POLITECNICO DI MILANO

SCUOLA INTERPOLITECNICA DI DOTTORATO

DOCTORAL PROGRAM IN INFORMATION TECHNOLOGY

Final Dissertation

# STUDY, DESIGN, AND EVALUATION OF EXPLORATION STRATEGIES FOR AUTONOMOUS MOBILE ROBOTS

**Alberto Quattrini Li**

Supervisor:
**Prof. Francesco Amigoni**

Co-ordinator of the Research Doctorate Course:
**Prof. Carlo Fiorini**

March 5th, 2015

**Politecnico di Milano**
***Dipartimento di Elettronica, Informazione e Bioingegneria***
DOCTORAL PROGRAM IN INFORMATION TECHNOLOGY

---

# STUDY, DESIGN, AND EVALUATION OF EXPLORATION STRATEGIES FOR AUTONOMOUS MOBILE ROBOTS

Doctoral Dissertation of:
**Alberto Quattrini Li**

Supervisor:
**Prof. Francesco Amigoni**

Tutor:
**Prof. Luciano Baresi**

The Chair of the Doctoral Program:
**Prof. Carlo Fiorini**

2014 – XXVII

*Ai miei nonni Lilia ed Enzo ...*

# Ringraziamenti

Per concludere questa tesi, desidero ringraziare tutte le persone senza le quali non avrei potuto portare a termine questo percorso, anche se, per ragioni di brevità, posso riportarne solo alcune.

Desidero innanzitutto ringraziare Prof. Francesco Amigoni, che con il suo costante supporto sia a livello scientifico sia a livello umano mi ha accompagnato alla fine di questo percorso di arricchimento durato tre anni. Spero di continuare a imparare molto da lui.

A special thank goes also to Prof. Volkan Isler, who hosted me at Robotic Sensor Network Lab at University of Minnesota for 6 months, and made me feel like at home, besides guiding me, even now that I am not there anymore. Also, many thanks to all the people in the lab, Patrick Plonski, Joshua Vander Hook, Narges Noori, Alessandro Renzaglia, Gabriel Oliveira, Ubaldo Ruiz, Nikolaos Stefas, and Pravakar Roy.

Further, I am grateful for the invaluable feedback from my reviewers Prof. Lino Marques, Prof. Edson Prestes, and Prof. Ioannis Rekleitis.

Un sentito ringraziamento va anche alla Prof. Licia Sbattella e al Dott. Roberto Tedesco con cui ho avuto ed ho l'onore di lavorare su temi che contribuiscono a migliorare la vita di persone con difficoltà.

Un caloroso ringraziamento va anche a tutte le persone che hanno reso divertente il dipartimento, Nicola Basilico, Marco Rocco, Matteo Luperto, Francesco Trovò, Diego Carrera, Jacopo Banfi.

Grazie anche agli studenti di cui sono stato correlatore/supervisore, Michele Giusto, Riccardo Cipolleschi, Alessandro Riva, Mattia Ornaghi, Sara Biancini, Stefano Carnovali, Simone Pignatelli, Alessandro Maria Rizzi, Sajjad Salehi, Giuseppe Andrea Ferraro, Federico Marini.

Ringrazio a tutta la mia famiglia che ha continuato a supportarmi in tutti i modi possibili durante questi anni lontano da casa.

Un sentito ringraziamento va anche ai miei ex compagni di corso, Alessandro, Iacopo, Andrea Parola, Andrea Rossali, Francesco, con i quali ho continuato ad avere una forte amicizia nonostante la lontananza.

Infine, un grazie infinito a Claudia che ha saputo incoraggiarmi e sostenermi durante i momenti difficili di questo intenso percorso e continua a farlo credendo in me.

# Abstract

In recent years, autonomous mobile robotics is increasingly becoming an effective way to carry out tasks that are difficult, dangerous, or simply boring for humans. Relevant examples include planetary exploration and search and rescue. While developing a system of autonomous robots, a designer should take care of fundamental issues that relate to locomotion, sensing, localization, and navigation. One of the most important and challenging aspects that could significantly impact on the system performance is the autonomous decision making to carry out the assigned task. This comprises the set of techniques that allow autonomous mobile robots to decide the next location to reach and to possibly coordinate among themselves, according to their current knowledge of the world they operate in. Consider for example the *exploration problem*, where mobile robots are employed for incrementally discovering and mapping the features of initially unknown environments, which could serve for more general missions, such as map building and search and rescue. The robots have to select the next locations where to perform sensing actions within the currently explored portion of the environment and who goes where. Clearly, the decisions made could have a significant impact on the performance of the exploration and, if made effectively, could really boost the robots' autonomy. However, despite the importance of the development of techniques to make mobile robots more autonomous, general techniques are not mature yet.

In this dissertation, we focus on some key activities for the multirobot exploration problem, namely the selection of interesting locations (*exploration strategy*) and their assignment to robots (*coordination method*). We aim at contributing to the achievement of three main research goals: to bridge the gap between theory and practice for exploration strategies; to improve exploration strategies and coordination methods employed for exploring an initially unknown environment by one or more robots; and to improve the experimental assessment of multirobot exploration systems.

The first goal is motivated by the fact that the exploration problem has been addressed in literature with two rather different approaches. On the one hand, this problem is theoretically studied, by providing worst-case bounds and competitive ratios for proposed methods, although, sometimes, assumptions are far from being realistic. On the other hand, methods are defined and tested in practical contexts of real robots. Against this background, we define the problem of calculating the optimal (offline) exploration paths in grid environments for a robot under realistic assumptions of limited and time-discrete visibility. Simulation results show the viability of our proposed approach for realistic environments. Further, we theoretically analyze some exploration strategies that evaluate candidate destination locations by combining their distance and their expected information gain and operate on graph-based environments. Specifically, we provide bounds on the number of edge traversals required to explore a generic graph by a single robot. Results show that, in the worst case, considering also information gain does not provide any advantage over considering only distance, while it does in the average case on graphs modeling realistic indoor environments.

Second, we define exploration strategies and coordination methods that base their decisions not only on metric information that derives from sensor readings, but also on *semantic information* which associates some high-level concepts to areas of the environment. This enables robots to privilege exploration of areas that are relevant (e.g., corridors), as our results obtained with a realistic simulator show.

Third, recognizing that the evaluation of autonomous multirobot systems has not reached a maturity level comparable to that of other disciplines, we provide some tools for improving the evaluation of exploration strategies and we contribute to experimentally evaluate what factors impact the performance of exploration. Specifically, we discuss how to compute the competitive ratio of practically-used exploration strategies given a specific setting. Also, we quantitatively assess in simulation to obtain a significant number of repeated experiments the impact of different perception/decision timings on the performance of exploring multirobot systems and the relative influence of exploration strategies and coordination methods.

We finally show some possible uses of some of the contributions pro-

vided in this dissertation for exploring abstract state spaces in the domain of pursuit-evasion.

The long-term goal is to pave the way towards the theoretical and practical development and the experimental assessment of effective exploration strategies and coordination methods to increase mobile robots autonomy. In an even broader perspective, the future objective is to develop a framework for a more general exploration problem where robots can “explore” other features of the environment, including its physical quantities.

# Sommario

Negli ultimi anni, la robotica mobile autonoma ha acquisito sempre più importanza in attività difficili, pericolose o semplicemente noiose da eseguire da parte degli essere umani, come ad esempio l'esplorazione di pianeti e il soccorso di vittime in ambienti disastrati. Durante la progettazione di un sistema di robot mobili autonomi, il progettista deve risolvere diverse problematiche importanti che includono la locomozione, la percezione, la localizzazione e la navigazione. Uno degli aspetti rilevanti che potrebbe avere impatti significativi sulla performance del sistema riguarda la parte decisionale che permette di compiere l'attività assegnata in modo autonomo. Questo comprende l'insieme di tecniche che permettono ai robot mobili autonomi di decidere la prossima locazione candidata da raggiungere (attraverso una *strategia di navigazione*), e, possibilmente, di coordinarsi tra di loro (attraverso un *metodo di coordinamento*). Tipicamente, queste decisioni vengono prese considerando la conoscenza del mondo esterno che il sistema robotico ha acquisito fino a quel momento. Si prenda in considerazione ad esempio il problema dell'esplorazione, in cui robot mobili autonomi vengono impiegati per scoprire e mappare in modo incrementale un ambiente inizialmente sconosciuto. Questa attività è alla base di molte applicazioni, come ad esempio il *search and rescue*. A partire dall'area di ambiente attualmente conosciuta, i robot devono decidere le prossime posizioni da cui effettuare una percezione e l'allocazione di esse ai vari robot. Nonostante l'importanza dello sviluppo di tecniche che permettono ai robot mobili di essere più autonomi, quest'ultime non sono ancora del tutto mature.

Questa tesi si focalizza su alcuni aspetti chiave del problema dell'esplo-

razione multirobot: la scelta di locazioni interessanti da raggiungere (attraverso una strategia di esplorazione) e il loro assegnamento ai robot (attraverso un metodo di coordinamento). Il lavoro presentato in questa tesi vuole contribuire a tre obiettivi di ricerca: costruire un ponte tra teoria e pratica per le strategie di esplorazione; migliorare le strategie di esplorazione e i metodi di coordinamento usati nei sistemi di esplorazione multirobot; e portare la loro valutazione sperimentale a una maturità comparabile a quella di altre discipline.

Innanzitutto, studiando i lavori presenti in letteratura, si può notare che il problema dell'esplorazione è trattato con due approcci diversi. Da un lato, questo problema è studiato in modo teorico: solitamente, i metodi sono proposti con delle garanzie teoriche, come *bound* e *competitive ratio* nel caso peggiore. Spesso però in questi tipi di lavori, le assunzioni fatte sul modello non sono realistiche. Dall'altro lato esistono lavori che definiscono e testano in modo empirico le tecniche per risolvere questo problema in alcuni contesti pratici, utilizzando robot reali (o realisticamente simulati). Rispetto a questi approcci radicalmente diversi, in questa tesi viene definito il problema di calcolare un percorso offline di esplorazione ottimo per un robot in ambienti rappresentati a griglia, con delle assunzioni più realistiche, ossia visibilità limitata e discreta. I risultati sperimentali mostrano la fattibilità di questo approccio per ambienti realistici. Un secondo contributo in questa direzione è dato dall'analisi teorica di alcune strategie di esplorazione che sono state usate in alcuni contesti reali. In particolare, vengono considerate strategie di esplorazione che valutano il prossimo punto da raggiungere con distanza e guadagno in termini di informazione. Il modello dell'ambiente considerato è basato su grafi. Attraverso quest'analisi, vengono forniti dei bound sulla distanza percorsa, misurata come numero di attraversamenti di archi, per esplorare un grafo generico. I risultati mostrano che, nel caso peggiore, considerando una strategia di esplorazione che considera anche il guadagno di informazione non si ha nessun vantaggio rispetto all'uso di una strategia di esplorazione che considera solo la distanza percorsa, mentre nel caso medio quest'informazione aiuta a migliorare le performance del sistema.

In secondo luogo, in questa tesi, vengono definite delle strategie di esplorazione e metodi di coordinamento che basano le decisioni non solo su informazioni metriche che potrebbero essere ricavate dalle letture dei sensori montati sui robot, ma anche su informazioni semantiche che associano concetti di alto livello ad aree dell'ambiente. Questo risulta utile quando, ad esempio, un comportamento desiderabile da parte dei robot è quello di privilegiare aree da esplorare considerate rilevanti (ad esempio i corridoi). I

risultati sperimentali confermano la bontà del sistema proposto.

Inoltre, riconoscendo che, nonostante in varie discipline dell'ingegneria ci siano solitamente standard condivisi per valutare le performance di un sistema, in robotica, non è ancora stata definita una metodologia standard di valutazione per confrontare sistemi diversi. In questo senso, un altro contributo di questa tesi sta nel fornire uno strumento per migliorare la valutazione delle strategie di esplorazione, per cui viene mostrato come si può calcolare il competitive ratio dato un ambiente specifico. Inoltre, vengono analizzati sperimentalmente alcuni dei fattori che hanno impatto sulla performance dell'esplorazione. Specificatamente, vengono valutati, in simulazione per ottenere un numero significativo di esperimenti ripetuti, l'influenza relativa delle strategie di esplorazione e dei metodi di coordinamento e l'impatto di cambiare le frequenze di percezione/decisione sulla performance di un sistema robotico per l'esplorazione.

Infine, viene mostrato un possibile uso di alcuni contributi dati in questa tesi per esplorare uno spazio più astratto degli stati in un contesto di guardia e ladro.

L'obiettivo di lungo termine è costruire una base verso uno sviluppo e una valutazione sperimentale teorica e pratica delle strategie di esplorazione e dei metodi di coordinamento per migliorare l'autonomia dei robot mobili. In una prospettiva più ampia, l'obiettivo futuro è sviluppare un framework per problemi di esplorazione più generali, in cui i robot "esplorano" altre caratteristiche dell'ambiente, come alcune sue grandezze fisiche.

# Contents

# 1 Introduction

Autonomous mobile robotics has seen a widespread development in recent years, due to its possible applications in everyday life, where the tasks would be boring for humans, such as house cleaning and surveillance. There are some other contexts where robots are required to operate without any human supervision, especially in environments that human beings cannot access because of their asperity and the impossibility of human telecontrol, like in some search and rescue settings (see for example Figure 1.1).

There are several challenges that a designer faces during the development of autonomous robots, from low level issues – e.g., sensors, actuators – to high level issues – e.g., control [116]. One of the most important aspects that affects autonomous mobile robots performance is the decision making for autonomously carrying out the assigned tasks. We refer to the latter as the set of techniques that allow autonomous mobile robots to decide the next location to reach (usually called *navigation strategies*) by possibly coordinating among themselves (typically with a *coordination method*), according to their current knowledge about the world they operate in.

To introduce the idea of navigation strategies, it is useful to start from considering the (huge) literature about path planning, which shows that most of stable methods have been developed in mid-1990s. In these approaches, users specify the goal and the robots can decide by themselves how to go there, even if the space between start and target poses is unknown (e.g., using algorithms like D* [76, Chapter 12], Learning Real-Time A$^*$ [112, Chapter 4], and PHA$^*$ [45]).

However, in several cases the goal might not be known *a priori* or the user cannot interact in real-time with the robots, and so these methods can-

**Figure 1.1:** *A robot searching for victims at RoboCup 2014 Rescue competition.*

not be plainly used. Consider for example a scenario in which a robot is given the high-level task of finding the keys of the user's car in a house. In such a case, the robot should be able to decide by itself where to look for the keys, as otherwise it would not ease the task to the humans. Thus, in general, what is needed is the development of navigation strategies that allow mobile robots to *autonomously* decide about their next target locations, besides how to go to a specific target [9]. Furthermore, the use of multiple robots can make the execution of the task more efficient, if they smartly coordinate among themselves. Consider for example a scenario where robots are autonomous lawnmowers: with more robots the lawn can be cut in less time if the robots evenly spread over the area. Both aspects of decision making, namely navigation strategies and coordination methods, would allow to augment robots' autonomy and efficiency.

An important task, where autonomous mobile robots could be fruitfully employed, is the *exploration problem*. The exploration problem has been studied in connection to autonomous mobile robots able to explore unknown environments and build a map of them. The map usually includes information about the spatial features of the environment discovered so far, for example the positions of the obstacles. There are several applications that benefit from this task, e.g., planetary exploration [100] and search and rescue [123].

This dissertation focuses on the exploration problem, and specifically is about the study, the design, and the evaluation of techniques for exploring initially unknown environments by one or more mobile robots, in order to discover and map their features.

## 1.1 The exploration problem

To better focus the problem we address and to frame the contributions of this dissertation, we provide a very general abstract model of the behavior of fully autonomous mobile robots while executing exploration tasks. Schematically, the main steps that robots usually perform are the following:

(a) perceive the surrounding environment,

(b) integrate perceived data in a map representing the portion of the environment known so far,

(c) decide where to go next (*exploration strategy*) and who goes where (*coordination method*),

(d) go to the destination locations chosen,

(e) return to Step (a).

Although over-simplified, the above model evidences some interesting issues of the exploration problem. Step (d) involves low-level planning (e.g., path planning and localization), while Step (a) relates to the acquisition of data coming from sensors mounted on robots, like laser range scanners, cameras, or sonar sensors. Step (c) regards the decisions that the robots can take autonomously. As already said, we will refer to it as the set of techniques that allow autonomous mobile robots to answer the question ‘where to go next?’ (exploration strategy) and ‘who goes where?’ (coordination method), given the knowledge robots possess so far thanks to Step (b). Typically, robots consider candidate destination locations on the boundary between the known and unknown portion of the environment, as those are the locations where the discovery of new portions of the environment is increased. These locations are usually called *frontiers*. Note that a coordination method is not necessary when just one robot is employed. The above cycle of steps terminates according to a termination criterion, usually related to time or to the amount of discovered area (we show some examples in Section 1.3). The focus of the whole dissertation is on Step (c) and the other steps are taken as given from methods from the state of the art.

In general, as already said, exploration strategies and coordination methods significantly impact on the task execution's performance. Therefore, the problem is to define *good* exploration strategies and coordination methods, i.e., techniques that allow robots to perform their task maximizing (or minimizing) some performance metric or optimality criterion. However, there are two major challenges that make the development of good techniques difficult. Consider, as an example, the situation represented in Figure 1.2, where the mission is to build the map of the environment. First,

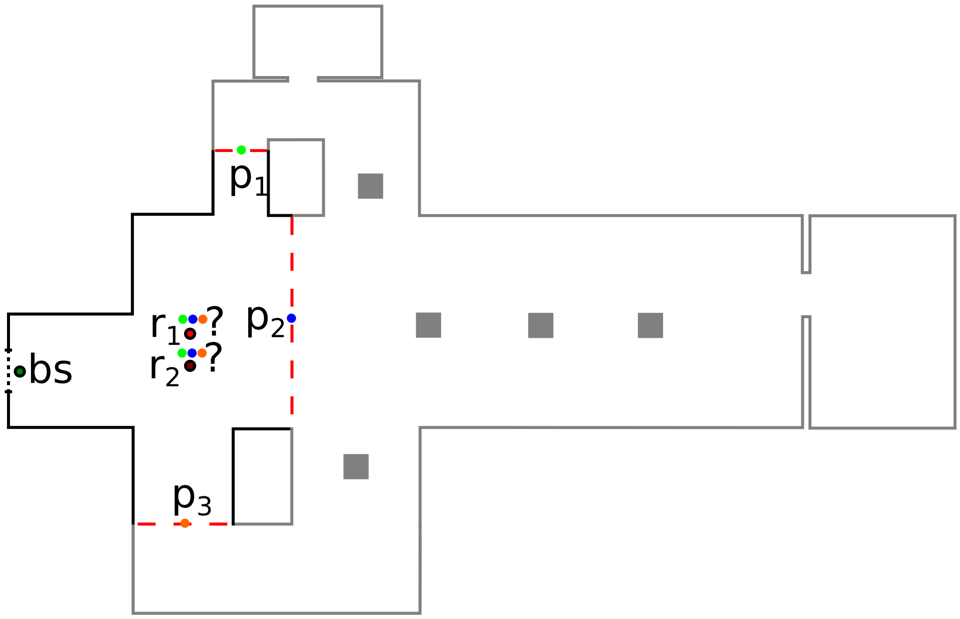


**Figure 1.2:** *A partially explored environment by robots $r_1$ (red dot) and $r_2$ (scarlet dot). Black line segments are the part of the environment known by the robots, while the grey ones are still unknown to the robots. The dotted black line segment represents the door where the robots entered in the environment. Dotted red lines are the frontiers between known and unknown portions of the environment. The robots $r_1$ and $r_2$ have to decide which location they should reach for performing the next perception among $p_1$, $p_2$, and $p_3$ (green, blue, and orange dots, respectively), possibly coordinating among themselves through a base station bs.*

univocally defining an optimality criterion that measures the global goodness of an exploration system could be not trivial, as it is strongly dependent on the mission assigned to the robots. For example, in the mapping mission represented in the figure, an optimality criterion could be the resulting quality of the map, while in a search and rescue mission could be the amount of area covered in a given time interval. Moreover, within the same mission, it is possible to consider different optimality criteria. In the example in the figure, besides the quality of the map, the time or traveled distance to map the whole environment can be considered as optimality criteria. However, optimality criteria could be conflicting, even in cases when only some of them are explicitly considered. Imagine that it has been decided that for the scenario represented in the figure the optimality criterion to be optimized is the exploration time. Nevertheless, the robots need a map of the environment without too much noise, as, otherwise, they cannot efficiently plan on it. This implies that an acceptable quality of the map is implicitly required. When there are conflicting optimality criteria (typically those that measure benefits and costs), a trade-off between them should be addressed. The definition of a global optimality criteria influences the design of the exploration system and is usually achieved by defining local criteria that should be optimized at each step of exploration. According to the mainstream approach of exploration strategies, the robots could pick a set of candidate locations, usually obtained by sampling the frontiers between known and unknown space (in the example reported in the figure, sampled candidate destination locations are $p_1$, $p_2$, and $p_3$) and evaluate them with a utility function $u()$. The utility function $u()$ attempts to capture the global optimality criteria, by combining different local evaluation criteria. For example, evaluation criteria used in $u()$ could include the traveling cost for moving from the current position of the robot to the candidate location and the information gain that is expected to be obtained at the candidate location in the attempt of optimizing the global exploration time, but also the overlap between the new perception at the candidate location and the already known portion of the environment, to have a map with better quality. Hence, the trade-off between benefits and costs should be addressed in the definition of $u()$. For the same very reason, namely the conflicting performance metrics, it is hard to univocally define coordination methods. For example, if the time to map the environment is considered as global metric, then it is expected that robots spread over the environment (e.g., $r_1$ and $r_2$ select $p_1$ and $p_3$, respectively). In other cases, where the objective is to improve the quality of the map, the coordination method should prefer to send more than one robot to the same candidate location.

The second challenge (which is strongly connected to the previous one) derives from the fact that typically in exploration no *a priori* knowledge is available. This leads to methods that exploit the current partial knowledge of the environment to decide where to go and who goes where, and thus leading to a local optimization of the process, as *next-best-view* approaches briefly described above do. In particular, the selection of next destination locations is performed at each step of the exploration process, by selecting the candidate location where $u()$ is maximized (or minimized). However, the solution found by such local optimization clearly could be not globally optimal. For example, consider again the scenario depicted in Figure 1.2 with a single exploring robot $r_1$. To optimally explore the environment by minimizing the traveled distance, $p_3$ should be selected first, leading the robot to explore the whole loop corridor on the bottom of the figure, then the corridor on the top, and finally the corridor on the right (assuming that the range of the sensors has a value that allows to cover the width of the corridor). Note that this reasoning has been possible because of the *a priori* knowledge of the full environment. However, considering the online scenario of the figure, for example, a utility function that uses the expected information gain as evaluation criterion, measured as the length of the line segment that represents the frontier, would lead to the selection of $p_2$ first, leading to a sub-optimal solution. This challenge is evident also for the coordination methods in case of multiple robots. An optimal coordination method would pick $p_1$ and $p_3$ first and assign them to $r_1$ and $r_2$. Instead, an online coordination method that optimizes the sum of the utilities $u()$ computed by an exploration strategy that considers again the expected information gain measured as described above would choose $p_2$ and $p_3$, again leading to a sub-optimal solution.

The above discussion highlights some of the critical issues and of the difficulties of the exploration problem that is addressed in this dissertation.

## 1.2 Motivation and objectives

In spite of the importance of the exploration problem, general techniques that allow mobile robots to be fully autonomous in performing the exploration task are not mature yet.

Exploration strategies are usually defined following two rather different approaches. On the one hand, exploration strategies are defined in theoretical settings. The environments could be represented geometrically, as usually done by computational geometry community. In this kind of works, some assumptions on the capability of the robots are made, like line-of-sight visibility [43] or continuous perception [38]. Another type of representation is based on graphs, as typically assumed by the theoretical computer science community to disregard environments' geometry for focusing on their topological and combinatorial aspects. In both approaches, proposed methods are assessed using theoretical tools like worst-case bounds [127] and competitive ratio [104] in some classes of environments. On the other hand, exploration strategies are defined in practical contexts of real (or realistically simulated) robots, as for example in the work of Julia et al. [67]. In such a situation, the performance of an exploration strategy cannot be guaranteed to be optimal, and, although many solutions have been proved effective in practice, it is difficult to compare different exploration strategies with the aim, for example, of selecting the best one for a given situation in which they have never been tested. This situation demands a more theoretical-oriented analysis of the current practical exploration strategies or, equivalently, for a more realistic account of theoretically-defined exploration strategies.

Further, most of the exploration strategies and coordination methods proposed in literature (e.g., [16, 56, 120]) base their decisions only on the current *metric map*, which represents the spatial features of the environment, like the position of obstacles. In the last years, several methods have been proposed to build *semantic maps* of environments (e.g., [89]), which label some spatial elements with high-level human concepts. For example, areas of a metric map can be labeled as ‘corridor’ or ‘room’. Despite the great effort in *constructing* semantic maps, the study of their *use* for exploration is still rather limited and can represent a way to enhance the process.

Finally, a lively debate on good experimental methodologies is currently ongoing in the autonomous robotics community, as the field has not reached yet a maturity level comparable to that of other disciplines [5]. One of the problems is the difficulty in reproducing experiments, as parameters are usually not fully reported in the descriptions of experiments, and thus it is not clear what factors impact the performance of exploring robotic systems.

In the general context of the multirobot exploration problem, the objective of this dissertation is thus threefold:

1. to bridge the gap between theory and practice for exploration strategies, by considering more realistic assumptions and providing methods to compute the optimal coverage path and bounds for some exploration strategies;

2. to improve exploration strategies and coordination methods, by considering also semantic information when making decisions;

3. to improve the experimental assessment of multirobot exploration systems, by providing a tool for computing the competitive ratio of exploration strategies in a given environment and by assessing the impact of some controllable parameters of the autonomous robotic system.

## 1.3 A framework for exploration

In general, the exploration problem, solved by one or more robots, is composed of different dimensions that could be controlled (to some degree) by the robot designer. Here we discuss some of those relevant to exploration, being aware that the list is far from being definitive and complete, in order to set the framework in which the contributions of this dissertation are inserted (some of these dimensions have been presented in [9]).

About robot's capability, we have that each robot has a *sensor model*, namely a model that represents the hardware constraints of a sensor (used for Step (a) of the behavior model of the robots presented in Section 1.1). For example, a camera is usually modeled as if it has an infinite line-of-sight visibility. Furthermore, robotic designers can decide its *perception model* that refers to Step (b), i.e., the integration of acquired data about the surroundings in the current map. That is, the current map can be refreshed when sensors data are received (continuously or at a given frequency) or when an event happens (discrete or event-based). There are two other dimensions that relate to Step (c). One is what we call *motion model*, which determines what reachable points are considered as candidate destination locations. We call continuous (discrete) motion model, when all (a subset of) the points of the current known portion of the environment are considered as possible destination locations. Moreover, let us define the *decision model* that accounts for how frequently decisions are revised: if continuously (or at a given frequency), the robot can possibly revise the decision along the path to a previously selected location, while, if discrete, a triggered event determines the computation of another candidate destination location (e.g., when a selected location is reached). Also, the decision on the next candidate location can be taken with a *lookahead* or not, namely a robot plans several steps ahead or just one step at a time. Note that these models are not entirely controllable by the robot designer, as some of them depend on the hardware of the robot (e.g., the sensors used, the computation power available on-board).

The *number of robots* is another dimension that is controllable by the robot designer: the robotic system can be composed of a single robot or multiple robots, which can be of the same type (*homogeneous*) or different types (*heterogeneous*). When multiple robots are employed, another dimension regards the constraint on the communication between robots, which is modeled by a *communication model* (e.g., line-of-sight, distance-based, obstacle-based). Note that, in general, the presence of multiple agents introduces another dimension represented by the *adversarial nature* of the setting, which is related to the possible presence of adversaries. Nevertheless, in the exploration problem, robots are usually designed in such a way that they cooperate to make a globally optimal decision.

The current environmental map, which the robots can reason on, can have different *representations*, namely it could be represented by the robots in a *continuous* (e.g., polygonal representations) or a *discrete* (e.g., grids or topological maps) way. Also, the robots could have some *a priori* knowledge on the environment to be explored. If the environment (or, more precisely, all the information needed for selecting the next candidate destination location) is fully known in advance, robots have a *global knowledge*. Conversely, if only partial or no environment's information is initially available, robots have a *partial knowledge* and should increase their knowledge on the environment to make more informed decisions.

The availability of global knowledge results in the possibility of computing the exploration strategy *offline* and, possibly, searching for an optimal solution, as the locations to reach are known in advance and all possible paths can be computed before robots actually employ the strategy. On the other hand, a partial knowledge is typically associated with the use of an *online* exploration strategy that makes decisions according to available information and that results in sub-optimal performance. To make this point clearer, consider the two common tasks of coverage and exploration. In coverage [31], the environment is known in advance and robots should cover (possibly, under some constraints) all the free area with the footprint of their sensors. In exploration, the environment is unknown at the beginning and robots have to "discover" it. The first case is characterized by a global knowledge and the optimal strategy, e.g., the shortest route, can be computed offline. The second case is an example of partial knowledge situation for which decisions have to be made online, due to the difficulty to reliably predict the states that robots will face. The optimal strategy cannot be found in general and sub-optimal greedy algorithms (e.g., next-best-view approaches) must be employed. Note that in both cases the optimal strategy can be the one that minimizes the traveled distance.

