Robotic Coverage and Exploration as Sequential Decision-Making Problems

Nassim Kaldé

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Robotic Coverage and Exploration as Sequential Decision-Making Problems

THÈSE

présentée et soutenue publiquement le 12/12/2017

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Nassim Kaldé

Composition du jury

Président :	Pr. Philippe Mathieu	Université des Sciences et Technologies de Lille (Lille 1) Laboratoire d'Informatique Fondamentale de Lille (LIFL)
Rapporteurs :	Pr. Philippe Mathieu	Université des Sciences et Technologies de Lille (Lille 1) Laboratoire d'Informatique Fondamentale de Lille (LIFL)
	Pr. René Mandiau	Université de Valenciennes et du Hainaut-Cambrésis (UVHC) Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines (LAMIH)
Examinateurs :	Dr. Laëtitia Matignon	Maître de Conférence, Université Claude Bernard Lyon 1 Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS)
	Dr. Aurélie Beynier	Maître de Conférence, Université Pierre et Marie Curie (UPMC) Laboratoire d'Informatique de Paris 6 (LIP6)
Directeurs :	Dr. François Charpillet	Directeur de recherche, Inria, Nancy Grand-Est Laboratoire lORrain d'Informatique et ses Applications (LORIA)
	Pr. Olivier Simonin	Institut National des Sciences Appliquées de Lyon (INSA Lyon) Centre of Innovation in Telecommunications and Integration of service (CITI-Inria)

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Mis en page avec la classe thesul.

Remerciements

Je remercie toutes les personnes qui partagent une ou plusieurs de ces valeurs : l'humilité, l'empathie, l'honnêteté, l'originalité, la sagesse, la réactivité ; mais aussi la jovialité, le scepticisme, l'autonomie, l'obstination, la gentillesse ; sans oublier le respect, la persévérance, le sérieux, la curiosité et le courage.

Contents

List of Figures

List of Tables

List of Definitions

Context

Chapter 1

Introduction

Contents

1.1 Context: coverage, exploration, and decision-making

This dissertation is a study about coverage and exploration tasks using artificial agents. These tasks are presented as Sequential Decision-Making processes through an Artificial Intelligence lens. Their solutions should ultimately be deployed in the field of Robotics. Envisioned robotic tasks include obstacle avoidance, patrolling, and search and rescue.

First of all, let us introduce the main context of this manuscript.

Preamble

On the one hand, coverage is the task in which robots must physically visit an *a priori* **known** environment to perceptually view it; whereas exploration considers an *a priori* **unknown** environment instead and asks to build its map. There are several applications involving

coverage and exploration in Robotics. In 2003 and 2012, NASA's rovers Spirit, Opportunity, and then Curiosity landed on planet Mars to investigate past water activity and geology. From 2012 to 2015, the DARPA's Robotics Challenge addressed search and rescue of injured people after natural disasters and decommissioning of power plants after nuclear disasters. Since 2002, we can cite the commercial success of iRobot Roomba's vacuums for cleaning the house. In our context, we could consider that exploration takes place the first time such cleaning occurs (while the robot builds a map), and coverage occurs subsequently (based on this map).

On the other hand, Sequential Decision-Making (SDM) is an Artificial Intelligence framework that provides formalisms to describe worlds in which agents interact with their environment to accomplish an objective. During these interactions, the agents are confronted to a series of choices. Therefore, SDM provides planning algorithms as well, so that those agents make rational decisions to fulfill their objective. There are several applications of SDM in game playing. In 1997, IBM's Deeper Blue beat the World Chess Champion Garry Kasparov. In 2016, Deepmind's AlphaGo beat the World Go Champion Lee Sedol. In 2017, University of Alberta CPRG's DeepStack defeated professional poker players in heads-up no-limit Texas hold'em. These demonstrated that the framework of SDM could attain and go beyond human level in particular games. In Robotics, a common application is motion planning of autonomous vehicles for navigation with obstacle avoidance. In our context, obstacle avoidance is mandatory to safely navigate but motion planning is not the central aspect of decision-making. Indeed, we will cast coverage and exploration tasks as games in which such trajectories are the player's moves to gather information. The main question of interest here is: Where to go next?

We introduce the main concepts involved in this manuscript. Our objective is to design and evaluate agents that make decisions in order to solve robot coverage or exploration tasks. Currently, many applications of exploration in Robotics rely on the widely spread frontier paradigm [Yamauchi, 1998, Koenig et al., 2001, Tovey and Koenig, 2003]. In this paradigm, agents perform greedy decision-making based on their current knowledge of the environment. In this information gathering paradigm, agents need to localize themselves and build a map that accumulates their local observations. Based on this map, the agent identifies the areas that remain unexplored and goes toward them. This usually involves a planning algorithm to compute the shortest path to these unexplored areas; its runtime depends on the path length. A team of frontier agents is reliable as there is not a single point of failure in the team.

This paradigm is applied to exploration but it can address coverage as well, although more efficient techniques are preferred in this case. Its good properties make it applicable to simulated and real environments under certain conditions. However, these agents are not easily extensible to prior information regarding the environment. For instance, if we had some hints regarding the dimensions of the environment, *e.g.*, from 20 to 40 m^2, or its density of furnitures, *e.g.*, from 10% to 30%; it is not clear how such agents could take this information into account to improve their behavior.

So, a first question is: **Is there any alternative paradigm which does not put restrictive assumptions on the information used by the agent?**

Moreover, the solutions provided by a frontier agent are satisfiable ones. Similarly, a fair random walker can provide satisfiable solutions as well, but we can admit that in general such behavior is highly suboptimal. Indeed, we could expect an ant agent guided only by pheromones to

perform better than the random walker [Koenig and Liu, 2001, Svennebring and Koenig, 2004]. Furthermore, the frontier agent could perform even better than the ant agent. In general, those agents can provide good solutions in given conditions, but these solutions do not inform much regarding the best attainable behavior.

So, a second question is: **Can we design agents that seek optimal behavior in a task of coverage or exploration?**

Until now, the discussion was mostly informal. We introduced the first concepts that appear in this thesis and the first questions that motivate our study. Below, we want to provide a better picture of how the coverage and exploration tasks fit in Robotics and in Graph Theory. The first locates our study in Robotics, and the second precises what we mean by optimality seeking and why it is extremely difficult.

1.1.1 Coverage and exploration in Robotics

In Robotics, the coverage and exploration tasks are related to problems of *Localization*, *Mapping*, and *Control* [Stachniss, 2009]. Firstly, the Localization problem asks to estimate the robot's pose given its history of actions and measurements and the environment's map [Thrun et al., 2001]. Secondly, the Mapping problem asks to build the environment's map given the robot's history [Elfes, 1989]. In between, the Simultaneous Localization and Mapping (SLAM) problem asks to jointly estimate the pose and the map; this is a chicken and egg problem as these estimates are mutually dependent [Durrant-Whyte and Bailey, 2006, Bailey and Durrant-Whyte, 2006]. These problems are addressed by designing filtering algorithms or state estimators that take as input the robot's history during execution. For instance, the robot's trajectory can be given off-line or on-line by a human operator, then the state estimator filters the robot's history to output a position, a map, or both. Thirdly, the Control problem is different, it asks to plan the robot's moves from a starting position to reach a goal given the map [LaValle, 2006, Latombe, 2012].

In this thesis, we are interested in *Active Sensing* problems at the intersection between state estimation and motion planning [Bajcsy, 1988, Mihaylova et al., 2003]. Precisely, Control and Mapping define many kinds of *exploration* tasks. These exploration tasks range from *Classic Exploration*, in which the current pose and map are reliable and the robot must gather new observations to extend its map; to *Integrated Exploration or Active SLAM*, in which the current pose and map are not reliable and the robot must additionally reduce their uncertainty.

In this thesis, we focus on the case where the pose is reliable. Consequently, we will refer to this variant as the Active Mapping problem.

Active Mapping Problem

We focus on the Active Mapping (AM) problem, *i.e.*, exploration in which positions are reliable. Precisely, the position is provided by an external means and the map is built by a Mapping filter at each time step. Then, the robots are tasked to efficiently gather information for extending their current map (or reducing its uncertainty). Intuitive strategies for Classic Exploration are described by the frontier paradigm where robots compute *satisfiable solutions* by iteratively getting closer to unexplored locations [Yamauchi, 1997]. These locations are either frontiers (locations that separate explored and unexplored areas) or their generalization as view points

(locations from which uncertain areas are visible). Some common exploration strategies are:

***Distance-based*:** as the pioneering *MinDist* in which each robot chooses its closest frontier [Yamauchi, 1998, Yamauchi et al., 1998]; *MinPos* in which each frontier attracts its closest robot [Bautin et al., 2012]; and *MTSP* where each robot chooses the first frontier along the minimum cost cycle visiting a cluster of frontiers [Faigl et al., 2012].

***Information-based* (heuristic value):** they select the frontier of maximum utility ($U = V - \beta C_w$) which combines its weighted shortest path cost C_w and its visibility from other frontiers V [Burgard et al., 2005][1]; the view of maximum utility ($U = Ae^{-\lambda C}$) which combines its maximal visible unknown area A and its shortest path cost C [González-Banos and Latombe, 2002]; the *Pareto* optimal frontier with minimum path cost and maximum visible frontiers and obstacles [Amigoni and Gallo, 2005]; the optimal frontier according to the multi-criteria decision-making strategy *MCDM* that combines distances, maximal visible unknown area, and perimeters of frontiers and obstacles with the Choquet fuzzy integral [Basilico and Amigoni, 2009], etc.

***Information-based* (entropy value):** as for instance *IG-CL* [Stachniss and Burgard, 2003] that selects the viewpoint of maximum utility ($U = \alpha\,\mathbb{E}[I] - C$), which linearly combines its expected information gain I (over potential measurements) and its shortest path cost C [2]; and as [Moorehead et al., 2001] that linearly combines many sources of information (frontier, traversability, reachability).

***Others*:** as the *market-based* strategy [Zlot et al., 2002] in which robots bid to buy frontiers during auctions; and the *motivation-driven* strategy [Macedo and Cardoso, 2004] in which robots are guided by hunger or curiosity; among others.

These techniques consider short-term objectives and provide good performances in practice, however simple environments exist in which greedy mapping strategies are suboptimal [Koenig et al., 2001, Tovey and Koenig, 2003]. Moreover, they evaluate view candidates based only on the current map output by the filter and the robot's position. They do not look many steps beyond the current view candidates to evaluate potential future maps. Indeed, this necessitates to query the map estimator with hypothetical histories of actions and measurements in advance.

In Integrated Exploration or Active SLAM, it is common to query the pose and map estimator for evaluating actions. These actions lead to novel measurements that must be considered as random variables prior to execution. Consequently, the actions are evaluated according to their expected payoff which usually combines navigation costs, and both localization and mapping uncertainty. For instance, parametric spiral curves reentering the explored area are evaluated in terms of state uncertainty and distance to known areas by querying a *Slam Extended Kalman Filter (EKF)* in [Sim et al., 2004]; frontier destinations are evaluated by their entropy reduction on an occupancy grid and their localization utility by querying an EKF in [Makarenko et al., 2002, Bourgault et al., 2002]; and frontier and loop-closing destinations are evaluated by their entropy reduction by querying a *Rao-Blackwellized Particle Filter* in

[1] A Value Iteration algorithm weights the path cost with the probability of cell occupancy. A frontier is penalized if it is visible from already selected frontiers.

[2] Each cell holds a probability distribution regarding its occupancy. A viewpoint is a cell whose entropy is higher than a predefined threshold. The information gain is the difference in entropy of visible cells after any measurement.

[Stachniss et al., 2005].

These studies use constrained policies and perform a single-step lookahead (up to the next destination). Moreover, they provide specific state estimators with many subtleties to scale in practice. In Sequential Decision-Making (SDM), these exploration tasks involve partial observability and stochastic acting and sensing. Consequently, they can be modeled in the Partially Observable Markov Decision Process (POMDP) formalism [Cassandra et al., 1994]. Furthermore, in theory, the associated stochastic planning techniques can address multi-step lookahead with unconstrained policies and simple *Bayesian* state estimators. Unfortunately, in practice, such planning algorithms do not scale well, and as a consequence their formalisms are often overlooked [Ross et al., 2008].

Nevertheless, we think that it is still useful to model such problems in formalisms of SDM, independently of the planning techniques that are considered to address them. It allows to abstract components related to actions, measurements, and state estimators, and consider simple yet expressive models of different exploration tasks. Subsequently, these models could be used to provide optimal baselines for small instances, improved with more efficient representations and state estimators, and solved by original planning algorithms. As a first step, we will focus on AM in which the agent can query its map estimator for multi-step lookahead.

To summarize, the AM problem asks the following question: **How can the robots compute a mapping strategy in an unknown environment when localization is reliable?**

We did not mention coverage yet but the link between exploration and coverage is straightforward.

Active Coverage Problem

The AM problem is linked to coverage by the following question: **How can the robots compute a coverage strategy in a known environment when localization is reliable?** To emphasize this link, we will now refer to this coverage task as the Active Coverage (AC) problem [3]. Techniques that apply to AC differ from those discussed previously; they are specialized and can provide approximately optimal solutions [Galceran and Carreras, 2013]. Nevertheless, from our point of view, AM and AC are so closely related that techniques for exploration should be general enough to consider coverage as a special case.

Precisely, the difference is the information available at the outset. One the one hand, if the robot knows nothing about the environment then it is the hardest exploration case where multi-step planning is intractable and greedy behavior is the best option. On the other hand, knowing the environment leads to the simplest coverage case where multi-step planning may be tractable in practice and outperform greedy strategies. In between, there is a continuum of information and multi-step planning may already be tractable in practice above some value in this continuum.

To exemplify, in Table C.1, we illustrate the behavior of the frontier agent MinDist [Yamauchi, 1997]. The map is a grid containing 4 cells that are free of obstacles and the robot is on the second cell. During execution the agent observes its location. As an additional

[3] In this manuscript, we use Active Coverage instead of *Coverage Path Planning* which is the commonly agreed term. Thereby, we want to emphasize that robotic coverage is the ground truth version of robotic exploration.

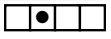
	Truth	Known	Used	Decision
Coverage			$p = 2$	$\arg\min_{f \in \{f_1, f_2\}} Cost(p^*(p, f))$
			$f_1 = 1$	where
Exploration			$f_2 = 3$	$Cost(p^*(p, f_1)) = Cost(p^*(p, f_2)) = 1$ hence go $\Leftarrow$ or $\Rightarrow$

Table 1.1: This table illustrates the behavior of the frontier agent with MinDist [Yamauchi, 1997] when addressing coverage or exploration. *Truth* is the ground truth occupancy map m (one row of 4 free cells in white) and robot's position p illustrated as a $\bullet$ in the second cell ($p = 2$). *Known* is the information available to the agent. In coverage, the agent knows the ground truth. In exploration, the agent only knows the size of the map and its position; the other elements are grayed out. *Used* is the information used by the agent to deliberate. For both coverage or exploration, the agent uses only its position p, and the presence of a frontier f_1 to its left, and of a frontier f_2 to its right. *Decision* is the decision of the agent after deliberation. Here, the agent goes toward the closest frontier. The agent computes the shortest path from p to f_1 ($p^*(p, f_1)$), and does the same for f_2. In this example the agent can either go to f_1 or f_2 as $Cost(p^*(p, f_1)) = Cost(p^*(p, f_2)) = 1$.

information, we provide a perfect bounding box that delimitates the environment relative to the robot's position. Note that such information is not required to enable the frontier agent.

In coverage, the frontier agent knows the ground truth: the map m and its position p. When the coverage mission starts, only the current cell is covered. Firstly, the agent extracts frontiers separating covered cells from cells that are not covered yet; the frontiers are the cells immediately to its left and to its right (f_1 and f_2). Secondly, the agent evaluates the shortest path to these frontiers; in this case the distance is evaluated as the number of moves. Thirdly, the agent goes to the nearest frontier; here they are both one step away so it can choose either left or right.

In exploration, the frontier agent knows only its current location in the bounding box. When the exploration mission starts, only the current cell is explored. Subsequently, the agent behaves the same as in the coverage task (left or right).

In both coverage and exploration, the frontier agent cannot account for the additional information (the bounding box). If the agent could take into account the bounding box information, the best move would be to first go to the left. Although this example illustrates the case of MinDist, the same decisions are made by at least the aforementioned distance-based strategies.

To summarize, firstly, we situated our study in Robotics, in this thesis we regard exploration as an Active Mapping problem and coverage as the closely related Active Coverage problem. Secondly, we expressed our will to model such tasks in the formalisms of Sequential Decision-Making in order to abstract their main components. Thirdly, we expressed the idea of addressing both Active Coverage and Active Mapping problems with the planning paradigm. In the following, we explain how the optimality criterion of exploration can be derived from the optimality criterion of coverage. Additionally, we report results of Graph Theory to justify that attaining optimality is extremely difficult in these tasks.

1.1.2 Coverage and exploration in Graph Theory

In Graph Theory, coverage and exploration tasks are problems in which *optimal solutions* are sought. In the following, the environment is denoted by a graph $G(V, E)$, where vertices V are locations, and edges E are links between locations.

Traveling Salesman Problem

We disable remote sensing in coverage, the robot knows the environment prior to execution and must efficiently visit all locations at least once when starting from a given position. From this, we can define the Minimum Coverage Path (MCP) problem (Definition C.1), in which a sequence of moves through the edges of the graph must visit all vertices at least once and with minimum cost. A function w_E weights each edge in E and represents the cost of going through a link. Optimal sequences must minimize the sum of their edge weights. The MCP can be reformulated as the Traveling Salesman Problem (TSP) which is an *NP-hard* optimization problem [Garey and Johnson, 2002, Wernli, 2012]. Hence, it is highly unlikely that a polynomial time algorithm exists for this problem.

Definition 1.1 (Minimum Coverage Path (MCP)).

Instance *Graph $G(V, E)$, a function $w_E : E \to Z^+$.*

Question *Find a minimum coverage* **path**. *That is, a sequence $S = \langle v_1, v_2, ..., v_k \rangle$ of vertices of V, such that $\forall i \in [1, k-1], (v_i, v_{i+1}) \in E$ and $\cup_{i=1}^{k} v_i = V$, with minimum cost $\sum_{i=1}^{k-1} w_E(v_i, v_{i+1})$.*

In Sequential Decision-Making (SDM), this coverage task involves deterministic acting and sensing with local observations that are known in advance. Consequently, its decision process can be modeled in the deterministic Completely Observable (State) Search Problem (COSP) formalism [Russell and Norvig, 1995] over agent's knowledge states. Moreover, in theory, deterministic planning techniques could compute optimal trajectories that cover all nodes at least once [Hart et al., 1968]. From our point of view, the MCP is the easiest case of Active Coverage (AC) and involves *long-term planning without uncertainty*.

What about exploration tasks in Graph Theory?

Canadian Traveler Problem

Again, we disable remote sensing in exploration, the environment is unknown prior to execution but the robot can localize itself and must optimally build a map. In other words, the set of locations, their relative links, and the weights are initially hidden from the robot. In complete absence of knowledge regarding the environment, defining a long-term optimality criterion for the robot is a tedious task. However, if we could initially reveal some properties of the environment such as its dimension or the number of rooms and corridors, exploration could be seen as computing the MCP for a given class of graphs.

Precisely, with such information, the exploration task can be thought of as a combination of the MCP and the Canadian Traveler Problem (CTP) [Papadimitriou and Yannakakis, 1991, Bar-Noy and Schieber, 1991]. The CTP is an on-line navigation task in which a robot has to make a single decision at each time step in order to go from a source node to a target

node in a partially known graph. There are several variants of this problem. In the best average-case, the robot makes a move so that its full path length is minimized when averaged over the class of potential graphs. Initially, this version was proven to be *#P-hard*, and later it was extended to *PSPACE-hardness* [Fried et al., 2013]. In SDM, this navigation task under uncertainty involves deterministic acting and sensing with local observations that are not known in advance but the robot has a prior regarding the environment. Hence, it has been studied by [Blei and Kaelbling, 1999, Nikolova and Karger, 2008] in the Partially Observable Markov Decision Process (POMDP) formalism over hidden world's states [Littman, 1996, Bonet, 2009].

Here, we view the exploration task as a coverage task in a partially known graph. The Covering CTP appeared recently in Graph Theory and combines the MCP and the CTP [Liao and Huang, 2014]. Hence, the previous optimization criterion can be restated as choosing a move that minimizes the coverage path length on average. In SDM, this coverage task under uncertainty requires to reward the robot according to its belief regarding the environment map. Consequently, its domain can be modeled in the Partially Observable Markov Decision Process with belief-dependent rewards (ρPOMDP) formalism ([Araya-López et al., 2010a]). From our point of view, this exploration task is the simplest case of Active Mapping (AM) and involves *long-term planning under uncertainty*.

	Truth	Known	Used	$\text{MCP}_{\Leftarrow}$	$\text{MCP}_{\Rightarrow}$	Decision
Coverage				$\Rightarrow \times 3$	$\Rightarrow, \Leftarrow \times 3$	$\arg\min_{a \in \{\Leftarrow, \Rightarrow\}} Cost(MCP_a(p,m))$ where $Cost(MCP_{\Leftarrow}(p,m)) = 4$ $Cost(MCP_{\Rightarrow}(p,m)) = 5$ hence go $\Leftarrow, \Rightarrow \times 3$
Exploration				$\Rightarrow \times 3$	$\Rightarrow, \Leftarrow \times 3$	$\arg\min_{a \in \{\Leftarrow, \Rightarrow\}} \mathbb{E}_m[Cost(MCP_a(p,m)]$ where $\mathbb{E}_m[Cost(MCP_{\Leftarrow}(p,m)] = 1 + 16/8$ $\mathbb{E}_m[Cost(MCP_{\Rightarrow}(p,m)] = 1 + 18/8$ hence go $\Leftarrow$
				$\Rightarrow \times 2$	$\Leftarrow \times 1$	
				$\Rightarrow \times 2$	$\Leftarrow \times 1$	
				$\Rightarrow \times 2$	$\Rightarrow, \Leftarrow \times 3$	
				$\Rightarrow \times 2$	$\Rightarrow, \Leftarrow \times 2$	
				$\Rightarrow \times 1$	$\Leftarrow \times 1$	
				$\Rightarrow \times 1$	$\Leftarrow \times 1$	

Table 1.2: This table illustrates the behavior of the planning agent when addressing coverage or exploration. *Truth* is the ground truth occupancy map m (one row of 4 free cells in white) and robot's position p illustrated as a $\bullet$ in the second cell ($p = 2$). *Known* is the information available to the agent. In coverage, the agent knows the ground truth. In exploration, the agent only knows its position p and the size of m. *Used* is the information used by the agent to deliberate. In coverage, the agent uses the ground truth. In exploration, the agent relies on what it knows to generate the hidden maps m. *Decision* is the decision of the agent after deliberation. In coverage, the agent computes the $\text{MCP}_{\Leftarrow}(p,m)$ starting with the left move; and does the same for the right move. The agent goes to the left as $Cost(\text{MCP}_{\Leftarrow}(p,m)) = 4 < Cost(\text{MCP}_{\Rightarrow}(p,m)) = 5$. In exploration, the agent computes the expectation of $\text{Cost}(\text{MCP}_{\Leftarrow}(p,m))$ over all hidden maps (occupied cells in black) m starting with the left move, and does the same for the right move. The agent goes to the left as $\mathbb{E}_m[Cost(\text{MCP}_{\Leftarrow}(p,m))] = 3 < \mathbb{E}_m[Cost(\text{MCP}_{\Rightarrow}(p,m))] = 3.25$.

To exemplify, in Table C.2, we illustrate the behavior of a planning agent that makes the best

decision. In this example, the map is a grid of 4 free cells and the robot is on the second cell. During execution, the agent observes its location. Hence, to infer that a cell is blocked by an obstacle, the agent must bounce against it. Again, as an additional information, we provide a perfect bounding box that delimitates the environment relative to the robot's position.

In coverage, the agent knows the ground truth (map and position) and exploits such information to compute the MCP that visits all cells. When the coverage mission starts, only the current cell is covered. In this case, the MCP is to go left once and to go right three times afterwards giving a path of 4 moves.

In exploration, the agent knows its position and the environment's bounding box, and will use this information to compute the MCP that visits all cells. However, as the map is hidden, the MCP is also hidden. Without any additional information, the agent could assume that the map is sampled uniformly. Firstly, the agent samples the potential maps (with free cells in white and occupied cells in black). Secondly, for each sampled map, the agent computes the MCP that starts with a left move and the MCP that starts with a right move. Thirdly, the agent computes the expectation of the path length of the MCP for each starting move. Finally, the agent goes to the left direction, as its expected MCP length is the lowest.

This example is only illustrative: an efficient solver should approximate the expected payoff of a move by generating only a few samples of maps, looking only a few steps ahead on any coverage path, and balancing the cost of any partial coverage path with its utility in terms of information gain.

To summarize, we reported studies related to coverage and exploration tasks in experimental Robotics and Graph Theory. In Robotics, these tasks are related to Active Coverage and Active Mapping, for which satisfiable solutions are already available. In Graph Theory, they are related to the Traveling Salesman Problem and Canadian Traveler Problem, whose optimal solutions may be extremely difficult to compute. Furthermore, we reported long-term optimization criteria that can be used to define planning agents that seek optimality although attaining optimality is highly unlikely given their classes of complexity. Additionally, we identified formalisms of SDM that represent starting points for modeling AC and AM problems. Finally, we gave an example to demonstrate the ability of a planning agent to make an optimal move on average.

In the following, we define the long-term goal of this dissertation and our short-term assumptions.

1.1.3 Long-term goal and short-term assumptions

Our long-term goal is to solve coverage and exploration tasks with real robots. Designing a single robot is challenging along three components:

Perception: the agent relies on its on-board sensors to provide data concerning its environment to its on-board decision-making unit;

Decision-making: the agent relies on its decision-making unit to output rational decisions to its on-board end-effectors; and

Action: the agent relies on its end-effectors to apply the decisions, in order to act upon the environment.

Consequently, we concentrate on decision-making and make strong assumptions regarding perception and action. We abstract the perception and action units to their simplest form, by inputting and outputting formatted data. Furthermore, we focus on designing simulated robots that take rational decisions during Active Coverage or Active Mapping missions.

Assumptions on continuity, perception, and action

Mobile robots have many degrees of freedom, motors, and sensors. Moreover, the physical environment is continuous and complex. Therefore, many roboticists conduct studies in which the environment's configuration and the robot's configuration, inputs, and outputs are continuous. Nevertheless, there are also many studies that rely on discrete representations of the environment, such as the *de facto standard* deterministic occupancy grids [Thrun, 2003], probabilistic occupancy grids [Elfes, 1989], and their sparse extensions [Hornung et al., 2013]. Similarly, we will manipulate discrete workspaces.

We put strong assumptions on perceptions as we do not consider the process of calibrating sensors, reading raw signals, and then transforming these into useful information for the decision-making unit (by denoising, filtering, smoothing). We consider formatted measurements at the input of the decision-making unit. Similarly, for actions, we do not consider the process of transforming a decision into a real action in the environment (by activating motors, setting velocity or torque). We consider formatted decisions at the output of the decision-making unit.

Nevertheless, the information required by the decision-making models reported in this dissertation is fairly reasonable, although not trivial. Indeed, it may require solving pre-deliberation perception problems as the data association problem (depending on the environment's dynamics); and post-deliberation action problems as kinematics or dynamics equations (depending on the robot's type).

There are already many real exploration systems reported in the literature of experimental Robotics and robotics integration softwares [Quigley et al., 2009] are now broadly adopted. Our work assumes that robots are localized by an external means, to do so we can rely on accurate *Indoor Positioning Systems* [Liu et al., 2007]. Similarly, we assume that robots can distinguish between classes of static objects (walls) and mobile objects (animals, pedestrians), to do so we can rely on recent results in *Computer Vision* [Krizhevsky et al., 2012]. Hence, an integration does not seem to represent a major limitation despite the aforementioned assumptions.

In the following, we identify challenges related to decision-making and our proposals for addressing them; these proposals will be subsequently detailed by different studies.

1.2 Challenge: identified issues and proposed studies

1.2.1 Challenge I: models of single-agent coverage and exploration

Firstly, we want to exploit the framework of Sequential Decision-Making (SDM) in Artificial Intelligence to provide first answers to our questions regarding agent paradigms with nonrestrictive information and optimality seeking in robot coverage or exploration.

How to specify an agent paradigm that uses all available information for coverage and exploration?

We mentioned that the frontier paradigm restricted the information used by the agent for decision-making. Furthermore, we illustrated a simple planning agent that used additional information to foresee what occurs beyond the frontier, whereas the frontier agent was blind to it. Envisioning what may occur beyond the frontier is appealing as it allows for finer grained decision-making. Nevertheless, the modalities are simplistic in our example and its generalization is unclear. We think that generalizing such planning agent for Active Coverage (AC) and Active Mapping (AM) will ease the adoption of decision-theoretic techniques that foresee beyond the frontier.

How to specify agents that seek optimality for coverage and exploration?

We already put some emphasis on short-term vs long-term objectives. We stated that frontier agents from Robotics greedily validated short-term objectives and provided good suboptimal behaviors. We also illustrated the behavior of a planning agent that used all information and decided according to the long-term objective defined by the Minimum Coverage Path problem. As expected, this agent made better decisions than the frontier agent in the same situation. We know that attaining optimality is highly unlikely given the complexity classes of the Traveling Salesman Problem and the Canadian Traveler Problem. Nevertheless, seeking long-term optimality to make sharper decisions may be a reasonable route for study.

As stated in our context, we will rely on the framework of SDM [Littman, 1996] for addressing such questions. This framework contains standard decision-process formalisms that are: expressive, which make them general, as they specify the interaction of the agents and their environment to accomplish a task; comprehensive, which make them strict, as they encapsulate only the relevant aspects of the task; compact, which make them transferable, as they specify a few items that describe the environment's dynamics and the agent's actions, perceptions, and rewards; and computational, which make them reproducible, as each specification is a simulator.

Hence, firstly, we model coverage and exploration domains by specifying several modalities of interaction: we rely on state space formalisms to simulate worlds in which a situated agent acts upon and senses the environment in a strict manner. Secondly, we enable the planning paradigm for coverage and exploration, in which the agent accesses the task model, prior information, and histories of actions and measurements: we rely on sufficient statistics that compile such information. That is, we model decision processes over these statistics and show how information gathering can be rewarded in the long-term.

We concentrate on Single-Robot Active Coverage and Active Mapping and assume a centralized planning agent. A centralized planning agent can consider a single or many synchronous robots seamlessly. However, we illustrate the case of a single planning agent that compute decisions for a single robot. The case of decentralized decision-making is different as it considers a planning agent for each robot. This time, observations and actions are not jointly made by the robots, and communications must be explicitly modeled. Decentralized decision-making is out of the scope of our current study, but such formalisms exist and must be further investigated [Bernstein et al., 2002].

1.2.2 Challenge II: centralized coverage of static environments

Secondly, we want to empirically demonstrate the current limitations of guaranteeing optimality with classical planning techniques on the models of robot coverage. Previously, we showed that computing the next move in exploration could involve planning solutions to sampled coverage tasks. Hence, as a first step, we will focus on coverage tasks and rely on heuristic search to obtain such solutions.

What are the current limitations for guaranteeing optimal, suboptimal, or anytime solutions to the coverage model?

We reported coverage models for the planning paradigm, at the agent's knowledge level with an access to the task model. Hence, the route is opened for studying coverage tasks as SDM games with planning agents that can perform non blind decision-making. Therefore, we make two additional steps in this direction. Firstly, as we cannot study all the coverage models and the general solvers that apply to them, we focus on the coverage model with deterministic control. Secondly, the planning agent can rely on a family of shortest path solvers called heuristic search that applies to this coverage model. We would like to assess its performances in terms of solution quality and decision-making cost.

As stated above, we will rely on heuristic search techniques that are standard planning techniques from SDM [Edelkamp and Schroedl, 2011]. These techniques are: shortest path solvers, which make them generic, as they apply to the broad class of problems that can be cast as shortest path problems; adjustable, which make them specific, as they rely on expert knowledge to evaluate if a given state is promising; and theoretically grounded, which make them attractive, as they enjoy properties and theorems that guarantee completeness, correctness, epsilon-suboptimality or also anytime performances under reasonable assumptions.

As the literature of heuristic search is vast, we restrict our study to the off-line search paradigm ([Dijkstra, 1959, Hart et al., 1968]). The on-line search ([Ramalingam and Reps, 1996, Koenig et al., 2004]) and real-time search ([Korf, 1990, Pemberton and Korf, 1992]) paradigms are out of the scope of our current study and must be further investigated. Firstly, we consider algorithms from the well known Best-First Search and Depth-First Search frameworks. Secondly, we conduct experiments with these techniques to solve instances of the coverage model; we tune them with expert knowledge and assess their average behavior to output optimal, suboptimal, and anytime solutions under resource constraints.

This is different from the frontier paradigm that provides satisfiable solutions, can guarantee completeness, but do not provide guarantees regarding its suboptimality. To emphasize the difference between the planning agent and the frontier agent: the planning agent evaluates frontiers with a long term objective of optimal coverage, that is each frontier value accounts for the shortest path to the frontier and the shortest coverage path beyond it. Instead, the frontier agent evaluates the frontiers with a short-term objective of satisfiable coverage, and considers only the shortest-path to the frontier.

1.2.3 Challenge III: decentralized exploration of dynamic environments and distributed roadmap of ambient environments

Thirdly, we want to consider more realistic setups. The previous models considered only a single robot; subsequently we considered an off-line planning agent to solve a deterministic coverage model. These models could be extended to a team of robots in dynamic environments, and then we could consider an off-line planning agent for solving the task. However, although extended models can be defined, the previous off-line planning agent would necessitate considerable resources to compute the robots' control.

How to enable decentralized robotic exploration in dynamic environments with pedestrians?

We study the more realistic setup of an exploration task that involves many robots that each computes decisions in a decentralized manner and relies on re-planning to correct obsolete plans. Moreover, we are interested in dynamic environments as robots increasingly share the physical space with humans at public or private places like museums (*e.g.* robot guides), gardens (*e.g.* robot lawn mowers), or nursing homes (*e.g.* robot health assistants).

How to enable distributed computation of safe navigation paths in ambient environments?

Until now, we mainly focused on coverage of known environments and map building in unknown environments, but in a near future more houses and facilities will be equipped with ambient intelligence, that embed sensing and processing units in the environment. In this context, coverage and exploration could involve a team of robots that rely on such environment-embedded units to extend their perception. Consequently, we investigate how robots could rely on such units to address obstacle avoidance.

Hence, we propose to: extend the application scope of Active Mapping to environments with moving objects; and collaboratively build navigation services to safely navigate the environment.

In the first case, we model Multi-Robot Active Mapping in a dynamic environment with pedestrians. The modalities of this model allow to rely on the frontier paradigm. Then, we propose an interactive paradigm that enables local interactions with the pedestrians. Finally, we quantitatively assess the performances of the team of robots when balancing frontier assignments and pedestrian following interactions in a decentralized manner.

In the second case, we model an ambient environment equipped with a load pressure sensing floor as a Cellular Automaton. Then, we address the problem of Clearest Path Obstacle Avoidance by the means of spatial computing in an asynchronous manner. The units embedded in the load pressure sensing floor collaboratively build a maximum clearance roadmap. Finally, we qualitatively assess the properties of the computed roadmaps in different kinds of environments with both synthetic and real data.

1.3 Outline: summary of the proposed studies

To end this introduction, here is the outline of the manuscript.

1.3.1 Part I: State Space Modeling for Active Coverage and Mapping

In this part, we specify domains of Single-Robot Active Coverage and Active Mapping. A domain firstly describes how an agent interacts with its environment. It encapsulates all aspects of the world (environment, agent, and their interactions) that are relevant to a given task, and necessary to simulate the courses of actions, observations, and rewards of the situated agent. Moreover, we provide models that can support decision-making by a planning agent in such domains. These models describe how the agent can anticipate and is rewarded while building solutions to a given task.

Chapter 2

This chapter is dedicated to modeling domains of Single-Robot Active Coverage in SDM formalisms. We assume that local observations are known in advance, hence the decision process will be defined over the agent's knowledge states. Indeed, in our coverage task, the world's state consists of the agent's position and the environment's map, and the map is initially revealed to the agent prior to execution.
Firstly, we report state-based formalisms of Sequential Decision-Making with completely-observable states. Secondly, we provide domains and decision-making models of coverage with different state-transition types that describe how the agent moves and maintains a coverage map. In these decision-making models, the planning agent knowns its current level of achievement by relying on a coverage map.

Chapter 3

This chapter is dedicated to modeling domains of Single-Robot Active Mapping in SDM formalisms. We assume partial observability of the world's state. In the exploration task, the world's state consists of the agent's position and the environment's map; the agent knows its position but it only locally observes the map during execution.
Firstly, we report state-based formalisms with partially observable states that differ in terms of state-transition and state-observation types. Secondly, we provide an exploration domain for each state-based formalism. Then, we report belief-based formalisms with observable beliefs that differ in terms of belief-transition types. Finally, we provide exploration models over agent's beliefs that are based on the exploration domains; in these models the planning agent explicitly maintains a belief regarding the true map of the environment.

1.3.2 Part II: Long-Term Planning for Deterministic Active Coverage

In this part, we empirically evaluate the performances of the planning agent on Single-Robot Active Coverage tasks as specified in the deterministic coverage model. Our approach is to search for a solution in the state space of the coverage model. Indeed, this model produces deterministic shortest path problems that can be addressed by heuristic search algorithms.

Chapter 4

This chapter is dedicated to solving instances of the deterministic domain of Single-Robot Active Coverage. For this purpose, we rely on the paradigm of off-line search. Off-line search algorithms can find an optimal sequence of moves linking the initial state of the agent to a goal. In our model, a goal is a state in which the agent has perceptually viewed the entire environment by

physically visiting it. Furthermore, these algorithms can be tuned with expert knowledge to improve their performances.
Firstly, we discuss the principles at the core of off-line search: unbounded deliberation and one shot shortest path planning. Secondly, we review the state of the art of its most popular techniques. Thirdly, we conduct a benchmark of representative techniques for computing optimal or suboptimal solutions to the deterministic coverage task in a complete or an anytime manner.

1.3.3 Part III: Short-Term Planning for Deterministic Active Mapping and Clearest Path Obstacle Avoidance

In this part, we extend the application scope of Multi-Robot Active Mapping to dynamic environments with pedestrians, and of Clearest Path Obstacle Avoidance to ambient environments with distributed sensing and processing units.

Chapter 5

This chapter is dedicated to solving instances of the deterministic domain of Multi-Robot Active Mapping in a crowded environment with pedestrians. We consider a team of robots that compute decisions in a decentralized manner. For this purpose, we will rely on the frontier paradigm and additionally propose an interactive paradigm in which robots seek to follow pedestrians during exploration.
Firstly, we extend the previous Single-Robot Active Mapping model with deterministic acting and sensing, to multiple robots and pedestrians. Secondly, we hybridize the frontier paradigm with an interactive paradigm. The interactive paradigm allows to locally follow moving objects in the environment. Such interactions are evaluated based on idle time, orientation, and distance penalties. Thirdly, we quantitatively evaluate in simulation how this hybrid paradigm performs in crowded environments.

Chapter 6

This chapter is dedicated to the problem of Clearest Path Obstacle Avoidance in an ambient environment with sensing and processing units embedded in the floor. Clearest Path Obstacle Avoidance is an alternative to Shortest Path Obstacle Avoidance, the former tries to produce trajectories that are as far as possible from the obstacles whereas the latter outputs trajectories that are in their vicinity. Clearest trajectories are preferred in dynamic environments where moving objects can easily collide with the robots. The floor units asynchronously compute such trajectories in a distributed manner. Indeed, maintaining the units synchronized at any time is difficult.
Firstly, we model the load sensing floor as an abstract Cellular Automaton whose cells represent the units. Secondly, we compute a Voronoi Diagram whose centroids are given by the obstacles perceived by the floor. Thirdly, we qualitatively evaluate in simulation on synthetic and real data how bisectors between Voronoi regions could serve as maximum clearance roads for mobile robots.

1.3.4 Conclusion and Perspectives

This part concludes the different studies and envision potential improvements and extensions.

Chapter 7

This chapter summarizes the single-agent models of coverage and exploration that are provided in this document; the benchmark of off-line search techniques on our deterministic coverage model; the study about multi-agent exploration in dynamic environments with pedestrians; and the study about navigation in ambient environments with embedded sensing units.

Additionally, we discuss further studies regarding the multi-agent models with decentralized decision-making for coverage and exploration; and the empirical performances of on-line and real-time planning techniques on our deterministic coverage model. And then, we propose potential improvements to the evaluation of interactions between robots and pedestrians; and the computation of maximum clearance roadmaps.

Part I

State Space Modeling for Active Coverage and Mapping

Chapter 2

Completely Observable Formalisms: Models of Active Coverage

Contents

2.1 Introduction

In this chapter, we address Single-Robot Active Coverage tasks from the point of view of Sequential Decision-Making (SDM) in Artificial Intelligence. There are several approaches for addressing the coverage tasks with autonomous agents and each approach makes specific assumptions regarding the information used by the agent for decision-making. For instance, we could consider ant agents that rely on their current observation to follow the gradient of pheromones or frontier agents that rely on their current knowledge to get closer to the frontiers.

Nevertheless, we envision a more general approach, in which an agent possesses as much information as possible to solve the coverage task. To do so, we rely on SDM formalisms to strictly specify the agent and environment interactions at the world's state level. Additionally, we enable the planning paradigm at the agent's knowledge state level, where a knowledge state informs the robot regarding locations that are currently covered and those that are not covered yet. In this paradigm, the agent is able to query future knowledge states in advance.

Hence, we formalize three extensions of the planning agent for coverage (previously illustrated in Table C.2). Our extensions enable remote sensing and consider three types of control. In coverage, the ground truth local observations are known in advance as the environment's map is

initially revealed. Additionally, we assume that the local observations made by the robot during execution perfectly match the ground truth local observations. That is, the robot's remote sensor is perfect. We consider the more general case of an unknown initial map and imperfect remote sensing in the next chapter dedicated to Single-Robot Active Mapping (Chapter 3).

Consequently, this chapter focuses on formalisms for observable knowledge states. We report decision-making models of Active Coverage that differ in terms of control. Any such model describes and simulates a process in which a planning agent covers the environment. These models support both short-term planning at the level of the current knowledge state or long-term planning by anticipating several knowledge states in advance.

These decision-making models serve to:

1. Introduce notations and primitives that are reused in Chapters 3 and 4;
2. Support the more general Active Mapping tasks in Chapter 3; and
3. Benchmark long-term planning algorithms in Chapter 4.

Firstly, in Section 2.2, we discuss the concept of a state-transition model that appears in most formalisms of SDM. Secondly, in Section 2.3, we report three coverage models that instantiate SDM formalisms for observable knowledge states and deterministic, nondeterministic, and stochastic control. Finally, in Section 2.4, we conclude this chapter dedicated to the instantiation of decision-making models for the coverage task.

2.2 Background: state-transition models

A state-transition model describes how the world's state changes with time and how the agent's decision affects it. When the rules that dictate the world's dynamics, *i.e.*, the dynamics of the environment and the agent, do not change over time, such rules are qualified as *stationary* or otherwise *nonstationary*.

Let us first consider an environment and an agent as two distinct entities, and identify relevant information for building a state-transition model.

2.2.1 Modeling environment dynamics

The environment's dynamics must be described by the state-transition model. Informally, it is a mapping from everything that ever happened during the previous time steps to what can happen during the next time step. When the environment can change with time without undergoing any external input, we say that it follows the *dynamic assumption*, *e.g.*, a clock described only by its time attribute incurs an intrinsic dynamic of time updates. Otherwise, if it does not change with time unless it undergoes an external input, we say that it follows the *static assumption*.

2.2.2 Modeling agent dynamics

The agent's dynamics must be described by the state-transition model as well. Like the environment, the agent may have *intrinsic dynamics*, *e.g.*, battery consumption for an idle robot, and *extrinsic dynamics*, *e.g.*, battery charging when plugged into a power supply. When specifying a process, we are concerned about describing how the actions of the agent affect itself

and the environment, thus the transition model must account for these interactions.

In the following when we refer to a state-transition model, we mean a model that summarizes the transition of the environment and the agent, either due to intrinsic or extrinsic signals. We describe transition models that comply with the stationarity assumption and incorporate relevant aspects of the dynamics of the environment, of the agent, together with their interaction, for a given coverage task. Our instantiations take place in static environments and illustrate *control models* that spans over: **deterministic control** in which action outcomes are predictable with certainty; **nondeterministic control** in which action outcomes can be constrained; and **stochastic control** in which action outcomes are probabilistically quantified.

Hereafter, we report formalisms of SDM that are suitable for deterministic, nondeterministic, and stochastic state-transition models. Then, we instantiate a decision-making model of coverage at the agent's knowledge level for each formalism.

2.3 Decision-making models for coverage

The case of deterministic control

In a completely observable and a *deterministic control* world, there is no uncertainty regarding the outcome of an agent's action. This outcome is unique; moreover, if the initial state is known then every outcome is also known in advance before any execution takes place. The Completely Observable (State) Search Problem (COSP) formalism is suitable for such worlds; its definition is reproduced from [Russell and Norvig, 1995] in Definition 2.1.

Definition 2.1 (Completely Observable (State) Search Problem (COSP)). *A Completely Observable (State) Search Problem can be defined as a tuple representing its core elements:* $\langle\ s_0$, *ApplicableAction, TransitionModel, GoalTest, StepCost* $\rangle$ *where* s_0 *represents the initial state;* $ApplicableAction(s)$ *returns a set of actions available to the agent for a given state* s*;* $TransitionModel(s, a)$ *outputs a singleton containing state* s' *attained from* s *by action* a*;* $GoalTest(s)$ *checks whether a given state* s *is a goal state; and* $StepCost(s, a, s')$ *evaluates the cost of applying action* a *from state* s *and transitioning to state* s'*.*

This formalism assumes that the state is completely observable by the agent during execution, that the agent knows the initial state (s_0), the applicable actions from any state s ($ApplicableAction(s)$), and the successor state ($\{s'\} = TransitionModel(s, a)$) for any pair of predecessor state s and action a. These elements alone (s_0, $ApplicableAction$, and $TransitionModel$) implicitly define the reachable state space S_{s_0}, *i.e.*, truly attainable states from s_0.

In this deterministic setup, the agent knows and controls everything. In long-term planning, a complete solution to the COSP is an optimal path through the reachable state space (S_{s_0}) starting at s_0 and ending at any state s' that validates the predicate $GoalTest(s')$. The long-term value of any satisfiable path is usually evaluated by the sum of its step costs. Optimal paths are satisfiable solution paths that minimize such value. We study long-term planning for deterministic coverage in Chapter 4.

Now, we report a first domain of coverage and its decision-making model.

2.3.1 Coverage model with deterministic control

As a reminder, in Single-Robot Active Coverage, the robot is tasked to find a path that visits some locations in order to remotely view the entire environment during execution. Moreover, the robot's positions are reliable at any time and the environment's map is initially revealed to the robot. We describe a first instantiation of a deterministic coverage domain that will serve to:

1. Support coverage with nondeterministic and stochastic control in Sections 2.3.2 and 2.3.3;
2. Support exploration with deterministic control and sensing in Chapter 3; and
3. Benchmark long-term planning algorithms in Chapter 4.

From an external point of view, in our coverage domains, the world's state represents the environment's map and the robot's pose. Additionally, the robot is equipped with a remote sensing device to view locations from afar. In coverage, as the map is known, the ground truth local observations are known from every location as well. In this chapter, we assume unbiased remote sensing [4], consequently, the robot's observations during execution will perfectly match the ground truth observations.

An instance of this first coverage domain could represent the environment as an occupancy grid denoted by $map : (rows, cols, occupancy)$ which is defined as a number of rows, a number of columns, and the occupancy of each cell among $\{free, occupied\}$. Additionally, an agent denoted by $robot : (pos, range)$ could be defined by its position on the map and the radius of its circular field of view (used to cover surrounding cells). Such an instance will be denoted by $cov_{det}(map, robot)$.

From the agent's point of view, as this first coverage domain assumes deterministic robot's moves, the decision process can be formalized as a COSP over knowledge states. A knowledge state encapsulates the world's state (environment's occupancy map and the robot's pose at any time), and the robot's coverage map. This coverage map accumulates the observations made from every location visited during execution. It informs the agent regarding the locations that are already covered and those that are not covered yet. The planning agent can rely on the following model to maintain the most complete information to support decision-making. This model is similar to the off-line coverage task described in [Li et al., 2012], and inspired by the deterministic vacuum-cleaning task in [Russell and Norvig, 1995]. Formally, this first coverage model instantiates a *search problem* as follows:

$\mathbf{s_0} : (\mathbf{p_0}, \mathbf{m_0}, \mathbf{c_0})$, the initial state corresponds to the robot position $p_0 \in P = [1..map.rows] \times [1..map.cols]$; the deterministic occupancy map m_0 with cell values in the set $\{f\ (free), \neg f\ (occupied)\}$; and the deterministic coverage map c_0 with cell values in the set $\{c\ (covered), \neg c\ (not\ covered)\}$. From now on, s_p, s_m, and s_c denote the position, the deterministic occupancy map, and the deterministic coverage map of any state s.

ApplicableAction(s), returns a subset of moving actions, $A = \{\Uparrow, \Rightarrow, \Downarrow, \Leftarrow\}$, where actions lead from the current position s_p to a neighboring position on the grid.

TransitionModel(s,a), deterministically outputs the next state, s', for a given action a applied in state s by querying s_m. First, it updates the pose, s_p, according to the action a and the

[4] We study biased remote sensing in the more general case of exploration in Chapter 3

Figure 2.1: This figure illustrates the deterministic position outcome p' ($\bullet$) obtained after applying action up ($\Uparrow$) from position p ($\bigcirc$) on any local occupancy map m_{loc}. Formally, we illustrate the position p' of the successor state from the predecessor s and the action a: $s' = (p', m_{loc}, c') = TransitionModel(s = (p, m_{loc}, c), \Uparrow)$. The example on the left represents a local occupancy map with two cells that are free of obstacles (white); the robot stands on the bottom cell ($\bigcirc$) and ends up in the top cell ($\bullet$) by moving up ($\Uparrow$) . On the right example, the top cell is occupied by an obstacle (black), hence the robot remains in the bottom cell when deciding to move up. Similar patterns can be obtained for other moving directions under rotational symmetry.

occupancy map s_m (Figure 2.1); then it updates the coverage map, s_c, by marking newly visible cells within *robot.range* as *covered*. $F(s_p, s_m)$ denotes the cells within the viewing range from the robot's position s_p in the map s_m.

GoalTest(s), returns *true* if the coverage map, s_c, is equal to the goal coverage map c_G. The *goal coverage map* is the function that represents all coverable cells in the environment. It is obtained by firstly identifying the cells that are attainable from p_0 by the moving actions. Formally, this set is denoted by P_A^{max}, it corresponds to the maximal connected component of free cells of m_0 that contains p_0 and whose connectivity depends on the actions. Secondly, the cells that are visible from any attainable cell are labeled as covered and the cells that are occluded from all attainable cells are labeled as not covered. Similarly, C_A^{max} corresponds to the set of cells that are visible from P_A^{max} within the field of view F.

Formally:

$$c_G(p) = \begin{cases} c & \text{if } p \in C_A^{max} \\ \neg c & \text{else } (p \in P \setminus C_A^{max}) \end{cases}, \tag{2.1}$$
$$C_A^{max} = \bigcup_{p \in P_A^{max}} F(p, m_0).$$

StepCost(s,a,s'), is the immediate cost and is defined below according to an optimization criterion. No cost is incurred in a goal knowledge state; and otherwise, a first term penalizes the number of decision steps and a second term rewards newly covered cells between two states s and s' [5]. Note that, if we compare different trajectories that all reach the goal by summing their step costs, these trajectories will differ only in their number of decision steps (this is similar to the Minimum Coverage Path (MCP) discussed in Chapter 1). Consequently, the agent could prefer trajectories with less decision steps. However, if we compare trajectories of the same length that do not reach the goal, these trajectories will differ in their number of covered cells. Hence, the agent could favor

[5]The notation, $f^{-1}(A)$, is used to query the preimage of a set A under the application f, however, when A is a singleton such that $A = \{a\}$ we will simply write $f^{-1}(a)$.

trajectories that allow to cover more cells.

Formally:

$$StepCost(s, a, s') = \begin{cases} 0 & \text{if GoalTest(s)} \\ 1 - |s'^{-1}_c(c) \setminus s^{-1}_c(c)| & \text{otherwise.} \end{cases} \tag{2.2}$$

This first instantiation is reported in Model 1 and supported by the primitives given in Section A.1. The full state space of the instance, $cov_{det}(map : (2, 2, \{\{(1,1),(1,2),(2,1)\} \rightarrow \{f\}, \{(2,2)\} \rightarrow \{\neg f\}\}), robot : ((1,1),0))$, is illustrated in Figure 2.2. Following the notation, this coverage instance takes place in a 2 rows $\times$ 2 columns occupancy grid, whose bottom right cell at $(2,2)$ is occupied by an obstacle $(\neg f)$ whereas the other cells are free of obstacles (f); the robot is on the top left cell at $(1,1)$ and covers only its current cell at any moment (within a range of 0).

Model 1 Model of the coverage task as a COSP

function *ApplicableAction*(s)
▷ Select the actions available from the given state s
$A \leftarrow \{\}$
for $\langle a, \delta_p \rangle \in \{\langle \Uparrow, (-1,0)\rangle, \langle \Downarrow, (+1,0)\rangle, \langle \Leftarrow, (0,-1)\rangle, \langle \Rightarrow, (0,+1)\rangle\}$ **do**
if $isfree(s_m, s_p)$ **then** $A \leftarrow A \cup \{a\}$
return A

function $TransitionModel(s, a)$
▷ Update the ground truth map m, the robot position p, and the coverage map c of the current state s according to action a
$s' \leftarrow s$; $s'_m \leftarrow updatemap(s', a)$; $s'_p \leftarrow updatepose(s', a)$; $s'_c \leftarrow updatecoverage(s', a)$
return {s'}

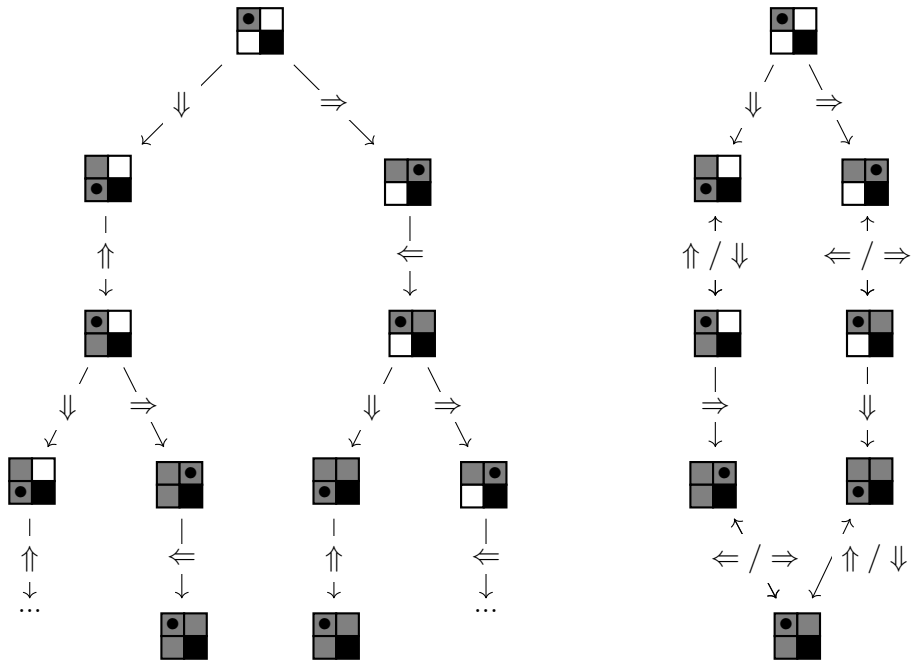
function *GoalTest*(s)
▷ Check if the coverage map c of the given state s is the goal coverage map c_G
return $s_c = c_G$

function *StepCost*(s, a, s')
▷ Compute the cost of applying action a in state s and reaching state s'
if *GoalTest*(s) **then**
return 0
return $1 - |s'^{-1}_c(c) \setminus s^{-1}_c(c)|$

We formalized the planning agent for Active Coverage with unbiased remote sensing and deterministic control. Now, we study the case of nondeterministic control.

The case of nondeterministic control with unquantified uncertainty

In a completely observable and *nondeterministic* control world, there may be uncertainty regarding the outcomes of an agent's action. More precisely, any agent's action can now have



(a) OR Tree **(b)** Or Graph

Figure 2.2: This figure illustrates the knowledge-state space of an instance of the coverage task with full observability and deterministic control denoted by $cov_{det}(map : (2, 2, \{\{(1, 1), (1, 2), (2, 1)\} \rightarrow \{f\}, \{(2, 2)\} \rightarrow \{\neg f\}\}), robot : ((1, 1), 0))$. This instance takes place on a 2 by 2 occupancy grid whose bottom right cell is occupied, and where the robot starts in the top left cell and covers only its current cell at any time step. Its state space is represented as an Or tree on the left in Figure 2.2a and as an Or graph on the right in Figure 2.2b. Every node encapsulates an agent's knowledge state: the background layer of colors represents free and occupied cells in white and black, the middle layer represents the covered cells in gray; and the foreground layer represents the robot as the •. The root node at the top represents the initial knowledge state, whereas the nodes below it with all covered cells (in gray) represent the goal knowledge states of the agent. Every arc is labeled with the action that activates a deterministic transition from one predecessor state to one successor state.

many potential outcomes. The transition model of a COSP (Definition 2.1) cannot take this uncertainty into account and thus needs to be extended. We report a second formalism of SDM that is suitable in such worlds and its instantiation as another domain of coverage.

