille, leur profondeur, le nombre d'observations, ou encore tout autre critère que l'on juge pertinent. Néanmoins, ces calculs de plans supplémentaires impactent énormément le temps de calcul des plans, ce qu'il ne faut pas négliger. Une première tentative d'implémentation à été réalisée pendant la thèse. Les résultats de cette version prototype sur quelques problèmes laissent présager une réelle possibilité de réduction de la taille ou de la profondeur du plan retourné par la sélection de la solution calculée minimisant l'un

de ces deux critères. Cependant, ces quelques résultats confirment aussi la grande augmentation du temps de calcul de la solution finale. L'implémentation efficace de cette amélioration et les analyses qui en découlent nécessitent une masse de travail supplémentaire considérable. C'est pour cela que cette idée d'amélioration ne figure pas dans cette thèse mais constitue tout de même une piste sérieuse de poursuite du travail de thèse.

Regroupement des branches du plan

Une piste intéressante d'amélioration de l'approche consiste à représenter le plan sous la forme d'un graphe au lieu d'un arbre de décision. Cette amélioration permettrait de réduire la taille et potentiellement la profondeur du plan solution en regroupant les branches du plan similaire. Le regroupement des branches du plan final impliquerait un temps de calcul supplémentaire pour cette opération, mais permettrait d'obtenir un plan prenant moins de place en mémoire, facilitant son embarquabilité.

Visualisation de l'exécution du plan

Pendant cette thèse j'ai pu travailler avec Virgile de la Rochefoucauld dans le cadre de son stage à l'ONERA. Pendant son stage, Virgile a réalisé un premier environnement de simulation de mission de robotique autonome permettant de visualiser le comportement du robot pendant une mission. Pendant cette collaboration, mon travail a consisté à développer les procédures de lecture et écriture du plan retourné par l'approche en langage Yaml afin que celui-ci puisse être exécuté dans le simulateur. Une première visualisation du comportement d'un robot dans une mission simple consistant à prendre en photo un objectif à été réalisée. Cette première visualisation de l'environnement encourageante est illustrée par la figure 11.1.

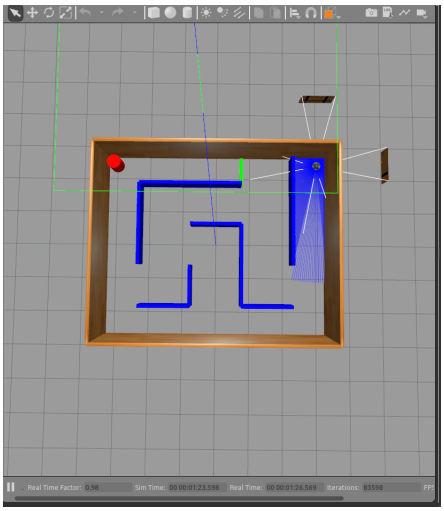

FIGURE 11.1 – Environnement de simulation développé par Virgile de la Rochefoucauld

Une des perspectives de cette thèse pourrait être de poursuivre ce travail de simulation de mission développé par Virgile de la Rochefoucauld afin de pouvoir générer facilement des environnements de simulation permettant la visualisation de l'exécution du plan retourné par notre approche sur de nombreux problèmes. Ces visualisations pourraient apporter un point de vue intéressant sur la manière dont le robot se comporte lors de l'exécution du plan généré par notre approche. De plus, il serait intéressant d'ajouter à cette plateforme de simulation une fonctionnalité permettant de générer directement un problème PDDL, de lancer le planificateur sur ce problème, et de visualiser automatiquement l'exécution du plan dans l'environnement de visualisation.

Analyse de l'approche sur un cas d'application réel

Pendant cette thèse, les différentes versions de l'approche développées ont été évaluées sur un bon nombre des benchmarks académiques. Cependant, on peut imaginer un grand nombre de scénarios supplémentaires permettant d'évaluer notre approche sur un éventail plus large de problèmes. De plus, une évaluation de l'approche sur un cas réel de mission de robotique autonome, comme par exemple une mission d'exploration d'environnement, serait intéressante pour déterminer le comportement de l'exécution du plan sur un robot réel, ou encore mesurer des gains réels en temps et en énergie économisés par la limitation des observations. Comme nous l'avons évoqué dans le chapitre 10, un travail supplémentaire doit être réalisé afin d'appliquer notre approche à un cas réel de manière plus robuste. Dans un premier temps, un processus de replanification en ligne peut être intégré à l'architecture dans le cas où une observation ne retourne pas les valeurs escomptées en raison d'une panne capteur, par exemple. Pour cela, il suffirait , par exemple, de recommencer le calcul d'un plan à partir de l'ensemble d'état de croyance courant en supprimant l'action d'observation problématique du domaine de planification. L'adaptation de l'approche à une nouvelle incertitude en cours de mission, comme l'apparition d'un nouvel obstacle peut aussi potentiellement être réalisée par un processus de replanification en ligne dans lequel les positions potentielles du nouvel obstacle peuvent, par exemple, être ajoutées au problème de planification avant de calculer un nouveau plan à partir de ces nouvelles informations.

Quatrième partie

Annexes

Annexe A

Description des benchmarks

A.1 Description des benchmarks de la littérature

Blocks

Dans ce domaine, il existe un certain nombre de cubes devant être empilés les uns sur les autres selon un ordre précis définissant le but du problème. Dans l'état initial du problème, un cube peut être posé sur la table ou sur un autre cube. Les différentes actions du domaine vont simplement être une action de déplacement d'un cube sur la table ou sur un autre bloc, en sachant que seul un bloc libre, c'est-à-dire sans aucun bloc posé au dessus de lui, peut être déplacé. Les incertitudes de ce domaine reposent sur le fait de savoir si un cube est libre, sur la table, ou bien posé sur un bloc bien précis. Chacune de ces incertitudes peut être levée par une observation.

Btcs et Ebtcs

Dans le domaine Btcs, il existe une ou plusieurs bombes cachées dans des paquets, ainsi qu'un certain nombre de toilettes dans lesquelles les paquets peuvent être jetés à condition que ces toilettes ne soient pas bouchées. Jeter un paquet contenant une bombe dans les toilettes a pour conséquence de désamorcer la bombe, mais aussi de boucher les toilettes. Deux actions sont disponibles dans ce domaine. La première permet de jeter un paquet dans les toilettes choisies à condition que celles-ci ne soient pas bouchées, et la deuxième permet de déboucher des toilettes en tirant la chasse. L'incertitude de ce domaine porte sur le fait de savoir si une bombe est présente dans un certain paquet ou non. Seule cette information peut être obtenue via une observation. Le domaine Ebtcs est une variante de btcs dans laquelle il est nécessaire de réaliser des observations, l'information sur la présence de bombes dans un paquet devenant une précondition à l'exécution de l'action de jeter un paquet.