Another characterization of the knowledge is given by the *kind of knowledge* used by robots to select next destination locations. This can be of different natures. A specific kind of knowledge induces how decisions are made. In exploration, criteria used in the utility function that evaluates candidate destination locations usually derive from the robot's knowledge about the spatial features that has been collected in the *metric map* built so far. However, in principle, there are other types of knowledge that can be used, like high-level information coming from human users. An example considered in this work includes a *semantic map*, which labels some spatial elements with high-level human concepts.

As said before, the *optimality criterion* is a crucial dimension for the design of exploration strategies and coordination methods, which are eventually evaluated according to the selected optimality criterion. It is strongly dependent on the *mission* that exploring robots should perform. For example, it is based on the *termination criterion* adopted to consider the mission accomplished. In a search and rescue setting, a typical termination criterion is the elapse of a time interval $T$ that represents the time available to the robots to explore a given environment. The optimality criterion in such a case is usually the amount of explored area. In a mapping application, a common termination criterion is the percentage $\mathfrak{g}$ of the area to explore and the optimality criterion is the time required to accomplish the mission. As already said, sometimes also costs are included as optimality criteria, such as the traveled distance or the number of perceptions (or steps) to perform.

Note that sometimes the termination criterion includes the fact that, when the task is terminated, the robots are asked to go back to the starting point (*tour* or closed path), or not (*path*). The latter is often considered, for example, in mapping or search and rescue scenarios where “coming back” might not be subject to optimization, e.g., it can be performed in a second stage after the more important exploration task is completed.

All these dimensions influence the design of the exploration strategies and coordination methods and the kind of evaluation to assess the performance of exploring robotic systems.

## 1.4 Original contributions

The original contributions of this dissertation can be divided following the three goals presented in Section 1.2. Specifically, on the theoretical side:

- We contribute to define the problem of calculating the optimal offline exploration paths under realistic assumptions (some of the results are presented in [104]) – i.e., considering the problem in grid environments for a robot with time-discrete and limited perception. The problem turns out to be a constrained variant of the classical coverage problem. We analyze the relation between our discretization and the continuous counterpart and formulate the discrete problem as a search problem. Thus, we develop and test the first algorithm to find the (approximated) optimal exploration path under such realistic assumptions.

- Considering environments modeled as graphs, we theoretically analyze some exploration strategies that are typically used by robots in practical settings and that consider distance and information gain as criteria in their utility functions (some of the results are presented in [110]). Specifically, bounds are provided for the worst-case number of edge traversals for exploring generic graphs and for the average-case number of edge traversals for exploring a specific class of graphs representing indoor environments.

On the practical side, the main original contribution is given by our definition of exploration strategies and coordination methods that embed information coming from semantic maps. We experimentally show that our approach provides a significant improvement in the exploration of *relevant areas* of indoor environments, when *a priori* information about such relevant areas is available (preliminary results have been presented in [33]). For example, if robots know that an area of an environment is labeled as 'corridor', then that area could be privileged by sending there more robots, as rooms are typically attached to corridors.

Further, we contribute in evaluating exploration strategies in a given environment and in identifying some of the factors that impact the performance of exploration (some results are presented in [7, 8, 104]):

- We show that the computation of the offline exploration path can be used for calculating the competitive ratio of exploration strategies in a given environment.
- We experimentally evaluate (in simulation) the impact of tuning perception/decision timings on the performance of multirobot exploration systems.
- We assess in simulation the relative influence of exploration strategies and coordination methods on the performance of the exploration process.

Finally, we show how some of the results obtained can be used for other purposes. Specifically, we show that some of the artificial intelligence techniques used in this dissertation can be used for exploring different state spaces in the context of pursuit-evasion games, in which a pursuer attempts to capture an adversarial evader that tries, in turn, to actively escape, when they both have a line-of-sight sensor model.

Note that, despite the importance of experimenting with real robots, to obtain a large amount of results that are easily reproducible, we usually performed experiments with realistic simulators.

## 1.5 Document structure

This document is structured as follows. In Chapter 2, we introduce the current state of the art about the exploration problem, focusing on both theoretical and practical works available in literature, and on the current experimental methodologies. Then we divide the document in four parts describing our contributions related to the three goals we defined for exploration and to some uses of some of the contributions outside the exploration problem.

Part I encloses the contributions for bridging the gap between theory and practice for exploration strategies. In Chapter 3, we formulate the discrete version of the offline exploration problem, showing its relation with the continuous counterpart. Also, we formulate it as a search problem, propose an algorithm for solving the problem, and report our experimental results. In Chapter 4, we formulate the exploration problem on a graph and theoretically analyze some exploration strategies that are typically used in realistic settings.

Part II encloses the contribution on improving practical multirobot exploration systems. Chapter 5 shows the formulation of the exploration problem in a search and rescue setting and Chapter 6 presents the proposed multirobot exploration system and its validation through extensive experimental simulated activities that display the effectiveness of the proposed approach.

Part III encloses the contributions on improving the experimental assessment of multirobot exploration systems. Chapter 7 shows how some of the contributions presented in Part I can contribute to better evaluate exploration strategies in some given settings. Referring to the model shown in Chapter 5, Chapter 8 presents experimental results on the study of the impact of perception/decision timings and Chapter 9 shows experimental results that indicate the influence of exploration strategies vs. coordination methods in some settings.

Part IV encloses a possible use of some of the contributions presented in this dissertation. Chapter 10 presents how some of the artificial intelligence techniques used for exploration can be used to explore a different, more abstract, state space in the context of pursuit-evasion games.

Chapter 11 concludes this dissertation and suggests further directions of research.

# 2 Related work

As anticipated in Chapter 1, the exploration task is the process in which one or more robots deployed in an initially unknown environment have to discover and map its features. This task is carried on in missions like map building [124], search and rescue [123], and coverage [31]. For example, in map building the features to be discovered can be the obstacles and the free space, while in search and rescue they can be the locations of victims or sources of danger (e.g., fires).

In this chapter, we report the current state of the art in the exploration problem. Specifically, in Section 2.1, we present a representative sample of works that theoretically analyzed the exploration problem, while, in Section 2.2, we survey those ones that proposed more practical approaches and tested them in real (or realistically simulated) settings. Finally, Section 2.3 shows some aspects of the current experimental methodologies for assessing the performance of robotic exploration systems.

## 2.1 Theoretical contributions

On the theoretical side, works can be classified according to the abstraction level at which environments are represented. In the following, we present works from the computational geometry community and those from the theoretical computer science community. The first one considers geometrically-represented environments (e.g., rectilinear polygonal environments) and the exploration task can be formulated as the coverage of the entire area of a polygon through a sensor or a tool with a given footprint. The latter, instead, usually takes into account more abstract graph-based representations of environments that disregard their geometry to focus on their topological and combinatorial aspects. The exploration task can be formulated as that of visiting all the edges or all the vertices of the graph.

### 2.1.1 Computational geometry approach

In computational geometry there are several problems that could be related to the exploration problem we consider in this dissertation. For each one, we show a representative sample of proposed approaches by works in the computational geometry community (some surveys are presented by Ghosh and Klein [54], Isler [64], Mitchell [87]). We also comment on how different problems typically solved in computational geometry can relate to the exploration problem. Note that, following the dimensions outlined in Section 1.3, all the works in computational geometry consider a continuous representation of the environment. Also, the type of knowledge that is used by the robot is metric-based.

One of the problems that has been studied for long time is the *watchman problem*. The watchman, which could correspond to a mobile robot, has to find a shortest tour (i.e., cycle) within an *a priori* known polygonal domain $P$, such that every point of $P$ is seen by some point of the cycle. The sensor is assumed to be modeled by a line-of-sight visibility, namely all the points that belong to the (possibly infinitely long if no obstacle is hit) lines of sight starting from the robot sensor are covered. The robot motion and perception models are continuous. The optimality criterion considered is the distance the robot has to travel to accomplish the task that should be minimized (measured either as Manhattan distance $L_1$ or Euclidean distance $L_2$). Chin and Ntafos [30] present an approach that first partitions the rectilinear polygon representing the environment (without obstacles) into rectilinear monotone polygons (in the paper they consider monotonicity, without loss of generality, with respect to y-axis, i.e., every orthogonal line to y-axis intersects the polygon at most twice). Then, it finds connections between adjacent partitions by constructing a tree and following an Euler tour of the tree that minimizes the traveled distance. The algorithm finds the shortest watchman tour (measured in $L_2$) in $O(n \log n)$, where $n$ is the number of edges of the polygonal environment. The work of Carlsson et al. [26] exploits the concept of essential cuts to find a watchman tour (whose length is measured in $L_2$) in polygons without holes (i.e., without obstacles). Specifically, an essential cut is an extension cut which is not dominated by any other extension cut. An extension cut $c$ is a segment that starts from a reflex vertex (a vertex with its internal angle strictly greater than $\pi$) along the direction of the corresponding edge and ends at the boundary of the polygon, and that entirely lies in the interior of the polygon. An extension cut $c$ is not dominated by any other extension cut $c'$ if all points of the polygon on the left of $c$ are also to the left of $c'$. The authors show that a closed curve is a watchman route if and only if the curve has at least one point to the left of (or on) each essential cut. The proposed algorithm computes the shortest watchman route in $O(n^6)$ in polygons without holes.

Note that works employing continuous perception and using traveled distance as optimality criterion are basically solving the exploration problem, because the knowledge of the environment is continuously updated and so the robot always stays within the current known portion of the environment. The comparison with an exploration strategy, however, could be not fair as the algorithms solving the watchman tour problem can exploit the *a priori* knowledge over the environment.

The online version of the watchman route problem (where no map is given at the beginning of the task) is first studied by Deng et al. [38]. The proposed approach exploits the concept of essential extensions. These are line segments $l$ computed extending all sides $S$ of a polygon $P$ (representing the environment) until $l$ first hit the boundary of the polygon (line segments $l$ do not include sides segments $S$). The authors demonstrate that the robot has to touch all essential extensions for completely covering a rectilinear polygon without holes with a (possibly closed) path. Given a robot initial position, the algorithm greedily finds a point $p$ for each essential extension in such a way that the length of the line segment between the robot current position and $p$ that touches one essential extension is minimized. The algorithm can find an optimal solution measured in $L_1$ in $O(nm^2)$ for rectilinear polygons without holes with $n$ sides and $m$ essential extensions.

All above works make assumptions that are rather far from practical robotic implementations, because they do not take into account constraints imposed by the hardware equipment. For example, they consider infinite visibility, while real laser range scanners or cameras (let alone other mechanical covering tools) do not have infinite operative range. Also, (close to) continuous perception is typically non-affordable on real robots because of the limited amount of on-board computing capability which does not allow to continuously process data coming from sensors.

Some works have tried to extend some of the results of the offline watchman route problem (where visibility range is infinite) to the case in which sensor range is limited. This problem has been firstly tackled by Ntafos [95]. The authors call *$d$-watchman* problem the one whose goal is to find the shortest tour for a watchman with a visibility range $d$, such that every point of the boundary of the polygon without holes is visible along the tour.

The more closely related problem to the exploration is called $d$*-sweeper*, where also the points in the area of the polygon should be visible along the tour. The proposed approach to solve the latter problem superimposes a grid to the polygon (removing portions of the grid that lie beyond the boundaries of the polygon) and then finds an approximate TSP (Traveling Salesman Problem) solution on the resulting graph connecting the adjacent cells of the grid. (The TSP solution is the shortest tour that visits all cells exactly once.) The proposed algorithm obtains solutions within $33\%$ of the cost of the optimal solution (measured in $L_1$) in polygons without holes if the sensor range is enough small with respect to the size of the polygon. The $d$-sweeper problem is sometimes called *lawn mowing* problem, as for example in the work of Arkin et al. [10]. The proposed algorithm uses a grid of squared cells and constructs a tour of length at most $2.5$ times the length of the optimal tour in a time $O(n \log n)$, where $n$ is the number of edges of the polygonal environment, which is assumed to be rectilinear and without holes. (A variant of the *lawn mowing* problem is the *milling* problem, in which there is an additional constraint on the area covered by the sensor that should strictly lie within the boundaries of the polygon.) Similarly, the work of Gabriely and Rimon [50] considers the coverage problem (which is basically the lawn mowing problem) and approximates the environment with a grid of squared cells (whose size is equal to the size of the tool covering the area), constructs a spanning tree, and generates a covering tour, which is basically a Hamiltonian cycle visiting all cells of the grid. (An Hamiltonian cycle is a tour that visits each vertex exactly once.) The proposed algorithm finds a covering tour in $O(N)$, where $N$ is the number of cells that approximate the environment. This method considers distance in $L_1$ as optimality criterion, but actually the algorithm avoids repetitive coverage attempting to minimize the traveled distance. Simulation results show that the algorithm is effective when the tool size is significantly smaller than the work-area characteristics. Fazli et al. [42] address the problem of repeated coverage of a known polygon by a robot with limited sensor range. Broadly speaking, the approach is the following: (a) the environment is segmented by a trapezoidation method [138], (b) guards (which correspond to scan points) are placed in the derived subpolygons in order to entirely cover them, and (c) a TSP is solved in order to connect all the guards. The proposed approach finds optimal solutions in terms of total path length for the robot. Authors experimentally show the effectiveness of the algorithm in cases where the tool size is significantly smaller than the work-area size. In the work of Mannadiar and Rekleitis [82], boustrophedon cellular decomposition is used to create a partition of a polygon (possibly with holes), from which a graph is extracted. From that graph, the solution to the Chinese postman problem is calculated. (This solution finds the shortest closed route such that each edge is visited at least once.) Each partition is then covered using simple motions, like back-and-forth movement. This approach has been tested in simulation in some arbitrary environments and also with a fixed wing UAV [135], showing its effectiveness. Fekete et al. [44] deal with the *myopic watchman problem with discrete vision* problem which is basically the same as the coverage problem. The proposed approach uses a linear function that combines traveled distance (in $L_1$) and number of steps (i.e., scan points). The proposed algorithm basically works by building boundary tours to cover the area close to the boundary and using strips to cover the interior. Then, some additional paths are added in order to merge the boundary tour and the strips. Finally, scan points are placed at fixed distance in such a way that all the area is covered. The algorithm constructs a tour, whose approximation is at most $\pi(\frac{r}{a} + \frac{r+1}{2})$, where $r$ is the sensor range and $a$ is the minimum edge length of the polygon. Note that the number of steps optimality criterion, where a step could be a turn or the integration of sensor data in the map, represents a measure that is relevant for real robot applications because of the limited availability of computational power and the time requirements of these applications. The work by Arkin et al. [11] uses a different approach by giving approximation algorithms for optimal covering tours and mainly focuses on minimizing the number of steps (or the linear combination of traveled distance and number of steps), where a step is a turn of the robot in rectilinear polygons with holes. The algorithms are based on covering the region with strips connected into cycles and provide constant-factor approximations (e.g., $3.75$-approximation for minimum-turn axis-parallel tours for a unit square cutter covering an integral orthogonal polygon with running time of $O(N^{2.376} + n^3)$, where where $N$ is the number of cells representing the environment and $n$ is the number of edges of the polygon).

Almost all the works presented above consider tours (closed paths) which, as said, in many practical applications, are not necessary. Moreover, some of the above approaches use $L_1$ as optimality criterion for the traveled distance, which is not fully realistic, as robots can usually rotate arbitrarily. Relatively few works consider $L_2$ or include also a criterion that consider the number of steps needed to cover an environment. Furthermore, most of the above works dealing with the watchman route problem (or with its variants) consider simple environments, namely polygons without holes, which are not representative of realistic environments, like city blocks and cluttered offices. Note that all the works we are aware of that deal with the watchman route problem with limited visibility assume to have *a priori* knowledge on the environment.

In computational geometry, several works deal with another long-studied problem: the *art gallery problem*, that is to place guards, minimizing their number, in an environment to completely guard its area. The work of Kröller et al. [73] solves the art gallery problem using an approach based on linear programming. Specifically, the authors formulate the art gallery problem as a relaxed linear program (primal problem), where guards are taken from a finite set, and convert the primal problem in the dual linear problem. Moreover, they formulate the separation problem for both the primal and the dual linear programs. The proposed algorithm runs iteratively, by solving the primal and dual linear programs, whose solutions are checked with the primal and dual separations, until the feasible solutions of the primal and the dual problems converge or the primal and dual separations fail. Experimental results in some environments show the viability of the approach even if, in some settings, solutions do not converge. Another work [19] uses an edge covering algorithm on the polygon representing the environment. If the solution found does not cover the whole interior of the polygon, the algorithm modifies the solution, by checking other solutions returned by the edge covering algorithm or by placing more sensors in the centroids of uncovered regions to obtain full coverage. Experimental results show that the cost of the solution found is close to the lower bound on the cost of the solution obtainable in a specific polygon. In the work of González-Baños [57], a randomized strategy is used to initially place guards and then a greedy algorithm minimizes their number by removing some of them in such a way that the total coverage of the area is preserved. The number of guards is not guaranteed to be minimum, but the authors show that with high probability the difference from the optimal minimum number of guards $c$ is $O(\log(n+h) \cdot \log(c \log(n+h)))$, where $n$ is the number of edges of the polygonal environment and $h$ the number of holes in the environment.

It seems that the works dealing with the art gallery problem could be related to the exploration problem. The reason is that minimizing number of guards could be equivalent to minimizing the number of perceptions (steps) that a robot has to perform. However, the solution found is not directly applicable to the exploration problem where a robot has a limited visibility. The reason derives from the fact that no constraint is imposed on the sensor range, which is infinite, although the perception can be considered time-discrete. Also, typically guards are placed in such a way that a guard in a point $q$ could be not visible from any other guards in points $q'$, thus not preserving the inherent constraint in the exploration problem, namely traveling within the current known portion of the environment. Indeed, actually, no path is involved in this problem. Note also that, as the watchman route problem, the environment is assumed to be known. Hence, they give an underestimate of the number of steps required to entirely explore an environment when considering the limited sensor range.

Summarizing, we classify the works described above, according to the following dimensions and report them in Table 2.1:

- Solution: the solution can be a closed tour (T), a path (P), or non-feasible in our exploration setting (x) (note that the latter refers to works dealing with art gallery problem where no path is present, but could be similar to the problem in which we have to minimize the number of steps).
- Perception: the timing of perception can be continuous (C) when perceptions are performed continuously while the robot moves along a trajectory, or discrete (D) when perceptions are performed only at selected scan points.
- Sensor range: the range of the covering tool or sensor can be infinite (I) when all the points belonging to the (possibly infinitely long if no obstacle is hit) lines of sight starting from the robot sensor are covered, or finite (F) when the points belonging to the lines of sight starting from the robot sensor are covered up to a fixed maximum range.
- Optimality criterion: the criterion that is optimized can be traveled distance in $L_1$ or $L_2$, or number of steps (e.g., scan points or turns).
- Type of environment: for example, polygons that are rectilinear and with/without holes (obstacles).

Some variants of the above problems have been also investigated. The watchmen route problem where more than one watchman is employed has received relatively little attention so far. Carlsson et al. [25] deal with the $m$-watchmen route problem, where $m$ watchmen should see every point of a polygonal domain. Rekleitis et al. [109] present a boustrophedon approach to compute a path for multiple robots that have to completely cover a polygonal environment. The approach by Fazli et al. [42] presented above also works in the case of multiple robots, by applying a clustering algorithm on the dual graph of the decomposition, so that each cluster is assigned to a mobile robot. Also, variants which consider some additional constraints have been studied. For example, the work of Shnaps and Rimon [115] studies the tethered coverage, where a robot, fixed to a point with a tether of length $l$, has to cover the entire environment.

| Paper | Solution | Perception | Sensor range | Optimality criterion | Environment |
|---|---|---|---|---|---|
| [30] | T | C | I | $L_2$ | rectilinear, without holes |
| [26] | T | C | I | $L_2$ | without holes |
| [38] | P | C | I | $L_1$ | rectilinear, without holes |
| [95] | T | C | F | $L_1$ | without holes |
| [10] | T | C | F | $L_2$ | rectilinear, without holes |
| [50] | T | C | F | $L_2$ | with holes |
| [42] | T | D | F | $L_2$ | with holes |
| [82] | P | C | F | $L_2$ | with holes |
| [44] | T | D | F | $L_1 + \#$ steps | with holes |
| [11] | T | C | F | #steps<br>$L_2 + \#$ steps<br>("steps" corresponds to "turns") | rectilinear, with holes |
| [73] | x | D | I | # steps | with holes |
| [19] | x | D | I | # steps | with holes |
| [57] | x | D | F | # steps | with holes |

**Table 2.1:** *Classification of representative papers from computational geometry on problems that can be related to exploration and coverage according to several dimensions.*

### 2.1.2 Theoretical computer science approach

In the theoretical computer science community, the exploration task is usually formulated as follows: a robot has to explore an initially unknown graph $G = (V, E)$, where the vertices $V$ correspond to the locations where it can move and the edges $E$ represent the direct connections between these locations. During the exploration process, the robot uniquely learns of each vertex (and of the corresponding incident edge) adjacent to a vertex that it visits. This scenario is called *fixed graph scenario*.

There are typically two variants of the exploration task studied with a graph-based representation of the environment. One is that of visiting all the edges. Deng and Papadimitriou [37] show that in Eulerian graphs (namely those that contain cycles which use a graph edge only once, also called Eulerian cycle) the minimum competitive ratio for any exploring algorithm is 2. Furthermore, they show that, for non-Eulerian graphs, this ratio is unbounded when the deficiency of the graph (i.e., the number of edges to be added to make the graph Eulerian) is unbounded. The authors propose an algorithm that explores a graph with deficiency $d$ using $2^{O(d \log d)}$ edge traversals. Panaite and Pelc [97] investigate the impact of the amount of *a priori* knowledge of the graph on the exploration performance. It is theoretically shown that the best exploration algorithm is 2-competitive (considering the number of edge traversals to explore the graph) when no *a priori* knowledge is available about the graph. The work by Fraigniaud et al. [48] considers a slightly different problem, in which the goal is that of visiting all edges, but the robot is not required to stop. Instead of considering the exploration time complexity, for example the number of edge traversals required to explore the whole graph, the authors theoretically look at the minimum memory size of the robot required to explore a graph. They prove that the worst-case space complexity is $\Theta(D \log d)$ to explore all graphs of diameter $D$ and maximum degree $d$. (The diameter of a graph is the length of the "longest shortest path" between any two graph vertices; the maximum degree refers to the maximum number of incident edges to a graph vertex.)

The second variant of the graph-based exploration task is that of visiting all the vertices of a graph. Usually, the optimal solution is calculated as a minimal-length tour that visits all the vertices of an undirected graph and returns to the starting one. One of the first algorithms devised for such online Traveling Salesman Problem is called Nearest Neighbor, which always chooses to move to the closest unvisited vertex. Rosenkrantz et al. [111] prove that such algorithm is at most $\log(|V|)$-competitive (where $|V|$ is the number of vertices in the graph), meaning that the ratio between the length of the solution it produces and that of the optimal solution is at most $\log(|V|)$. Kalyanasundaram and Pruhs [68] present a variant of the Depth-First Search algorithm, called ShortCut, which finds an exploration tour visiting all the vertices of an edge-weighted connected graph. The searcher operates under the fixed graph scenario and moves (traveling on known edges) to unvisited boundary vertices adjacent to at least a visited vertex. ShortCut is proved to be (at most) 16-competitive on planar graphs. The case of exploration of cycles under the fixed graph scenario is analyzed by two works. The work of [88] shows an algorithm that builds a tour to explore the vertices of a graph and proves that its competitive ratio is $1 + \sqrt{3}$ on simple cycles. In addition, they show that for unweighted graphs a standard Depth-First Search is 2-competitive. Asahiro et al. [12] prove that the weighted Nearest Neighbor algorithm, which chooses the next vertex to visit according to a weighted distance cost, achieves a competitive ratio of $1.5$ on cycles. Moreover, authors show that no exploration strategy can have a competitive ratio less than $1.25$ on cycles. The work of Megow et al. [86] considers the fixed graph scenario and shows that a reformulation of the ShortCut algorithm, called Blocking algorithm, has a constant competitive ratio for a class of graphs wider than planar graphs. More precisely, Blocking has a competitive ratio of $16(1 + 2g)$ for graphs of genus at most $g$ (a graph has genus at most $g$ if it can be drawn without crossing itself on a sphere with $g$ handles; a planar graph has genus $0$). Moreover, authors show that Blocking does not have a constant competitive ratio for general graphs. All the above results consider exploration *tours* that require the searcher to return to the starting vertex. This constraint is not usually imposed in practical robotic exploration, where the goal is to find an exploration *path*. As recognized by Kalyanasundaram and Pruhs [68], the combinatorics of the two problems are significantly different. In this respect, the most relevant previous work is that of Koenig et al. [72]. Authors consider the problem of a robot that has to visit all the vertices of a graph. A lower bound $\Omega(\frac{\log|V|}{\log\log|V|}|V|)$ on the number of edge traversals is provided for a mapping method called Greedy Mapping, in which the robot is able to perceive only the current vertex and selects the closest unvisited vertex. Tovey and Koenig [127] find an upper bound for Greedy Mapping (called Closest Unvisited in that paper) and other three variants. Specifically, one of the variants (Closest Unscanned) is similar to Closest Unvisited: the robot can perceive the current vertex and may be able to scan an arbitrary number of vertices from a distance, without visiting them. The robot chooses the closest unscanned vertex. Closest Unscanned with Replanning is a variant of Closest Unscanned, in which the robot is allowed to replan while traveling to a selected closest unscanned vertex. The last variant, called Closest Informative, considers the closest scanned vertex which has some unscanned vertices as neighbors. Authors prove that, such mapping methods that consider the distance as main criterion, namely that select the closest unvisited, unscanned, or informative vertex, have an upper bound of $O(|V|\ln|V|)$ edge traversals.

Basically all the works presented in this section consider the fixed graph scenario, so we have that the timing of the perception is discrete (recall that perception refers to the integration of the data coming from sensors in the current map). Also, the sensor range can be considered $\epsilon$, where $\epsilon$ is a small value close to $0$, as just the incident edges of the current vertex can be perceived. Note that just the work of Tovey and Koenig [127] considers a variant in which a more powerful sensor is available. Nevertheless, the provided results are those found exploiting only the knowledge of the current vertex and ignoring other vertices possibly perceived.

Summarizing, we classify the works described above, according to the following dimensions and report them in Table 2.2:

- Goal: visit all edges (E) or all vertices (V).
- Solution: the solution can be a closed tour (T) or a path (P).
- Optimality criterion: the criterion that is optimized can be the num-

| Paper | Goal | Solution | Optimality criterion | Graph |
|---|---|---|---|---|
| [37] | E | P | # edge traversals | directed, strongly connected |
| [97] | E | P | # edge traversals | undirected, connected |
| [48] | E | T | memory size | anonymous, undirected |
| [111] | V | T | traveled distance | complete, weighted, undirected |
| [68] | V | T | traveled distance | weighted, planar |
| [88] | V | T | traveled distance | trees and cycles |
| [12] | V | T | traveled distance | weighted undirected graph |
| [86] | V | T | traveled distance | undirected, connected, weighted |
| [72] | V | P | # edge traversals | undirected, connected, unweighted |
| [127] | V | P | # edge traversals | undirected, connected, unweighted |

**Table 2.2:** *Classification of representative papers from theoretical computer science on exploration according to several dimensions.*

ber of the number of visited vertices (steps) and the number of edge traversals (distance, unweighted or weighted).

- Type of graph: for example, general graphs or planar graphs.

Some other variants of the graph exploration problem include the non-ability of the robot to uniquely recognize already visited nodes [108], the constrained exploration of a graph, where a robot is tethered or must return from time to time to a fixed point [1, 13, 39], the exploration of directed graphs [46], and the use of multiple explorers [35, 61].

## 2.2 Practical contributions

The mainstream approach to robotic exploration [137] used in practical settings identifies some candidate locations on the *frontiers* between known (already explored) and unknown portions of the environment, evaluates them, and assigns them to robots, iteratively. In addressing the works available in literature (typically from autonomous mobile robotics community), we tear apart the aspects of evaluating the candidate locations (exploration strategy) and of allocating robots to candidate locations (coordination method). In the following, we present a representative sample of the several exploration strategies and coordination methods presented in literature, showing also some of those using semantic information for speeding up autonomous exploration of initially unknown environments.

### 2.2.1 Exploration strategies

In the literature, exploration strategies usually use a utility function that combines different criteria, which characterize each candidate location, to assess the goodness of different candidate locations. Most of the criteria employed by exploration strategies use only metric information, namely information that can be derived from metric maps that robots build. For example, the robotic exploration system of Gonzáles-Baños and Latombe [56] combines, in an exponential function, the distance $d(p, r)$ between a robot $r$ and a candidate location $p$ and the expected amount of information $A(p)$ that $r$ can acquire at $p$ (measured as the maximum amount of unknown area visible from $p$):

$$A(p)\exp\left(-\lambda d(p, r)\right), \tag{2.1}$$

where $\lambda$ weights the new information obtainable from a position and the cost of traveling to reach the position. The authors show the experimental effectiveness in indoor environments.