The state-transition model of the COSP must be extended to output a set of states instead of a single state as seen previously. This is necessary to account for *contingencies* that arise during planning, but it is insufficient to *quantify uncertainty* about the outcomes of any state-action pair. The Completely Observable Contingency Problem (COCP) formalism is suitable for this purpose; its definition is reproduced from the chapter 4 of [Russell and Norvig, 1995] in Definition 2.2.

Definition 2.2 (Completely Observable Contingency Problem (COCP)). *A Completely Observable Contingency Problem is represented as a tuple: ⟨ s_0, ApplicableAction, TransitionModel', GoalTest, StepCost ⟩. It is an extension to the Search Problem (SP) where the transition model, TransitionModel'(s,a), outputs a set of states to account for nondeterministic actions.*

This formalism assumes that the agent observes the state during execution; and knowns the initial state, the actions available from any state, and its potential successor states ($s' \in TransitionModel(s, a)$) for any pair of predecessor state s and action a. Again, these elements implicitly define the reachable state space S_{s_0}, *i.e.* potentially attainable states from s_0.

In this nondeterministic setup, the agent has no control over the contingencies. In long-term planning, a complete solution to the COCP is a *conditional plan* and not a trajectory as for the COSP. This plan represents the behavior of the agent during execution according to the real outcome of any contingency. A common assumption is to consider that contingencies are controlled by an opponent. This leads to the pessimistic optimization criterion often used in game playing. This best worst-case criterion states that the agent's behavior should maximize its own profit by considering an optimal opponent strategy. However, in extreme cases, the opponent can force the agent to cycle forever in the state space; in such cases it is necessary to arbitrate the opponent's control over the contingencies.

2.3.2 Coverage model with nondeterministic control

We describe a second instantiation of a nondeterministic coverage domain that will serve to:

1. Support coverage with stochastic control in Section 2.3.3; and
2. Support exploration with nondeterministic control and sensing in Chapter 3.

To illustrate uncertainty in actions outcomes, we define an extension of the deterministic coverage task where a robot is looking for a conditional plan to patrol a slippery environment in order to complete a coverage map. This instantiation is inspired by the nondeterministic vacuum-cleaning task in chapter 4 of [Russell and Norvig, 1995]. It will be denoted by $cov_{nondet}(map, robot)$ and modeled as a *contingency problem* as follows.

TransitionModel'(s,a), outputs a set of next states according to the slippery conditions: when the robot decides to move *up*, he can attain one of the *top, left, right, or center* cells (Figure 2.3). The same pattern applies to the other moving directions under rotational symmetry.

Other items, are the same as those that appear in the instantiation of cov_{det} (Section 2.3.1).

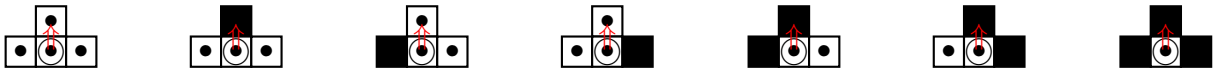


Figure 2.3: This figure illustrates any nondeterministic position outcome p' (•) that can be obtained after applying action up (⇑) from position p (◯) on any local occupancy map m_{loc}. Formally, we illustrate the position p' of any potential successor state s' from the predecessor state s and the action a: $s' = (p', m_{loc}, c') \in TransitionModel'(s = (p, m_{loc}, c), \Uparrow)$. The leftmost example represents a local occupancy map with four cells that are free of obstacles (white); the robot stands on the bottom cell (◯) and ends up in any of the top, left, bottom, and right cell (•) by moving up (⇑). Similar patterns can be obtained for other moving directions under rotational symmetry.

This second instantiation is reported in Model 2, the others elements of the instantiation are the same as those that appear in Model 1. The complete state space of the instance, $cov_{nondet}(map : (2, 2, \{\{(1, 1), (1, 2), (2, 1)\} \rightarrow f, \{(2, 2)\} \rightarrow \neg f\}), robot : ((1, 1), 0))$, of the slippery coverage task is represented in Figure 2.4. This coverage instance takes place in a 2 rows by 2 columns occupancy grid, whose bottom right cell at (2,2) is occupied by an obstacle whereas the other cells are free of obstacles; the robot stands on the top left cell at (1,1) and covers only its current cell (within a range of 0).

Model 2 Model of the coverage task as a COCP

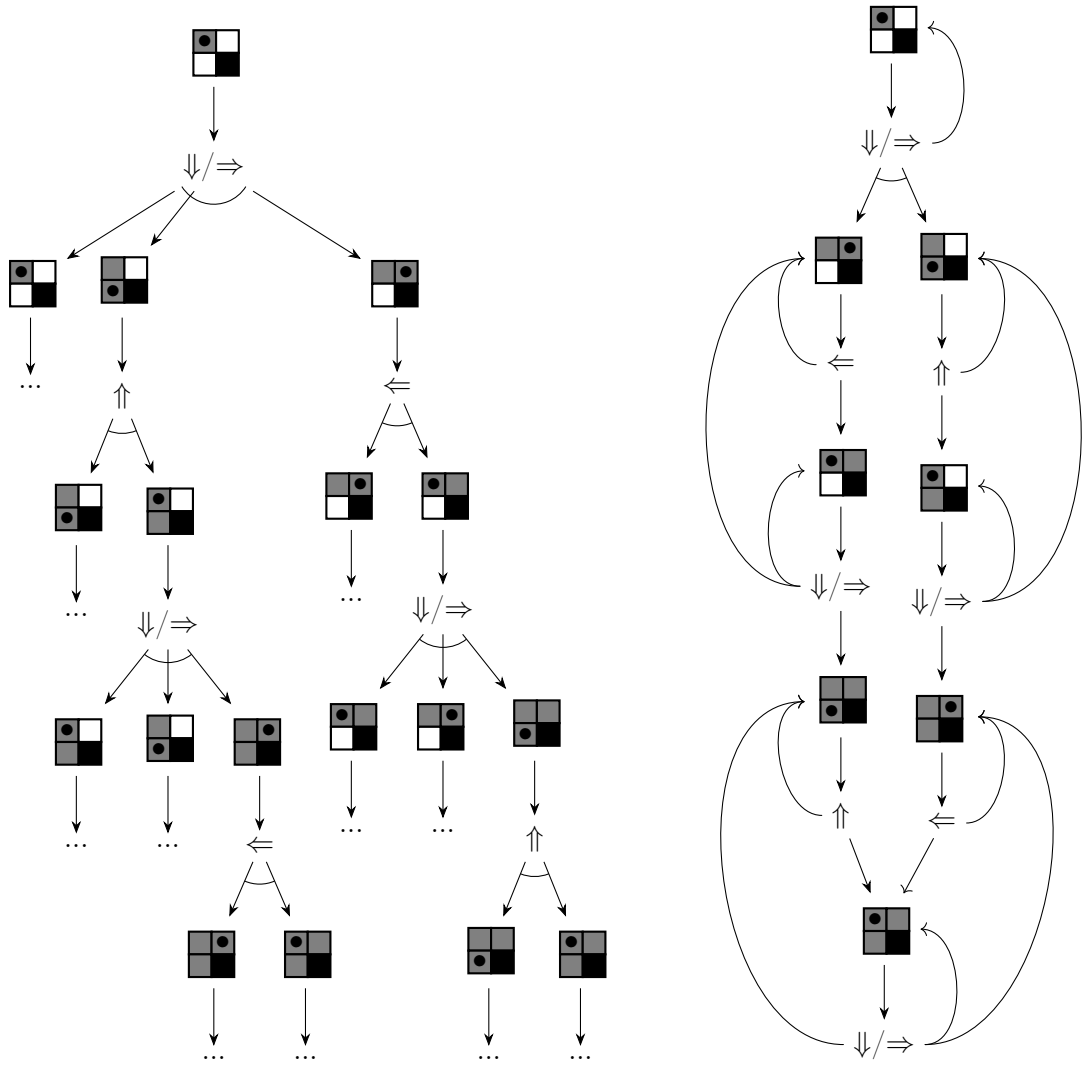
function $TransitionModel'(s,a)$
▷ Select other actions to emulate a nondeterministic process
$actions \leftarrow \{a\}$
if $a \in \{\Uparrow, \Downarrow\}$ **then** $actions \leftarrow actions \cup \{\Leftarrow, \Rightarrow\}$
if $a \in \{\Leftarrow, \Rightarrow\}$ **then** $actions \leftarrow actions \cup \{\Uparrow, \Downarrow\}$
▷ Compute the set of reachable states from s with action a
$S' \leftarrow \bigcup_{a \in actions} TransitionModel(s, a)$
$S' \leftarrow S' \cup \{s\}$
return S'

We generalized the planning agent for Active Coverage to nondeterministic control. Now, we consider the case of stochastic control.

The case of stochastic control with quantified uncertainty

In a completely observable and *stochastic* control world, there is uncertainty regarding the outcomes of an agent's action, moreover it is assumed that some outcomes occur more often than others. More precisely, any agent's action has many potential outcomes as in the previous nondeterministic model, but this time such outcomes must be quantified. While the COCP formulation accounts for nondeterministic actions, it does not quantify uncertainty of action outcomes explicitly. We report a third formalism of SDM that is suitable under such conditions and its instantiation as a last domain of coverage.

The state-transition model of the COCP must be extended to account for the likelihood of the agent's action outcomes. It must describe contingencies that arise during planning and



(a) And-Or Tree **(b)** And-Or Graph

Figure 2.4: This figure illustrates the knowledge-state space of an instance of the coverage task with full observability and nondeterministic control denoted by $cov_{nondet}(map : (2, 2, \{\{(1, 1), (1, 2), (2, 1)\} \rightarrow \{f\}, \{(2, 2)\} \rightarrow \{\neg f\}\}), robot : ((1, 1), 0))$. This instance takes place on a 2 by 2 occupancy grid whose bottom right cell is occupied, and where the robot starts in the top left cell and covers only its current cell at any time step. Its state space is represented as an And-Or tree on the left in Figure 2.4a and as an And-Or graph on the right in Figure 2.4b. Every node encapsulates a knowledge state of the agent: the background layer contains free and occupied cells in white and black; the middle layer contains covered cells in gray; and the foreground layer contains the robot's position (the •). The root node representing the initial state of the world is at the top, and the goal nodes are easily identified with all covered cells (in gray). Every arc is labeled with the action that activates a nondeterministic transition from one predecessor state to many potential successor states.

quantify the occurrence of every potential state outcome for a given pair of state and action. The Completely Observable Markov Decision Process (COMDP) formalism uses probabilities over the state outcomes of any state-action pair in its state-transition model. Its definition appears for example in chapter 17 of [Russell and Norvig, 1995], and in chapter 2 of [Puterman, 2014]; it is reproduced in Definition 2.3. The general structure of a Completely Observable Markov Decision Process (COMDP) is represented as a completely observable dynamic influence diagram in Figure 2.5.

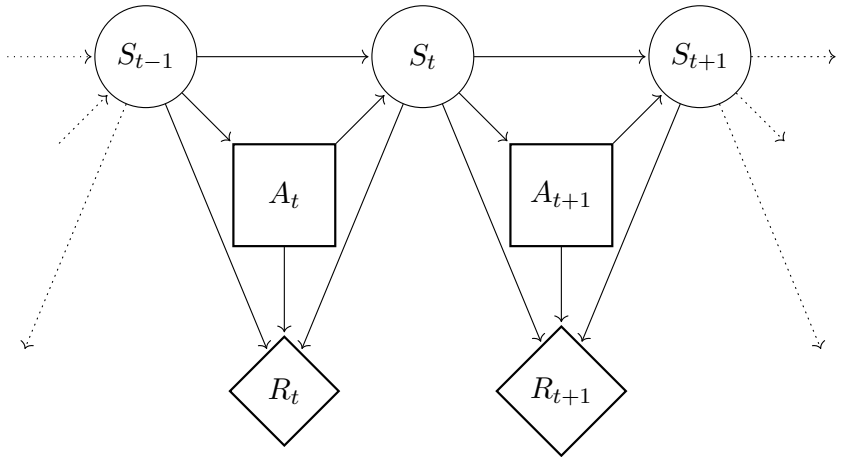


Figure 2.5: One slice of the decision process for a Completely Observable Markov Decision Process (COMDP) with two decision steps. Decision variables are represented as squares, utility nodes as diamonds, and state nodes as circles. S_t denotes the random variable that represents the state of the process at time t; A_t denotes the random variable that represents the chosen action at time t, this action depends on the available actions from S_{t-1}; R_t denotes the random variable that represents the reward obtained at time t, this reward generally depends on the triplet S_{t-1}, A_t, and S_t. The *Markovian* property assumes that the probability of transitioning to S_{t+1} depends only S_t and A_t.

Definition 2.3 (Completely Observable Markov Decision Process (COMDP)). *A Completely Observable Markov Decision Process is represented as a tuple* $\langle$ S, A, T, R, s_0 $\rangle$. s_0, *is the initial state of the process;* S *and* A *are the finite sets of states and actions;* $T : S \times A \times S \to [0, 1]$, *is a transition model that encodes the dynamics of the process, i.e.,* $T(s, a, s') = Pr(S_{t+1} = s' \mid S_t = s, A_{t+1} = a)$ *is the probability of transitioning to state,* s', *from state* s *after performing action* a*; and* $R : S \times A \times S \to \mathbb{R}$ *is a reward function, i.e.,* $R(s, a, s')$ *is the reward the agent receives when transitioning to state* s' *from state* s *after applying action* a.

This formalism describes a process indexed by discrete time (t), with finite sets of states and actions (S and A), and a known initial state (s_0). It assumes that the agent completely observes the state during execution. Moreover, T describes a stochastic control process that outputs the probability of a state s' at time $t + 1$ ($T(s, a, s') = Pr(S_{t+1} = s' \mid S_t = s, A_{t+1} = a)$) for any pair of predecessor state s at time t and action a at time $t + 1$. It is termed memoryless as it complies to the *Markovian property*, which states that the distribution of future states depends only on the current state.

In this stochastic setup, the agent has no control over the contingencies but is able to quantify their outcomes. In long-term planning, a complete solution to a COMDP is a *policy* that describes

the agent behavior in any state. A common assumption is to consider that contingencies are controlled by nature in a fair way as dictated by their probability. This leads to the average optimization criterion often used in decision theory. This best average-case states that the agent has to take a decision that maximizes the long-term expected reward from any state.

2.3.3 Coverage model with stochastic control

We describe a third instantiation of a stochastic coverage domain that will serve to:

1. Discuss the limitation of flat representations of this process; and then
2. Support exploration with stochastic control in Chapter 3.

To illustrate quantified action outcomes, we extend the nondeterministic coverage task, assuming that the slippery properties of the environment are given as in the slippery vacuum world task of chapter 17 in [Russell and Norvig, 1995]. To model cov_{sto}, a stochastic coverage task where the outcomes of actions are uncertain but probabilistically quantified, we instantiate the elements of a COMDP as in [Yehoshua et al., 2015] except that we do not consider an adversary but slippery properties of the grid:

$\mathbf{s_0}$: $(\mathbf{p_0}, \mathbf{m_0}, \mathbf{c_0})$, the initial state represents the initial robot position, the initial occupancy map, and the initial coverage map.

S, the state space is represented as the Cartesian product, $P \times M \times C$, of three features: the position space $P = [1..rows] \times [1..cols]$; the occupancy map space $M = \{m_i \mid m_i : P \rightarrow \{f, \neg f\}\}$; and the coverage map space $C = \{c_i \mid c_i : P \rightarrow \{c, \neg c\}\}$. To clarify, any occupancy map $m_i \in M$ is an application from P to $\{f, \neg f\}$, where f means *free of* obstacles and $\neg f$ means *occupied by* obstacles. Similarly, any coverage map $c_i \in C$ is an application from C to $\{c, \neg c\}$, where c or $\neg c$ respectively mean *already or not yet* covered by the robot's field of view.

A, the action set allows the robot to move in the four cardinal directions $\{\Uparrow, \Rightarrow, \Downarrow, \Leftarrow\}$.

T(s,a,s'), the transition model incorporates the quantified uncertainty as follows: a chosen action has its deterministic effect 70% of the time, otherwise the robot stochastically moves sideways 20% of the time (10% on each side) or stays in place (10%), as illustrated in Figure 2.6.

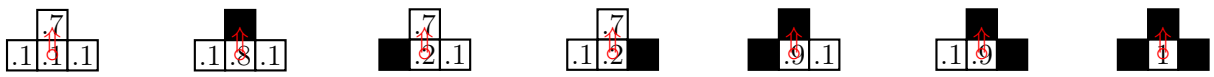


Figure 2.6: This figure illustrates the probability for obtaining any nondeterministic position outcome p' after applying action up ($\Uparrow$) from the bottom center cell p on any local occupancy map m_{loc}. More precisely, that is the probability of reaching the position p' of any potential successor state s' from the predecessor state s and the action a: $T(s, a, s') = Pr(s' = (p', m_{loc}, c') | s = (p, m_{loc}, c), a = \Uparrow)$. The leftmost example represents a local occupancy map with four cells that are free of obstacles (white); the robot stands on the bottom cell and respectively ends up in the top cell or any of the left, bottom, or right cell, with probability 0.7 or 0.1, by moving up ($\Uparrow$). Similar patterns can be obtained for other moving directions under rotational symmetry.

R(s,a,s'), the reward function takes the opposite of the step cost introduced in Section 2.3.1, $R(s, a, s') = -StepCost(s, a, s')$, to obtain a maximization problem.

This third instantiation is reported in Model 3, the other elements are the same as those described in Model 1. We generalized the planning agent for coverage to stochastic control. The question that remains after this instantiation is: How to properly define the state space S? Indeed, it is explicitly considered in the COMDP formalism whereas it is implicit in both the COSP and COCP formalisms. Moreover, what is the size of T and R in case one relies on a flat specification of these matrices? [6]

Model 3 Model of the coverage task as a COMDP

function $T(s,a)$
▷ Select other actions to emulate a stochastic process
$actions \leftarrow \{a\}$
if $a \in \{\Uparrow, \Downarrow\}$ **then** $actions \leftarrow actions \cup \{\Leftarrow, \Rightarrow\}$
if $a \in \{\Leftarrow, \Rightarrow\}$ **then** $actions \leftarrow actions \cup \{\Uparrow, \Downarrow\}$
▷ Compute the probabilities to reach s' from s with action a
$\forall s' \in S, T_{sa}(s') \leftarrow 0$; $T_{sa}(s) \leftarrow 0.1$
for $action \in actions$ **do**
$inc \leftarrow 0.1$
if $action = a$ **then** $inc \leftarrow 0.7$
$s' \leftarrow TransitionModel(s, action)$; $T_{sa}(s') = T_{sa}(s') + inc$
return T_{sa}

function $R(s, a, s')$
▷ Uses the opposite of the step cost for applying action a in state s and reaching state s'
return $-StepCost(s, a, s')$

If we consider a flat specification of this COMDP for a given size of the environment (known width w and height h), the state space has a cardinality of $|S| = |P \times M \times C| = |P|2^{2|P|}$ and the transition and reward matrices are each of size $|T| = |R| = |A||S|^2$, with $|P| = w \times h$, $|M| = |\{f, \neg f\}|^{|P|}$, and $|C| = |\{c, \neg c\}|^{|P|}$. The cardinality of S is thus exponential in the cardinality of P. So, in the worst case, fully specifying a COMDP for any instance $cov_{sto}(map : (2, 2, \cdot), robot : (\cdot, \cdot))$ in a 2×2 grid with only 4 actions (independently of the map occupancy; the robot pose and its range), requires the specification of 2^{10} states and 4 transition and reward matrices with 2^{20} elements each.

In addition to the storage requirements necessary to specify this COMDP, it is known that long-term planning can be achieved in polynomial time in the size of S, A, and the number of bits used to represent each matrix element [Littman et al., 1995]. However, as the cardinality of S is exponential in the size of P, it is worth thinking about how to restrict S prior to implementing a solver for this COMDP. A more compact state space can be obtained by including prior information or interaction constraints within or between state space features (*e.g.*, a belief network). Furthermore, the transition model can be simplified according to conditional independence (*e.g.*,

[6] Another possibility is to compute on demand the probability T and reward R associated to each triplet of predecessor state, action, and next state.

a Bayesian network) of this factorized influence diagram. In the following, we significantly reduce the flat specification with a given initial state and simple interaction constraints of the knowledge state features.

Restricting the state space with initial information

In the following, we specify the coverage task with stochastic control as a COMDP. In this flat specification, the initial state denoted by $s_0 = (p_0, m_0, c_0)$ and some interaction constraints among the state features are taken into account. This allows restricting the full state space S to a subset of states $S_{s_0} \subseteq S$ that are attainable from s_0.

$\mathbf{P_{s_0}}$, the *position set* contains all positions that are attainable from p_0, it represents the maximal connected component of cells that are linked to p_0 by a sequence of moving actions: [7]

$$P_{s_0} = P \cap P_A^{max} = P_A^{max}.$$

$\mathbf{M_{s_0}}$, the *deterministic occupancy map set* contains only m_0, that is the initial occupancy map which does not change with time according to the static world assumption:

$$M_{s_0} = M \cap \{m_0\} = \{m_0\}.$$

$\mathbf{C_{s_0}}$, the *deterministic coverage map set* contains coverage maps defined over the domain of positions:

$$C_{s_0} = \left\{ c_i \mid c_i(p) \in \begin{cases} \{c\} & \text{if } p \in c_0^{-1}(c) \\ \{c, \neg c\} & \text{else if } p \in C_A^{max} \setminus c_0^{-1}(c) \\ \{\neg c\} & \text{else } (p \in P \setminus C_A^{max}) \end{cases} \right\}.$$

The cardinality of S_{s_0} is now trivially upper-bounded by $|P|2^{|P|}$ given the initial state s_0. Therefore, representing S_0 for a 2×2 grid with four actions requires at most 2^6 states, and eight transition and reward matrices with at most 2^{12} elements each for any given initial state.
Specifically, for the initial state $s_0 = (p_0, m_0, c_0)$ given below, the state space S_{s_0} is even smaller as illustrated in Figure 2.7. Below, we detail several restrictive assumptions that lead to this compact state space: inconsistent coverage maps, inconsistent robot locations, and the grouping of goal states.

$$p_0 = (1,1),\ m_0(p) = \begin{cases} \neg f & \text{if } p = p_3 = (1,2) \\ f & \text{else } (P \setminus \{p_3\}) \end{cases},\ c_0(p) = \begin{cases} c & \text{if } p = p_0 \\ \neg c & \text{else } (P \setminus \{p_0\}) \end{cases}.$$

Indeed, the position space contains three positions, the occupancy map set is a singleton, and the coverage map set only fixes the coverage status of the initial position. Below, the following notation, $f_{|A}$, is used for restricting the domain of f to the set A.

$$\begin{aligned} P_{s_0} &= \{p_0 = (1,1), p_1 = (2,1), p_2 = (2,2)\}, \\ M_{s_0} &\leftarrow M_{s_0|P_{s_0}} = \{m_0^* : P_{s_0} \to \{f\}\}, \\ C_{s_0} &\leftarrow C_{s_0|P_{s_0}} = \left\{ c_i \mid c_i(p) \in \begin{cases} \{c\} & \text{if } p \in \{p_0\}, \\ \{c, \neg c\} & \text{else } (p \in \{p_1, p_2\}) \end{cases} \right\}. \end{aligned}$$

[7] As a reminder, P_A^{max} is the set of cells that are attainable from p_0 by several applications of the moving actions in m_0.

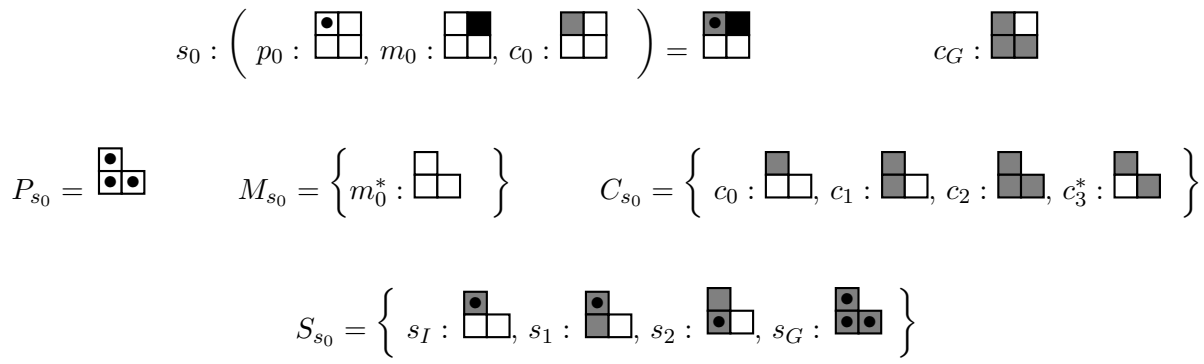


Figure 2.7: This figure illustrates the set of knowledge states of an instance of the coverage task with full observability and stochastic control denoted by $cov_{sto}(map : (2, 2, \{\{(1, 1), (2, 1), (2, 2)\} \rightarrow \{f\}, \{(1, 2)\} \rightarrow \{\neg f\}\}), robot : ((1, 1), 0))$. This instance takes place on a 2×2 occupancy grid whose top right cell is occupied, and where the robot starts in the top left cell and covers only its current cell at any time step. If the initial state s_0 is provided as a prior information: the deterministic goal coverage c_G can be extracted, and the sets of reachable positions P_{s_0}, potential occupancy maps M_{s_0}, and potential coverage maps C_{s_0} can be restrained. For instance, P_{s_0} contains only the top left and bottom cells, M_{s_0} is a singleton set of occupancy maps as we assumed static environments, and C_{s_0} contains three consistent coverage maps, where covered cells are connected, and one inconsistent coverage map. Hence, the state space S_{s_0} reachable from s_0 is obtained by discarding the elements with an asterisk (constant or inconsistent elements) and merging the goal knowledge states into s_G.

However, among the four possible *deterministic coverage maps* in $C_{s_0|P_{s_0}} = \{c_0, c_1, c_2, c_3^*\}$ given below, c_3^* is inconsistent (with a single robot, the covered cells must be connected by A). Hence, it discards the three following states: $(\{p_1, p_2, p_3\}, m_0^*, c_3^*)$.

$$c_0(p) = \begin{cases} c & \text{if } p \in \{p_0\} \\ \neg c & \text{ot.} \end{cases}, \; c_1(p) = \begin{cases} c & \text{if } p \in \{p_0, p_1\} \\ \neg c & \text{ot.} \end{cases},$$

$$c_2 : \{p_0, p_1, p_2\} \to \{c\}, \; c_3^*(p) = \begin{cases} c & \text{if } p \in \{p_0, p_2\} \\ \neg c & \text{ot.} \end{cases}.$$

Additionally, the robot cannot be on cells that are *not covered* yet. Thus, it discards the three following states: $(\{p_1, p_2\}, m_0^*, c_0)$ and (p_2, m_0^*, c_1).

Finally, we do not require the robot to finish in a particular *position*, so the following three goal states $(\{p_1, p_2, p_3\}, m_0^*, c_2)$ are merged into one unique goal state s_G.

Thus, $|S_{s_0}| = 4$ states, each action transition and reward matrix definition requires 16 elements, and the flat specification is as follows.

$$\begin{aligned}
&S_{s_0} = \{s_0 = (p_0, m_0^*, c_0), (p_0, m_0^*, c_1), (p_1, m_0^*, c_1), s_G = (., m_0^*, c_2)\}, \\
&A = \{\Uparrow, \Rightarrow, \Downarrow, \Leftarrow\}, \\
&T_{s_0}(., \Uparrow, .) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & .7 & .2 & .1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; T_{s_0}(., \Rightarrow, .) = \begin{bmatrix} .9 & 0 & .1 & 0 \\ 0 & .9 & .1 & 0 \\ 0 & .1 & .2 & .7 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
&T_{s_0}(., \Downarrow, .) = \begin{bmatrix} .3 & 0 & .7 & 0 \\ 0 & .3 & .7 & 0 \\ 0 & 0 & .9 & .1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; T_{s_0}(., \Leftarrow, .) = \begin{bmatrix} .9 & 0 & .1 & 0 \\ 0 & .9 & .1 & 0 \\ 0 & .1 & .9 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
&R_{s_0}(., ., .) = \begin{bmatrix} -1 & - & 0 & - \\ - & -1 & -1 & - \\ - & -1 & -1 & 0 \\ - & - & - & 0 \end{bmatrix}.
\end{aligned}$$

This specification explicitly considers conditions to restrict the state space of the coverage task. Such conditions allow reducing the size of the matrix specification of the decision process in case one relies on flat representations as opposed to on-demand computations.
If possible, they should be used directly when defining the structure of the model as the size of these matrices quickly grows for bigger instances of the coverage task. We provide a factorized representation of the decision process for the stochastic slippery grid coverage task in Figure 2.8. A factorized transition model could be extracted from this network using properties such as conditional independence.

2.4 Conclusion

In this chapter, we reported three formalisms of SDM for completely observable domains. These formalisms differ in terms of their state-transition model: *deterministic*, *nondeterministic*, and *stochastic*. We reported three models of coverage using each of these formalisms. These models can be used as specifications and simulators of tasks in which an agent seeks to cover the environment. Moreover, they support both short-term planning using specific heuristics from experimental Robotics and long-term planning using general algorithms from SDM in Artificial Intelligence.

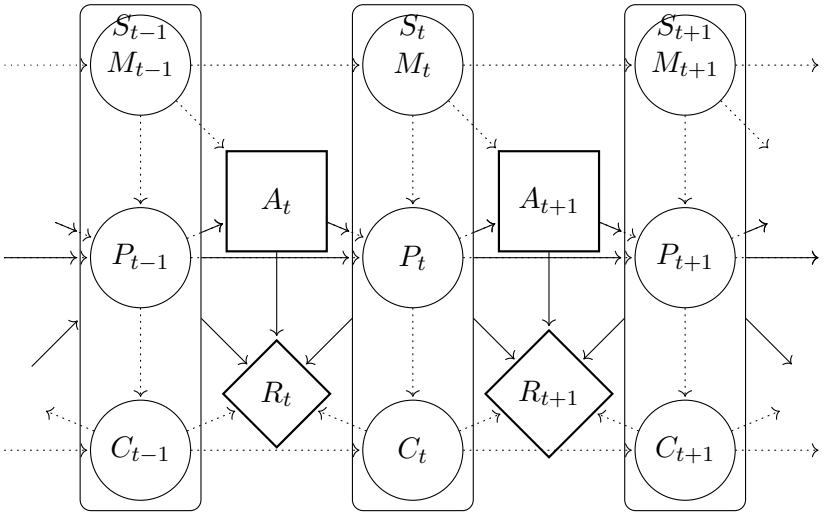


Figure 2.8: One slice of the influence diagram for the stochastic slippery grid coverage for a Factorized COMDP specification with two decision steps. Decision variables are represented as squares, utility nodes as diamonds and chance nodes as circles. Note that the state feature denoted by M_t is constant in static environments, which is the assumption for all our models. However, in dynamic environments, M_t could change over time, *e.g.* doors may be closed or opened at different time steps.

We tried to convey a rigorous mechanism for specifying Single-Robot Active Coverage tasks as SDM problems. This mechanism will apply again to specify Single-Robot Active Mapping tasks in Chapter 3. We will subsequently reuse the domain of Single-Robot Active Coverage to benchmark long-term solvers in Chapter 4. Next, we focus on the modeling of Active Mapping tasks.

In the next chapter, we report formalisms for hidden world's states. First, we use these formalisms over hidden world's states to instantiate exploration domains at a concrete level from the point of view of an external observer. Then, we motivate the necessity of an abstract level to represent the point of view of the agent. Finally, we report the translations from concrete to abstract level formalisms and instantiate decision-making models for the exploration task. These domains can support decision-making in active estimation problems that involve information gathering, such as Active Mapping.

Chapter 3

Partially Observable Formalisms: Models of Active Mapping

Contents

3.1 Introduction

In this chapter, we report several formalisms of Sequential Decision-Making (SDM) that are suitable to define domains of exploration. As a reminder, the map is not revealed a priori in an exploration task. We focus on formalisms for hidden world's states, where the world generates an observation that does not entirely reveal its state to the agent. We model domains for the exploration task that differ in terms of state-transition and state-observation models. These domains describe and simulate a process at the concrete level over hidden world's states in which an agent explores the environment. Then, we translate such processes over hidden world's states to observable agent's beliefs. These translations can support decision-making by

a planning agent for exploration.

Hence, we formalize three extensions of the planning agent for exploration (previously illustrated in Table C.2). Our extensions consider three types of control and sensing. In exploration, the ground truth local observations are not known in advance as the environment's map is initially hidden. Furthermore, we assume that the local observations made by the robot during execution can be biased. That is, the robot's remote sensor is imperfect.

Firstly, in Section 3.2, we introduce the concept of state-observation models that is compulsory in formalisms of SDM with a hidden world's state; and we report the generalization of each formalism introduced in Chapter 2 to partial observability. Secondly, in Section 3.3, we instantiate exploration domains at the world's state level, with deterministic, nondeterministic, and stochastic control and sensing. Thirdly, in Section 3.4, we motivate belief-based decision making to design planning agents for exploration. Then, in Section 3.5, we provide exploration models over agent's beliefs that are based on the domains of exploration over world's states. Finally, in Section 3.6, we conclude this chapter dedicated to instantiating domains and decision-making models for exploration.

3.2 Background: state-observation models

A state-observation model describes how a given world's state generates an observation to an agent. A robotic agent locally perceives the world's state through its sensors; this process of perception can be described using a state-observation model. For instance, let us consider a world in which the environment is a square room with colored walls and the agent is a robot fixed in the center of the room. The colors of the walls cannot change but the robot can change its orientation. Below, we assume that the agent faces the Northern wall and the walls are colored Green, Red, Green, and Red (in order North, East, South, and West) in the current state. Moreover, the robot is equipped with an inertial measurement unit which delivers its orientation (among North, East, South, and West), and a camera which identifies the color in the front (among Red and Green).

3.2.1 Modeling perfect sensors

In our example, with perfect sensors (calibrated imu and camera), the robot is able to observe its true orientation (North) and the true color of the wall (Green) when it faces the northern wall. [8] In terms of state-observation models, the current state of the world produces the north orientation and the green color measurements. In general, perfectly modeling a complex sensor consumes significant time and emulating it requires significant power in simulation. Hence, it may sometimes be beneficial to consider simple and lightweight state-observation models that roughly emulate a complex sensor. Moreover, even complex sensors are intrinsically imperfect as they are accurate only up to a certain degree.

3.2.2 Modeling imperfect sensors

In our example, with imperfect sensors (non calibrated imu and calibrated camera), the robot correctly identifies the color in the front but may sometimes incorrectly report its orientation

[8] In Chapter 2, we considered that the remote sensor of the robot was perfect. This remote sensor was implicitly modeled with an unbiased state-observation model. In the following, we will explicitly formalize it.

when it faces the northern green wall. In terms of state-observation models, the current world's state produces both the correct color and the incorrect orientation. Again, instead of perfectly modeling the inaccuracy of any particular sensor, we will emulate noisy sensors in a simple way: by generating a bias (random or not), adding it to the state, and observing the biased state.

In the following, when we refer to a state-observation model, we mean a model that summarizes the world's state perception by the agent, *i.e.*, the local and biased measurement of the world's state obtained by the agent. We define simple state-observation models that are either perfect or biased in different ways in order to support exploration. Our instantiations consider static environments and illustrate *sensor models* that span over: **deterministic sensing** in which measurement outcomes are predictable with certainty from a given state; **nondeterministic sensing** in which measurement outcomes are not quantified; and **stochastic sensing** in which measurement outcomes are probabilistically quantified.

First, let us briefly report formalisms that explicitly incorporate a state-observation model.

3.2.3 Generalizing to hidden states

We previously reported the Completely Observable (State) Search Problem (COSP), Completely Observable Contingency Problem (COCP), and Completely Observable Markov Decision Process (COMDP) formalisms for completely observable states and used them to instantiate coverage models at the knowledge state level in Chapter 2. Hereafter, we report the same formalisms when generalized to hidden states by incorporating an explicit state-observation model. Subsequently, we will use these formalisms to model domains of exploration. The following formalisms illustrate worlds with deterministic, nondeterministic, or stochastic state-transition models, and incorporate local and biased state-observation models.

Firstly, the COSP formalism has been reported to model a world with deterministic control and completely observable states. It can be generalized to hidden states as a Partially Observable (State) Search Problem (POSP) as reported in Definition 3.1.

Definition 3.1 (Partially Observable (State) Search Problem (POSP)). *A Partially Observable (State) Search Problem can be represented as a tuple* $\langle$ s_0, *ApplicableAction, TransitionModel, GoalTest, StepCost, ObservationModel*$^{(\prime)}$ $\rangle$. *ObservationModel*$^{(\prime)}(s, a, s')$ *returns the set of measurements available from a particular state* s' *attained from state* s *after applying action* a. *The other elements are those defined for the COSP formalism.*

This first formalism assumes that the world's state is partially observable through the agent's sensor during execution and that the process starts from an initial state (s_0) that may be hidden from the agent. The deterministic or nondeterministic state-observation model (*ObservationModel*$^{(\prime)}$) describes how the agent perceives any hidden world's state, it is formally defined in the next section. The remaining items [9] correspond to the COSP formalism.

Secondly, the COCP formalism has been reported to model a world with nondeterministic control and completely observable states, whose state-transition model did not set preferences among the

[9] In the general case, *StepCost*(s, a, s', z) can express the cost of starting in a state s, applying an action a, transitioning to a state s' and observing a measurement z. However, it is possible to define *StepCost*$'(s, a, s')$ instead by aggregating over all measurements z.

action outcomes. It can be generalized to hidden states as a Partially Observable Contingency Problem (POCP) as reported in Definition 3.2.

Definition 3.2 (Partially Observable Contingency Problem (POCP)). *A Partially Observable Contingency Problem is represented as a tuple* $\langle$ s_0, *ApplicableAction, TransitionModel', GoalTest, StepCost, ObservationModel*$^{(')}$ $\rangle$. *ObservationModel*$^{(')}(s, a, s')$ *returns the set of sensory inputs available from a particular state* s' *attained from state* s *after applying action* a. *The other elements are those defined in the COCP formalism.*

This second formalism extends the POSP to nondeterministic control; it makes the same assumptions regarding an initial state that may be hidden from the agent and a deterministic or nondeterministic state-observation model that describes the agent's perception. The remaining items correspond to the COCP formalism.

Thirdly, the COMDP formalism has been reported to model a world with stochastic control and completely observable states, whose state-transition model probabilistically quantified the state-action outcomes. It can be generalized to hidden states as a Partially Observable Markov Decision Process (POMDP) as reported in Definition 3.3.

Definition 3.3 (Partially Observable Markov Decision Process (POMDP)). *A Partially Observable Markov Decision Process is represented as a tuple* $\langle$ S, A, T, R, Z, O, s_0 $\rangle$. Z, *is a set of measurements or sensory inputs; and* $O : S \times A \times S \times Z \rightarrow [0, 1]$, *is a state-observation model.* $O(s, a, s', z) = \Pr(Z_{t+1} = z \mid S_t = s, A_{t+1} = a, S_{t+1} = s')$ *defines the probability of obtaining a measurement* z *from state* s' *after applying action* a *from state* s. *The remaining items are those defined in the COMDP formalism.*

This third formalism extends the POCP to quantified state-transition and state-observation models; it assumes, as well, that the initial state (s_0) may be hidden from the agent but quantifies any measurement outcome z at time $t+1$ ($O(s, a, s', z) = \Pr(Z_{t+1} = z \mid S_t = s, A_{t+1} = a, S_{t+1} = s')$) for any triplet of predecessor state s at time t, action a, and successor state s' at time $t + 1$. The general structure of a POMDP is represented as a partially observable dynamic influence diagram in Figure 3.1. POMDPs are discussed in greater details in chapter 6 of [Littman, 1996] and 2 of [Cassandra, 1998].

We briefly reported the POSP, POCP, and POMDP formalisms that generalize those defined in Chapter 2 to partial observability. Now, we formally define deterministic, nondeterministic, and stochastic state-observation models and instantiate an exploration domain for each of them in the aforementioned formalisms.

3.3 Domains of exploration

The case of deterministic sensing

A deterministic state-observation model always outputs a singleton $\{z_{sas'}\}$ = $ObservationModel(S_t = s, A_{t+1} = a, S_{t+1} = s')$ given the current world's state s at time t, a decision a, and a next state s' at time $t + 1$. Formally, the agent faces a deterministic sensing problem when $|ObservationModel^{(')}(s, a, s')| \leq 1$ for all input triplets.

Two particular cases of deterministic sensing worth mentioning are: unobservability when $ObservationModel(s, a, s') = \emptyset$ and observability when $ObservationModel(s, a, s') = \{s'\}$ for

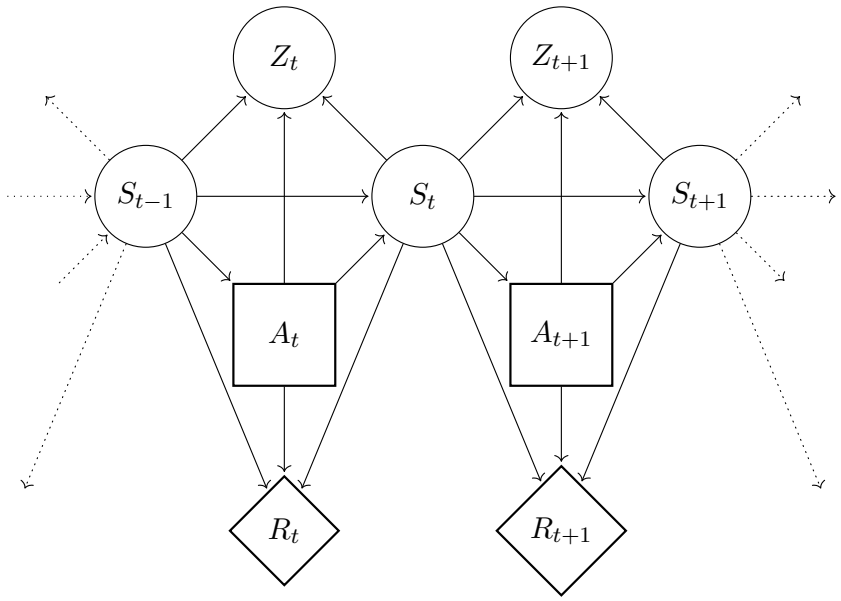


Figure 3.1: One slice of the decision process for a Partially Observable Markov Decision Process (POMDP) with two decision steps. Decision variables are represented as squares, utility nodes as diamonds and state nodes as circles. S_t denotes the state of the process at time t; A_t denotes the chosen action at time t, this action depends on the available actions from S_{t-1}; R_t denotes the reward obtained at time t, this reward generally depends on the triplet S_{t-1}, A_t, and S_t; Z_t denotes the random variable that represents the given measurement at time t, this measurement generally depends on the same triplet as R_t. The *Markovian* property assumes that the probability of transitioning to S_{t+1} depends only S_t and A_t.

all input triplets. In the following, we omit these two special cases, as the former falls out of our application scope whereas the latter has been addressed for coverage over knowledge states in Chapter 2.

Now, we can model an exploration task with deterministic control and sensing over world's states.

3.3.1 Exploration task with deterministic control and sensing

We describe a first domain of exploration with deterministic control and sensing. It will serve to:

1. Support the exploration domains with nondeterministic and stochastic sensing in Sections 3.3.2 and 3.3.3; and
2. Support the decision-making model of exploration with deterministic sensing in Section 3.5.1.

In any of the following exploration domains, the state of the world is hidden from the agent. More precisely, we will even consider mixed observability of the state [Ong et al., 2010, Araya-López et al., 2010b]. That is, some features that describe the state are observed partially whereas others are observed completely. In our context, the agent locally and partially observes the occupancy map, whereas it completely observes its position. The first exploration domain assumes deterministic control and sensing and can be formalized as a POSP where the agent has to efficiently gather information in order to map the environment.

An instance of this first exploration domain uses the same representation of the environment and of the robot as in Chapter 2. As a reminder: the environment is modeled as a deterministic occupancy grid denoted by $map : (rows, cols, occupancy)$ of $rows \times cols$ cells and where each cell is either *free of* or *occupied by* an obstacle; the agent is denoted by $robot : (pos, range)$ and described by its position pos and its circular viewing $range$. This instance will be denoted by $exp_{det}(map, robot)$ and formalized as a POSP as follows.

$\mathbf{s_0} : (\mathbf{p_0}, \mathbf{m_0})$, the initial state corresponds to the robot's position $p_0 \in [1..map.rows] \times [1..map.cols]$; and the deterministic occupancy map m_0 with cell values in the set $\{f\ (free), \neg f\ (occupied)\}$.

ApplicableAction(s), returns a subset of moving actions, $A = \{noop, \Uparrow, \Rightarrow, \Downarrow, \Leftarrow\}$, where actions lead from the current position s_p to a neighboring position. Here, we allow the *noop* operator so that the agent can stay in place. This will be useful with nondeterministic measurements as the agent may have to observe the same state many times.

TransitionModel(s,a), deterministically outputs the next state s' for a given action a applied in state s by querying s_m. It updates the pose s_p according to the desired action a and the occupancy map s_m (Figure 2.1).

ObservationModel(s), is a deterministic state-observation model which depends only on the attained state s. It always outputs the measurement z_s ($\{z_s = (s_p, s_v)\} = ObservationModel(S_{t+1} = s)$) which contains the position s_p and the local view s_v. This local view represents the known occupancy status of visible cells and the unknown occupancy status of occluded cells in the maximal field of view of the robot. This state-observation model uses ray casting to distinguish visible from occluded cells.

Concretely, the local view $s_v \in V(s) = \{v_i \mid v_i : F(s) \rightarrow \{f\ (free), \neg f\ (occupied), \neg k\ (unknown)\}\}$ maps any cell in the maximal field of view $F(s)$ of radius *range* centered on the robot's position s_p, to the free (f) or occupied $(\neg f)$ status of visible cells and the unknown occupancy status $(\neg k)$ of occluded cells.

GoalTest(·), is problematic. Strictly speaking there is no world's state that represents a goal. Indeed, the world's state represents the robot's position on an occupancy map. The latter component cannot change due to our static world assumption but the former can. Nevertheless, we are not modeling a path planning problem for obstacle avoidance that asks the robot to reach a given location on the grid. Later, we see how this issue can be addressed at the belief level.

StepCost(·), is the immediate cost defined according to an optimization criterion. To minimize the number of decision steps, it equals 1 whatever the input.

The corresponding domain is implemented in Domain 1. We formalized the control and sensor model of the planning agent for exploration with deterministic moves and deterministic ray casting. Now, we consider the nondeterministic case.

The case of nondeterministic sensing

A nondeterministic state-observation model outputs a set $Z_{sas'} = ObservationModel'(S_t = s, A_t = a, S_{t+1} = s')$ of measurements that can be returned given a current world's state s, an agent's action a, and a next world's state s'. Formally, the agent faces a nondeterministic sensing problem when $|ObservationModel'(s, a, s')| > 1$ for at least one input triplet.

Now, we can model an exploration task with nondeterministic control and sensing over world's states.

3.3.2 Exploration task with nondeterministic control and sensing

We describe a first domain of exploration with nondeterministic control and sensing. It will serve to:

1. Support the domain of exploration with stochastic sensing in Section 3.3.3; and
2. Support the decision-making model of exploration with nondeterministic sensing in Section 3.5.1.

To exemplify unquantified uncertainty in the measurements of the world's state, we extend the previous deterministic state-observation model. This extension considers that the robot uses a noisy range finder and explores an unknown environment with a slippery floor. This time, we use the nondeterministic control model introduced for cov_{nondet} in Section 2.3.2 and illustrated in Figure 2.3.

A noisy range finder can produce maximum readings when the line of sight is obstructed by an obstacle. Conversely, it sometimes produces short readings when the line of sight is free of intermediate obstacles. Therefore, we consider that the ground truth local map is subject to a noise that flips the occupancy status of at most k cells from occupied to free and *vice versa*. This instantiation will be denoted by exp_{nondet} and modeled as a POCP as follows.

Domain 1 Domain of the exploration task as a POSP

function *ApplicableAction*(s)
▷ Select the actions available from the given state s
$A \leftarrow \{noop\}$
for $\langle a, \delta_p \rangle \in \{\langle \Uparrow, (-1,0)\rangle, \langle \Downarrow, (+1,0)\rangle, \langle \Leftarrow, (0,-1)\rangle, \langle \Rightarrow, (0,+1)\rangle\}$ **do**
if $isfree(s_m, s_p)$ and $isfree(s_m, s_p + \delta_p)$ **then** $A \leftarrow A \cup \{a\}$
return A

---

function $TransitionModel(s, a)$
▷ Update the ground truth map m, the robot position p of the current state s according to action a
$s' \leftarrow s$; $s'_m \leftarrow updatemap(s', a)$; $s'_p \leftarrow updatepose(s', a)$
return {s'}

---

function $GoalTest(\cdot)$
▷ There is no goal at the world's state level
return $false$

---

function $StepCost(\cdot)$
▷ Compute the cost of staying in any state s
return 1

---

function $ObservationModel(s)$
▷ Deterministic sensing, percepts returned from state s
return $\{(s_p, s_v = F'(s_p, s_m))\}$

---

function $F'(p, m)$
▷ Select visible cells by casting rays in the field of view F (cells within radius $range$)
$F' \leftarrow F(p, m)$; $\forall p' \in F', v(p') \leftarrow \neg k$; $v(p) \leftarrow f$
▷ Pop cells p' in decreasing norm ordering ($||p' - p||$)
while $|F'| \neq 0$ **do**
$p'' \leftarrow pop(F')$; $p' \leftarrow p$; $occlusion \leftarrow false$
▷ Iterate on line of sight from p' to p'' (bresenham)
while $p' \neq p''$ **do**
$p' \leftarrow nextcell(p')$
if ¬occlusion **then** $v(p') \leftarrow m(p')$
$occlusion \leftarrow occlusion \vee m(p') = \neg f$; $F' \leftarrow F' \setminus \{p'\}$
return v

ObservationModel'(s), depends only on the attained state s. It is a simple nondeterministic state-observation model that outputs a set of local views, that are each deterministically generated from a biased local map. In simulation, firstly, the perfect expected measurement $z_{exp} = (p_{exp}, v_{exp}) = ObservationModel(s)$ is obtained by querying the previous deterministic sensor model; secondly, the occupancy status of up to k visible cells are flipped to generate a biased map; thirdly, the perfect measurement on the biased map is recorded.

The corresponding state-observation model is implemented in Domain 2. We generalized the control and sensor model of the planning agent for exploration to nondeterministic moves and nondeterministic ray casting. Now, we consider the stochastic case.

Domain 2 Domain of the exploration task as a POCP

Input: k the maximum number of occupancy status changes allowed among visible cells.

function $ObservationModel'(s)$
▷ Nondeterministic sensing
$z_{exp} = (p_{exp}, v_{exp}) \leftarrow ObservationModel(s)$; $Z_s \leftarrow \emptyset$
for $n = 0$ to $\min[k, v_{exp}^{-1}(\{f, \neg f\}) \setminus \{p_{exp}\}]$ **do**
▷ Generate the combinations of n visible cells in v_{exp}
$C_n \leftarrow combinations(n, v_{exp}^{-1}(\{f, \neg f\}) \setminus \{p_{exp}\})$
for $c_n^i \in C_n$ **do**
▷ Free the occluded cells in the map for maximum readings
$s'_m \leftarrow s_m$;
for $p \in v_{exp}^{-1}(\neg k)$ **do** $s'_m(p) \leftarrow f$
▷ Flip the occupancy status of the visible cells in c_n^i
for $p \in c_n^i$ **do** $s'_m(p) \leftarrow \neg s_m(p)$
▷ Generate a perfect measurement on the biased map
$s' = (s_p, s'_m)$; $Z_s \leftarrow Z_s \cup ObservationModel(s')$
return Z_s

The case of stochastic sensing

A stochastic state-observation model is a quantified version of a nondeterministic state-observation model. It outputs a discrete probability distribution defined over a set of percepts Z. More precisely, $O(s, a, s', z) = Pr(Z_{t+1} = z \mid S_t = s, A_{t+1} = a, S_{t+1} = s')$ represents the probability of observing $z \in Z$ from a state s' after applying action a in state s. Stochastic sensing allows to model and quantify sensor noise.

Now, we can specify an exploration task with stochastic control and sensing over world's states.

3.3.3 Exploration task with stochastic control and sensing

We describe a third instantiation of a stochastic exploration domain that will support the decision-making model of exploration with stochastic sensing in Section 3.5.2.

To illustrate quantified uncertainty in the measurements of the world's state, we extend the previous nondeterministic state-observation model by adding probabilities over the potential

observation outcomes. This extension considers that a robot slips on the floor and obtains measurements with known probabilities during exploration. We reuse the stochastic control model introduced for cov_{sto} in Section 2.3.3 and illustrated in Figure 2.6. We additionally provide a stochastic state-observation model. This instantiation will be denoted by exp_{sto} and modeled as a POMDP as follows.

Z, is the set of measurements an agent can receive from the world. We denote it by $Z = P \times V$ which corresponds to a Cartesian product of the robot position set P and the local view set V. For a given state s, we will consider that the set of potential measurements is restricted to $Z_s = P_s \times V_s = ObservationModel'(s)$. That is, the measurements output by the nondeterministic sensor model.

O(s, z), is the state-observation model which quantifies how likely it is to observe measurement z from a given state s of the world. We assume that this sensor model depends only on the attained state s. It is inspired by the correlation-based sensor model from chapter 6 of [Thrun, 2002] [10]:

$$\begin{aligned} O(s,z) &= \Pr(Z_t = (p_z, v_z) \mid S_t = (p,m)) \\ &\overset{\text{b.r.}}{=} \Pr(v_z \mid p_z, p, m) \Pr(p_z \mid p, m) \\ &\overset{\text{c.i.}}{=} \Pr(v_z \mid p, m) \Pr(p_z \mid p) \end{aligned}$$

where $\Pr(p_z \mid p) = 1_{\{p\}}(p_z)$ is a logical rule checking the state observable features and $\Pr(v_z \mid p, m) = \Pr(v_z \mid v_{exp} = F'(p,m))$ is the probability to obtain the local view v_z given the expected perfect local view v_{exp} from state s. A map matching distance can be used to define the probability of obtaining any local view $v_z \in V_s$.

We assume that it is less likely to obtain a lot of errors in the measured visibility map than only a few ones and define a matching score between local views that counts cells with the same status:

$$score(v, v') = \sum_{p \in dom(v)} \mathbb{1}_{\{v(p)\}}(v'(p)).$$

Finally, we derive a discrete probability distribution from this matching score:

$$\Pr(v_z \mid v_{exp}) = \frac{score(v_z, v_{exp})}{\sum_{v_z \in V(s)} score(v_z, v_{exp})}.$$

R(·), the reward function is the opposite of the step cost introduced in Section 3.3.1, $R(\cdot) = -StepCost(\cdot) = -1$, to obtain a maximization problem.

The corresponding state-observation model is implemented in Domain 3. We generalized the control and sensor models of the planning agent for exploration to stochastic moves and stochastic ray casting.

In this section, we modeled three state-observation models for partially observable states. The state-observation models are necessary to describe a process in which an agent explores and builds the map of an unknown environment. However, as the state of the world is hidden, such

[10] *b.r.* and *c.i.*, respectively stand for *Bayesian Rule*, *Conditional Independence*, and coordinates *Transform*.

Domain 3 Domain of the exploration task as a POMDP

function $O(s)$
▷ Stochastic sensing
$z_{exp} = (p_{exp}, v_{exp}) \leftarrow ObservationModel(s)$;
$Z_s = (P_s \times V_s) \leftarrow ObservationModel'(s)$;
$\forall z \in Z, O_s(z) \leftarrow 0$; $sumScore \leftarrow 0$
for $z = (p_z, v_z) \in Z_s$ **do**
$O_s(z) \leftarrow score(v_z, v_{exp})$; $sumScore \leftarrow sumScore + score(v_z, v_{exp})$
return $O_s / sumScore$

function $R(\cdot)$
▷ Uses the opposite of the step cost in state s
return $-StepCost(\cdot)$

processes are not useful for the agent itself. Hence, in the next section we center the point of view on the agent's belief regarding the world's state. Therefore, we report formalisms that translate hidden world's states to observable agent's beliefs and translate our previous domains to decision-making models of exploration.

3.4 Background: belief-transition models

In order to make decisions in a world with hidden states, the planning agent relies first on a formalism that describes the concrete process over world's states (initial state, actions, control, sensing, and step costs). Thus, any partially observable formalism defined in the previous section is suitable. However, as the state is hidden from the agent, rather than taking decisions based on states, the agent can decide according to:

Designer's procedures, leading to predefined decision-making. This predefined behavior relies on the ability of the designer to cope with every potential situations prior to execution and encode rational decisions in every case.

Observations, leading to *reactive decision-making* [Ferber, 1999]. It is deceptive, as an agent always acts in the same way given a same input. Nevertheless, nonstationarity and randomization allow for a better reactive behavior.

Histories of actions and observations, leading to *history-based decision-making*. It suffers from the curse of history, as a history rule is defined over all ever-growing sequences of actions and observations.

Sufficient statistics, leading to *belief(-state) decision-making*. Sufficient statistics summarize state uncertainty and are viewed as compact representations of the history of observations and actions.

The focus is on *belief(-state) decision-making* as an alternative to the designer's procedures, the reactive, and the history-based behaviors. From now on, this sufficient statistic will be called a belief(-state). Hereafter, we explain why these sufficient statistics are necessary and how they are maintained by the planning agent.

3.4.1 Modeling perceptual aliasing

Perceptual aliasing occurs when distinct world's states generate the same measurement, *i.e.*, making these states perceptually indistinguishable to the agent. This never occurs in completely observable formalisms as the implicit state-observation model is the identity function from states to themselves. It always occurs in unobservable formalisms, as the implicit state-observation model is a surjective function (from all states to the empty observation). In between, it may occur in partially observable formalisms.