Grid et Egrid

Dans le domaine Grid, un robot évolue dans une grille contenant des clés et des portes avec serrure. Les clés sont disséminées dans différentes cases de la grille et peuvent servir à ouvrir une porte fermée à condition que la forme de la clé soit celle de la porte en question. Le but de ce problème va être d'amener le robot ou les clés dans des cases bien précises de la grille, en sachant que le robot ne peut traverser les portes fermées. L'incertitude de ce problème réside dans le fait de ne pas connaître la forme de la serrure des portes au préalable, il existe donc une seule observation permettant d'observer la serrure afin de lever cette incertitude. Le domaine Egrid est une variante de Grid dans laquelle l'information sur la forme de la serrure est nécessaire au déverrouillage des portes alors que dans le domaine Grid, l'effet d'essayer d'ouvrir une serrure avec la mauvaise clé n'aura juste aucun effet, ce qui force le robot à effectuer des observations de la serrure pour obtenir la forme de la serrure.

Rovers et Erovers

Le domaine Rovers est, sans doute, le domaine de ces benchmarks le plus ressemblant à une application réelle de mission de robot d'exploration autonome. Dans ce domaine, un robot évoluant dans

une grille doit prendre des images d'un objectif, collecter un rocher, ou encore collecter des échantillons de sol, et envoyer les informations recueillies lors de ces collectes ou de ces prises d'image à une base distante. L'incertitude de ce domaine réside dans le fait de ne pas savoir si un objectif est visible ou non depuis une position de la grille, si un rocher est dans une position ou une autre, et si un échantillon de sol est bien dans une position ou non. Pour chacune de ces incertitudes il existe une observation. Le domaine Erovers est une variante de Rovers dans laquelle il est nécessaire d'être sûr que le rocher ou l'échantillon de sol soient à la bonne position avant de les récolter, ou bien que l'objectif est bien visible depuis la position courante avant de prendre une photo, forçant l'utilisation d'observations pour lever l'incertitude sur ces informations, alors que dans Rovers, la récolte d'un rocher ou un échantillon de sol, ou la prise d'une image d'un endroit ou l'objectif n'est pas visible est possible mais n'aura aucun effet.

Logistics et Elogistics

Dans le domaine Logistics, il existe un ou plusieurs paquets pouvant être chargés et déchargés d'un camion pouvant rouler d'un endroit d'une ville à son aéroport et inversement, ou chargés et déchargés d'un avion pouvant voler d'une ville à une autre. Le but de ce problème est d'acheminer un paquet d'un endroit à un autre, ces endroits pouvant ne pas se situer dans la même ville. L'incertitude de ce problème réside dans le fait de ne pas savoir si un paquet est bien dans un lieu ou non, et de ne pas savoir si un paquet se trouve dans un aéroport ou non. Le domaine Elogistics est une variante de Logistics dans laquelle il est nécessaire de savoir si le paquet est bien dans la position de l'avion ou du camion avant chargement, ce qui force l'ajout d'observations pour lever les incertitudes relatives à la position du paquet.

Doors

Dans le domaine Doors, l'environnement est constitué d'une grille de positions dans laquelle des portes sont ouvertes ou fermées. Le but du problème est de déplacer un agent d'une position à une autre. L'incertitude de ce problème réside dans le fait de ne pas savoir si une porte est ouverte ou non.

Colorballs

Dans le domaine Colorballs, l'environnement est constitué d'une grille de positions dans laquelle il existe une ou plusieurs balles de couleurs ainsi que des poubelles colorées. Le but de ce problème est de se déplacer dans la grille afin de jeter chaque balle dans la poubelle de couleur correspondante. L'incertitude de ce problème concerne la couleur et l'emplacement des balles dans la grille.

Wumpus

Dans ce domaine, un agent doit ramasser un trésor dans une grille de positions contenant un ou plusieurs monstres ainsi que un ou plusieurs trous. Le but du problème est de récolter le trésor en évitant les monstres et les trous, dont la position est incertaine. Contrairement aux autres problèmes décrits dans ce manuscrit, les observations ne sont pas directes, on ne peut pas observer la présence de monstre et de trou dans une case directement, mais par le biais des odeurs présentes dans les cases adjacentes aux monstres, et par le biais du vent présent dans les cases adjacentes aux trous.

A.2 Description des benchmarks développés pour cette thèse

Afin de tester les différentes approches développées dans cette thèse sur un éventail plus large de problèmes, nous avons décidé de créer un problème dans lequel il est nécessaire de réaliser un détour par rapport au chemin menant directement au but pour effectuer les observations permettant de lever l'incertitude. Nous avons nommé ce problème *laboratory*.

laboratory

L'environnement de *laboratory* est constitué de 5 salles : la salle *init*, la salle *lab*, la salle *goal*, la salle *room* et la salle *control room*. Les salles sont connectées de la manière décrite par la figure A.1.

L'incertitude repose sur le fait de savoir si la porte ouverte est la porte rouge ou la porte verte. Le but du problème est de mener un agent de la salle *init* à la salle *goal*.

Nous avons développé deux domaines *laboratory*. Dans le premier domaine, nommé *p1*, l'observation de la porte rouge se fait depuis la salle *room*, tandis que l'observation de la porte verte se fait depuis la salle *init*, forçant l'agent à s'éloigner de son chemin direct vers le but afin de lever l'incertitude sur l'ouverture de la porte située devant lui. Dans le domaine nommé *p2*, l'observation de l'ouverture de la porte rouge et de la porte verte doit être réalisée depuis la salle *control room* rendant le détour encore plus grand pour l'agent afin de réaliser l'observation.

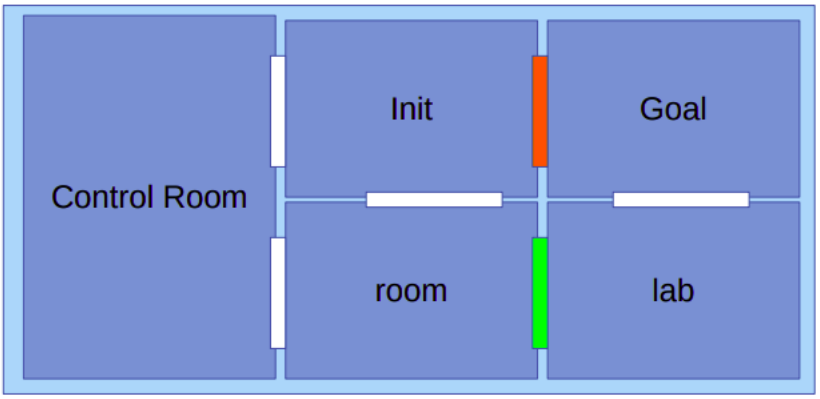

FIGURE A.1 – Illustration de l'environnement de *laboratory*

A.3 Description des benchmarks coûteux

La version de FF utilisé par CPCES n'est pas capable de gérer les contraintes numériques. Il n'est donc pas possible de fixer un coût à une observation sous forme numérique. Afin de tout de même considérer une observation comme coûteuse, nous avons eu l'idée de rajouter une action obligatoire avant de réaliser une observation. Pour cela, nous ajoutons une proposition aux préconditions des observations, et créons une action ayant pour effet cette proposition.

btcs+

Pour transformer le problème *btcs* en problème coûteux, nous avons modifié le domaine PDDL en ajoutant la proposition (*opened* ?*p*) à l'ensemble de préconditions initialement vide de l'observation *senseP* et en ajoutant l'action *open* ne possédant pas de précondition et produisant en effet la proposition (*opened* ?*p*). Cette action supplémentaire représente potentiellement l'ouverture de la boîte avant d'observer son contenu. La figure A.2 illustre ces modifications.

```
(:action senseP
  :parameters (?p - package ?b - bomb)
  :precondition (opened ?p)
  :observe (in ?p ?b)
)

(:action open
  :parameters (?p - package)
  :effect (opened ?p)
)
```