A system using the same two criteria, but combining them in a linear function, is that of Burgard et al. [22]:

$$U_{t'} - \beta \cdot V_{t'}^{i'}, \tag{2.2}$$

where $U_{t'}$ and $V_{t'}^{i'}$ are similar to the expected information gain and traveled distance, respectively, used by the work presented above, and $\beta$ is the relative weight of benefit versus cost of candidate location $t'$ for robot $i'$. Experiments performed in unstructured, office, and corridor environments demonstrate the effectiveness of the proposed method.

The exploration strategy proposed by Visser and Slamet [131] used by a robot that should communicate with a fixed base station uses the following utility function:

$$\frac{A(p)P(p)}{d(p)} \tag{2.3}$$

that considers also the estimated probability $P(p)$ of a successful communication between the robot (once at $p$) and a fixed base station. The inclusion of such criterion is motivated by the fact that the authors observed that, in a number of experiments, a reliable communication between robots is fundamental in order to efficiently explore the environment.

Amigoni and Caglioti [3] present an information-based approach that has been derived using the concept of relative entropy. The evaluation function used to compare the candidate destination locations is:

$$\frac{1}{N+A}\sum_{i \in A \cup \mathcal{N}} \ln\frac{\sigma_{\text{unc,i}}}{\sigma} + N\ln\frac{\sigma}{P} + \sum_{i \in \mathcal{A}} \ln\frac{\sigma}{\sigma_{\text{p,i}}} + N\ln\frac{2\pi c}{\sigma}, \tag{2.4}$$

where $N = |\mathcal{N}|$ is the expected number of new points sensed from $p$, $A = |\mathcal{A}|$ is the expected number of already sensed points that are sensed again from $p$, $\sigma_{\text{unc,i}}$ is the standard deviation of the contribution to the measurement error due to the robot pose uncertainty, $\sigma$ is the standard deviation of the sensor accuracy, $P$ is the expected perimeter of the area to be mapped, $\sigma_{\text{p,i}}$ is the prior standard deviation of the already sensed point $i$, and $c$ is the distance between the current position of the robot and $p$. In Equation (2.4), the smallest values of function identify the best candidate location.

Tovar et al. [126] use a utility function that evaluates a sequence of $m$ candidate locations (poses) with the following multiplicative function:

$$\sum_{i=1}^{m}(\exp(lv_i - sv_i)\prod_{j=1}^{q_i}(\frac{\exp(-|\theta_j|)}{\sqrt{s_j}+1})\times(\frac{1}{n_i}\sum_{k=1}^{n_i}p_k + Ne_i)\mathit{fmin}_i(d_l)), \quad (2.5)$$

where $q_i$ is the total number of robot stops to reach location $i$, $lv_i$ is the length of the closest frontier edge at $i$, $s_j$ is the distance from the robot to the next possible location $j$, $sv_i$ is the distance from the next possible location $i$ to the closest frontier edge, $\theta_j$ is the orientation change to reach the next configuration $j$, $p_k$ identifies the probability of viewing landmark $k$ at $i$, $n_i$ and $Ne_i$ are the number of landmarks and of corners inside a visibility region at $i$, and $\mathit{fmin}_i$ is a function that penalizes $i$ when it is too close to an obstacle, according to distance $d_l$. The authors test the proposed approach in simulation and on real robots, although no quantitative comparison with other methods is provided.

Marjovi et al. [84] propose a method for multi-robot exploration and fire searching. The utility is computed according to the ratio between utility and cost, where

$$\text{utility} = \sum_{i=1|i\neq R}^{m} \text{dist}[(X_{f_k}, Y_{f_k}), (X_{r_i}, Y_{r_i})], \quad (2.6)$$

which computes utility according to the distance dist between the frontier $f_k = (X_{f_k}, Y_{f_k})$, currently evaluated by robot $R$, and the position of other robots $r_i = (X_{r_i}, Y_{r_i})$, and

$$\text{cost} = \text{dist}(A^*_{k=0,n}[(X_R, Y_R), (X_{f_k}, Y_{f_k})]), \quad (2.7)$$

which calculates the distance using $A^*$ method between the frontier $f_k$ and the robot $R$. The idea of such exploration strategy is to let the robots disperse and explore the environment in a more efficient way. The experimental results with real robots in some environments show the efficiency and the reliability of the proposed method, which is also compared with an optimal method.

Rekleitis [106] proposes an exploration strategy whose objective is to minimize the uncertainty of the map. Experiments performed in random graphs show the viability of the proposed method. The same authors, in [139], builds on the latter to study the use of a topological map privileging the accuracy on the produced map.

A more general and flexible method based on Multi-Criteria Decision Making (MCDM) that provides a solid approach, grounded in decision-theoretical field, for selecting the next candidate location in exploration is proposed by Basilico and Amigoni [16]. Consider a set of candidate locations $\mathcal{P}$, a set of robots $\mathcal{R}$, and a set of criteria $N$. Call $u_j(p, r)$ the utility value for candidate location $p \in \mathcal{P}$ and robot $r \in \mathcal{R}$ according to criterion $j \in N$. The larger $u_j(p, r)$, the better the pair $p$ and $r$ according to $j$. To apply MCDM, utilities need to be normalized to a common scale $I = [0, 1]$. The authors use a linear relative normalization for each $u_j$. With a slight abuse of notation, $u_{(j)}$, with $(j) \in N$, is the $j$-th criterion according to an increasing ordering with respect to utilities: for candidate location $p$ and robot $r$, $u_{(1)}(p, r) \leq \ldots \leq u_{(n)}(p, r) \leq 1$, where $n = |N|$ (it is assumed that $u_{(0)}(p, r) = 0$). The MCDM strategy integrates the criteria in $N$ with the following function:

$$u(p, r) = \sum_{j=1}^{n} (u_{(j)}(p, r) - u_{(j-1)}(p, r))\mu(\mathscr{A}_{(j)}), \tag{2.8}$$

where $\mu : 2^N \rightarrow [0, 1]$ ($2^N$ is the power set of set $N$) are weights, and the set $\mathscr{A}_{(j)}$ is defined as $\mathscr{A}_{(j)} = \{i \in N | u_{(j)}(p, r) \leq u_i(p, r) \leq u_{(n)}(p, r)\}$. Specifically, $\mu(\{\emptyset\}) = 0$, $\mu(N) = 1$, and, if $N' \subset N'' \subset N$, then $\mu(N') \leq \mu(N'')$. That is, $\mu$ is a normalized *fuzzy measure* on the set of criteria $N$ that will be used to associate a weight to each group of criteria. The weights specified by the definition of $\mu$ describe the relationships between criteria. Criteria belonging to a group $G \subseteq N$ are said to be redundant if $\mu(G) < \sum_{i \in G} \mu(i)$, synergic if $\mu(G) > \sum_{i \in G} \mu(i)$, and independent otherwise. Clearly, weight $\mu$ for each subset of criteria should be defined reasonably. Principled methods for selecting weights are discussed by Basilico and Amigoni [16]. Intuitively, Equation (2.8) provides a sort of "distorted" weighted average that accounts for synergies and redundancies between criteria. Using Equation (2.8) is a more principled way than other utility functions (e.g., [131]) to compute utilities, because it allows to consider criteria's importance and their mutual dependency relations. This approach is shown to experimentally outperform other exploration strategies like that of Visser and Slamet [131].

Only few exploration strategies use information coming from semantic maps of environments (built, for example, by semantic mapping systems of Wolf and Sukhatme [133] and Mozos et al. [89]). An early attempt in this direction is that of Kuipers and Byun [74], in which candidate locations with a large distinctiveness (e.g., located at the intersections of corridors) are privileged. Specifically, the authors, instead of explicitly considering semantic information, use a geometric measure, which derives from finding points that are equally-distant to close obstacles, and apply a hill-climbing control strategy to find the robot's exploration path, so that distinctiveness is maximized. They show the goodness of the exploration path found through a qualitative analysis of the solution obtained during simulations.

The work by Stachniss et al. [121] exploits the knowledge on the structure of an indoor environment (represented as a hidden Markov model) to drive robots to select, first, candidate locations that are in corridors. For each robot $r$ and each candidate location $p$, the difference between the initial utility of $p$ (which is equal for all frontiers and is initialized at $1$, not considering any features of $p$) and the distance between the current position of $r$ and $p$ is calculated. The initial utility of candidate locations that are in corridors is multiplied by $\gamma$ (set to $5$ in the experimental activity). Then the method greedily allocates the candidate locations to the robots by selecting the pair $r$ and $p$ that maximizes the above difference. Experimental results (performed in simulation) show that the approach is effective in decreasing the time required to explore some environments with respect to an approach that does not update the initial utility of corridors.

Another work that uses semantic information to improve exploration is presented by Calisi et al. [24]. In this case, contextual information related to the mission (e.g., the relative importance of a goal with respect to another goal), to the environment (e.g., the presence of rooms and corridors and the difficulty for traversing a given area and for detecting victims in that area), and to the agents (e.g., the presence of loop closures for improving localization of robots) is represented by a PROLOG rule-based system and exploited to enhance the performance of a robotic system operating in a search and rescue scenario. The experiments (performed in simulation) use a single robot and show that the proposed approach can significantly increase the area mapped by the robot within 15 minutes.

Another system that exploits the structure of the environment for determining the best candidate locations and assigning them to the robots is presented by Wurm et al. [134]. The known portion of the map of the envi-

| Paper | Evaluation criteria | Metric |
|---|---|---|
| [56] | M | # motions and perceptions |
| [22] | M | exploration time |
| [131] | M | total explored area |
| [3] | M | traveled distance, # steps |
| [126] | M | traveled distance, # perceptions |
| [84] | M | # repeated nodes, exploration time |
| [106] | M | traveled distance, map uncertainty |
| [16] | M | total explored area |
| [74] | S | map correctness |
| [121] | S | exploration time |
| [24] | S | total explored area |
| [134] | S | exploration time |

**Table 2.3:** *Classification of representative papers about exploration strategies according to some dimensions.*

ronment is segmented and a single robot is assigned to (one of the frontiers of) each segment. The utility function used to assign a robot $r$ to a frontier $p$ belonging to a portion of the map considers the distance from the current position of $r$ to $p$. Experimental results (both in simulation and with real robots) show that the approach can significantly reduce the overall exploration time for realistic environments with respect to a closest-frontier approach that assigns to each robot the closest candidate location.

Note that all these approaches that consider also semantic information use a utility function that considers basically just the cost to reach a candidate location. The experimental comparison performed by Amigoni [2] showed that, in some common settings, exploration strategies that balance utility and cost tend to have better performance than those that use only cost. Nevertheless, all these works that embed semantic information in the exploration strategy show that the *total* area explored in a given time interval can be improved by using semantic information.

Note that all the works presented in this section assume that the solution is a path.

Summarizing, we classify the works described above, according to the following dimensions and report them in Table 2.3:

- Evaluation criteria: the criteria used in the exploration strategies can be metric-based (M) or also semantic-based (S).
- Metric: the metric considered to assess the goodness of the proposed exploration strategy, for example, traveled distance to explore the whole area of the environment or explored area in a given time interval.

### 2.2.2 Coordination methods

Coordination between multiple exploring robots, namely assigning robots to candidate locations, is achieved in different ways in the literature.

A series of works (e.g., [21, 22, 121] and, partially, [47]) propose an interesting approach in which the coordination method is embedded within the exploration strategy. In particular, the utility value of a candidate location is reduced according to the number of robots that can view it, in order to discourage the assignment of more robots to the same candidate location. Experimental results show that this coordinated behavior has better performance than uncoordinated behavior (in which different robots can select the same location to reach) and slightly worse performance than a method that finds the optimal allocation over all possible permutations of candidate locations to robots, where the optimality criterion depends on the difference between utility and cost of visiting the candidate locations.

Several other works (e.g., [118, 140]) are based on market mechanisms. Specifically, coordination of mobile robots is performed by a central executive that, beyond collecting local maps and combining them into a single global map, manages an auction-like mechanism by asking bids to the robots and assigning tasks (i.e., locations to reach) according to the received bids. Bids contain information about expected utility for pairs robot-location; utilities are calculated by the exploration strategy adopted. Experimental results show that auction-based coordination methods (as expected) outperform the uncoordinated methods.

All of the presented approaches for coordination attempt to spread the robots around the environment. The (often) implicit assumption is that the exploration problem is considered to involve, according to the classification of Gerkey and Mataric [52], single-task robots (ST) and single-robot tasks (SR), where the task is to reach a candidate location. ST means that each robot executes one task at a time and SR means that each task requires one robot. Thus, all the above works act basically as ST-SR.

An extension to the ST paradigm usually considered is provided by Faigl et al. [41], who propose a method for allocating not only one candidate destination location, but all the current frontiers to the robots (MT, multi-task robots), leading to the multiple traveling salesman problem formulation.

An extension to the SR paradigm is given by the the approach of Hawley and Butler [60], which proposes an auction-based coordination method not only for task assignment, but also for coalition formation, leading partially to a multi-robot task (MR), when there are more robots than candidate locations.

Table 2.4 classifies some of the most significant papers on coordinated robotic exploration according to the classification of Gerkey and Mataric [52].

| | | **Robots** | |
|---|---|---|---|
| | | SR | MR |
| **Tasks** | ST | [21, 22, 60, 118, 121, 140] | [60] |
| | MT | [41] | |

**Table 2.4:** *Classification of representative papers about coordination methods according to the classification of Gerkey and Mataric [52].*

Recently, embedding the constraint on the limited communication range between robots started to receive more attention (e.g., [36]). Also, the study of the heterogeneity of the robots in terms of capabilities has received more attention (e.g., [107]).

## 2.3 Evaluation

A lively debate on good experimental methodologies is currently ongoing in the autonomous robotics community, which has recognized that the lack of sound procedures, globally accepted benchmarks, and well-established metrics for autonomous robotic systems slows down research and make the transfer of existing research results to market products rather difficult [5, 6].

One of the problems lies on answering (at least partially) to the question 'which approach performs better in a given setting?'. Coming down to exploration, as shown in Section 2.1, on the one hand, exploration strategies are defined in theoretical settings (e.g., exploration of graphs) and are assessed using theoretical tools like worst-case bounds and competitive ratio, which is the ratio between the cost of the solution found by an online algorithm and that of the optimal solution found by an offline algorithm. This allows to have a comparison of the proposed methods by comparing either their competitive ratio or their upper and lower bounds that are usually provided in the worst case. However, these bounds (and as we will show in Chapter 7), could be very loose in some settings and not so informative about whether a method is better than another. On the other hand, the exploration strategies are defined and assessed in practical settings (see Section 2.2). The experimental evaluation and comparison is mainly performed by testing some state-of-the-art methods and the proposed one in some settings. As there is no reference performance, results are compared in a relative way, against to each other, and not in an absolute way (e.g., [2, 67]). See for example Figure 2.1. Thus, an important issue, which has

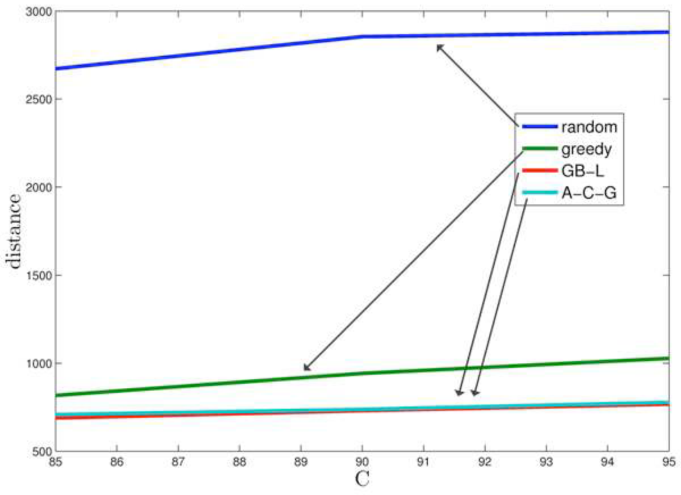


**Figure 2.1:** *An example of relative comparison among exploration strategies [2].*

been largely overlooked so far, is the availability of a reference (optimal) performance for conducting the experimental evaluation.

Another question is about which metric should be used for evaluation. This is related to the difficulty of establishing a single optimality criterion that we evidenced when discussing the corresponding dimension of our framework. The number of sensing actions performed by the robots and the total amount of distance traveled to completely explore an environment are two common metrics, but there currently is no agreement on this [15].

Another relevant question is about which data should be used for evaluation. In real environments, evaluation can be clearly performed only *ex post*, once the environment has been completely mapped. Data sets acquired by real robots (like those on Rawseeds [28] and Radish [63]) are usually not so dense to cover every perception in every pose of the environment and cannot be used to evaluate exploration strategies and, more generally, systems that decide the next actions online. Simulation is sometimes the only viable solution [4].

Furthermore, the details reported about the experimental settings presented in works available in literature are sometimes not sufficient to replicate experiments and thus it is not clear what parameters could affect the exploration performance [6]. In the following, we show two aspects that have not been considered and investigated so far in the experimental evaluation of the exploring systems but that could have an impact on their performance, namely the impact of perception and decision timing, and the relative impact of exploration strategies vs. coordination methods on the exploration performance.

Explicit information about when perceptions (Step (b) presented in Section 1.1) and decisions (Step (c) presented in Section 1.1) are performed are rarely reported in papers. According to the available data, we can broadly classify the published works according to the following classes.

**Event-based vs. frequency-based perception**. In case of *event-based perceptions*, only the data acquired at the destination location is integrated into the current map of the environment. This means that a robot “blindly” navigates from its initial location to the destination location moving along a path inside the known free space, without updating the map. In *frequency-based perception*, data is continuously acquired and integrated into the map, at some fixed frequency, as the robot navigates toward the destination location. This frequency can depend either on the sensor acquisition frequency or, more often, on the update frequency of the map building method employed (e.g., the map is updated after a certain distance, after a certain rotation angle, or after a certain amount of time). Note that these timings are related only to map updates for exploration purposes (Step (b) presented in Section 1.1). Indeed, depending on the methods used for map building and localization, the data acquired during navigation can be used for localization and path following, but we do not consider the timing of these acquisitions if they do not enrich the map built by exploration.

**Event-based vs. frequency-based decision**. In the first case, a decision about the next destination location is made only when the previous destination location has been reached (or after that location has been declared unreachable, for example after a timeout expired). In the second case, decisions about the next destination location are continuously made, at some fixed frequency (for example, in [92] a new decision is basically made every 4 s), while the robot is moving toward the current destination location (implementing a sort of opportunistic behavior).

Table 2.5 classifies some of the most significant papers on robotic exploration according to the above dimensions. The aim of the table is not to exhaustively classify all works in the area, but only to provide an overview based on a significant sample of published papers.

Most works consider event-based perception and decision. This is not surprising because of ease of implementation of such configuration. Although they use different exploration strategies and different map representations, these works basically consider reaching the current destination

| | | **perception** | |
|---|---|---|---|
| | | event-based | frequency-based |
| **decision** | event-based | [3, 23, 49, 56, 75, 81, 117, 126, 136] | [22, 62, 131] |
| | frequency-based | - | [40, 92, 118] |

**Table 2.5:** *Perception and decision modalities employed by some papers.*

location as the event that triggers the acquisition of new data from sensors and their incremental integration into the current map as well as the decision making about the next destination location. Relatively less works use frequency-based perception and decision. Unsurprisingly, no work considers event-based perception and frequency-based decision. This is expected since it makes little sense to decide repeatedly about the next destination location on the basis of the same information about the environment.

Some recent works have partially addressed the study of the effects of perception and decision timing on exploration performance. For instance, Holz et al. [62] explicitly recognize one of the main problems of event-based decision under frequency-based perception: since the map is continuously updated according to data coming from long range sensors, the robot might have fully explored a region (e.g., a dead-end corridor) before actually reaching the selected destination location. In the work of Holz et al. [62], a heuristic is proposed to avoid to reach such destination locations if there is nothing the robot could discover there. This amounts to discard the old decision about the destination location. A similar problem is recognized by Keidar and Kaminka [69] and some (fast) frontier detection algorithms are proposed as a support for making decisions about destination locations at a high frequency, up to 10 Hz (as it can be deduced from experiments).

Nevertheless, beyond these partial attempts, and to the best of our knowledge, a systematic analysis of the impact of perception and decision timing on the performance of exploration is still missing.

Another aspect that could influence the exploration process but not fully investigated so far include the relative impact of exploration strategies and coordination methods on the exploration process. Indeed, as shown in Section 2.2, coordinated multirobot exploration has been mainly studied for map building [71, 78] and for search and rescue [84] focusing in a rather separated way on evaluation of either coordination methods or exploration strategies.

# Part I
# Bridge the gap between theory and practice

The first part of this dissertation presents the contributions towards the goal of bridging the gap between theory and practice.

On the one hand, as discussed in Section 2.1, most of the works in the computational geometry area consider assumptions that do not apply directly to real scenarios for the problem of exploration. Against this background, in Chapter 3, we tackle the problem of computing the optimal exploration path for a robot with time-discrete and limited visibility. The optimality criteria considered are those typically used for evaluating the performance of an exploration system and are the number of steps and the traveled distance. By considering some more realistic robotic assumptions, we make a step toward filling the gap between theoretical results and practical applications of robot exploration.

On the other hand, as discussed in Section 2.2, most of the proposed techniques are experimentally tested in some specific scenarios. In order to better understand the performance of exploration strategies and to more solidly explain some experimental findings reported in the literature, in Chapter 4, we theoretically analyze some exploration strategies that have been used in literature. Specifically, we analyze those that evaluate candidate locations with distance and/or information gain. Also, we consider some realistic assumptions over the robot's capability, by assuming that a robot has a sensor with a finite range $r$, and over the termination criterion, that prescribes the perception of a fraction $\mathfrak{g}$ of the environment. The analysis is performed using a graph-based representation of the environment.

# 3 Offline exploration problem

In this chapter, we consider a single robot with limited-visibility range scanner sensors operating in a two-dimensional world. For this robot, exploring an environment means to follow an *exploration path* and operate a sequence of perceptions at a finite number of distinct locations along this path (time-discrete perception) such that all the free area of the environment is perceived. Given this context, we address the *optimal exploration* problem, whose solution is an exploration path that minimizes some objective functions (or optimality criteria). In this chapter, optimality will be associated to two common objective functions: the number of perceptions to be performed along the exploration path and the total traveled distance, namely the length of the exploration path.

We deal with the exploration problem from a particular offline perspective. More precisely, starting from the assumption that a given environment is initially known, we try to answer the question of what is the optimal exploration path that *any* online exploration strategy could achieve in that specific environment. In other words, we search for an optimal solution which could be compared with those found by online exploration strategies working in initially unknown environments. The problem we solve can be also formulated as follows. Consider an initially unknown environment, run any online exploration strategy that discovers the environment along some exploration path. Then, *a posteriori*, we answer the question: what would have been the optimal exploration path for discovering the environment?

Our problem differs from closely related problems, like art gallery [129] and watchman route [30] formulations (see Section 2.1.1). In the case of art gallery, proposed methods find the minimum set of guards to completely cover a given environment. In works related to watchman tour, instead, the similar objective of finding a shortest continuous exploration tour (a closed path starting and ending at the same point) for arbitrary two-dimensional polygonal environments is addressed. Using art gallery terminology, our problem formulation constrains each guard (a location on the path from where a perception of the environment is made) to be visible from at least another guard that has been already placed. In principle, this constraint, which we will refer to as *reachability constraint*, makes the optimal exploration path (considering the two optimality criteria mentioned above) incrementally "travelable" by an exploring robot with no *a priori* knowledge about the environment. For such a robot, indeed, every new perception should be performed in the space within which it can move safely, i.e., in the space the robot has already perceived as free up to that time.

More specifically, the reachability constraint we consider imposes that a *scan point* $q$ (i.e., a location where robot perceives the environment) should be visible from at least another previously placed scan point $q'$, namely $q$ should be in at least a disk, whose center is $q'$ and radius (sensor range) $r$. Notice that, while this constraint is implicit in online exploration, it is, in general, not enforced when the environment is already known in advance (in problems like the art gallery, it would imply an unnecessary over-constraining of the problem, where not even a path is considered). Figure 3.1 shows an example in which, assuming no arbitrary initial robot starting point, the art gallery problem would require 2 scan points (in blue), whereas the optimal exploration problem we consider in this chapter would require 3 scan points (blue and red points). Note that similar considerations hold also for variants of the watchman tour that consider also the number of scans or turns in the objective function. In particular, the solution for such variants requires the robot to stop and perceive the surroundings at the scan points in blue, while the optimal exploration path is required to stop also at the red-colored scan point. Also, typically the watchman tour prescribes that, once the whole area is covered, the robot should go back to the starting point. Instead, in the exploration problem, as said, usually a path (not necessarily closed) is considered as solution. The reachability constraint is motivated by the fact that we are looking for an optimal solution of the exploration problem that is online feasible. Note that the optimal exploration problem corresponds to the coverage problem with the reachability constraint.

In the following, we characterize the optimal exploration problem for a single robot, by formulating it with reachability constraint in the continuous space and in the discrete space, where it can be more conveniently tackled for resolution. We analyze the theoretical implications of operating

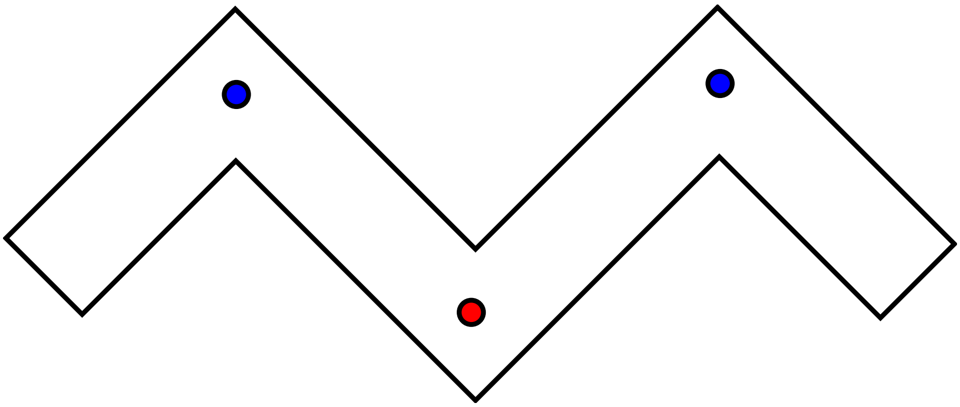


**Figure 3.1:** *Solution to the* art gallery *problem (blue points) vs. solution to the* optimal exploration *problem (blue and red points). Note that solution to the watchman problem does not require the robot to stop at the red point, while the optimal exploration problem does.*

in the discrete space with respect to two aspects. First, we study the relationship between the continuous space formulation and the discrete space formulation in which the environment is represented with a regular grid of given resolution. Second, we focus on the inherent sub-optimality that solving a discretized version of the problem would introduce. We consider number of perceptions and traveled distance as optimality criteria and we provide bounds on the discrete optimal solution's quality with respect to the continuous one. Then, we define the discrete space problem as a classical AI search problem, which is solved with A$^*$. A set of parameters and speedup techniques are proposed in order to balance the algorithm's expected computation time and the solution's quality. Experimental activities we performed in simulation show that our algorithm behaves satisfactorily well for realistic environments. We compare our results with those of a state-of-the-art coverage method [42] that can be readily adapted to solve our optimal exploration problem.

## 3.1 Optimal exploration problem

We assume to have a single holonomic autonomous mobile robot moving in an arbitrary continuous two-dimensional bounded environment $\mathscr{P} \subset \mathbb{R}^2$. The interior points of $\mathscr{P}$ can belong to obstacles of arbitrary shape, whose set is denoted by $\mathscr{P}_o$, or can belong to the free space where the robot can move, denoted by $\mathscr{P}_f = \mathscr{P} \setminus \mathscr{P}_o$. The robot is considered as a point (this assumption is without loss of generality if obstacles are "grown" to account for the real size of the robot, as usual in path planning [76]). Going over the dimensions presented in Section 1.3, the robot is equipped with a $360°$ range sensor with a finite range $r$ able to perceive free space and outer boundary of obstacles (e.g., two laser range scanners with $180°$ FOV each mounted at the same height and back-to-back). Specifically, we assume that the sensor model prescribes that a point $q$ within $r$ from the robot and such that the inner of the line segment between $q$ and the robot does not contain any point of $\mathscr{P}_o$ is said to be *visible* from the current robot position. With such a sensor, we can ignore the orientation of the robot and consider only its position in the plane. From any position, the robot can perceive the surrounding environment and update a map that keeps track of the portion of $\mathscr{P}$ discovered so far. We assume that perception model is time-discrete, as in several works in literature (see Section 2.3). That is, given a path covered by the robot, perceptions are performed only at a finite subset of scan points on that path. Given two subsequent scan points, no additional information about the environment is integrated into the map when moving from one to the next (although information coming from sensors could be used for collision avoidance). This implies that also the decision model is time-discrete. The robot operates according to the behavior model defined in Section 1.1. Since we are interested in studying optimal exploration, we assume that the movements (Step (d)) and the perceptions (Step (a)) of the robot are error-free. As a consequence, the robot perfectly knows its position. We also assume that the environment is static and known in advance, so we can formulate an offline problem.

We now provide a general formulation of the optimal offline exploration problem we tackle in this chapter.