Formally, with a deterministic or nondeterministic state-transition model, the set of states that are perceptually aliased by action a and observation z is:

$$S^{az} = \{s' \mid \forall s \in S, s' \in TransitionModel^{(')}(s,a) \wedge z \in ObservationModel^{(')}(s,a,s')\},$$

where S is the state space implicitly induced by s_0, *ApplicableAction*, and $TransitionModel^{(')}$. Similarly, with a stochastic state-transition model, the set of perceptually aliased states is:

$$S^{az} = \{s' \mid \forall s \in S,\ T(s,a,s') > 0 \wedge O(s,a,s',z) > 0\}.$$

Hence, the planning agent must maintain these perceptually aliased states (called a belief) or a distribution over them (called a belief state).

3.4.2 Translating to observable beliefs

A state estimator allows to maintain an observable belief(-state) regarding the world's hidden state. Such belief(-state) compiles all that ever occurred, starting from some prior information and followed by any sequence of actions and observations. Consequently, there is a single belief for each possible history of actions and observations.

We can now report decision-making formalisms over beliefs, which involve a state estimator, and then use them to instantiate decision-making models for exploration. The POSP, POCP, and POMDP formalisms were previously reported for partially observable states and used to model domains of exploration. Hereafter, we report their translations to observable belief(-state)s by incorporating an explicit belief-transition model.

First, the COCP formalism (Definition 2.2) can be generalized to observable beliefs as the Completely Observable (Belief) Contingency Problem (COCP$_{b^{az}}$); it is reported in Definition 3.4. The translation from the state space to a belief space formalism appears in [Bonet and Geffner, 2000]. As the agent cannot directly access the state s, it has to reason over a belief b representing the perceptually aliased states over time. The detailed translation from the POSP or POCP that are defined over states to the COCP$_{b^{az}}$ over beliefs is reported in Section A.2.

Definition 3.4 (Completely Observable (Belief) Contingency Problem (COCP$_{b^{az}}$)). *A Completely Observable (Belief) Contingency Problem is defined as a tuple* $\langle\ b_0$, *ApplicableAction*$_b$, *TransitionModel'*$_b$, *GoalTest*$_b$, *StepCost'*$_b\ \rangle$. *A belief is a set of possible states:* b_0 *is the initial belief;* *ApplicableAction*$_b(b)$ *returns the set of actions available from a given belief;* $TransitionModel'_b(b,a)$ *outputs a set of beliefs that can be attained from* b *after applying action* a *and observing any potential measurement;* $GoalTest_b(b)$ *checks that a given belief is a goal belief; and* $StepCost'_b(b,a,z)$ *returns the cost of transitioning from one belief to another by applying action* a *and receiving observation* z.

Second, the COMDP formalism (Definition 2.3) can be generalized over observable belief states as the Completely Observable (Belief state) Markov Decision Process (COMDP$_{b^{az}}$); it is reported in Definition 3.5. The translation from the state space formalism to a belief-state space formalism where belief states are represented as discrete probability distributions appears in [Cassandra et al., 1994, Kaelbling et al., 1998, Bonet and Geffner, 2000]. The detailed translation from the POMDP that is defined over states to the COMDP$_{b^{az}}$ defined over belief states is reported in Section A.3.

Definition 3.5 (Completely Observable (Belief state) Markov Decision Process (COMDP$_{b^{az}}$)). *A Completely Observable (Belief state) Markov Decision Process is defined as a tuple $\langle$ B, A_b, T'$_b$, R'$_b$, b_0 $\rangle$. b_0, is an initial belief state, a discrete probability distribution over a state space S. It is an element of the belief-state space B. A_b returns the set of actions available from a belief state b. T'$_b(b, a, b')$ outputs the probability of attaining the belief state b' from b after applying action a over all potential measurements. R'$_b(b, a, z)$ returns the belief-state reward received after transitioning from one belief state to another by applying action a and receiving observation z.*

As the percepts are generated by the world, the agent only resolves the true updated belief(-state) during execution. Thus, *contingencies* appear in this belief(-state) space when several observations are available from a predicted belief(-state).

In Sections A.4 and A.5, we identify the cases when perceptual aliasing occurs justifying the necessity of a prediction step in the belief-transition model. Similarly, we identify the cases when contingency occurs justifying the necessity of an update step in the belief-transition model. Two formalisms over belief(-state)s were reported; they involve nondeterministic belief-transition models for such cases of perceptual aliasing and contingencies. Hereafter, we will use these formalisms to instantiate decision-making models for exploration.

3.5 Decision-making models for exploration

The case of deterministic or nondeterministic sensing

As the agent cannot directly access the world's state s, it has to reason over a set of states called a belief b, representing the possible world's states. Furthermore, such beliefs can be maintained during execution based on the actual sequence of output actions and input measurements by relying on a belief-transition model. Even further, this belief-transition model can be used to simulate future histories of actions and measurements. Hereafter, we report the belief-transition model of the COCP$_{b^{az}}$ simplified for our exploration domain as our deterministic and nondeterministic sensor models depend only on the attained state.

Formally, for a given prior belief b, the posterior beliefs b^{az} after applying action a are computed as follows:

TransitionModel'$_{\mathbf{b}}(\mathbf{b}, \mathbf{a})$, the transition model over updated beliefs is nondeterministic:

$$TransitionModel'_b(b, a) = \{b^{az} \mid \forall z \in Z^a\},$$

where Z^a denotes the set of potential observations available from the predicted belief b^a:

$$Z^a = \bigcup_{s' \in b^a} ObservationModel^{(')}(s').$$

Firstly, state estimation involves a prediction step that applies action a to all states s in b in order to obtain a predicted belief b^a:

$$b^a = \bigcup_{s \in b} TransitionModel^{(\prime)}(s, a).$$

Secondly, it involves an update step that accounts for an observation z to obtain an updated belief:

$$b^{az} = \{s' \in b^a \mid z \in ObservationModel^{(\prime)}(s')\}.$$

Now, we can model an exploration task with deterministic or nondeterministic control and sensing over agent's beliefs.

3.5.1 Exploration model over agent's beliefs

We translate the exploration task from a POSP or a POCP to a $\text{COCP}_{b^{az}}$. The translation of our POSP or POCP-based $exp_{(non)det}$ to a $\text{COCP}_{b^{az}}$ is straightforward:

$\mathbf{b_0}$, is the set of potential initial states. Let us consider that the robot starts in the upper left cell of the map, with the current cell occupancy status set to $free$. Thus, the set of potential initial states, $b_0 = P_0 \times M_0$ with:

$$P_0 = \{p_0 = (1,1)\},\ M_0 = \left\{ m_i \;\middle|\; m_i(p) \in \begin{cases} \{f\} & \text{if } p \in \{p_0\} \\ \{f, \neg f\} & \text{else } (p \in P \setminus \{p_0\}) \end{cases} \right\}.$$

$\mathbf{ApplicableAction_b}$, is specified with the union set operator $\cup$ to consider all available state actions:

$$ApplicableAction_b(b) = \cup_{s \in b} ApplicableAction(s).$$

That is, we assume that the robot can safely bounce against walls without incurring damages.

$\mathbf{GoalTest_b}$, is difficult to specify. Indeed, as mentioned earlier there is no goal at the state level. Hence, we cannot define a goal belief as in the $\text{COCP}_{b^{az}}$ formalism by using a conjunction of world's state goal tests. In our exploration domain, the agent is able to localize itself, hence in case all visible cells can be visited, the goal belief is easily defined as the singleton belief:

$$|b| = 1.$$

However, if some visible cells cannot be visited, a singleton belief may not be attainable (except with deterministic sensing). In the general case, when it is not possible to define a goal belief, we rely on step costs over states and beliefs.

$\mathbf{TransitionModel'_b}$, is specified as above from the state-transition model $TransitionModel^{(\prime)}$ and the sensing model $ObservationModel^{(\prime)}$.

$\mathbf{StepCost'_b(b, a, z)}$, is specified as a combination of the time-step cost and a belief-based cost representing state uncertainty:

$$StepCost'_b(b, a, z) = StepCost(\cdot) - |b \setminus b^{az}| = 1 - |b \setminus b^{az}|.$$

This is done to minimize the expenditure of time and uncertainty in the agent's belief.

The corresponding COCP$_{b^{az}}$ is implemented in Model 4, and the update process is simplified as the state-observation model is only dependent on the attained state in our exploration task. The belief space for a deterministic control and sensing exploration task $exp_{det}(2, 2, 0)$ is illustrated as an And-Or tree or graph in Figure 3.2. We instantiated the decision-making model of a planning agent for exploration over beliefs. Now, we consider belief states.

Model 4 Model of the exploration task as a COCP$_{b^{az}}$

function $ApplicableAction_b(b)$
  $A_b \leftarrow \bigcup_{s \in b} ApplicableAction(s)$
  return A_b

---

function $TransitionModel'_b(b, a)$
  ▷ Predict a next belief according to the action a
  $b^a \leftarrow \{\}$
  for $s \in b$ **do** $b^a \leftarrow b^a \cup TransitionModel^{(\prime)}(s, a)$

  ▷ Predict a set of observations perceived from b^a
  $Z^a \leftarrow \{\}$
  for $s \in b^a$ **do** $Z^a \leftarrow Z^a \cup ObservationModel^{(\prime)}(s)$

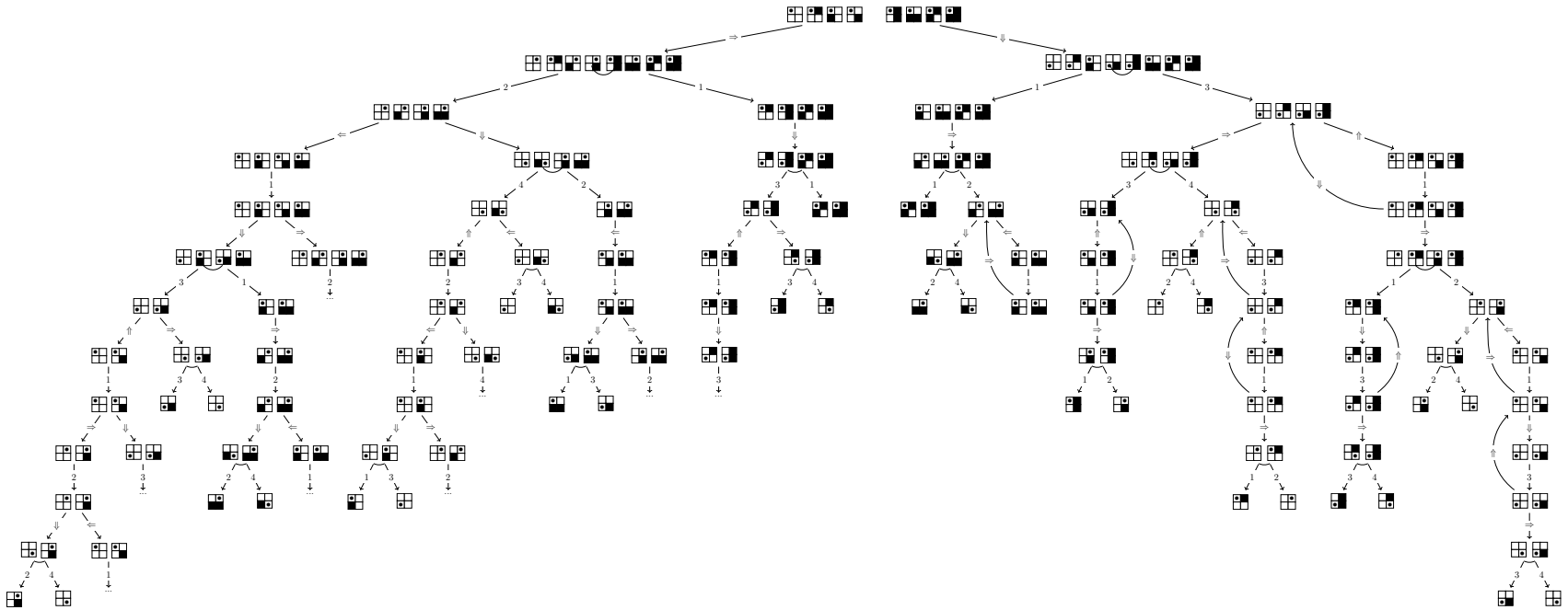
  ▷ Compute updated beliefs b^{az} for each predicted observation in Z^a
  $b' \leftarrow \{\}$
  for $z \in Z^a$ **do**
    $b_{az} \leftarrow \{\}$
    for $s' \in b^a$ **do**
      if $z \in ObservationModel^{(\prime)}$(s') **then** $b^{az} \leftarrow b^{az} \cup s'$
    $b' \leftarrow b' \cup \{b^{az}\}$
  return $\{b'\}$

---

function $GoalTest_b(b)$
  ▷ Check if b is a goal belief
  return $|b| = 1$

---

function $StepCost_b(b, a, z)$
  ▷ The maximum state costs and the reduction in the belief's cardinality
  return $StepCost(\cdot) - |b \setminus b^{az}|$



(a) And-Or Tree **(b)** And-Or Graph

Figure 3.2: This figure illustrates the belief space of an instance of the exploration task with partial state-observability, deterministic control and sensing modeled by a POCP and denoted by $exp_{det}(map : (2, 2, \{\{(1, 1), (1, 2), (2, 1)\} \rightarrow f, \{(2, 2)\} \rightarrow \neg f\}), robot : ((1, 1), 0))$. This instance takes place on a 2 by 2 occupancy grid whose bottom right cell is occupied by an obstacle (black), and top left cell is occupied by a robot (the $\bullet$). The concrete state is hidden to the robot, as it only observes its *position* $\in (1, 2, 3, 4)$ at any time step. Therefore it maintains a belief over the hidden state. This belief space is represented as an And-Or belief tree (left side) and belief graph (right side), where each arc either outputs a predicted belief from a prior belief by applying an action, or a posterior belief from a predicted belief by updating with a measurement. Every node encapsulates the potential states of the world, in which the agent may be after a sequence of actions and measurements from the initial prior. The initial prior is the root node, the initial belief concerning the world's state (the true state is the fourth one but the agent does not know it). Goal beliefs are the beliefs that cannot be further disambiguated.

The case of stochastic sensing

Now, belief states are considered, and a stochastic planning model with a belief state estimator is detailed for the partially observable domain with stochastic sensing. Belief states are discrete probability distributions over the world's hidden states. The following state estimator is the same as the belief-state transition model presented in [Bonet and Geffner, 2000], or in [Cassandra et al., 1994]. Hereafter, we report the belief-state transition model of the COMDP$_{b^{az}}$ simplified for our exploration domain as our stochastic sensor model depends only on the attained state.

Formally, for a given belief state b, the probability of attaining the next belief state b' after applying action a is computed as follows:

$\mathbf{T'_b}(\mathbf{b}, \mathbf{a}, \mathbf{b'})$, the transition model over updated belief states is stochastic as it defines a probability distribution over updated beliefs b' at $t+1$:

$$T_b(b, a, b') = Pr(B_{t+1} = b' \mid B_t = b, A_t = a) = \sum_{z \in Z} \mathbb{1}_{\{b^{az}\}}(b') \Pr(z \mid b, a).$$

Firstly, state estimation involves a prediction step that applies action a to all states s in b to obtain a predicted belief state b^a:

$$b^a(s') = Pr(S_{t'} = s' \mid B_t = b, A_t = a) = \sum_{s \in S} b(s) T(s, a, s').$$

Secondly, it involves an update step that corrects the belief state prediction according to observation z to obtain a updated belief state b^{az}:

$$b^{az}(s') = Pr(S_{t+1} = s' \mid B_t = b, A_t = a, Z_{t+1} = z) = \frac{O(s', z) b^a(s')}{\Pr(z \mid b, a)}.$$

Finally, the probability of observing z, after applying action a from a belief state b is:

$$Pr(z \mid b, a) = Pr(Z_{t+1} = z \mid B_t = b, A_t = a) = \sum_{s' \in S} O(s', z) b^a(s').$$

Now, we can model an exploration task with stochastic control and sensing over agent's belief states.

3.5.2 Exploration model over agent's belief states

We translate the exploration task from a POMDP to a COMDP$_{b^{az}}$. The translation of our POMDP exp_{sto}, over belief states to a COMDP$_{b^{az}}$ is as follows:

$\mathbf{b_0}$, is the initial belief state, that is a distribution over states that summarizes initial state uncertainty. Let us consider that the robot initially knows where it is located (upper left corner), and its current cell occupancy status (free of obstacle). As everything else is unknown, this prior considers that all such states (let denote this set of states by S_0) are equally likely to be the initial state. Hence, a random variable S_0 is distributed according to b_0, such that:

$$P(S_0 = s) = b_0(s) = \mathbf{U}_{S_0}(\{s\}) = \frac{\mathbb{1}_{S_0}(s)}{|S_0|}.$$

B, the belief-state space is the continuous space representing all discrete probability distributions over S (|S-1|-simplex).

$\mathbf{A_b}$, the belief state action set is specified with the $\cup$ set aggregation operator:

$$A_b(b) = \bigcup_{s \in S, b(s)>0} A(s).$$

$\mathbf{T'_b}$, the belief-state transition model is specified as above and relies on the state-transition and state-observation models that appear in the POMDP domain of exploration.

$\mathbf{R'_b}$, the belief-state reward combines the underlying time-step reward and the reduction in state uncertainty between consecutive beliefs:

$$R'_b(b,a,z) = R(\cdot) + D_{KL}(b^{az}||b) = -1 + \sum_{s \in S} b^{az}(s) log(\frac{b^{az}(s)}{b(s)}).$$

$D_{KL}(b^{az}||b)$ is the *Kullback-Leibler* divergence from the posterior belief state b^{az} to the prior belief state b. [Kullback and Leibler, 1951, Kullback, 1997]. It is nonnegative and quantifies the information gained by applying action a from the belief state b, and receiving the measurement z afterwards. Such belief-dependent rewards are useful to reward actions that reduce state uncertainty. The Partially Observable Markov Decision Process with belief-dependent rewards (ρPOMDP) framework introduced in [Araya-López et al., 2010a] allows applying classical POMDP solving algorithms with such belief-dependent rewards. Other alternatives for measuring the posterior entropy, or the change in entropy are mentioned in the literature of active sensing [Bajcsy, 1988, Mihaylova et al., 2003].

We instantiated the decision-making model of a planning agent for exploration over belief states. This is the most general decision-making model of Single-Robot Active Mapping that is reported in this manuscript. In general, it is extremely costly to compute finite-horizon policies to the Partially Observable Markov Decision Process with the standard value iteration algorithms, as they suffer from both the curse of dimensionality and the curse of history [Kaelbling et al., 1998, Pineau et al., 2006, Ross et al., 2008].

The curse of dimensionality is caused by the number of potential hidden states. In our case, if the robot hesitates between n maps, then it has to reason over belief states that are probability distributions over these n maps. This raises important issues regarding the representation, the prediction, and the update of belief states. Indeed, maintaining probability distributions over these occupancy grids is costly as the number of potential maps exponentially grows with the size of the environment and a single step of the *Bayesian filter* will quickly become expensive. Hence, it is important to consider compact representations based on maps features and more efficient filters.

Similarly, the curse of history is another issue, planning many steps ahead quickly becomes unmanageable. Indeed, there is a single belief for each history, but the number of histories grows exponentially in the number of lookahead decision steps. Nevertheless, recently, novel techniques are able to address both curses by relying on *Particle Filters* and *Monte-Carlo* sampling [Silver and Veness, 2010]. They can represent a good starting point for a study dedicated to solving this last decision-making model.

3.6 Conclusion

To end this chapter we provide a table summarizing formalisms of SDM that are suitable for the coverage or the exploration task according to state-observability, deterministic or nondeterministic acting and sensing, and initial state knowledge in Section A.6.

We followed a systematic process for modeling coverage and exploration tasks as SDM problems, by answering questions concerning initial knowledge, state-observability, types of control and sensing. This systematic process led us from the instantiation of a coverage task with observable knowledge states, to the modeling of an exploration task with observable belief states.

We instantiated these models to build simulators dedicated to single-agent coverage and exploration tasks, and then design middle to long-term *model-based* (planning) or *model-free* (learning) agents. Although these instances always illustrate a two-dimensional grid, an extension to an n-dimensional grid is straightforward. Moreover, modeling a single-agent task should be understood as modeling a centralized decision-making agent task. Hence, modeling decentralized agents that interact in a world where a team of robots tries to cover or explore the environment is not addressed yet.

In the following chapter, we continue on our journey toward encouraging long-term planning for coverage and exploration tasks, as an alternative to the already widely spread short-term planning techniques. We focus on selecting an appropriate long-term model-based planner and conducting experiments with instances of an SDM domain. In this sense, we narrow our study to the first model of coverage over observable knowledge states with deterministic control and sensing.

Part II

Long-Term Planning for Deterministic Active Coverage

Chapter 4

Off-line Planning

Contents

4.1 Introduction

In the previous part, we specified coverage and exploration tasks in standard formalisms of Sequential Decision-Making (SDM). We provided domains that simulate worlds with strict assumptions regarding control and sensing and in which an agent interacts with its environment to achieve Active Coverage (AC) or Active Mapping (AM). These domains are independent of the considered solving paradigm. We additionally provided decision-making models of coverage and exploration that can support a planning agent. Therefore, we focus on the planning paradigm to compute solutions to the Single-Robot AC model.

This will serve to demonstrate the utility of such decision-making models and assess the empirical performances of planning algorithms on the AC problem. Indeed, in Sequential Decision-Making, novel planners are usually evaluated on combinatorial games, *e.g.*, the n-puzzle, n-queens, tower of hanoi, motion planning, etc. Moreover, in experimental Robotics, an exploration strategy is usually compared to: other greedy strategies as in a tournament [Bautin et al., 2012, Faigl et al., 2012], a decision-theoretic baseline [Holz et al., 2010], a

sampled upper bound on the optimal coverage path [Faigl and Kulich, 2015], or also the optimal coverage path [Li et al., 2012]. Currently, there is no consensus regarding the best practice, but as we want to design agents that seek optimality, providing optimal baselines is a good way to estimate the gap between greedy agents and planning agents. Hence, we want to extend the proof of concept that appears in [Li et al., 2012] to a more comprehensive benchmark.

Firstly, in Section 4.2, we provide some background regarding the link between the Completely Observable (State) Search Problem (COSP) formalism and the deterministic shortest path problem; and discuss the off-line search paradigm that can address such shortest path problems. Secondly, in Section 4.3, we report the state of the art in off-line search; these techniques can solve the deterministic coverage model introduced in Section 2.3.1. Here, off-line means that the agent plans a full solution before executing it. This deliberation is usually qualified as unbounded as it assumes infinite resources. Finally, in Section 4.4, we conduct experiments with representative off-line search techniques to assess their average performances when solving instances of the coverage task. Therefore, we will bound deliberation by fixing time and space resources and consider three kinds of guarantees: optimal, suboptimal, and anytime output solutions.

4.2 Background

4.2.1 Planning as search in Or graphs.

The COSP formalism generates a state space Or graph (Definition 2.1) as illustrated for the deterministic coverage model in Figure 2.2. Indeed, successive applications of the state-transition operator from a given initial state generates vertices that hold states, directed arcs that deterministically transition from a predecessor to a successor state via an action, and weights that account for the step cost of any transition.

Definition 4.1 (Directed Or Graph). *An Or graph $G(V, E, c)$ is a weighted directed graph composed of a set of Or vertices, V, a set of arcs, $E \subseteq V \times V$ and an arc weighting function, $c : E \to \mathbb{R}$. An Or node encapsulates a state of the world from which an agent has to choose one action, and some extra information. An arc is a directed link, (u,v), from a node, u, to a node, v, which represents a deterministic transition from the state encapsulated in u, to the state encapsulated in v. Additionally, u is termed the predecessor of v, while v is termed the successor of u. An arc-weight represents the cost of transitioning via the corresponding link.*

When there is a single initial state and a single goal state [11], the COSP model can be cast as a *single-source single-sink* (abbreviated *single-pair*) shortest path problem. We focus on this variant; the others are out of the scope of our study (single source, single sink, or any pair).

Solution form, evaluation, and optimality

Any finite path $p = (e_1), ..., (e_n) \in \mathbf{P}$ that starts from a source s, goes through the arcs e_i, and ends in a sink t, is a satisfiable solution to the COSP model. The cost of a path is the sum of its arc weights and is denoted by d. For any satisfiable path p we have:

$$d(p = (e_1), ..., (e_n)) = \sum_{i \in [1,n]} c(e_i).$$

[11] More generally, it can be extended to a cluster of states that verify a goal property. For instance in coverage, the goal cluster is the set of goal states with a full coverage map and the robot's position does not matter.

The optimal path cost is the minimum path cost among all satisfiable paths and is denoted by d^*:

$$d^*(s,t) = \min_{p=(s=n_0,n_1),\ldots,(n_{k-1},n_k=t)\in\mathbf{P}} d(p).$$

The optimal paths are satisfiable paths with minimum distance d^* and are denoted by p^*:

$$p^*(s,t) \in \arg\min_{p=(s=n_0,n_1),\ldots,(n_{k-1},n_k=t)\in\mathbf{P}} d(p).$$

In the following, we rely on off-line search algorithms that address deterministic shortest path problems generated by a COSP model. A search algorithm is a means to lazily generate an Or graph, node by node from the source in order to identify an optimal path to the sink, if it exists.

4.2.2 Off-line or complete search.

An off-line search algorithm takes as input a state space model that can generate an Or graph; then, it searches a complete solution to the shortest path problem encapsulated in the graph. A complete solution is a path that links the source to the sink. Off-line search algorithms are suitable for one-shot shortest path problems in static domains, *i.e.*, state space graphs whose topology and arc-costs do not change over time. Once a complete solution is found, it is provided to an agent that translates arcs to actions and executes the planned solution as in the open-loop control paradigm. To summarize, there are two steps: first, planning computes a solution; second, execution applies the solution. As we simplify the execution step to a sequence of action assignment statements, we only focus on the planning step in the following.

Uninformed search.

In uninformed or blind search, the state space is browsed in a predetermined manner from the source until the sink is found. It is also termed blind search because the state space browsing method uses no hint regarding the direction of the sink in the state space. We focus on forward search algorithms that compute estimates of the optimal cost so far, g^*, and extract lowest-cost trajectories, p^*. The unknowns g^* are the optimal path costs from the source s to any node t ($g^*(t) = d^*(s,t)$). The unknowns p^* are the optimal paths from the source s to any node t ($p^*(t) = p^*(s,t)$).

Forward search algorithms generate and expand nodes from the source while maintaining source distance estimates g and parent pointers p to trace back the trajectory from the source. They take inspiration from the collection of mutually recursive forward Bellman equations [Bellman, 1957] in order to compute the unknowns g^*, as shown in Equation (4.1):

$$g^*(n) = \begin{cases} 0 & \text{if } \mathit{source}(n) \\ \min_{n'\in Pred(n)} g^*(n') + c(n',n) & \text{otherwise.} \end{cases} \tag{4.1}$$

Complete search algorithms are either uninformed or informed. That is, the former type of algorithms do not rely on expert knowledge for guiding the generations and expansions of nodes, whereas the latter does.

Informed search.

Informed or heuristic search algorithms browse the state space from the source toward the sink and may generate fewer nodes than uninformed search algorithms. The search is guided thanks to a cost-to-go heuristic, *i.e.*, an estimate h of the optimal distance h^* from any node to the sink. It allows to focus search in the direction of the sink and prune whole parts of the search space. Formally, the unknowns h^* represent the optimal path cost from any node s to the sink t ($h^*(s) = d^*(s,t)$). Similarly to optimal source distances, optimal sink distances h^* satisfy the following set of backward Bellman equations:

$$h^*(n) = \begin{cases} 0 & \text{if } sink(n) \\ \min_{n' \in Succ(n)} c(n,n') + h^*(n') & \text{otherwise.} \end{cases}$$

We introduce two definitions that play an important role in guaranteeing optimality of the solution path output by informed algorithms: admissibility and consistency of the heuristic in Definitions 4.2 and 4.3.

Definition 4.2 (Heuristic admissibility). *A heuristic function, h, is admissible iff, $h(s) \leq h^*(s), \forall s \in V$. It never over estimates the optimal cost of a path from a state, s, to a goal.*

Definition 4.3 (Heuristic consistency). *A heuristic function, h, is consistent iff, $0 \leq h(s) \leq c(s,s') + h(s'), \forall s \in V, s' \in Succ(s)$. It respects the triangle equation so that the estimated cost of a path from a state, s, to a goal does not exceed the cost of reaching a successor state, s', plus the estimated cost of a path from this successor, s', to a goal.*

Next, we present the frameworks of heuristic search and some techniques that we will consider in our experiments.

4.3 Off-line search frameworks

We report two popular frameworks of heuristic search. The first one is Best-First Search that expands nodes in the generated state space according to their promise; the second one is Depth-First Search that expands nodes among the successors of the current node according to a local node-ordering strategy.

4.3.1 Best-First Search framework

Best-First Search (BFS) is a framework of algorithms that expand nodes from the generated but not yet expanded part of the state space based on their priority. We discuss iterative, frontier, and recursive subclasses of the BFS framework, and report their details and pseudo-code in Section B.1.

Iterative Best-First Search (I-BFS) [Dechter and Pearl, 1985], is a BFS algorithm that generates a state space graph by maintaining a first structure (*Open*) of frontier nodes that are generated but not yet expanded and a second structure (*Closed*) of interior nodes that are already expanded. Each node, $n = (s(n), g(n), p(n))$, contains a state $s(n)$, a source distance estimate $g(n)$, and a parent pointer $p(n)$. Additionally, a priority key $f(n)$ can be computed from the aforementioned elements.
This algorithm is complete, *i.e.*, it finds a solution if one exists. Moreover, its solutions are g-optimal, *i.e.*, they have minimum path costs. However, as it maintains the partially

expanded search graph in memory (Open and Closed), its time and space complexity is exponential in the depth of the optimal path in the worst case.

Frontier Best-First Search (F-BFS) [Korf et al., 2005], is an I-BFS algorithm that only stores frontier nodes in Open. Each node, $n = (s(n), g(n), o(n))$, maintains an s-state and a g-value as in I-BFS, but replaces the p-pointer by an o-operator, $o : Neigh(n) \to \{1, 0\}$. This operator indicates if a neighbor, $n' \in Neigh(n)$, has already been generated (1) or not (0). Additionally, a priority key $f(n)$ can be computed from the aforementioned elements. This algorithm reduces memory requirements in practice and its space complexity is proportional to the width of the state space.

Recursive Best-First Search (R-BFS) [Korf, 1993], is a BFS algorithm that stores neither frontier nodes (Open) nor interior nodes (Closed). Each node, $n = (s(n), g(n), F(n))$, maintains an s-state, a static g-value as in I-BFS, but it additionally has a stored F-key which corresponds to the f-key of the best frontier node, n', ever generated below n (initially $F(n) = f(n)$). This stored F-key is used for prioritizing recursive calls on most promising subgraphs whose frontiers' priority does not exceed local upper bounds.
This algorithm has a space-complexity that is linear in the search depth as it only maintains the current searched path in memory (traversed nodes and their siblings).

In uninformed search, successors g-values and f-keys are computed as $f(n') = g(n') = g(n) + c(e),\ \forall n' \in Succ(n)$. When the arc weights in G are unitary, I-BFS becomes Iterative Breadth-First search (I-BreadthFS); or otherwise Iterative Uniform-Cost Search (I-UCS) also known as Iterative Dijkstra's Algorithm (I-Dijkstra) ([Felner, 2011, Dijkstra, 1959]). Similarly, Frontier Breadth-First search (F-BreadthFS), Frontier Uniform-Cost Search (F-UCS), Recursive Breadth-First search (R-BreadthFS), and Recursive Uniform-Cost Search (R-UCS) are defined depending on the type of edge costs.

In informed search, heuristic knowledge is seamlessly incorporated when computing the priority key of each generated node. When BFS algorithms are informed, each node, n, additionally maintains an h-value representing a sink distance estimate and uses a priority f-key which combines g-values and h-values. This priority key, $f(n) = \text{agg}(g(n), h(n))$, is the estimated value of a path from the source to the sink that is constrained to go through n.

A Star (A*) [Hart et al., 1968, Hart et al., 1972], is the most popular best-first heuristic search algorithm and is the basis of several other algorithms [Rios and Chaimowicz, 2010]. Its priority f-key, $f(n) = g(n) + h(n)$, is the sum of the computed cost-so-far g and the static cost-to-go h. It is able to focus search in the direction of the sink and prune the state space depending on the accuracy of its static heuristic. It is complete as it always returns an optimal solution (g-optimal) when available, with an admissible and non-negative heuristic ($0 \leq h(n) \leq h^*(n)$). However, in practice, it usually completely fills the memory before returning an optimal solution.

Furthermore, parametric priority keys allow to obtain a range of heuristically guided best-first expansion strategies:

Weighted A* (wA*) [Pohl, 1970], uses $f(n) = g(n) + w \times h(n)$ where $w \geq 0$ weights the importance of the heuristic information in the cost function. Increasing w generates more aggressive browsing toward a goal, and returns bounded suboptimal solutions faster.

Consequently, I-BFS, F-BFS, and R-BFS easily incorporate heuristic knowledge. For instance, Iterative A* (I-BFS(A*)) is expansion optimal, *i.e.*, it expands the fewest number of nodes when using a consistent heuristic, but it is suboptimal regarding Open-list storage. Therefore, versions with controlled storage as Iterative Partial Expansion A* (I-BFS(PEA*)) [Yoshizumi et al., 2000] are proposed.
R-BFS requires a few adjustments to additionally backpropagate the h-value of the best frontier node ever generated in the subtree rooted under any stack node. Unfortunately, it suffers from a node reexpansion overhead in domains with a wide range of costs and where the best frontiers are sparsely distributed in the state space. Therefore, versions with controlled reexpansion bounds are proposed, see R-BFS$_{CR}$ in [Hatem et al., 2015].

This ends our brief review of the BFS framework in off-line deterministic planning. We now turn our attention to a second framework.

4.3.2 Depth-First Search framework

Depth-First Search (DFS) is a framework of algorithms that browse a state space search tree by expanding the next node among the current node's successors, and then backtracking when all successors have been expanded [12]. The original DFS ends when a sink is expanded (solution found) or when it backtracks to the root (no solution); but it can also cycle forever.
When it terminates, its time complexity is exponential in the solution depth in the worst case and it may expand all nodes several times; and its space complexity is linear in the solution depth as it maintains only the searched path in memory. Nevertheless, DFS returns suboptimal solutions and is incomplete as it can be trapped in loops. Below, we describe two of its complete and optimal extensions; the description of DFS and the pseudo-code of the following algorithms are reported in Section B.2.

Depth-Limited Search (DLS), is a DFS algorithm that uses a depth cut-off T to truncate the search tree at a fixed depth. Any node, $n = (s(n), g(n))$, contains a state $s(n)$ and a source depth estimate $g(n)$. More generally, let us reintroduce a priority key $f(n) = g(n)$. In this algorithm, any node n with a priority higher than T is pruned during search ($f(n) > T$). This algorithm is complete if T is greater than the depth of an optimal path to the target t ($T \geq f^*(t)$); however, in general it does not return an optimal path. Similarly, Cost-Limited Search (CLS) can be defined with a cost cut-off. On the other hand, the two following algorithms can return optimal paths.

Iterative-Deepening Depth-First Search (ID-DFS) [Korf, 1985], consists in repeated *Depth-Limited Searches* with an increasing lower bound T on the optimal solution depth to the target t ($T \leq f^*(t)$). In each repetition, the current search tree is entirely browsed up to T unless a solution is found earlier. If the current repetition ends without success, T is increased to the lowest depth among pruned nodes ($T = min_{pruned} f(n)$), and then another repetition starts. This algorithm is complete and optimal; it emulates Breadth-First search (BreadthFS) in linear space with a monotonic cost function.

Branch-and-Bound Depth-First Search (BnB-DFS) [Zhang and Korf, 1995], is a DFS algorithm that maintains an upper bound T on the optimal solution depth to the target t ($T \geq f^*(t)$). Any node n with a priority higher than T is pruned during search ($f(n) > T$).

[12] Contrary to BFS techniques where iterative and recursive versions produce two distinct families, choosing between Iterative DFS (I-DFS) or Recursive DFS (R-DFS) is an implementation choice.

Moreover, the upper bound T is lowered every time a shallower solution to the target t is found ($T = \min(T, f(t))$). When an upper bound on the optimal depth is known, BnB-DFS algorithms are complete, optimal, and require linear space in the depth of the solution.

Informed DFS algorithms can incorporate heuristic knowledge in at least two ways: the first is *node-ordering* to prioritize some children over others; and the second is *node-pruning* when costs are monotonically nondecreasing along any branch of the search tree. Regarding node-pruning, the priority computed by A* ($f = g + h$) or wA* ($f = g + wh$) can be used to prune nodes in both ID-DFS and BnB-DFS.

For instance, in Iterative-Deepening A Star (ID-DFS(A*)) [Korf, 1985] when the current repetition ends without success, the current lower bound on the optimal path cost is updated to the lowest priority among pruned nodes. Unfortunately, this algorithm can suffer from a node reexpansion overhead, *i.e.*, the next iteration may expand only a few new nodes. Hence, novel versions with controlled bounds such as IDA^*_{CR} [Sarkar et al., 1991] are proposed in the literature. Similarly, Depth-First Branch-and-Bound A Star (BnB-DFS(A*)) [Zhang and Korf, 1995] uses admissible heuristic knowledge in a straightforward manner which allows to prune nodes earlier.

This ends our brief review of the DFS framework in off-line deterministic planning. Hereafter, we summarize this review and motivate the experiments that follow.

4.3.3 Summary of off-line search

To summarize this state of the art in off-line deterministic search, we provide a quick reference in Section B.3. These techniques are designed for a deterministic domain but they can be applied to nondeterministic domains by the means of replanning. Indeed, off-line search techniques can address exploration of unknown environments by implementing a greedy mapping strategy for on-line information gathering [Koenig et al., 2001]. This is the case of the frontier paradigm [Yamauchi, 1997, Bautin et al., 2012] in AM, where off-line planning and on-line execution are intertwined. To do so, at every decision step, trajectories are planned off-line in the known part of the environment to reach candidate locations, then the robots execute such trajectories on-line to view the unknown part from such locations. Nowadays, this is the common usage of search algorithms in the literature of AM: solving repeated navigation problems to candidate locations.

Similarly, the coverage task can be addressed by solving repeated navigation problems; that is iteratively get closer to candidate locations that are not covered yet. However, in our coverage domain, we are able to anticipate the candidate locations and consequently plan an optimal coverage path. Therefore, we now experiment with a selection of off-line techniques on different instances of the coverage task. We want to evaluate how general search algorithms with specific heuristic knowledge are performing in terms of resource consumption when computing optimal, supoptimal, or anytime plans.

4.4 Experiments

We conduct experiments with a selection of off-line search algorithms on instances of the single-agent deterministic control coverage problem. The purpose of this experimental study is to:

1. Identify the difficulty of instances above which any straightforward implementation surrenders according to predefined memory and time limits.
2. Identify the trade-off between execution and search costs when dealing with suboptimal off-line search algorithms.
3. Emphasize the anytime behavior of BnB-DFS algorithms.

Instances.

Instances of a single-agent deterministic control coverage task are denoted by $cov_{det}(map = (rows, cols, occupancy), robot = (pos, range))$. Each instance describes a map represented by a two-dimensional occupancy grid of $rows \times cols$ cells whose *occupancy* status is deterministic among, $\{f = free, \neg f = occupied\}$, and a robot represented by a two-dimensional position vector, *pos*, and a circular field of view of radius, *range*. We consider sizes of *map* in the integer interval, $[1..rowsMax] \times [1..colsMax]$, and densities of obstacle in the real interval $[0, densityMax]$ where $densityMax < 1$. We consider positions of a *robot* in any free cell of a given *map* and radii of circular field of view in the integer interval $[0..rangeMax]$.

#obstacles	#rows	#columns	#starts	#instances
0	1	[1-25]	[1-13]	169
0	2	[2-13]	[1-7]	48
0	3	[3-9]	[3-10]	45
0	4	[4-7]	[3-8]	23
0	5	5	6	6
5	5	5	6	300

Table 4.1: Instances of deterministic coverage.

Our set of instances contains two subsets shown in Table 4.1. The first subset contains rectangular maps with no obstacles, sizes ranging from 1 to approximately 25 cells, and where the robot starts on every position (no duplicate position with respect to horizontal, vertical, and diagonal symmetries). The second subset contains 50 square maps of 25 cells containing 5 obstacles, and where the robot starts on 6 distinct positions that are uniformly sampled without replacement. These maps are uniformly sampled without replacement among the set of symmetrically invariant maps (no duplicate maps with respect to horizontal, vertical, and horizontal symmetries).

Solvers.

Solvers are chosen among iterative and recursive best-first search algorithms (abbreviated I-BFS and R-BFS), and iterative-deepening and branch-and-bound depth-first search algorithms (abbreviated ID-DFS and BnB-DFS).

Heuristics.

A first *coverage* heuristic, computes a lower bound on the optimal number of field of view observations the robot has to make in any state, $s = (p, m, c)$, in order to cover the cells that are not yet covered among coverable cells. It relies on P_A^{max}, the maximal connected component of free cells of m containing the position p, whose connectivity depends on the action set A; and

C_A^{max}, the set of all cells that are visible from any cell of P_A^{max}. Precisely, it considers an even more restrictive set, $C_A^{\neg c} \subseteq C_A^{max}$, containing those cells that are still not covered.

$$C_A^{\neg c}(s) = \{p \in C_A^{max} \mid s_c(p) = \neg c\}$$

Maximal coverable cells (MaxCells) (Equation (4.2)), builds a relaxation where the map is transformed into one corridor where all covered cells are on the left of the agent, while remaining coverable cells are on its right. The width of the corridor corresponds to the diameter of the field of view of the robot. This heuristic computes the cost of the optimal trajectory for this relaxation. The optimal trajectory is to keep moving to the right until all coverable cells are covered. It is computed as the ceil of the division of the area of not covered cells in state s by the gap between two minimally overlapping observations. This heuristic is inspired by the counting heuristic that appears in [Li et al., 2012], but here, we normalize the number of not covered cells by the maximum number of cells that are visible in one move. In the following, s is any state from the state space S; a is any action from the action space $A(s)$; $T(s,a)$ is the deterministic (s,a)-transition; $F(s_p, s_m)$ is the set of visible cells from the robot's pose s_p and the environment's map s_m of s.

$$h(s) = \left\lceil \frac{|C_A^{\neg c}(s)|}{\max_{s \in S, a \in A(s), s' \in T(s,a)} |F(s'_p, s'_m) \setminus F(s_p, s_m)|} \right\rceil \tag{4.2}$$

Two other *navigation* heuristics, focus on computing the shortest path cost from the robot to any viewpoint V_i. V_i is the set of all free positions from which any cell of $C_A^{\neg c,i}$ is visible. $C_A^{\neg c,i}$ is the i^{th} element of the partition of $C_A^{\neg c}$ into maximal connected component of not covered cells, where the connectivity relies on navigation actions. Formally, a viewpoint can be described as follows:

$$V_i(s) = \{p \in P_A^{max} \mid F(s_p, s_m) \cap C_A^{\neg c,i}(s) \neq \emptyset\}.$$

Minimum navigation distance (MinDist) (Equation (4.3)), is the minimum cost among shortest paths from the robot to all viewpoints in s. This is a well known heuristic in experimental Robotics for greedy Active Mapping in the frontier paradigm [Yamauchi, 1997]. It represents the cost of navigating from the current position of the agent on a shortest path to its closest viewpoint in order to cover new cells.

$$h(s) = \min_i \min_{v \in V_i(s)} d^*(s_p, v) \tag{4.3}$$

It can be computed by Uniform-Cost Search (UCS) in the navigation space of free cells, starting from the agent position and ending with success when any position of any viewpoint is selected for expansion or ending with failure when the frontier is empty.

Maximum navigation distance (MaxDist) (Equation (4.4)), is the maximum cost among shortest paths from the robot to closest views in s. It represents the cost of navigating on a shortest path to the farthest viewpoint in order to continue coverage.

$$h(s) = \max_i \min_{v \in V_i(s)} d^*(s_p, v) \tag{4.4}$$

It can be computed by UCS as before except that success is delayed to the point when one position of every viewpoint has been selected for expansion, and the lowest-cost has been recorded separately for each of these positions.

Finally, **None** represents the lack of expert knowledge, *i.e.*, $h(s) = 0$ for any input state s.

These heuristics are admissible in the coverage domain, they underestimate the cost of a minimum coverage path from any state. Furthermore, some of their combinations by addition are also admissible. For instance, we will consider Minimum navigation distance and Maximal coverable cells (MinDist+MaxCells) in the following.

Metrics.

First, we rely on usual metrics to monitor solution cost expressed in path length, memory consumption expressed in number of stored nodes, and time consumption expressed in number of generated nodes.

More precisely, for the linear memory search algorithms, BnB-DFS, R-BFS, and ID-DFS, memory consumption corresponds to the maximum number of nodes on and around any searched path, *i.e.*, approximately the lengthiest searched path depth multiplied by its average branching factor in our case.

Concerning the exponential memory search algorithms I-BFS, memory consumption is the maximum size of the interior plus frontier lists during search. This size is related to generated nodes entering the frontier list and expanded nodes entering the interior list.

Second, we sometimes consider the difficulty of a particular instance of the coverage task:

Informally, if we vary each parameter describing an instance of the coverage task separately when keeping all the other parameters fixed: increasing the size of the map may increase the difficulty as the coverage path gets lengthier; increasing the density of obstacles may decrease the difficulty as the number of applicable moves gets smaller; increasing the range of view may decrease the length of the coverage path as more cells are visible at once; changing the starting position usually yields different coverage paths with different costs.

Formally, the difficulty that arises from the interactions between the attributes of a given instance will be summarized as the size of its state space. This size corresponds to the branching factor to the exponent of the depth of the optimal solution of the given instance. We approximate the difficulty as the average branching factor to the exponent of the optimal depth. The average branching factor is obtained *a priori* by summing the number of navigation actions available from all positions and dividing this sum by the number of positions in the given map. The optimal depth is recorded *a posteriori* for any instance that was solved at least once by any solver and heuristic.

Resources.

Resources are limited by a memory-out of 10^6 nodes storage, and a time-out of 2.10^7 node (re)generations.

Details.

Actions are considered in the following order: up, right, down, and left. Node ordering is disabled for depth-first search algorithms. Ties are resolved by favoring lower h-values for nodes

with the same f-key or F-key in best-first search algorithms.

Hereafter, we conduct a first empirical study of optimal solvers with admissible heuristics in the deterministic coverage domain.

4.4.1 Optimal solvers

In this first experimental study, we use the following optimal solvers with an admissible heuristic: I-BFS(A*), R-BFS(A*), ID-DFS(A*), and BnB-DFS(A*), using the node-value computed by A*. Our main objective is to identify the instance difficulty of the coverage task above which any straightforward implementation of these techniques surrenders according to predefined memory-out and time-out limits. The set of instances contains the two distinct subsets shown in Table 4.1. We illustrate the raw data of these experiments in Section B.4.

Solved instances.

We report the number of successful and unsuccessful terminations for all solvers and heuristics in Figure 4.1. Each row corresponds to a search algorithm and each column corresponds to a heuristic.

For all solvers, the coverage heuristics, MaxCells and MinDist+MaxCells, solve more instances than the navigation heuristics, MinDist and MaxDist.
The linear memory solvers BnB-DFS, ID-DFS, and R-BFS fail only with time-out and mostly with the navigation heuristics. The optimal depth is always lower than 100 steps for our instances and the number of successors of each node is lower than 4, hence, a memory limitation of $1.10^6 (\gg 4{\times}100)$ nodes is more than necessary. In these cases, ID-DFS and R-BFS may experience too much node regeneration and BnB-DFS may not efficiently prune the state space.
The exponential memory solver I-BFS fails only with memory-out when using the navigation heuristics. In this case, it could not keep its frontier and interior lists small enough.

Resource consumption.

We plot the distribution of memory and time consumption for all solvers and heuristics in Figure 4.2. Each row corresponds to a search algorithm; memory boxplots are on the left panel; time boxplots are on the right panel; and heuristics are in different colors. Each Tukey boxplot represents the lower quartile, the median, and the upper quartile of the distribution. The vertical red lines represent memory-out on the left panel and time-out on the right panel. Hereafter, we detail these results.

Memory consumption: For the linear memory solvers: the median memory consumption is near 40 nodes for ID-DFS, near 50 nodes for R-BFS, and there are slight variations according to the heuristics. These results show that R-BFS searches deeper than ID-DFS with the same amount of resources.
The median memory consumption is near 100 nodes for BnB-DFS and it is almost constant for all heuristics. This invariance is due to the initial diving phase of BnB-DFS. During, this diving phase, it finds a first suboptimal solution and then it rises to find better solutions. In our implementation, this diving phase depends only on the action order. Hence, this memory

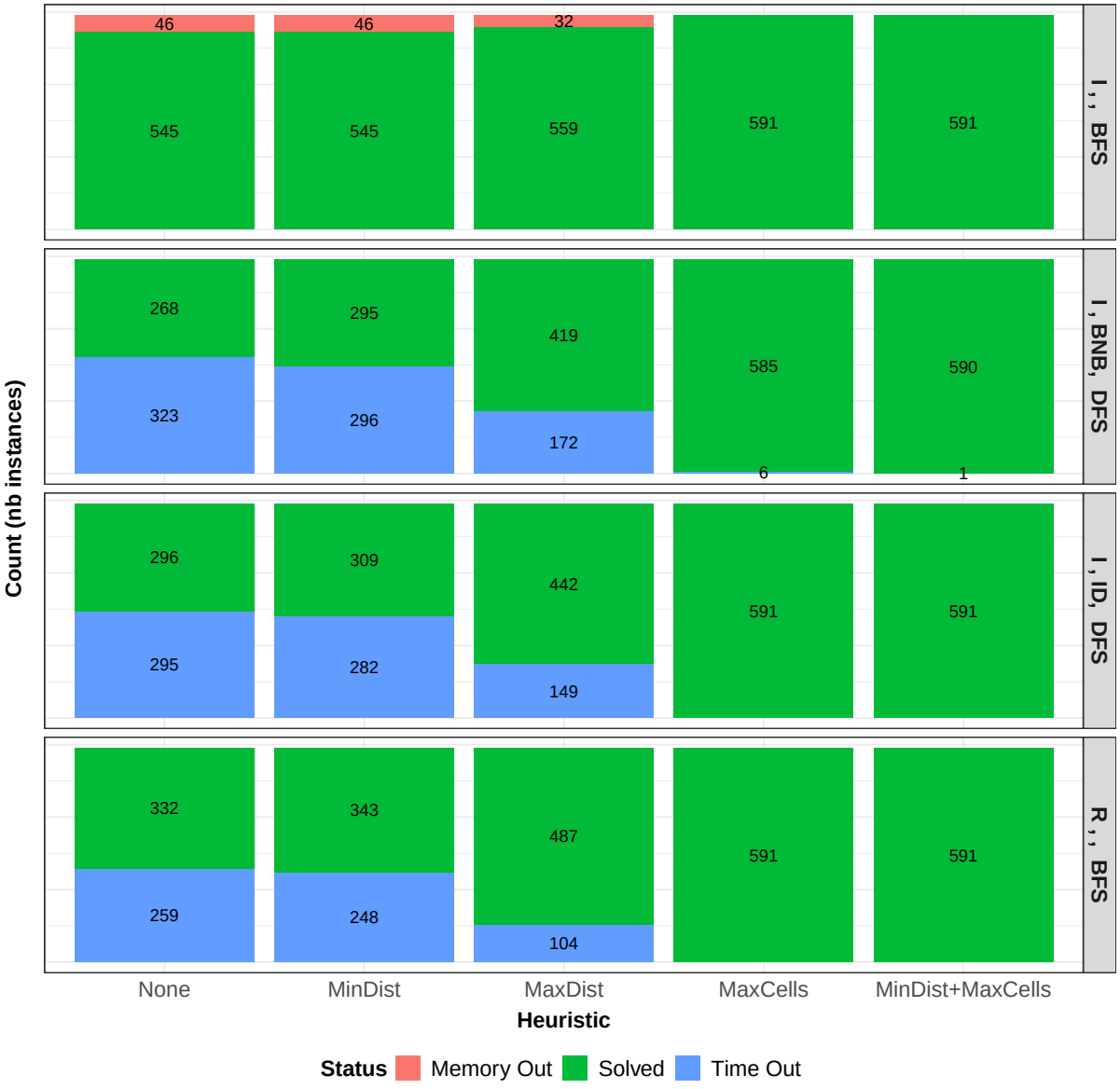


Figure 4.1: Optimally solved instances per algorithm and heuristic.

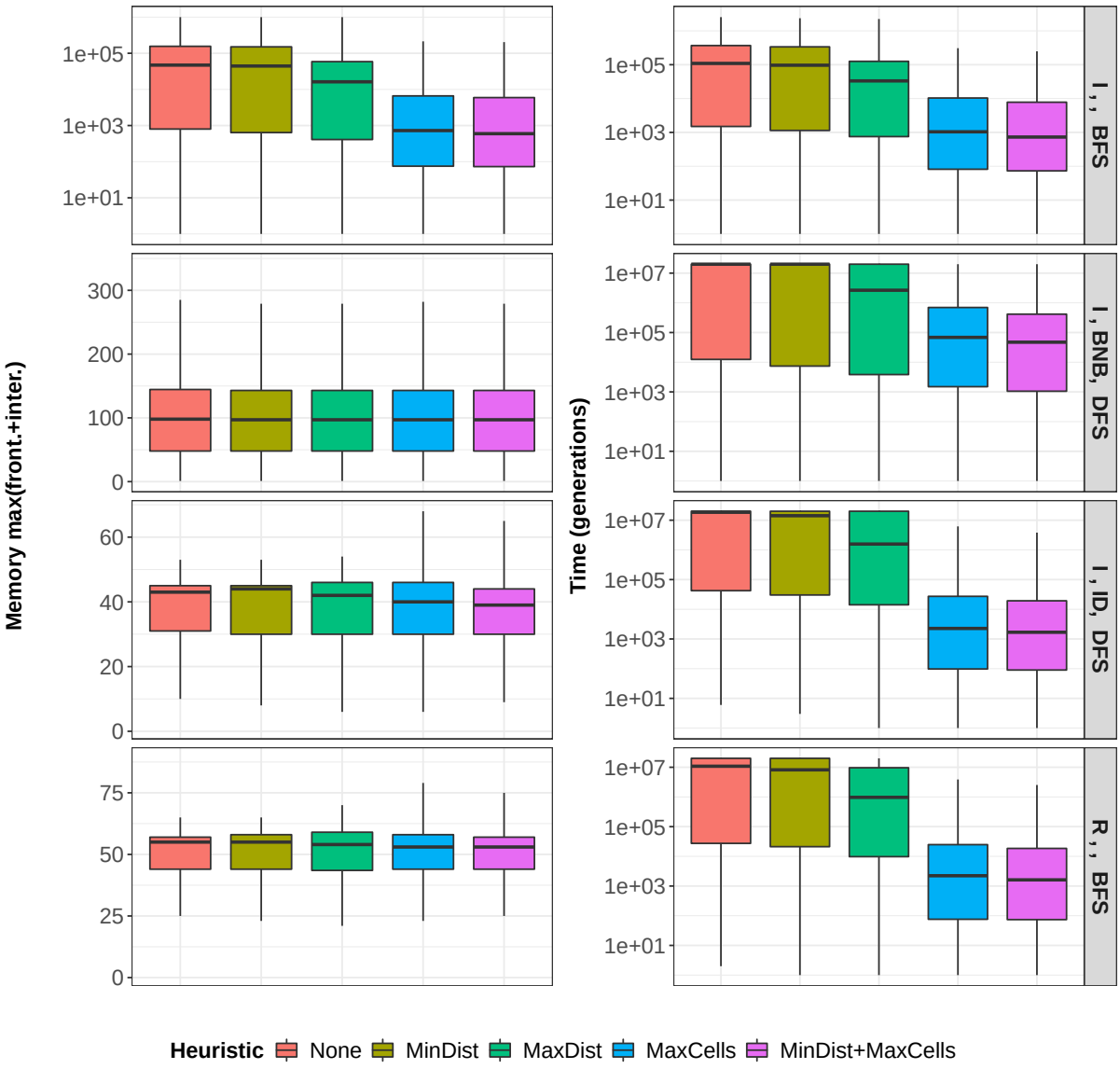


Figure 4.2: Memory and time consumption distribution of optimal solvers per algorithm and heuristic.

consumption is due to storing the first and most costly solution.

For the exponential memory solver I-BFS, this distribution significantly varies according to the heuristic. The median memory consumption is lower than 1.10^3 nodes for the coverage heuristics MaxCells and MinDist+MaxCells and higher than 1.10^4 for the navigation heuristics MinDist and MaxDist. This means that the coverage heuristics are pruning more states during search and can keep their frontier and interior lists small enough to search longer.

Time consumption: For all solvers, the median time consumption of the coverage heuristics is always at least 10 times lower than the navigation heuristics. The ranking between the different heuristics appears clearly: MinDist+MaxCells performs the best, followed by MaxCells, then MaxDist, and finally MinDist.

Cumulative resource consumption.

We plot the empirical cumulative distribution of memory and time consumption in Figure 4.3 to further disambiguate the performances of each solver and heuristic. Each row corresponds to a search algorithm; cumulative memory (respectively time) charts appear on the left (respective right) panel; and heuristics are in different colors on each chart. The vertical red lines represent memory-out or time-out.

Cumulative time consumption: Time charts show a ranking of the different heuristics that is preserved across all algorithms. MinDist+MaxCells dominates MaxCells, which in turn dominates MaxDist, MinDist, and *None* in terms of instances solved for any time limitation. For instance, with 1.10^5 generations: R-BFS can solve approximately 87.5% of the instances using MinDist+MaxCells and only 37.5% of them using MaxDist (50% less); conversely, for solving 50% of the instances: R-BFS requires about 1.10^3 generations using MinDist+MaxCells and 1.10^6 using MaxDist ($\times 1.10^3$ worse).

Cumulative memory consumption: Memory charts show a different tendency for the linear memory solvers. For BnB-DFS, the cumulative distributions perfectly match for all heuristics. This is explained again by referring to its heuristic invariant descending phase in our implementation.
For ID-DFS and R-BFS, with the coverage heuristics, MaxCells and MinDist+MaxCells, the maximum memory consumption is slightly above those observed for the navigation heuristics. Indeed, the coverage heuristics allowed to search deeper and found solutions when the others failed prematurely due to time issues (see the corresponding time-outs).