FIGURE A.2 – Modifications du domaine PDDL de *btcs+*

rovers+ et erovers+

Pour transformer les problèmes *rovers* et *erovers* en problèmes coûteux, nous avons modifié le domaine PDDL afin d'ajouter respectivement les propositions (*soilsensorready*), (*vissensorready*) et (*rocksensorready*) à l'ensemble de préconditions des observations *sense_soil*, *sense_vis*, et *sense_rock* indiquant que chaque capteur doit être prêt avant d'être utilisé. Les actions *runsoilsensor*, *runvissensor* et *runrocksensor* sont ajoutées et ont pour effet de générer les propositions (*soilsensorready*), (*vissensorready*) et (*rocksensorready*) permettant de réaliser les observations. Ces actions peuvent représenter l'action d'allumage d'un capteur avant son utilisation. Afin de rendre le problème plus coûteux encore, nous avons ajouté respectivement les propositions de précondition (*openedrocksensor*), (*openedsoilsensor*) et (*openedvissensor*) à ces actions. Ces propositions sont générées par les actions *openvissensor*, *opensoilsensor* et *openrocksensor* représentant les actions d'ouverture des capteurs avant leur allumage. Une difficulté supplémentaire est ajoutée, l'action d'ouverture du capteur *soilsensor* fermant le capteur *rocksensor* et inversement. La figure A.3 illustre ces modifications.

```
(:action runsoilsensor
 :parameters ()
 :precondition (openedsoilsensor)
 :effect (soilsensorready))

(:action runvissensor
 :parameters ()
 :precondition (openedvissensor)
 :effect (vissensorready))

(:action runrocksensor
 :parameters ()
 :precondition (openedrocksensor)
 :effect (rocksensorready))

(:action openvissensor
 :parameters ()
 :precondition (not(openedvissensor))
 :effect (openedvissensor))

(:action opensoilsensor
 :parameters ()
 :precondition (not(openedsoilsensor))
 :effect (and (openedsoilsensor) (not(openedrocksensor))))

(:action openrocksensor
 :parameters ()
 :precondition (not(openedrocksensor))
 :effect (and(openedrocksensor) (not(openedsoilsensor))))

(:action sense_vis
 :parameters (?x - rover ?t - objective ?z - waypoint )
 :precondition (and (ata ?x ?z) (vissensorready))
 :observe (visible_from ?t ?z))

(:action sense_rock
 :parameters (?x - rover ?t - objective ?z - waypoint )
 :precondition (and (ata ?x ?z) (rocksensorready))
 :observe (at_rock_sample ?z))

(:action sense_soil
 :parameters (?x - rover ?t - objective ?z - waypoint )
 :precondition (and (ata ?x ?z)(soilsensorready) )
 :observe (at_soil_sample ?z))
```

FIGURE A.3 – Modifications du domaine PDDL de *rovers+* et *erovers+*

elogistics+

Pour transformer le problème *elogistics* en problème coûteux, nous avons modifié le domaine PDDL afin d'ajouter respectivement les propositions (*openedhangar* ?*loc*), (*openedhangarair* ?*loc*) et (*openedhangarairplane* ?*loc*) aux observations *sensepackageloct*, *sensepackageapt* et *sensepackageapa* représentant le fait que les paquets sont contenus dans un hangar qui doit être ouvert avant d'observer si le paquet est bien la. Les actions d'ouverture des hangars des différents lieux *openhangartruckloc*, *openhangartruckair*, et *openhangarairplane* sont ajoutées afin de générer les propositions nécessaires à la réalisation des observations. Ces modifications sont illistrées par la figure A.4.

```
(:action sensepackageloct
 :parameters (?obj - obj ?loc - location ?truck - truck)
 :precondition (and(attl ?truck ?loc)(openedhangar ?loc))
 :observe (atol ?obj ?loc))

(:action openhangartruckloc
:parameters (?truck - truck ?loc - location)
:precondition (attl ?truck ?loc)
:effect (openedhangar ?loc))

(:action sensepackageapt
 :parameters (?obj - obj ?loc - airport ?truck - truck)
 :precondition (and (atta ?truck ?loc) (openedhangarair ?loc))
 :observe (atoa ?obj ?loc))

(:action openhangartruckair
:parameters (?truck - truck ?loc - airport)
:precondition (atta ?truck ?loc)
:effect (openedhangarair ?loc))

(:action sensepackageapa
 :parameters (?obj - obj ?loc - airport ?airplane - airplane)
 :precondition (and (ataa ?airplane ?loc) (openedhangarairplane ?loc) )
 :observe (atoa ?obj ?loc))

(:action openhangarairplane
:parameters (?airplane - airplane ?loc - airport)
:precondition (ataa ?airplane ?loc)
:effect (openedhangarairplane ?loc))
```

FIGURE A.4 – Modifications du domaine PDDL de *elogistics+*

doors+

Afin de transformer le problème *doors* en problème coûteux, nous avons ajouté l'action *runsensor* représentant l'allumage du capteur permettant l'observation et produisant la proposition (*sensorready* ?*i*). Cette proposition est ajoutée en précondition de l'action $sense - door$ afin que l'action *runsensor* soit obligatoirement réalisé avant chaque observation. Ces modifications sont illustrées par la figure A.5.

```
(:action sense-door
   :parameters (?i  ?j )
   :precondition (and (ata ?i) (adj ?i ?j) (sensorready ?i))
   :observe (opened ?j) )

 (:action runsensor
   :parameters (?i)
   :precondition (ata ?i)
   :effect (sensorready ?i) )
```

FIGURE A.5 – Modifications du domaine PDDL de *doors+*

A.4 Description des benchmarks modifiés pour avoir une largeur contingente > 1

Dans la littérature, peu de problèmes ont une largeur contingente supérieure à 1. Afin d'évaluer plus efficacement notre approche sur les benchmarks de cette classe, nous avons décidé de modifier des problèmes de la littérature afin d'augmenter leur largeur contingente d'un niveau, permettant d'obtenir une largeur contingente supérieure à 1.

colorballs*

Afin d'augmenter la largeur contingente du problème *colorballs*, nous avons ajouté au domaine PDDL une observation *observegarbage* permettant d'observer la position d'une poubelle. Cette modification est illustrée par la figure A.6. L'incertitude sur la position des poubelles est ensuite ajoutée au problème sous la forme d'un *oneof* sur la proposition de position de la poubelle concernée. Les modifications du problème sont illustrées par la figure A.7.

```
(:action observegarbage
     :parameters (?t ?pos)
     :precondition (ata ?pos)
     :observe (garbageata ?t ?pos))
```

FIGURE A.6 – Modification du domaine PDDL de *colorballs*

```
(oneof
  (garbageata t1 p11)
  (garbageata t1 p12)
  (garbageata t1 p21)
  )
  (unknown(garbageata t1 p11))
  (unknown(garbageata t1 p12))
  (unknown(garbageata t1 p21))
```