**Problem C** (Optimal offline exploration in the continuous space)**.** *Given an environment $\mathscr{P}$, a robot with sensor range $r$, and an initial robot position $q_0 \in \mathscr{P}_f$, find a sequence of positions (scan points in $\mathscr{P}_f$) $Q = \langle q_0, q_1, \ldots, q_n \rangle$ such that:*

- *every free point $q \in \mathscr{P}_f$ is perceived by the robot from at least a position $q_i \in Q$;*
- *the reachability constraint is met: position $q_{i+1}$ should lie in the free space discovered with the perceptions performed at positions $\langle q_0, \ldots, q_i \rangle$;*
- *$Q$ is optimal, i.e., it minimizes a given cost function.*

We denote as $M_i^{\mathscr{P}} = M_i^{\mathscr{P}_o} \cup M_i^{\mathscr{P}_f}$ the partial map of $\mathscr{P}$ built by the robot from positions $\langle q_0, \ldots, q_i \rangle$. The problem is specified with two cost functions (describing two different optimality criteria). More precisely, we denote as Problem $C$-$\#$ the above problem when the minimization of the *number of steps* (number of scan points where the perception actions of Step (a) above are performed) is adopted, that is

$$\min n. \tag{3.1}$$

Similarly, we will refer to Problem $C$-$\mathscr{D}$ when the minimization of the *traveled distance* is considered, namely

$$\min \sum_{i=0,1,\dots,n-1} \mathscr{D}(q_i, q_{i+1}), \tag{3.2}$$

where $\mathscr{D}(q_i, q_{i+1})$ is the length of the shortest path lying in $M_i^{\mathscr{P}_f}$ connecting $q_i$ to $q_{i+1}$. In general, the optimal solution $Q^*$ is different if we use the number of steps or the traveled distance. We call $n_C^*$ and $\mathscr{D}_C^*$ the costs of the optimal solutions, obtained with number of steps and distance, respectively.

The Problem $C$-$\#$ for arbitrary two-dimensional polygonal environments is NP-hard, as it can be easily traced back to the art gallery problem [96, 114]. Also the Problem $C$-$\mathscr{D}$ is NP-hard as finding a coverage tour (a closed path starting and ending at the same point, namely $q_o = q_n$) for arbitrary two-dimensional polygonal environments has been shown to be NP-hard [10]. Therefore, finding an exploration path with the further constraint on reachability is also NP-hard, being a path more general than a tour.

In order to look for an approximation of the optimal solution for both problems, we translate Problem $C$ from the continuous to the discrete space. The continuous environment $\mathscr{P}$ is tiled by a finite regular grid of cells $G^e$ such that each point $q \in \mathscr{P}$ belongs to a cell of the grid. Cells are identical squares with edge of length $e$ and can be either free or occupied by obstacles. If there exists a point $q \in \mathscr{P}_o$ that belongs to a cell $c \in G^e$, then $c \in G_o^e$; if all points belonging to a cell $c$ are $q \in \mathscr{P}_f$, then $c \in G_f^e$ (see Figure 3.2). Each cell partially lying over the environment external boundary is considered an obstacle cell. Hence, the grid $G^e$ representing the environment is partitioned in sets of cells $G_f^e$ and $G_o^e$ containing the free and the occupied cells, respectively. The free space $G_f^e$ and the obstacles $G_o^e$ can have any form. In the following, with a slight notation overload, we use the same symbol $c$ to indicate both the cell $c \in G^e$ and the coordinates of the position of its center $(c_x, c_y)$ in a global coordinate system.

The robot (again, considered as a point) starts from the center of a cell $c_0 \in G_f^e$ and its basic movements are from the center of its current cell

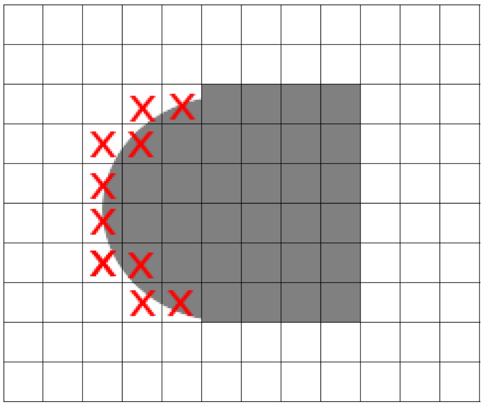


**Figure 3.2:** *Cells indicated with red cross are considered obstacles, even if only partially containing an obstacle (in gray).*

to the center of another adjacent free cell. We assume that the grid is $8$-connected with an additional constraint preventing the robot from moving between obstacle cells that share a vertex. The range sensor (again, limited to $r$) perceives the state (free or occupied) of any cell $c_i$ whose center can be connected to the position of the robot with a straight line segment whose length is strictly less than $r$ and crossing only free cells (and without passing between obstacle cells that share a vertex). We can thus formulate optimal offline exploration in the discrete space.

**Problem D** (Optimal offline exploration in the discrete space)**.** *Given a grid $G^e$ representing an environment $\mathscr{P}$, a robot with sensor range $r$, and an initial robot position $c_0 \in G^e_f$, find a sequence of positions $\mathcal{C} = \langle c_0, c_1, \ldots, c_n \rangle$ such that:*

- *every free cell $c \in G^e_f$ is perceived by the robot from at least a position $c_i \in \mathcal{C}$;*
- *the reachability constraint is met: position $c_{i+1}$ should lie in the free space discovered with the perceptions performed at positions $\langle c_0, \ldots, c_i \rangle$;*
- *$\mathcal{C}$ is optimal, i.e., it minimizes a given cost function.*

We denote $M_i^{G^e} = M_i^{G^e_o} \cup M_i^{G^e_f}$ the partial map of $G^e$ built by the robot from cells $\langle c_0, \ldots, c_i \rangle$. Similarly to the continuous case, we call Problem $D$-$\#$ and Problem $D$-$\mathscr{D}$ the two versions of the problem using the number of steps and the traveled distance, respectively. Also, we denote with $n^*_D$ and $\mathscr{D}^*_D$ the costs of the optimal solutions, namely the minimum number of steps and the shortest traveled distance, respectively. In this case, the quality of the solution clearly depends on the cell size $e$.

Note that the knowledge representations in Problems $C$ and $D$ are continuous and grid-based, respectively. Also, the termination criterion is to map the whole free space. Given the *a priori* knowledge of the environment, we can apply an offline algorithm to compute the solution. The solution then is a path, namely the robot does not have to return back to the starting point, differently from most of the works presented in Section 2.1.1.

We now show that Problems $D$-$\#$ and $D$-$\mathscr{D}$ are NP-complete for grid graphs (another way to represent our grid environments). In a grid graph each node corresponds to a point of the plane with integer coordinates and edges connect nodes at unitary (Euclidean) distance [66].

**Proposition 3.1.** *The Problem $D$-$\#$ is NP-complete.*

*Proof.* We consider the decision version of Problem $D$-$\#$, namely given a positive integer $k$ decide if there exists an exploration path that perceives all the free cells with no more than $k$ perceptions. The problem is in NP since we can always verify in linear time the validity of a given solution. We can show that the problem is NP-hard by reducing from the decision version of the Minimum Connected Dominating Set problem (MCDS-$k$). MCDS can be described as follows [58]. Given a graph $G = (V, E)$ (where $V$ are the vertices where robots can stay and $E$ are edges that connect the vertices) find the subgraph $G' = (V', E') \subseteq G$ such that:

- $G'$ is connected;
- for every $v \in V$ either $v \in V'$ or there exists an edge $e = (v, q) \in E$ such that $q \in V'$;
- $|V'|$ is minimum.

The decision version of this problem MCDS-$k$ is, given an integer $k$, to decide whether there exists a connected dominating set with $|V'| \leq k$ or not. Such problem is NP-complete for general grid graphs as shown in [34].

A simple reduction can be obtained as follows. Given any grid graph $G$ we efficiently build an instance of Problem $D$-$\#$ according to the following steps:

1. consider a robot with range $r$;
2. consider $e = r$ and derive the grid corresponding to $G$, that is place a free cell at the coordinates of each vertex (properly scaled by a factor $e$);

3. add obstacle cells to every empty spot of the grid (we obtain a grid where no diagonal movements are possible).

Given such a construction, it follows immediately that MCDS-$k$ admits a solution if and only if Problem $D$-$\#$ admits an exploration path with $k$ steps. Given such exploration path, indeed, we can obtain in linear time a dominating set for $G$ by selecting the nodes of $G$ that correspond to perception-cells, i.e., those cells where a perception is prescribed in the exploration path. Our reachability constraint guarantees that the set is connected. Since every cell of the grid is visible from at least a perception cell of the exploration path, and since $e = r$, every node of $G$ is either in the dominated set or adjacent to a node in the dominated set. (Notice that in our constructed instances no diagonal perceptions are possible.) □

**Proposition 3.2.** *The Problem $D$-$\mathscr{D}$ is NP-complete.*

*Proof.* Let us consider the decision version of Problem $D$-$\mathscr{D}$, namely given a distance $d$ decide if there exists an exploration path which perceives the whole environment traveling no more than $d$ units. This problem can be easily shown to be in NP since we can always verify in linear time that a given exploration path travels less than or exactly $d$ units and senses all the free cells of the environment. To show its NP-hardness, we consider the Hamilton Circuit problem for grid graphs (HC-G) that is shown to be NP-complete in [66].

The idea behind the reduction (presented below) is to constructing an instance of Problem $D$-$\mathscr{D}$ in such a way that each cell $c_i$ that corresponds to a node $v_i \in V$ has to be visited. This is obtained by adding some dummy cells that could be perceived only from that cell $c_i$.

Given any grid graph $G = (V, E)$, that is any instance of HC-G, we can define a reduction to $D$-$\mathscr{D}$ as follows:

1. consider a robot with range $r \in \mathbb{Z}_{>0}$;

2. for each $v_i \in V$, consider a cross-shaped grid $g_i$ where $e = r$; $g_i$'s center cell is indicated as $c_i$ and since it is logically and univocally associated to node $v_i$ we call it *node cell*; the center cell is surrounded by 4-adjacent linear segments of $r$ free cells that we call *dummy cells*; every other cell to form a square of edge $2r + 1$ is an obstacle cell; (Fig. 3.3 reports the cross-shaped grid $g_i$ for node $v_i \in V$ where $e = r = 1$.)

3. compose a grid with the cross-shaped grids such that $(v_1, v_2) \in E$ if and only if $c_1$ and $c_2$ are at the shortest possible distance of $2r + 1$

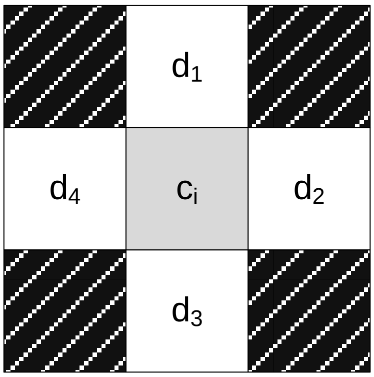


**Figure 3.3:** *Cross-shaped grid element for a node $v_i \in V$; $c_i$ is the node cell associated to $v_i$, while $d_1, \ldots, d_4$ are dummy cells.*

in the grid (recall that the distance between two cells is the Euclidean distance between their centers); in other words, the obtained grid is isomorphic to the graph (notice that, since the grid graph is planar, such construction can be done efficiently by considering the planar coordinates of each node in the graph);

4. if there exists $v_x \in V$ such that $degree(v_x) = 1$, then add a linear segment of $3$ dummy cells adjacent to a dummy cell of $g_x$ that is not adjacent to any other $g_i$ (do this for just one $v_x$) (Fig. 3.4 reports the resulting grid applying step 4 to a graph $G$);

5. if there not exists in $G$ a node $v_x$ with degree $1$, since $G$ is finite, then $v_x$ should have two neighboring nodes with degree less or equal to $3$; choose any of such neighbors $v_1$ and $v_2$ and add a dummy cell adjacent to a dummy cell of $g_1$ that is not adjacent to any other $g_i$, do the same for $g_2$ (Fig. 3.5 reports the resulting grid applying step 5 to a graph $G$);

6. fill every empty spot of the grid with an obstacle cell, call $O_G$ the obtained grid.

Now we can show that $G$ admits an Hamilton Circuit if and only if $O_G$ admits an exploration path that perceives the whole environment covering no more that $(2r + 1)(n - 1) + 2$ travel units.

Let us consider the above construction and temporarily ignore steps 4 and 5. In the $O_G$ built in this way, an exploration path of length $(2r + 1)(n - 1)$ visits each node cell exactly once. This follows from these two considerations: (a) *the minimum distance between two node cells is* $2r + 1$

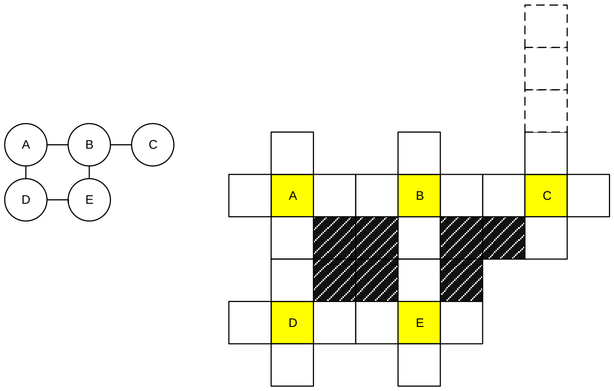


**Figure 3.4:** *Example of application of step 4 to a grid graph containing a node* $v_x \in V$ *such that* $degree(v_x) = 1$.

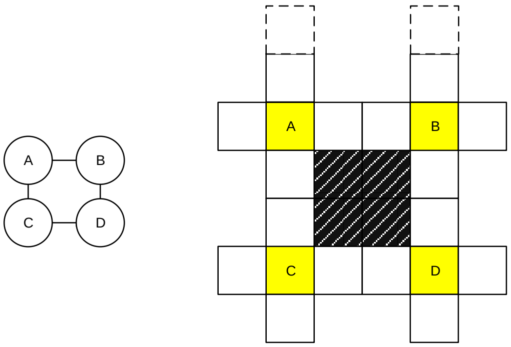


**Figure 3.5:** *Example of application of step 5 to a grid graph that does not contain a node* $v_x \in V$ *such that* $degree(v_x) = 1$.

(by construction) (b) *any exploration path that senses all the free cells with maximum traveled distance of* $(2r+1)(n-1)$ *has to visit all the node cells.* To prove (b) let us first consider nodes cells $c_i$ such that $degree(v_i) \leq 3$. In the corresponding $g_i$ of any such $c_i$ there is, by construction, at least one dummy cell which can be perceived only when the robot occupies $c_i$, therefore, to perceive all the free cells of the environment, each node cell with maximum degree 3 has to be visited. The same holds for cells with degree 4 despite showing it is more tricky. If $degree(v_i) = 4$ then, in principle, the robot can reach any dummy cell $d$ of $g_i$ which is at distance $r$ from $c_i$ and, consequently, perceive $c_i$ without visiting it. Differently from the previous case, doing so would not preclude the perception of any free cell in the environment. We provide the intuition of why this is not possible if, at the same time, the robot has to perceive the whole environment within a traveled distance of $(r+2)(n-1)$. When stopping at dummy cell $d$ the robot has traveled a distance of $r+1$ from the last occupied node cell $c_j$. This means that it gets a perception of $c_i$ by "saving" $r$ units of distance. When back at $c_j$, the total traveled distance in this portion of the exploration path would be $2(r+1)$. So, to "handle" cell $c_i$ the robot would travel more distance than the $2r+1$ units that a total budget of $(2r+1)(n-1)$ would impose. (Notice that, if between $c_j$ and $c_i$ only $r$ free cells were present, then these considerations would not hold. However, by construction this cannot occur.) The same reasoning can be made for a robot reaching $c_i$ and then coming back, in this case the total local cost would be $2(2r+1)$. Therefore, given a budget of $(2r+1)(n-1)$, if an exploration path exists, it will go from $c_j$ to $c_i$ and then proceed towards another node cell $c_z \neq c_j$. (Notice that, when $c_i$ is left at the end of the exploration path, then it must be visited to perceive the dummy cells surrounding it.) From the previous considerations it also follows that (c) any path with traveled distance less than $(2r+1)(n-1)$ cannot be an exploration path, i.e., it cannot perceive all the free cells of the environment.

Let us now reintroduce steps 4 and 5 in our construction. Such steps are mutually exclusive, so let us first consider step 4. Trivially, the presence of a degree 1 node prevents the existence of any Hamilton circuit in $G$. In this case, $O_G$ never admits an exploration path of length $(2r+1)(n-1)+2$ since a robot would have to travel $(2r+1)(n-1)$ units to cover the node cells and at least 3 additional units to sense the dummy cells inserted in step 4. Differently, step 5 adds two dummy cells that would require to spend 2 additional travel units, forcing any admissible exploration path to start and stop at them. Since the two additional cells are associated to neighboring nodes, from an exploration path of length $(2r+1)(n-1)+2$ we can easily build an Hamilton circuit for $G$ by dropping the first and last visits in the path and joining them in the path. In the example of Figure 3.5, an exploration path with length $(2r+1)(n-1)+2$ is $A' \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow B \rightarrow B'$, and the corresponding built Hamilton circuit is $A \rightarrow B \rightarrow C \rightarrow D \rightarrow B \rightarrow A$. (Notice that, our reduction requires to separately handle the degenerate case of a graph with two nodes of degree 1. Being such case trivial, we do not report it for the sake of presentation.) $\square$

## 3.2 Bounds on solutions quality

We now focus on the optimal solutions for Problems $C$ and $D$, both when considering the number of steps and the traveled distance as optimality criterion, and present some bounds and formal relations that bind solutions in the continuous and in the discrete.

First of all, we provide an analytical expression of the maximum area that can be perceived with a single perception in the continuous and in the discrete. This will be used in the following to compute lower bounds of the solutions. Given sensor range $r$ and cell edge length $e$, we call $p_D(r,e)$ the maximum cardinality set of perceivable cells in a grid $G^e$, or equivalently $p_D(r') = p_D(r,e)$, where $r' = \frac{r}{e}$. Formally, $p_D(r') = S^*$ such that $|S^*| = \max_{c \in G^e} |S_c|$, where $S_c = \{c' \in G^e \mid d(c,c') < r'\}$ represents the set of cells visible from $c_i$ and $d(\cdot)$ is the Euclidean distance. Similarly, for the continuous case, $p_C(r) = U_q^*$ such that $A(U_q^*) = \max_{q \in \mathscr{P}_f} A(U_q)$ where $U_q = \{q' \in \mathscr{P}_f \mid d(q,q') < r\}$ and where $A(\cdot)$ is the total continuous area covered by the argument, whether a grid or a polygon in $\mathbb{R}^2$. The maximum number of perceived cells is given in the following proposition.

**Proposition 3.3.** *Given an environment $\mathscr{P}$, a tiling grid $G^e$, and a robot with sensor range $r$:*

$$p_D(r') = 1 + 4\Big(\Big\lceil r' - 1 \Big\rceil\Big) + 4 \sum_{i=1}^{\lceil r'-1 \rceil} \Big(\Big\lceil \Big(\sqrt{r'^2 - i^2}\Big) - 1 \Big\rceil\Big).$$

*Proof.* The expression derives from a simple geometrical construction. The first term refers to the cell in which the robot is; the second one refers to the cells that are along the x-axis and y-axis of a coordinate system whose origin is the center of the cell in which the robot is; the last one refers to all the cells in each quarter. Note that in order to rule out cells whose center is distant from the robot position by exactly $r'$, we subtract one and we take the upper integral value. $\square$

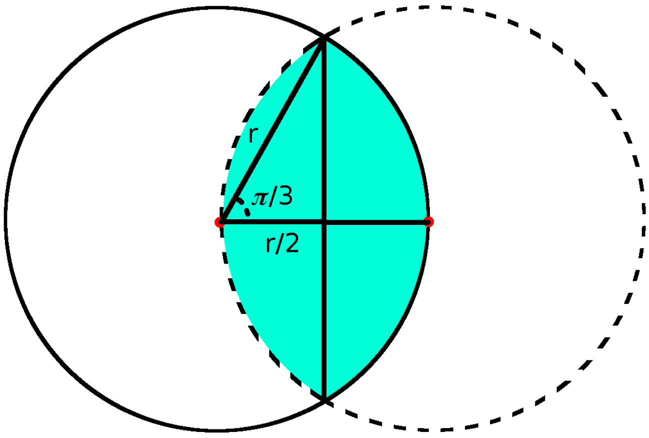


**Figure 3.6:** *Area to remove (cyan) from the perception circle (continuous line) given a robot with range $r$.*

Note that if $r = 1$ and $e = 1$, then $p_D(r') = 1$. This derives from the assumptions made on the sensor model: a cell is perceived when its center is at distance less than $r'$ from the position of the robot.

Consequently an upper bound on the maximum area that can be perceived in the discrete case is $A(p_D(r')) = e^2 p_D(r')$. No simple analytic relation holds between $A(p_D(r'))$ and $A(p_C(r))$ being it strongly dependent on $r$ and $e$. In particular, no inequality can be derived since, depending on $r$ and $e$, the first quantity could be greater than the second one or *vice versa*.

The above expressions do not consider the reachability constraint that, except for the first one, will affect any perception. To consider it, we define the *new* perceivable area, that is the area acquired with a perception from an already mapped location and that does not belong to the already mapped free area.

**Proposition 3.4.** *Given an environment $\mathscr{P}$, the maximum new area perceived by a robot with sensor range $r$ is*

$$A(p_C^{new}(r)) = \pi r^2 - 2(\frac{\pi}{3} r^2 - r^2 \sin(\frac{\pi}{3}) \cos(\frac{\pi}{3})) \simeq \frac{2}{3} \pi r^2.$$

*Proof.* The robot can make perceptions only from points located in the known free area, due to the reachability constraint. So, given $r$, an area of at least $2r^2(\frac{\pi}{3} - \sin(\frac{\pi}{3}) \cos(\frac{\pi}{3}))$ is already perceived (see Figure 3.6) and must be subtracted from the maximum perceivable area $\pi r^2$. □

**Proposition 3.5.** *Given an environment $\mathscr{P}$, a tiling grid $G^{\mathrm{e}}$, and a robot with sensor range $r$, the maximum number of new cells perceived is*

$$p_D^{new}(r') = p_D(r') - 2\Big(2 \sum_{i=\lceil\frac{\lceil r'-1\rceil}{2}\rceil + rem(\lceil r'\rceil,2)}^{\lceil r'-1\rceil} \Big(\Big\lceil\Big(\sqrt{r'^2 - i^2}\Big) - 1\Big\rceil\Big) + \Big(rem(\lceil r'\rceil,2)\Big\lceil\Big(\sqrt{r'^2 - (\lceil r'-1\rceil/2\rceil)^2}\Big) - 1\Big\rceil\Big)\Big) - \lceil\frac{\lceil r'-1\rceil}{2}\rceil$$

*where rem$(\cdot)$ is the remainder of division operation.*

*Proof.* The proof develops similarly to Proposition 3.4. The three final terms of the sum represent the number of already perceived cells (whose center falls in the area in cyan shown in Figure 3.6 and calculated using the same geometrical scheme used in Proposition 3.3):

$$\sum_{i=\lceil\frac{\lceil r'-1\rceil}{2}\rceil + \mathrm{rem}(\lceil r'\rceil,2)}^{\lceil r'-1\rceil} (\lceil(\sqrt{r'^2 - i^2}) - 1\rceil)$$

accounts for the cells whose center falls in the quarter of area in cyan, $\mathrm{rem}(\lceil r'\rceil, 2)\lceil(\sqrt{r'^2 - (\lceil r'-1\rceil/2\rceil)^2}) - 1\rceil$ corresponds to the cells whose centers possibly lie on half of the vertical line in the cyan area, and $\lceil\frac{\lceil r'-1\rceil}{2}\rceil$ refers to the cells whose centers are on the horizontal segment depicted inside the cyan area. These three terms are subtracted to $p_D(r')$. $\square$

Note that the robot travels at least $r$ and $\lceil r' - 1\rceil e$ units from its current location to perceive the maximum new area in the continuous and in the discrete cases, respectively. As intuition suggests, cells with smaller $e$ better approximate the continuous area.

**Proposition 3.6.** *Given an environment $\mathscr{P}$, a tiling grid $G^{\mathrm{e}}$, and a robot with sensor range $r$,* $\lim_{e\to 0} A(p_D(r')) = A(p_C(r))$.

*Proof.* The proof trivially derives from the fact that as $e$ goes to $0$ the area $e^2$ of the cell approaches to $0$, then the difference between the area covered by cells whose centers are in the circle and the area of the circle becomes smaller and smaller, thus leading the perceived area in the discrete to the same perceived area in the continuous. $\square$

### 3.2.1 Bounds on number of steps

Let us first consider Problems $C$-$\#$ and $D$-$\#$. We denote as $\bar{p}_C^{\mathrm{new}}(r, q_0)$ and $\bar{p}_D^{\mathrm{new}}(r', c_0)$ the new perceived area in a given point $q_0 \in \mathscr{P}_f$ and in a given cell $c_0 \in G^e_f$, respectively. From Propositions 3.4 and 3.5 we can derive lower bounds $\underline{n_C}$ and $\underline{n_D}$ for the optimal solution $n^*_C$ and $n^*_D$ over the number of steps of the optimal exploration path for Problems $C$-$\#$ and $D$-$\#$, respectively. Formally, we provide the following statement.

**Proposition 3.7.** *Given an environment $\mathscr{P}$, a tiling grid $G^e$, a robot with sensor range $r$ and with initial position $q_0 \in \mathscr{P}_f$ and $c_0 \in G^e_f$, to which $q_0$ belongs, we have the following lower bounds* $\underline{n_C} = 1 + \left\lceil \frac{A(\mathscr{P}_f) - A(\bar{p}^{new}_C(r,q_0))}{A(p^{new}_C(r))} \right\rceil \leq n^*_C$ *and* $\underline{n_D} = 1 + \left\lceil \frac{A(G^e_f) - A(\bar{p}^{new}_D(r',c_0))}{A(p^{new}_D(r'))} \right\rceil \leq n^*_D$.

*Proof.* For the continuous case the robot initially senses $A(\bar{p}^{\text{new}}_C(r, q_0))$ from $q_0$ and then it has to sense $A(\mathscr{P}_f) - A(\bar{p}^{\text{new}}_C(r, q_0))$ with a maximum perceivable new area of $A(p^{\text{new}}_C(r))$, as shown in Proposition 3.4. The discrete case develops similarly around Proposition 3.5. □

There is no clear evident relation between $n^*_C$ and $n^*_D$ as well as between $n^*_D$ in tiling grids $G^e$ and $G^{e'}$ with $e' \geq e$. Nevertheless, the cost of the solution in the discrete could decrease with $e$ because, as shown in the following proposition, the amount of continuous free area that can be represented with different discretizations of the environment $\mathscr{P}$ is less than or equal to the actual amount of continuous free area. We will assume that different grids share the same boundary.

**Proposition 3.8.** *Given an environment $\mathscr{P}$, two tiling grids $G^e$ and $G^{e'}$, if $\frac{e'}{e} = 2^k$ ($k \in \mathbb{Z}_{\geq 0}$), then $A(G^{e'}_f) \leq A(G^e_f) \leq A(\mathscr{P}_f)$ and $A(G^{e'}_o) \geq A(G^e_o) \geq A(\mathscr{P}_o)$.*

*Proof.* Given a point $q \in \mathscr{P}_o$, using a grid $G^e$, $q$ will belong to a cell $c \in G^e_o$. When using a grid $G^{e'}$, $q$ will belong to a cell $c' \in G^{e'}_o$. Given that $e' \geq e$, it is $A(c') \geq A(c)$. Note that as the different grids share the same boundary and as they have the relation $\frac{e'}{e} = 2^k$, by construction it cannot happen that a cell $c' \in G^{e'}$ is not considered obstacle, while the cells $c \in G^e$ that are contained in $c'$ are considered so. Thus, $A(G^{e'}_o) \geq A(G^e_o)$. The fact that $A(G^{e'}_f) \leq A(G^e_f)$ follows trivially. The relationship $A(G^e_o) \geq A(\mathscr{P}_o)$ easily derives considering that a point $q \in \mathscr{P}$ does not have any area or length, so, if $q \in \mathscr{P}_o$ belongs to a cell $c \in G^e_o$ the area of the cell $c$ is not smaller than the one of $q$. The relation $A(G^e_f) \leq A(\mathscr{P}_f)$ follows trivially. (Recall that, according to our discretization construction, each cell partially lying over the environment external boundary is by definition an obstacle cell.) □

Note however that, as expected, because of Proposition 3.6, when $e \to 0$ we have that $A(\mathscr{P}_f) = A(G^e_f)$ and so $n^*_D = n^*_C$.

### 3.2.2 Bounds on traveled distance

In this section we present bounds and relations between solutions for Problems $C$-$\mathscr{D}$ and $D$-$\mathscr{D}$. Let us start with a simple lower bound $\underline{\mathscr{D}_C}$ and $\underline{\mathscr{D}_D}$ for the traveled distance of an optimal solution $\mathscr{D}^*_C$ and $\mathscr{D}^*_D$.