Resource vs difficulty.

We plot the resource consumption against the instance difficulty for each solver and heuristic in Figure 4.4. Each row corresponds to a search algorithm; memory (respectively time) charts appear on the left (respective right) panel; heuristics are in different colors on each chart; the x-axis represents the difficulty of the instances. These charts do not show data for unsolved instances, hence, when the regression is truncated at a given value of difficulty, it means that there is no solved instance above this value for this combination of algorithm and heuristic.

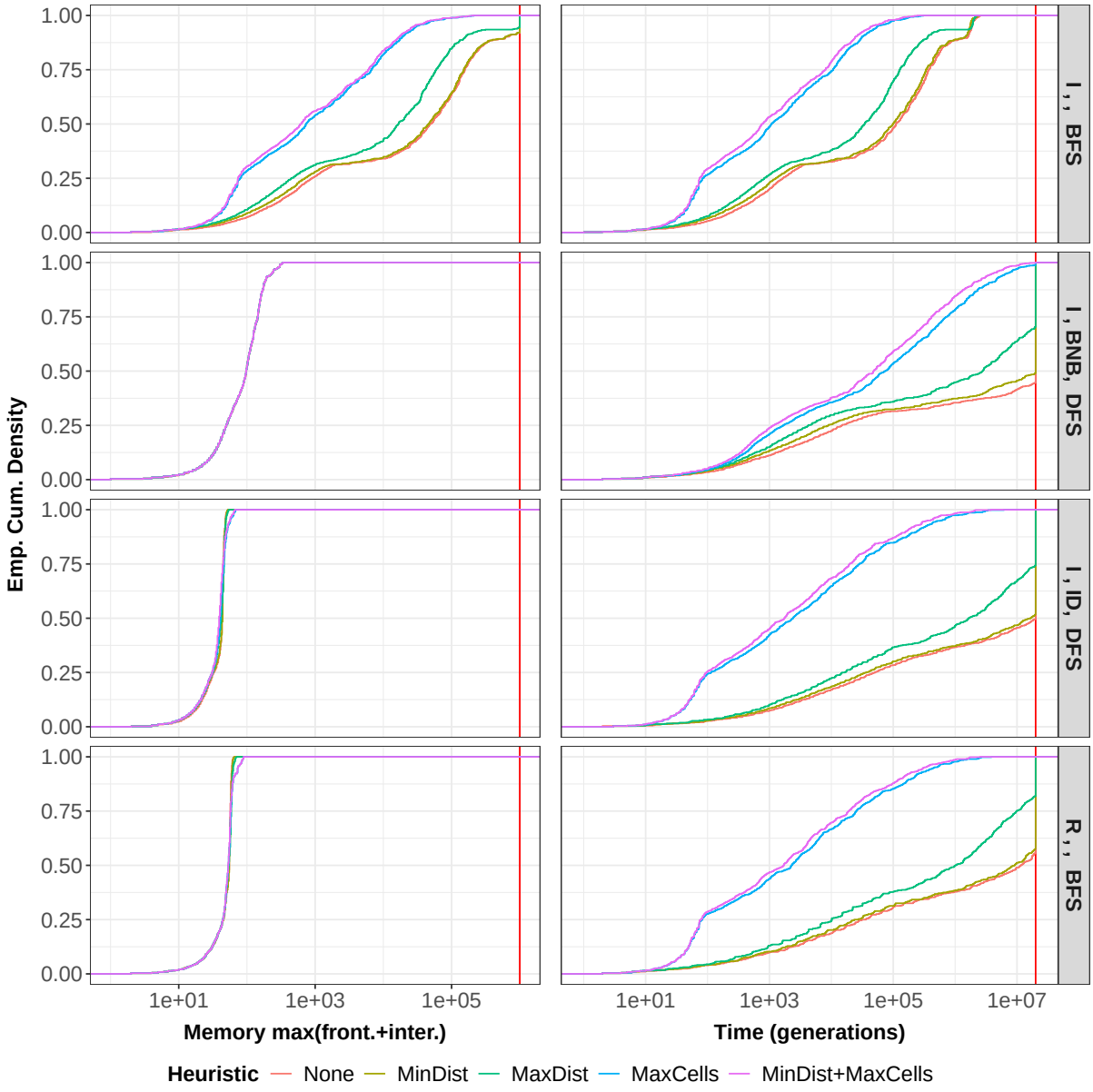


Figure 4.3: Cumulative memory and time consumption distribution per algorithm and heuristic.

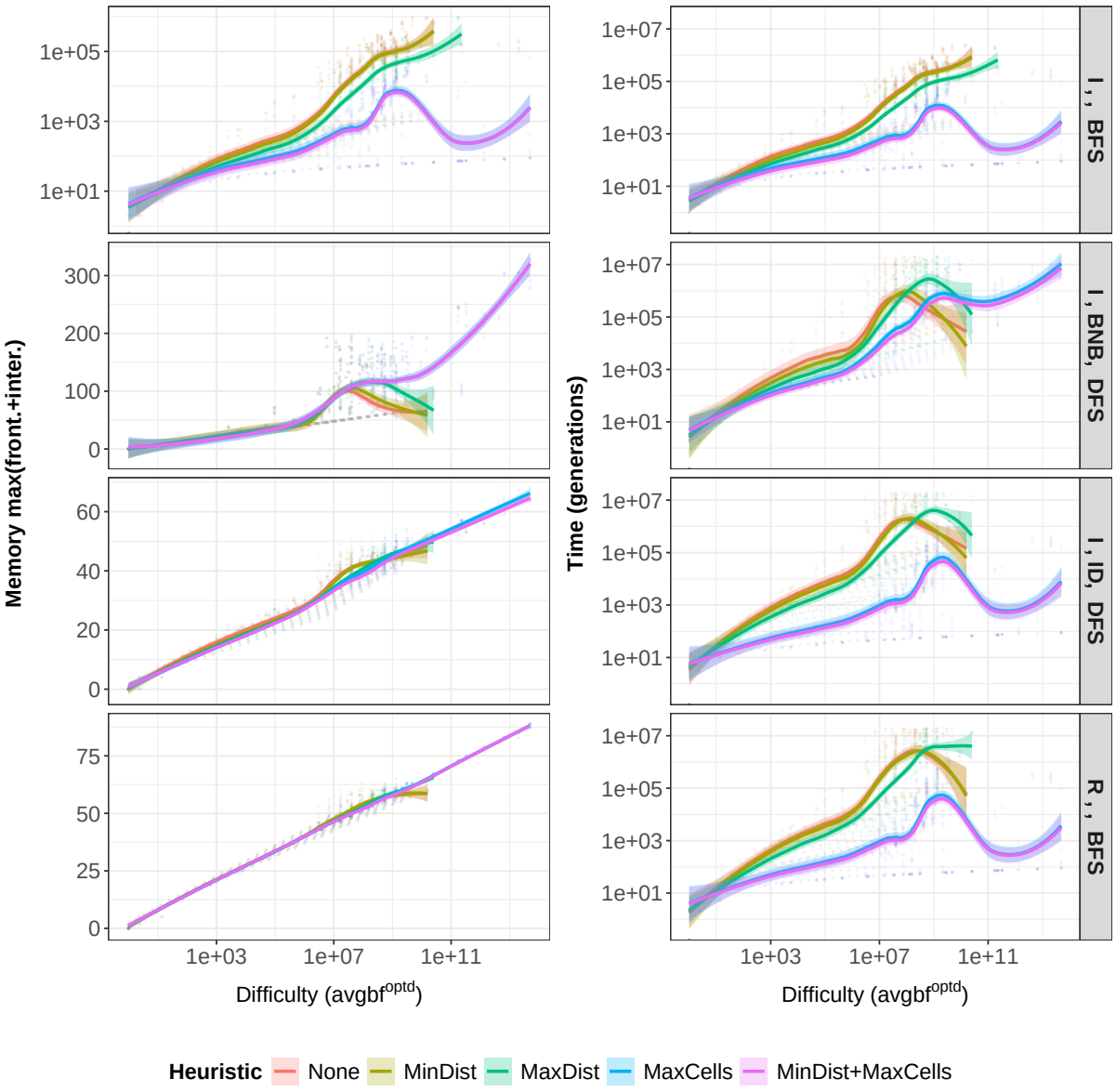


Figure 4.4: Memory and time consumption against difficulty: The instances that were not solved are not shown, each dot is a raw data, each thick line is obtained by a locally weighted polynomial regression (LOESS(0.1)) and its shaded area is a 0.95 confidence interval.

The memory charts are truncated for difficulty above 1^{10} when using the navigation heuristics, but the coverage heuristics could solve instances of higher difficulty. The time charts show that the difference in time consumption between the coverage and the navigation heuristics steadily increases with the difficulty. The coverage heuristics provide a speed-up between 10^1 to 10^3 at a difficulty of 10^9 across all solvers when compared to the navigation heuristics. The difference between MaxCells and MinDist+MaxCells is less visible, we expect this disambiguation to happen clearly with more difficult instances.

Summary of experimentations with optimal solvers.

We evaluated combinations of representative off-line shortest path solvers with admissible heuristics, for optimally solving small instances of the deterministic coverage domain. This comparison was made in terms of number of solved instances, resource consumption, and scalability to difficulty, with fixed amount of time and memory.

We demonstrated that some combinations of solvers and heuristics are able to solve optimally, almost all selected instances. For instance, I-BFS performed better than every other solver independently of the heuristic, and coverage heuristics performed better than the distance heuristics independently of the solver. The best performances are attained when combining I-BFS with the coverage heuristics under our resource constraints.

We now turn our attention to the performances of suboptimal solvers, that is we discard optimality-seeking, but still want guarantees in terms of execution costs. Indeed, optimality seeking is seldom necessary in real world Robotics, as such deliberation is prohibitive before any execution takes place. In coverage or mapping applications, one would prefer a robot that thinks and acts fast, to one that thinks too much before acting optimally (if possible at all). Hence, we will assess how the controlled increase in solution costs reduces the resource requirements.

4.4.2 Suboptimal solvers

In this second experimental study, we use the following suboptimal solvers: Iterative Weighted A* (I-BFS(wA*)), Iterative-Deepening Weighted A* (ID-DFS(wA*)), Recursive Weighted A* (R-BFS(wA*)), and Depth-First Branch-and-Bound Weighted A* (BnB-DFS(wA*)), obtained by overweighting an admissible heuristic with a scalar weight w ($2 \geq w > 1$). We discretize the range of weights from 1 to 2 by an increment of 0.1 [13]. We focus on the four non-null heuristics, MinDist, MaxDist, MaxCells, and MinDist+MaxCells. Our main objective is to identify the tradeoff between execution and search costs when dealing with suboptimal off-line search algorithms. The set of instances contains the two distinct subsets shown in Table 4.1, as in the previous experimental study of optimal search algorithms.

Solved instances.

We report the number of successful terminations when increasing the heuristic weight in Figure 4.5. Each column corresponds to a search algorithm; each row corresponds to a heuristic; each barplot represents the number of solved instances for a given heuristic weight. For all solvers, the global tendency is a reduction of time-out and memory-out when increasing the

[13] As a reminder, overweighting an admissible heuristic with a weight $w = 1+\epsilon > 1$ guarantees that the returned solution is at most ϵ-suboptimal.

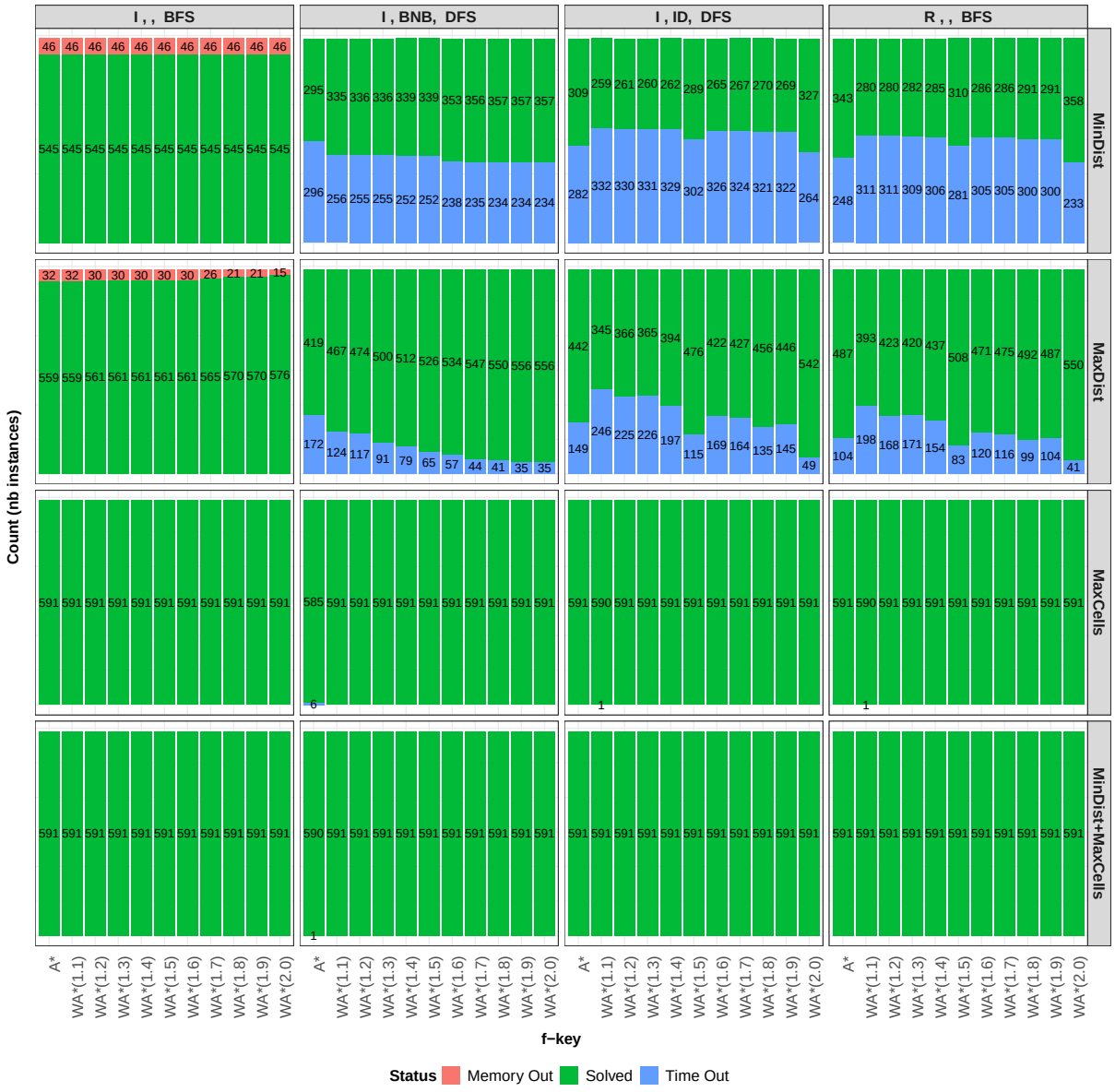


Figure 4.5: Suboptimally solved instances per algorithm and weighted heuristic.

heuristic weight, but there are still some local artifacts.

For all search algorithms, the coverage heuristics (MaxCells and MinDist+MaxCells) already solved all instances with a weight of 1.2. Hence, these heuristics could solve all instances with a solution cost not higher than 20% of the optimal under our resource limitations. In the following, we describe in greater details the change in solution costs and resource consumption when increasing the heuristic weight.

No search algorithm solved all instances when using the navigation heuristics (MinDist and MaxDist). At the global level, MaxDist suboptimally solved more instances faster than MinDist when increasing the weight. Still, we observe some artifacts as the number of solved instances sometimes decreases along the range of weights. I-BFS and BnB-DFS are not prone to this local tendency whereas ID-DFS and R-BFS are. Indeed, the latter algorithms are known to suffer from node reexpansion. Moreover, node reexpansion worsens with real-valued nodes as it may lead to regenerate entire subtrees in order to expand only a single additional node.

Solution cost.

We report the distribution of execution cost for every solver and heuristic when increasing the heuristic weight in Figure 4.6. Each column corresponds to a search algorithm; each row corresponds to a heuristic; and each Tukey boxplot summarizes the distribution of execution cost for a given heuristic weight. We used a default figure of 100 as an upper bound on the solution cost for unsolved instances. At the global level, with the navigation heuristics MinDist or MaxDist: the median is constant for I-BFS, monotonically decreasing for BnB-DFS, and nonmonotonic for ID-DFS or R-BFS; and with the coverage heuristics MinDist+MaxCells and MaxCells: it is monotonically increasing for all solvers.

The navigation heuristics did not solve all instances, hence all profiles witness either an increase in solution costs due to more suboptimal solutions being found, or a reduction in solution costs due to newly solved difficult instances. In the latter case, the true solution cost replaces the default figure of 100.
When increasing the heuristic weight from 1 to 2 with MinDist: I-BFS shows no difference, this is related to no newly solved instances and suggests that pruning occurs too rarely as the heuristic may not be inadmissible enough; BnB-DFS is monotonically decreasing, this is related to new instances being solved (replacing the default figure) and suggests that bounding occurs more often; ID-DFS and R-BFS show more variability, they are both affected by node reexpansion and this may be accentuated with real-valued nodes (but attenuated for the weight 1.5) [14].
Regarding MaxDist, the execution performances show the same tendency but are comparatively better.

The coverage heuristics solved all instances with a weight above 1.2. Hence, all profiles mainly witness the increase in solution costs due to more suboptimal solutions being found.
When increasing the suboptimal weight to 2 with MaxCells: I-BFS and R-BFS show a slight increase of the median cost from 21 to 24 (up to 14% suboptimal) – indeed both algorithms

[14] To illustrate this idea: let consider our set of weights $w \in [1.0, 0.1, 2.0]$ and an interval of integer heuristic values $h \in [1..5]$. When multiplying our weights to these heuristic values, the histogram of wh-values that are still integers is {5,0,1,0,1,2,1,0,1,0,5}. The real weight that attains the highest number of integer values is 1.5.

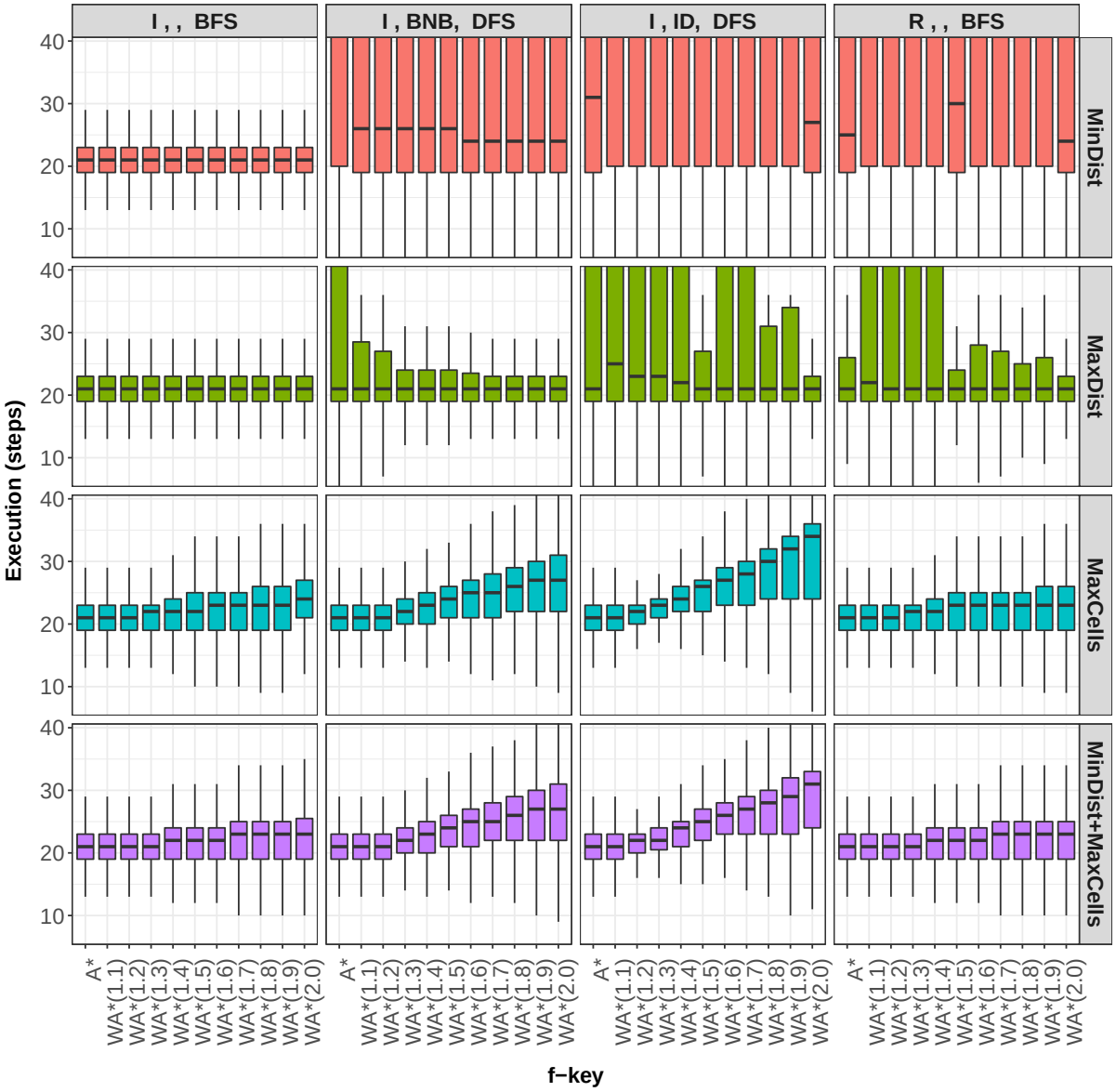


Figure 4.6: Suboptimal execution cost per algorithm and weighted heuristic.

expand nodes in the same order up to tie-breaking and find the same solutions. ID-DFS and BnB-DFS show a more pronounced growth, the median increases from 21 to 34 respectively 27 (up to 62% respectively 29% suboptimal).
Similarly, with MinDist+MaxCells: for I-BFS and R-BFS, the median cost slightly increases from 21 to 23 (up to 10% suboptimal); and for ID-DFS and BnB-DFS, the median cost substantially rises from 21 to 31 respectively 27 (up to 47% respectively 29% suboptimal).

Resource consumption.

We report the distribution of time and memory consumption of each solver and heuristic when increasing the heuristic weight in Figures 4.7 and 4.8. Each column corresponds to a search algorithm; each row corresponds to a heuristic; and each Tukey boxplot represents the distribution of resource consumption for a given heuristic weight.

Time consumption: We report the distribution of time consumption in Figure 4.7. At the global level, the median time consumption is a decreasing function of the weight for all search algorithms and heuristics, up to the noise induced by real-valued nodes in ID-DFS and R-BFS.

The navigation heuristics did not solve all instances, hence we rely on the median speed-up they produce when subject to suboptimal weights; although this is only a lower bound on the true median speed-up, *i.e.*, if we had access to the true time consumption necessary to solve all instances.
When increasing the weight to 2 with MinDist: I-BFS and BnB-DFS produce monotonically decreasing median profiles (speed-up of 1.1 and 2.8); and ID-DFS and R-BFS are still affected by real-valued nodes and produce a speed-up of 1.4 and 1.3.
Concerning MaxDist, the tendency is emphasized with medians that are consistently lower than for MinDist: I-BFS and BnB-DFS respectively produce a speed-up of 3.0 and 15.2; and ID-DFS and R-BFS produce a median speed-up of 4.7 and 4.8.

The coverage heuristics nearly solved all instances, hence we can rely on the true median speed-up.
When increasing the weight to 2 with MaxCells, we generally observe higher speed-ups than for navigation heuristics: I-BFS and BnB-DFS respectively produce a speed-up of 7.6 and 37; and ID-DFS and R-BFS produce a speed-up of 9.5 and 8.1.
Concerning MinDist+MaxCells, the tendency is similar: I-BFS and BnB-DFS produce a speed-up of 2.7 and 44; and ID-DFS and R-BFS produce a speed-up of 6.9 and 2.7.

Memory consumption: We report the distribution of memory consumption in Figure 4.8. At the global level, the median is constant for all heuristics with BnB-DFS; monotonically decreasing for all heuristics with I-BFS; nonmonotonic for MinDist and MaxDist, respectively increasing with MaxCells and MinDist+MaxCells, for ID-DFS and R-BFS.

When increasing the suboptimal weight to 2 with I-BFS: the median memory consumption decreases from 595 to 181 (3.3 space-saving factor) when using MinDist+MaxCells; from 726 to 96 (7.56 space-saving factor) when using MaxCells; from 16198 to 5466 (3 space-saving factor) when using MaxDist; and from 44374 to 40948 (1.1 space-saving factor) when using MinDist.

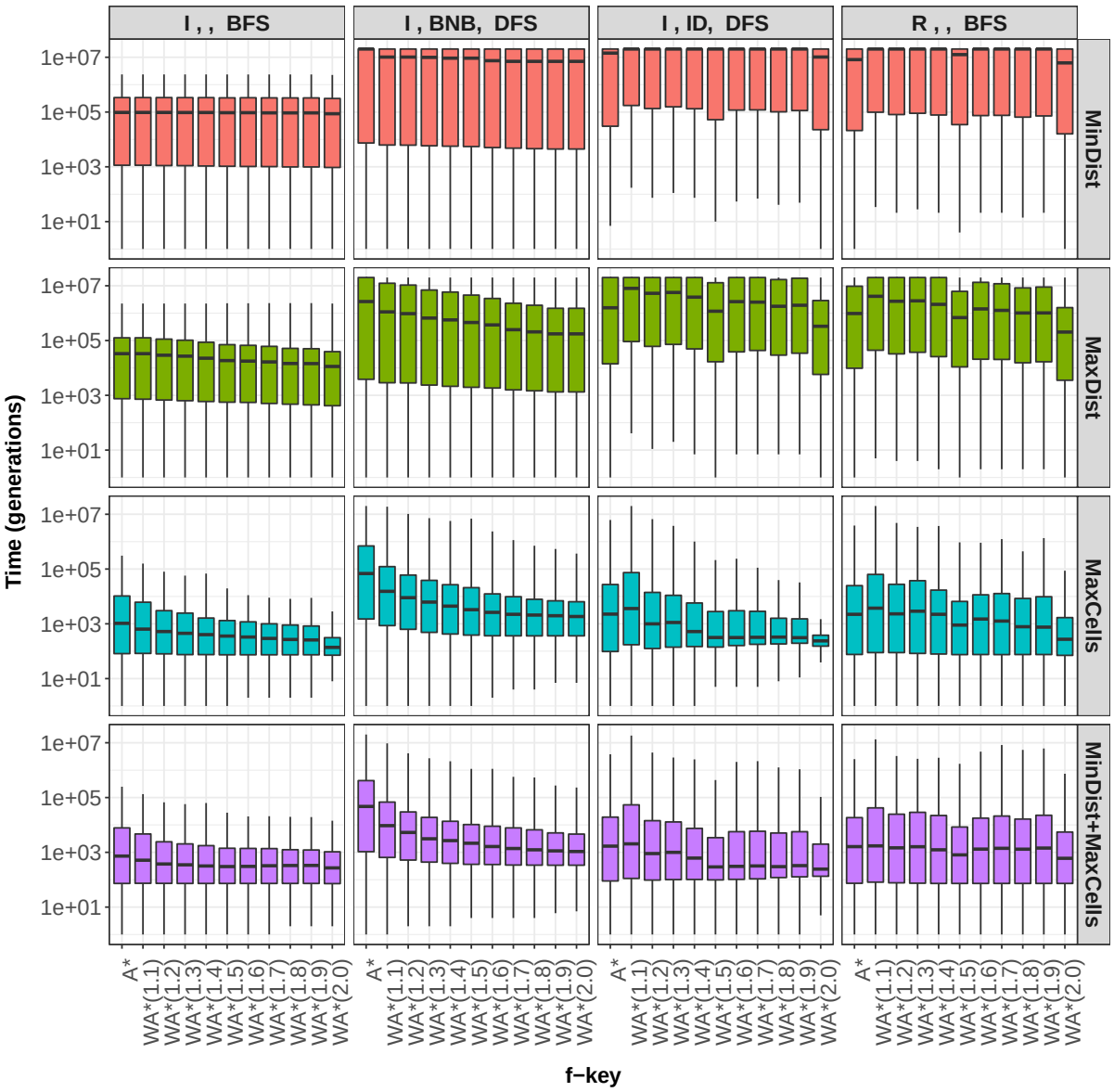


Figure 4.7: Time consumption distribution of suboptimal solvers per algorithm and weighted heuristic.

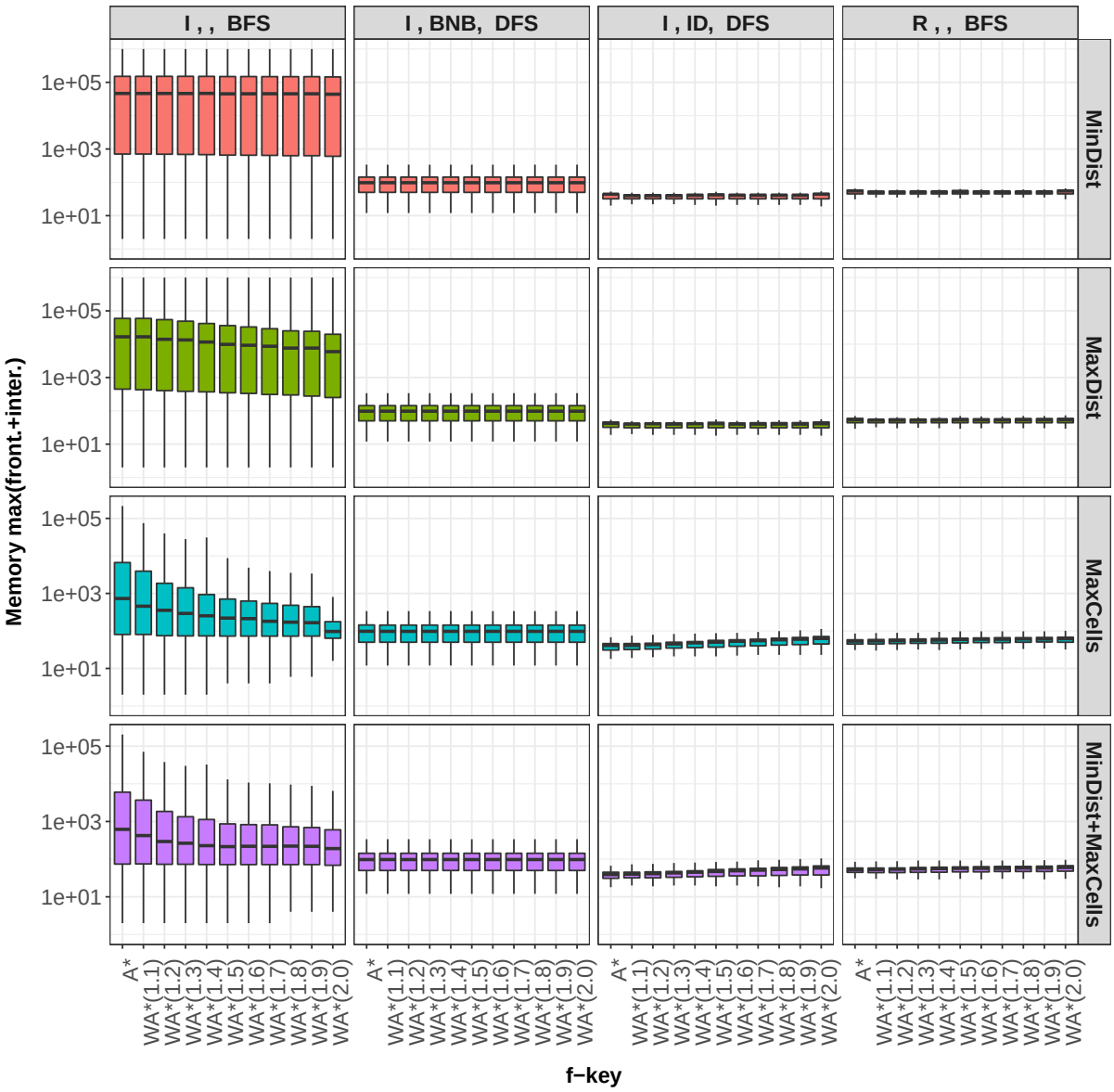


Figure 4.8: Memory consumption distribution of suboptimal solvers per algorithm and weighted heuristic.

With BnB-DFS: memory consumption is constant for all heuristics and weights. Indeed, the lengthiest searched path is obtained during its initial state space diving phase, which is heuristic invariant in our implementation. Its lower, median, and upper quartiles are respectively 48, 97, and 143 whatever heuristic and weight.

With ID-DFS, the median increases from 39 to 58 (1.48 space-increase factor) when using MinDist+MaxCells, and from 40 to 63 (1.57 space-increase factor) for MaxCells. It remains around 42 using MaxDist, and 44 for MinDist.

With R-BFS, the median memory consumption increases from 53 to 59 (1.11 space-increase factor) when using MinDist+MaxCells; and from 53 to 61 (1.15 space-increase factor) when using MaxCells. It remains around 54 when using MaxDist; and 55 when using MinDist.

Summary of experimentations with suboptimal solvers.

We evaluated combinations of off-line solvers with inadmissible heuristics for suboptimally solving small instances of the deterministic domain in a controlled manner. This comparison was made in terms of number of solved instances, suboptimality, speed-up, and space-saving or space-increase factors, with fixed amount of time and memory.

We experimentally demonstrated that the number of solved instances globally increases for all algorithms and weighted heuristics (except for I-BFS and MinDist). We assumed that the local nonmonotonic tendency observed for ID-DFS and R-BFS could be explained by real-valued nodes which aggravates node reexpansion, but this should be further investigated.

For coverage heuristics with a weight of 2 (at most 100% suboptimal): the median solution cost is between 10% to 62% suboptimal; the median speed-up factor is between 2.7 and 44; and the median space-saving factor is between 3.3 and 7.56 for the exponential memory solver, whereas the median space-increase factor is between 1.11 and 1.58 for the others (except BnB-DFS where it remains constant). With these heuristics, the median loss in solution quality is more than compensated in terms of median resource consumption.

For navigation heuristics with a weight of 2: the median cost does not change much with I-BFS, suggesting that these heuristics are not yet inadmissible enough; the median or upper quantiles of the costs decrease for the other solvers, suggesting that the default figure gets replaced by the true cost of newly solved instances; the lower bound on the median speed-up factor is between 1.1 and 15.2; and the median space-saving factor is between 1.1 and 3 for the exponential memory solver, whereas it remains constant for the others. With these heuristics, the increased number of solved instances allowed to reduce the median time consumption for all solvers; the space-saving mainly witnesses more pruning when using MaxDist with I-BFS. Assessing the increase in execution costs in this case should be further investigated with already solved instances only.

We now turn our attention to the progressive behavior of anytime solvers such as BnB-DFS. This last experimental study seeks to demonstrate the effectiveness of BnB-DFS in finding a first solution quickly and then improving it over time. The other reported (sub)optimal solvers are not suited for instances whose difficulty do not allow to output a solution in a given amount of time before the execution starts. This is why we currently envision them only for building

benchmarks. However, anytime solvers are interruptible and output the best solution found so far upon interruption, which may be of interest in experimental Robotics. For instance, the good solution obtained by a greedy algorithm on an instance of the AC problem can be improved by BnB-DFS and will eventually converge to the optimal solution.

4.4.3 Anytime solvers

In this last experimental study, we are interested in the anytime behavior of some off-line solvers, more particularly we focus on BnB-DFS(wA*), that is BnB-DFS using wA* for node-pruning. Our implementation is heuristic dependent for upper bounding solution costs, and heuristic invariant for ordering children nodes during the depth-first search. We consider all heuristics, namely MinDist, MaxDist, MaxCells, MinDist+MaxCells, and None. We consider the following heuristic weights, 1, 1.5, and 2, to overweight an initially admissible heuristic and make it inadmissible. That is, with these weights, we can respectively obtain at most a 0%, 50%, and 100% suboptimal solution upon convergence (given sufficient resources). Our main objective in this study is to evaluate the anytime behavior of BnB-DFS(wA*) when using different heuristics and suboptimal weights.
Although the whole set of instances was input to BnB-DFS, we only illustrate its anytime behavior on a single empty square map of 25 cells by considering all possible starting positions for a single robotic agent.

Solution cost vs resource consumption.

We plot solution costs against time consumption for BnB-DFS in Figure 4.9. Each row corresponds to a different starting location on the 5×5 empty grid; each column is a different node-pruning value used by BnB-DFS; and heuristics are in different colors on each chart. The vertical colored lines represent the time of convergence, that is when the algorithm can guarantee the quality of the returned solution. The vertical red lines represent time-out. Memory consumption is not reported, it measures the number of nodes stored to find the initial solution (solution depth plus siblings along trajectory). Indeed, this initial solution is usually the deepest and hence the most space demanding. In our implementation, this first solution is not affected by heuristics and weights.

This initial solution invariance for a given instance is noticeable on the time performance profile, as the first solution for all heuristics and weights is systematically the same for a given starting location (the upper left starting point in the charts). Nevertheless, once this first solution is found, the different heuristics produce different performance profiles.

At a global level, we usually observe the following ranking, when comparing the performance profiles of the heuristics, in terms of solution quality during search: None, is the worst, followed by MinDist, MaxDist, MaxCells, and MinDist+MaxCells. Still, there are some artifacts where the order of the two coverage heuristics are inverted.

The navigation and none heuristics do not converge to an ϵ-optimal solution before time runs out, but the coverage heuristics do for any starting position in the chosen map and any heuristic weight. This is noticeable by the vertical lines that are drawn before the time-out (vertical red line). Nevertheless, solutions output by the former heuristics are never more than 200% suboptimal for all cases.

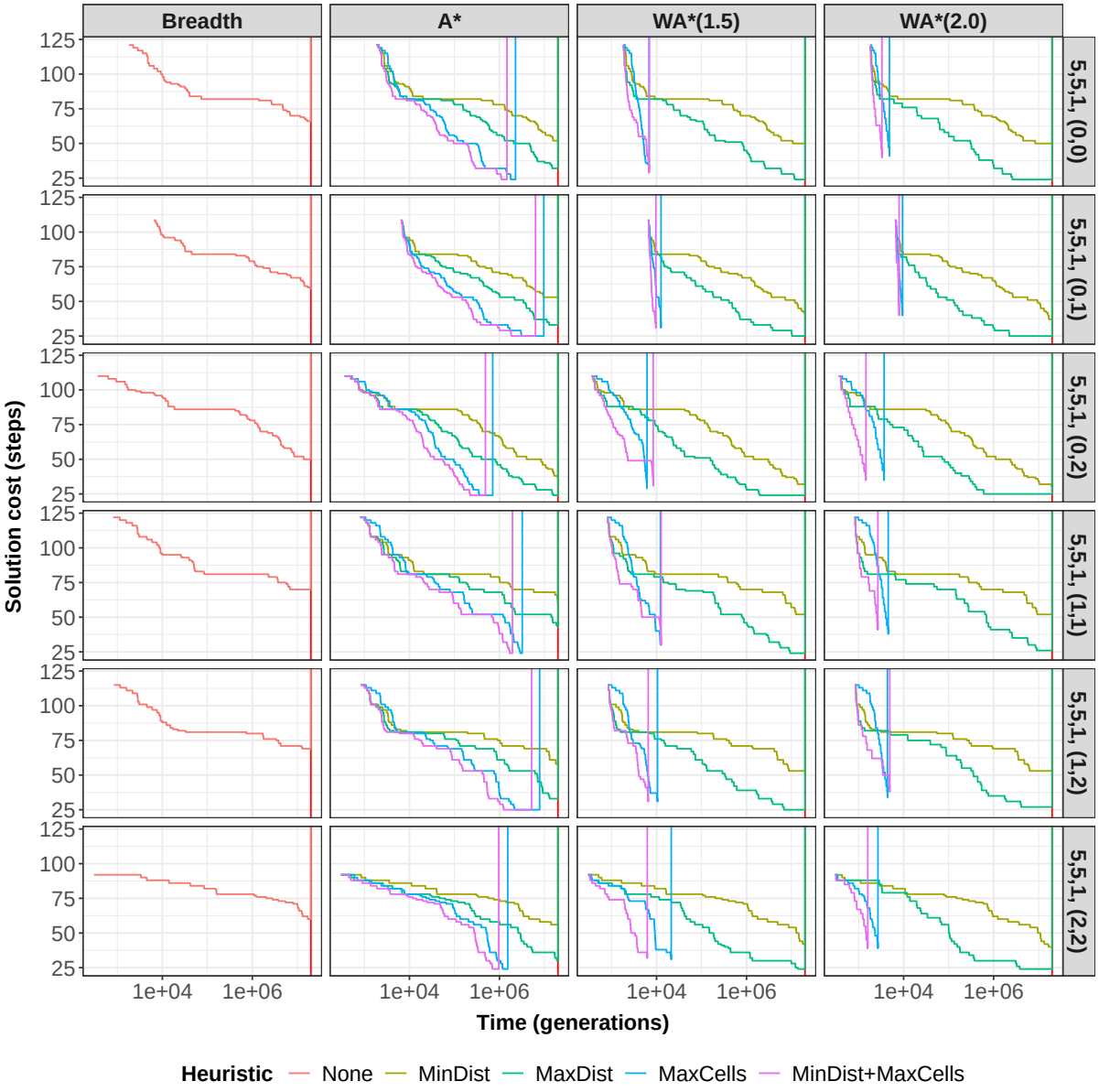


Figure 4.9: Performance profile of Branch-and-Bound Depth-First Search (BnB-DFS) for each starting position on a 5x5 empty map: solution cost in number of steps versus computation time in number of node generations.

Moreover, we also observe that for coverage heuristics, convergence to suboptimal solutions is faster when increasing the weights. For instance, if we consider the first row, and compare the solution cost at convergence by trading it off with the time consumption for the weights 1, and 2: the time consumption went from over 10^6 to below 10^4 generations, that is a speed-up factor over 10^2, and the solution cost increased from 25 (that is optimal) to less than 44 (that is below 76% suboptimal). In the same row, concerning the navigation heuristics at time-out, we observe that MinDist does not improve, whereas MaxDist does, when varying the weight. More precisely, its solution cost at time-out is optimal for both weights, but it did not converge yet.

Summary of experimentations with anytime solvers.

To summarize, although BnB-DFS does not appear as the best candidate among optimal neither suboptimal solvers in our two previous experimental studies, its anytime property is an important asset. It allows outputting a stream of solutions with increasing quality even when it fails to converge to the optimal or the ϵ-suboptimal solution on time, whereas the others output nothing at all when they fail as optimal or ϵ-suboptimal solvers. This property can be used to improve the solution quality of an ant or frontier agent that addresses deterministic coverage after their first trial.

4.5 Conclusion

In this chapter, we studied the resolution of the single robot deterministic coverage task. We relied on the Single-Robot AC decision-making model (Section 2.3.1) and on standard planning techniques for SDM in the corresponding formalism (Definition 2.1). First, we conducted a brief review of the state of the art in off-line search. Then, we selected representatives of the most popular frameworks of off-line heuristic search. Finally, we assessed their performances for computing optimal, suboptimal, and anytime solutions to instances of the coverage model; we provided expert knowledge with three simple heuristics: two navigation heuristics inspired by distance-based strategies in the frontier paradigm for AM and a coverage heuristic that counts the number of remaining cells to cover.

The optimal benchmark demonstrates the utility of the coverage domain for transparently applying general planning techniques and tuning them with specialized heuristics for AM. It allows to evaluate such algorithms and heuristics in the average case to build empirical knowledge regarding optimality seeking in AC.

The suboptimal benchmark demonstrates the possibility to reduce the requirements regarding optimality in a controlled manner. We solved instances of the coverage task by modulating expert knowledge in order to guide and prune search. It provides ϵ-suboptimal baselines for the AC and AM algorithms, in case optimal baselines are intractable.

The anytime benchmark demonstrates the possibility to automatically improve satisfiable solutions. It is encouraging as it opens the route for improving the solutions obtained by frontier or ant agents for instance. For this purpose, it suffices to implement a node-ordering strategy that imitates the behavior of such agents.

We provided novel baselines to compare the performances of AC and AM algorithms

on small environments for a single robot. The habit in AM is to directly compare the average performances of greedy mapping strategies on several environments [Amigoni, 2008, Holz et al., 2010, Bautin et al., 2012, Faigl et al., 2012]. More recently, comparisons based on the competitive ratio of greedy strategies have been suggested. The competitive ratio is the maximum ratio of performances between a given mapping strategy and the Minimum Coverage Path (MCP) over a set of environments. Good exploration strategies are able to keep their competitive ratio as low as possible over such environments. To compute this ratio, [Faigl and Kulich, 2015] proposes to approximate the MCP by sampling thousands of satisfiable coverage trajectories; and [Li et al., 2012] proposes a proof of concept to compute the MCP by the means of heuristic search. We contributed to this approach by extending their proof of concept to a more comprehensive benchmark with other standard techniques and heuristics.

Furthermore, we wanted to encourage the adoption of domains for which (sub)optimal baselines are known on a large set of small instances along with the resources that were necessary to compute them. Such studies will help in understanding when (sub)optimality is tractable in coverage tasks. Additionally, all solved instances can be used to better approximate the policy of a planning agent for exploration as illustrated in Table C.2. There are many perspectives to this work, *e.g.*, reproducing this study for centralized decision-making with a team of robots; learning from (sub)optimal coverage behaviors on small instances and then generalizing to bigger instances.

In the next part, we will consider more realistic setups in which many robots navigate in dynamic environments and make decisions in a decentralized manner.

Part III

Short-Term Planning for Deterministic Active Mapping and Clearest Path Obstacle Avoidance

Chapter 5

Decentralized Active Mapping in Crowded Environments

Contents

5.1 Introduction

The Multi-Robot Active Mapping (MRAM) problem consists in coordinating a team of robots to build a map of an unknown environment. This problem raises several issues concerning task allocation, robot control, and communication among others. We study MRAM in crowded environments, where the flows of pedestrians could severely impact the performances of the fleet. Nevertheless, the pedestrians demonstrate adaptive skills for taking advantage of these flows while moving. Therefore, we propose a novel exploration strategy that explicitly allows for a simple type of Human-Robot Interaction (HRI) to exploit these moving abilities.

Our proposed approach for exploration in crowded environments combines the classical frontier paradigm with a novel interactive paradigm. In the frontier paradigm, robots are driven toward the immediate source of information, that is the limit between what is already known about the environment from what remains unknown about it. In the interactive paradigm, robots are driven toward a nearby moving object, that is another object that intelligently navigates in the environment. Hence, we consider interactions where robots can locally choose a pedestrian guide

to follow. Additionally, we parametrically balance interaction and frontier assignments of mobile robots during exploration. Finally, we evaluate to which extent accounting for interactions with moving objects could impact the performance in terms of coverage ratio, traveled distance, and elapsed time to completion.

Mobile robots already intervene in our daily life to provide services, *e.g.*, guidance [Burgard et al., 1999] and assistance [Kosuge and Hirata, 2004]; or even leisures, *e.g.*, company [Arkin et al., 2003] and dance [Michalowski et al., 2007]. The presence of mobile robots into daily activities should take people into account whilst seeking social compliance. Therefore, human activities and motion patterns are already studied [Bennewitz et al., 2005], so that a robot can learn a model of the human behavior to generate a socially compliant control. The rationale of our study is that by observing pedestrians walking nearby, a robot could imitate them to efficiently navigate in the environment.

In this study, MRAM consists in controlling mobile robots in order to build the map of an unknown environment. Accounting for moving objects' flows into a robotic exploration system in simulation can constitute an intermediate step toward deploying robots into real crowds. Indeed, it paves the way for subsequent and more advanced HRI-based approaches to exploration. However, this additional feature raises new concerns regarding clean reconstruction and efficient coordination.

Mapping dynamic environments

In our context, the process of building a map in a crowded environment refers to reconstructing the elements of the background: the walls and static objects. More precisely, this process involves filtering the sequence of actions and perceptions of the robots to incrementally build an occupancy grid.

In general, the perception of any mobile robot, that is the measurements it gathers through its sensors, can be (1) biased due to calibration or intrinsic inaccuracy; (2) incomplete due to occlusions induced by moving or static obstacles; (3) aliased due to the inability to distinguish the current state; and (4) ambiguous due to the difficulty to distinguish between moving and static objects.

These issues hinder the filtering process of mapping without even considering the active control computed on-board by the robots. In the following, we focus on active control and assume that: (1) sensors are perfect and (4) there is no ambiguity between moving and static objects. Otherwise, we agree with the hypotheses of (2) occlusion and (3) aliasing.

These assumptions are realistic, as techniques already exist for filtering such measurements in the literature. For instance, an occupancy grid mapping technique with three states of occupancy (free, occupied, animated) is reported in [Dubois et al., 2011].

Navigating dynamic environments

In our context, the flow of pedestrians creates temporal reachability of known or unknown areas. In other words, the reachable space evolves dynamically according to the flow. Hence, exploration targets may appear and disappear according to their current reachability. This can

cause the exploration mission to terminate earlier than expected, if a door remains closed or if a crowd blocks a corridor for example.

Nevertheless, pedestrians understand the dynamics of their environment, can sense, decide and act adequately. In this sense, we assume that every pedestrian has an adaptive behavior that depends on its local environment and allows to walk through dense areas. We want to exploit such pedestrian skills as possible heuristics for the exploration task. Therefore, we propose a weighted heuristic for selecting areas to explore or initiating human-robot interactions.

Firstly, in Section 5.2, we report the state of the art in MRAM and we situate our approach among HRI applications in mobile robotics. Secondly, in Section 5.3, we formalize the multi-agent system for exploration in crowded environments and present the framework of our study. Thirdly, in Section 5.4, we define the hybrid exploration approach (robot-frontier/interaction) and propose a human-aware exploration heuristic for establishing pedestrian following interactions. Then, in Section 5.5, we conduct experiments with our hybrid approach in simulation to underline the variability of the performance depending on the environment. Finally, in Section 5.6, we discuss our results and perspectives regarding machine learning for adaptive heuristic parameterization.

5.2 Related Work

First, this section presents previous work in the field of MRAM. Then, we situate our study among mobile robotic applications of HRI.

Multi-Robot Active Mapping

The MRAM problem consists in acquiring an accurate representation of the environment by efficiently coordinating the actions of robots. Representation accuracy refers to the degree of closeness to the ground truth. Coordination of the robots arises from the teamwork involved during the task; it can be evaluated at several levels, *e.g.*, energy consumption, trajectory overlapping, *etc.* Solutions to MRAM are strategies to control robots in order to visit an unknown environment, make observations, and accumulate these observations on a map. The proposed solutions can be roughly classified into fixed, reactive, goal-based, and utility-based agent behaviors.

Within fixed approaches, the agent's behavior is fixed during execution and corresponds to simple navigation rules that are decided at the outset, *e.g.*, following walls, circular, or boustrophedon patterns [Baronov and Baillieul, 2007, Morlok and Gini, 2007]. Nevertheless, these agents are not aware of sources of information, hence they may be wandering for a long time in areas that are already known.

Within reactive approaches, the agent's behavior is hardwired to its perceptions. In the bio-inspired ant paradigm [Koenig and Liu, 2001, Svennebring and Koenig, 2004, Ferranti et al., 2007, Glad et al., 2010, Andries and Charpillet, 2013], agents perform reactive decision-making based on their local observation. They do not need to localize and are memoryless (the environment is the memory). The observation reveals the surrounding locations and the pheromone trails left by the other ants. Based on this observation, the ant considers the pheromones as an incentive for exploration and goes toward locations that have less pheromones. This can usually be achieved in constant time. The ants are robust to displacement and other

ants failures. Nevertheless, implementing ant robots requires to enable stigmergy via the trails left in the environment. A first setup requires on-board hardware to physically lay and sense trails; a second setup requires environment-embedded hardware to virtually lay and sense trails; and a third setup requires memory, localization, and communication to represent virtual trails on a map and communicate such map. The following goal-based agents work in the third setup and make more informed decisions.

In goal-based approaches, agents offer a good computation/performance trade-off, making them particularly suitable for deployment in real embedded systems [Koenig et al., 2001, Tovey and Koenig, 2003]. These agents focus on candidate locations, *i.e.*, locations that allow to gather novel data. In the widely spread frontier paradigm, robots are incrementally assigned to frontier candidates (separating the known and free space from the unknown space); this serializes the exploration task into navigation subgoals. Several distance-based methods are proposed to evaluate such frontiers: in MinDist by [Yamauchi, 1997], the robots go toward their closest frontier; in *MinPos* by [Bautin et al., 2012], the frontiers attract their closest robot; and in *MTSP* by [Faigl et al., 2012], the frontiers are clustered and the robots choose the first frontier along a minimum cost cycle that goes through a given cluster.

In utility-based approaches, an agent considers a more general value of candidate locations. Most of the time, this value combines the distance to the location and a heuristic estimate of the information available with a single observation from the location. For instance, in [Burgard et al., 2005], the robots choose the frontier of maximum utility $U = V - \beta C_w$, where V is the value of an observation from the frontier (discounted according to the frontier visibility from already selected frontiers), C_w is the shortest path cost to the frontier (weighted by the occupancy probability), and β tunes their relative importance; in [González-Banos and Latombe, 2002], robots choose the candidate location of maximum utility $U = Ae^{-\lambda C}$, where A is the maximal unknown area visible from the candidate location, C is the shortest path cost to the candidate location, and λ tunes their relative importance; in [Amigoni and Gallo, 2005] the robots select among the Pareto optimal frontiers that minimize the path cost C, and maximize the expected information gain I (visible frontiers) and overlap O (visible obstacles); and in [Basilico and Amigoni, 2009], a *multi criteria decision-making* method *MCDM* combines these values with the Choquet fuzzy integral.

Other utility-based approaches rely on *Information Theory* to quantify the information gain of a single observation from a candidate location by estimating their *Shannon* entropy value. In *IG-CL* [Stachniss and Burgard, 2003], each cell holds a probability distribution that describes its partial coverage by an obstacle. First, candidate locations are the cells with an entropy value higher than a predefined threshold. Then, the robot selects the cell that maximizes $U = \alpha\,\mathbb{E}[I] - C$, where α is a scalar weight, the first term is the expectation of the information gain I over all potential measurements from the candidate location, and C is the shortest path cost; and in [Moorehead et al., 2001] the same principle applies with multiple sources of information.

The field of Active Mapping is rich and other original approaches exist such as [Zlot et al., 2002] in which robots bid to buy frontiers during auctions; or [Macedo and Cardoso, 2004] where robots are driven by curiosity, surprise, and hunger.

In our present study, we consider goal-based agents with frontiers to reach and pedestrians to

interact with as navigation subgoals during exploration. We are looking for a parametric heuristic that evaluates/balances frontiers and interactions for exploiting pedestrian adaptive skills.

Human-Robot Interaction

HRI is defined as the study of humans, robots, and their mutual influences [Goodrich and Schultz, 2007]. To the best of our knowledge, the office-conversant robot Jijo-2 is the only HRI application of mobile robotic exploration. This robot exhibits socially embedded learning mechanisms [Asoh et al., 2001] by gathering information while conversing with other people. Thus, it realizes semi-supervised learning by incorporating local oracle heuristics while exploring.

We present an application in mobile robotics considering close interactions established by proximity or direct perception between pedestrians and robots. This type of interactions belongs to the *Intimate Interaction* class defined by Takeda into his HRI classification [Takeda et al., 1997]. Our study bridges together intimate HRI applications and MRAM goal-based algorithms.

5.3 Multi-agent system model

In this section, we describe the domain of our exploration task.

5.3.1 World's state

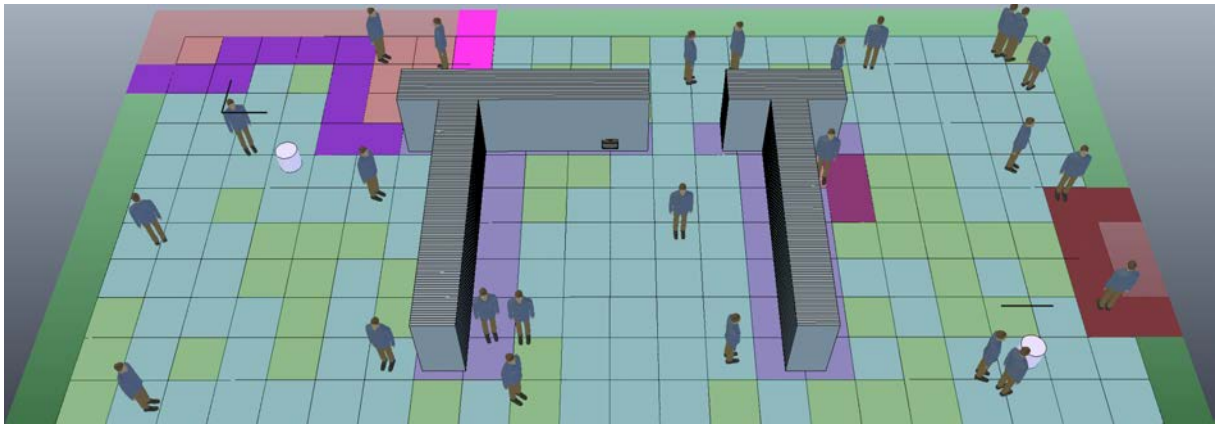


Figure 5.1: Multi-robot and pedestrian exploration system simulated using V-REP [Rohmer et al., 2013].

We propose to model a multi-agent system for exploration in crowds (Figure 5.1). In Equation (5.1), the environment is represented by a static occupancy map m, the set of robots and pedestrians are denoted by $\mathbf{R}$ and $\mathbf{H}$. The map m is described by a discrete grid of $l \times w$ cells; each cell is either free of any static obstacle (f) or occupied by a static obstacle ($\neg f_s$). It does not change with time as it describes the static occupancy of the environment. However, the configuration of the robots and pedestrians can change with time and represents the dynamic occupancy of the environment. Hence, at each time step, the state of the world is described by a tuple $s = (m, \mathbf{R}, \mathbf{H})$. At any time, either a robot $R_i \in \mathbf{R}$ or a pedestrian $H_j \in \mathbf{H}$ has a

configuration $(x, y, \theta)_i$ which describes its coordinates and yaw angle.