FIGURE A.7 – Modification du problème PDDL de *colorballs*

wumpus*

Afin d'augmenter la largeur contingente du problème *wumpus*, nous avons ajouté une contrainte *oneof* dans le problème pour chaque case dans laquelle une incertitude concerne la présence du wumpus ou d'un fossé. Grâce à cette contrainte, il est nécessaire d'observer deux fois chaque case pour vérifier si celle-ci est sans danger. Ces modifications sont illustrées dans la figure A.8.

```
   (oneof
(pitat p23)
(wumpusat p23)
(safe p23)
)

(oneof
  (pitat p32)
  (wumpusat p32)
  (safe p32)
)
```

FIGURE A.8 – Modification du problème PDDL du *wumpus*

Annexe B

Tableaux de résultats

TABLE B.1 – Résultats sur les problèmes conformants et contingents de la littérature de CTCF, Contingent-FF, CLG et PO-PRP. *t* représente le temps de calcul en secondes, *s* représente la taille du plan en nombre d'actions et d'observations, *d* représente la profondeur du plan en nombre d'actions et observations et *o* représente le nombre d'observations au total. NO signifie qu'aucune solution n'est trouvée, TO indique un dépassement du temps imposé, MO signifie un dépassement mémoire, NV signifie que le plan calculé n'est pas valide, DQ signifie que le planificateur est disqualifié à cause d'un plan non valide dans ce benchmark, et PR signifie que le planificateur n'a pas réussi à lire le problème PDDL.

Problem	CTCF				CT-FF				CLG				PO-PRP			
	t	s	d	o	t	s	d	o	t	s	d	o	t	s	d	o
btcs/p10	1.25	19	19	0	0.01	19	10	9	0.00	20	20	0	0.00	46	28	9
btcs/p30	3.75	59	59	0	0.63	59	30	29	0.40	60	60	0	0.02	146	88	29
btcs/p50	10.40	99	99	0	8.40	99	50	49	2.30	100	100	0	0.08	246	148	49
btcs/p70	27.32	139	139	0	51.61	139	70	69	8.90	140	140	0	0.14	346	208	69
grid/p2	0.76	9	9	0	0.04	9	9	0	0.08	9	9	0	0.04	9	9	0
grid/p3	0.92	21	21	0	9.42	174	43	15	0.22	114	30	7	0.04	38	29	3
grid/p4	1.83	36	36	0	206.11	464	68	32	2.72	872	51	31	0.28	109	53	7
grid/p5	1.93	26	26	0	3.56	356	70	17	0.64	212	40	7	0.1	64	42	3
rovers/p2	0.60	8	8	0	0.01	13	10	1	0.00	11	7	1	0.00	14	11	1
rovers/p4	0.82	13	13	0	0.00	23	14	3	0.00	24	14	3	0.02	29	21	3
logistics/p1	0.57	9	9	0	0.02	10	7	1	0.00	9	9	0	NV			
logistics/p3	0.68	14	14	0	0.03	18	8	2	0.00	14	14	0	DQ			
logistics/p5	0.77	29	29	0	0.60	172	26	7	0.02	29	29	0	DQ			
logistics/p7	0.90	31	31	0	0.25	247	27	11	0.02	31	31	0	DQ			
blocks/p3	1.08	6	4	1	0.00	6	4	1	0.04	6	4	1	0.00	8	6	1
blocks/p7	6.46	89	16	7	0.07	55	9	7	1.88	55	9	7	0.14	56	29	7
blocks/p11	8.27	169	30	7	0.40	117	18	7	16.62	115	18	7	2.02	104	40	7
blocks/p15	7.48	244	41	7	3.49	163	25	7	88.72	157	22	7	5.32	108	49	7
ebtcs/p10	6.76	20	11	9	0.04	29	11	9	0.02	29	11	9	PR			
ebtcs/p30	24.45	60	31	29	0.28	89	31	29	0.60	89	31	29	PR			
ebtcs/p50	60.42	100	51	49	4.09	149	51	49	5.42	149	51	49	0.04	246	148	49
ebtcs/p70	160.71	140	71	69	23.82	209	71	69	22.6	209	71	69	0.14	346	208	69
egrid/p3	NO				316.23	121	39	7	0.22	111	28	7	0.04	38	29	3
egrid/p4	14.73	233	42	7	4.82	658	55	46	3.20	831	56	29	0.28	108	52	7
egrid/p5	NO				3.25	957	70	35	0.66	208	40	7	0.12	64	42	3
erovers/p2	1.55	11	9	1	0.01	13	10	1	0.00	11	9	1	0.02	14	11	1
erovers/p4	3.84	22	15	3	0.03	23	14	3	0.00	20	13	3	0.00	29	21	3
erovers/p20	11.27	57	32	6	0.04	56	32	8	0.08	58	36	6	NO			
erovers/p40	51.58	123	68	14	0.19	115	60	18	0.46	127	80	14	NO			
erovers/p60	251.42	274	132	25	1.78	267	125	30	1.94	243	127	25	NO			
erovers/p120	TO				15.02	416	234	43	6.14	339	191	35	NO			
erovers/p200	163.00	65	45	5	MO				1.68	57	37	5	0.15	56	47	5
erovers/p250	253.27	64	43	5	MO				2.46	65	43	5	0.18	64	54	5
erovers/p300	363.51	65	41	5	MO				3.52	71	42	5	0.22	56	48	5
erovers/p500	TO				MO				TO				0.4	71	62	6
elogistics/p1	1.18	10	7	1	0.03	10	7	1	0.00	10	7	1	0.00	15	10	1
elogistics/p3	1.97	18	9	2	0.03	18	8	2	0.00	18	8	2	0.02	25	15	2
elogistics/p5	6.98	143	23	7	0.16	172	26	7	0.04	147	21	7	0.04	103	40	7
elogistics/p7	9.60	210	22	11	0.15	247	26	11	0.08	210	22	11	0.06	114	43	11
colorballs/p21	NO				0.02	28	10	7	0.06	26	9	7	DQ			
colorballs/p31	NO				0.08	98	26	20	0.20	95	21	19	DQ			
colorballs/p41	79.47	368	61	47	1.0	281	49	47	0.76	295	52	47	NV			
colorballs/p51	NO				14.54	589	95	85	2.58	586	65	83	DQ			
laboratory/p1	NO				0.00	5	4	1	0.00	5	4	1	0.00	9	6	1
laboratory/p2	NO				0.00	7	5	1	0.00	7	5	1	0.00	10	7	1
wumpus/p3	NO				0.00	19	11	2	0.00	38	12	6	0.00	18	13	2
wumpus/p4	NO				1.49	117	25	15	0.06	166	21	24	0.06	100	40	11
wumpus/p5	NO				144.45	303	41	32	0.34	754	41	101	0.16	264	59	34
wumpus/p6	NO				TO				1.20	1599	44	210	0.92	600	108	67

TABLE B.2 – Résultats sur les problèmes conformants et contingents de la littérature de CTCFE, CTCF+, CTCFL et BCTCF. t représente le temps de calcul en secondes, s représente la taille du plan en nombre d'actions et d'observations, d représente la profondeur du plan en nombre d'actions et observations et o représente le nombre d'observations au total. NO signifie qu'aucune solution n'est trouvée, TO indique un dépassement du temps imposé.