**Proposition 3.9.** *Given an environment $\mathscr{P}$, a tiling grid $G^e$, and a robot with sensor range $r$, $\underline{\mathscr{D}_C} = (\underline{n_C}-1)r \leq \mathscr{D}^*_C$ and $\underline{\mathscr{D}_D} = (\underline{n_D}-1)\lceil r'-1\rceil e \leq \mathscr{D}^*_D$.*

*Proof.* To completely cover the environment, the robot should at least perform $\underline{n_C}$ steps in the continuous case and $\underline{n_D}$ in the discrete case, as shown in Proposition 3.7. Because of the reachability constraint (Section 3.1), the robot should travel a distance at least $r$ and $\lceil r' - 1\rceil e$ from its current location in order to perform a maximum perception $A(p^{\text{new}}_C(r))$ and $A(p^{\text{new}}_D(r'))$ at the next step, respectively. Clearly, the initial perception performed in $q_0$ ($c_0$) does not require any movement by the robot. □

Note that, also in this case, in general, there is not a clear relation between $\mathscr{D}^*_D$ and $\mathscr{D}^*_C$, as it strongly depends on $r$, $e$, and $\mathscr{P}$. Similarly to the case relative to the number of steps, no clear pattern can be identified between $\mathscr{D}^*_D$ in environments $G^e$ and $G^{e'}$ with $e' \geq e$. When $e \to 0$, then the difference between the perceived area in the continuous case and in the discrete case goes to 0, as shown in Section 3.2.1, and $\mathscr{D}^*_D$ goes to $\mathscr{D}^*_C$.

Now, consider an exploration path $\mathcal{C} = \langle c_0, c_1, \dots, c_n\rangle$ in the discrete case and the corresponding exploration path $Q = \langle q_0, q_1, \dots, q_n\rangle$ in the continuous case (not necessarily optimal). According to Nash [91], which provides an approximation upper bound on the path on a grid with respect to a continuous path between two arbitrary points, we have the following upper bound on length $\mathscr{D}_D$ of the path $\mathcal{C}$ with respect to the length $\mathscr{D}_C$ of the path $Q$.

**Proposition 3.10.** *Given an exploration path $\mathcal{C} = \langle c_0, c_1, \dots, c_n\rangle$ in $G^e$ and an exploration path $Q = \langle q_0, q_1, \dots, q_n\rangle$ in $\mathscr{P}$, if $q_i = c_i$ for all $i$, then $\mathscr{D}_D \leq 1.08\mathscr{D}_C$.*

*Proof.* Let us take every pair of consecutive points $q_{i-1}$ and $q_i$ and the corresponding cells $c_{i-1}$ and $c_i$. Let us consider the length of the sub-path connecting these consecutive points $\mathscr{D}_C(q_{i-1}, q_i)$ and $\mathscr{D}_D(c_{i-1}, c_i)$. Considering the results obtained by Nash [91], we have that $\mathscr{D}_D(c_{i-1}, c_i) \leq$

$1.08\mathscr{D}_C(q_{i-1}, q_i)$ and then $\sum_{i=1}^{|\mathcal{C}|-1} \mathscr{D}_D(c_{i-1}, c_i) \leq 1.08 \sum_{i=1}^{|Q|-1} \mathscr{D}_C(q_{i-1}, q_i)$. □

Proposition 3.10 holds for all exploration paths in a two-dimensional grid environment, and so is true also for the optimal exploration path.

If the continuous environment $\mathscr{P}$ is tiled with $G^e$ where $e$ is very small, we have the following proposition for the length of an exploration path $\mathscr{D}_D^e$ in the tiling grid $G^e$.

**Proposition 3.11.** *Given an exploration path $\mathcal{C} = \langle c_0, c_1, \ldots, c_n \rangle$ in $G^e$ and an exploration path $Q = \langle q_0, q_1, \ldots, q_n \rangle$ in $\mathscr{P}$, if $q_i = c_i$ for all $i \in [0, \ldots, n]$, $\lim_{e \to 0} \mathscr{D}_D^e = \mathscr{D}_C$.*

*Proof.* As $e$ goes to 0, $\mathscr{P}$ is tiled in a increasingly fine-grained way such that $\mathscr{D}_D(c_{i-1}, c_i) = \mathscr{D}_C(q_{i-1}, q_i) \leq 1.08\mathscr{D}_C(q_{i-1}, q_i)$ for every pair of consecutive cells $c_{i-1}$ and $c_i$ and the corresponding points $q_{i-1}$ and $q_i$. Then, summing all sub-paths lengths proves the proposition. □

We now look at how the length of an exploration path $\mathcal{C} = \langle c_0, c_1, \ldots, c_n \rangle$ changes using different cell sizes. Again, we will assume that different grids share the same boundary even if some of the results presented can be extended to the more general case. Let us first introduce some considerations about the center of the cells belonging to grids with different edge lengths.

**Proposition 3.12.** *Given an environment $\mathscr{P}$, two tiling grids $G^e$ and $G^{e'}$, where $e' \geq e$, a number $(\lfloor \frac{e'}{e} \rfloor)^2$ of cells $c \in G^e$ will have their centers in the same cell $c' \in G^{e'}$. We say that these cells $c$ belong to cell $c'$.*

*Proof.* Grids $G^e$ and $G^{e'}$ are composed of square cells whose edge sizes are $e$ and $e'$, respectively. So, centers are distanced of $e$ and $e'$ in $G^e$ and $G^{e'}$. Hence, a number $(\lfloor \frac{e'}{e} \rfloor)^2$ of cells $c \in G^e$ have their centers contained in a cell $c' \in G^{e'}$. □

Given grids $G^e$ and $G^{e'}$, the following proposition allows to map positions of the robot in center of cells $c \in G^e$ to center of cells $c' \in G^{e'}$, considering that cells $c \in G^e$, as said in Proposition 3.12, belong to cell $c'$.

**Proposition 3.13.** *Given an environment $\mathscr{P}$, two tiling grids $G^e$ and $G^{e'}$, where $e' \geq e$, starting from left to right and from top to bottom, the center of a cell $c \in G^e$ with coordinates $(c_x, c_y)$ which belongs to a cell $c' \in G^{e'}$ can be moved to the center of $c'$ by a translation, with respect to a global coordinate system, $\mathbf{v}(c) = [v_x, v_y]^\mathsf{T} = \left[\frac{e'/e-1}{2} - rem(\frac{c_x}{e'}), \quad \frac{e'/e-1}{2} - rem(\frac{c_y}{e'})\right]^\mathsf{T}$, where $rem(\cdot)$ is the remainder of division operation.*

*Proof.* The proof derives from the way the environment $\mathscr{P}$ is tiled with grid $G^e$ and grid $G^{e'}$. The term $\frac{e'/e-1}{2}$ accounts for the shift of the center of a cell $c' \in G^{e'}$ with respect to the upper left cell $c \in G^e$ that belongs to $c'$. The term *rem*$(\frac{c_x}{e'})$ (*rem*$(\frac{c_y}{e'})$) correctly shifts cells $c \in G^e$ that belong to cell $c' \in G^{e'}$, according to Proposition 3.12. □

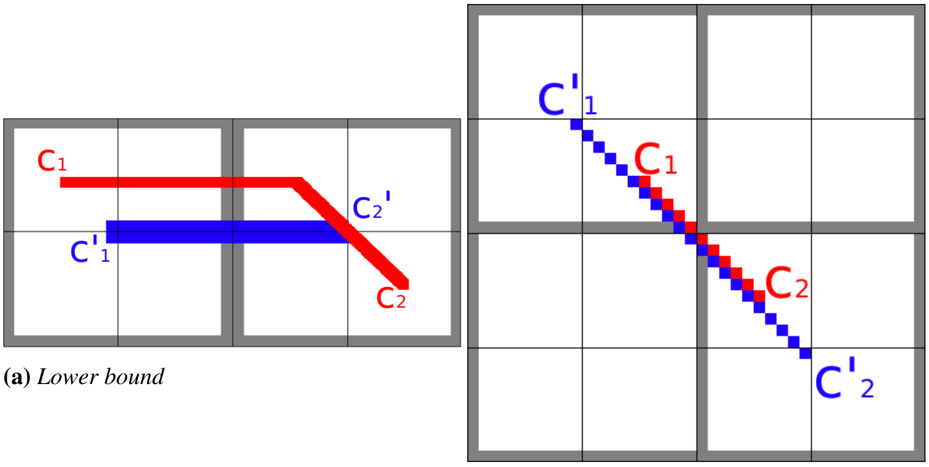
A sufficient condition to have a cell of $G^{e'}$ with the same center of any cell of $G^e$ is that $e' = 3ke$ (with $k = 0, 1, 2, \ldots$). Note that, if we remove the assumption that the grids $G^e$ and $G^{e'}$ share the upper left corner, the number of cells $c \in G^e$ belonging to a cell $c' \in G^{e'}$ is again $(\lfloor \frac{e'}{e} \rfloor)^2$, but the translation $\mathbf{v}(c)$ calculated in Proposition 3.13 changes.

The way to map positions of the robot in center of cells $c \in G^e$ to center of cells $c' \in G^{e'}$ is exploited in the following proposition which shows the relation between path lengths in different grids, sharing common reference system.

**Proposition 3.14.** *Given two exploration paths $\mathcal{C} = \langle c_0, c_1, \ldots, c_n \rangle$ in $G^e$ and $\mathcal{C}' = \langle c'_0, c'_1, \ldots, c'_n \rangle$ in $G^{e'}$, if $c_i$ belongs to $c'_i$ for all $i \in [0, \ldots, n]$, the following lower/upper bounds for the ratio $\frac{\mathscr{D}^{e'}_D}{\mathscr{D}^e_D}$ between the length of $\mathcal{C}$ and $\mathcal{C}'$ in $G^e$ and $G^{e'}$, where $\frac{e'}{e} = 2^k$, with $k \in \mathbb{Z}_{\geq 0}$, hold:*

$$\frac{1}{(1 + (1 - 1/2^k)\sqrt{2})} \leq \frac{\mathscr{D}^{e'}_D}{\mathscr{D}^e_D} \leq 2^k.$$

*Proof.* Let us first derive the lower bound for $\frac{\mathscr{D}^{e'}_D}{\mathscr{D}^e_D}$. Remember that, because of the robot motion model we defined on a grid, the robot has to go along the shortest path on the grid, namely traversing the centers of cells (8-connected), and thus possibly not going directly straight from $c_i$ to $c_{i+1}$. Minimum distance between two cells with centers $c_1 = (x_1, y_1)$ and $c_2 = (x_2, y_2)$ in a grid map with edge length $e$ ($c_1$ in line of sight of $c_2$, and *vice versa*) is calculated as: $\mathscr{D}^e_D(c_1, c_2) = e(v + o\sqrt{2})$, where $v = ||x_1 - x_2| - |y_1 - y_2||$ is the number of straight movements and $o = \frac{|x_1-x_2|+|y_1-y_2|-v}{2}$ is the number of oblique movements ($o$ is always an integer, because the idea of the formula is to perform first all the necessary straight movements, so that the robot reaches a cell where it has to perform only oblique movements). Note that it is always possible to normalize the $x$ and $y$ of the cells as the grid is regular. Intuitively, the lower bound derives from the fact that the robot, when moving in $G^{e'}$, could require a shorter traveled distance due to the shift of the cell centers and some oblique movements could be not necessary anymore. Basically the proof



**(a)** *Lower bound*

**(b)** *Upper bound*

**Figure 3.7:** *Bounds of the ratio between traveled distances in grid maps with different edges (black lines delimit cells with edge* $e$*, grey lines delimit cells with edge* $e'$*, red line and blue line refer to a portion of the path with edge* e *and* e'*, respectively).*

develops from considering the path in $G^e$ between the most distant cells $c_1$ and $c_2$ of $G^e$ that belong to two adjacent cells $c_1'$ and $c_2'$ in $G^{e'}$. The path in $G^{e'}$ that connects the two adjacent cells is shorter than the path in $G^e$ (see Figure 3.7a). Without loss of generality, consider cells $c_1 = (x, y)$ and $c_2 = (x + 2e'/e - 1, y + e'/e - 1)$ in $G^e$ that belong to two adjacent cells in $G^{e'}$, with coordinates (with respect to a fixed coordinate system) $c_1' = (x+\frac{e'/e-1}{2}, y+\frac{e'/e-1}{2})$ and $c_2' = (x+2e'/e-1-\frac{e'/e-1}{2}, y+e'/e-1-\frac{e'/e-1}{2})$, obtained applying the transformation of Proposition 3.13. Taking adjacent cells on the oblique accounts for $\sqrt{2}$ for both $\mathscr{D}_D^{e'}$ and $\mathscr{D}_D^{e}$, and so by calculating the ratio, $\sqrt{2}$ simplifies. Note that the considered distances $\mathscr{D}_D^{e}$ and $\mathscr{D}_D^{e'}$ are normalized to the distance unit $e$. To reason with the distance unit $1$, it is necessary to multiply both distances by $e$. After some passages, we obtain the number of straight movements in $G^e$, which is $v = ||2e'/e - 1| - |e'/e - 1||$.

Since $e'/e \geq 1$, we have $v = e'/e$. Similarly, the number of oblique movements in $G^e$ is $o = \frac{|2e'/e-1|+|e'/e-1|-e'/e}{2}$. Since $e'/e \geq 1$, we have $o = e'/e - 1$.

In the grid map $G^{e'}$, the number of straight movements is $v = e'/e$, and the number of oblique movements is $o = 0$.

So the lower bound is $\frac{\mathscr{D}_D^{e'}}{\mathscr{D}_D^{e}} \geq \frac{e'/e}{(e'/e+(e'/e-1)\sqrt{2})}$, as this reduction can cumulate along the paths whose costs are $\mathscr{D}_D^{e'}$ and $\mathscr{D}_D^{e}$. With some simplifications,

| | | Problem | |
|---|---|---|---|
| | | $C$ | $D$ |
| Optimality criterion | # | $\underline{n_C} = 1 + \left\lceil \frac{A(\mathscr{P}_f) - A(\bar{p}_C^{\text{new}}(r, q_0))}{A(p_C^{\text{new}}(r))} \right\rceil \leq n_C^*$ | $\underline{n_D} = 1 + \left\lceil \frac{A(G_f^e) - A(\bar{p}_D^{\text{new}}(r', c_0))}{A(p_D^{\text{new}}(r'))} \right\rceil \leq n_D^*$ |
| | $\mathscr{D}$ | $\underline{\mathscr{D}_C} = (\underline{n_C} - 1)r \leq \mathscr{D}_C^*$ | $\underline{\mathscr{D}_D} = (\underline{n_D} - 1)\lceil r' - 1 \rceil e \leq \mathscr{D}_D^*$ |

**Table 3.1:** *Lower bounds for the optimal exploration solution considering the number of steps # (see Proposition 3.7) and the traveled distance $\mathscr{D}$ (see Proposition 3.9) for the Problems C and D, where $A(\mathscr{P}_f)$ ($A(G_f^e)$) computes the area of the free space in the continuous (discrete, where the edge length is e), $\bar{p}_C^{new}(r, q_0)$ ($\bar{p}_D^{new}(r', c_0)$) is the new area perceived in the point $q_0$ ($c_0$) given the range r ($r' = \frac{r}{e}$), $A(p_C^{new}(r))$ ($A(p_D^{new}(r'))$) is the maximum new perceivable area in the continuous (discrete) by the robot.*

we obtain:

$$\frac{\mathscr{D}_D^{e'}}{\mathscr{D}_D^{e}} \geq \frac{1}{(1 + (1 - e/e')\sqrt{2})} = \frac{1}{(1 + (1 - 2^k)\sqrt{2})}.$$

The upper bound comes from the fact that the robot can travel longer distance in $G^{e'}$ than in $G^e$, because cell centers are shifted in the grid map with edge $e'$, as shown in Proposition 3.13. The idea of the proof is to consider the shortest path between two nearby cells in $G^e$ that belong to two adjacent cells in $G^{e'}$ (see Figure 3.7b). Hence, consider, without loss of generality, two adjacent cells $c_1 = (x, y)$ and $c_2 = (x + 1, y + 1)$ in $G^e$ (the result holds also for horizontal or vertical adjacent cells). Suppose that $c_1$ belongs to $c_1'$ and $c_2$ belongs to $c_2'$, where $c_1', c_2' \in G^{e'}$ are adjacent, and specifically, their centers are $c_1' = (x - (\frac{e'/e-1}{2}), y - (\frac{e'/e-1}{2}))$ and $c_2' = (x + 1 + (\frac{e'/e-1}{2}), y + 1 + (\frac{e'/e-1}{2}))$. The distance between $c_1$ and $c_2$ is $\mathscr{D}_D^e(c_1, c_2) = \sqrt{2}$. The distance between $c_1'$ and $c_2'$ is $\mathscr{D}_D^{e'} \frac{e'}{e\sqrt{2}}$. Therefore, the (worst-case) upper bound is $\frac{\mathscr{D}_D^{e'}}{\mathscr{D}_D^{e}} \leq e'/e = 2^k$, because the same ratio holds for every subpath in $\mathcal{C}$ and $\mathcal{C}'$. $\square$

Note that there could be a degenerate case in which the path $\mathcal{C}$ in $G^e$ can be fully contained in a single cell $c'$ of $G^{e'}$, when the whole free space $G_f^{e'}$ can be covered from the center of $c'$. In such a case, the traveled distance in $G^{e'}$ is obviously 0, and the lower bound of previous proposition does not hold. Note also that, when the edge length $e$ tends to 0, the lower bound of the ratio $\frac{\mathscr{D}_D^{e'}}{\mathscr{D}_D^{e}}$ becomes $\frac{1}{1+\sqrt{2}}$.

The above results (summarized in Tables 3.1 and 3.2) provide interesting theoretical insights on our problem, especially analyzing how the use of discrete representations can affect the quality of the solutions. Starting from

| Relation | Bounds |
|---|---|
| $\mathscr{D}_C$-$\mathscr{D}_D$ | $\mathscr{D}_D \leq 1.08\mathscr{D}_C$ |
| $\mathscr{D}_D^{\mathrm{e}}$-$\mathscr{D}_D^{\mathrm{e}'}$ | $\frac{1}{(1+(1-1/2^k)\sqrt{2})} \leq \frac{\mathscr{D}_D^{\mathrm{e}'}}{\mathscr{D}_D^{\mathrm{e}}} \leq 2^k$ |

**Table 3.2:** *Relations between length of the paths in the continuous $\mathscr{D}_C$ and in the discrete $\mathscr{D}_D$ (see Proposition 3.10), and in the discrete $\mathscr{D}_D^{\mathrm{e}}$ and $\mathscr{D}_D^{\mathrm{e}'}$, where the edge lengths have the relation $\frac{\mathrm{e}'}{\mathrm{e}} = 2^k$, with $k \in \mathbb{Z}_{\geq 0}$ (see Proposition 3.14).*

these basic results, in the next section we try to enrich them by discussing a practical approach to tackle resolution of the optimal exploration problem.

## 3.3 Solving the optimal exploration problem

In the following, we show how we tackle Problem *D*, by presenting its formulation as a search problem and the solving approach together with some speedup techniques.

### 3.3.1 Formulation as a search problem

To practically solve Problem *D*, we formulate a corresponding search problem, following a classical approach in Artificial Intelligence [112, Chapter 3]. The solution of the search problem is a (approximate) solution of Problem $D$. A state $s$ is a pair $(c, M^{G^{\mathrm{e}}})$ composed of the current position $c$ of the robot in $G_f^{\mathrm{e}}$ and of the map $M^{G^{\mathrm{e}}} \subseteq G^{\mathrm{e}}$ built so far during exploration. Our search problem for solving Problem $D$ is formulated as follows.

**Initial state**. The initial state $s_0 = (c_0, M_0^{G^{\mathrm{e}}})$ is given by the initial position of the robot $c_0$ in $G_f^{\mathrm{e}}$ and by the initial map $M_0^{G^{\mathrm{e}}}$, which contains the cells of $G^{\mathrm{e}}$ perceived from $c_0$.

**Actions**. From a state $s = (c, M^{G^{\mathrm{e}}})$, applicable actions for the robot are to move to a free cell $c' \in M^{G_f^{\mathrm{e}}}$ reachable from $c$ and perceive the environment surrounding $c'$. A free cell $c'$ is reachable from $c$ when there is a safe path (within free cells of $M^{G_f^{\mathrm{e}}}$ and not colliding with any obstacle) between $c$ and $c'$, according to our reachability constraint. The path is calculated using a wavefront propagation algorithm on $M^{G^{\mathrm{e}}}$ [76, Chapter 8], considering cost $e$ for vertical and horizontal movements and cost $e\sqrt{2}$ for diagonal movements (recall that we consider 8-connected grids). In principle, from a state $s = (c, M^{G^{\mathrm{e}}})$, there are as many actions as many reachable free cells $c'$ in the current map $M^{G^{\mathrm{e}}}$. However, to limit the number of these actions (and the branching factor of the search tree used to calculate a solution), we consider only reachable frontier free cells $c'$ that are on the boundary between known and unknown parts of the map $M^{G^e}$, as usually done in practical scenarios of robotics (Section 2.2). This does not prevent our approach to find the optimal solution when considering Problem $D$-$\#$ because the new area perceived from any non-frontier cell is not larger than the area perceived from at least a frontier cell. However, this way of selecting actions can produce sub-optimal solutions for Problem $D$-$\mathscr{D}$. In our case, the robot reaches always the boundary between the known and unknown part of the environment, but there could be some cases in which the robot performs better if it stops before reaching the boundary, for example, when the robot is in a dead-end corridor, or when two frontiers are visible from another third location and going to one of them requires a second scan. So, there could be an extra traveled distance at each step, which is $\delta_{D_i} = \hat{\delta}_{D_i} - \delta^*_{D_i}$ ($\delta_{D_i} \in [0, r]$), where $\hat{\delta}_{D_i}$ is the traveled distance determined by our approach considering destination location $c_i$ on the frontier and $\delta^*_{D_i}$ is the minimum distance that should have actually traveled to perceive the same portion of the environment.

**Transition function**. The new state resulting from performing applicable action "move to $c'$" in state $s = (c, M^{G^e})$ is $s' = (c', M'^{G^e})$, where $M'^{G^e}$ is the map $M^{G^e}$ updated with the new perception in $c'$.

**Goal test**. A state $s = (c, M^{G^e})$ is a goal state when $M^{G^e}$ is a complete map of the free space of the environment $G^e$, namely when all the free cells of $G^e_f$ are present in $M^{G^e}$.

**Step cost**. The step cost for going from a state $s = (c, M^{G^e})$ to a successor state $s' = (c', M'^{G^e})$ reflects the optimality criteria we consider and can be either $\mathfrak{c}_{\#} = 1$, when the optimality criterion is the number of steps, or $\mathfrak{c}_{\mathscr{D}} = \mathscr{D}(c, c')$, when the optimality criterion is the traveled distance.

A solution to the above search problem is a finite sequence of states $S = \langle s_0, s_1, \ldots, s_n \rangle$ such that $s_0$ is the initial state and $s_n$ is a state that satisfies the goal test. An optimal solution is a solution with minimum cost. From a solution $S$, a corresponding solution $\mathcal{C}$ to Problem $D$ can be derived immediately.

It is worth noting that searching for an optimal exploration path is different from searching for an optimal path between two given points in an unknown environment (e.g., using algorithms like D* [76, Chapter 12], Learning Real-Time A* [112, Chapter 4], and PHA* [45]). First, in our problem, we don't know *a priori* the position of the robot at the end of exploration. Hence, we cannot operate in a state space in which each state is a position (cell) of the robot in the environment, but we need a more complex representation of states that accounts also for the portion of the environment discovered so far. Second, our approach is offline and the robot is not required to physically move between states (positions).

### 3.3.2 Solution of the search problem

In principle, we can employ any search algorithm, but from some preliminary experiments, we assessed that A$^*$ performs slightly better than branch and bound and other search algorithms. Despite the choice of an algorithm can be further investigated, it lies outside the scope of this dissertation and so, for the experimental activities presented here, we decided to use A$^*$. In order to apply it, we need to define a heuristic function that, given a state $s$, returns the estimated cost of a solution from $s$.

For both optimality criteria, given a tiled environment $G^e$, a robot with sensor range $r$, and an initial robot position $c_0$, the heuristics $h_{\#}(s)$ and $h_{\mathscr{D}}(s)$ are calculated by exploiting a precomputed solution $S'$ of a relaxed version of the problem obtained by setting $e' > e$ and $r' > r$ (for example, in environments of Figure 3.9, we empirically found as values that preserve the heuristics admissibility in the settings we considered $e' = 2e$ and $r' = 1.1r$). Given $S'$, the heuristic value for a state $s$, which is met in searching for a solution $S$ to the original problem is obtained as follows. At $s$ (which could possibly belong to the solution $S$), the robot has mapped a percentage $\mathfrak{p} = \dfrac{|\{c \mid c \in M^{G^e_f}\}|}{|G^e_f|}$ of the free space of the environment. The heuristic value $h_{\#}(s)$ or $h_{\mathscr{D}}(s)$ of $s$ is calculated as the cost incurred to reach $s' \in S'$ starting from $s_0 \in S'$, where $s'$ is the first state encountered in $S'$ such that the mapped percentage $\mathfrak{p}'$ in $s'$ is greater than or equal to the mapped percentage $\mathfrak{p}$ in $s$. In other terms, this means that the cost of each state of the solution $S'$ in the relaxed problem is a lower bound to a related state according to the percentage $\mathfrak{p}$ in the original problem.

Now let us illustrate how the relaxed problem is solved to obtain $S'$. For the number of steps, the heuristic function of the relaxed problem $\bar{h}_{\#}(s')$, with $s' = (c', M^{G^{e'}})$, is calculated as the number of unknown free cells divided by the maximum number of cells perceivable from a single perception:

$$\bar{h}_{\#}(s') = \frac{|\{c' | c' \in G^{e'}_f, c' \notin M^{G^{e'}_f}\}|}{p^{\text{new}}_D(r')}, \tag{3.3}$$

where $p^{\text{new}}_D(r')$ is the maximum number of new perceivable cells when sensor range is $r'$ and grid has cell size $e'$ (see Proposition 3.5). The heuristic function is admissible as it is the lower bound for the exploration problem considering the number of steps as optimality criterion, as shown in Proposition 3.7. As a consequence, solving with A* and $\bar{h}_{\#}(s')$ guarantees to find an optimal solution for the relaxed problem [112, Chapter 3]. Now, $h_{\#}(s)$ is greater than $\bar{h}_{\#}(s')$, because $\bar{h}_{\#}(s')$ does not entirely take into account the structure of the environment, while $h_{\#}(s)$ is a solution to the relaxed problem in a specific environment. Moreover, $h_{\#}(s)$ is admissible, being solution of a relaxed problem. Thus, A*, using $h_{\#}(s)$, expands a number of nodes less than or equal to that of using $\bar{h}_{\#}(s')$ in order to find the optimal solution.

For the traveled distance, the heuristic function of the relaxed problem $\bar{h}_{\mathscr{D}}(s')$, with $s' = (c', M^{G^{e'}})$, is simply calculated as the heuristic for the number of steps $\bar{h}_{\#}(s')$ multiplied by sensor range $r'$ and the cell edge length $e'$:

$$\bar{h}_{\mathscr{D}}(s') = \frac{|\{c' | c' \in G_f^{e'}, c' \notin M^{G_f^{e'}}\}|}{p_D^{\text{new}}(r')} r' e'. \tag{3.4}$$

The idea is that, in order to completely map the free space of environment $G^e$, the robot has to acquire a number of perceptions, and each scan is at least performed at the boundary of perceived area at each step. The above heuristic function is admissible because it is the lower bound for the exploration problem considering the traveled distance as optimality criterion, as shown in Proposition 3.9. Similar to the case of number of steps, $h_{\mathscr{D}}(s)$ is greater than $\bar{h}_{\mathscr{D}}(s)$, as $\bar{h}_{\mathscr{D}}(s')$ does not totally consider the structure of the environment, while $h_{\mathscr{D}}(s)$ does. Therefore, also when considering distance as optimality criterion, less nodes are expanded with $h_{\mathscr{D}}(s)$ than with $\bar{h}_{\mathscr{D}}(s')$.

Given the heuristic functions, we provide a sketch about how the basic A* operates. Starting from the initial state, the goal test is applied. If it is true, then the search ends. Otherwise, the algorithm picks the first state from a list of states, to which the transition function has been not applied yet and ordered in an ascending fashion according to their cost summed to the heuristic function, and checks again whether the selected state is a goal. This process goes on until the goal state is encountered. The optimality and the completeness on such state space is guaranteed by the properties of A*.

### 3.3.3 Speeding up solution computation

The worst-case computational complexity of our algorithm is exponential in the number of steps needed to completely map an environment (i.e., in

$n$). To tackle this limitation in practice, we list a number of speedup techniques that are expected to reduce the computational effort, at the expense of possibly worsening the quality of the solutions.

**Footprint sensor**. The laser range scanner can be substituted by a (less realistic) footprint sensor that perceives the state (free or occupied) of any cell whose center lies within the circle centered in the robot and with radius $r$, without considering the presence of obstacles. Using the footprint sensor preserves the completeness of the algorithm but could underestimate the cost of the solution obtainable with the more realistic laser range scanners.