Formally,

$$
\begin{aligned}
&s = (m, \mathbf{R}, \mathbf{H}), \text{ is the world's state;} \\
&m : P = [1..l] \times [1..w] \rightarrow \{f\ (free), \neg f_s\ (static)\}, \text{ is the static occupancy map;} \\
&\mathbf{R} = \{R_1, .., R_n\}, \text{ is the set of robots; and} \\
&\mathbf{H} = \{H_1, .., H_m\}, \text{ is the set of pedestrians.}
\end{aligned} \tag{5.1}
$$

5.3.2 Robot's measurements and communications

A robot R_i locally observes the world's state s within a circular field of view denoted by $F_i(s)$, which is centered on the agent's position and of radius $range_v$. For instance, a robot's measurement $\mathbf{Z}_i$ corresponds to v_Z the occupancy status of local cells in its field of view; and R_Z and H_Z the configuration of robots and pedestrians in its field of view (Equation (5.2)). When a robot observes the occupancy grid, each cell in its local view is either occluded ($\neg k$) by static objects (walls) or mobile objects (robot, pedestrian), or visible (free of any object (f), occupied by a static object ($\neg f_s$) or a mobile object ($\neg f_m$)). We consider that measurements are perfect and deterministic.

Formally,

$$
\begin{aligned}
&Z_i = (v_z, R_Z, H_Z), \text{ is the measurement of } R_i; \\
&R_Z \subset \mathbf{R}, \text{ is the subset of robots in } F_i(s); \\
&H_Z \subset \mathbf{H}, \text{ is the subset of pedestrians in } F_i(s); \text{ and} \\
&v_Z : F_i(s) \rightarrow \{f\ (free), \neg f_s\ (static), \neg f_m\ (mobile), \neg k\ (occluded)\}, \text{ is the local view.}
\end{aligned} \tag{5.2}
$$

Additionally, let $Z_i^{0:t}$ be the set of measurements, namely the local t-time history, that R_i has experienced up to time t (Equation (5.3)). In our context, we consider that robots share their local histories by communicating when they meet within a radius of communication $range_c$. Hence, $\Theta_i^{0:t}$ is the global t-time history, which aggregates the latest local t-time histories that any robot R_i received from its teammates up to time t. In practice, robots exchange directly their global t-time histories.

Formally,

$$
\begin{aligned}
&Z_i^{0:t} = Z_i^{0:t-1} \cup \{Z_i^t\}, \text{ is the local history of measurements; and} \\
&\Theta_i^{0:t} = \bigcup_{j=1}^{|\mathbf{R}|} Z_j^{0:t_j}, \text{ is the global history of communicated measurements.}
\end{aligned} \tag{5.3}
$$

5.3.3 Robot's knowledge map

During exploration, the robots have to maintain a belief regarding the occupancy map of the environment. In our context, each robot represents such belief as a deterministic knowledge map k_i using its current global t-time history (Equation (5.4)). Every cell of this knowledge map has 4 possible states: the *unknown* state ($\neg k$) represents the unavailability of occupancy information (*e.g.* occluded cells) while the other states represents the availability of such information. More

precisely, the *free* state (f) represents the absence of any object; the *static occupancy* ($\neg f_s$) state represents the presence of static objects; and the *mobile occupancy* ($\neg f_m$) state represents the presence of mobile objects.

Formally,

$$k_i : P \to \{f\ (free), \neg f_s\ (static), \neg f_m\ (mobile), \neg k\ (unknown)\} \text{ is } R_i\text{'s knowledge map} \quad (5.4)$$

Knowledge map update

When R_i makes or receives a new measurement Z, the update is simply as follows: firstly, the robot consults the associated local view v; secondly, it browses all cells that are non occluded in v; thirdly, the occupancy status of each such cell is copied onto the corresponding cell in k. To summarize, as static obstacles cannot be moved by the agents in our context, every measurement independently updates a cell's state on the knowledge map as given in Figure 5.2.

Figure 5.2: Cell occupancy status transition diagram.

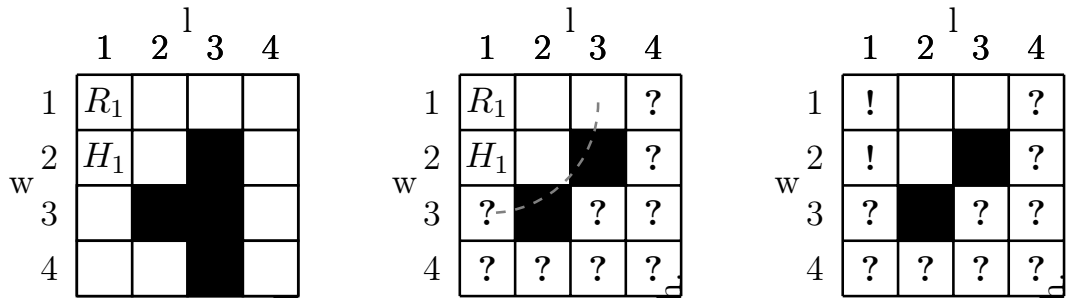
Knowledge goal predicate

The robots know when the exploration mission terminates. The mission ends when there is no reachable robot's configuration that allows to view currently unknown areas. Firstly, each robot identifies reachable cells from its current position based on its knowledge map; for this purpose, it considers that cells with mobile occupancy are traversable [15]. Secondly, it determines candidate locations among reachable cells; these candidate locations allow to view currently unknown areas. If there is no candidate location the mission is over. Each robot independently computes and checks this goal predicate at each time step.

To summarize, in Figure 5.3, we illustrate a world's state, where the environment is drawn as an occupancy grid (black means occupied by a static object and white means free of any object), a robot is drawn as R_1 and a pedestrian is drawn as H_1 in Figure 5.3a. The robot is located on cell R_1 at $(1, 1)$ and the pedestrian on cell H_1 at $(1, 2)$. The maximum field of view of R_1 is within the dashed arc in Figure 5.3b. Z_1^1 consists of 7 cells: 3 are free, 2 are static, and 2 are mobile. The local history $Z_1^{0:1}$ provides the knowledge map in Figure 5.3c.

We have modeled a multi-agent system for exploration in crowded environments. The environment is represented by a discrete occupancy grid, agents are characterized by their identifier and

[15]This is necessary to ensure that exploration does not terminate prematurely. Indeed, if mobile objects currently block a dead-end, the exploration must terminate only after the dead-end is viewed.



(a) State $s = (m, \{R_1\}, \{H_1\})$ **(b)** History $Z_1^{0:1} = \{Z_1^1\}$ **(c)** Knowledge k

Figure 5.3: World's state, robot's measurement and robot's knowledge at time $t=1$. There is one robot, one pedestrian and 4 blocked cells in the current world's state. The robot only sees the information in its field of view as represented by the gray dashed arc, its current knowledge map is limited to this first measurement.

their configuration, and measurements are made by casting rays within the viewing range of a robot. Our study is based on this representation of a multi-agent system. In the next section, we present the frontier/interaction exploration approach.

5.4 Hybrid Frontier-Interactive exploration

First, let us consider the MRAM problem defined as a target allocation problem of robots in an unknown environment [Bautin et al., 2012, Burgard et al., 2005, Faigl et al., 2012]. A solution to the MRAM problem defines a strategy to explore the environment by assigning robots from $\mathbf{R}$ to targets from $\mathbf{T}$ at each time step. To achieve this, firstly, we can identify the targets $\mathbf{T}$ that the robots must explore in the environment; secondly, we can define a cost matrix $\mathbf{C_{RT}}$ that evaluates the cost of a given robot-target assignment; and thirdly, we must find an assignment matrix $\mathbf{A_{RT}}$ that selects a task for each robot by optimizing the costs (*cf.* Figure 5.4).

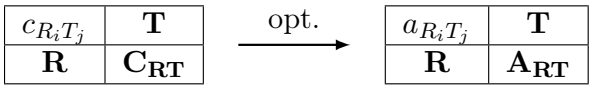


Figure 5.4: Multi-agent exploration as a task allocation problem.

5.4.1 Selecting frontier and pedestrian targets

We show how different sets of targets define the classical frontier-based exploration, our new interactive approach, and the hybrid approach (frontier/interaction).

Frontier paradigm A frontier is a free cell in the known area that is contiguous to the unexplored area of the environment [Yamauchi, 1997]. The classical frontier-based exploration is defined by choosing the targets from the set of frontiers $\mathbf{F}$.

Formally,

$$c_{R_iF_j} \text{ is the cost for } R_i \text{ to reach } F_j$$
$$a_{R_iF_j} \leftarrow \begin{cases} 1 & \text{if } R_i \text{ must go to } F_j \\ 0 & \text{otherwise.} \end{cases}$$

In our context of populated environments, we must extend the definition of a frontier to account for the case where a pedestrian stands on a cell next to the unexplored area. Indeed, if we do not account for such cases, exploration may terminate earlier as those frontiers remain undetected. Hence, in our case, a frontier cell is either free or occupied by a mobile object and is contiguous to the unknown part of the environment. Moreover, the frontier paradigm may fail when the chosen frontier becomes unreachable as its path is blocked by pedestrians.

Interactive paradigm We introduce an interactive approach that explicitly considers the pedestrians. This is done for establishing human-robot interactions (opening a door, guiding though a crowd, *etc.*). Targets are now chosen from the set of pedestrians **H**.

Formally,

$$c_{R_iH_j} \text{ is the cost for } R_i \text{ to interact with } H_j$$
$$a_{R_iH_j} \leftarrow \begin{cases} 1 & \text{if } R_i \text{ must interact with } H_j \\ 0 & \text{otherwise.} \end{cases}$$

A purely interactive approach can be inefficient in sparsely populated environments. Indeed, without any perception of human presence, the robots adopt a wait-and-see policy and pause the exploration. Hence, we propose to hybridize the frontier paradigm and the interactive paradigm.

Hybrid paradigm We hybridize the frontier paradigm and the interactive paradigm to enable interactions with pedestrians and to reach frontiers as well. Thus, we combine the two target sets (frontiers and pedestrians) to define a new set **G**.

Formally,

$$c_{R_iG_j} \text{ is the cost for } R_i\text{'s assignment to } G_j$$
$$a_{R_iG_j} \leftarrow \begin{cases} 1 & \text{if } R_i \text{ is assigned to } G_j \\ 0 & \text{otherwise.} \end{cases}$$

This approach requires to smartly adjust interaction and frontier assignments to overcome the two aforementioned issues (wait-and-see policy and the congested frontier). We defined the targets that are considered in our study. Now, we will define the cost of each pair of robot and target assignment.

5.4.2 Defining frontier and interaction costs

In this study, robots can interact only by following pedestrians. The optimization criterion is to explore a possibly populated environment with minimum distance and time. Therefore, we define costs using distances and weighted penalties, see Figure 5.5. The weight σ balances interaction and frontier penalties. First we detail distances, then we introduce penalties and explain the different parameters used in the cost formula.

(a) Distances **(b)** Penalties

Figure 5.5: Distances and penalties.

Distance First, we incorporate distances between robots and targets as immediate costs (Figure 5.5a). Thus, we initialise $\mathbf{D_{RG}}$ with normalized robot-frontier and robot-pedestrian distances ($\mathbf{D_{RF}}$, $\mathbf{D_{RH}}$).

$$\mathbf{D_{RG}} = (\mathbf{D_{RF}} \mid \mathbf{D_{RH}})$$

$$\mathbf{D_{RX}} = \begin{bmatrix} d_{R_1X_1} & \cdots & d_{R_1X_{|\mathbf{X}|}} \\ \vdots & \ddots & \vdots \\ d_{R_nX_1} & \cdots & d_{R_nX_{|\mathbf{X}|}} \end{bmatrix}$$

Simple distance heuristics as Euclidean or Manhattan metrics suffer from the two following drawbacks:

If a robot navigates toward a frontier but a crowd hinders its navigation: the robot cannot adapt its exploration depending on navigation feasibility as remote but reachable frontiers are not reevaluated as good options. The distance is prohibitive and the next target is always chosen between the last frontier and close pedestrians. A solution consists in computing the shortest path length on static navigation maps (pedestrian as static obstacles), which is set to a high value when a target is momentarily unreachable.

If a robot follows a pedestrian walking nearby but the pedestrian suddenly comes to a halt: the robot cannot decide whether to maintain or stop the current interaction depending on the pedestrian activity. Due to distances, the robot will resume exploration only if the pedestrian moves again. A solution is to make an *a priori* evaluation of an interaction and then update this evaluation while the interaction takes place. Our solution consists in introducing penalties.

Penalty We tackle these two drawbacks with a heuristic that associates penalties to each robot-frontier or robot-pedestrian pair (Figure 5.5b). A penalty $p_{R_iX_j}$ is defined as the sum of a time penalty and an orientation penalty. The time penalty $t_{R_iX_j}$ is the time elapsed since a frontier discovery or a pedestrian remains idle. The orientation penalty $o_{R_iX_j}$ is the smallest unsigned angle between the orientation of a robot and the orientation of a frontier/pedestrian. More precisely, the orientation penalty $o_{X_iY_j}$ evaluates the necessary reorientation of entity X_i to follow entity Y_j in two steps, see (Figure 5.6). The first one is for rotating entity X_i from an initial starting yaw angle x_a toward the target position Y_j and is termed $|\widehat{x_aX_iY_j}|$. The second one is for aligning with the target entity Y_j's yaw angle y_a which is noted $|\pi - \widehat{X_iY_jy_a}|$. As a frontier should be approached from the already explored space, a frontier's orientation is set toward the unknown.

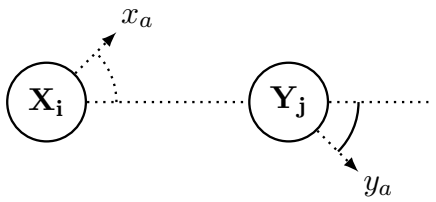


Figure 5.6: Orientation penalty with x_a the starting yaw angle of entity X_i, and the ending yaw angle y_a of entity Y_j.

Thus, we define $\mathbf{P_{RG}}$ with normalized robot-frontier and robot-pedestrian penalties ($\mathbf{P_{RF}}$, $\mathbf{P_{RH}}$).

$$\mathbf{P_{RG}} = (\sigma \cdot \mathbf{P_{RF}} \mid (1-\sigma) \cdot \mathbf{P_{RH}}),\ \sigma \in [0,1]$$

$$\mathbf{P_{RX}} = \begin{bmatrix} p_{R_1X_1} & \cdots & p_{R_1X_{|\mathbf{X}|}} \\ \vdots & \ddots & \vdots \\ p_{R_nX_1} & \cdots & p_{R_nX_{|\mathbf{X}|}} \end{bmatrix}$$

$$p_{R_iX_j} = t_{R_iX_j} + o_{R_iX_j}$$

$$o_{X_iY_j} = |\widehat{x_aX_iY_j}| + |\pi - \widehat{X_iY_jy_a}|$$

The parameter σ sets more or less weight on the frontier penalties or on the interaction penalties. When this parameter is high, it increases the frontier penalties and decreases the interaction penalties. This results in favoring interactions over frontiers.

Distance and penalty The hybrid cost matrix $\mathbf{C_{RG}}$ which incorporates distances $\mathbf{D_{RG}}$ and penalties $\mathbf{P_{RG}}$ is given in Equation (5.5).

$$\mathbf{C_{RG}} = \alpha \cdot \mathbf{D_{RG}} + (1-\alpha) \cdot \mathbf{P_{RG}},\ \alpha \in [0,1] \tag{5.5}$$

The parameter α modulates the distance cost and the penalty heuristic. When α is high (resp. low), the importance of the penalties is reduced (resp. increased). Distances and penalties are counterbalanced with α, while σ sets more or less penalties on frontiers or on interactions.

We present the influence of α and σ on the cost formula in (Figure 5.7). The values range from 0 to 1 for each parameter and the formula written on each side are obtained when one parameter is set to its extreme value. We heuristically defined the cost of each pair of robot and target. Now, we report the strategies that are used for optimizing the assignments.

5.4.3 Optimizing the robot-target assignments

We use two different strategies for robot-target assignments. The first strategy is a local *greedy* strategy where any robot R_i independently chooses its minimum cost target G^* among its current targets (visible pedestrians $\mathbf{H}$ and known frontiers $\mathbf{F}$). This is similar to the strategy used with distances only in [Yamauchi, 1997]. It can be summarized as follows:

1. Initialize $\mathbf{G} = \{\mathbf{H} \cup \mathbf{F}\}$;

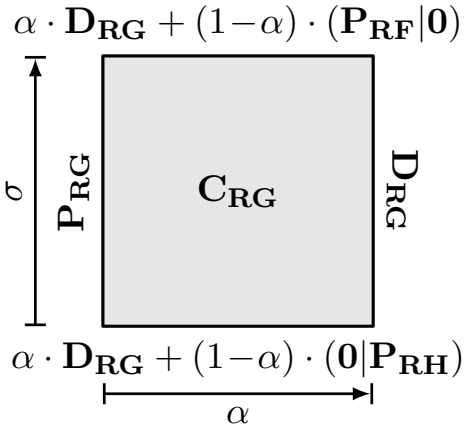


Figure 5.7: Mixed cost matrix $\mathbf{C_{RG}}$ parameterized by α and σ.

2. Find $G^* = \arg\min_{G_j \in \mathbf{G}} c_{R_i G_j}$.

The second strategy is a group *greedy* strategy, where any robot R_i additionally considers its visible teammates (R_V). Firstly, the robot finds the pair of robot and target with the lowest cost (R^*, G^*). Secondly, if the current robot is in this pair ($R^* = R_i$) then its assignment is found; otherwise the assigned target is removed from the set of targets and the assigned robot is removed from the set of visible teammates. Thirdly, these steps repeat until the robot finds its assignment or the set of targets is empty. It is summarized below:

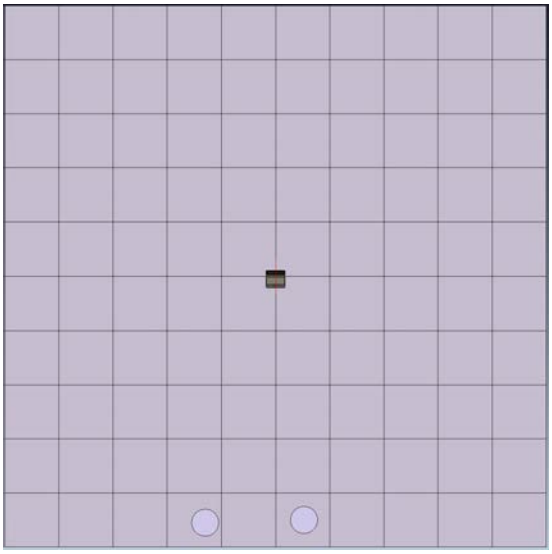
1. Initialize $\mathbf{G} = \{\mathbf{H} \cup \mathbf{F}\}$ and $R_V = R_Z$;
2. Find $(R^*, G^*) = \arg\min_{R_j \in R_V, G_j \in \mathbf{G}} c_{R_j G_j}$;
3. If $(R^* = R_i)$ then return G^*; else $R_Z = R_Z \setminus R^*$ and $\mathbf{G} = \mathbf{G} \setminus G^*$;
4. Go to step 2 unless $\mathbf{G}$ is empty.

We introduced a hybrid approach, defined a parametric cost based on a penalty heuristic, and reported two assignment strategies. Now, we evaluate the exploration performance of this heuristic for these two assignment strategies, assuming different values of α and σ.

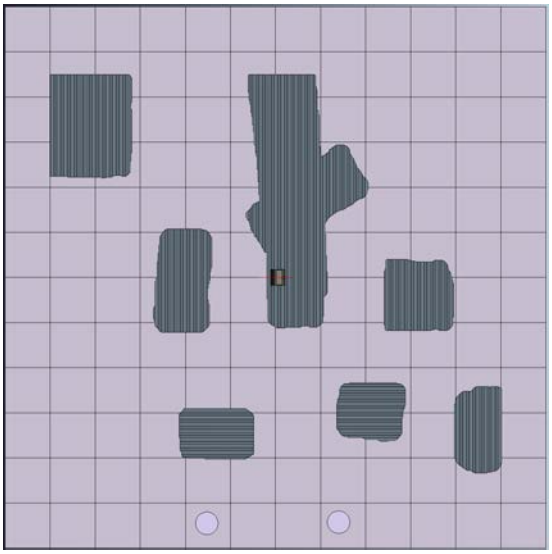
5.5 Experiments

In our experiments, we use the V-REP robot simulator [Rohmer et al., 2013]. The robots share their exploration map, so the frontiers that are discovered are known by every robot. Contiguous frontier cells are grouped together into a frontier area. Inside a frontier area, the targeted cell minimizes the sum of distances to the other cells. Assignments are locally computed by each robot. To optimize its assignment, every robot takes into account the entire set of frontiers known until now, but only the robots and pedestrians perceived locally. Planning is done using a potential field propagated on the grid; any optimal off-line search algorithm reported in Chapter 4 can be used. In particular, we run Uniform-Cost Search (UCS) forward from each robot.

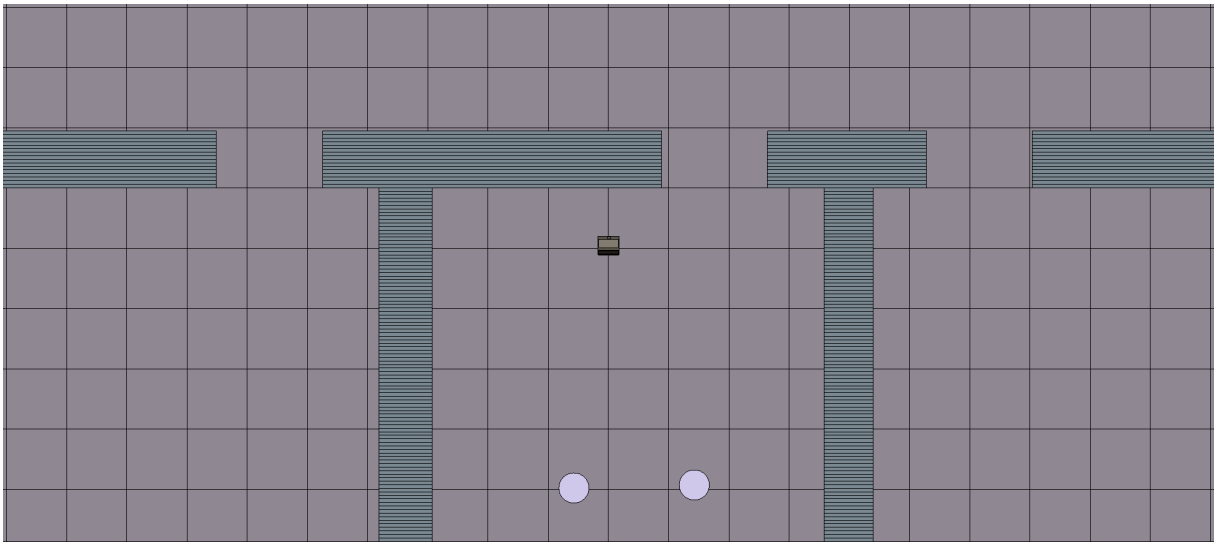
Protocol More precisely, the parameters are as follows:



(a) Empty (10×10 cells, $25m^2$)



(b) Cave (12×12 cells, $36m^2$)



(c) Structured (22×11 cells, $60.5m^2$)

Figure 5.8: Environment maps.

- Map: Three environments are considered: the first contains no obstacle (*empty*), the second has unstructured obstacles (*unstructured*), and the third contains a corridor and three rooms (*structured*). Maps are discretized with $0.5m$ square cells as shown in Figure 5.8.
- Pedestrian: Every pedestrian moves at a speed of $0.56m/s$ in a straight line and avoids obstacles by stopping and rotating as implemented by the *Walking Bill* model provided by V-REP. Environments are populated with 0 or 30% of pedestrians (up to 1 pedestrian/m^2).
- Robot: Every robot moves at a speed of $0.5m/s$ from one cell to any of its neighbors in the cardinal and ordinal directions. The viewing range of each robot is set to $2m$ and the communication range is set to $100m$. Consequently, the knowledge maps are exchanged at any time on all maps. Two explorers are used for each experiment and start from fixed positions, these starting positions are illustrated as circles in each environment.
- Modulators: α and σ are discretized from 0 to 1 with a step of 0.25, so we have 25 pairs of parameters.
- Metrics: Each scenario is evaluated with classical Active Mapping (AM) metrics: average coverage, distance and time by each robot. In addition, we use a common metric in HRI, called the *Robotic Attention Demand* (RAD), which measures the autonomy of a robot during its task [Olsen and Goodrich, 2003, Steinfeld et al., 2006]. Here we consider the number of interactions initiated during exploration.

Results First, let us consider environments without pedestrians. We study the influence of α by fixing σ to 1. This is legitimate, since no pedestrian implies no interaction penalty. The weight α allows to adjust the distance and frontiers penalty; distances are overweighted when α increases. The performances averaged over 10 runs are plotted in Figure 5.9 for local *greedy*, and in Figure 5.10 for group *greedy*.

In Figure 5.9, with local *greedy*, we distinguish two steps regarding the *empty* and *structured* maps: firstly, distance and time increase, and secondly, they both decrease until $\alpha = 1$. In the *unstructured* map, they both monotonically decrease when increasing α. In Figure 5.10, with group *greedy*, distance and time decrease when increasing α for all maps. In these cases, when α is high, penalties fade and robots do less round trips between remote frontiers in the scene. Thus, frontier penalties are not useful in these nonpopulated environments.

Now we consider the presence of pedestrians; the environments are populated at 30%. Figures 5.11 and 5.12 respectively report the average performances of local *greedy* and group *greedy*, for 10 runs of each pair of weights (α,σ). As a reminder, when σ increases, the penalty of the interactions is reduced, hence favoring interactions over frontiers.

In Figure 5.11, with local *greedy*, full coverage with lowest distance and time are at $(\alpha, \sigma) = (0, 0)$ in the *empty* map. Only interaction penalties are counted and frontiers are favored over interactions. Consequently, no interaction was initiated (RAD) but an average of 28 frontiers were assigned. In the *unstructured* map, the best average performance is at $(0.75, 0)$. Distances are overweighted compared to penalties and only interactions are penalized. Nevertheless, an average of 18 interactions were initiated against 26 frontiers assignments. In the *structured* map, the best performances and lowest standard deviations are at $(0.5, 0)$. Distances and penalties are equally weighted and only interactions are penalized. An average of 31 frontier assignments

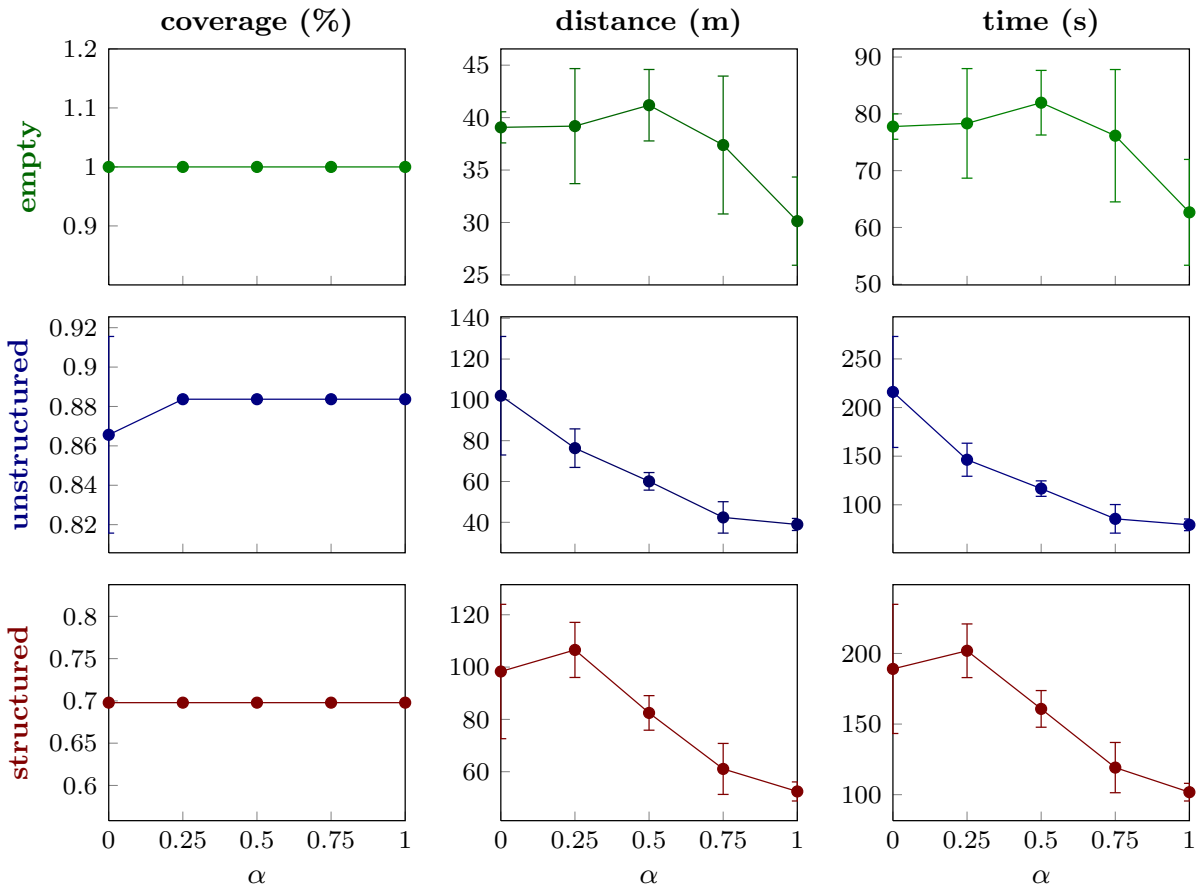


Figure 5.9: Performance metrics of *LocalGreedy* in the empty, unstructured and structured environments, without pedestrians.

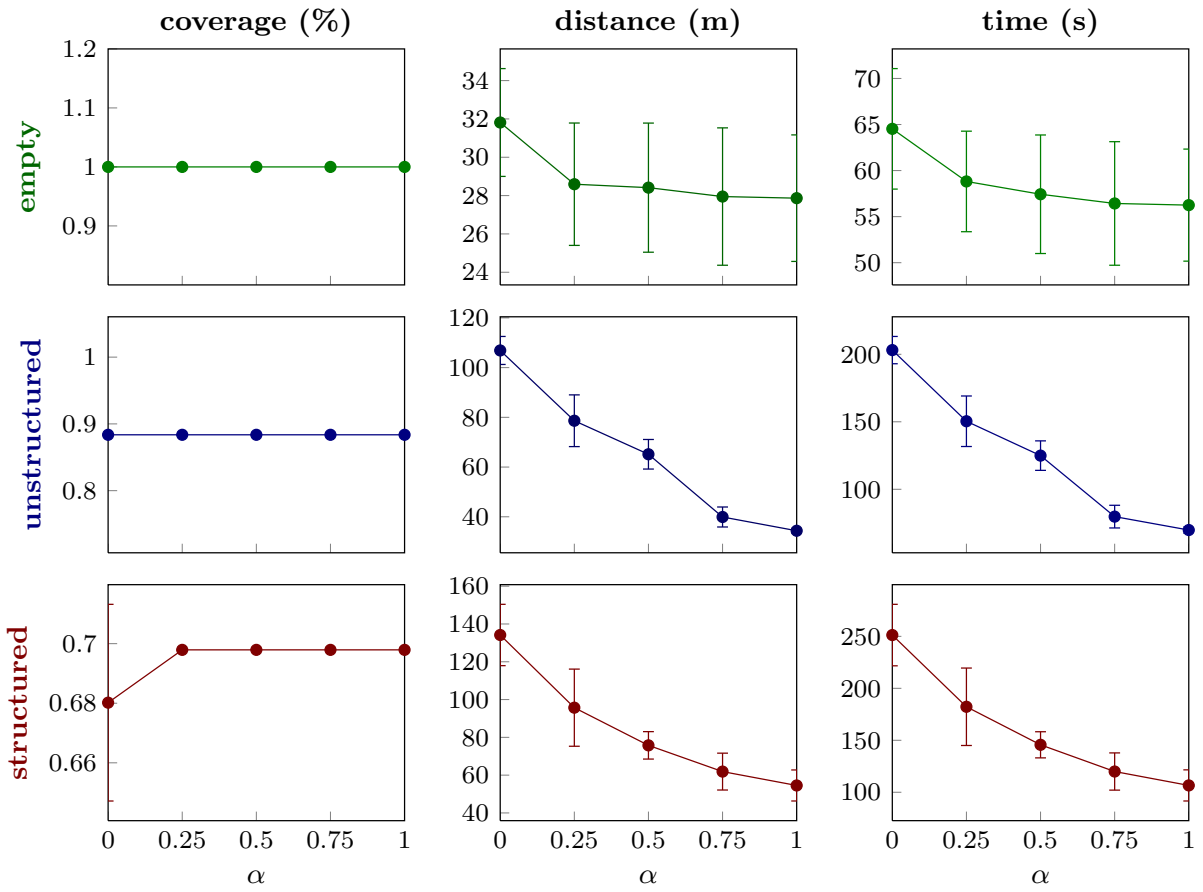


Figure 5.10: Performance metrics of *GroupGreedy* in the empty, unstructured and structured environments, without pedestrians.

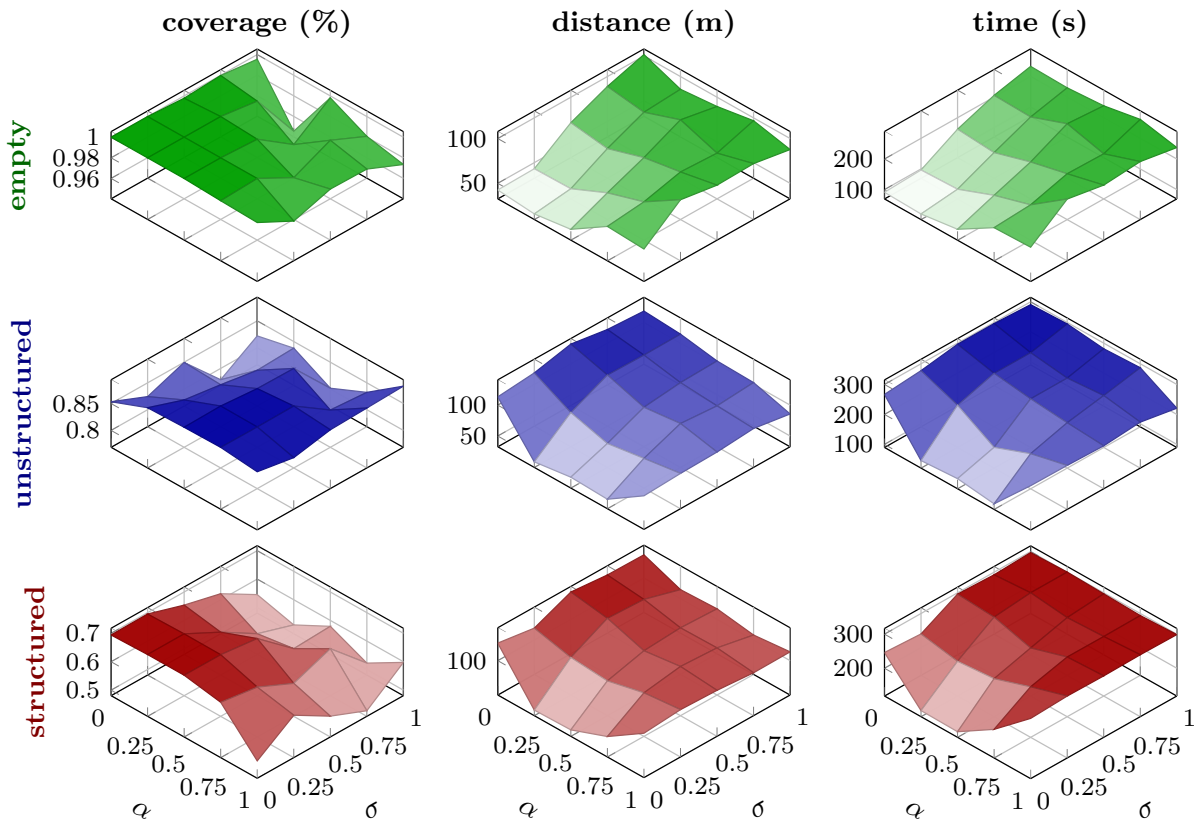


Figure 5.11: Performance metrics of *LocalGreedy* in the empty, unstructured and structured environments, with pedestrians.

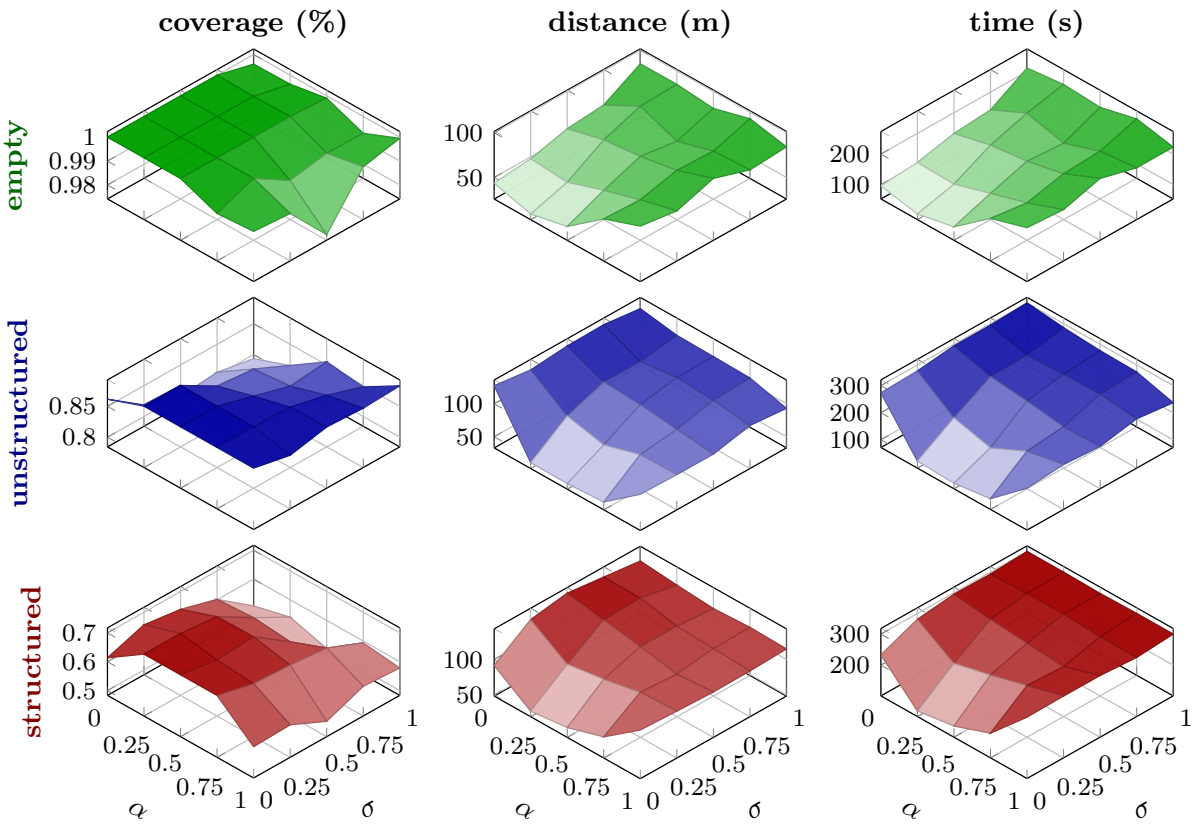


Figure 5.12: Performance metrics of *GroupGreedy* in the empty, unstructured and structured environments, with pedestrians.

and 18 interaction assignments took place. In general, both frontier and interaction targets are evaluated with distances, but only the interaction targets are penalized.

In Figure 5.12, with group *greedy*, the best performance is at $(0.25, 0)$ for the *empty* map. Penalties are overweighted compared to distances and only interactions are penalized. An average of 16 frontier assignments and 7 interaction assignments is noticed. In the *unstructured* map, the maximum coverage with minimum traveled distance and time is at $(0.5, 0)$. The average number of frontiers assigned is 8 times the number of interactions (only 4). In the last map, the best average performance is at $(0.25, 0)$ for a frontier/interaction ratio of $29/4$. In general, the distance does not suffice for choosing the best targets when there are pedestrians. Instead, an equilibrium with penalties is necessary to obtain the best performance ($\alpha \neq 1$). Here with ($\sigma = 0$), the frontiers were chosen considering only distances but interactions were highly penalized. Thus, in these experiments, our penalty heuristic is not yet able to promote interactions.

5.6 Conclusion

In this study, we have defined the interactive paradigm of exploration by targeting the pedestrians perceived by the robots. Interactive exploration paves the way for exploiting pedestrian navigation heuristics and mapping pedestrian flows in crowded environments. The hybrid approach, based on both the frontier and interactive paradigms, aims at bringing out the best of both approaches. For this purpose, we designed a parametric heuristic to equilibrate frontier and interaction targets assignments.

This heuristic incorporated planned distances to the targets and penalties regarding their idle state and orientation. We have shown in simulation that incorporating this heuristic into exploration mainly penalizes the interactions. To enable efficient exploration in dynamic environments, it is paramount to discover the cases where interactions are beneficial. There are several ways to continue this work, firstly, better modeling of the pedestrians is necessary: the *Social Force Model* [Helbing and Molnar, 1995] proposes local rules that reproduce the behavior of pedestrian crowds at a global level; secondly, better features could be extracted for evaluating interactions: pedestrian's intention, and relative position to the frontiers; thirdly, on-line tuning is necessary as in general these weights will depend on the environment: *Embodied Evolution* [Watson et al., 2002] may be used for optimizing and exchanging parameters during execution.

This work opens up prospects for exploiting pedestrian adaptiveness in robotic exploration of populated environments. It has been published as a workshop paper at an international conference in Artificial Intelligence in [Kaldé et al., 2014b], and as a conference paper at a national conference on *Multi-Agent System* in [Kaldé et al., 2014a], and as an international journal article in [Kaldé et al., 2015].

Chapter 6

Distributed Roadmap for Navigating in Ambient Environments

Contents

6.1 Introduction

In the previous chapter, we focused on decentralized exploration with mobile robots in order to cooperatively build an occupancy map of the environment. We saw that in crowded environments, the reachability of the different areas changes over time due to the pedestrians' flows. Hence, in such environments, mobile robots need to frequently replan their paths as the navigability changes. Indeed, the planned paths quickly become obsolete as they are blocked by pedestrians, or suboptimal as better paths are freed. In general, maintaining the navigable paths over time will allow mobile robots to plan their moves more efficiently in the environment. Here, we propose to build a navigability map by computing Voronoi Diagrams (VDs) in order to extract navigable paths that are equidistant to the obstacles (Clearest Path Obstacle Avoidance (CPOA)). As an intermediate step toward tasking mobile robots to do so, we will rely on ubiquitous computation with environment-embedded processing and sensing units.

This chapter addresses the problem of computing a VD in a distributed fashion without any synchronization mechanism. To our knowledge, no previous work asynchronously solves this problem. We investigate a simple case of asynchronism and address this challenge in a decentralized fashion on a grid of communicating cells with a *Von Neumann* neighborhood. We

describe algorithms for extracting single-site, area and pseudo-line VDs. Then, we apply these algorithms on maps with simple polygonal obstacles, in order to extract safe roadmaps.

Ambient intelligence [Boekhorst, 2002] results from the interaction of numerous devices. Such interactions take place in sensor networks on which embedded processing units run and cooperate with each other. Self-organizing approaches can simplify the design of such systems when it comes to scalability and robustness. This is the objective of spatial computing [Maignan and Gruau, 2011], where data are distributed in a network of processing units and the computation can be done in various synchronization modes. We will take advantage of the powerful and general Cellular Automaton (CA) model to represent our distributed network.

We address the problem of computing a VD on this type of sensor network. A VD can be defined as the set of regions associated to sites. Each region represents the set of points that are closer to one site than any other site. Every node in the network has a local perception of the sites dataset and communicates with its neighboring nodes to compute a part of the final diagram.

Several studies already deal with distributed computing of VDs, but most of them address this problem with an embedded synchronization, *e.g.*, parallel or sequential computing in predefined orders of the units. These execution orders imply full control over the nodes. This assumption has several drawbacks, such as scalability and robustness. If a single unit fails to communicate, the overall process can suffer from this situation. Moreover, synchronizing a high number of units may be really difficult. For the purpose of scalability and robustness, our approach considers no synchronization for computing a VD.

Our study focuses on an *asynchronous mode* in which every unit is selected according to a uniform distribution. We also define a *synchronous* mode in which all units compute simultaneously. We aim to design simple mechanisms that exhibit the same behavior whatever the synchronization.

Our contribution to compute VDs in asynchronous mode and using 1-norm is inspired by sequential image processing as it requires to build a convergent Labeled Distance Transform. Firstly, in Section 6.2, we provide some background regarding the sensor network formalization, the synchronization modes, and the VD. Secondly, in Section 6.3, we report some related work regarding parallel or sequential computation of *Voronoi* tessellations from the fields of CA and *Image Processing*. Thirdly, in Section 6.4, we present our approach for asynchronously computing VDs. Then, in Section 6.5, we extend our study to new types of sites and compute roadmaps in simulation. Finally, in Section 6.6, we conclude with a short table that qualitatively compares several methods.

6.2 Background

In this study, we rely on a two-dimensional CA to model a grid of processing and communicating units, as reported in Definition 6.1.

Definition 6.1 (2D Cellular Automaton (CA)). *A typical* 2D Cellular Automaton *is defined as a tuple* $\mathbb{F} = \langle A, Q, u, f \rangle$*. A, is a matrix of cells (dimension $w \times h$). Q, is the set of cell states. $u : A \to A^k$, is a cellular neighborhood function (*e.g.*, u_1 denotes the* Von Neumann *neighborhood: itself, North, East, South and West). We consider a cell, $c \in A$, and its state, $c^t \in Q$, at time,*

t. The automaton evolves in discrete time as the cells, c, apply a local rule, $f : Q^k \to Q$*, which considers the current neighbors state,* u^t*, to compute their next state,* $c^{t+1} \leftarrow f(u^t(c))$.

Synchronizing a CA consists in choosing which cells update their state at each time step. The first type of synchronization we consider is termed synchronous: all cells simultaneously compute the next state of the automaton. The second one is termed fully asynchronous: one randomly chosen cell contributes to the next automaton state.

We also rely on a discrete Generalized VD as defined in [Okabe et al., 2009]; this definition is adapted to our CA in Definition 6.2.

Definition 6.2 (Voronoi Diagram (VD)). *The Voronoi Diagram is a tessellation of the matrix of cells, A, in* Voronoi *regions associated to* Voronoi *sites. Each* Voronoi *region,* $VR(S_i)$*, defines the set of cells which are closer to the site,* S_i*, than any other site,* S_j*. The distance used can be any metric,* e.g.*,* Minkowski *distances. A site is defined as a subset of the lattice,* e.g.*, contiguous cells.*

Computing a VD consists in identifying cells that belong to a same region. In the following, we study the computation of a VD on a grid of processing and communicating units. Moreover, this grid will be subject to different types of synchronization.

6.3 Related work

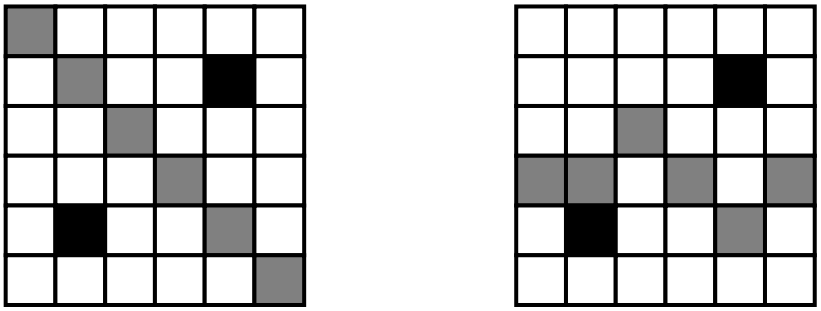
In 1964-67, Blum introduced a new thinning shape descriptor, the Medial Axis Transform (MAT) [Blum, 1967]. The first MAT definition referred to as the "grass-fire metaphor" is time-based: fire fronts propagate from fire sources (shape contour) and travel at constant velocity across the grass field (into the shape). Two fire fronts annihilate each other when they meet on the medial axis. Distance-based definitions were also provided: if we consider a distance field generated from the shape contour, the MAT is defined alternatively as the locus of derivative discontinuities, centers of maximal disks, or minimum distances to many contour points.

Cellular Automata

In the field of *Cellular Automata*, the first spatio-temporal definition of MAT (grass-fire metaphor) naturally translates to synchronous evolution rules. A 1-norm and an ∞-norm VD were synchronously computed in [Adamatzky, 1996] and then completed in [Maniatty and Szymanski, 1997] by extracting the discrete bisectors between cellular sites. A 2-norm VD was investigated in [Sudha et al., 1999].

These CA models were readily usable as simplified simulators of natural computing. A retrospective of VD variations computed using intrinsically asynchronous chemical processors and modeled using synchronous CA rules is provided in [de Lacy Costello et al., 2012, de Lacy Costello, 2014]. Even if these models do not always reflect ground truth, they are meaningful to explain the concepts involved. Moreover, several applications of VD, from path planning [Tzionas et al., 1997, Adamatzky and de Lacy Costello, 2003, Adamatzky and de Lacy Costello, 2002] and classification [Tzionas et al., 1994], to shape recognition [Adamatzky et al., 2002] are reported in the CA literature.

However, the time-based definition has some drawbacks. Results obtained when simulating the grassfire metaphor using 1-norm for the synchronous and asynchronous cases are different after convergence, *cf.*, Figure 6.1. Thus, when it comes to asynchronism, the metaphor can be expressed alternatively as a fire propagation altered by a *random wind*.



(a) Synchronous mode **(b)** Asynchronous mode

Figure 6.1: Synchronous and asynchronous fire propagation with two sites (black), two burnt regions (white), one discrete bisector (gray).

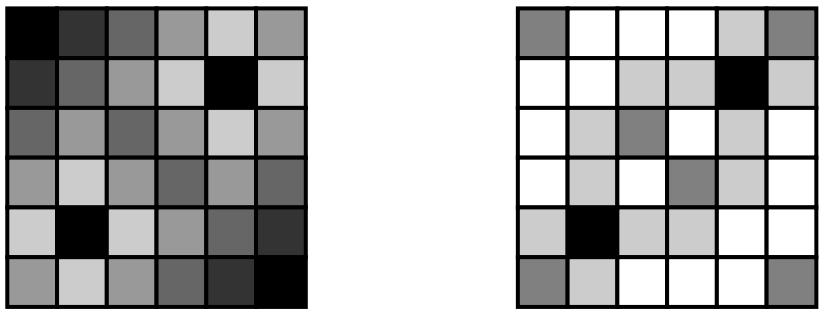
Image processing

In the field of *Image Processing*, distance-based definitions were investigated to extract shape skeletons (MAT). In [Rosenfeld and Pfaltz, 1966], a 1-norm distance transform of the foreground objects in an image was computed using sequential local operations. They were applied on each pixel in forward and backward raster sequences. The distance skeleton was extracted as the set of local maxima pixels. A 2-norm distance transform was used to extract planar skeletons as the set of maximal disks centers in [Danielsson, 1980].

The Labeled Distance Transform (LDT) proposed in [Fischler and Barrett, 1980], provides semantic information about the contour (coordinates, directions, features) over the distance transform. Thus, the skeleton is the set of pixels with locally varying semantics. This method generalizes extraction and, as such, any distance transform can be considered. In [Arcelli and Sanniti di Baja, 1986], a VD was extracted as a pruned 1-norm exoskeleton (skeleton of the background). More recent studies take advantage of nowadays available Graphics Processing Units (GPUs) hardware, to compute VDs using parallel local rules [Hoff III et al., 1999].

Concerning the distance-based definitions of the MAT, a 1-norm integer gradient can be asynchronously computed using a convergent rule described in [Barraquand et al., 1992]. In Figure 6.2a, darker shades represent higher values of this gradient. However, the associated u_1 gradient patterns colored in Figure 6.2b, cannot determine a frontier between the two *Voronoi* regions. This *wave fusion* phenomenon, *i.e.*, two waves collide and then combine, leaves the gradient unable to discern a full bisector. A solution involving both the *Von Neumann* and *Moore* patterns is reported in [Arcelli and Sanniti di Baja, 1985].

Our work tackles the *random wind* and the *wave fusion* phenomena, using only 1-norm and u_1 neighborhood, to asynchronously compute a VD. Therefore, we build a site semantic layer on top



(a) Integer gradient **(b)** u_1 patterns

Figure 6.2: Wave expansion: integer gradient and local patterns with two sites (black), no discernible bisector.

of a gradient layer as the LDT. This information gradient is not affected by the asynchronism and provides a correct VD upon convergence.

6.4 Asynchronous computation of a discrete Voronoi Diagram

First, we present the rules for computing an information gradient on the CA. It provides correct *Voronoi* regions and can be computed using the synchronous or asynchronous modes. In a second part, we discuss bisector extraction between two *Voronoi* regions based on semantic information from the information gradient. For the sake of simplicity, we deal with a single cell site. Multi-cell sites are addressed in the next section.

6.4.1 Information gradient

The information gradient is a combination of a distance transform and a semantic overlay. The distance map provides, for each cell c of the automaton, the 1-norm distance to the closest site in S. The semantic layer provides the identifier of the closest site for any cell, hence, it determines the *Voronoi* region.

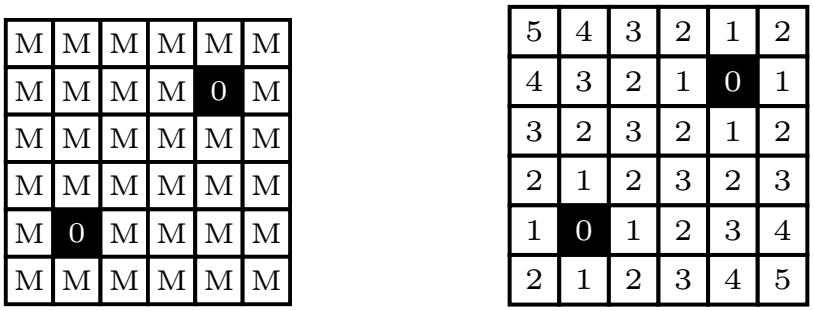
Distance layer First of all, we consider two kinds of cells at any time, the *sites* and the *non-sites*. The objective is to compute the distance for each non-site cell to its closest site. The distance transform, d_1, is initialized to 0 for the sites and to a random positive value M for the non-sites. Each cell participates in computing the 1-norm integer gradient using the following rule G:

$$G(c) = c.d_1 = \begin{cases} 0 & \text{if } site(c) \\ m+1 & \text{otherwise.} \end{cases}$$
$$m = \min_{k \in u_1(c) \setminus \{c\}} k.d_1$$

When a site is selected at time t, it never updates its distance to the closest site. Otherwise, for a non-site, its closest site distance d_1 is updated by taking the minimum d_1+1 value among its u_1-neighbors.

If we consider static sites, as long as we update every cell often enough, the distance transform d_1 will converge at a time T. In other words, for each pair of cell and closest site (c_{ij}, S_{kl}) with respective coordinates (i, j) and (k, l) in A, we have $\lim_{t \to T} c_{ij}.d_1 = d(c_{ij}, S_{kl}) = |i - k| + |j - l|$.

If we consider dynamic sites, *e.g.*, a site moves on the lattice, some cells will change their site-state from site to non-site and inversely, and compute d_1 accordingly. The distance transform will eventually converge depending on the speed of change and number of state transitions relative to T, the correction time of the distance transform.



(a) Initialization **(b)** d_1 async/sync

Figure 6.3: Integer gradient using 1-norm: initialization and propagation with sites (black) and non-sites (white).

As an illustration, for the initialization in Figure 6.3a, the computed gradient is represented after convergence in Figure 6.3b. It converges in the synchronous mode as well as in the asynchronous mode. The wavefront expansion algorithm in [Barraquand et al., 1992] is a time-step equivalent to our synchronous mode, but here, we allow multiple updates of any cell. Also, their expansion algorithm maintains a list of currently considered cells (the wavefront) at each time step, whereas we do not. Sequential methods from the field of Image Processing would consider multiple scans of the entire automaton in predefined orders. In our asynchronous mode, a cell is chosen according to a uniform distribution at each time t.

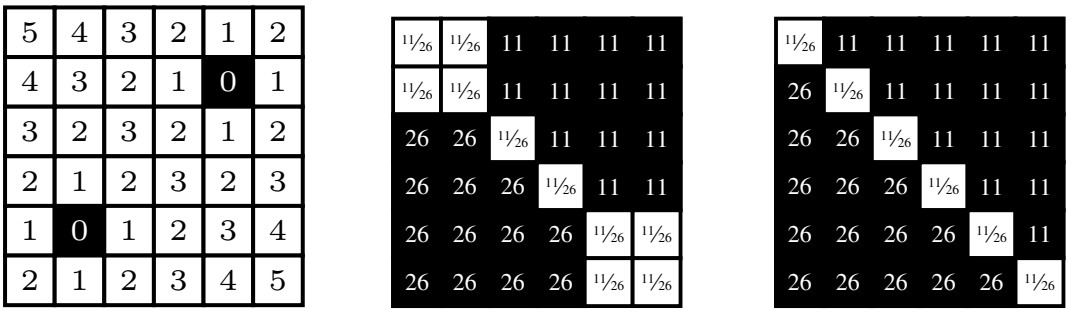
Semantic layer Now, we incorporate a semantic information into any cell: a cell identifier denoted by *id*. Spreading a site identifier across the automaton can help marking the corresponding *Voronoi* region. For this purpose, we take advantage of the gradient layer to ensure a correct semantic layer computation. Thus, we use both the *gradient value* d_1 and the *identifier* *id* as a pair of information to propagate.