Problem	CTCFE				CTCF+				CTCFL				BCTCF			
	t	s	d	o	t	s	d	o	t	s	d	o	t	s	d	o
btcs/p10	1.19	19	19	0	1.18	19	19	0	1.12	19	19	0	1.15	19	19	0
btcs/p30	3.80	59	59	0	3.79	59	59	0	3.13	59	59	0	3.21	59	59	0
btcs/p50	10.60	99	99	0	10.01	99	99	0	5.99	99	99	0	6.13	99	99	0
btcs/p70	28.63	139	139	0	26.75	139	139	0	10.75	139	139	0	10.50	139	139	0
grid/p2	0.75	9	9	0	0.69	9	9	0	0.48	9	9	0	0.45	9	9	0
grid/p3	1.04	19	19	0	0.86	21	21	0	0.76	21	21	0	0.72	21	21	0
grid/p4	2.21	36	36	0	1.69	36	36	0	1.02	37	37	0	0.98	37	37	0
grid/p5	1.87	31	31	0	1.77	26	26	0	0.83	26	26	0	0.77	26	26	0
rovers/p2	0.60	8	8	0	0.57	8	8	0	0.60	9	9	0	0.57	9	9	0
rovers/p4	0.82	13	13	0	0.78	13	13	0	0.81	15	15	0	0.78	15	15	0
logistics/p1	0.57	9	9	0	0.55	9	9	0	0.58	9	9	0	0.53	9	9	0
logistics/p3	0.69	14	14	0	0.65	14	14	0	0.67	14	14	0	0.65	14	14	0
logistics/p5	0.67	29	29	0	0.73	29	29	0	0.75	29	29	0	0.71	29	29	0
logistics/p7	1.06	31	31	0	0.84	31	31	0	0.85	31	31	0	0.82	31	31	0
blocks/p3	1.02	6	4	1	1.18	6	4	1	1.63	6	4	1	1.55	6	4	1
blocks/p7	5.44	88	16	7	6.26	89	16	7	9.42	90	16	7	8.92	90	16	7
blocks/p11	6.12	174	30	7	6.88	161	26	7	10.24	179	29	7	9.63	179	29	7
blocks/p15	7.06	244	41	7	7.52	244	41	7	11.16	230	35	7	10.47	230	35	7
ebtcs/p10	6.17	20	11	9	7.37	20	11	9	11.73	20	11	9	10.98	20	11	9
ebtcs/p30	22.12	60	31	29	25.82	60	31	29	40.21	60	31	29	39.92	60	31	29
ebtcs/p50	55.35	100	51	49	61.64	100	51	49	83.43	100	51	49	83.59	100	51	49
ebtcs/p70	150.41	140	71	69	158.53	140	71	69	155.32	140	71	69	155.56	140	71	69
egrid/p3	NO				4.93	60	23	3	4.15	62	25	3	4.16	62	25	3
egrid/p4	NO				14.41	254	49	7	13.48	244	49	7	13.55	244	49	7
egrid/p5	NO				6.92	96	28	3	4.74	96	28	3	4.76	96	28	3
erovers/p2	1.11	11	9	1	1.66	11	8	1	1.66	11	7	1	1.66	11	7	1
erovers/p4	2.50	22	15	3	4.06	22	12	3	4.15	22	12	3	4.15	22	12	3
erovers/p20	6.77	56	31	6	10.50	57	32	6	9.21	56	33	6	9.23	56	33	6
erovers/p40	30.53	124	68	14	47.59	123	68	14	27.81	124	68	14	27.87	124	68	14
erovers/p60	145.60	271	130	25	240.79	274	132	25	85.42	278	135	25	85.49	278	135	25
erovers/p120	TO				TO				226.33	394	212	35	226.49	394	212	35
erovers/p200	143.81	65	45	5	157.37	65	40	5	14.40	60	34	5	14.47	60	34	5
erovers/p250	226.97	63	43	5	245.28	64	38	5	17.40	65	40	5	17.46	65	40	5
erovers/p300	331.18	66	43	5	351.92	65	36	5	19.34	64	32	5	19.36	64	32	5
erovers/p500	TO				TO				35.87	98	47	6	35.64	98	47	6
elogistics/p1	1.09	10	7	1	1.52	10	7	1	1.63	10	7	1	1.63	10	7	1
elogistics/p3	1.83	18	8	2	2.64	18	9	2	2.84	18	9	2	2.86	18	9	2
elogistics/p5	5.72	133	23	7	8.49	147	21	7	9.27	147	21	7	9.28	147	21	7
elogistics/p7	8.81	204	24	11	13.02	210	22	11	14.32	210	22	11	14.31	210	22	11
colorballs/p21	NO				7.69	27	9	7	8.61	26	8	7	8.59	26	8	7
colorballs/p31	NO				21.15	91	19	19	23.75	91	19	19	23.83	91	19	19
colorballs/p41	NO				69.11	274	43	47	66.11	270	38	47	66.14	270	38	47
colorballs/p51	TO				315.37	581	89	83	152.30	561	67	83	151.73	561	67	83
laboratory/p1	NO				1.51	6	4	1	1.55	6	4	1	1.52	6	4	1
laboratory/p2	NO				1.54	7	5	1	1.63	7	5	1	1.55	7	5	1
wumpus/p3	NO				NO				NO				4.70	27	11	3
wumpus/p4	NO				NO				NO				253.45	168	38	18
wumpus/p5	NO				NO				NO				TO			
wumpus/p6	NO				NO				NO				TO			

TABLE B.3 – Résultats sur les problèmes contingents de largeur > 1 de CTCF, Contingent-FF, CLG et PO-PRP. t représente le temps de calcul en secondes, s représente la taille du plan en nombre d'actions et d'observations, d représente la profondeur du plan en nombre d'actions et observations et o représente le nombre d'observations au total. NO signifie qu'aucune solution n'est trouvée, TO indique un dépassement du temps imposé, NV signifie que le plan calculé n'est pas valide et DQ signifie que le planificateur est disqualifié à cause d'un plan non valide dans ce benchmark.

Problèmes	CTCF				CT-FF				CLG				PO-PRP			
	t	s	d	o	t	s	d	o	t	s	d	o	t	s	d	o
rovers/p6	1.02	23	23	0	0.12	448	66	11	NO				0.04	110	39	11
rovers/p8	0.80	23	23	0	0.04	170	83	3	NO				0.04	39	28	3
erovers/p6	NO				0.12	346	48	11	NO				0.04	110	39	11
erovers/p8	3.35	62	21	3	0.03	95	36	3	NO				0.02	39	28	3
doors/p3	2.60	13	8	2	0.02	14	9	3	0.00	13	8	2	0.00	19	12	2
doors/p4	4.39	26	14	3	0.04	26	14	5	0.02	23	12	3	0.00	29	18	3
doors/p5	NO				0.37	182	37	39	0.14	144	24	24	0.04	152	44	24
doors/p6	TO				10.86	310	46	59	0.52	252	29	35	0.06	218	55	35
doors/p9	TO				TO				322.26	46024	95	6560	5.02	37016	168	6560
doors/p11	TO				TO				TO				MO			
wumpus*/p3	NO				0.04	19	11	2	NO				0.02	18	13	2
wumpus*/p4	NO				0.49	91	25	11	NO				0.04	93	30	11
wumpus*/p5	NO				71.7	411	39	40	NO				0.22	298	62	40
wumpus*/p6	NO				TO				0.78	921	42	122	0.62	526	89	70
colorballs*/p21	NO				0.07	45	15	13	NO				DQ			
colorballs*/p31	NO				0.26	133	26	32	NO				DQ			
colorballs*/p41	NO				155.86	405	56	79	NO				DQ			
colorballs*/p51	NO				208.75	1091	87	179	NO				NV			

TABLE B.4 – Résultats sur les problèmes contingents de largeur > 1 de CTCFE, CTCF+, CTCFL et BCTCF. t représente le temps de calcul en secondes, s représente la taille du plan en nombre d'actions et d'observations, d représente la profondeur du plan en nombre d'actions et observations et o représente le nombre d'observations au total. NO signifie qu'aucune solution n'est trouvée, TO indique un dépassement du temps imposé.