**Weaker goal test**. We can consider a weaker goal test for which a state $s = (c, M^{G^e})$ ($s' = (c', M^{G^{e'}})$) in the original search problem (in the relaxed problem) is a goal state when $M^{G^e}$ ($M^{G^{e'}}$) contains a fraction $\mathfrak{g}$ ($0 \leq \mathfrak{g} \leq 1$) of the cells in $G^e_f$ ($G^{e'}_f$). This relaxation can better match the requirements of some relevant applications. Think, for example, to a search and rescue scenario where an incomplete map of the environment providing a general idea of its structure can suffice instead of a fully detailed map. The heuristic functions for the relaxed problem are still admissible and become:

$$\bar{h}'_{\#}(s') = \frac{|\{c' | c' \in G^{e'}_f, c' \notin M^{G^{e'}_f}\}| \cdot \mathfrak{g}}{p^{\text{new}}_D(r')}, \tag{3.5}$$

$$\bar{h}'_{\mathscr{D}}(s') = \frac{|\{c' | c' \in G^{e'}_f, c' \notin M^{G^{e'}_f}\}| \cdot \mathfrak{g}}{p^{\text{new}}_D(r')} r' e'. \tag{3.6}$$

**Clustering frontier cells**. Adjacent frontier cells in the current map $M^{G^e}$ are grouped in clusters $\mathfrak{C}$, according to the 8-adjacency of the grid. The clusters are limited in size. If a cluster $\mathfrak{C}$ with $b$ frontier cells contains more than $2/3$ of $|p'_D(r,e)| = |\{c \in p_D(r,e) \mid |\text{adjacency}(c)| \leq 5\}|$, where $|\text{adjacency}(c)|$ is the cardinality of the set of free known cells 8-adjacent to $c$, then $\mathfrak{C}$ is split in clusters $\mathfrak{C}'$ of fixed equal size, starting from left to right, up to down, so that $|\mathfrak{C}'| < |p'_D(r,e)|/2$. (The value $|p'_D(r,e)|$ represents the cardinality of the set of cells on the boundary of the footprint area; we empirically tried different values $\pm 10\%$ for this threshold obtaining similar results.) Limiting the size of the clusters allows to avoid misrepresenting a cluster when a representative cell is elected among those belonging to it. Such cell is simply defined as the one closest to the cluster's centroid. More precisely, for a cluster $\mathfrak{C}$ with $b \geq 1$ frontier cells $c_1, c_2, \dots, c_b$ the representative cell $c_{\mathfrak{C}}$ is selected as

$$c_{\mathfrak{C}} = \arg\min_{c_i \in \mathfrak{C}} \mathscr{D}\left(c_i, \text{centroid}(\mathfrak{C})\right), \tag{3.7}$$

where centroid$(\mathfrak{C})$ is a function that computes the centroid point of the cluster $\mathfrak{C}$, namely, the centroid's $x$ and $y$ are obtained by an average over the $x$ and $y$ of the $b$ cells $c_j$ belonging to $\mathfrak{C}$. Then, only cells $c_{\mathfrak{C}}$ representative of different disjoint clusters of frontier cells in $M^{G^e}$ are considered for generating actions in a state $s = (c, M^{G^e})$, drastically reducing the number of these actions and the branching factor of the search tree. This speedup technique preserves the completeness of the algorithm, because, given a cluster, at least one cell is kept, but the solution could be sub-optimal, as the exploration path is searched on a subset of reachable cells.

**Eliminating small clusters**. We can discard small clusters that contain less than $k$ frontier cells. The rationale is that their contribution to the exploration of the environment is small. If small clusters are eliminated, then the search algorithm could be prevented from finding any solution when some free cells of $G^e$ are visible only reaching a small cluster.

**Duplicate states list**. When selecting a state to expand, it is possible to discard duplicates that have been already expanded. Specifically, a state $s' = (c', M'^{G^e})$ can be safely discarded if there exists a state $s = (c, M^{G^e})$ already expanded such that $c' = c$, $M'^{G^e} \subseteq M^{G^e}$, and $g(s') \geq g(s)$, where $g(s)$ is the cost to reach the state $s$ from the initial state $s_0$. The completeness is guaranteed because duplicate states will eventually lead to the same states already expanded. Moreover, optimality is guaranteed as the algorithm does not consider duplicate states whose cost is greater than the cost of at least one already expanded equal state.

Summarizing, starting from the solution that can be found for Problem $C$ to the search problem with some speedup techniques defined here, we have the approximation scheme illustrated in Figure 3.8.

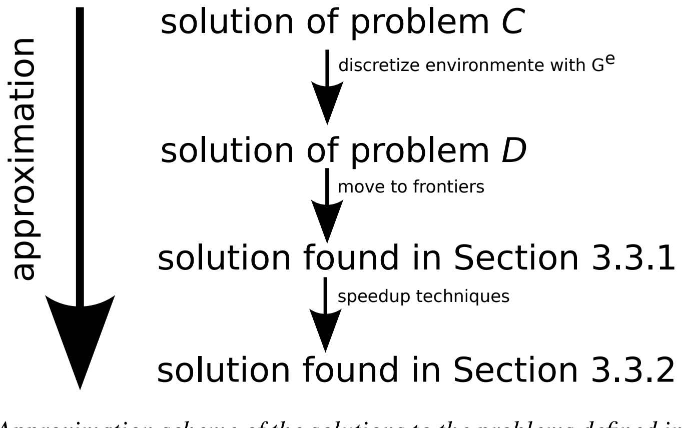

approximation
solution of problem C
discretize environmente with G^e
solution of problem D
move to frontiers
solution found in Section 3.3.1
speedup techniques
solution found in Section 3.3.2

**Figure 3.8:** *Approximation scheme of the solutions to the problems defined in this chapter.*

## 3.4 Simulation results

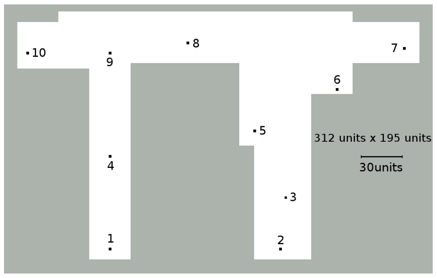
In order to assess the validity of our approach in finding (approximations of) optimal exploration paths, extensive simulated experiments have been conducted in three different environments. They are called *indoor*, *openspace*, and *obstacles*, and are shown in Figure 3.9. The line segment reported in the figure measures $30$ units. If we consider a unit equivalent to $0.1$ m (which is reasonable, given the cell size and the sensor range values discussed below), we can say that the size of the environments is realistically large. Also, these environments have been used to compare some online exploration strategies in [2]. The environments are discretized in square grids, using three different resolutions. We consider three cell sizes, corresponding to edge length $e$ of $1$, $2$, and $4$ units. We consider three values for the sensor range $r$, namely 20, 25, and 30 units, and three values for the weaker goal test $\mathfrak{g}$, namely $0.85$, $0.90$, and $0.95$. For an environment, we call *setting* a combination of $e$, $r$, and $\mathfrak{g}$. For each setting and for each of the $10$ random initial positions (shown as points in Figure 3.9), we run our approach[1] using $\mathfrak{c}_{\#}$ or $\mathfrak{c}_{\mathscr{D}}$ as cost function. We set a timeout of $10$ hours for each run.

In all experiments in the three environments we adopt the footprint sensor, the clustering of frontier cells, and the duplicate states list. Table 3.3 reports the complete experimental results for the indoor environment, where the elimination of small clusters is not used as speedup technique. The values reported in each entry are the average and the standard deviation (in parentheses) over the runs that terminated (maximum of 10, corresponding to the initial positions) for the corresponding setting. We also report the number of runs that have not terminated within the timeout.

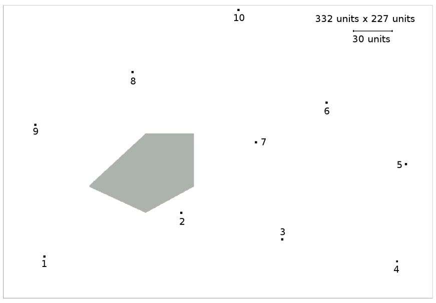
From Table 3.3, it emerges that both the number of steps and the traveled distance decrease when the sensor range $r$ increases. Unsurprisingly, a robot with a wider sensor can explore the environment more efficiently. Another expected behavior is that the number of steps and the traveled distance increase when the robot is required to explore an increasingly larger fraction $\mathfrak{g}$ of the environment. Solutions' cost decreases with the cell size $e$. Indeed, when increasing the cell size, less cells corresponding to free space are generally available to cover, as Proposition 3.8 shows in a particular case.

The computation time for finding solutions varies greatly with the setting (algorithms run on a UNIX machine with a $2.70$ GHz i7-3820QM CPU

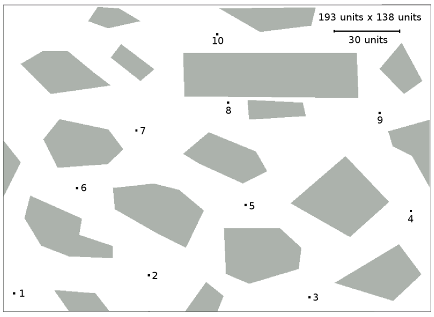
[1]We implemented it in `C++`, publicly available at `https://sourceforge.net/projects/optimalexplorationpath/`.



**(a)** *indoor*



**(b)** *openspace*



**(c)** *obstacles*

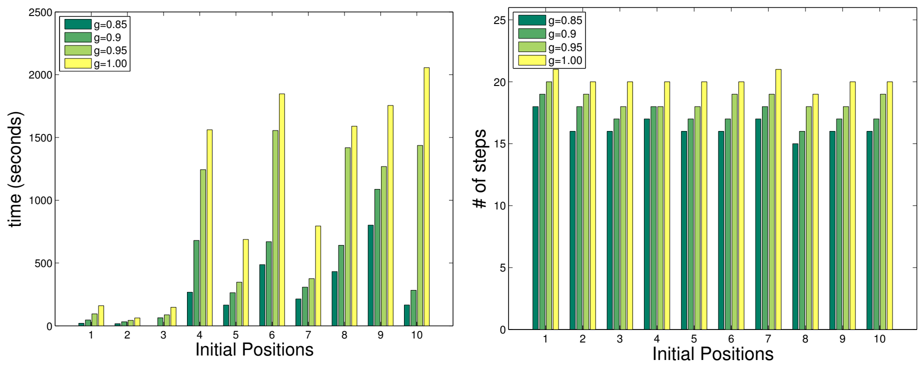
**Figure 3.9:** *The three environments (points represent different initial positions for the robot).*

and 16 GB RAM). For example, finding the optimal exploration path requires an average time of $0.58$ seconds for $e = 4$, $r = 30$, and $\mathfrak{g} = 0.85$, while $4$ out of $10$ runs do not terminate within $10$ hours and the remaining $6$ runs terminate in $1831.20$ seconds on average for $e = 1$, $r = 20$, and $\mathfrak{g} = 0.95$. In general, computation time increases when $e$ decreases, $r$ decreases, and $\mathfrak{g}$ increases. Figure 3.10a shows that the computation time required for finding the optimal exploration path highly depends on the initial position in the environment (results are similar for other settings). In particular, positions that are basically on the top corridor of the indoor environment (e.g., positions 6, 8, or 10) require more time to find the optimal exploration path, as the search has to follow two main branches, corresponding to going first right and then left, or *vice versa*. Differently, searching from position 1 basically amounts to perform a "focused" depth-first search, which is very fast.

Note that Figure 3.10a reports also the time for $\mathfrak{g} = 1.00$. For the same setting of Figure 3.10a, Figure 3.10b shows that the number of steps of the optimal solution grows almost linearly with $\mathfrak{g}$, up to $\mathfrak{g} = 1.00$. Moreover,

| | | # OF STEPS | | | DISTANCE | | |
|---|---|---|---|---|---|---|---|
| | $e$ | $r = 20$ | $r = 25$ | $r = 30$ | $r = 20$ | $r = 25$ | $r = 30$ |
| $\mathfrak{g} =$ **85%** | 1 | 22.3 (0.7)$^{*(1)}$ | 16.3 (0.8) | 12.9 (0.6) | 470.6 (26.6) | 420.2 (32.1) | 387.2 (28.7) |
| | 2 | 21.8 (0.9) | 15.3 (0.9) | 12.5 (0.7) | 416.5 (27.1) | 386.3 (21.0) | 368.8 (18.2) |
| | 4 | 20.2 (1.1) | 13.2 (0.9) | 11.3 (0.8) | 342.1 (15.7) | 338.7 (29.7) | 321.2 (19.3) |
| $\mathfrak{g} =$ **90%** | 1 | 23.7 (0.8)$^{*(3)}$ | 17.4 (0.8) | 13.9 (0.6) | 500.3 (26.9)$^{*(2)}$ | 458.1 (31.7) | 433.9 (27.8) |
| | 2 | 23.1 (0.6) | 16.4 (0.8) | 13.3 (0.7) | 449.0 (26.4)$^{*(1)}$ | 422.7 (26.5) | 407.0 (28.7) |
| | 4 | 21.2 (1.1) | 14.1 (0.9) | 11.8 (0.6) | 366.4 (14.7) | 367.9 (29.3) | 348.2 (25.1) |
| $\mathfrak{g} =$ **95%** | 1 | 25.3 (0.5)$^{*(4)}$ | 18.6 (0.7) | 14.8 (0.4) | 538.1 (13.0)$^{*(4)}$ | 504.6 (35.3) | 473.3 (42.4) |
| | 2 | 24.5 (0.8) | 17.4 (0.7) | 14.4 (0.7) | 473.7 (15.7)$^{*(3)}$ | 461.7 (29.5) | 448.7 (38.2) |
| | 4 | 22.4 (1.3) | 15.0 (0.7) | 12.8 (0.6) | 391.6 (21.1) | 402.9 (27.6) | 387.3 (27.1) |

**Table 3.3:** *Results (average and standard deviation) for the indoor environment ($^{*(\#)}$: # of timeouts).*



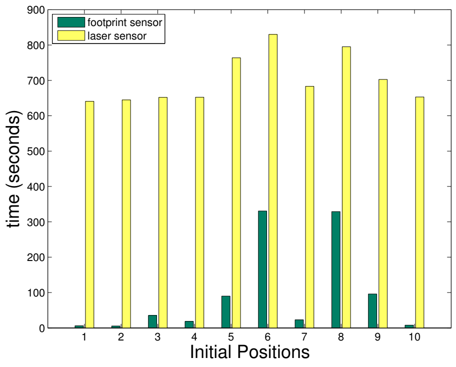
**(a)** *Computation time*

**(b)** *Optimal number of steps*

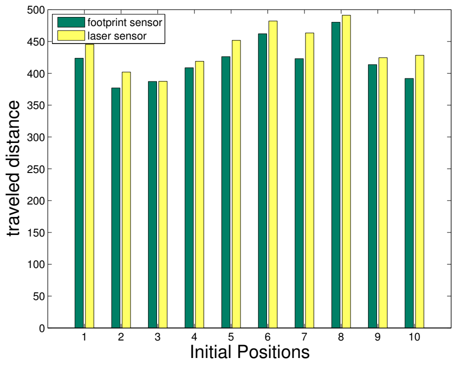
**Figure 3.10:** *Results of experiments for different initial positions and $\mathfrak{g}$ (indoor, cost function is $\mathfrak{c}_{\#}$, $e = 1$, $r = 25$).*

the number of steps required to explore the indoor environment is about the same independently of initial positions (as evidenced by the small standard deviations of Table 3.3). Analogous considerations hold for the traveled distance.

In all the above experiments, we have used the footprint sensor. As expected, using the more realistic laser range scanner sensor (for which we precomputed the set of perceived cells $\bar{p}_D^{\text{new}}(r, c)$ from every cell $c$ in $G_f^e$) increases the computation time (see Figure 3.11a for an example). Indeed, according to an ANOVA analysis with a $p$-value $< 0.05$, the difference is statistically significant with $p$-value$=1.13 \cdot 10^{-10}$. However, rather surprisingly, the quality obtained with the footprint sensor is very similar to that obtained with the more realistic sensor (Figure 3.11b, where the difference of traveled distances is not statistically significant, with $p$-value$= 0.1883$). These results provide an *a posteriori* justification of the use of the footprint



**(a)** *Computation time*



**(b)** *Traveled distance*

**Figure 3.11:** *Results of experiments for different initial positions and sensor models (indoor, cost function is $\mathfrak{c}_{\mathscr{D}}$, $e = 1$, $r = 25$, $\mathfrak{g} = 0.85$).*

sensor in generating data in Table 3.3.

The experiments of Table 3.3 have been run with frontier cells clustering. Without this, the cost of computing a solution explodes: our algorithm does not find any solution within the timeout, even for simple settings. For example, for $e = 4$, $r = 30$, and $\mathfrak{g} = 0.85$, the algorithm with clustering finds the solution in an average of $0.58$ seconds, generating about $141$ nodes, while without clustering it meets the timeout for every initial position, after having generated about $500{,}000$ nodes.

Using as heuristics $h_{\#}(s)$ and $h_{\mathscr{D}}(s)$ (i.e., the solutions pre-calculated from relaxed problems) consistently reduces the computation time compared to using $\bar{h}_{\#}(s)$ and $\bar{h}_{\mathscr{D}}(s)$ (e.g., in the indoor environment, considering the setting $\mathfrak{c}_{\mathscr{D}}$, $e = 1$, $\mathfrak{g} = 0.95$, and $r = 25$, the computation time with the former heuristic is five times smaller with respect to the latter heuristic), and allows to obtain a solution for all the initial positions. Indeed, as expected from the discussion of previous section, the heuristics obtained from a relaxed problem are better estimates of the cost of the solution than the other two simpler heuristics, even if the difference about computing time when using heuristics $h_{\#}(s)$ and $h_{\mathscr{D}}(s)$ with respect to $\bar{h}_{\#}(s)$ and $\bar{h}_{\mathscr{D}}(s)$ is not statistically significant ($p$-value$=$ $0.0785$), in the example above mentioned. Note that the costs of the solutions obtained using the two heuristics are the same, experimentally showing that the heuristics obtained by the solutions calculated from relaxed problems are in fact admissible.

Tables 3.4 and 3.5 report experimental results for openspace and obstacles environments, respectively. Also in these experiments we have considered the footprint sensor and the clustering of frontier cells. Moreover,

| | | # OF STEPS | | | DISTANCE | | |
|---|---|---|---|---|---|---|---|
| | $e$ | $r = 20$ | $r = 25$ | $r = 30$ | $r = 20$ | $r = 25$ | $r = 30$ |
| $\mathfrak{g} = 85\%$ | 1 | 65.6 (0.7)$^{*(1)+(1)}$ | 43.3 (0.5) | 29.9 (0.9) | 1425.7 (32.0) | 1210.4 (100.9) | 965.6 (63.3) |
| | 2 | 63.8 (0.8)$^{+(1)}$ | 40.1 (3.1) | 29.5 (0.7) | 1287.8 (28.3) | 1070.2 (34.9) | 913.9 (18.9) |
| | 4 | 60.5 (1.4) | 33.5 (4.6) | 25.8 (2.5) | 1073.0 (23.4) | 919.7 (71.5) | 792.6 (33.8) |
| $\mathfrak{g} = 90\%$ | 1 | 69.6 (0.8)$^{*(1)+(2)}$ | 45.9 (0.7) | 31.8 (0.6) | 1532.4 (51.7)$^{*(1)}$ | 1280.6 (78.8) | 1040.4 (34.4) |
| | 2 | 68.2 (0.9)$^{+(2)}$ | 42.5 (2.9) | 31.3 (1.1) | 1386.5 (40.3) | 1145.5 (36.6) | 988.1 (46.3) |
| | 4 | 64.2 (1.1) | 35.5 (4.8) | 27.5 (3.0) | 1141.9 (29.6) | 993.0 (81.1) | 840.7 (30.9) |
| $\mathfrak{g} = 95\%$ | 1 | 74.1 (1.1)$^{*(1)+(2)}$ | 48.8 (1.0) | 33.8 (0.6) | 1686.9 (92.8)$^{*(1)}$ | 1416.8 (186.9) | 1116.9 (24.3) |
| | 2 | 71.5 (0.6)$^{*(1)+(2)}$ | 46.3 (4.2) | 33.2 (1.0) | 1471.7 (57.6) | 1221.8 (51.7) | 1061.2 (48.7) |
| | 4 | 68.6 (1.0)$^{*(1)+(2)}$ | 39.0 (5.8) | 29.2 (3.3) | 1216.8 (26.9) | 1093.1 (90.4) | 887.3 (35.2) |

**Table 3.4:** *Results (average and standard deviation) for the openspace environment ($^{*(\#)}$: # of runs terminated due to the timeout, $^{+(\#)}$: # of runs terminated because of empty frontier).*

| | | # OF STEPS | | | DISTANCE | | |
|---|---|---|---|---|---|---|---|
| | $e$ | $r = 20$ | $r = 25$ | $r = 30$ | $r = 20$ | $r = 25$ | $r = 30$ |
| $\mathfrak{g} = 85\%$ | 1 | 24.0 (0.7)$^{*(5)}$ | 15.7 (0.5) | 11.2 (0.4) | 501.8 (14.1) | 404.4 (14.9) | 340.8 (10.7) |
| | 2 | 23.4 (0.5) | 14.5 (0.5) | 11.0 (0.5) | 453.7 (10.3) | 372.6 (14.0) | 317.6 (16.5) |
| | 4 | 21.9 (0.6) | 11.9 (0.6) | 9.2 (0.4) | 391.2 (10.8) | 308.5 (22.5) | 270.2 (14.1) |
| $\mathfrak{g} = 90\%$ | 1 | 26.0 (0.7)$^{*(5)}$ | 16.8 (0.4)$^{*(1)}$ | 12.1 (0.6) | 554.1 (24.0) | 438.6 (13.6) | 363.3 (15.2) |
| | 2 | 25.3 (0.7) | 15.6 (0.5) | 11.9 (0.6) | 494.1 (11.0) | 402.6 (16.3) | 346.1 (20.9) |
| | 4 | 23.6 (0.7) | 12.9 (0.6) | 9.8 (0.6) | 417.3 (14.8) | 337.3 (26.2) | 290.1 (14.6) |
| $\mathfrak{g} = 95\%$ | 1 | 27.7 (0.5)$^{*(5)+(1)}$ | 18.0 (0.5)$^{*(2)}$ | 13.1 (0.7) | 607.3 (41.3)$^{*(3)}$ | 479.1 (13.2) | 399.6 (14.9) |
| | 2 | 27.0 (0.0)$^{*(2)}$ | 16.6 (0.5) | 12.7 (0.5) | 538.1 (11.1) | 436.1 (16.8) | 377.6 (19.6) |
| | 4 | 24.9 (0.6) | 13.6 (0.5) | 10.5 (0.5) | 448.5 (13.3) | 362.3 (21.8) | 320.8 (12.5) |

**Table 3.5:** *Results (average and standard deviation) for the obstacles environment ($^{*(\#)}$: # of runs terminated due to the timeout, $^{+(\#)}$: # of runs terminated because of empty frontier).*

we have eliminated clusters smaller than $k = 36$ cells and $k = 12$ cells of size $e = 1$ for the openspace and the obstacles environments, respectively ($k$ is scaled accordingly using $e = 2$ and $e = 4$). Preliminary experiments showed that eliminating small clusters provides a consistent reduction of computation time, without affecting too much the solution quality. These values have been empirically set considering that clusters are larger in the openspace environment. Eliminating small clusters, some runs terminate without a solution because of empty frontier, namely because there are no available destination positions. These runs are reported in the tables together with runs that terminate due to the timeout. As expected, runs terminating because of empty frontier are more frequent in the openspace environment, where we used a larger $k$, and for larger values of $\mathfrak{g}$, for which a larger amount of area has to be discovered. All the previous considerations relative to the indoor environment, including those about speedup techniques, hold also for the openspace and obstacles environments.

Now, let us show the comparison of our method against a state-of-the-art technique that can be easily applied to our settings, up to some adaptations. We consider the work presented by Fazli et al. [42], which, as already discussed in Section 2.1.1, deals with the similar problem of (offline) optimal repeated coverage. That approach is based on these steps: (a) the environment is segmented using trapezoidation, (b) guards (scan points) are placed in the derived subpolygons, and (c) a TSP is solved. Since we consider grid-based maps, we add the constraint that the robot can only move at centers of cells and that perception requires centers of cells to be within sensor range $r$. The initial pose of the robot is considered as a guard from which the robot starts its path. In order to consider a percentage of area to map $\mathfrak{g} < 100\%$, after having found guards to be placed in the environment to fully cover it, the algorithm iteratively discards the largest number of placed guards furthest from the initial robot position keeping the covered area greater than $\mathfrak{g}$. In order to consider a path instead of a tour, once the TSP solution connecting all the remaining guards is computed, we iteratively remove an arc from it, compute the new cost, and finally calculate the mean of all these modified TSP solutions; formally, $\mathscr{D} = \sum_{i \in V} \mathfrak{c}_i - \frac{\sum_{i \in V} \mathfrak{c}_i}{|V|}$, where $\mathscr{D}$ is the length of the path, $V$ is the set of arcs in the TSP tour, and $\mathfrak{c}_i$ is the cost of an arc $i$. We iterate the algorithm only once, since we do not need repeated coverage.

Results of this modified algorithm in the indoor, openspace, and obstacles environment are shown in Tables 3.6, 3.7, and 3.8, respectively. As it

| | | # OF STEPS | | | DISTANCE | | |
|---|---|---|---|---|---|---|---|
| | $e$ | $r = 20$ | $r = 25$ | $r = 30$ | $r = 20$ | $r = 25$ | $r = 30$ |
| $\mathfrak{g} =$ **85%** | 1 | 113.6 (5.3) | 87.8 (5.5) | 59.1 (1.5) | 1226.8 (15.6) | 1099.4 (24.4) | 1012.5 (31.1) |
| $\mathfrak{g} =$ **90%** | 1 | 120.1 (2.7) | 92.7 (2.9) | 65.4 (0.9) | 1296.9 (16.7) | 1162.9 (13.9) | 1123.8 (10.8) |
| $\mathfrak{g} =$ **95%** | 1 | 127.4 (0.7) | 99.8 (0.9) | 67.5 (1.5) | 1397.7 (21.1) | 1265.3 (14.5) | 1180.1 (28.3) |

**Table 3.6:** *Results (average and standard deviation) for the indoor environment using an adapted version of Fazli et al. [42].*

| | | # OF STEPS | | | DISTANCE | | |
|---|---|---|---|---|---|---|---|
| | $e$ | $r = 20$ | $r = 25$ | $r = 30$ | $r = 20$ | $r = 25$ | $r = 30$ |
| $\mathfrak{g} =$ **85%** | 1 | 181.5 (4.3) | 113.4 (2.8) | 90.0 (2.6) | 3118.5 (79.9) | 2505.0 (76.4) | 2192.8 (83.5) |
| $\mathfrak{g} =$ **90%** | 1 | 193.7 (3.7) | 121.5 (2.5) | 96.8 (2.7) | 3323.9 (57.7) | 2678.3 (58.9) | 2353.1 (67.8) |
| $\mathfrak{g} =$ **95%** | 1 | 207.6 (3.6) | 130.7 (2.5) | 105.4 (2.7) | 3559.1 (64.6) | 2880.8 (60.9) | 2564.2 (68.1) |

**Table 3.7:** *Results (average and standard deviation) for the openspace environment using an adapted version of Fazli et al. [42].*

can be observed, the modification of the approach proposed by Fazli et al. [42] obtains much higher costs than our approach. This could be explained by the fact that, being the environment segmentation operated by Fazli et al.

| | | # OF STEPS | | | DISTANCE | | |
|---|---|---|---|---|---|---|---|
| | $e$ | $r = 20$ | $r = 25$ | $r = 30$ | $r = 20$ | $r = 25$ | $r = 30$ |
| $\mathfrak{g} = \mathbf{85\%}$ | 1 | 218.1 (19.8) | 166.3 (17.9) | 138.4 (16.6) | 1493.1 (80.1) | 1278.8 (95.8) | 1157.3 (116.8) |
| $\mathfrak{g} = \mathbf{90\%}$ | 1 | 237.5 (16.8) | 182.4 (19.3) | 153.4 (17.1) | 1617.3 (77.0) | 1393.6 (96.4) | 1263.3 (101.8) |
| $\mathfrak{g} = \mathbf{95\%}$ | 1 | 264.0 (12.7) | 205.9 (16.6) | 172.3 (13.9) | 1783.4 (57.3) | 1554.9 (99.6) | 1424.9 (92.6) |

**Table 3.8:** *Results (average and standard deviation) for the obstacles environment using an adapted version of Fazli et al. [42].*

[42] not optimal, even if optimal TSP tours are computed over it, the final solutions are significantly worse than those obtained by our algorithm.

Our proposed method can compute the optimal offline exploration path for problem $D$ (which can be "travelable" by an online exploring robot thanks to the reachability constraint) in environments for a single robot with limited and time-discrete visibility, as shown by extensive experimental activities performed in simulation in environments of realistic size. The approach allows a trade-off between solution quality and computation time by tuning $e$ and $\mathfrak{g}$ and with some speedup techniques. This contribution is a step towards the study and the resolution of problems typically studied by computational geometry community, but with some more realistic assumptions. We show in Chapter 7 how the proposed approach can be used to improve the analysis and the evaluation of exploration strategies. Further, we show in Chapter 10 how the search-based approach used here can inspire a method to search in an abstract state space.

# 4 Online exploration

As already said in Chapter 2, the assessment of practical exploration strategies performed in the field of robotics is mainly empirical or based on idealized assumptions. Very few works have considered more realistic settings for providing bounds on the quality of solutions produced by exploration strategies, prominently [72] and [127], described in Section 2.1.2. In their approach, a single robot should visit (or perceive) all the vertices of an undirected graph, whose edges have unitary cost.