Let us independently initialize the *id* of each cell to a unique constant identifier id_i. Sites reset their *id* to id_i at each step. Non-sites collect these sites' identifiers into a set id_c by neighboring mechanisms. Collected identifiers are taken from every *Von Neumann* neighbor k meeting two conditions. The first condition δ checks that k has the lowest gradient value among its siblings. The second condition ϵ additionally verifies that k holds a smallest set of identifiers. Once collected, these identifiers are merged into id_c. The rule IG synthesizes the gradient and

semantic layers computation:

$$IG(c) = (c.d_1, c.id) = \begin{cases} (0, id_i) & \text{if } site(c) \\ (m{+}1, id_c) & \text{otherwise.} \end{cases}$$
$$id_c = \{k.id \mid k \in u_1(c) \backslash \{c\} \ s.t. \ \delta \wedge \epsilon\}$$
$$\delta = (k.d_1 = m)$$
$$\epsilon = (|k.id| = \min_{k \in u_1(c) \backslash \{c\}; \delta} |k.id|)$$

Figure 6.4 shows the *gradient* layer and two *semantic* layers after convergence of rule IG. Cells are numbered, row by row, from left to right, to provide unique constant identifiers, id_i, for initialization. In Figure 6.4a, the correct identifiers id are spread over the lattice upon convergence of the gradient. In Figures 6.4b and 6.4c, three regions are noticeable after convergence; each region is composed by the cells possessing the same set of identifiers ({11}, {26}, and {11,26}). The semantic parsimony condition, ϵ, provides larger regions for the sites collecting either the identifier {11} or {26}, and in between, a thin boundary region collects both identifiers {11,26}, see Figure 6.4c.



(a) d_1 async/sync **(b)** $id\ \delta$ async/sync **(c)** $id\ \delta\epsilon$ async/sync

Figure 6.4: Gradient and semantic layers with *sites* 11 and 26.

We designed a rule that can tackle the *random wind* and *wave fusion* phenomena using a two layer CA in the asynchronous mode. The *distance layer* computes a 1-norm integer gradient from the site cells. The *semantic layer* can then correct the identifier propagation until convergence of the distance map. We will now study bisector extraction between two sites, based on the synchronization-free rule, IG.

6.4.2 Bisector extraction

Thick bisector An intuitive way to extract the frontier between two *Voronoi* regions is by marking the boundaries of these regions. Thus, a non-site receiving a different identifier from at least one of its neighbors is a frontier, see the skeleton generation algorithm from [Fischler and Barrett, 1980]. The following rule, Bis_{thick}, formalizes this intuition.

$$Bis_{thick}(c) : c.bis \leftarrow \begin{cases} 1 & \text{if } \exists k \in u_1(c), k.id \neq c.id \\ 0 & \text{otherwise.} \end{cases}$$

In order to enumerate all kinds of bisector, we reproduce the examples from [Adamatzky, 1996]. Six cases are generated by varying the spacing between two simple sites according to odd or even

1-norm distance. Let us consider two sites, a and b, and their horizontal and vertical distances, $s_x = |x_a - x_b|$ and $s_y = |y_a - y_b|$. If $s_x = s_y$, then s_x is either an odd or an even distance (2 cases). Otherwise, we have $|\{odd, even\}^2| = 4$ other cases. Bisectors extracted using Bis_{thick} are given in Figure 6.5.

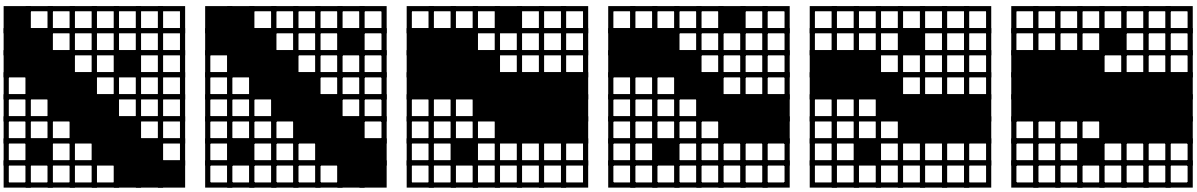


Figure 6.5: Thick bisector extraction cases for two sites.

Cases 1, 2, 3, and 6 are for odd distances between the sites; the bisector has a thickness of three cells. For even distances between the sites, the bisector has a thickness of two cells. A complete bisector is extracted for all cases but is too thick for odd distances. Therefore, we design a thin bisector extraction rule which exploits more of the available semantics. This rule is based on the discrete bisector definition.

Thin bisector We reproduce the definition of a discrete bisector from [Adamatzky, 1996] in Definition 6.3.

Definition 6.3 (Discrete bisector). *The discrete bisector, Bis, of cells, a and b, is defined as the set of cells, c, such that the 1-norm distances, $d(a, c)$ and $d(b, c)$, differ by at most 1.*

$$Bis(a, b) \leftarrow \{c \in A \mid 1 \geq |d(a, c) - d(b, c)|\}$$

We can discern two types of bisector cells. The first type, corresponds to non-sites exactly equidistant to several sites. The second type, detects (modulo 1)-equidistance to multiple sites. From a cell-centered view, first-type bisector cells collected several site-identifiers (set *bis* to 1). Second-type bisector cells collected only one identifier and have a neighbor which owns only one, but different, identifier (set *bis* to 2). The following rule, Bis_{thin}, can extract thin bisectors.

$$Bis_{thin}(c) : c.bis \leftarrow \begin{cases} 1 & \text{if } |c.id| > 1 \\ 2 & \text{if } |c.id| \leq 1 \\ & \wedge(\exists k \in u_1(c), |k.id| = 1 \wedge k.id \neq c.id) \\ & \wedge(\nexists k \in u_1(c), site(k)) \\ 0 & \text{otherwise.} \end{cases}$$

Figure 6.6 illustrates thin bisector extraction for the cases of two-sites spacing. Thinner bisectors are extracted for cases 1, 2, 3, and 6, when compared to the previous intuitive rule. Every bisector has the minimum thickness allowed in our model.

Based on the *information gradient* and the *complete bisectors*, we can already compute the VD of simple cellular sites in a distributed manner and without synchronization. To recapitulate this section, we provide an example of such computation in Figure 6.7. Sites are represented as black cells (Figure 6.7a). The *gradient layer* from each site is updated in a distributed fashion

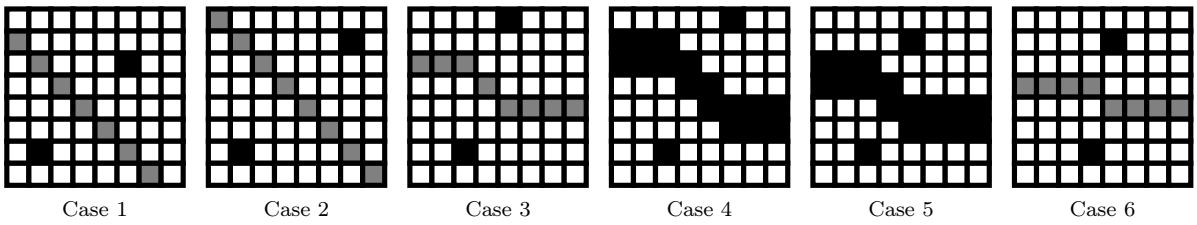


Figure 6.6: Thin bisector extraction cases for two sites (black dots), cells with bis 1 are in black and cells with bis 2 in gray.

using rule *IG* (Figure 6.7b). The *semantic layer* is updated according to the current gradient layer; each color represents an identifier. Identically colored cells share the same identifier and form a *Voronoi* region (Figure 6.7c). Based on this *information gradient*, bisector cells are identified locally using Bis_{thick} or the thinner rule Bis_{thin} (Figures 6.7d and 6.7e). The VD and the bisectors are correct as soon as the gradient layer converges to the real closest site distances. Moreover, the synchronous and asynchronous modes provide the same results when the gradient converges.

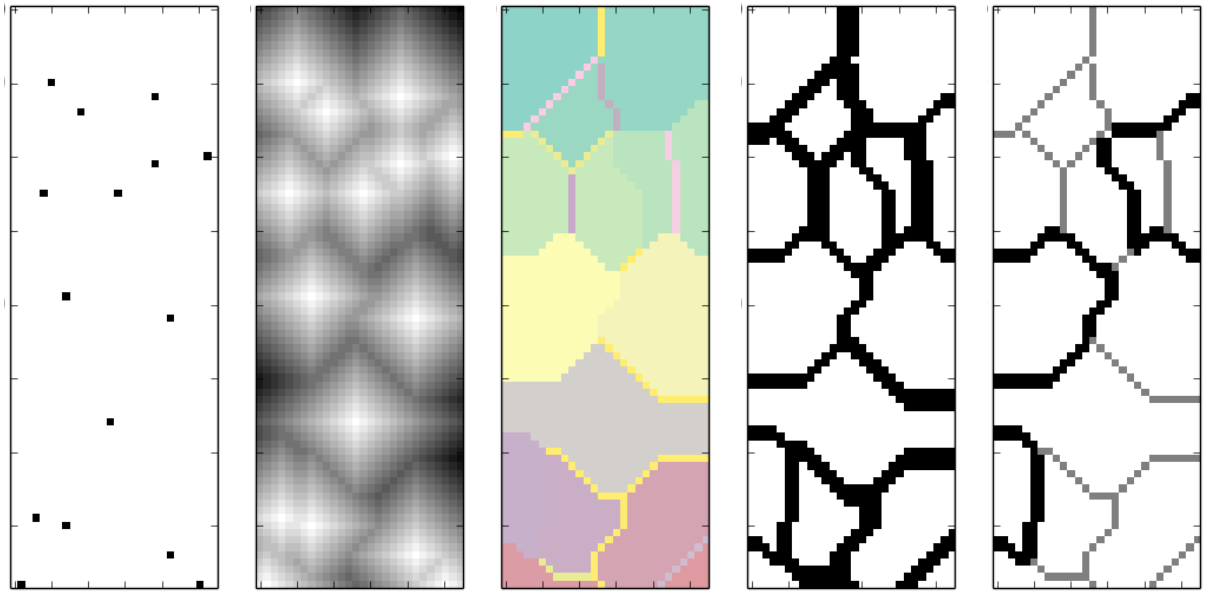


Figure 6.7: Discrete Voronoi Diagram extraction with cell sites.

The next section extends these principles for extracting Area Voronoi Diagrams (AVDs) and Line Voronoi Diagrams (LVDs). Our extension considers sites of contiguous cells forming areas or simple polygonal edges. We show how these contiguous regions are discovered locally by neighboring mechanisms; and also how cells from a same region can implement a consensus mechanism to compute a common identifier.

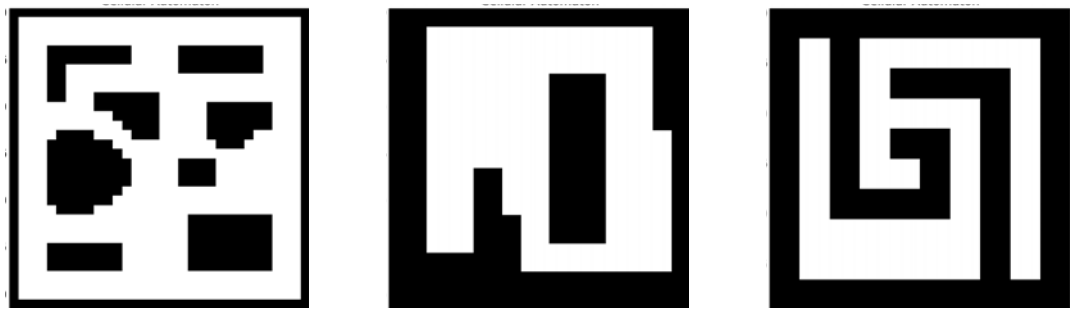
6.5 Experiments

First, we present the typical maps on which we want to extract maximum clearance roads for robotic navigation. Then, we extend our study step by step from cellular sites to area sites and then to line sites VDs. To achieve this, new rules are required to allow each cell from a same generalized site to propagate a unique semantic. In other words, contiguous site cells will decide on a common identifier to propagate as one single entity on the lattice.

Simulator and maps

The rules for identification, propagation, and bisector extraction presented in this study were implemented using, Python and C++, and executed in the synchronous and asynchronous modes.

Some of the maps used for the experiments in Figure 6.8 are reproduced from [Adamatzky, 1996, Tzionas et al., 1997]. These maps have multiple obstacles, narrow passages, and maze-like structures with concavities. We show how AVDs can create simple roadmaps but are not satisfying for maze-like structures and concavity.



(a) Map 1 32x32 **(b)** Map 2 32x32 **(c)** Map 3 30x30

Figure 6.8: Occupancy grids with free cells (white) and occupied cells (black). Each map is cast as a cellular automaton that uses occupied cells as sites.

6.5.1 Area Voronoi Diagram

To compute AVDs [Okabe et al., 2009], we consider *site areas* aggregating contiguous *cellular sites*.

Area identification We design a component labeling procedure to identify *areas* of contiguous *cellular sites*. We assume that each cell has a unique initial identifier id_i and that the set of unique identifiers is totally ordered. The site area identifier is locally computed using rule AId by retaining the minimum identifier id_a of the neighboring sites.

$$AId(c) = c.id = \begin{cases} c.id & \text{if } \neg site(c) \\ id_a & \text{otherwise.} \end{cases}$$

$$id_a = \min_{k \in u_1(c), site(k)} (k.id)$$

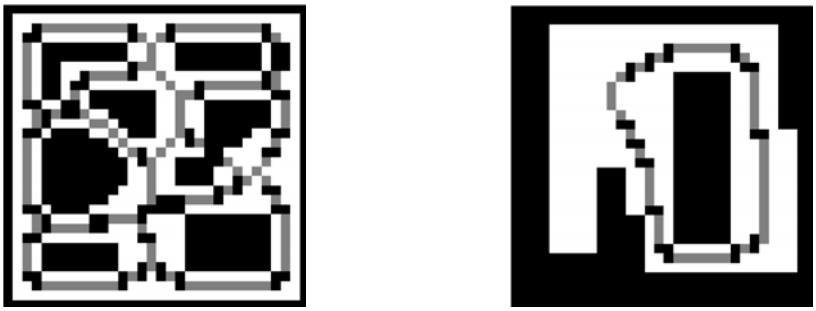
At the global level, the *area identification consensus* converges when every cell in a same area shares a common identifier.

Algorithm When a cell is selected, it executes three steps: the first one is the *area identification*, which is followed by the *propagation*, and finally the *bisector extraction*; they are respectively processed by the rules AId, IG, and Bis. An intermediate step is introduced to allow a site cell to update its initial identifier as the newly computed area identifier. Thus, each cell executes Algorithm 1 when selected.

Algorithm 1 Area Voronoi diagram (AVD)

$AId(c)$
if $site(c)$ **then**
  ▷ update initial to area identifier
  $c.id_i \leftarrow c.id$
$IG(c)$
$Bis(c)$

This algorithm was executed by every cell on the three lattices from Figure 6.8. The first lattice provides 9 *Voronoi* regions corresponding to the 9 areas (border and 8 simple polygons). The second lattice provides 2 regions and the third lattice only 1. These AVDs are complete.



(a) Map 1 **(b)** Map 2

Figure 6.9: Thin skeleton computed by an Area Voronoi Diagram.

Limitation for robotic navigation These examples bring forward the fact that the bisectors extracted using an AVD are not satisfying for establishing a maximum clearance roadmap useful in robotic navigation. For instance, the concavities were ignored in the first two maps. Hence, in the context of robotic navigation on the areas' bisectors, a robot could not navigate into these concavities. Moreover, as presented above, no road was extracted for the maze-like structure in Map 3. Therefore, we now consider VDs computed for line sites. To do so, we will have to subdivide the boundary of a site-area into its edges. Thus, we design edge-identification rules.

6.5.2 Line Voronoi Diagram

In this section, we compute LVDs [Okabe et al., 2009]. To achieve this, we generate bisectors between the edges of a shape. First, we discuss *edge-patterns* on the contour of a site-area. Then,

we provide *edge-identification* mechanisms to reduce the number of unwanted bisectors. Finally, we compute LVDs using these corrected identifiers.

Edge patterns In our approach, we consider an identifier for each horizontal, vertical, or diagonal edge of an area. We use edge detectors as in [Tzionas et al., 1997], but in strict u_1-neighborhood. Every cell detects its edge-pattern p from one class of the possible u_1-masks, see Figure 6.10. These classes represent rotational symmetry. However, as mentioned in [Sudha et al., 1999], using only p as identifier will result in some artifact bisectors. We aim to reduce the number of such artifacts.

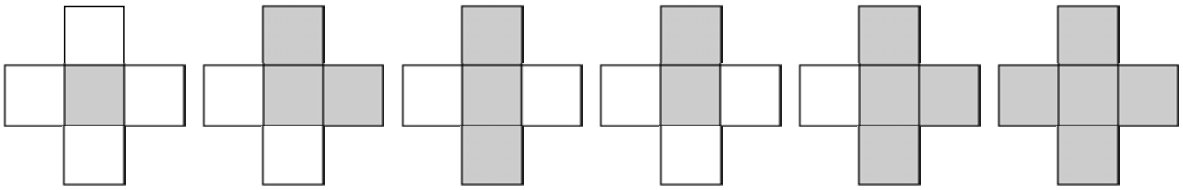


Figure 6.10: Classes of site patterns and their cardinality.

Edge identification Reducing the number of artifacts involves local detection and correction of conflicting edge-patterns causing these artifacts. Conflicts are typical cases of neighboring edge-patterns that create unwanted bisectors. If a conflict is locally detected then it is resolved by choosing another identifier instead of the edge-pattern p:

$$EId(c) : c.id \leftarrow \begin{cases} solve(c) & \text{if } conflict(c) \\ c.p & \text{otherwise.} \end{cases}$$

Conflict resolution is enabled via some cells acting as conciliators, V, for two non-neighbor cells, while others alternatively act as yielders or cooperators, II and IV. Detailed conflict detection and resolution are given in Table 6.1.

In Figure 6.11, three conflicts (II/V), (IV/V), and (II/IV) are illustrated. If each cell transmits its edge-pattern on the lattice, it will result in a high number of bisector cells (drawn in black in Figure 6.11a). But, if some cells provide different identifiers according to the directions, we can solve the local conflicts and reduce the number of artifacts. Cells from the classes II and IV express this behavior, see Figure 6.11b.

Algorithm We can now compute LVDs by replacing AId with EId in Algorithm 1. The extracted roadmaps are provided in Figure 6.12. It is now possible to enter the concavities and the number of artifacts is reduced.

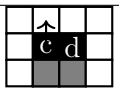
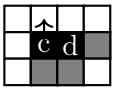
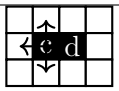
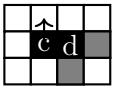
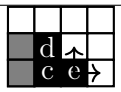
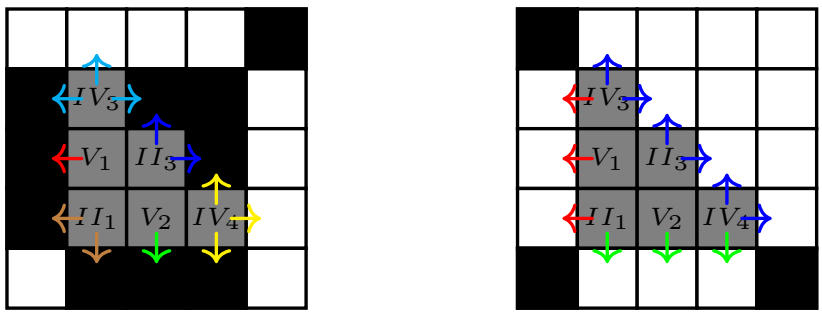
conflict(c) pre-conditions	cases	solve(c) id provided in direction(s)
$c.p \in I$		$c.id_i$
$c.p \in II \wedge d.p \in II$		$uniq(c,d)$
$c.p \in II \wedge d.p \in \{III, IV, V\}$		$d.p$
$c.p \in III \wedge d.p \in III$		$uniq(c,d)$
$c.p \in III \wedge d.p \in \{IV, V\}$		$d.p$
$c.p \in V \wedge d.p \in II \wedge e.p \in III$		$inform(e,d.p)$

Table 6.1: Edge identification consensus, column 1 gives the pre-conditions for a conflict detection, column 2 are the conflict cases requiring consensus identification, column 3 provides the identifier chosen for the arrow direction.

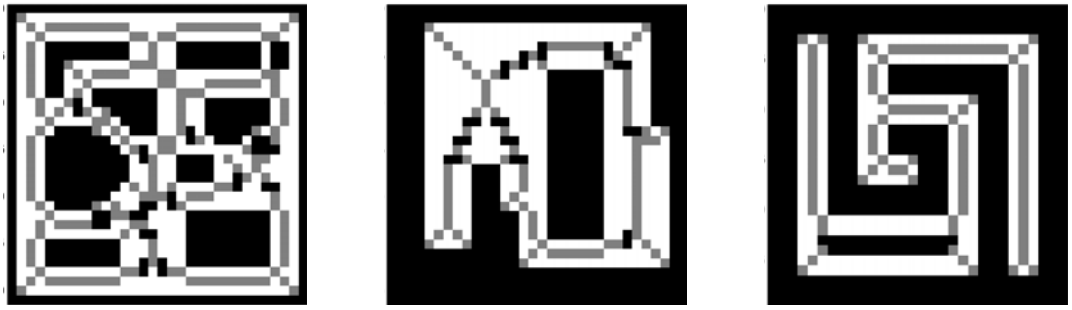


(a) Edges **(b)** Consensus

Figure 6.11: Conflicting edge patterns and resolution.

Algorithm 2 Line Voronoi diagram (*LVD*)

```
EId(c)
if site(c) then
    ▷ update initial to edge identifier
    c.id_i ← c.id
IG(c)
Bis(c)
```



(a) Map 1 **(b)** Map 2 **(c)** Map 3

Figure 6.12: Thin skeleton computed by an Line Voronoi Diagram.

6.6 Conclusion

In this chapter, we studied the Clearest Path Obstacle Avoidance (CPOA) task in an ambient environment equipped with load-pressure sensing units embedded in the floor. We designed local rules that can be deployed on such processing and sensing units to compute navigable paths that are equidistant to the obstacles in the environment. Such paths allow to maintain a navigable map of the environment and ease path planning of mobile robots.

We proposed to compute VDs in a distributed fashion without any explicit synchronization. We tackled the *random wind* variations due to the asynchronism and the *wave fusion* phenomenon of the gradient by combining a distance layer and a semantic layer. This allowed to extract complete bisectors and generate discrete VDs for simple sites in 1-norm. This work was then extended to AVDs and simple LVDs.

Our application was oriented toward robotic navigation. We extracted maximum clearance roadmaps using a network of regularly distributed sensors simulated by the CA. This work could be extended to dynamic routing in distributed networks considering overloaded sites, multi-robot planning, and natural computing simulation if any synchronization mechanism is provided.

Our approach satisfies both synchronous or asynchronous computation on a static or dynamic map with moving obstacles. The following table compares our method to others depending on qualitative criteria, see Table 6.2. These criteria should be further investigated with quantitative metrics.

This work has been published as a technical report [Kaldé et al., 2014d], and as an article at an international conference in *Artificial Intelligence* [Kaldé et al., 2014c]. An application of these techniques on a real sensing floor is reported elsewhere, in chapter 7 of [Andries, 2015].

	Synchronous	Asynchronous	Centralized	Decentralized	Static	Dynamic.
LVD (ours)	Ok	Ok	Ok	Ok	Ok	Ok
[Barraquand et al., 1992]	Ok	-	Ok	-	Ok	-
[Tzionas et al., 1997]	Ok	-	-	Ok	Ok	-
[Adamatzky, 1996]	Ok	-	-	Ok	Ok	-

Table 6.2: Qualitative comparison of different techniques for computing Voronoi Diagrams along the following axes: synchronization of units, centralization of computation, dynamics of environment

Summary

Chapter 7

Conclusion and Perspectives

Contents

In this chapter, we summarize the contents of this thesis and provide perspectives for each study. Firstly, we studied the Active Coverage (AC) and Active Mapping (AM) tasks of Robotics as Sequential Decision-Making (SDM) problems in Artificial Intelligence. We discuss our specification of Single-Robot AC and AM as decision processes; and the resolution of the deterministic Single-Robot AC decision problem with shortest path planning algorithms.
Secondly, we studied the Multi-Robot AM task in dynamic environments with pedestrians; and the Clearest Path Obstacle Avoidance (OA) task in ambient environments with embedded processing and sensing units. We discuss our proposal to exploit pedestrians' flows during exploration; and to exploit the embedded units to compute clearest paths for navigation.

7.1 Conclusion

In this section, we summarize our contributions about modeling and solving AC and AM tasks in the planning paradigm, and our contributions concerning AM in dynamic environments and OA in crowded environments.

7.1.1 Part I: State Space Modeling for Active Coverage and Mapping

In the first part, we have reinforced the link between coverage and exploration tasks. In our opinion, these tasks can be represented at both ends of a continuum of prior information

regarding the environment. When prior information is not available, the robots are facing the most difficult exploration task for which greedy strategies may be the best option. Nowadays, greedy mapping [Koenig et al., 2001, Tovey and Koenig, 2003] is widely spread in experimental Robotics, but unfortunately this paradigm is blind to prior information. At the opposite, when a complete prior is available, the robots are facing the easiest coverage task for which optimal plans may be tractable. There are several studies dedicated to *Coverage Path Planning* [Galceran and Carreras, 2013] but it is unclear how to adapt such techniques to an incomplete prior. Finally, the solvers dedicated to exploration and coverage differ and there is no explicit link between them. Therefore, we reinforced this link by modeling both tasks in the framework of SDM.

Our approach was exploratory, it witnessed a fundamental will to grasp and express the variants of coverage and exploration within a same framework. In this framework, we defined domains that abstract coverage and exploration tasks to their essential building blocks: the robot's actions, control model, measurements, and sensor model. Moreover, we defined decision processes that abstract state estimators outputting pose and map estimates useful for autonomous robots. These building blocks are only a few but their combinations describe many exploration tasks with strict assumptions. This alone allows to clearly name and strictly specify several variants of exploration that lay between *Classic Exploration* where pose and map estimates are reliable and *Integrated Exploration* where they are noisy.

Finally, we defined decision problems with an optimization criterion regarding long-term coverage and exploration behaviors. We took inspiration from problems related to coverage and exploration in Graph Theory. For instance, we relied on the Traveling Salesman Problem to evaluate the Minimum Coverage Path in Active Coverage, and on the Canadian Traveler Problem to evaluate its expectation over a class of environments in Active Mapping. Strictly speaking, we defined the immediate rewards received by the robots at each time step, and such rewards naturally lead to long-term criteria according to the planning horizon. More precisely, we penalized lengthy coverage paths but rewarded information gathering. Optimal coverage alone is already *NP-hard*. Nevertheless, defining optimal exploration from optimal coverage allows to view coverage as a special case of exploration. Furthermore, the results obtained on coverage directly transfer to exploration.

In Chapter 2, we generalized the planning agent for coverage, that firstly appears in Table C.2, to three types of control (deterministic, nondeterministic, and stochastic), with deterministic local sensing, but predictable measurements outcomes thanks to the complete prior regarding the environment. We modeled three decision processes over knowledge states, the most general is based on the Completely Observable Markov Decision Process formalism; each decision process can support decision-making by a planning agent for coverage.

In Chapter 3, we generalized the planning agent for exploration, that firstly appears in Table C.2, to three types of control and sensing (deterministic, nondeterministic, and stochastic), but with unpredictable measurements outcomes due to the incomplete prior regarding the environment. We modeled three domains of exploration over hidden world's states, the most general is based on the Partially Observable Markov Decision Process formalism; and specified the associated decision processes over agent's belief states, the most general is based on the Completely Observable (Belief state) Markov Decision Process formalism, to support decision-making by a planning agent for exploration.

We provided several domains and decision processes with increasing complexity and realism. These domains are simulators dedicated to single-agent coverage and exploration tasks. They define strict modalities of interaction between a robot and its environment and are independent of the solving paradigm. The associated processes maintain as much information as possible for the robot in a compact way, and are necessary to build planning agents for coverage or exploration. We hope to encourage the design of long-term model-based (planning) or model-free (learning) agents for coverage and exploration. Hence, we will propose several improvements and extensions in the perspectives.

7.1.2 Part II: Long-Term Planning for Deterministic Active Coverage

In the first part, we have modeled several domains of coverage and exploration for a single robot. Additionally, we have provided decision processes to support decision-making by a planning agent. In the second part, we investigated the performances of a planning agent in such domains. To do so, we focused on the deterministic coverage task.

On the one hand, any Minimum Coverage Path computed in the coverage domain can be used to approximate the *Expected* Minimum Coverage Path in exploration (see Table C.2). In other words, if we can efficiently solve Active Coverage (AC) then we can more efficiently address Active Mapping (AM). Therefore, we evaluated standard planning techniques on AC in simulation, in terms of computation and execution costs. Such experiments are necessary to build empirical knowledge that is currently lacking, as such techniques are usually evaluated on generalized puzzles. Unfortunately, not much is known regarding their scalability to other applications. To address this issue, we evaluated their average performances on the deterministic coverage domain.

On the other hand, there is no consensus regarding the best practice for evaluating exploration strategies. It is common to make relative comparisons by monitoring the average traveled distance and elapsed time over several environments [Faigl et al., 2012]. Other studies consider the global competitive ratio instead, *i.e.*, the maximum performance ratio between an on-line exploration strategy and the off-line Minimum Coverage Path (MCP) over many environments. Good strategies have the lowest competitive ratio. In [Faigl and Kulich, 2015], the MCP is estimated among thousands of satisfiable trajectories, whereas [Li et al., 2012] reports a proof of concept where it is exactly computed with off-line search. To encourage the adoption of competitive ratios, we extended the work of [Li et al., 2012] by conducting a more extensive benchmark.

To summarize, solving instances of the coverage task is not only valuable for improving the exploration strategy of a planning agent, it is also useful to evaluate the competitive ratio of greedy strategies and hence identify the environments where planning agents may be useful.

In Chapter 4, we evaluated the performances of the planning agent for coverage. This planning agent has been illustrated in Table C.2, and later its decision model has been specified in the Completely Observable (State) Search Problem (COSP) formalism in Section 2.3.1. We reviewed and applied standard planning techniques on instances of the single-robot coverage task described by the deterministic AC decision process. We focused on off-line heuristic search, with its two most popular frameworks: Best-First Search and Depth-First Search. We selected four pioneering algorithms: Iterative Best-First Search, Recursive Best-First

Search, Iterative-Deepening Depth-First Search, and Branch-and-Bound Depth-First Search. We reported four admissible heuristics to prune the search space: two navigation heuristics MinDist and MaxDist evaluate the distance from the robot to the nearest or farthest frontier; and two coverage heuristics MaxCells and MinDist+MaxCells evaluate the minimum number of remaining observations (the latter also accounts for the nearest frontier distance).

We empirically demonstrated that the coverage heuristics consistently performed better than the navigation heuristics by several orders of magnitude in terms of time consumption; they could optimally solve small instances of the coverage task under scarce resources; and 2-optimally solve the same instances by even further reducing search costs. Finally, we underlined the anytime behavior of Branch-and-Bound Depth-First Search which could improve the first solution found by a greedy coverage agent over time.

We considered small instances on rectangular grids that never exceeded 30 cells, hence we expect but cannot guarantee that these results will generalize to bigger instances. Nevertheless, these results are encouraging as we limited the time consumption to 2.10^7 node generations, whereas it is not rare to allow for much more in combinatorial games, *e.g.*, 1.10^9 generations are reported for the 15-puzzle [Korf, 1985], or 1.10^{12} generations are reported for the 25-puzzle or the Rubik's cube [Korf and Taylor, 1996, Korf, 1997].

Our current usage of off-line search algorithms for optimal AC significantly differs from the applications of these techniques in greedy AM. We use these techniques on the deterministic domain of coverage with a complete prior; the associated decision process allows to foresee all future frontiers in advance given sufficient resources; and the planning agent can compute an optimal coverage trajectory through the current and future frontiers. Distance-based exploration strategies use these techniques on the deterministic domain of navigation to evaluate the shortest path between robots and current frontiers in MinDist [Yamauchi, 1997] or *MinPos* [Bautin et al., 2012]; and *MTSP* [Faigl et al., 2012] relies on the minimum cost cycle through the current frontiers.

To conclude, we use these techniques on the coverage domain that is exponentially bigger than the navigation domain and plan beyond the current frontiers as we can anticipate future frontiers with certainty thanks to the complete prior. This part ends our work dedicated to modeling coverage and exploration tasks of Robotics as Sequential Decision-Making problems. We want to encourage more studies in this direction, hence, we will provide some guidance in the perspectives.

7.1.3 Part III: Short-Term Planning for Deterministic Active Mapping and Clearest Path Obstacle Avoidance

In the second part, we have studied the off-line resolution of Single-Robot Active Coverage with heuristic search. Furthermore, we have empirically demonstrated that small instances could be solved either optimally, suboptimally (with guarantees), or in an anytime manner under bounded resources. In this third part, we considered a more realistic setup, in which many robots navigate in crowded or ambient environments in a decentralized manner. Our previous models of decision-making do not address these cases.

Decentralized decision-making for exploring dynamic environments

Mobile robots increasingly share the physical space with humans but studies about exploration remain mainly focused on robust robot control for pose and map inference. Nevertheless, it becomes urgent to address exploration systems that are aware of humans. This will be necessary for coordinating teams of robots and humans on the field in a scenario of search and rescue. Hence, as a first step, we studied the scenario of exploration in dynamic environments with pedestrians. Moving objects are usually considered as obstacles that must be avoided, but pedestrians can efficiently navigate through the crowds and may be important assets for mobile robots in such environments. We investigated a simple kind of physical Human-Robot Interaction during exploration.

In Chapter 5, we assumed that robots were equipped with field-of-view sensors that could distinguish between free space, space occupied by a mobile object, or space occupied by a static object; and that such sensors were perfect and deterministic. This allowed to extend the robot's knowledge representation from three (free, static occupancy, and occlusion) to four occupancy status (mobile occupancy) for each cell. Consequently, we extended the concept of a frontier to avoid early termination in case a pedestrian blocked the access to an unexplored area. We modeled a multi-agent system with deterministic control and sensing that enables the frontier paradigm in such environments.

We proposed an interactive paradigm to exploit pedestrians' flows; and focused on interactions in which a robot locally follows a pedestrian. We hybridized this paradigm to consider both frontier and interaction assignments, and relied on an on-line decentralized task allocation framework to assign robots to such targets. Targets were chosen among either frontier cells to attain or pedestrians to follow in the environment. We proposed to compute assignment costs with planned distances to the targets and heuristic penalties associated to the targets. These penalties measured the idleness and orientation of the pedestrians and the discovery time and orientation of the frontiers, relative to the robot.

Our cost function was parametric, it firstly weighted distances against penalties, and secondly weighted frontier against interaction penalties. We quantitatively evaluated the average performance of a team of two robots in simulation within three environments with fixed weights. We used metrics of exploration (coverage ratio, traveled distance, and elapsed time), and a Human-Robot Interaction metric (robot attention demand). The best performances did not promote interactions and mainly favored frontiers. A more accurate cost function and a more realistic pedestrian behavior is required. Nevertheless, our multi-agent system and the hybrid paradigm can support further studies for exploration in such environments. We will discuss potential improvements in the next section.

Distributed roadmap computation for navigating ambient environments

We saw that mobile robots can locally observe the environment through their sensors, compile these observations on an occupancy map, and plan their moves according to this map. Although robots can build such maps in a decentralized manner by remotely communicating with their teammates, path planning is still difficult in dynamic conditions. Indeed, each robot requires several iterations of replanning to attain a given location, as the computed plans quickly become obsolete and need to be repaired. Unfortunately, maintaining a traversability map up

to date is costly and necessitates continuous patrolling. We studied how to collaboratively build navigation services by relying on ambient intelligence in order to reduce the on-board computational burden of mobile robots.

In Chapter 6, we studied the Clearest Path Obstacle Avoidance task in ambient environments, which are equipped with sensing, processing, and communicating units. We assumed that the ambient environment was equipped with a floor that is made of tiles able to sense load pressure, store and process information, and communicate with neighboring tiles. We showed that these embedded units could reliably compute safe navigation roads to help mobile robots navigate in the environment. To do so, we proposed a novel technique for computing a maximum clearance roadmap by the means of spatial computing. This computation relies on processing and sensing units embedded in a load pressure sensing floor. We see it as an intermediate step toward tasking embedded units and mobile robots to do it together.

Our method is distributed, asynchronous, and computes a safe roadmap for navigating in ambient environments. We formalized the environment as a two-dimensional grid cellular automaton. This formalization abstracts the tessellation of units on the floor as a lattice of cells. Each cell runs a simple program, that only uses as input its own binary (free, occupied) load data and neighboring values. We proposed two ways to compute maximum clearance roadmaps on the lattice in an asynchronous and distributed manner, by seeding obstacles on the floor as *Voronoi* sites on the lattice, and extracting bisectors between *Voronoi* regions as safe roads on the floor. We then qualitatively evaluated the roadmaps computed by our method on a synthetic binary dataset (deterministic occupancy grids).

This ends our conclusion regarding the studies that are reported in this manuscript. Now, we will turn our attention to the perspectives.

7.2 Perspectives

In this section, we list a series of natural extensions of our work; such extensions could be immediately investigated.

7.2.1 Part I: State Space Modeling for Active Coverage and Mapping

We envision the following perspectives to extend or improve the domains and decision-making models of coverage and exploration.

A first extension is to generalize the COMDP-based centralized agent for Single-Robot coverage to Multi-Robot coverage. For instance, an extension for a team of two synchronous robots requires to consider the vector of joint poses, actions, and transitions, and the sum of the vector of costs of the robots. This extension allows to compute joint optimal coverage plans off-line, and to evaluate the competitive ratio of on-line Multi-Robot exploration strategies.

A second extension is to generalize the COMDP-based centralized agent for Multi-Robot coverage (previous point) to decentralized agents. For instance, an extension for two asynchronous robots can consider that the knowledge state is completely observable only when robots are within communication range (they share their current location and coverage map). This extension could be based on Decentralized MDPs (Dec-MDPs) [Bernstein et al., 2002].

It is necessary to improve the scalability of the COMDP$_{b^{az}}$-based centralized agent for Single-Robot exploration. Hence, a first step is to translate the ρPOMDP exploration domain to mixed observability to scale up the implementation [Ong et al., 2010, Araya-López et al., 2010b]. However, this decision process still suffers from the curse of dimensionality. That is, if the robot hesitates between n maps, the belief state is a distribution over n maps. Therefore, a second step is to reason over maps features. For instance, a hierarchy of features might be automatically obtained by training a Convolutional Neural Network with Autoencoders to cluster potential maps [Bengio et al., 2009].

7.2.2 Part II: Long-Term Planning for Deterministic Active Coverage

On the one hand, we provide some guidance for continuing this study, and pointers to other paradigms of deterministic search that apply to this deterministic coverage decision process.

We considered the off-line paradigm for solving small instances of the deterministic coverage domain. This study must be continued by proposing more accurate heuristics to reduce search costs even further, and facilitate the computation of competitive ratios. Additionally, it is important to note that there are two other paradigms of search. For instance, on-line search should be investigated [Ramalingam and Reps, 1996, Koenig et al., 2004]; it allows to repair optimal paths when the domain changes. In theory, if we know the optimal coverage path for a map A and this map is locally modified as $B = A + \delta$; then the optimal coverage path of A can be repaired to obtain the optimal solution for B. Repairing this coverage path can be more efficient than planning it from scratch depending on δ. Furthermore, real-time search should be investigated [Korf, 1990, Pemberton and Korf, 1992]; it allows to concurrently plan and execute satisfiable solutions. Again, in theory, some real-time search algorithms learn the heuristic over time and can converge to optimality over many trials [16].

On the other hand, we provide some pointers to algorithms that can address the stochastic coverage and exploration decision processes.

Computing policies to the COMDP-based coverage decision-process requires to plan without knowing the true state outcome of each action in advance. Indeed, the true state outcome of each action is unveiled only during execution after each action is applied. This state-contingency problem is addressed in the literature of stochastic shortest path algorithms that rely on *Bellman* backups, see [Russell and Norvig, 1995, Sutton and Barto, 1998, Bonet and Geffner, 2000]. The popular Upper Confidence Bound for Trees (UCT) search algorithm from the Monte-Carlo Tree Search (MCTS) framework also applies here and relies on *Monte-Carlo* backups instead, see [Browne et al., 2012].

Computing policies to the COMDP$_{b^{az}}$-based exploration decision-process requires to plan without knowing the true belief-state outcome of each measurement in advance. Indeed, states are perceptually aliased due to state uncertainty, and contingencies are resolved after each ob-

[16] In practice, we tested the pioneering Learning RTA* (LRTA*) [Korf, 1990] on the deterministic coverage domain with 5 steps lookahead and the MinDist+MaxCells heuristic. It required several thousands trials to converge to optimality on one of our 5×5 grid. Although optimal coverage paths were found several times before convergence, convergence is slow and the performances of LRTA* are nonmonotonic over trials (this proof of concept is not reported). Nevertheless, this is encouraging and more recent algorithms must be investigated.

servation during execution. This observation-contingency problem is addressed in the literature of stochastic search as well. However, another characteristic of exploration is that information gathering must be rewarded. This is addressed in the literature of active sensing [Bajcsy, 1988, Mihaylova et al., 2003]. Some techniques for solving ρPOMDPs (POMDPs for information gathering) are based on *Bellman* backups as in [Araya-López et al., 2010a] [17] . In their study, the standard value iteration algorithms for POMDPs are generalized for belief-based rewards. Other techniques for solving POMDPs are based on *Monte-Carlo* backups as in [Silver and Veness, 2010]. In their study, they proposed Partially Observable UCT (PO-UCT) that generalizes UCT over hidden states and where the decision is based on sampled histories. This technique needs to be extended to belief-based rewards.

7.2.3 Part III: Short-Term Planning for Deterministic Active Mapping and Clearest Path Obstacle Avoidance

Decentralized decision-making for exploring dynamic environments

We made a contribution about decentralized decision-making for exploring crowded environments, by exploiting the flows of pedestrians. Here, we list several ways to improve or extend this study.

We used the greedy mapping task-allocation framework to assign robots to targets. In this framework, any pair of visible robot and known target is evaluated independently. This independence assumption is common in Multi-Robot Active Mapping, but the value of a first assignment depends on the other assignments, insomuch as paths can cross and fields of view can overlap. Thus, a first perspective is to extend the standard greedy mapping framework to evaluate joint assignments at once. Each joint assignment is an instance of the Multi-Agent Pathfinding problem [Wang et al., 2008].

We manually selected idle time and orientation features to penalize pedestrians behaviors relative to the robot. Such features and the way to combine them may vary depending on the environment's type or the local situation. Hence, a second perspective is to automatically select the features and learn how to combine them. *Embodied Evolution* [Watson et al., 2002] may be used to train a Recurrent Neural Network on robot's sensory inputs and reinforce synaptic weights when interactions are deemed beneficial to exploration.

We relied on simple pedestrian behaviors that do not reflect reality. Hence, a third perspective is to consider a more realistic pedestrian behavior, such as the *Social Force Model* [Helbing and Molnar, 1995] that globally imitates the behavior of pedestrian crowds by applying local rules on individuals.

Distributed roadmap computation for navigating ambient environments

We made a contribution about distributed and asynchronous computation of safe roadmaps for navigating in ambient environments. Here, we list several ways to improve or extend this study.

[17] In practice, we advocate using functional representations instead of flat representations when implementing these value-iteration algorithms. Indeed, due to space exhaustion caused by matrix inputs, we were only able to solve instances that were smaller than 9 cells with limited horizon of depth 4 and undiscounted reward (this proof of concept is not reported).

We considered an environment with sensing and processing units that were embedded on a floor. We relied on a cellular automaton model for simulating a grid of units whose communication and synchronization could be easily controlled over time. The discretization of the floor in tiles and the discretization of the automaton in cells are similar. However, the robot's configuration space may be different than the discretization of the floor. Hence, a first perspective is to reduce this coupling by discretizing the cellular automaton to fit the robot's requirements.

We limited the communication range of each unit to a *Von Neumann* neighborhood, which allowed to compute a 1-norm gradient and information layer from which *Voronoi* cells are identified. The obtained roadmap was a set of horizontal, vertical, or diagonal edges, and this is not practical for omni-directional robots. Consequently, a second suggestion is to extend the communication range, and compute other distance transforms.

We qualitatively assessed the thickness of the roadmap and its connectivity on selected environments. It is important to derive quantitative metrics to evaluate these properties automatically on a more extensive set of environments. Moreover, quantitative metrics should also be used to evaluate the average number of updates per cell that is required to either compute a roadmap from scratch or to repair it when subject to disturbances, with different types of asynchronism.

This ends our dissertation.

Firstly, we hope that the studies dedicated to coverage and exploration in Robotics from the point of view Sequential Decision-Making, will help to: first, disambiguate the different versions of coverage and exploration tasks that can be addressed; second, encourage the systematic adoption of coverage and exploration domains with strict assumptions; third, facilitate the use of competitive ratios for evaluating on-line exploration strategies; and fourth, promote the development of reinforcement learning or planning agents that seek optimality.

Secondly, we hope that the studies dedicated to exploration in crowded environments, and navigation in ambient environments will help to: first, motivate the design of human-aware robotic systems where Human-Robot Interaction occupies a central place; and second, encourage studies regarding synergetic computation by both mobile robots and their ubiquitous environment.

Bibliography

[Adamatzky, 1996] Adamatzky, A. (1996). Voronoi-Like Partition of Lattice in Cellular Automata. *Mathematical and Computer Modelling*, 23(4).

[Adamatzky and de Lacy Costello, 2002] Adamatzky, A. and de Lacy Costello, B. (2002). Collision-Free Path Planning in the Belousov-Zhabotinsky Medium assisted by a Cellular Automaton. *Naturwissenschaften*, 89(10).

[Adamatzky and de Lacy Costello, 2003] Adamatzky, A. and de Lacy Costello, B. (2003). Reaction-Diffusion Path Planning in a Hybrid Chemical and Cellular-Automaton Processor. *Chaos, Solitons & Fractals*, 16(5).

[Adamatzky et al., 2002] Adamatzky, A., de Lacy Costello, B., and Ratcliffe, N. (2002). Experimental Reaction–Diffusion Pre-Processor for Shape Recognition. *Physics Letters A*, 297(5).

[Amigoni, 2008] Amigoni, F. (2008). Experimental Evaluation of Some Exploration Strategies for Mobile Robots. In *International Conference on Robotics and Automation*.

[Amigoni and Gallo, 2005] Amigoni, F. and Gallo, A. (2005). A Multi-Objective Exploration Strategy for Mobile Robots. In *Proceedings of the International Conference on Robotics and Automation*.

[Andries, 2015] Andries, M. (2015). *Object and Human Tracking, and Robot Control through a Load Sensing Floor*. Theses, Université de Lorraine.

[Andries and Charpillet, 2013] Andries, M. and Charpillet, F. (2013). Multi-Robot Exploration of Unknown Environments with Identification of Exploration Completion and Post-Exploration Rendez-Vous using Ant Algorithms. In *Proceedings of the International Conference on Intelligent Robots and Systems*.

[Araya-López et al., 2010a] Araya-López, M., Buffet, O., Thomas, V., and Charpillet, F. (2010a). A POMDP Extension with Belief-Dependent Rewards. In *Advances in Neural Information Processing Systems*.

[Araya-López et al., 2010b] Araya-López, M., Thomas, V., Buffet, O., and Charpillet, F. (2010b). A Closer Look at MOMDPs. In *Proceedings of the International Conference on Tools with Artificial Intelligence*.

[Arcelli and Sanniti di Baja, 1985] Arcelli, C. and Sanniti di Baja, G. (1985). A Width-Independent Fast Thinning Algorithm. *Transactions on Pattern Analysis and Machine Intelligence*, (4).

[Arcelli and Sanniti di Baja, 1986] Arcelli, C. and Sanniti di Baja, G. (1986). Computing Voronoi Diagrams in Digital Pictures. *Pattern Recognition Letters*, 4(5).

[Arkin et al., 2003] Arkin, R., Fujita, M., Takagi, T., and Hasegawa, R. (2003). An Ethological and Emotional Basis for Human-Robot Interaction. *Robotics and Autonomous Systems*, 42(3).

[Asoh et al., 2001] Asoh, H., Motomura, Y., Asano, F., Hara, I., Hayamizu, S., Itou, K., Kurita, T., Matsui, T., Vlassis, N., and Bunschoten, R. (2001). Jijo-2: An Office Robot that Communicates and Learns. *Intelligent Systems*, 16(5).

[Bailey and Durrant-Whyte, 2006] Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous Localization and Mapping (SLAM): Part II. *Robotics & Automation Magazine*, 13(3).

[Bajcsy, 1988] Bajcsy, R. (1988). Active Perception. *Proceedings of the IEEE*, 76(8).

[Bar-Noy and Schieber, 1991] Bar-Noy, A. and Schieber, B. (1991). The Canadian Traveller Problem. In *SODA*, volume 91.

[Baronov and Baillieul, 2007] Baronov, D. and Baillieul, J. (2007). Reactive Exploration through Following Isolines in a Potential Field. In *Proceedings of the American Control Conference*.

[Barraquand et al., 1992] Barraquand, J., Langlois, B., and Latombe, J.-C. (1992). Numerical Potential Field Techniques for Robot Path Planning. *Transactions on Systems, Man and Cybernetics*, 22(2).

[Basilico and Amigoni, 2009] Basilico, N. and Amigoni, F. (2009). Exploration Strategies based on Multi-Criteria Decision Making for an Autonomous Mobile Robot. In *Proceedings of the European Conference on Mobile Robots*, pages 259–264.

[Bautin et al., 2012] Bautin, A., Simonin, O., and Charpillet, F. (2012). MinPos : A Novel Frontier Allocation Algorithm for Multi-robot Exploration. In *Proceedings of the International Conference on Intelligent Robotics and Applications*.

[Bellman, 1957] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

[Bengio et al., 2009] Bengio, Y. et al. (2009). Learning Deep Architectures for AI. *Foundations and trends® in Machine Learning*, 2(1).

[Bennewitz et al., 2005] Bennewitz, M., Burgard, W., Cielniak, G., and Thrun, S. (2005). Learning Motion Patterns of People for Compliant Robot Motion. *The International Journal of Robotics Research*, 24(1).

[Bernstein et al., 2002] Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2002). The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of operations research*, 27(4).

[Blei and Kaelbling, 1999] Blei, D. M. and Kaelbling, L. P. (1999). Shortest Paths in a Dynamic Uncertain Domain. In *IJCAI Workshop on Adaptive Spatial Representations of Dynamic Environments*, volume 4, page 2.

[Blum, 1967] Blum, H. (1967). A Transformation for Extracting New Descriptors of Shape. *Models for the perception of speech and visual form*, 19(5).

[Boekhorst, 2002] Boekhorst, F. (2002). Ambient Intelligence, the Next Paradigm for Consumer Electronics: How Will It Affect Silicon? *Digest of Technical Papers International Solid-State Circuits Conference.*, 1.

[Bonet, 2009] Bonet, B. (2009). Deterministic POMDPs Revisited. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

[Bonet and Geffner, 2000] Bonet, B. and Geffner, H. (2000). Planning with Incomplete Information as Heuristic Search in Belief Space. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*.

[Bourgault et al., 2002] Bourgault, F., Makarenko, A. A., Williams, S. B., Grocholsky, B., and Durrant-Whyte, H. F. (2002). Information Based Adaptive Robotic Exploration. In *Proceedings of the International Conference on Intelligent Robots and Systems*, volume 1.

[Browne et al., 2012] Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S., et al. (2012). A Survey of Monte Carlo Tree Search Methods. *Transactions on Computational Intelligence and AI in Games*, 4(1).

[Burgard et al., 1999] Burgard, W., Cremers, A., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. (1999). Experiences with an Interactive Museum Tour-Guide Robot. *Artificial intelligence*, 114(1).

[Burgard et al., 2005] Burgard, W., Moors, M., Stachniss, C., and Schneider, F. (2005). Coordinated Multi-Robot Exploration. *Transactions on Robotics*, 21(3).

[Cassandra, 1998] Cassandra, A. (1998). *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. PhD thesis.

[Cassandra et al., 1994] Cassandra, A., Kaelbling, L. P., and Littman, M. (1994). Acting Optimally in Partially Observable Stochastic Domains. In *AAAI*, volume 94.

[Danielsson, 1980] Danielsson, P.-E. (1980). Euclidean Distance Mapping. *Computer Graphics and image processing*, 14(3).

[de Lacy Costello, 2014] de Lacy Costello, B. (2014). Calculating Voronoi Diagrams using Simple Chemical Reactions. *ArXiv e-prints*.

[de Lacy Costello et al., 2012] de Lacy Costello, B., Jahan, I., and Adamatzky, A. (2012). Sequential Voronoi Diagram Calculations using Simple Chemical Reactions. *ArXiv e-prints*.

[Dechter and Pearl, 1985] Dechter, R. and Pearl, J. (1985). Generalized Best-First Search Strategies and the Optimality of A*. *Journal of the ACM*, 32(3).

[Dijkstra, 1959] Dijkstra, E. (1959). A Note on Two Problems in Connexion with Graphs. *Numerische mathematik*, 1(1).

[Dubois et al., 2011] Dubois, A., Dib, A., and Charpillet, F. (2011). Using HMMs for Discriminating Mobile from Static Objects in a 3d Occupancy Grid. In *Proceedings of the International Conference on Tools with Artificial Intelligence*.

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous Localization and Mapping: Part I. *Robotics & Automation Magazine, IEEE*, 13(2).

[Edelkamp and Schroedl, 2011] Edelkamp, S. and Schroedl, S. (2011). *Heuristic Search: Theory and Applications*.

[Elfes, 1989] Elfes, A. (1989). Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*, 22(6).

[Faigl and Kulich, 2015] Faigl, J. and Kulich, M. (2015). On Benchmarking of Frontier-Based Multi-Robot Exploration Strategies. In *Proceedings of the European Conference on Mobile Robots*.

[Faigl et al., 2012] Faigl, J., Kulich, M., and Přeučil, L. (2012). Goal Assignment using Distance Cost in Multi-Robot Exploration. In *Proceedings of the International Conference on Intelligent Robots and Systems*.

[Felner, 2011] Felner, A. (2011). Position Paper: Dijkstra's Algorithm versus Uniform Cost Search or a Case Against Dijkstra's Algorithm. In *Annual Symposium on Combinatorial Search*.

[Ferber, 1999] Ferber, J. (1999). *Multi-Agent Systems: an Introduction to Distributed Artificial Intelligence*, volume 1.

[Ferranti et al., 2007] Ferranti, E., Trigoni, N., and Levene, M. (2007). Brick&Mortar: an On-Line Multi-Agent Exploration Algorithm. In *Proceedings of the International Conference on Robotics and Automation*.

[Fischler and Barrett, 1980] Fischler, M. and Barrett, P. (1980). An Iconic Transform for Sketch Completion and Shape Abstraction. *Computer Graphics and Image Processing*, 13(4).

[Fried et al., 2013] Fried, D., Shimony, S. E., Benbassat, A., and Wenner, C. (2013). Complexity of Canadian Traveler Problem Variants. *Theoretical Computer Science*, 487:1–16.

[Galceran and Carreras, 2013] Galceran, E. and Carreras, M. (2013). A Survey on Coverage Path Planning for Robotics. *Robotics and Autonomous Systems*, 61(12).

[Garey and Johnson, 2002] Garey, M. and Johnson, D. (2002). *Computers and Intractability*, volume 29.

[Glad et al., 2010] Glad, A., Simonin, O., Buffet, O., and Charpillet, F. (2010). Influence of Different Execution Models on Patrolling Ant Behaviors: from Agents to Robots. In *International Conference on Autonomous Agents and Multiagent Systems*.

[González-Banos and Latombe, 2002] González-Banos, H. H. and Latombe, J.-C. (2002). Navigation Strategies for Exploring Indoor Environments. *The International Journal of Robotics Research*, 21(10-11).

[Goodrich and Schultz, 2007] Goodrich, M. A. and Schultz, A. C. (2007). Human-Robot Interaction: A Survey. *Foundations and Trends in Human-Computer Interaction*, 1(3).

[Hart et al., 1968] Hart, P., Nilsson, N., and Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Transactions on Systems Science and Cybernetics*, 4(2).

[Hart et al., 1972] Hart, P., Nilsson, N., and Raphael, B. (1972). Correction to a Formal Basis for the Heuristic Determination of Minimum Cost Paths. *ACM SIGART Bulletin*, (37).

[Hatem et al., 2015] Hatem, M., Kiesel, S., and Ruml, W. (2015). Recursive Best-First Search with Bounded Overhead. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[Helbing and Molnar, 1995] Helbing, D. and Molnar, P. (1995). Social Force Model for Pedestrian Dynamics. *Physical review E*, 51(5).

[Hoff III et al., 1999] Hoff III, K., Keyser, J., Lin, M., Manocha, D., and Culver, T. (1999). Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH.

[Holz et al., 2010] Holz, D., Basilico, N., Amigoni, F., and Behnke, S. (2010). Evaluating the Efficiency of Frontier-Based Exploration Strategies. In *International Symposium on Robotics (ISR) and German Conference on Robotics (ROBOTIK)*.

[Hornung et al., 2013] Hornung, A., Wurm, K., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An Efficient Probabilistic 3D Mapping Framework based on Octrees. *Autonomous Robots*, 34(3).

[Kaelbling et al., 1998] Kaelbling, L. P., Littman, M., and Cassandra, A. (1998). Planning and Acting in Partially Observable Stochastic Domains. *Artificial intelligence*, 101(1).

[Kaldé et al., 2014a] Kaldé, N., Charpillet, F., and Simonin, O. (2014a). Comparaison de stratégies d'exploration multi-robot classiques et interactives en environnement peuplé. In *Principe de Parcimonie, Journées Francophones sur les Systèmes Multi-Agents*.