Problem	CTCFE				CTCF+				CTCFL				BCTCF			
	t	s	d	o	t	s	d	o	t	s	d	o	t	s	d	o
rovers/p6	1.18	23	23	0	0.95	23	23	0	0.89	23	23	0	0.83	23	23	0
rovers/p8	0.82	23	23	0	0.76	23	23	0	0.79	23	23	0	0.73	23	23	0
erovers/p6	8.69	125	30	11	14.23	192	26	11	14.87	168	21	11	14.89	168	21	11
erovers/p8	2.74	55	21	3	4.08	63	20	3	4.37	68	22	3	4.38	68	22	3
doors/p3	1.69	13	8	2	2.71	13	8	2	2.96	13	8	2	2.73	13	8	2
doors/p4	2.74	24	12	3	4.52	26	14	3	4.35	23	12	3	4.09	23	12	3
doors/p5	19.31	158	30	24	32.72	164	34	24	33.16	150	28	24	32.50	150	28	24
doors/p6	TO				TO				49.95	261	35	35	49.81	261	35	35
doors/p9	TO				TO				TO				TO			
doors/p11	TO				TO				TO				TO			
wumpus*/p3	NO				NO				NO				2.89	17	9	2
wumpus*/p4	NO				NO				NO				78.30	129	36	12
wumpus*/p5	NO				NO				NO				TO			
wumpus*/p6	NO				NO				NO				TO			
colorballs*/p21	11.05	47	14	13	12.82	41	11	11	13.43	41	11	11	13.49	41	11	11
colorballs*/p31	NO				43.09	132	25	29	36.85	141	25	29	36.95	141	25	29
colorballs*/p41	NO				TO				106.84	424	42	74	106.79	424	42	74
colorballs*/p51	NO				TO				TO				TO			

TABLE B.5 – Résultats sur les problèmes contingents *coûteux* de CTCF, Contingent-FF, CLG et PO-PRP. t représente le temps de calcul en secondes, s représente la taille du plan en nombre d'actions et d'observations, d représente la profondeur du plan en nombre d'actions et observations et o représente le nombre d'observations au total. NO signifie qu'aucune solution n'est trouvée.

Problèmes	CTCF				C-FF				CLG				PO-PRP			
	t	s	d	o	t	s	d	o	t	s	d	o	t	s	d	o
btcs+/p10	1.25	20	20	0	0.03	37	28	9	0.00	20	20	0	0.02	56	38	9
btcs+/p30	5.62	60	60	0	2.84	117	88	29	0.34	60	60	0	0.04	176	118	29
btcs+/p50	18.76	100	100	0	39.40	197	148	49	2.28	100	100	0	0.1	296	198	49
btcs+/p70	51.0	140	140	0	237.62	277	208	69	8.56	140	140	0	0.18	416	278	69
rovers+/p2	0.58	8	8	0	0.00	13	13	0	0.00	9	9	0	0.02	16	13	1
rovers+/p4	0.81	13	13	0	0.00	19	19	0	0.00	15	15	0	0.02	35	27	3
rovers+/p6	1.00	25	25	0	61.77	117	117	0	NO				0.2	45	39	2
rovers+/p8	0.79	23	23	0	0.10	89	89	0	NO				0.08	32	28	1
erovers+/p2	NO				0.02	32	30	1	0.00	15	13	1	0.02	16	13	1
erovers+/p4	NO				0.10	45	38	3	0.00	27	20	3	0.02	35	27	3
erovers+/p6	NO				4.89	310	65	11	NO				0.06	132	47	11
erovers+/p8	NO				3.23	87	51	3	NO				0.06	45	32	3
elogistics+/p1	NO				0.02	11	8	1	0.00	11	8	1	0.0	16	11	1
elogistics+/p3	NO				0.04	41	26	2	0.00	21	12	2	0.0	27	17	2
elogistics+/p5	NO				0.15	200	32	7	0.04	152	24	7	0.0	107	42	7
elogistics+/p7	NO				0.30	279	42	11	0.08	219	25	11	0.18	139	55	11
doors+/p3	NO				0.02	21	15	2	0.02	15	10	2	0.0	21	14	2
doors+/p4	NO				0.04	37	24	3	0.02	26	15	3	0.0	31	21	3
doors+/p5	NO				0.31	341	78	24	0.22	168	32	24	0.04	182	56	25
doors+/p6	NO				1.54	555	100	35	0.82	287	39	35	0.06	294	73	41

TABLE B.6 – Résultats sur les problèmes contingents coûteux de CTCFE, CTCF+, CTCFL et BCTCF. t représente le temps de calcul en secondes, s représente la taille du plan en nombre d'actions et d'observations, d représente la profondeur du plan en nombre d'actions et observations et o représente le nombre d'observations au total. NO signifie qu'aucune solution n'est trouvée, TO indique un dépassement du temps imposé.

Problem	CTCFE				CTCF+				CTCFL				BCTCF			
	t	s	d	o	t	s	d	o	t	s	d	o	t	s	d	o
btcs+/p10	1.24	20	20	0	1.46	20	20	0	1.24	20	20	0	1.22	20	20	0
btcs+/p30	5.58	60	60	0	6.65	60	60	0	4.99	60	60	0	4.93	60	60	0
btcs+/p50	18.52	100	100	0	19.78	100	100	0	14.92	100	100	0	14.62	100	100	0
btcs+/p70	50.68	140	140	0	50.19	140	140	0	35.46	140	140	0	35.11	140	140	0
rovers+/p2	0.58	8	8	0	0.58	8	8	0	0.60	9	9	0	0.62	9	9	0
rovers+/p4	0.81	13	13	0	0.81	13	13	0	0.83	15	15	0	0.85	15	15	0
rovers+/p6	1.13	23	23	0	0.99	25	25	0	0.88	23	23	0	0.92	23	23	0
rovers+/p8	0.79	23	23	0	0.78	23	23	0	0.77	23	23	0	0.81	23	23	0
erovers+/p2	NO				1.70	14	11	1	1.71	15	11	1	1.72	15	11	1
erovers+/p4	NO				4.17	30	21	3	4.40	32	22	3	4.42	32	22	3
erovers+/p6	NO				14.79	235	41	11	16.44	202	35	11	16.56	202	35	11
erovers+/p8	NO				4.37	71	26	3	4.69	88	34	3	4.70	88	34	3
elogistics+/p1	NO				1.66	11	8	1	1.70	11	8	1	1.78	11	8	1
elogistics+/p3	NO				2.87	20	11	2	2.95	20	11	2	3.11	20	11	2
elogistics+/p5	NO				9.06	152	23	7	9.96	151	23	7	10.38	151	23	7
elogistics+/p7	NO				13.89	217	25	11	15.29	218	25	11	16.11	218	25	11
doors+/p3	NO				2.73	15	10	2	2.84	15	10	2	2.84	15	10	2
doors+/p4	NO				4.44	26	15	3	4.18	26	15	3	4.36	26	15	3
doors+/p5	NO				32.69	190	38	24	38.46	174	36	24	37.84	174	36	24
doors+/p6	NO				TO				71.47	296	45	35	58.95	296	45	35

Cinquième partie

Bibliographie

Bibliographie

Albore, A., H. Palacios et H. Geffner. 2009, ≪A translation-based approach to contingent planning≫, dans *Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, Pasadena, USA.