In this chapter, we present a contribution that significantly complement the analysis of Tovey and Koenig [127] by considering other exploration strategies, which combine information gain with distance; a termination criterion that prescribes the perception of a fraction $\mathfrak{g} \in (0, 1]$ of the vertices; and by embedding a sensor range $r$ in the worst-case bounds. Our main contributions include upper and lower bounds on the worst-case number of edge traversals to explore a finite undirected connected graph, which complement those reported in [127] and [72] to our setting. Moreover, we present an average-case analysis on the performance of some exploration strategies in a class of graphs that model indoor environments, which, to the best of our knowledge, has never appeared in the literature. The long-term goal of our analysis is to better understand the performance of exploration strategies and to explain some experimental findings reported in the literature.

## 4.1 Problem formulation

The environment is represented by a graph $G = (V, E)$, where the vertices $V$ correspond to the locations where an autonomous mobile robot can move and the edges $E$ represent the direct connections between these locations. The graph is assumed to be undirected, connected, and finite. For sake of simplicity, we assume that edges have unitary costs (as in the work of Tovey and Koenig [127]). So, with respect to the dimensions presented in Section 1.3, we have that the knowledge representation is discrete.

The robot operates according to the general behavior model presented in Section 1.1. In particular, the robot starts exploring in a vertex $v_0 \in V$ at a time step 0 having no *a priori* knowledge about the graph $G$. Thus, we have an online scenario, differently from Chapter 3. The robot is equipped with a sensor with a finite range $r \in \mathbb{R}_{>0}$ that perceives all vertices within the range $r$. More formally, a robot at $v_i$ at time step $i$, perceives the vertices $P_i = \{v' \in V \mid d(v_i, v') \leq r\}$ and updates its knowledge about already perceived (known) vertices as $V_i = V_{i-1} \cup P_i$, where $V_{i-1}$ is the set of known vertices at the previous step $i-1$ and $d(v_i, v')$ is a function that computes the geodesic distance between the two vertices $v_i$ and $v'$ in $G$. The latter corresponds to Step (a) and Step (b) (see Section 1.1). Note that when $i = 0$, then $V_{i-1} = V_{-1} = \emptyset$, namely the graph is initially unknown. The perception model allows the robot to acquire knowledge about the incident edges of vertices $v' \in P_i$ and to recognize whether there is an edge between two known vertices $v', v'' \in V_i$. At each time step $i$, the set of partially perceived vertices on the frontier, namely the candidate vertices, is $F_i = \{v' \in V \mid v' \notin V_i \wedge (\exists (v'', v') \in E \mid v'' \in V_i)\}$, (similarly to the model of [127]). Note that if $r = \epsilon$ (where $\epsilon$ is a small constant that tends to 0 and allows perceiving just the vertex where the robot finds itself and the incident edges), then vertices are perceived only when physically visited by the robot, as in the fixed graph scenario typically studied in the literature. Although the perception of *all* vertices at distance up to $r$ from the robot current vertex could be unrealistic in some scenarios (due to the presence of obstacles), this footprint model leads to interesting theoretical results that are in accordance with many results obtained by experimenting with real exploring robots. We assume that the perception model of the robot is discrete: the robot perceives the surrounding environment and updates $V_i$ to $V_{i+1}$ only when it is in the next position $v_{i+1}$ and not continuously while moving (time-discrete perception is often assumed by online exploration algorithms [8], as also shown in Section 2.3). This implies that the decision model is event-based, namely a new decision is performed when the next candidate vertex is reached.

At each time step $i$, the robot chooses to move to one of the candidate vertices $F_i$, according to an exploration strategy $\mathscr{S}$ (Step (c)). Although it would be possible to define a motion model for which the robot can move to any vertices $v \in V_i \cup F_i$, we impose to move on frontier vertices, as usually done with real exploring robots. Since we are interested in the theoretical analysis of the online exploration strategies, we assume that the perceptions (Step (a)) and the movements (Step (d)) of the robot are error-free (i.e., deterministic). As a consequence, the robot perfectly knows its position in the environment.

The exploration process continues until a percentage $\mathfrak{g} \in (0, 1]$ of the vertices of $G$ are perceived by the robot, namely until $\frac{|V_i|}{|V|} \geq \mathfrak{g}$. Note that the exploration terminates at some finite time step $k$ because the robot chooses vertices that provide some new information about the graph (i.e., candidate vertices $F_i$ have unexplored adjacent vertices and so $|V_{i+1}| > |V_i|$) and the considered graphs are finite. So, in the end, the robot follows a sequence of vertices $\mathcal{C} = \langle v_o, v_1, \cdots, v_k \rangle$, called *exploration path*, where $v_{i+1} \in F_i$, with $0 \leq i < k$ ($v_0$ is the starting vertex). Note that, in general, the robot could revisit some already visited vertices. The goodness of the exploration path is measured as number of edge traversals, the optimality criterion.

We consider exploration strategies that evaluate a candidate vertex $v \in F_i$ from the current position $v_i$ adopting the following criteria:

- $d_i(v_i, v)$ is the geodesic distance between $v_i$ and $v$ in $G'_i = (V_i \cup \{v\}, E'_i)$, which is the graph $G_i$ induced by $V_i$ on $G$, augmented with $v$ and with the edges (in $E$) between $v$ and vertices in $V_i$,
- $g(v, V_i)$ is the expected information gain at $v$, and is equal to the number of vertices the robot perceives in $v$ minus those already known. More formally, at step $i$, given a frontier vertex $v$, $g(v, V_i) = |P(v) \setminus V_i|$. Being $v \notin V_i$, the function $g()$ could be estimated from datasets of environments, but here we assume it as granted. In the particular case of $r = \epsilon$ ($\epsilon \to 0$) we define $g(v, V_i)$ as the number of edges incident to $v$ that are connected to vertices not in the current map $V_i$ at step $i$.

We consider three exploration strategies:

- $\mathscr{S}_d$, which selects locations by simply minimizing the distance $d()$ (as for example in the analysis of Tovey and Koenig [127]),
- $\mathscr{S}_g$, which chooses candidate locations maximizing the information gain $g()$ (as, e.g., in the work of Amigoni [2]),
- $\mathscr{S}_{dg}$, based on $\mathscr{S}_d$ but breaking ties favoring vertices with larger information gain $g()$ (thus providing a more informed version of $\mathscr{S}_d$ compared to Greedy Mapping presented by Tovey and Koenig [127]).

In all the three cases, further ties are broken randomly with uniform probability.

The problem we address in this chapter is the following. Given a sensor range $r \in \mathbb{R}_{>0}$, a percentage $\mathfrak{g} \in (0, 1]$ of the environment to map, and an exploration strategy $\mathscr{S} \in \{\mathscr{S}_d, \mathscr{S}_g, \mathscr{S}_{dg}\}$, can we determine some performance bounds on the exploration path $\mathcal{C}$ in terms of number of edges traversed by the robot in any undirected, connected, and finite graph $G$?

## 4.2 Worst-case analysis

Here we provide a comparison of the three exploration strategies by presenting some bounds on their worst-case performance, measured as traveled distance (number of edge traversals).

Let us first introduce the number of vertices to explore in a finite undirected connected graph $G = (V, E)$, according to a goal percentage $\mathfrak{g} \in (0, 1]$, which trivially is $n = \lceil |V| \cdot \mathfrak{g} \rceil$. If $\mathfrak{g} = 1$, then $n = |V|$, that is, the robot has to explore all the vertices of the graph, as usually assumed in graph exploration literature.

To study the upper bounds on the traveled distance of the exploration strategies, let us first derive an upper bound on the number of frontiers selected as destination locations while exploring a graph according to the goal percentage $\mathfrak{g}$ and the sensor range $r$, independently of the exploration strategy. Recall that the number of selected frontiers is not the number of frontiers that could be available to the robot at a time step $i$, but it is $k$, the cardinality of the exploration path $\mathcal{C}$ minus 1.

**Proposition 4.1.** *Given a goal percentage $\mathfrak{g} \in (0, 1]$ and a robot sensor range $r \in \mathbb{R}_{\geq 1}$, the maximum number of selected frontiers in the exploration sequence $\mathcal{C}$ is $\bar{k} = 2\frac{(\lceil |V| \cdot \mathfrak{g} \rceil - 1)}{\lfloor r \rfloor + 1} - 1$, on any finite undirected connected graph $G = (V, E)$, where the weight of each edge is* 1.

*Proof.* Recall that the robot has to explore $n = \lceil |V| \cdot \mathfrak{g} \rceil$ vertices. Given $|V|$, $\mathfrak{g}$, and $r$, we build the worst-case graph, which is a star like that of Figure 4.1, as follows. The robot can discover vertices at distance less than or equal to $r$ from $v_i$ (in particular $v_0$), and so frontier vertices are at least at distance $\lfloor r \rfloor + 1$ from a vertex $v_i$. We try to maximize the number of selected frontiers $k$, by placing them at the extremes of lines of vertices of length $\lfloor \frac{\lfloor r \rfloor + 1}{2} \rfloor$ attaching to the line between $v_0$ and $v_1$ in the middle. It is easy to check that, the case of odd $\lfloor r \rfloor$ is worse than the even case. Hence we calculate $k$, given $n = \lceil |V| \cdot \mathfrak{g} \rceil$ and $r$, as follows: $n = 1 + \frac{\lfloor r \rfloor + 1}{2} k$ and,

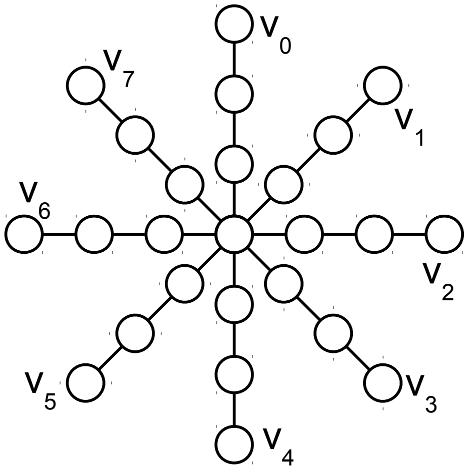


**Figure 4.1:** *Star graph with* $\mathfrak{g} = 1$, $|V| = 25$, *and* $r = 5$.

after some math $k = 2\frac{(n-1)}{\lfloor r\rfloor+1}$, and, as $v_0$ is not counted in the number of selected frontiers, $\bar{k} = 2\frac{(n-1)}{\lfloor r\rfloor+1} - 1 = 2\frac{(\lceil|V|\cdot\mathfrak{g}\rceil-1)}{\lfloor r\rfloor+1} - 1$. □

Intuitively, Proposition 4.1 derives from the idea that the robot selects a frontier vertex $v$ and, once there, perceives only $v$. Since each frontier vertex should be at least at distance $r + 1$ from other frontier vertices, the star graph of Figure 4.1 is a worst-case graph. Note that it is easy to see that the upper bound in Proposition 4.1 is tight, as we are considering the worst-case graph and the possible perceptions of the robot for the number of selected frontiers of the exploration path. Note also that if $\mathfrak{g} = 1$ and $r = \epsilon$ ($\epsilon \to 0$), then we have the trivial bound on the number of selected frontiers $\bar{k} = |V| - 1$.

Now, let us show bounds on the number of edge traversals during exploration. The following result allows us to restrict our analysis to smaller range $r$, as, if $r$ is greater than a value that depends on the size of the graph, the worst-case upper bound on the number of edge traversals is linear for any exploration strategy.

**Proposition 4.2.** *Given a goal percentage* $\mathfrak{g} \in (0, 1]$ *and* $r \geq \lfloor\frac{\lceil|V|\cdot\mathfrak{g}\rceil-1}{2}\rfloor$, *for any exploration strategy* $\mathscr{S}$, *the upper bound on the number of edge traversals is UB* $= \frac{3}{2}\lceil|V|\cdot\mathfrak{g}\rceil - 2$, *on any finite undirected connected graph* $G = (V, E)$, *where the weight of each edge is equal to* $1$.

*Proof.* Let us start considering $r = \lfloor\frac{|V|-1}{2}\rfloor$ (for sake of simplicity, in the proof, we consider $\mathfrak{g} = 1$; to include $\mathfrak{g}$, $|V|$ needs to be substituted with

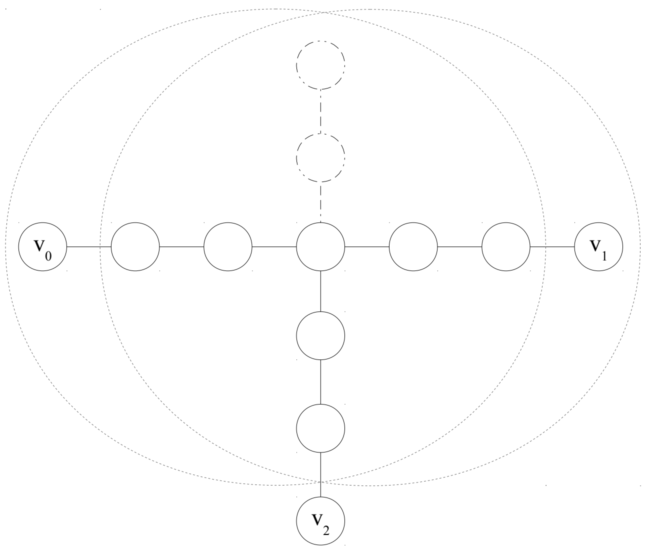


**Figure 4.2:** *For* $|V| = 12$ *and* $r = \frac{|V|}{2} - 1 = 5$ *there are at most two frontier vertices* $v_1$ *and* $v_2$.

$\lceil |V| \cdot \mathfrak{g} \rceil$). To prove the proposition, we incrementally build the worst-case graph, considering separately the case of even and odd values of $|V|$.

If the value of $|V|$ is even, $r = \frac{|V|}{2} - 1$. Thus, starting from an arbitrary $v_0$, the maximum number of frontier selections is 2 for any $|V| \geq 6$ because of Proposition 4.1 (as shown in Figure 4.2). Basically, we have two steps in the exploration path: from $v_0$ to a vertex $v_1$ and from $v_1$ to a vertex $v_2$. The distance between $v_0$ and $v_1$ is always of length $r + 1$. Thus, the first part of our worst-case graph is composed by $V_0$, which contains at least a line of $\frac{|V|}{2}$ vertices, plus $v_1$. If the remaining $\frac{|V|}{2} - 1$ vertices are attached to $v_0$ or $v_1$, there are no more frontier vertices, because they are within the range $r$ of a perception performed from $v_0$ or $v_1$. If they are attached to a vertex on the line between $v_0$ and $v_1$, forming a new line of vertices, there could be a new frontier vertex $v_2$, which is not within the range $r$ of a perception performed at $v_0$ and $v_1$. Let us show where the line of vertices containing $v_2$ should be attached to have the worst case. The worst case happens when that line containing $v_2$ is attached at distance 1 from $v_0$, on the line that links $v_0$ and $v_1$. It is easy to check that this shape maximizes the distance $d(v_1, v_2)$, since $d(v_0, v_1)$ is necessarily fixed to $r + 1$. Hence,

$$UB_{even} = d(v_0, v_1) + d(v_1, v_2) = (r + 1) + 2r = \frac{3}{2}|V| - 2. \qquad (4.1)$$

If $|V|$ is odd, there are still at most 2 frontier selections. Following the same reasoning, the number of vertices between $v_0$ and $v_1$ is $r = \frac{|V|-1}{2}$. Thus, the number of vertices we can use to compose our worst-case graph is $|V| - (r+2) = |V| - \frac{|V|-1}{2} - 2 = \frac{|V|-3}{2}$. If they are attached to $v_0$ or $v_1$, again, there are no more frontier vertices. There are no more frontier vertices even if they are attached at distance 1 from $v_0$ or $v_1$. In case the value of $|V|$ is odd, the worst case happens when the new line, which contains $v_2$, is attached at distance 2 from $v_0$, on the line that links $v_0$ and $v_1$. Hence,

$$UB_{odd} = d(v_0, v_1) + d(v_1, v_2) = (r+1) + 2(r-1) = \frac{3}{2}(|V|-1) - 1. \quad (4.2)$$

Note that we have the following relationship between the case of $|V|$ even and odd $UB_{odd} \leq UB_{even} = UB$. Also note that, considering a greater $r$, the upper bound found trivially holds, as, each increment of $r$ implies an increment of $d(v_0, v_1)$ but an equal decrement of $d(v_1, v_2)$. After a certain radius, the number of frontiers lowers to 1 and the trivial upper bound bound becomes $|V| - 1$. If the sensor range $r$ is greater than $|V| - 1$ the number of frontier selections is 0 as the number of edge traversals. $\square$

Now, let us show how the upper bound on the number of edge traversals changes according to $\mathfrak{g} \in (0, 1]$ and $r$ (with $r \in \mathbb{R}_{\geq 1}$ and $r < \lceil \frac{\lceil |V| \cdot \mathfrak{g} \rceil}{2} \rceil - 1$, for Proposition 4.2), differently from $UB_{TK} = |V| + 2|V| \ln(|V|)$, the worst-case bound on the performance of $\mathscr{S}_d$ which has been provided by [127] and that is independent of $\mathfrak{g}$ and $r$.

**Proposition 4.3.** *Given a goal percentage $\mathfrak{g} \in (0, 1]$ and a robot sensor range $r \in \mathbb{R}_{\geq 1}$, a worst-case upper bound on traveled distance for $\mathscr{S}_d$ is*

$$UB_{\mathscr{S}_d} = 2\lceil |V| \cdot \mathfrak{g} \rceil \left( \ln \frac{2\lceil |V| \cdot \mathfrak{g} \rceil + \lfloor r \rfloor (\lfloor r \rfloor - 2) - 7}{(\lfloor r \rfloor + 1)^2} - \frac{\lceil |V| \cdot \mathfrak{g} \rceil + \lfloor r \rfloor (\lfloor r \rfloor - 2) - 5}{(\lceil |V| \cdot \mathfrak{g} \rceil - 2)(\lfloor r \rfloor + 1)} + 2 \right)$$

*edge traversals, on any finite undirected connected graph $G = (V, E)$, where the weight of each edge is equal to* 1.

*Proof.* The proof follows that of [127]. Lemma 6.2 of [127] states the following.

> Define $S^t = \{v^i \in V | r^i \geq t\}$ for an orderly marking sequence $\{v^i, r^i, M^i\}$ on a given connected graph $G = (V, E)$. Then, it holds that $|S^t| \leq 2|V|/t$.

An orderly marking sequence is basically an exploration path, which includes the vertices the robot visits during the exploration. The symbols used in the above Lemma have the following correspondence with the symbols used in this chapter: $v^i = v_i$, $M^i = V_i$, and $r^i = d(v_{i-1}, v_i) - 1$, namely the radius of a circle centered on $v_{i-1}$, which is given by the distance between the current frontier $v_i$ and the preceding frontier $v_{i-1}$ minus 1 (because of the movement towards the frontier) in the exploration path. Note that, as $r^i$ represents the radius within which, from $v_{i-1}$, all the vertices are in $V_{i-1}$, intuitively, $S^t$ represents all the vertices $v_i$ that are at a distance of at least $t+1$ from the next vertex $v_{i+1}$. By construction, each pair of frontier vertices in the exploration path are at least at distance $r+1$.

Considering the worst case on the traveled distance, the exploration path has a number of selected frontiers equal to $\bar{k}$, according to Proposition 4.1. Recall that the total number of vertices to perceive is $n = \lceil |V| \cdot \mathfrak{g} \rceil$. Let us define $h$ as the number of different $t$-classes $S^t$, where $S^t$ is defined as above. $h$ is a positive integer with value at most $\bar{k}$.

To enumerate all the $t$-classes used in our exploration path, let us define a function $f()$ which orders them, starting from $f(1) =$ the smallest $t$-distance (which must be greater than or equal to $r$); until $f(h) =$ the biggest $t$-distance in the exploration path (which must be less than or equal to $n-2$, because the maximum travelable distance between two different vertices is $n-1$). Let us also define $f(h+1) = n-1$, $f(0) = 0$, $|S^{n-1}| = 0$.

We can find the traveled distance (number of edge traversals) with the following formula (which exploits $\bar{k}$ and the radius $r^i$ that should be summed to 1 to have the actual traveled distance):

$$\sum_{i=1}^{\bar{k}} 1 + r^i = \bar{k} + \sum_{i=1}^{\bar{k}} r^i =$$

(disregarding the order of the selected frontiers in the path)

$$= \bar{k} + \sum_{t=0}^{h} f(t) \cdot (|S^{f(t)}| - |S^{f(t+1)}|) =$$

(by applying some math and given the fact that $|S^{n-1}| = 0$)

$$= \bar{k} + \sum_{t=0}^{h-1} (f(t+1) - f(t)) \cdot |S^{f(t+1)}| \leq$$

(Lemma 6.2 of [127])

$$\leq \bar{k} + 2n \sum_{t=0}^{h-1} \frac{f(t+1) - f(t)}{f(t+1)}. \tag{4.3}$$

We have to find the set of values of $f()$ that maximize the above sum. Because of the Proposition 4.1, in most cases the $h$ counterimages of $f()$ do not cover all the codomain $\{\lfloor r \rfloor, ..., n-2\}$. It is easy to see that the worst case happens when all the missing counterimage values are between $r+h-1$ and $n-2$. Thus, when $f(1) = \lfloor r \rfloor$; $f(2) = \lfloor r \rfloor + 1$; ...; $f(h-1) = \lfloor r \rfloor + h - 2$; and $f(h) = n-2$. Hence Equation (4.3) becomes:

$$= \bar{k} + 2n \left( \sum_{t=0}^{h-2} (\frac{f(t+1) - f(t)}{f(t+1)}) + \frac{f(h) - f(h-1)}{f(h)}) \right) \leq$$

using the considerations on the values of $f()$ and doing some math

$$\leq \bar{k} + 2n \left( \sum_{t=\lfloor r \rfloor + 1}^{h + \lfloor r \rfloor - 1} \frac{1}{t} + 2 - \frac{f(h-1)}{f(h)} \right) \approx$$

limit approximation for the sum and explicitly reporting the first and last value of $f()$

$$\approx \bar{k} + 2n \left( \ln(\frac{h + \lfloor r \rfloor - 2}{\lfloor r \rfloor + 1}) - \frac{h + \lfloor r \rfloor - 2}{n-2} + 2 \right).$$

Now we have to find the value of $h$ that maximizes the formula. By analyzing the first and second derivative with respect to $h \in \{1, \cdots, \bar{k}\}$, we can find that the maximum value is for $h = \bar{k}$. Therefore, after some math, we have that the value is bounded by

$$2\lceil |V| \cdot \mathfrak{g} \rceil \left( \ln \frac{2\lceil |V| \cdot \mathfrak{g} \rceil + \lfloor r \rfloor(\lfloor r \rfloor - 2) - 7}{(\lfloor r \rfloor + 1)^2} - \frac{\lceil |V| \cdot \mathfrak{g} \rceil + \lfloor r \rfloor(\lfloor r \rfloor - 2) - 5}{(\lceil |V| \cdot \mathfrak{g} \rceil - 2)(\lfloor r \rfloor + 1)} + 2 \right) - \frac{2}{\lfloor r \rfloor + 1} - 1$$

that is slightly lower than

$$2\lceil |V| \cdot \mathfrak{g} \rceil \left( \ln \frac{2\lceil |V| \cdot \mathfrak{g} \rceil + \lfloor r \rfloor(\lfloor r \rfloor - 2) - 7}{(\lfloor r \rfloor + 1)^2} - \frac{\lceil |V| \cdot \mathfrak{g} \rceil + \lfloor r \rfloor(\lfloor r \rfloor - 2) - 5}{(\lceil |V| \cdot \mathfrak{g} \rceil - 2)(\lfloor r \rfloor + 1)} + 2 \right).$$

$\square$

Note that, if we consider the bound independent of $\mathfrak{g}$ and $r$ that in our setting corresponds to $\mathfrak{g} = 1$ and $r = \epsilon$ ($\epsilon \to 0$), we have a bound slightly different of that in [127]. However, by including in the limit approximation the first term $\frac{r}{r}$ and the last term $\frac{f(h)-f(h-1)}{f(h)}$ we considered apart from the sum $\sum_{t=0}^{h-2}(\frac{f(t+1)-f(t)}{f(t+1)})$, we exactly obtain the same bound of [127], as it would be $\bar{k} + 2\lceil |V| \cdot \mathfrak{g} \rceil \sum_{t=0}^{h-1}(\frac{f(t+1)-f(t)}{f(t+1)}) \leq \bar{k} + 2\lceil |V| \cdot \mathfrak{g} \rceil (\sum_{t=\lfloor r \rfloor+1}^{h+\lfloor r \rfloor} \frac{1}{t}) \approx \bar{k} + 2\lceil |V| \cdot \mathfrak{g} \rceil \ln(h + \lfloor r \rfloor/(\lfloor r \rfloor + 1)) \leq \bar{k} + 2n \ln(\bar{k} + \lfloor r \rfloor/(\lfloor r \rfloor + 1))$. With $\mathfrak{g} = 1$ and $r = \epsilon$ ($\epsilon \to 0$), $\bar{k} = |V| - 1$ and an upper bound on the traveled distance is $|V| + 2|V| \ln |V|$.

Now, let us show a lower bound for $\mathscr{S}_d$.

**Proposition 4.4.** *Given a goal percentage* $\mathfrak{g} \in (0, 1]$ *and a robot sensor range* $r \in \mathbb{R}_{\geq 1}$*, the worst-case lower bound of* $\mathscr{S}_d$ *on the traveled distance*

$$LB_{\mathscr{S}_d} = \Omega\left(\frac{\log\lceil |V| \cdot \mathfrak{g} \rceil - 2\log(\lfloor r \rfloor + 1)}{\log\log\lceil |V| \cdot \mathfrak{g} \rceil}\lceil |V| \cdot \mathfrak{g} \rceil\right)$$

*edge traversals.*

*Proof.* The proof is structured in two parts. First we show how the worst-case graph is constructed, depending on the sensor range $r$. This construction is based on the worst-case graph in [72]. Then, we show that, on that graph, the robot (re)traverses a certain part of the graph several times.

The worst-case graph consists of three main components (see, e.g., Figure 4.3). The first one, that we call *stem* is a line graph, whose number of vertices is $m^m$, where $m \geq 3$ is a parameter that allows to obtain the robot behavior described below. Let us call the vertices on the stem $v_0$, $v_1$, $v_2$, ..., $v_{m^m}$, where $v_0$ is the vertex where the robot starts the exploration. The second main component is a *loop of level* $i$, where $w + 1 \leq i \leq m$, and $w = \lceil \log_m(\lfloor r \rfloor + 1) \rceil$, which is a parameter that depends on the sensor range $r$ and defines the way the loops are attached to the stem, with which loops share just one vertex. The number of loops of level $i$ is $m^{m-i}$. Loops of different levels $i$ have different length. Specifically, the number of vertices of loops of level $i = w + 1$ is $3m^w$. When $i > w + 1$, the number of vertices in each loop of level $i$ is

$$\lceil (\frac{i-w}{2} + 1) \cdot m^w \rceil + \sum_{j=w+1}^{i-1} m^j. \tag{4.4}$$

The loops are attached to the vertices on the stem in the following way. We start to attach loops of level $i$ in an incremental way to vertices on the stems,

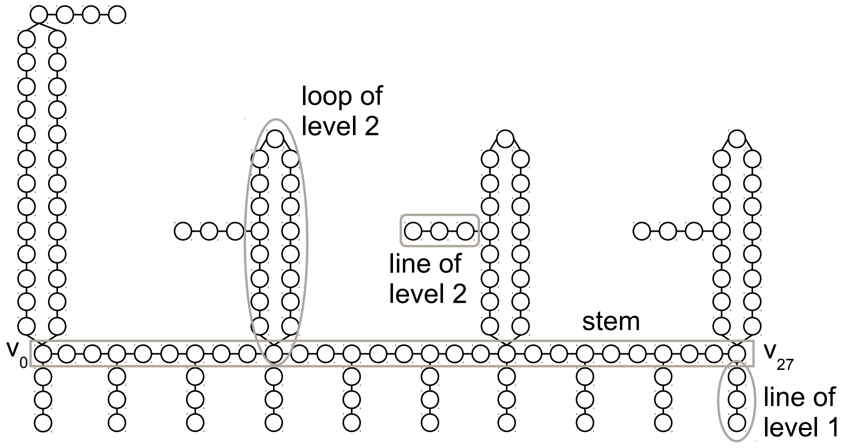


**Figure 4.3:** *Worst-case graph for $\mathscr{S}_d$ with $m = 3$ and $r = 2$.*

distanced by $m^i$: first all loops of level $i$ are attached to the stem starting from $v_{m^m}$, next all loops of level $i+1$ are attached to the stem starting from $v_0$, then loops of level $i+2$ starting from $v_{m^m}$, and so on, until $i = m$. The third main component of the worst-case graph we are building is a number of lines attached either to the stem or the loops. The number of vertices of those lines is $r + 1$. Let us call the lines attached to the stem lines of level $i = w$. They are attached to the stem similarly to loops, that is, starting from $v_0$ and distanced by $m^w$. Then, all the loops of level $i$ have a line attached to a vertex of the loop. When $i = w + 1$, the line is attached at distance $\lceil \frac{3}{2} m^w \rceil$ from the vertex on the stem. When $i > w + 1$, the lines are attached to a vertex in the loop in such a way that there are two paths of the same length from the vertex that the loop shares with the stem and the vertex that the loop shares with the line.