[Kaldé et al., 2014b] Kaldé, N., Charpillet, F., and Simonin, O. (2014b). Comparison of Classical and Interactive Multi-Robot Exploration Strategies in Populated Environments. In *Workshop on Multi-Agent Coordination in Robotic Exploration at the European Conference on Artificial Intelligence*.

[Kaldé et al., 2014c] Kaldé, N., Simonin, O., and Charpillet, F. (2014c). Asynchronous Computing of a Discrete Voronoi Diagram on a Cellular Automaton using 1-Norm: Application to Roadmap Extraction. In *Proceedings of the International Conference on Tools with Artificial Intelligence*.

[Kaldé et al., 2014d] Kaldé, N., Simonin, O., and Charpillet, F. (2014d). Discrete Voronoi-like Partition of a Mesh on a Cellular Automaton in Asynchronous Calculus. Technical report, INRIA.

[Kaldé et al., 2015] Kaldé, N., Simonin, O., and Charpillet, F. (2015). Comparison of Classical and Interactive Multi-Robot Exploration Strategies in Populated Environments. *Acta-Polytechnica, Czech-Technical-University*, 55(3).

[Koenig et al., 2004] Koenig, S., Likhachev, M., and Furcy, D. (2004). Lifelong Planning A*. *Artificial Intelligence*, 155(1).

[Koenig and Liu, 2001] Koenig, S. and Liu, Y. (2001). Terrain Coverage with Ant Robots: A Simulation Study. In *Proceedings of the international conference on Autonomous agents*.

[Koenig et al., 2001] Koenig, S., Tovey, C., and Halliburton, W. (2001). Greedy Mapping of Terrain. In *Proceedings of the International Conference on Robotics and Automation*, volume 4.

[Korf, 1985] Korf, R. (1985). Depth-First Iterative-Deepening: An Optimal Admissible Tree Search. *Artificial intelligence*, 27(1).

[Korf, 1990] Korf, R. (1990). Real-Time Heuristic Search. *Artificial intelligence*, 42(2).

[Korf, 1993] Korf, R. (1993). Linear-Space Best-First Search. *Artificial Intelligence*, 62(1).

[Korf et al., 2005] Korf, R., Zhang, W., Thayer, I., and Hohwald, H. (2005). Frontier Search. *Journal of the ACM*, 52(5).

[Korf, 1997] Korf, R. E. (1997). Finding Optimal Solutions to Rubik's Cube using Pattern Databases. In *AAAI/IAAI*.

[Korf and Taylor, 1996] Korf, R. E. and Taylor, L. A. (1996). Finding Optimal Solutions to the Twenty-Four Puzzle. In *Proceedings of the National Conference on Artificial Intelligence*.

[Kosuge and Hirata, 2004] Kosuge, K. and Hirata, Y. (2004). Human-Robot Interaction. In *Proceedings of the International Conference on Robotics and Biomimetics*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet Classification with Deep Convolutional Neural Networks. In *Advances in neural information processing systems*.

[Kullback, 1997] Kullback, S. (1997). *Information Theory and Statistics*.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. *The annals of mathematical statistics*, 22(1).

[Latombe, 2012] Latombe, J.-C. (2012). *Robot Motion Planning*, volume 124. Springer Science & Business Media.

[LaValle, 2006] LaValle, S. M. (2006). *Planning Algorithms*. Cambridge university press.

[Li et al., 2012] Li, A. Q., Amigoni, F., and Basilico, N. (2012). Searching for Optimal Off-Line Exploration Paths in Grid Environments for a Robot with Limited Visibility. In *AAAI*.

[Liao and Huang, 2014] Liao, C.-S. and Huang, Y. (2014). The Covering Canadian Traveller Problem. *Theoretical Computer Science*, 530(Supplement C):80 – 88.

[Littman, 1996] Littman, M. (1996). *Algorithms for Sequential Decision Making*. PhD thesis, Brown University.

[Littman et al., 1995] Littman, M., Dean, T., and Kaelbling, L. P. (1995). On the Complexity of Solving Markov Decision Problems. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

[Liu et al., 2007] Liu, H., Darabi, H., Banerjee, P., and Liu, J. (2007). Survey of Wireless Indoor Positioning Techniques and Systems. *Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6).

[Macedo and Cardoso, 2004] Macedo, L. and Cardoso, A. (2004). Exploration of Unknown Environments with Motivational Agents. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*.

[Maignan and Gruau, 2011] Maignan, L. and Gruau, F. (2011). Convex hulls on cellular spaces: Spatial Computing on Cellular Automata. *Conference on Self-Adaptive and Self-Organizing Systems Workshops*.

[Makarenko et al., 2002] Makarenko, A., Williams, S., Bourgault, F., and Durrant-Whyte, H. (2002). An Experiment in Integrated Exploration. In *International Conference on Intelligent Robots and Systems*.

[Maniatty and Szymanski, 1997] Maniatty, W. and Szymanski, B. (1997). Fine-Grain Discrete Voronoi Diagram Algorithms in L1 and Linf Norms. *Mathematical and Computer Modelling*, 26(4).

[Michalowski et al., 2007] Michalowski, M., Sabanovic, S., and Kozima, H. (2007). A Dancing Robot for Rhythmic Social Interaction. In *Proceedings of the International Conference on Human-Robot Interaction*.

[Mihaylova et al., 2003] Mihaylova, L., Lefebvre, T., Bruyninckx, H., Gadeyne, K., and De Schutter, J. (2003). A Comparison of Decision Making Criteria and Optimization Methods for Active Robotic Sensing. In *Numerical Methods and Applications*.

[Moorehead et al., 2001] Moorehead, S. J., Simmons, R., and Whittaker, W. L. (2001). Autonomous Exploration using Multiple Sources of Information. In *Proceedings of the International Conference on Robotics and Automation*, volume 3.

[Morlok and Gini, 2007] Morlok, R. and Gini, M. (2007). Dispersing Robots in an Unknown Environment. In *Distributed Autonomous Robotic Systems*.

[Nikolova and Karger, 2008] Nikolova, E. and Karger, D. (2008). Route Planning under Uncertainty: The Canadian Traveller Problem. In *AAAI*.

[Okabe et al., 2009] Okabe, A., Boots, B., Sugihara, K., and Chiu, S. N. (2009). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, volume 501.

[Olsen and Goodrich, 2003] Olsen, D. and Goodrich, M. (2003). Metrics for Evaluating Human-Robot Interactions. In *Proceedings of PERMIS*.

[Ong et al., 2010] Ong, S. C., Png, S. W., Hsu, D., and Lee, W. S. (2010). Planning Under Uncertainty for Robotic Tasks with Mixed Observability. *The International Journal of Robotics Research*, 29(8).

[Papadimitriou and Yannakakis, 1991] Papadimitriou, C. and Yannakakis, M. (1991). Shortest Paths without a Map. *Theoretical Computer Science*, 84(1).

[Pemberton and Korf, 1992] Pemberton, J. and Korf, R. E. (1992). *Making Locally Optimal Decisions on Graphs with Cycles*. University of California, Computer Science Department.

[Pineau et al., 2006] Pineau, J., Gordon, G., and Thrun, S. (2006). Anytime Point-Based Approximations for Large POMDPs. *Journal of Artificial Intelligence Research*, 27.

[Pohl, 1970] Pohl, I. (1970). Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3).

[Puterman, 2014] Puterman, M. (2014). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*.

[Quigley et al., 2009] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). ROS: An Open-Source Robot Operating System. In *ICRA workshop on open source software*, volume 3.

[Ramalingam and Reps, 1996] Ramalingam, G. and Reps, T. (1996). An Incremental Algorithm for a Generalization of the Dhortest-Path Problem. *Journal of Algorithms*, 21(2).

[Rios and Chaimowicz, 2010] Rios, L. H. O. and Chaimowicz, L. (2010). A Survey and Classification of A* based Best-First Heuristic Search Algorithms. In *Advances in Artificial Intelligence–SBIA*.

[Rohmer et al., 2013] Rohmer, E., Singh, S., and Freese, M. (2013). V-REP: A Versatile and Scalable Robot Simulation Framework. In *Proceedings of the International Conference on Intelligent Robots and Systems*.

[Rosenfeld and Pfaltz, 1966] Rosenfeld, A. and Pfaltz, J. (1966). Sequential Operations in Digital Picture Processing. *Journal of the ACM*, 13(4).

[Ross et al., 2008] Ross, S., Pineau, J., Paquet, S., and Chaib-Draa, B. (2008). Online Planning Algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32.

[Russell and Norvig, 1995] Russell, S. and Norvig, P. (1995). Artificial Intelligence: Modern Approach.

[Sarkar et al., 1991] Sarkar, U., Chakrabarti, P., Ghose, S., and De Sarkar, S. (1991). Reducing Reexpansions in Iterative-Ddeepening Search by Controlling Cutoff Bounds. *Artificial Intelligence*, 50(2).

[Silver and Veness, 2010] Silver, D. and Veness, J. (2010). Monte-Carlo Planning in Large POMDPs. In *Advances in neural information processing systems*.

[Sim et al., 2004] Sim, R., Dudek, G., and Roy, N. (2004). Online Control Policy Optimization for Minimizing Map Uncertainty during Exploration. In *Proceedings of the International Conference on Robotics and Automation*, volume 2.

[Stachniss, 2009] Stachniss, C. (2009). *Robotic Mapping and Exploration*, volume 55.

[Stachniss and Burgard, 2003] Stachniss, C. and Burgard, W. (2003). Exploring Unknown Environments with Mobile Robots using Coverage Maps. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

[Stachniss et al., 2005] Stachniss, C., Grisetti, G., and Burgard, W. (2005). Information Gain-based Exploration Using Rao-Blackwellized Particle Filters. In *Robotics: Science and Systems*, volume 2.

[Steinfeld et al., 2006] Steinfeld, A., Fong, T., Kaber, D., Lewis, M., Scholtz, J., Schultz, A., and Goodrich, M. (2006). Common Metrics for Human-Robot Interaction. In *Proceedings of the SIGCHI/SIGART Conference on Human-Robot Interaction*.

[Sudha et al., 1999] Sudha, N., Nandi, S., and Sridharan, K. (1999). A Parallel Algorithm to Construct Voronoi Diagram and its VLSI Architecture. In *Proceedings of the International Conference on Robotics and Automation*, volume 3.

[Sutton and Barto, 1998] Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*, volume 1. MIT press Cambridge.

[Svennebring and Koenig, 2004] Svennebring, J. and Koenig, S. (2004). Building Terrain-Covering Ant Robots: A Feasibility Study. *Autonomous Robots*, 16(3).

[Takeda et al., 1997] Takeda, H., Kobayashi, N., Matsubara, Y., and Nishida, T. (1997). Towards Ubiquitous Human-Robot Interaction. In *Proceedings of the Working Notes for IJCAI Workshop on Intelligent Multimodal Systems*.

[Thrun, 2002] Thrun, S. (2002). Probabilistic Robotics. *Communications of the ACM*, 45(3).

[Thrun, 2003] Thrun, S. (2003). Learning Occupancy Grid Maps with Forward Sensor Models. *Autonomous robots*, 15(2).

[Thrun et al., 2001] Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2001). Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*, 128(1-2).

[Tovey and Koenig, 2003] Tovey, C. and Koenig, S. (2003). Improved Analysis of Greedy Mapping. In *Proceedings of the International Conference on Intelligent Robots and Systems*, volume 4.

[Tzionas et al., 1997] Tzionas, P., Thanailakis, A., and Tsalides, P. (1997). Collision-Free Path Planning for a Diamond-Shaped Robot Using Two-Dimensional Cellular Automata. *Transactions on Robotics*, 13(2).

[Tzionas et al., 1994] Tzionas, P., Tsalides, P., and Thanailakis, A. (1994). A New, Cellular Automaton-Based, Nearest Neighbor Pattern Classifier and its VLSI Implementation. *Transactions on Very Large Scale Integration Systems*, 2(3).

[Wang et al., 2008] Wang, K.-H. C., Botea, A., et al. (2008). Fast and Memory-Efficient Multi-Agent Pathfinding. In *ICAPS*.

[Watson et al., 2002] Watson, R. A., Ficici, S. G., and Pollack, J. B. (2002). Embodied Evolution: Distributing an Evolutionary Algorithm in a Population of Robots. *Robotics and Autonomous Systems*, 39(1).

[Wernli, 2012] Wernli, D. (2012). Grid Exploration. Master's thesis, ETH Zürich, Department of Computer Science.

[Yamauchi, 1997] Yamauchi, B. (1997). A Frontier-Based Approach for Autonomous Exploration. In *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation*.

[Yamauchi, 1998] Yamauchi, B. (1998). Frontier-based Exploration using Multiple Robots. In *Proceedings of the International Conference on Autonomous agents*.

[Yamauchi et al., 1998] Yamauchi, B., Schultz, A., and Adams, W. (1998). Mobile Robot Exploration and Map-Building with Continuous Localization. In *Proceedings of the International Conference on Robotics and Automation*, volume 4.

[Yehoshua et al., 2015] Yehoshua, R., Agmon, N., and Kaminka, G. (2015). Frontier-based RTDP: A New Approach to Solving the Robotic Adversarial Coverage Problem. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*.

[Yoshizumi et al., 2000] Yoshizumi, T., Miura, T., and Ishida, T. (2000). A* with Partial Expansion for Large Branching Factor Problems. In *AAAI/IAAI*.

[Zhang and Korf, 1995] Zhang, W. and Korf, R. (1995). Performance of Linear-Space Search Algorithms. *Artificial Intelligence*, 79(2).

[Zlot et al., 2002] Zlot, R., Stentz, A., Dias, M. B., and Thayer, S. (2002). Multi-robot Exploration Controlled by a Market Economy. In *Proceedings of the International Conference on Robotics and Automation*, volume 3.

Appendices

Appendix A

Models of coverage and exploration

A.1 Primitives

The primitives that support coverage and exploration are reported in Primitive 1.

Primitive 1 Primitives to support models of coverage

Input: m a deterministic occupancy map of the environment with values in $\{f\ (free), \neg f\ (occupied)\}$ and dimensions $P = [1..rows] \times [1..cols]$ cells; c a deterministic coverage map with values in $\{c\ (covered), \neg c\ (not\ covered)\}$ and dimensions P.

---

function $isfree(m, p)$
▷ Check if the cell at position p is free in the occupancy map m
$isfree \leftarrow false$
if $p \in P$ **then** $isfree \leftarrow m(p) = f$
return $isfree$

---

function $updatemap(s, a)$
▷ Update the occupancy map s_m according to the environment dynamics
return s_m

---

function $updatepose(s, a)$
▷ Update the robot position s_p according to the occupancy map s_m after applying a
$p \leftarrow s_p$
for $\langle action, \delta_p \rangle \in \{\langle noop, (0,0)\rangle, \langle \Uparrow, (-1,0)\rangle, \langle \Downarrow, (+1,0)\rangle, \langle \Leftarrow, (0,-1)\rangle, \langle \Rightarrow, (0,+1)\rangle\}$ **do**
if $a = action$ and $isfree(s_m, p + \delta_p)$ **then** $p \leftarrow p + \delta_p$
return p

---

function $updatecoverage(s, a)$
▷ Update the coverage map s_c according to the ground truth occupancy map s_m and the robot position s_p reached after applying action a
for $cell \in F(s_p, s_m)$ **do** $s_c(cell) \leftarrow c$
return s_c

---

function $F(p, m)$
▷ Select cells within radius $range$
$F \leftarrow \{p' \in P \mid ||p' - p|| \leq range\}$
return F

A.2 State estimator over beliefs

We detail the items of the Completely Observable (Belief) Contingency Problem ($\text{COCP}_{b^{az}}$) formalism. The translation from the state space to a belief space process where beliefs are represented as set of states is as follows:

$\mathbf{b_0}$, the initial belief is a set of potential initial states. With a known initial state it is the

singleton $\{s_0\}$, otherwise in the worst case it is the entire state space S.

$\mathbf{ApplicableAction_b(b)}$, aggregates the actions available from each state s, in a belief b.

$$ApplicableAction_b(b) = \operatorname*{agg}_{s \in b} ApplicableAction(s)$$

$\mathbf{TransitionModel'_b(b, a)}$, the transition model over updated beliefs is nondeterministic:

$$TransitionModel'_b(b, a) = \{b^{az} \mid \forall z \in Z^a\}.$$

Where Z^a denotes the set of potential observations available from the predicted belief b^a:

$$Z^a = \bigcup_{s \in b} \bigcup_{s' \in TransitionModel^{(\prime)}(s,a)} ObservationModel^{(\prime)}(s, a, s').$$

First, state estimation involves a prediction step that applies action a to all states s in b in order to obtain a belief prediction b^a.

$$b^a = \bigcup_{s \in b} TransitionModel^{(\prime)}(s, a)$$

Second, it involves an update step that accounts for an observation z to obtain an updated belief:

$$b^{az} = \bigcup_{s \in b} \{s' \in TransitionModel^{(\prime)}(s, a) \mid z \in ObservationModel^{(\prime)}(s, a, s')\}$$

$\mathbf{GoalTest_b(b)}$, checks that all states s in a belief b, are goal states. Of course, the state could be a goal state long before the agent realizes it.

$$GoalTest_b(b) = \bigwedge_{s \in b} GoalTest(s)$$

$\mathbf{StepCost'_b(b, a, z)}$, the belief step cost is usually defined over b, a, z by aggregating the underlying state costs:

$$StepCost'_b(b, a, z) = \operatorname*{agg}_{s \in b, s' \in b' = b^{az}} c(s, a, s', z)$$

However, in information gathering problems, actions that reduce the uncertainty regarding the hidden state must be rewarded. In such cases, belief costs can be defined without requiring to look at the underlying states. For instance, the difference in cardinality between an updated belief b^{az} and a starting belief b increases when state uncertainty increases:

$$StepCost'_b(b, a, z) = |b^{az}| - |b|$$

b_0, $ApplicableAction_b$ and $TransitionModel_b$ implicitly define a belief space B, where beliefs are elements of the power set of S.

A.3 State estimator over belief states

We detail the items of the Completely Observable (Belief state) Markov Decision Process ($\text{COMDP}_{b^{az}}$) formalism. The translation from the state space problem to a belief-state space problem where belief states are represented as discrete probability distributions works as follows:

$\mathbf{b_0}$, the initial belief state is a discrete prior distribution over S that encodes uncertainty in the initial state. If the initial state, s_0, is known, b_0 is defined by the indicator function:

$$b_0(s) = \mathbb{1}_{\{s_0\}}(s).$$

Otherwise, in the worst case, b_0 is defined by a discrete uniform distribution over S:

$$b_0(s) = U_S(s).$$

B, the belief-state space is the continuous space representing all discrete probability distributions over S (|S-1|-simplex). A belief state, $b \in B$, thus satisfies:

$$\forall s \in S, 0 \leq b(s) \leq 1,$$
$$\sum_{s \in S} b(s) = 1.$$

$\mathbf{A_b(b)}$, aggregates the actions available from a belief state b, *e.g.*, using the union set operator:

$$A_b(b) = \bigcup_{s \in S, b(s) > 0} A(s).$$

$\mathbf{T'_b(b, a, b')}$, the transition model over updated belief states is stochastic as it defines a probability distribution over updated beliefs b' at $t+1$:

$$\begin{aligned} T_b(b, a, b') &= Pr(B_{t+1} = b' \mid B_t = b, A_t = a) \\ &= \sum_{z \in Z} \Pr(B_{t+1} = b', Z_{t+1} = z \mid B_t = b, A_t = a) \\ &= \sum_{z \in Z} \Pr(b' \mid b, a, z) \Pr(z \mid b, a) \\ &= \sum_{z \in Z} \mathbb{1}_{\{b^{az}\}}(b') \Pr(z \mid b, a). \end{aligned}$$

First, state estimation involves a prediction step that applies action a to all states s in b to obtain a predicted belief state $Pr(B_{t'} = b' \mid B_t = b, A_t = a) = \mathbb{1}_{\{b^a\}}(b')$.

$$\begin{aligned} b^a(s') &= Pr(S_{t'} = s' \mid B_t = b, A_t = a) \\ &= \sum_{s \in S} T(s, a, s') b(s). \end{aligned}$$

Second, it involves an update step that corrects the belief-state prediction according to observation z to obtain a belief-state update $Pr(B_{t+1} = b' \mid B_t = b, A_t = a, Z_{t+1} = z) =$

$\mathbb{1}_{\{b^{az}\}}(b')$. The updated belief state b^{az} is computed as follows:

$$\begin{aligned}
b^{az}(s') &= Pr(S_{t+1} = s' \mid B_t = b, A_t = a, Z_{t+1} = z) \\
&= \frac{\Pr(s', b, a, z)}{\Pr(b, a, z)} \\
&= \frac{\Pr(s', z \mid b, a)}{\Pr(z \mid b, a)} \\
&= \frac{\sum_{s \in S} \Pr(s, s', z \mid b, a)}{\Pr(z \mid b, a)} \\
&= \frac{\sum_{s \in S} \Pr(z \mid s, s', b, a) \Pr(s' \mid s, b, a) \Pr(s \mid b, a)}{\Pr(z \mid b, a)} \\
&\overset{\text{c.i.}}{=} \frac{\sum_{s \in S} \Pr(z \mid s, a, s') \Pr(s' \mid s, a) \Pr(s \mid b)}{\Pr(z \mid b, a)} \\
&= \frac{\sum_{s \in S} O(s, a, s', z) T(s, a, s') b(s)}{\Pr(z \mid b, a)}.
\end{aligned}$$

Finally, the probability of observing z, after applying action a from a belief state b is:

$$\begin{aligned}
Pr(z \mid b, a) &= Pr(Z_{t+1} = z \mid B_t = b, A_t = a) \\
&= \sum_{s' \in S} \Pr(S_{t+1} = s', Z_{t+1} = z \mid B_t = b, A_t = a) \\
&= \sum_{s' \in S} \sum_{s \in S} O(s, a, s', z) T(s, a, s') b(s).
\end{aligned}$$

$\mathbf{R'_b(b, a, z)}$, the belief reward is usually defined over b, a, z by taking the expectation over the underlying world state costs:

$$\begin{aligned}
R'_b(b, a, z) &= \mathbb{E}_{s, s' \in S}[R(S_t = s, A_t = a, S_{t+1} = s', Z_{t+1} = z) \mid B_t = b, A_t = a, Z_{t+1} = z] \\
&= \sum_{s, s' \in S} Pr(s, s' \mid b, a, z) R(s, a, s', z) \\
&= \sum_{s, s' \in S} \frac{Pr(s, s', b, a, z)}{Pr(b, a, z)} R(s, a, s', z) \\
&= \sum_{s, s' \in S} \frac{Pr(s, s', z \mid b, a)}{Pr(z \mid b, a)} R(s, a, s', z) \\
&= \sum_{s, s' \in S} \frac{O(s, a, s', z) T(s, a, s') b(s)}{Pr(z \mid b, a)} R(s, a, s', z).
\end{aligned}$$

In domains where information gathering moves must be rewarded, belief-state rewards can be defined without requiring to look at the underlying state rewards:

$$\begin{aligned}
R'_b(b, a, z) &= D_{KL}(b^{az} || b) \\
&= \sum_{s \in S} b^{az}(s) log(b^{az}(s)) - \sum_{s \in S} b^{az}(s) log(b(s)) \\
&= \sum_{s \in S} b^{az}(s) log(\frac{b^{az}(s)}{b(s)}).
\end{aligned}$$

$D_{KL}(b^{az}||b)$ is the *Kullback-Leibler* divergence from b^{az} to b [Kullback and Leibler, 1951, Kullback, 1997]. It measures the difference between the prior belief state b and the posterior belief state b^{az}. It is nonsymmetric, nonnegative, and for discrete distributions it is defined only if $b(s) = 0$ implies $b^{az}(s) = 0$ for any state $s \in S$. Moreover, the term in the sum is null when $b^{az}(s)$ tends toward 0. This measure is null when the two distributions are equal, otherwise it quantifies the information gained from the belief state b by applying action a and receiving the measurement z afterwards. Such belief-dependent rewards are useful to reward reduced state uncertainty associated to potential histories of actions and measurements.

A.4 Contingency and aliasing over beliefs

Belief prediction and update are laid out for a known then unknown initial state.

When the initial state s_0 is known and a deterministic state-transition model $TransitionModel$ is used: perceptual aliasing and contingency do not occur. Indeed, any course of state-action outcomes starting from the known initial state is fully determined by repeated applications of this state-transition model before execution. Afterwards, the sequence of actions is sufficient to resolve the state during execution. Hence, the Completely Observable (State) Search Problem (COSP) formalism is sufficient as there is no state uncertainty nor contingency.

When the initial state s_0 is known and a nondeterministic state-transition model $TransitionModel'$ is used: perceptual aliasing or contingency can occur. Indeed, some courses of state-action outcomes starting from the known initial state may not be fully determined before execution. For instance, the state-transition model predicts a belief b_0^a from the initial belief b_0 after taking any action a; the set of potential measurements from b_0^a is given by Z_1^a; and the state-observation model updates a belief b_1^{az} from the predicted belief b_0^a after receiving any potential measurement z, are as follows:

$$
\begin{aligned}
b_0 &= \{s_0\} \\
b_0^a &= TransitionModel'(s_0, a) \\
Z_1^a &= \bigcup_{s' \in b_0^a} ObservationModel^{(')}(s_0, a, s') \\
b_1^{az} &= \{s' \in b_0^a \mid z \in ObservationModel^{(')}(s_0, a, s')\}
\end{aligned}
$$

A contingency occurs when there exists at least two distinct potential measurements from any predicted belief that lead to two distinct updated beliefs ($|Z_1^a| > 1 \wedge \exists z', z'' \in Z_1^a \; s.t. \; b_1^{az'} \neq b_1^{az''}$) with a deterministic or nondeterministic state-observation model. Perceptual aliasing occurs when there is at least one non-singleton updated belief ($Z_1^a \neq \emptyset \wedge \exists z \in Z_1^a \; s.t. \; |b_1^{az}| > 1$). The agent can resolve such contingencies among updated beliefs only during execution and must maintain a belief to cope with perceptual aliasing during execution.

When the initial state s_0 is unknown and a deterministic or nondeterministic state-transition model $TransitionModel^{(')}$ is used: state uncertainty is immediate, and perceptual aliasing or contingency eventually occurs. A similar reasoning to the previous case applies to explain them. The initial belief b_0; the predicted belief b_0^a; the potential percepts Z_1^a; and the updated belief

b_1^{az} are now given by:

$$
\begin{aligned}
b_0 &= S \\
b_0^a &= \bigcup_{s \in b_0} TransitionModel^{(\prime)}(s,a) \\
Z_1^a &= \bigcup_{s \in b_0} \bigcup_{s' \in TransitionModel^{(\prime)}(s,a)} ObservationModel(s,a,s') \\
b_1^{az} &= \bigcup_{s \in b_0} \{s' \in TransitionModel^{(\prime)}(s,a) \mid z \in ObservationModel(s,a,s')\}
\end{aligned}
$$

To summarize, with a deterministic or nondeterministic sensing model, not being able to identify the initial state or using a nondeterministic state-transition model eventually leads to contingencies over the belief space.

A.5 Contingency and aliasing over belief states

Belief-state prediction and update are laid out for a known then unknown initial state.

When the initial state s_0 is known and a stochastic state-transition model is used: perceptual aliasing and contingencies can occur. For instance, the state-transition model predicts a belief state b_0^a, which quantifies any state-outcome s' as $b_0^a(s')$, for any action a taken from the prior b_0; the set of potential measurements available from any state s' whose $b_0^a(s')$ is non-null is given by Z_1^a; and the state-observation model updates a posterior belief state b_1^{az}, which quantifies any state-outcome s' as $b_1^{az}(s')$, by updating b_0^a for any measurement z. A contingency occurs as the real posterior is only identified at runtime ($|Z_1^a| > 1 \wedge \exists z', z'' \in Z_1^a \ s.t. \ b_1^{az'} \neq b_1^{az''}$). Perceptual aliasing occurs when an updated belief state b_1^{az} is not an indicator function over S ($\nexists s' \in S \ s.t. \ b_1^{az}(s') = 1$). The agent is able to resolve the contingencies among updated beliefs only during execution, and has to maintain its belief state to cope with perceptual aliasing.

$$
\begin{aligned}
b_0(s) &= \mathbb{1}_{\{s_0\}}(s) \\
b_0^a(s') &= Pr(S_1 = s' \mid S_0 = s_0, A_0 = a) = T(s_0, a, s') \\
Z_1^a &= \bigcup_{s' \in S, b_0^a(s') > 0} \{z \in Z \mid O(s,a,s',z) > 0\} \\
b_1^{az}(s') &= Pr(S_1 = s' \mid S_0 = s_0, A_0 = a, Z_1 = z) \\
&= \frac{Pr(z \mid s_0, a, s') Pr(s' \mid s_0, a)}{Pr(z \mid s_0, a)} \\
&= \frac{O(s_0, a, s', z) T(s_0, a, s')}{Pr(z \mid s_0, a)},
\end{aligned}
$$

When the initial state s_0 is unknown and a stochastic transition model is used: state uncertainty is immediate, perceptual aliasing and contingencies can occur. A similar kind of reasoning applies to explain them. The initial belief state b_0; the predicted belief state b_0^a; the potential

measurements Z_1^a; and the posterior belief state b_1^{az} are now given by:

$$
\begin{aligned}
b_0(s) &= U_S(s) \\
b_0^a(s') &= Pr(S_1 = s' \mid B_0 = b_0, A_0 = a) = \sum_{s \in S} T(s, a, s') b_0(s) \\
Z_1^a &= \bigcup_{s \in S, b_0(s) > 0} \bigcup_{s' \in S, T(s,a,s') > 0} \{z \in Z \mid O(s, a, s', z) > 0\} \\
b_1^{az}(s') &= Pr(S_1 = s' \mid B_0 = b_0, A_0 = a, Z_1 = z) \\
&= \frac{\sum_{s \in S} Pr(z \mid s', s, b_0, a) Pr(s' \mid s, b_0, a) Pr(s \mid b_0, a)}{Pr(z \mid b_0, a)} \\
&= \frac{\sum_{s \in S} O(s, a, s', z) T(s, a, s') b_0(s)}{Pr(z \mid b_0, a)}
\end{aligned}
$$

To summarize, as expected from a quantified nondeterministic model, contingencies and state uncertainty appear from the stochastic state-observation model.

A.6 Summary of Sequential Decision-Making formalisms

Table A.1 summarizes the different formalisms of single-agent Sequential Decision-Making.

			Deterministic control	Nondeterministic control: Unquantified	Nondeterministic control: Quantified
Observable state	Deterministic sensing		ODP (COSP)	ONP (COCP)	OSP (COMDP)
Unobservable state			UDP (COSP/$_b$)	UNP (COSP$_{b^a}$)	USP (CODMDP$_{b^a}$)
Partially observable			PDP (COSP/COCP/$_b$)	PNP (COCP$_{b^{az}}$)	PSP (COMDP$_{b^{az}}$)
	Nondeterministic sensing	Unquantified	PDP (COSP/COCP/$_b$)	PNP (COCP$_{b^{az}}$)	PSP (COMDP$_{b^{az}}$)
		Quantified	PSP (COMDP$_{b^{az}}$)	PSP (COMDP$_{b^{az}}$)	PSP (COMDP$_{b^{az}}$)

Table A.1: Sequential Decision-Making formalisms classification according to state-observability, control and sensing types, and initial state knowledge. (SP: Search Problem, CP: Contingency Problem, MDP: Markov Decision Process, $_b$: over beliefs)

Appendix B

Off-line planning

B.1 Best-First Search

Iterative Best-First Search (I-BFS) works as follows. Each node, $n = (s(n), g(n), p(n))$, contains a state $s(n)$, a source distance estimate $g(n)$, and a parent pointer $p(n)$. Additionally, a priority key $f(n)$ can be computed from the aforementioned elements.

1. Closed is empty and Open contains the source $n_0 = (s_0, 0, none)$ with a priority $f(n_0) = g(n_0) = 0$. Then, steps (2-6) are executed until Open is empty in which case it exits with failure (no solution):
2. Select the minimum f-key node, $n = (s, g, p)$, from Open.
3. Move n from Open to Closed.
4. If s is a sink, exit with success (a solution path is given by the parent pointers from the sink).
5. Expand n by generating its successors $Succ(n)$.
6. For each successor $n' = (s', g', n) \in Succ(n)$,
   (a) if s' is neither in Closed nor Open: insert n' into Open with a priority $f' = g'$,
   (b) otherwise s' is encapsulated in a node n'' from Closed or Open: if a better path through s' is found ($g' < g''$), update n'''s p-pointer and g-value to n and g' respectively; moreover, if in Closed, move n'' from Closed to Open.

Frontier Best-First Search (F-BFS) works as follows. Each node, $n = (s(n), g(n), o(n))$, maintains an s-state and a g-value as in I-BFS, but replaces the p-pointer by an o-operator, $o : Neigh(n) \rightarrow \{1, 0\}$. This operator indicates if a neighbor, $n' \in Neigh(n)$, has already been generated (1) or not (0). Additionally, a priority key $f(n)$ can be computed from the aforementioned elements.

1. Initially, Open contains the source node, $n_0 = (s_0, 0, o_0)$, with all operators unused.
   Then, it iteratively executes steps (2-6) as in I-BFS, by discarding instructions related to Closed, until Open is empty or contains only infinite cost nodes, in which case it exits with failure.
   Step 5 is updated: whenever a node, n, is selected for expansion, a set, $Succ$, of successors, $n' : (s', g', o'(n) = 1)$, is generated along with a set, $Pred$, of dummy predecessors, $n' :$

$(s', \infty, o'(n) = 1)$, with infinite cost (when in directed graphs).
Step 6 is updated: whenever the state of a newly generated node, n', is already present in Open via node, n'': o'' is updated as the bitwise or, $\vee_b$, of o'' and o'.

Recursive Best-First Search (R-BFS) works as follows. Each node, $n = (s(n), g(n), F(n))$, maintains an s-state, a static g-value as in I-BFS, but it additionally has a stored F-key which corresponds to the f-key of the best frontier node, n', ever generated below n (initially $F(n) = f(n)$). This stored F-value is used for prioritizing recursive calls on most promising subgraphs whose frontiers' priority does not exceed local upper bounds. This procedure describes a recursive call upon a subroot n with a local upper bound B to explore the subgraph rooted at n.

1. Initial call on the source $n_0 = (s_0, 0, 0)$ with a local upper bound $B = \infty$.
2. If n's f-key exceeds B, the call ends with a cutoff and returns the g-value.
3. If n is a sink, the call ends with success and reconstructs the solution path by unwinding the stack of calls.
4. Successors, $n' = (s', g', F')$ of n, are generated. If they have been previously generated $(f(n) < F(n))$, their F-keys are overwritten with $\max(F(n), f(n'))$.
5. The successor, n_1, with the best F-key is iteratively selected for a recursive call with an upper bound, $B' = \min(B, F(n_2))$, where n_2 is the successor with the second best F-key. This is done to update n_1's stored key, until every successor F-key exceeds B, in which case the call returns the minimum F-key among the successors.
The recursion is on n_1 because it is the root of the subgraph containing the frontier node with the best f-key, and B' is set to cutoff the recursive call as soon as the most promising frontier node is now in the subgraph below n_2.

Iterative Best-First Search (I-BFS), Frontier Best-First Search (F-BFS), and Recursive Best-First Search (R-BFS) are reported in Algorithm 3.

B.2 Depth-First Search

Recursive DFS (R-DFS) works as follows. Any node, $n = (s(n), g(n))$, contains a state $s(n)$ and a source cost estimate $g(n)$. More generally, let us reintroduce a node value $f(n)$.

1. Initial call on the source, n_0.
2. If the current trajectory end node, n, is a sink, search ends with success and the current trajectory is a solution path.
3. Otherwise, the current end node successors are generated.
4. The trajectory through, n, is deepened by iteratively expanding each successor by a recursive call.
5. The trajectory through, n, is shortened by backtracking to its predecessor when no path through one successor led to a solution.

Depth-First Search (DFS), Depth-Limited Search (DLS), Iterative-Deepening Depth-First Search (ID-DFS), and Branch-and-Bound Depth-First Search (BnB-DFS) are given in Algorithm 4.

Algorithm 3 Off-line search (Best-First)

Input: G : Completely Observable (State) Search Problem OR-Graph (COSP OR-Graph)

---

procedure I-BFS($U \leftarrow \{source\}$)
  MIN Queue $Open$; $push(Open, U, f(U))$; $Closed \leftarrow \emptyset$
  while $\neg empty(Open)$ **do**
    $n : (s, g, p(n)) \leftarrow top(Open)$; $move(n, Open, Closed)$
    if $goal(s(n))$ **then return** success, reconstruct path (solution found)
    for $n' \in Succ(n)$ **do**
      if $\nexists n'' \in Open \cup Closed$ *s.t.* $s(n'') = s(n')$ **then** $push(Open, n', f(n'))$
      else if $g(n') < g(n'')$ **then**
        $g, p(n'') \leftarrow g, p(n')$; $adjust(Open\ or\ Closed, n'', f(n''))$
        if $n' \in Closed$ **then** $move(n', Closed, Open)$
  return failure (no solution)

---

procedure F-BFS($U \leftarrow \{source\}$)
  MIN Queue $Open$; $push(Open, U, f(U))$
  while $\neg empty(Open) \wedge \exists n \in Open$ *s.t.* $f(n) \neq \infty$ **do**
    $n : (s, g, o(n)) \leftarrow pop(Open)$
    if $goal(s(n))$ **then return** success (goal reached)
    ▷ successors n' ($o(n')[n] = 1$) generated with legal unused operators
    ▷ predecessors n' ($o(n')[n] = 1, f(n') = \infty$) generated with unused operators from $Open$
    for $n' \in Succ(n) \cup Pred(n)$ **do**
      if $\nexists n'' \in Open$ *s.t.* $s(n'') = s(n')$ **then** $push(Open, n', f(n'))$
      else $o(n'') \leftarrow o(n') \vee_b o(n'')$
        if $g(n') < g(n'')$ **then** $g(n'') \leftarrow g(n')$; $adjust(Open, n'', f(n''))$
  return failure (no solution)

---

procedure R-BFS($n \leftarrow source, B \leftarrow \infty$)
  if $f(n) > B$ **then return** $f(n)$
  if $goal(s(n))$ **then return** $success(n)$ and unwind path (solution found)
  for $n' : (s', g(n'), F(n') = f(n')) \in Succ(n)$ **do**
    if $f(n) < F(n)$ **then** $F(n') \leftarrow \max(F(n), f(n'))$
  if $|Succ(n)| = 0$ **then return** ∞
  if $|Succ(n)| = 1$ **then** $insert(Succ, (-, -, \infty))$
  ▷ select most F-promising nodes n_1, n_2
  $n_{\{1,2\}} \leftarrow \{1^{\text{st}}, 2^{\text{nd}}\} \arg\min_{n' \in Succ(n)} F(n')$
  while $F(n_1) \leq B$ **do**
    $res \leftarrow$ R-BFS($n_1, \min(B, F(n_2))$)
    if $res = success(\{n_j ...\})$ **then return** $success(\{n, n_j ...\})$
    ▷ update most F-promising nodes n_1, n_2
    $F(n_1) \leftarrow res$; $n_{\{1,2\}} \leftarrow \{1^{\text{st}}, 2^{\text{nd}}\} \arg\min_{n' \in Succ(n)} F(n')$
  return $F(n_1)$

Algorithm 4 Off-line search (Depth-First)

Input: G : COSP OR-Graph

---

procedure DFS/CFS($n \leftarrow source$)
▷ Recursive version
if $goal(n)$ **then return** $success(n)$ and unwind path (solution found)
generate, and f-sort successor nodes breaking ties in favor of goal nodes $Succ$
for $n_i \in Succ$ **do** $result \leftarrow DFS/CFS(n_i)$
if $result = success(\{n_j, ...\})$ **then return** $success(\{n, n_j, ...\})$
return $failure$

---

procedure DLS/CLS($n \leftarrow source, limit \leftarrow \infty$)
▷ Recursive version
if $goal(n) \wedge f(n) \leq limit$ **then return** $success(n)$, unwind path (solution found)
if $f(n) > limit$ **then return** $cutoff(f(n))$
$cutstatus \leftarrow false$; $nextcutvalue \leftarrow \infty$
generate, and f-sort successor nodes breaking ties in favor of goal nodes $Succ$
for $n_i \in Succ$ **do** $result \leftarrow DLS/CLS(n_i, limit)$
if $result = success(\{n_j, ...\})$ **then return** $success(\{n, n_j, ...\})$
if $result = cutoff(rescutvalue)$ **then**
$cutstatus \leftarrow true$; $nextcutvalue \leftarrow \min(resnextcutvalue, nextcutvalue)$
if $cutstatus$ **then return** $cutoff(nextcutvalue)$
else return $failure$

---

procedure ID-DFS($n \leftarrow source, lower \leftarrow f(n)$)
▷ Increasing a lower bound on the solution cost
while true **do** $result \leftarrow DLS/CLS$
if $result = cutoff(nextcutvalue)$ **then** $lower \leftarrow nextcutvalue$
else return $result$

---

procedure BnB-DFS($n \leftarrow source, upper \leftarrow \infty$)
▷ Recursive version
▷ Decreasing an upper bound on the solution cost
if $goal(n) \wedge f(n) < upper$ **then return** $success(n, f(n))$, unwind path (solution found)
if $f(n) \geq upper$ **then return** $cutoff$
generate, and f-sort successor nodes breaking ties in favor of goal nodes $Succ$
$search \leftarrow (failure, upper)$
for $n_i \in Succ$ **do** $result \leftarrow BnB - DFS(n_i, upper)$
if $result = success(\{n_j, ..., n_k\}, f(n_k))$ **then**
$upper \leftarrow f(n_k)$; $search \leftarrow success(\{n, n_j, .., n_k\}, upper)$
return $search$

B.3 Summary of off-line search

Tables B.1 and B.2 summarize the mentioned algorithms at the level of their node expansion strategy, node evaluation function, solution optimality, and time-space complexity (uninformed). We reproduced the complexity results from [Russell and Norvig, 1995] and from the original papers when necessary.

Algorithm	Selection for expansion	$g-$Evaluation	$g-$Optimality	Time/Space complexity
Best-First Search Framework				
Iterative Best-First Search (I-BFS) [Dechter and Pearl, 1985]	most g-promising Open node	g	Yes*	$O(b^{C^*/c})/O(b^{C^*/c})$
Iterative Breadth-First search (I-BreadthFS)	–	depth	–	$O(b^d)/O(b^d)$
Iterative Uniform-Cost Search/Dijkstra's Algorithm (I-UCS) [Felner, 2011, Dijkstra, 1959]	–	cost	–	$O(b^{C^*/c})/O(b^{C^*/c})$
Frontier Best-First Search (F-BFS) [Korf et al., 2005]	most g-promising Open node	g	Yes*	
Frontier Breadth-First search (F-BreadthFS)	–	depth	–	
Frontier Uniform-Cost Search (F-UCS)	–	cost	–	
Recursive Best-First Search (R-BFS) [Korf, 1993]	most G-promising subtree root	g	Yes*	$O(b^{C^*/c})/O(C^*/c)$
Recursive Breadth-First search (R-BreadthFS)	–	depth	–	$O(b^d)/O(d)$
Recursive Uniform-Cost Search (R-UCS)	–	cost	–	$O(b^{C^*/c})/O(C^*/c)$
Depth-First Search Framework				
Depth-First Search (I-DFS,R-DFS)	trajectory end-node successor	g	No	$O(b^{C^*/c})/O(C^*/c)$
Depth-Limited (DL)	s.t. $g(succ) \leq g\text{-}limit$	g	Yes*	$O(b^{C^*/c})/O(C^*/c)$
Depth-Limited Search (DLS)	–	depth	–	$O(b^d)/O(d)$
Cost-Limited Search (CLS)	–	cost	–	$O(b^{C^*/c})/O(C^*/c)$
Iterative-Deepening (ID) [Korf, 1985]	s.t. $g(succ) \leq \Uparrow g\text{-}lowerbound$	g	Yes*	$O(b^{C^*/c})/O(C^*/c)$
Iterative-Deepening Depth-First Search (ID-DFS)	–	depth	–	$O(b^d)/O(d)$
Iterative-Deepening Cost-First Search (ID-CFS)	–	cost	–	$O(b^{C^*/c})/O(C^*/c)$
Branch-and-Bound (BnB) [Zhang and Korf, 1995]	s.t. $g(succ) <\Downarrow g\text{-}upperbound$	g	Yes*	$O(b^{C^*/c})/O(C^*/c)$
Branch-and-Bound Depth-First Search (BnB-DFS)	–	depth	–	$O(b^d)/O(d)$
Branch-and-Bound Cost-First Search (BnB-CFS)	–	cost	–	$O(b^{C^*/c})/O(C^*/c)$

Table B.1: Complete uninformed search algorithms, d depth, b branching factor, C^* cost of the optimal solution path, c minimum cost of an edge, $K > 0$ exponentially deepening bound factor, $\Uparrow$ / $\Downarrow$: increasing,decreasing

Algorithm	Selection for expansion	$f-$Evaluation	$g-$Optimality
Best-First Search Framework			
Iterative Best-First Search (I-BFS) [Dechter and Pearl, 1985]	most f-promising Open node	agg(g,h)	Yes*
Iterative A* (I-BFS(A*)) [Hart et al., 1968, Hart et al., 1972]	–	$g+h$	Yes*
Iterative Greedy (I-BFS(Greedy))	–	h	No
Iterative Weighted A* (I-BFS(wA*)) [Pohl, 1970]	–	$g+wh$	Yes*$_\epsilon$
Iterative Partial Expansion A* (I-BFS(PEA*)) [Yoshizumi et al., 2000]	generate $Succ$/store $Succ^{=}/F$-promising	$g+h$	Yes*
Frontier Best-First Search (F-BFS) [Korf et al., 2005]	most f-promising Open node	agg(g,h)	Yes*
Recursive Best-First Search (R-BFS) [Korf, 1993]	most F-promising subtree root	agg(g,h)	Yes*
Constrained-Re-expansion R-BFS (R-BFS$_{CR}$) [Hatem et al., 2015]			
Depth-First Search Framework			
Depth-First Search (I-DFS,R-DFS)	trajectory end-node successor	agg(g,h)	No
Iterative-Deepening (ID)	s.t. $f(succ) \leq\Uparrow f\text{-}lowerbound$	agg(g,h)	Yes*
Iterative-Deepening A Star (ID-DFS(A*)) [Korf, 1985]	–	$g+h$	Yes*
Iterative-Deepening Weighted A* (ID-DFS(wA*))	–	$g+wh$	Yes*$_\epsilon$
Iterative-Deepening Partial Expansion A* (ID-DFS(PEA*)) [Yoshizumi et al., 2000]	s.t. $f(succ) \leq\Uparrow F(parent)+C$	$g+h$	Yes*
Constrained-Re-expansion ID-DFS(A*) (ID-DFS(A*)$_{CR}$) [Sarkar et al., 1991]			
Branch-and-Bound Depth-First Search (BnB-DFS) [Zhang and Korf, 1995]	s.t. $f(succ) <\Downarrow f\text{-}upperbound$	agg(g,h)	Yes*

Table B.2: Complete informed search algorithms

B.4 Raw data from the benchmark of optimal off-line search

We present the raw data in Figure B.1, where each point represents the memory and time resources consumed by an optimal solver using a heuristic on an instance of the coverage task, until either a normal or a resource-out termination.

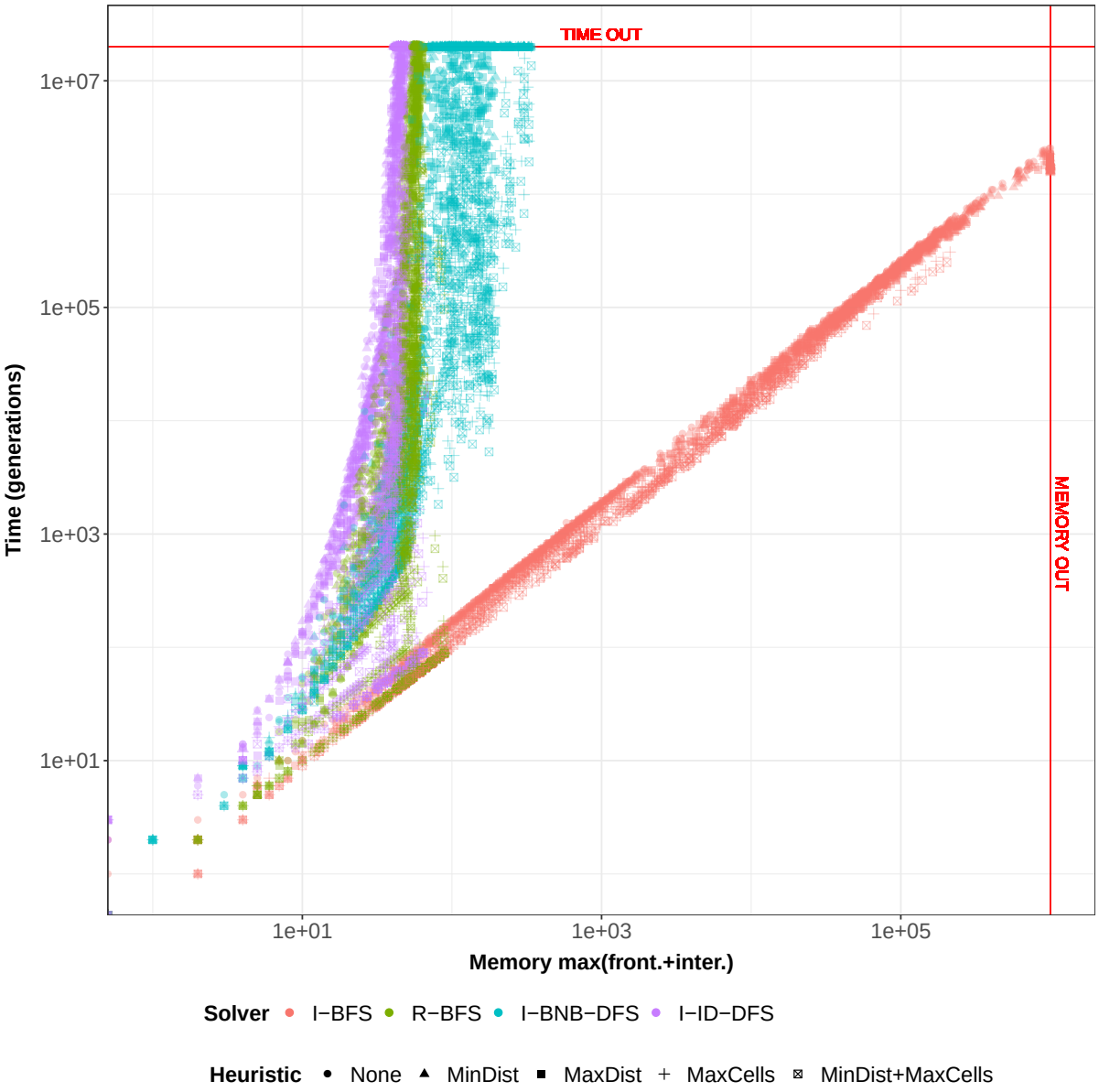


Figure B.1: Time vs memory consumption per algorithm and heuristic for each problem instance.

One such experiment is shown in Figure B.2, which illustrates the graph output by an Iterative A* (I-BFS(A*)) algorithm using the Maximal coverable cells (MaxCells) heuristic, for a 3 by 3 map where the robot starts in the middle of the top row. The metrics defined above are monitored, *e.g.*, the solution cost corresponds to the optimal path cost (in this case an optimal

path is (⇐⇓⇓⇒⇒⇑⇐⇑⇒) of length 9); the time consumption corresponds to the number of nodes that were generated until termination; and the memory consumption corresponds to the maximum size of the frontier plus interior sets over the entire run of the algorithm.

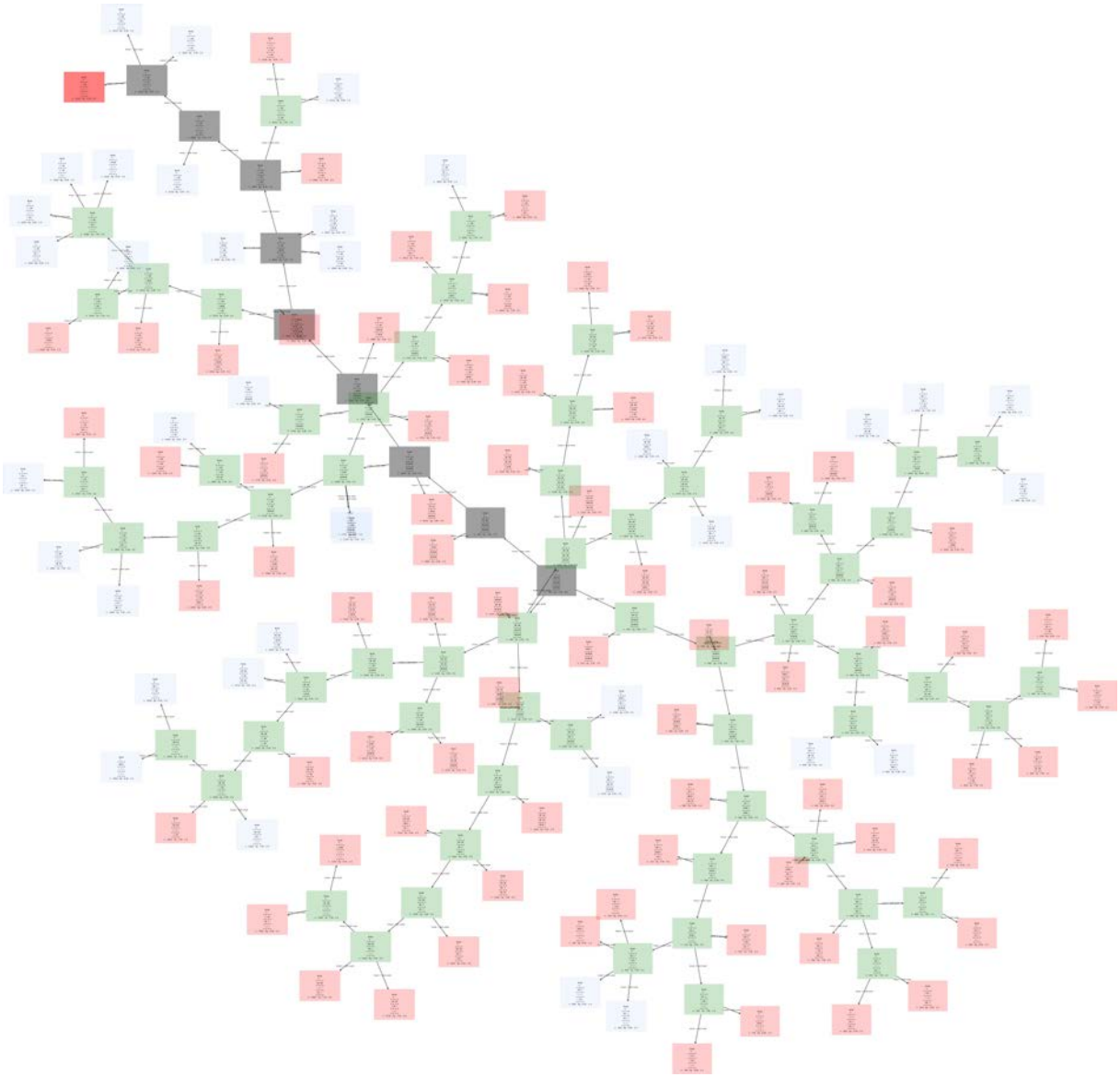


Figure B.2: A sample data point: the graph generated by a run of Iterative A* (I-BFS(A*)) with the Maximal coverable cells (MaxCells) heuristic on a 3×3 empty map where the robot starts in the middle of the top row. Frontier nodes are drawn in red, interior nodes in green, and redundant nodes in blue.

Appendix C

Résumé étendu

C.1 Contexte : couverture, exploration et prise de décision

Cette dissertation est une étude des tâches de couverture et d'exploration par des agents artificiels. Ces tâches sont présentées comme des processus de Prise de Décision Séquentielle à travers le prisme de l'Intelligence Artificielle. Leurs solutions pourraient à terme être déployées sur le terrain en Robotique. Les application robotiques envisagées incluent l'évitement d'obstacles, la patrouille et le sauvetage.

En premier lieu, nous présentons le contexte de ce manuscrit.

Préambule

D'une part, la couverture est une tâche dans laquelle des robots doivent physiquement visiter un environnement *a priori* **connu** pour l'observer dans les moindres détails ; tandis que l'exploration considère que l'environnement est *a priori* **inconnu** et vise à en construire une carte. Il existe de nombreuses applications mettant en œuvre de la couverture ou de l'exploration en Robotique. En 2003 et en 2012, les robots *Spirit*, *Opportunity* et *Curiosity* de la *NASA* ont été envoyés sur Mars pour recueillir des échantillons de roche afin d'étudier la géologie. De 2012 à 2015, le *Robotics Challenge* de la *DARPA* s'est focalisé sur le sauvetage de blessés après des catastrophes naturelles et le démantèlement de réacteurs énergétiques après des catastrophes nucléaires. Depuis 2002, on peut également citer le succès commercial des aspirateurs *Roomba* de *iRobot* pour nettoyer la maison. Dans notre contexte, nous pourrions considérer que l'exploration a lieu la première fois que le nettoyage est effectué (pendant que le robot construit une carte) et que la couverture a lieu par la suite (en se basant sur la carte).