Albore, A., M. Ramirez et H. Geffner. 2011, ≪Effective heuristics and belief tracking for planning with incomplete information≫, dans *Twenty-First International Conference on Automated Planning and Scheduling (ICAPS)*, Freiburg, Germany.

Baier, J. A. et J. A. Pinto. 2003, ≪Planning under uncertainty as golog programs≫, *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 15, n° 4, p. 383–405.

Bertoli, P., A. Cimatti, M. Roveri et P. Traverso. 2001, ≪Planning in nondeterministic domains under partial observability via symbolic model checking≫, dans *International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 2001, p. 473–478.

Blum, A. L. et M. L. Furst. 1997, ≪Fast planning through planning graph analysis≫, *Artificial intelligence*, vol. 90, n° 1-2, p. 281–300.

Bonet, B. et H. Geffner. 2011, ≪Planning under Partial Observability by Classical Replanning : Theory and Experiments≫, dans *International Joint Conference on Artificial Intelligence (IJCAI)*, Barcelona, Spain, doi :10.5591/978-1-57735-516-8/IJCAI11-324.

Bonnet, B. et H. Geffner. 1998, ≪Hsp : Heuristic search planner≫, dans *Artificial Intelligence Planning Systems 1998 (AIPS-98)*, Pittsburgh, Pennsylvania, USA.

Brafman, R. et G. Shani. 2012a, ≪A multi-path compilation approach to contingent planning≫, dans *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26.

Brafman, R. et G. Shani. 2012b, ≪Replanning in Domains with Partial Information and Sensing Actions≫, *Journal of Artificial Intelligence Research*, vol. 45, doi :10.1613/jair.3711, p. 565–600.

Brafman, R. I. et G. Shani. 2014, ≪On the properties of belief tracking for online contingent planning using regression.≫, dans *European Conference on Artificial Intelligence (ECAI)*, p. 147–152.

Bryant, R. E. 1992, ≪Symbolic boolean manipulation with ordered binary-decision diagrams≫, *ACM Computing Surveys (CSUR)*, vol. 24, n° 3, p. 293–318.

Bylander, T. 1994, ≪The computational complexity of propositional strips planning≫, *Artificial Intelligence*, vol. 69, n° 1-2, p. 165–204.

Church, A. 1936, ≪An unsolvable problem of elementary number theory≫, *American journal of mathematics*, vol. 58, n° 2, p. 345–363.

Cimatti, A., M. Pistore, M. Roveri et P. Traverso. 2003, ≪Weak, strong, and strong cyclic planning via symbolic model checking≫, *Artificial Intelligence*, vol. 147, n° 1-2, p. 35–84.

Cimatti, A., M. Roveri et P. Traverso. 1998, ≪Automatic obdd-based generation of universal plans in non-deterministic domains≫, dans *The Fifteenth National Conference on Artificial Intelligence, AAAI-98*, p. 875–881.

De Giacomo, G., Y. Lespérance et H. J. Levesque. 1999, ≪Congolog, a concurrent programming language based on the situation calculus : foundations≫, *Artificial Intelligence*, vol. 121, n° 1-2, p. 109–169.

De Moura, L. et N. Bjørner. 2008, «Z3 : An efficient smt solver», dans *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, p. 337–340.

Fikes, R. E. et N. J. Nilsson. 1971, «Strips : A new approach to the application of theorem proving to problem solving», *Artificial intelligence*, vol. 2, n° 3-4, p. 189–208.

Ghallab, M., D. Nau et P. Traverso. 2004, *Automated Planning : theory and practice*, Elsevier.

Grastien, A. et E. Scala. 2017, «Intelligent belief state sampling for conformant planning.», dans *International Joint Conference on Artificial Intelligence (IJCAI)*, p. 4317–4323.

Haslum, P. et P. Jonsson. 1999, «Some results on the complexity of planning with incomplete information», dans *European Conference on Planning*, Springer, p. 308–318.

Helmert, M. 2006, «The fast downward planning system», *Journal of Artificial Intelligence Research*, vol. 26, p. 191–246.

Hoffmann, J. 2001, «Ff : The fast-forward planning system», *AI magazine*, vol. 22, n° 3, p. 57–57.

Hoffmann, J. et R. Brafman. 2005, «Contingent planning via heuristic forward search with implicit belief states», dans *Proc. International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 2005.

Hoffmann, J. et R. I. Brafman. 2006, «Conformant planning via heuristic forward search : A new approach», *Artificial Intelligence*, vol. 170, n° 6-7, p. 507–541.

Kaelbling, L. P., M. L. Littman et A. R. Cassandra. 1998, «Planning and acting in partially observable stochastic domains», *Artificial intelligence*, vol. 101, n° 1-2, p. 99–134.

Kolobov, A. 2012, «Planning with markov decision processes : An ai perspective», *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, n° 1, p. 1–210.

Komarnitsky, R. et G. Shani. 2016, «Computing contingent plans using online replanning», dans *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30.

Kuter, U., D. Nau, E. Reisner et R. P. Goldman. 2008, «Using classical planners to solve nondeterministic planning problems», dans *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling*, Sydney, Australia, p. 190–197.

Levesque, H. J., R. Reiter, Y. Lespérance, F. Lin et R. B. Scherl. 1997, «Golog : A logic programming language for dynamic domains», *The Journal of Logic Programming*, vol. 31, n° 1-3, p. 59–83.

McCarthy, J. et P. J. Hayes. 1981, «Some philosophical problems from the standpoint of artificial intelligence», dans *Readings in artificial intelligence*, Elsevier, p. 431–450.

McDermott, D., M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld et D. Wilkins. 1998, «Pddl-the planning domain definition language», dans *Yale Center for Computational Vision and Control Tech Report*.

McMillan, K. L. 1993, «Symbolic model checking», dans *Symbolic Model Checking*, Springer, p. 25–60.

Muise, C., V. Belle et S. McIlraith. 2014, «Computing Contingent Plans via Fully Observable Non-Deterministic Planning», dans *AAAI Conference on Artificial Intelligence (AAAI)*, Quebec City, QC, Canada.

Muise, C., S. McIlraith et C. Beck. 2012, «Improved Non-Deterministic Planning by Exploiting State Relevance», dans *International Conference on Automated Planning and Scheduling (ICAPS)*, Sau Paulo, Brazil.