D'autre part, la Prise de Décision Séquentielle (PDS) est un cadre d'étude proposé en Intelligence Artificielle et qui fournit des formalismes pour décrire des mondes dans lesquels des agents interagissent avec leur environnement pour accomplir un objectif. Pendant ces interactions, les agents sont confrontés à une série de choix. Ainsi, la PDS fournit également des planificateurs, pour que les agents prennent des décisions rationnelles afin d'accomplir leur objectif. Il existe de nombreuses applications de la PDS dans le jeu. En 1997, *Deeper Blue* d'*IBM* bat Garry Kasparov, le Champion du Monde d'Échecs. En 2016, *AlphaGo* de *Deepmind* bat Lee Sedol, le Champion du Monde de Go. En 2017, *DeepStack* du *CPRG* de l'Université d'Alberta rivalise avec des joueurs professionnels de poker dans la variante du *Texas hold'em heads-up no-limit*. Ces applications démontrent que le cadre de la PDS peut atteindre et même aller au delà des

performances humaines dans certains jeux. En Robotique, une application courante est la planification de mouvement pour la navigation de véhicules autonomes afin de leur permettre d'éviter les obstacles. Dans notre contexte, l'évitement d'obstacle est obligatoire pour naviguer de manière sûre mais la planification de mouvement n'est pas l'aspect central de la prise de décision. En effet, nous allons traduire nos tâches de couverture et d'exploration comme des jeux dans lesquels les trajectoires planifiées sont les coups disponibles au joueur pour collecter de l'information. La question centrale qui va nous intéresser est la suivante : Où se rendre ensuite ?

Nous avons introduit les concepts principaux mis en œuvre dans ce manuscrit. Notre objective est de concevoir et d'évaluer des agents qui planifient des décisions afin de résoudre des tâches de couverture ou d'exploration robotique. A l'heure actuelle, de nombreuses applications d'exploration en Robotique s'inscrivent dans le paradigme frontière [Yamauchi, 1998, Koenig et al., 2001, Tovey and Koenig, 2003]. Dans ce paradigme, des agents prennent des décisions de manière gloutonne en se basant sur leur connaissance actuelle de l'environnement. Pour collecter l'information, les agents doivent se localiser et construire une carte qui accumule leur observations locales. En se basant sur cette carte, l'agent identifie les zones qui restent inexplorées et se dirigent vers elles. Ceci nécessite l'utilisation d'un algorithme de planification de trajectoire pour calculer le plus court chemin pour rejoindre ces zones inexplorées ; son temps de calcul dépend de la longueur du chemin. L'utilisation d'une flottille d'agents frontières est fiable car il n'y a pas de point individuel de défaillance dans l'équipe.

Ce paradigme s'applique à l'exploration mais peut également s'appliquer à la couverture, bien que des techniques dédiées et plus efficaces soient préférées dans ce cas. Ses bonnes propriétés le rendent applicable à des environnements simulés ou réels dans certaines conditions. Cependant, les agents frontières ne sont pas facilement extensibles à de l'information à priori concernant l'environnement. Par exemple, si nous avions certains indices concernant les dimensions de l'environnement, *e.g.*, de 20 à 40 m^2, ou la densité des meubles, *e.g.*, de 10% à 30% ; il n'est pas évident de définir une extension à ce paradigme pour prendre en compte cette information complémentaire et améliorer le comportement d'exploration.

Ainsi, une première question est : **Existe-t-il un paradigme alternatif qui ne place pas d'hypothèse restrictive concernant l'information utilisée par les agents ?**

De plus, les solutions fournies par un agent de frontière sont simplement satisfaisantes. Un marcheur aléatoire peut également fournir des solutions satisfaisantes, mais en général ce type de comportement est hautement sous-optimal. En effet, nous nous attendons à ce qu'un agent fourmi guidé par des phéromones ait un meilleur comportement qu'un marcheur aléatoire [Koenig and Liu, 2001, Svennebring and Koenig, 2004]. De même, nous nous attendons à ce qu'un agent frontière ait un comportement encore meilleur que celui de l'agent fourmi. De manière générale, ces agents peuvent fournir de bonnes solutions dans des conditions données, et nous pouvons même avoir un apriori concernant leurs performances respectives mais ces solutions n'informent pas vraiment à propos du meilleur comportement possible.

Ainsi, une seconde question est : **Peut-on concevoir des agents qui visent à atteindre un comportement optimal dans une tâche de couverture ou d'exploration ?**

Jusqu'à présent, la discussion a été relativement informelle. Nous avons introduit les premiers

concepts qui apparaissent dans cette thèse et les premières questions qui motivent notre étude. À présent, nous souhaitons transmettre une image plus précise de ce qu'est la couverture et l'exploration en les situant dans la Robotique et la Théorie des Graphes. La première sous-section situe notre étude en Robotique, identifie des limites et souligne un point de rupture entre couverture et exploration dans cette littérature. La deuxième sous-section s'inspire de définitions en Théorie des Graphes, rapproche couverture et exploration par des critères d'optimisation liés, précise ce que l'on entend par quête d'optimalité et pourquoi ceci est extrêmement difficile.

C.1.1 Couverture et exploration en Robotique

En Robotique, les tâches de couverture et d'exploration sont liées aux problèmes de *Localisation*, de *Cartographie* et de *Contrôle* [Stachniss, 2009]. Premièrement, le problème de Localisation demande d'estimer la pose du robot à partir de l'historique des actions et mesures et de la carte de l'environnement [Thrun et al., 2001]. Deuxièmement, le problème de Cartographie demande de construire une carte de l'environnement à partir de l'historique du robot [Elfes, 1989]. Entre les deux, le problème de Localisation et Cartographie Simultanées (SLAM) demande d'estimer la pose et la carte de manière jointe ; il correspond à une instance du paradoxe de l'œuf et de la poule car les deux estimées sont mutuellement dépendantes [Durrant-Whyte and Bailey, 2006, Bailey and Durrant-Whyte, 2006]. Ces problèmes sont adressés en concevant des algorithmes de filtrage ou des estimateurs d'état qui prennent en entrée l'historique du robot à l'exécution. Par exemple, la trajectoire du robot peut être définie hors-ligne ou en-ligne par un opérateur humain, puis l'estimateur d'état filtre l'historique robotique pour calculer une pose, une carte ou les deux. Troisièmement, le problème du Contrôle est différent, il demande à planifier les mouvements d'un robot depuis une pose de départ pour atteindre une pose d'arrivée en se basant sur une carte [LaValle, 2006, Latombe, 2012].

Dans cette thèse, nous nous intéressons aux problèmes de *Perception Active* à l'intersection entre l'estimation d'état et la planification du contrôle [Bajcsy, 1988, Mihaylova et al., 2003]. Précisément, Contrôle et Cartographie définissent plusieurs types de tâches d'exploration. Ces tâches d'exploration vont de l'*Exploration Classique*, dans laquelle la pose et la carte actuelles sont fiables et le robot doit collecter de nouvelles observations pour étendre sa carte ; à l'*Exploration intégrée* ou le SLAM Actif, dans laquelle la pose et la carte ne sont pas fiables et le robot cherche également à réduire leur incertitude.

Dans cette thèse, nous nous focalisons sur le cas où la pose est fiable. Par conséquent, dans la suite cette variante sera appelée le problème de CArtographie Active.

Problème de CArtographie Active

Nous nous focalisons sur le problème de CArtographie Active (CAA), *i.e.*, l'exploration avec une localisation fiable. Précisément, la pose est fournie par un moyen externe et la carte est reconstruite à l'aide d'un filtre de Cartographie à chaque instant. Puis, les robots sont chargés de collecter l'information de manière efficace pour étendre leur carte actuelle (ou réduire son incertitude). Des stratégies intuitives pour l'Exploration Classique sont décrites par le paradigme frontière où les robots calculent des *solutions satisfaisantes* en s'approchant itérativement des zones inexplorées [Yamauchi, 1997]. Ces zones sont soit des frontières (bordure qui sépare les zones explorées des zones inexplorées) ou leur généralisation comme des vues candidates (lieux depuis lesquels des zones incertaines sont visibles). Certaines stratégies d'exploration classique

sont basées sur des critères de :

***Distance* :** comme la stratégie pionnière *MinDist* dans laquelle chaque robot choisit sa frontière la plus proche [Yamauchi, 1998, Yamauchi et al., 1998] ; *MinPos* dans laquelle chaque frontière attire son robot le plus proche [Bautin et al., 2012] ; et *MTSP* où chaque robot choisit la première frontière le long d'un cycle de coût minimum visitant un ensemble de frontières [Faigl et al., 2012].

***Information* (heuristique) :** elles sélectionnent la frontière d'utilité maximum ($U = V - \beta C_w$) qui combine son coût de plus court chemin pondéré C_w et sa visibilité depuis d'autres frontières V [Burgard et al., 2005] [18] ; la vue d'utilité maximum ($U = Ae^{-\lambda C}$) qui combine la surface inconnue de visibilité maximale A et son coût de plus court chemin C [González-Banos and Latombe, 2002] ; la frontière optimale de *Pareto* combinant coût du chemin, visibilité des frontières et des obstacles [Amigoni and Gallo, 2005] ; la stratégie de prise de décision multi-critère *MCDM* qui combine distances, visibilité de la zone inconnue et périmètres des frontières et obstacles par le biais de l'intégrale floue de Choquet [Basilico and Amigoni, 2009], etc.

***Information* (entropie) :** comme par exemple *IG-CL* [Stachniss and Burgard, 2003] qui sélectionne le point de vue d'utilité maximale ($U = \alpha\, \mathbb{E}[I] - C$), qui combine linéairement le gain d'information espéré I (sur les mesures possibles) et le coût d'un plus court chemin C [19] ; ou comme [Moorehead et al., 2001] qui combine linéairement plusieurs sources d'information (frontière, transversabilité, atteignabilité).

***Autres* :** comme la stratégie de *marché* [Zlot et al., 2002] dans laquelle des robots échangent des frontières lors d'enchères ; et la stratégie *motivationnelle* [Macedo and Cardoso, 2004] où les robots sont guidés par la faim ou la curiosité ; entre autres.

Ces techniques considèrent des objectifs à court terme et fournissent de bonnes performances en pratique, cependant ces stratégies de cartographie gloutonne sont sous-optimales dans certains environnements [Koenig et al., 2001, Tovey and Koenig, 2003]. De plus, elles évaluent les vues candidates en se basant uniquement sur la carte actuelle fournie par le filtre et la pose du robot. Elles ne se projettent pas plusieurs pas au delà de la vue candidate pour évaluer de futures cartes potentielles. En effet, cela nécessite d'envoyer des requêtes à l'avance sur l'estimateur de carte avec des historiques d'action et de mesure hypothétiques.

Dans l'Exploration Intégrée ou le SLAM Actif, il est naturel d'interroger l'estimateur de carte et de pose pour évaluer des actions à l'avance. Ces actions mènent à de nouvelles mesures qui doivent être considérées comme des variables aléatoires à priori de l'exécution. Par conséquent, les actions sont évaluées en fonction de leur profit espéré qui combine naturellement les coûts de navigation, l'incertitude de la localisation et de la cartographie.

Par exemple, des spirales paramétrées qui entrent et sortent de la zone explorée sont évaluées en termes d'incertitude d'état et de distance aux zones connues en interrogeant un *Filtre Étendu de*

[18] Un algorithme d'itération de valeur pondère le coût d'un chemin avec la probabilité d'occupation d'une cellule. Une frontière est pénalisée si elle est visible depuis des frontières déjà sélectionnées.

[19] Chaque cellule maintient une distribution de probabilité relative à son occupation. Un point de vue est une cellule dont l'entropie est supérieure à un seuil prédéfini. Le gain d'information est la différence en entropie des cellules visibles après n'importe quelle mesure.

Kalman (EKF) dans [Sim et al., 2004] ; des destinations frontières sont évaluées par la réduction d'entropie sur une grille d'occupation et leur utilité de localisation en interrogeant un EKF dans [Makarenko et al., 2002, Bourgault et al., 2002] ; et des destinations frontières ou de fermeture de boucle sont évaluées par leur réduction d'entropie en interrogeant un *Filtre Particulaire de Rao-Blackwellization* dans [Stachniss et al., 2005].

Ces études utilisent des politiques contraintes et réalise un seul pas d'anticipation (jusqu'à la prochaine destination). De plus, elles fournissent des estimateurs d'état spécifiques avec de nombreuses subtilités pour passer à l'échelle en pratique. En Prise de Décision Séquentielle (PDS), ces tâches d'exploration sont caractérisées par de l'observabilité partielle, un contrôle et une perception stochastiques. Par conséquent, elles peuvent être modélisées par le formalisme du Processus de Décision Markovien Partiellement Observable (POMDP) [Cassandra et al., 1994]. De plus, en théorie, les techniques de planification stochastiques associées permettent de réaliser de l'anticipation à plusieurs pas de temps, avec des politiques non contraintes, et de simples estimateurs d'état *Bayésiens*. Malheureusement, en pratique, les algorithmes de planification associés passent mal à l'échelle, et par conséquent leurs formalismes sont souvent négligés [Ross et al., 2008].

Cependant, nous pensons qu'il reste utile de modéliser ces problèmes à travers les formalismes de la PDS, indépendamment des techniques de planification qui sont considérées pour les résoudre. Ceci permet d'abstraire des composants liés aux actions, mesures et estimateurs d'état, afin de considérer des modèles simples mais expressifs de différentes tâches d'exploration. Par la suite, ces modèles pourraient être utilisés pour fournir des bases optimales pour des instances de petites tailles, améliorés avec des représentations et des estimateurs d'état plus efficaces, et résolus par des algorithmes de planification originaux. Comme un premier pas, nous nous focalisons sur la CAA dans laquelle l'agent peut interroger son estimateur de carte pour anticiper plusieurs pas à l'avance.

Pour résumer, le problème de CAA pose la question suivante : **Comment les robots peuvent calculer une stratégie de cartographie dans un environnement inconnu lorsque la localisation est fiable ?**

Nous n'avons pas mentionné la couverture pour l'instant mais le lien entre exploration et couverture est immédiat.

Problème de COuverture Active

Le problème de CAA est lié à la couverture par la question suivante : **Comment les robots peuvent calculer une stratégie de couverture dans un environnement connu lorsque la localisation est fiable ?** Pour mettre l'accent sur ce lien, nous allons à présent faire référence à cette tâche de couverture comme le problème de COuverture Active (COA) [20]. Les techniques qui s'appliquent à la COA différent de celles qui ont été abordées précédemment ; elles sont spécialisées et peuvent fournir des solutions optimales approchées [Galceran and Carreras, 2013]. Cependant, de notre point de vue, CAA et COA sont si proches que les techniques d'exploration

[20]Dans ce manuscrit, nous utilisons COuverture Active au lieu de *Coverage Path Planning* qui est la dénomination la plus répandue. Par ce biais, nous souhaitons souligner le fait que la tâche de couverture robotique correspond à la version vérité terrain de l'exploration robotique. Autrement dit, il s'agit d'une relaxation de l'exploration lorsque la vérité terrain est révélée.

devraient être suffisamment générales pour considérer la couverture comme un cas particulier.

Précisément, la différence réside dans l'information disponible dès le départ. D'une part, si le robot ne connaît rien de son environnement alors il s'agit du cas d'exploration le plus difficile où la planification à plusieurs pas de temps pourrait être incalculable et les stratégies gloutonnes pourraient être la meilleure option. D'autre part, la connaissance de l'environnement mène au cas de couverture le plus simple où la planification à plusieurs pas de temps pourrait être calculable en pratique et surpasser les stratégies gloutonnes. Entre les deux, il existe un continuum d'information et la planification à plusieurs pas de temps pourrait déjà être calculable au dessus d'une certaine valeur sur ce continuum.

	Truth	Known	Used	Decision
Coverage			$p = 2$ $f_1 = 1$ $f_2 = 3$	$\arg\min_{f \in \{f_1, f_2\}} Cost(p^*(p, f))$ where $Cost(p^*(p, f_1)) = Cost(p^*(p, f_2)) = 1$ hence go $\Leftarrow$ or $\Rightarrow$
Exploration				

Table C.1: Cette table illustre le comportement d'un agent frontière avec MinDist [Yamauchi, 1997] sur la couverture et l'exploration. *Truth* représente la carte d'occupation réelle m (une ligne de 4 cellules libres en blanc) et la position du robot p illustrée par • dans la seconde cellule ($p = 2$). *Known* est l'information disponible à l'agent. Dans la couverture, l'agent connaît la vérité terrain. Dans l'exploration, l'agent connaît seulement la dimension de la carte et sa position ; les autres éléments sont grisés. *Used* est l'information utilisée par l'agent pour prendre une décision. Dans la couverture ou l'exploration, l'agent considère seulement sa position courante p, la présence d'une frontière f_1 sur sa gauche et d'une frontière f_2 sur sa droite. *Decision* est la décision de l'agent après délibération. Ici, l'agent se dirige vers la frontière la plus proche. L'agent calcule le plus court chemin de p à f_1 ($p^*(p, f_1)$), et fait de même pour f_2. Dans cet exemple, l'agent peut soit se rendre à f_1 ou f_2 car $Cost(p^*(p, f_1)) = Cost(p^*(p, f_2)) = 1$.

Pour exemplifier, dans la Table C.1, nous illustrons le comportement d'un agent frontière MinDist [Yamauchi, 1997]. La carte est une grille contenant 4 cellules qui sont libres d'obstacle et le robot est dans la deuxième cellule. Durant l'exécution, l'agent observe sa pose. Comme information supplémentaire, nous fournissons une boite englobante parfaite qui servira à délimiter l'environnement de manière relative à la pose du robot. Notons que cette information n'est pas nécessaire pour mettre en œuvre un agent frontière.

Dans la couverture, l'agent frontière connaît la vérité terrain : la carte m et sa position p. Quand la mission de couverture démarre, seule la cellule courante est couverte. Dans un premier temps, l'agent extrait les frontières séparant les cellules couvertes des cellules qui ne sont pas encore couvertes ; les frontières sont les cellules immédiatement à sa gauche et à sa droite (f_1 et f_2). Dans un deuxième temps, l'agent évalue le chemin le plus court jusqu'à ces frontières ; dans ce cas la distance est évaluée par le nombre de mouvements. Dans un troisième temps, l'agent se dirige vers la frontière la plus proche ; ici elles sont toutes deux à un pas donc l'agent est indécis entre la gauche et la droite.

Dans l'exploration, l'agent frontière connaît seulement sa pose courante dans la boite englobante.

Quand la mission d'exploration démarre, seule la cellule courante est explorée. Par la suite, l'agent agit de la même manière que dans la tâche de couverture (indécision entre gauche et droite).

Dans la couverture et l'exploration, l'agent frontière ne peut pas prendre en compte l'information additionnelle (la boite englobante). Si l'agent pouvait prendre en compte l'information de boite englobante, le meilleur mouvement serait de se rendre à gauche en premier. Bien que cette exemple illustre le cas de MinDist, les mêmes décisions sont prises par au moins toutes les stratégies de distance présentées précédemment.

Pour résumer, dans un premier temps, nous avons situé notre étude en Robotique, dans cette thèse nous voyons l'exploration comme un problème de CArtographie Active et la couverture comme le problème étroitement lié de COuverture Active. Dans un second temps, nous avons exprimé notre volonté de modéliser ces tâches dans les formalismes de la Prise de Décision Séquentielle afin d'abstraire leur principaux composants. Dans un troisième temps, nous avons exprimé l'idée d'adresser COuverture Active et CArtographie Active dans le paradigme de planification. Dans la suite, nous expliquons comment le critère d'optimalité de l'exploration peut être dérivé du critère d'optimalité de la couverture. De plus, nous rapportons des résultats de la Théorie des Graphes pour justifier qu'atteindre l'optimalité est extrêmement difficile dans ces tâches.

C.1.2 Couverture et exploration en Théorie des Graphes

En Théorie des Graphes, les tâches de couverture et d'exploration sont des problèmes dans lesquels des *solutions optimales* sont recherchées. Dans la suite, l'environnement sera dénoté par un graphe $G(V, E)$, où les nœuds V sont des pièces, et les arcs E sont des liens reliant les pièces.

Problème du Voyageur de Commerce

Nous désactivons la perception à distance dans la couverture, le robot connaît l'environnement avant l'exécution et doit visiter efficacement toutes les pièces au moins une fois en démarrant depuis une position de départ. De là, nous pouvons définir le problème de Chemin de Couverture Minimum (MCP) (Definition C.1), dans lequel une séquence de mouvements à travers les arcs du graphe doit visiter tous les nœuds au moins une fois et avoir un coût minimum. Une fonction w_E pondère chaque arc de E et représente le coût du déplacement à travers un lien. Les séquences optimales recherchées sont celles qui minimisent la somme des poids de leurs arcs. Le MCP peut être reformulé comme le Problème du Voyageur de Commerce (TSP) qui est un problème d'optimisation *NP-difficile* [Garey and Johnson, 2002, Wernli, 2012]. Ainsi, il est hautement improbable qu'il existe un algorithme en temps polynomial pour résoudre ce problème.

Definition C.1 (Chemin de Couverture Minimum (MCP)).

Instance *Une graphe $G(V, E)$, une fonction $w_E : E \to Z^+$.*

Question *Trouver un* **chemin** *de couverture minimum. C'est à dire, une séquence $S = \langle v_1, v_2, ..., v_k \rangle$ de nœuds de V, telle que $\forall i \in [1, k-1], (v_i, v_{i+1}) \in E$ et $\cup_{i=1}^{k} v_i = V$, de coût minimum $\sum_{i=1}^{k-1} w_E(v_i, v_{i+1})$.*

En Prise de Décision Séquentielle (PDS), la tâche de couverture met en œuvre du contrôle et de la perception déterministes avec des observations locales qui sont connues à l'avance. Par conséquent, son processus décisionnel peut être modélisé dans le formalisme du Problème de

Recherche Complètement Observable (COSP) déterministe [Russell and Norvig, 1995] sur des états de connaissance de l'agent. De plus, en théorie, des techniques de planification déterministe pourraient calculer des trajectoires optimales pour couvrir tous les nœuds au moins une fois [Hart et al., 1968]. De notre point de vue, le MCP est le cas le plus simple de COuverture Active (COA) et met en œuvre de la *planification à long terme sans incertitude*.

Qu'en est il de l'exploration en Théorie des Graphes ?

Problème du Voyageur Canadien

À nouveau, nous désactivons la perception à distance dans l'exploration, l'environnement est inconnu à priori de l'exécution mais le robot peut se localiser et doit construire une carte de manière optimale. En d'autres termes, l'ensemble des pièces, leurs liens relatifs et les poids associés peuvent être initialement cachés au robot. En absence complète de connaissance à propos de l'environnement, la définition d'un critère d'optimalité à long terme est difficile. Cependant, si nous pouvions révéler initialement certaines propriétés de l'environnement telles que sa dimension ou le nombre de pièces et de couloirs, l'exploration pourrait être vue comme le calcul d'un MCP pour une classe donnée de graphes.

Précisément, avec ces informations, la tâche d'exploration pourrait être comprise comme une combinaison du MCP et du Problème du Voyageur Canadien (CTP) [Papadimitriou and Yannakakis, 1991, Bar-Noy and Schieber, 1991]. Le CTP est une tâche de navigation en-ligne dans laquelle un robot doit prendre une décision à la fois afin de se rendre d'un nœud source à un nœud cible dans un graphe partiellement connu. Il existe de nombreuses variantes de ce problème. Dans le meilleur cas-moyen, le robot prend une décision afin que la longueur de son chemin soit minimisée en moyenne sur la classe des graphes potentiels. Initialement, cette version a été prouvée *#P-difficile*, puis par la suite *PSPACE-difficile* [Fried et al., 2013]. En PDS, cette tâche de navigation dans l'incertain met en œuvre un contrôle et une perception déterministes avec des observations locales qui ne sont pas connues à l'avance mais le robot possède un apriori à propos de l'environnement. Ainsi, elle a été étudiée par [Blei and Kaelbling, 1999, Nikolova and Karger, 2008] dans le formalisme du Processus de Décision Markovien Partiellement Observable (POMDP) sur des états cachés du monde [Littman, 1996, Bonet, 2009].

Ici, nous voyons la tâche d'exploration comme une tâche de couverture dans un graphe partiellement connu. Le CTP couvrant apparaît en 2014 en Théorie des Graphes et combine le MCP et le CTP [Liao and Huang, 2014]. Ainsi, le critère d'optimisation précédent peut être reformulé pour choisir un mouvement qui minimise la longueur d'un chemin de couverture en moyenne. En PDS, la couverture dans l'incertain nécessite de récompenser le robot en fonction de sa croyance à propos de la carte de l'environnement. Par conséquent, son domaine peut être modélisé dans le formalisme du Processus de Décision Markovien Partiellement Observable à récompense basée sur la croyance (ρPOMDP) ([Araya-López et al., 2010a]). De notre point de vue, cette tâche d'exploration est le cas le plus simple de CArtographie Active (CAA) et met en œuvre de la *planification à long terme dans l'incertain*.

Pour exemplifier, dans la Table C.2, nous illustrons le comportement d'un agent de planification qui prend une décision optimale. Dans cet exemple, la carte est une grille de 4 cellules et le robot est sur la deuxième cellule. Durant l'exécution, l'agent observe sa position. Ainsi, pour

	Truth	Known	Used	$\text{MCP}_{\Leftarrow}$	$\text{MCP}_{\Rightarrow}$	Decision
Coverage				$\Rightarrow \times 3$	$\Rightarrow, \Leftarrow \times 3$	$\arg\min_{a \in \{\Leftarrow, \Rightarrow\}} Cost(MCP_a(p,m))$ where $Cost(MCP_{\Leftarrow}(p,m)) = 4$ $Cost(MCP_{\Rightarrow}(p,m)) = 5$ hence go $\Leftarrow, \Rightarrow \times 3$
Exploration				$\Rightarrow \times 3$	$\Rightarrow, \Leftarrow \times 3$	$\arg\min_{a \in \{\Leftarrow, \Rightarrow\}} \mathbb{E}_m[Cost(MCP_a(p,m)]$ where $\mathbb{E}_m[Cost(MCP_{\Leftarrow}(p,m)] = 1 + 16/8$ $\mathbb{E}_m[Cost(MCP_{\Rightarrow}(p,m)] = 1 + 18/8$ hence go $\Leftarrow$
				$\Rightarrow \times 3$	$\Rightarrow, \Leftarrow \times 2$	
				$\Rightarrow \times 2$	$\Leftarrow \times 1$	
				$\Rightarrow \times 2$	$\Leftarrow \times 1$	
				$\Rightarrow \times 2$	$\Rightarrow, \Leftarrow \times 3$	
				$\Rightarrow \times 2$	$\Rightarrow, \Leftarrow \times 2$	
				$\Rightarrow \times 1$	$\Leftarrow \times 1$	
				$\Rightarrow \times 1$	$\Leftarrow \times 1$	

Table C.2: Cette table illustre le comportement d'un agent de planification sur la couverture ou l'exploration. *Truth* représente la carte d'occupation réelle m (une ligne de 4 cellules libres en blanc) et la position du robot p illustrée par $\bullet$ dans la seconde cellule ($p = 2$). *Known* est l'information disponible à l'agent. Dans la couverture, l'agent connaît la vérité terrain. Dans l'exploration, l'agent connaît seulement sa position p et la dimension de m. *Used* est l'information utilisée par l'agent pour prendre une décision. Dans la couverture, l'agent utilise la vérité terrain. Dans l'exploration, l'agent considère tout ce qu'il sait pour générer des cartes cachées m. *Decision* est la décision prise par l'agent après délibération. Dans la couverture, l'agent calcule le $\text{MCP}_{\Leftarrow}(p,m)$ qui démarre par un mouvement sur la gauche ; et fait la même chose pour la droite. L'agent décide d'aller à gauche car $Cost(\text{MCP}_{\Leftarrow}(p,m)) = 4 < Cost(\text{MCP}_{\Rightarrow}(p,m)) = 5$. Dans l'exploration, l'agent calcule l'espérance de $\text{Cost}(\text{MCP}_{\Leftarrow}(p,m))$ sur toutes les cartes cachées (cellules occupées en noir) m en commençant par un déplacement sur la gauche, et fait de même en commençant par la droite. L'agent décide d'aller à gauche car $\mathbb{E}_m[Cost(\text{MCP}_{\Leftarrow}(p,m))] = 3 < \mathbb{E}_m[Cost(\text{MCP}_{\Rightarrow}(p,m))] = 3.25$.

inférer qu'une cellule est bloquée par un obstacle, l'agent doit rebondir contre l'obstacle. À nouveau, comme information additionnelle, nous fournissons une boite englobante parfaite qui délimite l'environnement de manière relative à la position du robot.

Dans la couverture, l'agent doit connaître la vérité terrain (carte et position) et exploiter cette information pour calculer le MCP qui visite toutes les cellules. Quand la mission de couverture démarre, seule la cellule courante est couverte. Dans ce cas, le MCP est de se rendre à gauche une première fois et à droite trois fois de suite juste après, ce qui donne une trajectoire de 4 mouvements.

Dans l'exploration, l'agent connaît sa position et la boite englobante de l'environnement, et va utiliser cette information pour calculer le MCP qui visite toutes les cellules. Cependant, comme la carte est cachée, le MCP est aussi caché. Sans information additionnelle, l'agent pourrait considérer que la carte est échantillonnée de manière uniforme. Dans un premier temps, l'agent échantillonne les cartes potentielles (avec des cellules libres en blanc et des cellules occupées en noir). Dans un deuxième temps, pour chaque carte potentielle, l'agent calcule le MCP qui démarre avec un mouvement par la gauche et le MCP qui démarre avec un mouvement par la droite. Dans un troisième temps, l'agent calcule l'espérance de la longueur du MCP pour chaque mouvement de départ. Finalement, l'agent va à gauche, car l'espérance de la longueur du MCP associée est la plus basse.

C'est exemple n'est qu'illustratif : un solveur efficace devrait approximer le profit espéré d'un mouvement en générant un nombre faible d'échantillons de cartes, en anticipant seulement quelques pas à l'avance sur chaque chemin de couverture et en pondérant le coût de n'importe quel chemin de couverture avec son utilité en terme de gain d'information.

Pour résumer, nous avons rapporté des études liées aux tâches de couverture et d'exploration en Robotique expérimentale et en Théorie des Graphes. En Robotique, ces tâches sont liées à la COuverture Active et la CArtographie Active, pour lesquelles des solutions satisfaisantes sont déjà disponibles. En Théorie des Graphes, elles sont liées aux Problème du Voyageur de Commerce et Problème du Voyageur Canadien, dont les solutions optimales pourraient être extrêmement difficiles à calculer. De plus, nous avons rapporté des critères d'optimisation à long terme qui peuvent être utilisés pour définir des agents de planification qui visent l'optimalité bien que l'atteindre soit hautement improbable en considérant les classes de complexité citées plus haut. Additionnellement, nous avons identifié des formalismes de la PDS qui représentent des points d'entrée pour modéliser les problèmes de COA et de CAA. Finalement, nous avons fourni un exemple pour démontrer la capacité d'un agent de planification à faire un mouvement optimal en moyenne.

Dans la suite, nous définissons l'objectif à long terme de cette dissertation et nos hypothèses à court terme.

C.1.3 Objectif à long terme et hypothèses à court terme

Nous objectif à long terme est de résoudre les tâches de couverture et d'exploration avec des robots réels. Cependant, concevoir un seul robot est difficile selon trois composants :

Perception : l'agent compte sur ses capteurs embarqués pour fournir des données concernant son environnement à son unité de prise de décision embarquée ;

Prise de décision : l'agent compte sur son unité de prise de décision pour fournir des décisions rationnelles à ces actionneurs ; et

Action : l'agent compte sur ses actionneurs pour appliquer les décisions afin d'agir sur l'environnement.

Par conséquent, nous nous concentrons sur la prise de décision et faisons des hypothèses fortes à propos de la perception et de l'action. Nous abstrayons les unités de perception et d'action à leur forme la plus simple, en considérant des données formatées à l'entrée et à la sortie. De plus, nous nous focalisons sur la conception de robots simulés qui prennent des décisions rationnelles pendant des missions de COuverture Active ou de CArtographie Active.

Hypothèses sur la continuité, la perception et l'action

Les robots mobiles possèdent de nombreux degrés de liberté, des moteurs et des capteurs. De plus, l'environnement physique est continu et complexe. Pour cela, de nombreux roboticiens conduisent des études dans lesquelles la configuration de l'environnement, la configuration des robots, leurs entrées et sorties sont continues. Cependant, il existe également de nombreuses études qui se basent sur des représentations discrètes de l'environment, comme les conventionnelles grilles d'occupation déterministe [Thrun, 2003], les grilles d'occupation probabiliste [Elfes, 1989] et leur extension éparse [Hornung et al., 2013]. De manière similaire, nous allons manipuler des espaces de travail discrets.

Nous mettons des hypothèses fortes sur les perceptions comme nous ne considérons pas le processus de calibration de capteurs, de lecture de signaux bruts et de leur transformation en information utile pour l'unité de prise de décision (en débruitant, filtrant et lissant). Nous considérons des mesures formatées à l'entrée de l'unité de prise de décision. De manière similaire, pour les actions, nous ne considérons pas le processus de transformation d'une décision en une action réelle sur l'environnement (en activant des moteurs, en réglant la vitesse et le couple). Nous considérons des décisions formatées à la sortie de l'unité de prise de décision.

Cependant, l'information nécessaire aux modèles de prise de décision rapportés dans cette dissertation est relativement raisonnable, bien que non triviale. En effet, elle pourrait nécessiter la résolution de problèmes de perception pré-délibération comme le problème de l'association de données (en fonction de la dynamique de l'environnement) ; et des problèmes d'action post-délibération comme la résolution d'équations cinématiques et dynamiques (en fonction du type de robot).

Il y a déjà de nombreuses applications réelles de systèmes d'exploration qui apparaissent dans la littérature de Robotique expérimentale et les logiciels d'intégration robotique [Quigley et al., 2009] sont aujourd'hui largement adoptés. Notre travail considère que les robots peuvent se localiser par un moyen externe, pour cela nous pouvons compter sur des Systèmes de Positionnement en Intérieur précis [Liu et al., 2007]. De manière similaire, nous considérons que les robots peuvent distinguer entre différentes classes d'objets statiques (murs) et d'objets mobiles (animaux, piétons) ; pour ce faire nous pouvons compter sur des résultats récents en Vision par Ordinateur [Krizhevsky et al., 2012]. Ainsi, une intégration ne semble pas représenter une limitation majeure malgré les hypothèses listées précédemment.

Dans la suite, nous identifions des défis liés à la prise de décision et listons nos propositions pour les adresser ; ces propositions seront ensuite détaillées par différentes études.

C.2 Challenge : défis identifiés et études proposées

C.2.1 Défi I : modèles de couverture et d'exploration mono-agent

Premièrement, nous voulons exploiter le cadre de la Prise de Décision Séquentielle (PDS) en Intelligence Artificielle pour fournir des premières réponses à nos interrogations concernant des paradigmes agent avec de l'information non restrictive et des comportements visant l'optimalité en couverture et en exploration robotique.

Comment spécifier un paradigme agent qui utilise toute l'information disponible pour la couverture et l'exploration ?

Nous avons mentionné que le paradigme frontière limitait l'information utilisée par l'agent pour la prise de décision. De plus, nous avons illustré un agent de planification simple qui utilisait de l'information additionnelle pour anticiper ce qui se produit au delà de la frontière, tandis que l'agent frontière était aveugle à cette information supplémentaire. Envisager ce qui peut se produire au délà de la frontière est attrayant car cela permet une prise de décision à granularité plus fine. Cependant, les modalités de notre exemple sont simplistes et sa généralisation est peu claire. Nous pensons que généraliser cet agent de planification pour la COuverture Active (COA) et la CArtographie Active (CAA) facilitera l'adoption de techniques de théorie décisionnelle pour anticiper au delà de la frontière.

Comment spécifier des agents qui recherchent l'optimalité pour la couverture et l'exploration ?

Nous avons déjà mis l'emphase sur la différence entre les objectifs court terme vs. long terme. Nous avons mentionné que les agents frontières largement adoptés en Robotique expérimentale validaient de manière gloutonne des objectifs à court terme et fournissaient de bonnes performances sous-optimales. Nous avons également illustré le comportement d'un agent de planification qui utilise toute l'information et décide en fonction d'un critère d'optimisation à long terme défini par le problème de Chemin de Couverture Minimum. Comme attendu, cet agent a pris une meilleure décision que l'agent frontière dans la même situation. Nous savons qu'atteindre l'optimalité est hautement improbable en connaissance des classes de complexité des Problème du Voyageur de Commerce et Problème du Voyageur Canadien. Cependant, rechercher l'optimalité à long terme pour prendre des décision plus informées pourrait être une route d'étude raisonnable.

Comme dit dans notre contexte, nous allons nous appuyer sur le cadre de la PDS [Littman, 1996] pour adresser ces questions. Ce cadre contient des formalismes de processus décisionnels standards qui sont : expressifs, ce qui les rends génériques, car ils spécifient l'interaction entre des agents et leur environnement pour accomplir une tâche ; exhaustifs, ce qui les rend strictes, car ils encapsulent seulement les aspects pertinents d'une tâche ; compacts, ce qui les rend transférables, car ils spécifient peu d'items qui décrivent la dynamique de l'environnement et les actions, perceptions et récompenses de l'agent ; et calculatoires, ce qui les rend reproductibles, car chaque spécification est un simulateur.

Ainsi, dans un premier temps, nous modélisons des domaines de couverture et d'exploration en spécifiant différentes modalités d'interaction : nous nous appuyons sur des formalismes à espace d'état pour simuler des mondes dans lesquels un agent situé agit et perçoit l'environnement de manière stricte. Dans un second temps, nous activons le paradigme de planification pour la couverture et l'exploration, dans lequel l'agent accède au modèle de la tâche, à l'information disponible à priori et aux historiques d'actions et de mesures : nous nous servons de statistiques suffisantes pour compiler cette information. C'est à dire, nous modélisons des processus décisionnels sur ces statistiques et montrons comment la collecte d'information peut être récompensée sur le long terme.

Nous nous concentrons sur la COuverture Active et CArtographie Active Mono-Robot et considérons un agent de planification centralisé. Un agent de planification centralisé peut contrôler un seul ou plusieurs robots synchronisés de manière transparente. Cependant, nous illustrons le cas d'un agent de planification qui optimise les décisions d'un seul robot. Le cas de la prise de décision décentralisée est différent car il considère un agent de planification par robot. Cette fois, les observations et les actions ne sont pas faites de manière jointe par les robots et les communications doivent être modélisées explicitement. La prise de décision décentralisée est hors cadre de notre étude actuelle, mais ces formalismes existent et devraient être examinés plus en détails par la suite [Bernstein et al., 2002].

C.2.2 Défi II : couverture centralisée d'environnements statiques

En deuxième, nous voulons démontrer de manière empirique les limitations actuelles pour garantir l'optimalité avec des techniques de planification classique sur les modèles de couverture robotique. Précédemment, nous avons montré que le calcul du prochain mouvement en exploration pourrait faire intervenir la planification de solutions pour des tâches de couverture échantillonnées. Ainsi, comme un premier pas, nous allons nous focaliser sur des tâches de couverture et nous appuyer sur la recherche heuristique pour obtenir ces solutions.

Quelles sont les limitations actuelles pour garantir l'optimalité, la sous-optimalité, la progressivité des solutions au modèle de couverture ?

Nous avons rapporté des modèles de couverture pour le paradigme de planification, au niveau de la connaissance de l'agent avec un accès au modèle de la tâche. Ainsi, la route est ouverte pour résoudre les tâches de couverture comme des jeux de PDS avec des agents de planification qui peuvent réaliser de la prise de décision non aveugle. Pour ce faire, nous réalisons deux pas supplémentaires dans cette direction. Dans un premier temps, comme nous ne pouvons pas étudier tous les modèles de couverture et les solveurs génériques qui s'y appliquent, nous nous focalisons sur le modèle de couverture à contrôle déterministe. Dans un second temps, l'agent de planification peut s'appuyer sur une famille de solveurs de plus court chemin appelée la recherche heuristique qui s'applique au modèle de couverture. Nous souhaitons évaluer ses performances en terme de qualité de solution et coût de la prise de décision.

Comme dit plus haut, nous allons nous appuyer sur des techniques de recherche heuristique qui sont des techniques de planification standards issues de la PDS [Edelkamp and Schroedl, 2011]. Ces techniques sont : des solveurs de plus court chemin, ce qui les rend génériques, car ils s'appliquent à la large classe des problèmes qui peuvent être réécrits comme des problèmes de plus court chemins ; ajustables, ce qui les rend spécifiques, car ils s'appuient sur de la

connaissance experte pour évaluer si un état donné est prometteur ; et théoriquement enracinés, ce qui les rend attractifs, car ils jouissent de propriétés et de théorèmes qui garantissent complétude, correction, epsilon sous-optimalité ou aussi des performances progressives sous des hypothèses relativement raisonnables.

Comme la littérature de la recherche heuristique est vaste, nous restreignons notre étude au paradigme de recherche hors ligne ([Dijkstra, 1959, Hart et al., 1968]). Les paradigmes en-ligne ([Ramalingam and Reps, 1996, Koenig et al., 2004]) et temps-réel ([Korf, 1990, Pemberton and Korf, 1992]) sont hors cadre de notre étude actuelle et devront être examinés plus en détail par la suite. En premier, nous considérons les algorithmes issus des cadres bien connus Best-First Search et Depth-First Search. En deuxième, nous conduisons des expériences avec ces techniques pour résoudre des instances du modèle de couverture ; nous les spécialisons avec de la connaissance experte et évaluons leur comportement moyen pour l'obtention de solutions optimales, sous-optimales et progressives sous contraintes de ressources limitées.

Ceci est différent du paradigme par frontière qui fournit des solutions satisfaisantes, peut garantir la complétude, mais ne propose pas de garantie forte concernant la sous-optimalité du comportement de l'agent. Pour souligner la différence entre l'agent de planification et l'agent frontière : l'agent de planification évalue les frontières avec un objectif à long terme de couverture, c'est à dire que la valeur de chaque frontière prend en compte le plus court chemin jusqu'à la frontière et le chemin de couverture minimum au delà. À l'opposé, l'agent frontière évalue les frontières avec un objectif à court terme de couverture satisfaisante, et considère seulement le plus court chemin jusqu'à la frontière.

C.2.3 Défi III : exploration décentralisée en environnements dynamiques et carte de navigation distribuée en environnements ambiants

Dans un troisième temps, nous voulons considérer des cadres plus réalistes. Les modèles précédents n'illustrent qu'un seul robot et nous avons considéré un agent de planification hors-ligne pour résoudre le modèle de couverture déterministe. De manière similaire, ces modèles pourraient être étendus à une équipe de robots en environnement dynamique et nous pourrions considérer un agent de planification hors-ligne pour les résoudre. Cependant, bien que des modèles étendus puissent être définis, l'agent de planification hors-ligne précédent aurait besoin de ressources considérables pour calculer le contrôle joint des robots.

Comment réaliser de l'exploration robotique décentralisée en environnement dynamique avec des piétons ?

Nous étudions le cadre plus réaliste d'une tâche d'exploration qui met en œuvre plusieurs robots qui calculent chacun des décisions de manière décentralisée et s'appuient sur de la re-planification pour corriger des plans obsolètes. De plus, nous sommes intéressés par des environnements dynamiques car les robots partagent de plus en plus l'espace physique avec les humains dans des lieux publics ou privés tels que des musées (*e.g.* guides robotique), jardins (*e.g.* tondeuses robotique), ou maisons de santé (*e.g.* assistant sanitaire robotique).

Comment calculer des routes de navigation sûre de manière distribuée en environnements ambiants ?

Jusqu'à présent, nous nous sommes focalisés sur la couverture d'environnements connus et la cartographie d'environnements inconnus, mais dans un futur proche de plus en plus de maisons et d'aménagements seront équipés d'intelligence ambiante, qui intègre des unités de perception et de calcul dans l'environnement. Dans ce contexte, la couverture et l'exploration pourraient faire intervenir une équipe de robots qui s'appuient sur de telles unités intégrées à l'environnement afin d'étendre leur capacité de perception et de calcul. En conséquence, nous examinons comment des robots pourraient se baser sur de telles unités pour adresser l'évitement d'obstacle.

Ainsi, nous proposons d'étendre le cadre applicatif de la CArtographie Active à des environnements comprenant des objets mobiles ; et de construire de manière collaborative des services pour naviguer de manière sûre dans l'environnement.

Dans le premier cas, nous modélisons la CArtographie Active Multi-Robot dans un environnement dynamique avec des piétons. Les modalités de ce modèle permettent de s'appuyer sur le paradigme frontière. Par la suite, nous proposons un paradigme interactif qui permet d'initier des interactions locales avec les piétons. Finalement, nous évaluons de manière quantitative les performances d'une équipe de robots qui alternent entre des assignations aux frontières et aux suivis de piétons de manière décentralisée.

Dans le second cas, nous modélisons un environnement ambiant équipé d'un sol qui capte la pression par le biais d'un Automate Cellulaire. Ensuite, nous adressons le problème de l'Évitement d'Obstacle par Chemin Clairsemé via le calcul spatial de manière asynchrone. Les unités intégrés dans le sol construisent de manière collaborative une carte de navigation de chemins clairsemés. Finalement, nous évaluons de manière qualitative les propriétés de ces carte de navigation sur différents types d'environnements à partir de données synthétiques et réelles.

C.3 Plan : résumé des études proposées

Pour clore cette introduction, ci-après nous détaillons la structure du manuscrit.

C.3.1 Partie I : Modélisation à Espace d'État pour la COuverture Active et la CArtographie Active

Dans cette partie, nous spécifions des domaines de COuverture Active et de CArtographie Active Mono-Robot. Un domaine décrit comment un agent interagit avec son environnement. Il encapsule tous les aspects du monde (environnement, agent et leurs interactions) qui sont pertinents pour une tâche donnée, et nécessaires pour simuler des trajectoires d'actions, d'observations et de récompenses de notre agent situé. De plus, nous proposons des modèles qui peuvent supporter de la prise de décision par un agent de planification dans de tels domaines. Un modèle décrit comment l'agent peut anticiper et être récompensé lorsqu'il construit des solutions pour une tâche donnée.

Chapitre 2

Ce chapitre est dédié à la modélisation des domaines de COuverture Active Mono-Robot dans des formalismes de la PDS. Nous considérons que les observations locales sont connues à l'avance et déterministes à l'exécution, par conséquent le processus de décision sera défini sur des états de connaissance de l'agent. En effet, dans notre tâche de couverture, l'état du monde consiste en la position de l'agent et la carte de l'environnement, sachant que la carte est révélée à l'agent à priori de l'exécution.
Dans un premier temps, nous rapportons des formalismes issus de la Prise de Décision Séquentielle avec des états complètement observables. Dans un deuxième temps, nous fournissons des domaines et des modèles de prise de décision avec différents types de transition d'état qui décrivent comment l'agent se déplace et maintient une carte de couverture. Dans ces modèles de prise de décision, l'agent de planification connaît son niveau actuel de réalisation en se basant sur une carte de couverture.

Chapitre 3

Ce chapitre est dédié à la modélisation des domaines de CArtographie Active Mono-Robot dans des formalismes de la PDS. Nous considérons une observabilité partielle de l'état du monde. En effet, dans une tâche d'exploration, l'état du monde consiste en la position de l'agent et la carte de l'environnement ; l'agent connaît sa position mais n'observe que localement la carte à l'exécution.
Premièrement, nous rapportons des formalismes de la PDS avec des états partiellement observables qui différent en terme de types de transition et d'observation d'état. Deuxièmement, nous fournissons un domaine d'exploration pour chaque formalisme rapporté. Ensuite, nous rapportons des formalismes avec des croyances observables qui différent en termes de type de transition de croyance. Finalement, nous fournissons des modèles d'exploration sur les croyances de l'agent qui sont basés sur nos domaines d'exploration. Dans ces modèles l'agent de planification maintient explicitement une croyance à propos de la carte réelle de l'environnement.

C.3.2 Partie II : Planification à Long Terme pour la COuverture Active Déterministe

Dans cette partie, nous évaluons de manière empirique les performances d'un agent de planification pour des tâches de COuverture Active Mono-Robot comme spécifiées par notre modèle de couverture déterministe. Notre approche consiste à chercher une solution dans l'espace d'état du modèle de couverture. En effet, ce modèle produit des problèmes de plus court chemin déterministes qui peuvent être résolus par des algorithmes de recherche heuristique.

Chapitre 4

Ce chapitre est dédié à la résolution d'instances du domaine de COuverture Active Mono-Robot déterministe. Pour ce faire, nous nous appuyons sur le paradigme de la recherche hors-ligne. Les algorithmes de recherche hors-ligne peuvent trouver une séquence optimale de mouvements reliant l'état initial de l'agent à un état but. Dans notre modèle, un but est un état dans lequel l'agent a observé l'environnement complet en le visitant physiquement. De plus, ces algorithmes peuvent être accordés avec de la connaissance experte pour améliorer leurs performances.
Dans un premier temps, nous discutons des principes au cœur de la recherche hors-ligne : la délibération non bornée et la planification unique de chemin. Dans un second temps, nous passons

en revue l'état de l'art de ses techniques les plus populaires. Dans un troisième temps, nous conduisons un banc d'essai de techniques représentatives pour calculer des solutions optimales ou sous-optimales à la tâche de couverture déterministe de manière complète et progressive.

C.3.3 Partie III : Planification à Court Terme pour la CArtographie Active et Évitement d'Obstacle par Chemin Clairsemé

Dans cette partie, nous étendons le cadre applicatif de la CArtographie Active Multi-Robot à des environnements dynamiques avec des piétons, et de l'Évitement d'Obstacle par Chemin Clairsemé à des environnements ambiants avec des unités de perception et de calcul distribuées.

Chapitre 5

Ce chapitre est dédié à la résolution d'instances du domaine de CArtographie Active Multi-Robot déterministe en environnement peuplé de piétons. Nous considérons une équipe de robots qui prennent des décisions de manière décentralisée. Pour ce faire, nous nous appuyons sur le paradigme par frontière et proposons également un paradigme interactif dans lequel des robots cherchent à suivre des piétons durant l'exploration.
Premièrement, nous étendons le modèle précédent de CArtographie Active Mono-Robot, à contrôle et perception déterministes, à plusieurs robots et piétons. Deuxièmement, nous hybridons le paradigme frontière avec un paradigme interactif. Le paradigme interactif permet de suivre localement des objets mobiles dans l'environnement. De telles interactions sont évaluées en se basant sur des pénalités liées à la durée d'inactivité, l'orientation et la distance. Troisièmement, nous évaluons quantitativement en simulation comment le paradigme hybride se comporte en environnement peuplé.

Chapitre 6

Ce chapitre est dédié au problème d'Évitement d'Obstacle par Chemin Clairsemé en environnement ambiant avec des unités de perception et de calcul intégrées dans le sol. L'Évitement d'Obstacle par Chemin Clairsemé est une alternative à l'Évitement d'Obstacle par Plus Court Chemin, le premier produit des trajectoires aussi éloignées que possible des obstacles tandis que le second produit des trajectoires dans leur voisinage. Les trajectoires clairsemées sont préférées dans des environnements dynamiques où des objets mobiles peuvent facilement entrer en collision avec les robots. Les unités du sol calculent de telles trajectoires de manière distribuée et asynchrone. En effet, maintenir les unités synchronisées à n'importe quel instant est difficile.
Premièrement, nous modélisons les unités de pression intégrées au sol à l'aide d'un Automate Cellulaire abstrait dont chaque cellule représente une telle unité. Deuxièmement, nous calculons un Diagramme de *Voronoï* dont les centroïdes correspondent aux obstacles perçus par le sol. Troisièmement, nous évaluons qualitativement en simulation sur des données synthétiques et réelles comment des bissectrices entre régions de *Voronoï* pourraient servir de routes clairsemées pour des robots mobiles.

C.3.4 Conclusion et Perspectives

Cette partie conclue les différentes études et envisage des améliorations et extensions potentielles.

Chapitre 7

Ce chapitre résume les modèles de couverture et d'exploration mono-agent qui sont fournis dans ce document ; le banc d'essai des techniques de recherche hors ligne sur notre modèle de couverture déterministe ; l'étude d'exploration multi-agent en environnement dynamique avec des piétons ; et l'étude de navigation en environnement ambiant intégrant des unités de perception au sol.

Additionnellement, nous discutons d'études futures à propos des modèles multi-agent avec prise de décision décentralisée pour la couverture et l'exploration ; et de l'évaluation de performances des techniques de planification en-ligne et temps-réel sur notre modèle de couverture. Puis, nous proposons des améliorations pour l'évaluation des interactions entre robots et piétons ; et le calcul de carte de navigation à chemins clairsemés.

Résumé

Pouvoir se déplacer intelligemment dans un environnement inconnu est primordial pour des robots mobiles (Évitement d'Obstacle (EO)). Ceci est nécessaire pour explorer et construire une carte de l'environnement (CArtographie Active (CAA)), carte qui servira à d'autres tâches comme la patrouille (COuverture Active (COA)). Cette thèse se focalise sur la prise de décision pour planifier les déplacements de robots autonomes afin de naviguer, couvrir ou explorer l'environnement. Ainsi, nous nous basons sur la Prise de Décision Séquentielle (PDS) en Intelligence Artificielle et proposons deux contributions concernant : (1) les processus décisionnels de CAA et COA, et (2) la planification à long terme pour la COA. De plus, récemment, les robots mobiles ont commencé à partager l'espace physique avec les humains en fournissant des services comme du ménage à la maison. Dans ces cas, le comportement du robot doit s'adapter à la dynamique du monde. Par conséquent, nous proposons deux autres contributions pour : (3) la CAA en environnements de foule, et (4) l'EO par chemin clairsemé en environnements ambiants.

1. Nous spécifions des processus décisionnels de COA et de CAA mono-robot pour supporter la planification à long terme. Ainsi, nous utilisons le cadre de la PDS pour spécifier des domaines qui simulent et récompensent des réalisations d'actions et d'observations futures. Par exemple, nous modélisons les tâches d'exploration comme des problèmes d'estimation active en utilisant le formalisme du Processus de Décision Markovien Partiellement Observable à récompense basée sur la croyance. Ces domaines peuvent supporter des travaux actuels sur la CAA gloutonne à court terme, et des travaux futurs sur la planification à long terme.

2. Nous résolvons des instances du domaine de la COA mono-robot à contrôle déterministe par la biais de la planification à long terme. Pour cela, nous utilisons la recherche heuristique sur des problèmes de plus courts chemins en domaine statique et à information complète. Nous envisageons le paradigme de recherche hors-ligne pour calculer un plan complet avant exécution. En premier, nous réalisons un état de l'art des techniques de recherche dans ce paradigme. Puis, nous évaluons certaines techniques en simulation afin de planifier des solutions optimales, sous-optimales, et progressives à différentes instances de la tâche de couverture.

3. Nous étendons le cadre applicatif de la CAA multi-robot à des foules. Nous considérons que les robots distinguent perceptuellement objets statiques ou mobiles. Nous évaluons en simulation les performances d'un cadre en ligne d'allocation robot-frontière pour la cartographie gloutonne décentralisée. Puis, nous l'hybridons en incluant des interactions de suivi d'objets mobiles. Finalement, nous évaluons quantitativement la paramétrisation des coûts d'assignation par des pénalités d'inactivité et d'orientation. Les résultats démontrent que certains paramètres ne pénalisent pas la collecte d'information. De nouvelles études pourraient s'intéresser à leur adaptation en ligne.

4. Nous étudions l'EO par chemin clairsemé en environnement ambient équipé de capteurs de pression au sol. Nous proposons une approche en ligne de calcul de carte de navigation clairsemée, *i.e.*, à équidistance des obstacles, par le calcul spatial. Pour cela, nous calculons un diagramme de Diagramme de *Voronoï* de manière distribuée et asynchrone. Nous l'évaluons et la validons qualitativement en simulation sur des données synthétiques et réelles. De nouvelles études pourraient s'intéresser à la synchronisation sélective des unités de calcul et de perception.

Mots-clés: intelligence artificielle, planification, recherche, couverture, exploration, multi-agent

Abstract

The ability to intelligently navigate in an unknown environment is essential for mobile robots (Obstacle Avoidance (OA)). This is needed to explore and build a map of the environment (Active Mapping (AM)); this map will then support other tasks such as patrolling (Active Coverage (AC)). In this thesis, we focus on decision-making to plan the moves of autonomous robots in order to navigate, cover, or explore the environment. Therefore, we rely on the framework of Sequential Decision-Making (SDM) in Artificial Intelligence to propose two contributions that address: (1) decision processes for AC and AM and (2) long-term planning for AC. Furthermore, mobile robots recently started sharing physical spaces with humans to provide services such as cleaning the house. In such cases, robot behavior should adapt to dynamic aspects of the world. In this thesis, we are interested in deploying autonomous robots in such environments. Therefore, we propose two other contributions that address: (3) short-term AM in crowded environments and (4) clearest path OA in ambient environments.

1. We specify Single-Robot AC and AM decision processes to support long-term planning. To do so, we use the framework of SDM to specify domains that simulate and reward future sequences of actions and observations. For instance, we model exploration tasks as active estimation problems using the Partially Observable Markov Decision Process with belief-dependent rewards formalism. Such domains can support both current work dedicated to short-term greedy AM and future studies about long-term planning.

2. We solve instances of the Single-Robot AC domain with deterministic control by relying on long-term planning. We focus on heuristic search to solve deterministic shortest path problems in static domains with initially complete information. We consider the paradigm of off-line search (compute full plan before execution). First, we conduct a state of the art of techniques from this paradigm. Then, we benchmark some techniques in simulation for planning optimal, suboptimal, or anytime solutions to different instances of the coverage task.

3. We extend the application scope of Multi-Robot AM to crowded environments. We assume that robots are able to perceptually disambiguate between static and moving items. We evaluate in simulation the performances of an on-line frontier-allocation framework for decentralized greedy mapping. Then, we hybridize this framework to include object-following interactions with the moving items. Finally, we quantitatively evaluate the inclusion of parametrized penalties of idleness and orientation into assignment costs. The results demonstrate that information gathering is not penalized for some fixed parameters. Further studies could investigate on-line tuning.

4. We address the clearest path OA problem in ambient environments equipped with load-pressure sensors embedded in a floor. We propose an on-line approach to compute maximum clearance roadmaps, *i.e.*, equidistant to obstacles, by the means of spatial computing. Therefore, we compute *Voronoi* diagrams in a distributed and asynchronous manner. We qualitatively evaluate and validate this approach in simulation on both synthetic and real data. Further studies could investigate selective synchronization of the processing and sensing units.

Keywords: artificial intelligence, planning, search, coverage, exploration, multi-agent