Mundhenk, M., J. Goldsmith, C. Lusena et E. Allender. 2000, «Complexity of finite-horizon markov decision process problems», *Journal of the ACM (JACM)*, vol. 47, n° 4, p. 681–720.

Palacios, H. et H. Geffner. 2006, ≪Compiling uncertainty away : Solving conformant planning problems using a classical planner (sometimes)≫, dans *AAAI Conference on Artificial Intelligence*, Boston, Massachusetts, USA, p. 900–905.

Palacios, H. et H. Geffner. 2007, ≪From conformant into classical planning : Efficient translations that may be complete too.≫, dans *International Conference on Automated Planning and Scheduling (ICAPS)*, Providence, Rhode Island, USA, p. 264–271.

Palacios, H. et H. Geffner. 2009, ≪Compiling uncertainty away in conformant planning problems with bounded width≫, *Journal of Artificial Intelligence Research*, vol. 35, p. 623–675.

Piedade, S., A. Grastien, C. Lesire et G. Infantes. 2020, ≪Contingent planning using counter-examples from a conformant planner≫, dans *The Sixteenth International Conference on Autonomic and Autonomous Systems (ICAS)*, Lisbon, Portugal, p. 16–22.

Piedade, S., C. Lesire et G. Infantes. 2018, ≪Conditional plans synthesis for decision under uncertainty applied to satellite.≫, dans *Workshop Planning and Learning, International Joint Conference on Artificial Intelligence (IJCAI)*, Stockholm, Sweden.

Pryor, L. et G. Collins. 1996, ≪Planning for contingencies : A decision-based approach≫, *Journal of Artificial Intelligence Research*, vol. 4, p. 287–339.

Puterman, M. L. 2014, *Markov decision processes : discrete stochastic dynamic programming*, John Wiley & Sons.

Richter, S., M. Helmert et M. Westphal. 2008, ≪Landmarks revisited.≫, dans *AAAI Conference on Artificial Intelligence*, vol. 8, p. 975–982.

Rintanen, J. 2004, ≪Complexity of planning with partial observability.≫, dans *International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 4, p. 345–354.

Teichteil-Koenigsbuch, F., G. Infantes et U. Kuter. 2008, ≪Rff : A robust, ff-based mdp planning algorithm for generating policies with low probability of failure≫, *Sixth International Planning Competition at International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 8.

To, S. T. 2012, ≪A new approach to contingent planning using a disjunctive representation in and/or forward search with novel pruning techniques≫, *Journal of Artificial Intelligence Research*.

To, S. T., E. Pontelli et T. C. Son. 2009, ≪A conformant planner with explicit disjunctive representation of belief states.≫, dans *International Conference on Automated Planning and Scheduling (ICAPS)*, Thessaloniki, Greece.

Tovey, C. et S. Koenig. 2000, ≪Gridworlds as testbeds for planning with incomplete information≫, dans *AAAI Conference on Artificial Intelligence*, Austin, Texas, USA, p. 819–824.

Turing, A. M. 1936, ≪On computable numbers, with an application to the entscheidungsproblem≫, *J. of Math*, vol. 58, n° 345-363, p. 5.

Weld, D. et J. Penberthy. 1992, ≪Ucpop : A sound, complete, partial order planner for adl≫, dans *1992 International Conference on Principles of Knowledge Representation and Reasoning*, p. 103–114.

Yoon, S. W., A. Fern et R. Givan. 2007, ≪Ff-replan : A baseline for probabilistic planning.≫, dans *International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 7, p. 352–359.

Titre Synthèse de plans conditionnels pour la décision dans l'incertain

Résumé De nos jours, les robots autonomes sont confrontés à des environnements complexes et incertains nécessitant la planification automatique des différentes tâches devant être accomplies pour mener à bien la mission. Dans cette thèse, nous essayons de résoudre des problèmes dans lesquels l'incertitude est modélisée comme un ensemble d'état initiaux possibles de l'environnement, et nous nous intéressons aux méthodes de planification hors-ligne (avant le départ de la mission), le calcul de plan en ligne entraînant un coût de calcul supplémentaire non négligeable pendant la mission. La planification contingente est une de ces méthodes. Celle-ci consiste à calculer un plan contingent traitant l'incertitude du problème tout en laissant la possibilité d'effectuer des décisions en ligne rapides et conditionnées par des observations de l'environnement. La planification contingente semble particulièrement adaptée aux missions de robotique autonome du fait de la facilité d'embarquabilité des plans contingents, mais celle-ci présente néanmoins une complexité d'autant plus élevée que le nombre d'observations à réaliser est grand. De plus, la réalisation d'une observation en cours de mission peut être coûteuse pour l'agent devant la réaliser. Cette thèse consiste donc à développer un planificateur contingent traitant des problèmes comportant de l'incertitude sous forme d'un ensemble d'états initiaux possibles en limitant le nombre d'observations du plan. Pour cela, nous avons proposé d'utiliser un planificateur conformant (dont le but est de calculer un plan menant au but du problème quel que soit l'état initial possible et sans réaliser d'observation) afin de calculer le plus de branches conformantes possibles dans le plan contingent. Si un plan conformant ne peut pas être calculé, l'approche se sert ensuite des informations retournées par le planificateur conformant pour sélectionner l'observation à réaliser. Une première approche a été développée puis améliorée au fil de la thèse afin d'aboutir à un planificateur contingent complet, doté d'une représentation compacte des états de croyance, et qui contrairement à une grande partie des planificateurs contingents de la littérature, n'est pas limité aux problèmes de taille contingente inférieure à un. Les résultats de la comparaison de notre approche avec les planificateurs contingents de la littérature indiquent que malgré un temps de calcul plus élevé que ces planificateurs sur une grande partie des problèmes étudiés, notre approche limite efficacement le nombre d'observations du plan, rendant les plans générés compétitifs en terme de taille et de profondeur.

Title Conditional plans synthesis for decision with uncertainty

Abstract Nowadays, autonomous robots have to face complex and uncertain environments, which require an automated task planning process to accomplish the mission. In this thesis, we try to solve problems in which the uncertainty is modeled as a set of possible initial states, and we focus on offline planning methods (before the mission), online plan computations adding an additional computation cost during the mission. Among these planning methods, contingent planning consists in computing a contingent plan dealing with uncertainty and giving the possibility to make fast online decisions conditioned by the results of environment observations. Contingent planning seems particularly suitable to autonomous robots missions by the ease of integration of contingent plans, but contingent planning has a complexity issue proportional to the number of observations added to the plan. Moreover, performing an observation during the mission can be costly for the agent. This thesis consists in developing a contingent planner solving problems with uncertainty modeled as a set of possible initial states and limiting the number of observations in the plan. To that end, we propose to use a conformant planner (a planner that computes a plan leading to the goal of the problem whatever the possible initial state and without performing any observation) to compute as much conformant branches as possible. If a conformant plan cannot be computed, the approach uses the information returned by the conformant planner to select the needed observation. A first approach has been developed and improved all along the thesis leading to a complete contingent planner, with a compact belief state representation. Contrary to the majority of state-of-the-art contingent planners, this approach is not limited to problems with a contingent width lower than 1. The results of the comparison with state-of-the-art contingent planners indicate that despite an higher computation time on the majority of the studied problems, our approach efficiently limits the number of observations in the plan, making the generated plans competitive in terms of size and depth.