Therefore, combining all the elements of the worst-case graph, we obtain (after some math and simplifications) the number of vertices:

$$n \approx m^m + 1 + r\frac{m^m - m^w}{m^w(m-1)} + \frac{m^{m+1} - m^{w+2} + wm^{w+1} - wm^w}{(m-1)^2} + \\ + 2\frac{m^m(m-w)}{m-1} - 2\frac{m^m - m^w}{(m-1)^2} + 3m^{m-1} + (m^{m-w}+1)(r+1) \tag{4.5}$$

which asymptotically is $\theta(m^m)$ (with $n = \lceil |V| \cdot \mathfrak{g} \rceil$).

The robot, which starts at $v_0$ and employs $\mathscr{S}_d$ as the exploration strategy, explores the graph described above as follows. First, the robot explores all the vertices of the stem and of the loops of all levels, ending up at $v_{m^m}$. Let us show the reason why in this first traversal of the stem the robot does not explore the lines attached to the stem or to the loops. Considering that the robot starts exploring a loop of level $i = w + 1$ from its vertex shared with the stem, the distance between the frontier on the line attached to that loop and the robot position, when it explored the loop, is at least $3m^w + \lfloor \frac{3}{2} m^w \rfloor + \lfloor \frac{\lfloor r \rfloor}{2} \rfloor + 1 - (2\lfloor r \rfloor + 1)$. That distance is computed by considering the path that the robot should travel by backtracking to the loop and could be easily derived by knowing that the robot started from the vertex of the loop shared with the stem, the sensor range $r$ of the robot, and the length of the loop. Instead, the distance between the last frontier vertex of the loop and the closest frontier vertex on the stem is at most $3\lfloor r \rfloor + 2$. (Note that the other path between the last frontier vertex of the loop and the frontier vertex on the line of the same loop is greater than $3\lfloor r \rfloor + 2$ by construction.) To prove that the robot chooses the frontier vertex on the stem the following inequality should hold

$$3r + 2 < 3m^w + \lfloor \frac{3}{2} m^w \rfloor + \lfloor \frac{r}{2} \rfloor + 1 - (2r + 1). \tag{4.6}$$

After some math, we have $\frac{9}{2}\lfloor r \rfloor + 4 < \frac{9}{2} m^w$. As $m^{w-1} - 1 < r \leq m^w - 1$, that inequality becomes $\frac{9}{2} m^w - \frac{1}{2} < \frac{9}{2} m^w$, which is always true. For any loop of level $i > w + 1$, the inequality is still always true. Moreover, the distance from the last frontier vertex selected by the robot on the loop of level $i$ to the nearest frontier vertex on the line of level $w$ is always greater than or equal to the distance to the nearest frontier vertex on the stem. Thus, the frontier vertices on the lines are never chosen in the worst case at the first traversal.

Then, the robot traverses the stem back from $v_{m^m}$ to $v_0$ exploring the lines at level $w$. Next, from $v_0$ to $v_{m^m}$ the robot explores the lines at level $w + 1$. This way of traveling over the vertices of the stem goes on until the lines of the last level $m$ are explored. This behavior is caused by the fact that the distance from the current selected frontier vertex on a line of loop of level $i - 1$ to the nearest frontier vertex on another line of a loop of level $i - 1$ is less than or equal to the distance to the nearest frontier vertex on the lines loop of level $i$. The minimum difference between the two distances happens when two loops of level $i - 1$ and $i$ are attached to the same vertex of the stem. By construction, those distances are $m^i$ and

$$\lceil(\frac{i-w}{2}+1)\cdot m^w\rceil+\sum_{j=w+1}^{i-1} m^j+d+1, \tag{4.7}$$

where $d$ is the number of perceived vertices of the lines and can have values $\lfloor\frac{\lfloor r\rfloor}{2}\rfloor \leq d \leq \lfloor r\rfloor$, respectively. Thus, considering the maximum distance from the vertex on the stem shared by the loops at level $i-1$ and $i$ to the frontier vertex on the line of the loop of level $i-1$ ($d_1=\lfloor r\rfloor$) and considering the minimum distance to the line of loop of level $i$ ($d_2=\lfloor\frac{\lfloor r\rfloor}{2}\rfloor$) we have the following:

$$\begin{aligned} m^i+\lceil(\frac{i-w}{2}+1)\cdot m^w\rceil+\sum_{j=w+1}^{i-1} m^j+d_1+1 \leq \\ \leq \lceil(\frac{i+1-w}{2}+1)\cdot m^w\rceil+\sum_{j=w+1}^{i} m^j+d_2+1 \end{aligned} \tag{4.8}$$

(simplifying the equal terms)

$$d_1-d_2 \leq \lceil\frac{i+3-w}{2}\cdot m^w\rceil-\lceil\frac{i+2-w}{2}\cdot m^w\rceil \tag{4.9}$$

(substituting $d_1$ and $d_2$ with their bounds)

$$\lceil\frac{\lfloor r\rfloor}{2}\rceil \leq \lceil\frac{i+3-w}{2}\cdot m^w\rceil-\lceil\frac{i+2-w}{2}\cdot m^w\rceil \tag{4.10}$$

(remind that $r \leq m^w-1$)

$$\lceil\frac{m^w-1}{2}\rceil \leq \lceil\frac{i+3-w}{2}\cdot m^w\rceil-\lceil\frac{i+2-w}{2}\cdot m^w\rceil. \tag{4.11}$$

Notice that, if $m^w$ is even, the difference on the right-hand side is $\frac{m^w}{2}$, thus the inequality is always true. If $m^w$ is odd and $i+3-w$ is odd, the difference is equal to $\frac{m^w+1}{2}$, while, if $i+3-w$ is even, becomes $\frac{m^w-1}{2}$. In either case the inequality is always true. In the particular case of $i=w$, we arrive at the same result, as $\frac{m^w-1}{2} \leq \lceil\frac{3}{2}\cdot m^w\rceil-m^w$, which is always true. This proves that, after the first stem traversal, when re-traversing the stem, the robot explores all the lines of the loops at the same level $i$.

Given all the reasonings above, the number of edge traversals is:

$$\begin{aligned}LB_{\mathscr{S}_d} &\geq (m-w)m^m + \sum_{i=w+1}^{m} m^{m-i}\left(\lfloor r\rfloor + 2(i-w)m^w + 4\sum_{j=w}^{i-1} m^j\right) + \\ &\quad + 3m^{m-1} + (m^{m-w}+1)(\lfloor r\rfloor + 1) \\ &= (m-w)m^m + \lfloor r\rfloor \frac{m^m - m^w}{m^w(m-1)} + 2\frac{m^{m+1} - m^{w+2} + wm^{w+1} - wm^w}{(m-1)^2} + \\ &\quad + 4\frac{m^m(m-w)}{m-1} - 4\frac{m^m - m^w}{(m-1)^2} + 3m^{m-1} + (m^{m-w}+1)(\lfloor r\rfloor + 1).\end{aligned}$$

Note that $LB_{\mathscr{S}_d}$ is $\Omega((m-w)m^m)$. Hence, as $n$ is $\theta(m^m)$, $w \leq \log_m(\lfloor r\rfloor + 1) + 1$, and, because $m \geq \frac{\log m^m}{\log\log m^m}$ as shown in [72], we have that $LB_{\mathscr{S}_d}$ is

$$\Omega\left(\left(\frac{\log n}{\log\log n} - \log_m(\lfloor r\rfloor + 1)\right) n\right).$$

Changing the $\log$ base we have the following result

$$\begin{aligned}\log_m(\lfloor r\rfloor + 1) &= \frac{\log(\lfloor r\rfloor + 1)}{\log m} \\ &\leq \frac{2\log(\lfloor r\rfloor + 1)}{\log\log n}.\end{aligned}$$

Since it holds that

$$\begin{aligned}\log m^2 &\geq \log(m\log m) \\ 2\log m &\geq \log\log m^m \\ \log m &\geq \frac{1}{2}\log\log m^m.\end{aligned}$$

Putting all together we have as final result the lower bound

$$LB_{\mathscr{S}_d} = \Omega\left(\frac{\log\lceil |V|\cdot\mathfrak{g}\rceil - 2\log(\lfloor r\rfloor + 1)}{\log\log\lceil |V|\cdot\mathfrak{g}\rceil}\lceil |V|\cdot\mathfrak{g}\rceil\right).$$

□

The idea behind Proposition 4.4, similarly to [72], is that, in the worst-case, the robot travels back and forth the graph several times, resulting in a superlinear bound on the number of edge traversals.

Analyzing the worst case of the exploration strategy $\mathscr{S}_g$ that considers just the information gain, instead, we have the following upper bound:

**Proposition 4.5.** *Given a goal percentage $\mathfrak{g} \in (0,1]$ and a robot sensor range $r \in \mathbb{R}_{\geq 1}$, the worst-case upper bound on the traveled distance for $\mathscr{S}_g$ is*

$$UB_{\mathscr{S}_g} = \left(\lceil |V| \cdot \mathfrak{g} \rceil - \frac{\lceil |V| \cdot \mathfrak{g} \rceil - 1}{\lfloor r \rfloor + 1}\right)\left(2\frac{\lceil |V| \cdot \mathfrak{g} \rceil - 1}{\lfloor r \rfloor + 1} - 1\right)$$

*edge traversals, on any finite undirected connected graph $G = (V, E)$, where the weight of each edge is equal to* $1$.

*Proof.* The proof trivially derives from considering the following scenario: the robot at each time step $i$ could have to traverse all the vertices perceived up to $i$ (which possibly could be the worst case for any strategy). More formally, we have that:

$$\sum_{i=0}^{\bar{k}-1} |V_i| + 1 \leq$$

(to maximize the number of traversed vertices at each time step $i$, as we assume that the robot can perceive just one vertex at a time, we have an additional term that takes into account all the initial perceived vertices $|P_0| = |V_0| = \lceil |V| \cdot \mathfrak{g} \rceil - \bar{k}$)

$$\begin{aligned}
&\leq \sum_{i=1}^{\bar{k}} (\lceil |V| \cdot \mathfrak{g} \rceil - \bar{k} + i) = \\
&= (\lceil |V| \cdot \mathfrak{g} \rceil - \bar{k})\bar{k} + \sum_{i=1}^{\bar{k}} i = \\
&= (\lceil |V| \cdot \mathfrak{g} \rceil - \bar{k})\bar{k} + \frac{\bar{k}(\bar{k}-1)}{2} = \\
&= \left(\lceil |V| \cdot \mathfrak{g} \rceil - \frac{\lceil |V| \cdot \mathfrak{g} \rceil - 1}{\lfloor r \rfloor + 1}\right)\left(2\frac{\lceil |V| \cdot \mathfrak{g} \rceil - 1}{\lfloor r \rfloor + 1} - 1\right).
\end{aligned}$$

□

Assuming $\mathfrak{g} = 1$ and $r = \epsilon$ ($\epsilon \to 0$), the worst-case upper bound for $\mathscr{S}_g$ is $\frac{|V|(|V|-1)}{2}$, as we need to consider $\bar{k} = |V| - 1$, which coincides with the well-known upper bound for any exploration strategy in a fixed graph scenario $O(|V|^2)$.

Looking at the lower bound for $\mathscr{S}_g$, we have the following proposition.

**Proposition 4.6.** *Given a goal percentage $\mathfrak{g} \in (0, 1]$ and a robot sensor range $r \in \mathbb{R}_{\geq 1}$, the worst-case lower bound on the traveled distance for $\mathscr{S}_g$ is*

$$LB_{\mathscr{S}_g} = \frac{\lfloor r \rfloor + 1}{2} \left( \frac{\lceil |V| \cdot \mathfrak{g} \rceil - \lfloor r \rfloor}{\lfloor r \rfloor + 1} \right) \left( \frac{\lceil |V| \cdot \mathfrak{g} \rceil - \lfloor r \rfloor}{\lfloor r \rfloor + 1} - 1 \right)$$

*edge traversals, on any finite undirected connected graph $G = (V, E)$, where the weight of each edge is equal to* $1$.

*Proof.* Let us consider a line graph. The robot starts from the middle of the line. In case of even number of vertices we can choose arbitrary one of the two vertices in the middle. The proof develops from the idea that the robot can go back and forth to frontier vertices traveling over all of the already perceived vertices. Trivially the number of chosen frontiers along the exploration path is $k \geq \frac{\lceil |V| \cdot \mathfrak{g} \rceil - \lfloor r \rfloor}{\lfloor r \rfloor + 1} - 1$. At each time step the robot can perceive $r + 1$ vertices, thus the lower bound is

$$\sum_{i=1}^{k} (i(\lfloor r \rfloor + 1)) = (r+1)\frac{k(k+1)}{2} \geq$$

$$\geq \frac{\lfloor r \rfloor + 1}{2} \left( \frac{\lceil |V| \cdot \mathfrak{g} \rceil - \lfloor r \rfloor}{\lfloor r \rfloor + 1} \right) \left( \frac{\lceil |V| \cdot \mathfrak{g} \rceil - \lfloor r \rfloor}{\lfloor r \rfloor + 1} - 1 \right).$$

□

Assuming again $\mathfrak{g} = 1$ and $r = \epsilon$ ($\epsilon \to 0$), we have the worst-case lower bound for $\mathscr{S}_g$ is $\frac{|V|(|V|-1)}{2}$, which also in this case coincides with the well-known lower bound for any exploration in a fixed graph scenario $O(|V|^2)$.

Looking at $\mathscr{S}_{dg}$, the worst-case bounds are trivially the same as $\mathscr{S}_d$, as the following proposition shows.

**Proposition 4.7.** *The worst-case upper bound on the traveled distance for $\mathscr{S}_{dg}$ is $UB_{\mathscr{S}_{dg}} = UB_{\mathscr{S}_d}$ and the lower bound is $LB_{\mathscr{S}_{dg}} = LB_{\mathscr{S}_d}$ on any finite undirected connected graph $G = (V, E)$, where the weight of each edge is equal to* $1$.

*Proof.* To prove the upper bound we can notice that at each step $i$, the frontier set of $\mathscr{S}_{dg}$ is a subset of the frontier set of $\mathscr{S}_d$. Thus, the set of all the possible exploration paths, on a given $G$, for $\mathscr{S}_{dg}$, is a subset of all the possible exploration paths, on that $G$, for $\mathscr{S}_d$. Thus the upper bound for $\mathscr{S}_d$ holds.

Looking at the worst-case lower bound graph for $\mathscr{S}_d$ we can notice that, the behavior of $\mathscr{S}_{dg}$ necessarily diverges from that of $\mathscr{S}_d$, when $\lfloor r \rfloor =$

$m^w - 1$ for a given $w$ ($1 \leq w \leq m$). This happens at the first stem traversal, because the robot cannot choose to enter in a loop before to explore the entire stem. In those cases it is trivial to add $O(m^m)$ single vertices on the loops, at distance $\lfloor r \rfloor + 1$ from the stem, to make feasible the behavior described for $\mathscr{S}_d$. $\square$

Theoretical results obtained in this section give some interesting insights for better understanding the performance of exploration strategies and for further explaining some experimental findings reported in the literature. Table 4.1 summarizes the worst-case bounds provided in this section.

| Worst-case upper bound | Worst-case lower bound |
|---|---|
| $UB_{\mathscr{S}_d} = 2\lceil \lvert V\rvert \cdot \mathfrak{g}\rceil \left(\ln \frac{2\lceil \lvert V\rvert\cdot\mathfrak{g}\rceil + \lfloor r\rfloor(\lfloor r\rfloor - 2) - 7}{(\lfloor r\rfloor+1)^2} - \frac{\lceil \lvert V\rvert\cdot\mathfrak{g}\rceil + \lfloor r\rfloor(\lfloor r\rfloor-2)-5}{(\lceil \lvert V\rvert\cdot\mathfrak{g}\rceil - 2)(\lfloor r\rfloor+1)} + 2\right)$ | $LB_{\mathscr{S}_d} = \Omega\left(\frac{\log\lceil \lvert V\rvert\cdot\mathfrak{g}\rceil - \log(\lfloor r\rfloor+1)^2}{\log\log\lceil \lvert V\rvert\cdot\mathfrak{g}\rceil}\lceil \lvert V\rvert\cdot\mathfrak{g}\rceil\right)$ |
| $UB_{\mathscr{S}_g} = \left(\lceil \lvert V\rvert\cdot\mathfrak{g}\rceil - \frac{\lceil \lvert V\rvert\cdot\mathfrak{g}\rceil - 1}{\lfloor r\rfloor+1}\right)\left(2\frac{\lceil \lvert V\rvert\cdot\mathfrak{g}\rceil-1}{\lfloor r\rfloor+1} - 1\right)$ | $LB_{\mathscr{S}_g} = \frac{\lfloor r\rfloor+1}{2}\left(\frac{\lceil \lvert V\rvert\cdot\mathfrak{g}\rceil - \lfloor r\rfloor}{\lfloor r\rfloor+1}\right)\left(\frac{\lceil \lvert V\rvert\cdot\mathfrak{g}\rceil - \lfloor r\rfloor}{\lfloor r\rfloor+1} - 1\right)$ |
| $UB_{\mathscr{S}_{dg}} = UB_{\mathscr{S}_d}$ | $LB_{\mathscr{S}_{dg}} = LB_{\mathscr{S}_d}$ |

**Table 4.1:** *Worst-case bounds on the number of edge traversals for the three exploration strategies $\mathscr{S}_d$, $\mathscr{S}_g$, and $\mathscr{S}_{dg}$, on any finite undirected connected graph $G = (V, E)$, given a goal percentage $\mathfrak{g} \in (0, 1]$ and a robot sensor range $r \in \mathbb{R}_{\geq 1}$, where the weight of each edge of G is equal to* 1.

As we can see, $\mathscr{S}_g$ is the exploration strategy with the highest worst-case upper bounds, while $\mathscr{S}_d$ and $\mathscr{S}_{dg}$ have the same worst-case upper bounds, namely, there is no gain in the worst case using, besides distance, the information gain as evaluation criterion. This is in line, for example, with some results obtained in specific environments in real (or realistically simulated) settings, namely including information gain in the exploration strategies does not shorten the paths for completely exploring the environments [67, 120].

Another consideration that can be made by the worst-case analysis of the three exploration strategies is that the impact of increasing perception range $r$ on the length of exploration is significant for small values of $r$, becoming less significant for large values of $r$. This holds for all exploration strategies. Note that, despite the fact that the derived bounds have the same asymptotic complexity of those that do not embed $\mathfrak{g}$ and $r$ in the computation of the bounds, the bounds we found have lower actual values, because we explicitly considered the percentage $\mathfrak{g}$ and the sensor range $r$. This analysis is of interest especially when dealing with robots, as their movements are physical and not just computational. For example, Figure 4.4 shows the trend of the worst-case bounds for $\mathscr{S}_d$ ($\mathscr{S}_{dg}$) compared to the ones found in [127] ($UB_{TK}$) and [72] ($LB_{KTS}$), and to the worst-case bounds of $\mathscr{S}_g$, considering $\lvert V\rvert = 1000$, $\mathfrak{g} = 1$, and $r \in \{0, \cdots, \lceil\lceil \lvert V\rvert \cdot \mathfrak{g}\rceil/2\rceil - 1\}$.

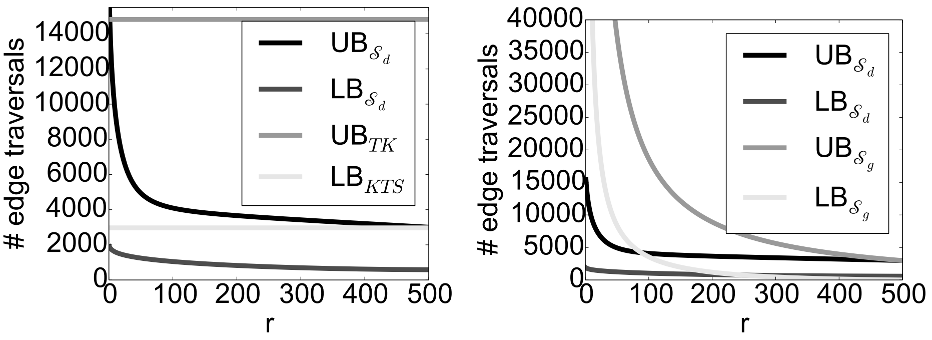


**Figure 4.4:** *The trend of worst-case bounds of $\mathscr{S}_d$ with respect to $UB_{TK}$ and $LB_{KTS}$ (left) and $\mathscr{S}_g$ (right, which, for clarity, is a zoomed portion of the complete plot), considering $|V| = 1000$, $\mathfrak{g} = 1$, and $r \in \{0, \cdots, \lceil\lceil|V| \cdot \mathfrak{g}\rceil/2\rceil - 1\}$.*

Note that the plot of the curves related to the lower bounds shows just their trends and not their actual values, as they depend on the worst-case graphs built as shown in the various proofs. In Figure 4.4 (left), we can see that $UB_{TK}$ is higher than $UB_{\mathscr{S}_d}$ (except for the case of $r < 2$ as we discussed in this section). This can be explained by the fact that $UB_{TK}$ does not fully consider the more powerful sensor that allows the perception of more vertices. A similar explanation holds for $LB_{\mathscr{S}_d}$ and $LB_{KTS}$. In Figure 4.4 (right) we can see that $UB_{\mathscr{S}_g}$ is higher than $UB_{\mathscr{S}_d}$. Also, we can observe that for low values of $r$, a small increase of $r$ leads to a large decrease of $UB_{\mathscr{S}_d}$ and $UB_{\mathscr{S}_g}$. On the other hand, we have an almost constant (and similar) trend after a certain value of $r$. From a theoretical point of view, this can be explained by looking at the function $UB_{\mathscr{S}_d}$, and noting that, when $\ln(\frac{2\lceil|V|\cdot p\rceil+(\lfloor r\rfloor+1)(\lfloor r\rfloor-1)}{(\lfloor r\rfloor+1)^2}) = \frac{2\lceil|V|\cdot p\rceil+(\lfloor r\rfloor+1)(\lfloor r\rfloor-1)}{(\lfloor r\rfloor+1)(\lceil|V|\cdot p\rceil-2)}$, $UB_{S_d}$ is linear. Looking at the $\bar{r}$ that makes the above relation true, we can observe that, when $n = 50$, $\bar{r} \approx 12$; $n = 500$, $\bar{r} \approx 67$; $n = 5000$, $\bar{r} \approx 342$; and $n = 50000$, $\bar{r} \approx 1652$. The increase of $\bar{r}$ is less than linear with respect to $n$. More generally, our analysis shows that with increasing $r$, the exploration process is shortened, which is an intuitively evident result consistent with several experimental findings (e.g., [2]).

Another insight is that, in the worst case, considering a goal percentage $\mathfrak{g}$ of vertices to perceive has the same effect of scaling (by $\mathfrak{g}$) the number of vertices of the graph representing the environment. Given the worst-case bounds found for the three exploration strategies, an exploration strategy that considers distance as criterion scales with $\mathfrak{g}$ better than one that just considers information gain.

## 4.3 Average-case analysis

As $\mathscr{S}_g$ exploration strategy has higher worst-case bounds compared to $\mathscr{S}_d$ and $\mathscr{S}_{dg}$ (which have the same worst-case bounds as shown in the previous section), we now compare $\mathscr{S}_d$ and $\mathscr{S}_{dg}$ in the average case to determine whether their performance is different, in the case of $r = \epsilon$.

Let us first introduce some notation. At a time step $i$, while the robot is at $v_i$, there is a subset $F_{i_{\min}} = \arg\min_{v \in F_i} d(v_i, v) \subseteq F_i$ of vertices in the frontier set that have the same distance from $v_i$. So, depending on how a vertex is chosen from $F_{i_{\min}}$, the robot could have better or worse performance. We call

$$\Gamma(v_i, V_i, G) = \frac{1}{|F_{i_{\min}}|} \cdot \sum_{v \in F_{i_{\min}}} (d(v_i, v) + \Gamma(v, V_i \cup P_{i+1}, G)). \tag{4.12}$$

When $\mathfrak{g} < 1$ and $\frac{|V_i|}{|V|} \geq \mathfrak{g}$ at a time step $i$, namely when the exploration process stops, the function $\Gamma()$ is set to $0$. It is fairly natural to define the average complexity on G given the starting vertex $v_0$ as $AC(v_0, G) = \Gamma(v_0, P_0, G)$. Note that having no *a priori* knowledge on the graph to be explored, to perform such computation, we would need to average over all of the possible graph structures.

Thus, to focus our analysis, we consider some graphs that model realistic indoor environments. Let us define vertex-labeled (finite undirected connected) graphs $G = (V, E)$ for which:

- some vertices $\mathscr{C} \subseteq V$ are labeled as ‘corridor’,
- the other vertices $\mathscr{R} \subseteq V$ are labeled as ‘room’ ($\mathscr{C} \cap \mathscr{R} = \emptyset$ and $\mathscr{C} \cup \mathscr{R} = V$),
- some connected sub-graphs $\mathscr{R}_i$ are defined on $G$, where all vertices are labeled as ‘room’ (in the following, with slight abuse of notation, we refer to $\mathscr{R}_i$ to indicate the set of vertices of such sub-graph; $\bigcup \mathscr{R}_i = \mathscr{R}$),
- $\mathscr{E} = \{v \in \mathscr{R} \mid \exists w \in \mathscr{C} : (v, w) \in E\}$ are the room-type vertices that act as doorways between rooms and corridors, they are further labeled as ‘entrance’,
- $\neg \exists v \in \mathscr{R}, w \in \mathscr{C} : v \notin \mathscr{E} \wedge (v, w) \in E$ (within a room only the entrance vertices can be attached to the corridors).

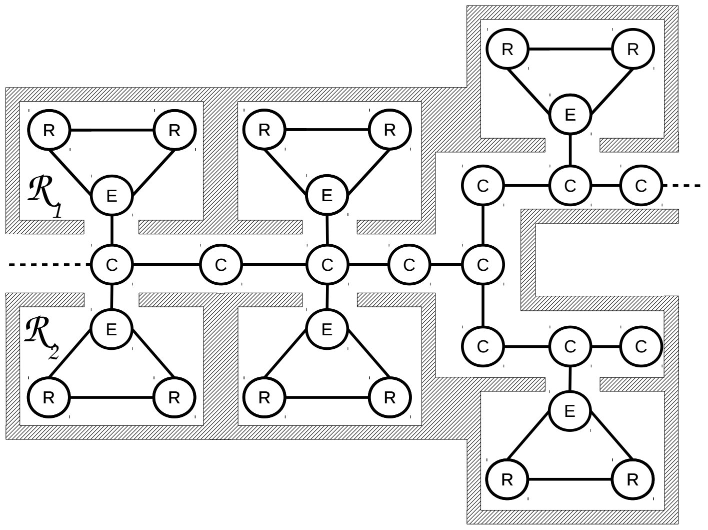


**Figure 4.5:** *Example of graph with labels (*C *'corridor';* E *'entrance';* R *'room').*

Basically, these labeled graphs are semantic maps of indoor environments that can be built autonomously by mobile robots [89]. A realistic portion of an indoor environment and the corresponding labeled graph are shown in Figure 4.5.

Let us also define a cluster of rooms as a set of $\mathscr{R}_i$, whose entrance vertices are attached to the same corridor vertex. For example, in Figure 4.5, $\mathscr{R}_1$ and $\mathscr{R}_2$ compose a cluster. More formally, given a vertex $v \in \mathscr{C}$ such that $\delta(v) > 2$ ($\delta(v)$ is the degree of $v$), a cluster of rooms is a set $K_v = \{\mathscr{R}_i \mid w \in \mathscr{R}_i \cap \mathscr{E}, (v, w) \in E\}$ (we call $K = \bigcup_{v \in \mathscr{C}} K_v$). Moreover, let us call *placement function* $q : 2^{|\mathscr{C}|} \to \mathbb{N}$ such that, given a subset $P$ of $\mathscr{C}$, $|K| = \sum_{p \in P} q(p)$. Basically $q()$ is a function that, given a subset of vertices of $\mathscr{C}$, returns the number of clusters attached to vertices in that subset. Considering such a graph, we could estimate the number of edge traversals given partial information on the corridors structure and the number of rooms for each corridor, and thus averaging just on the possible positions of the rooms to obtain $E[AC(v_0, G)]$.

We now define a class of graphs $\mathscr{G}$ on the set of graphs labeled as above. We say that $G = (V, E)$ belongs to $\mathscr{G}$ if satisfies the following properties:

Property 1 the subgraph on $G$ induced by vertices in $\mathscr{C}$ is a tree, where each leaf is attached to a vertex $v$ with $\delta(v) = 2$, Property 2 $\forall \mathscr{R}_i : \exists! v \in \mathscr{R}_i \wedge v \in \mathscr{E}$,

Property 3 given $\mathscr{R}_i$ and called $v$ the entrance vertex $v \in \mathscr{R}_i \cap \mathscr{E}$, vertices $v' \in \mathscr{R}_i \setminus \mathscr{E}$ are at distance 1 from $v$,

Property 4 $\forall v \in \mathscr{E} : \delta(v) > 2$,

Property 5 $\forall v \in \mathscr{C} : \exists e \in \mathscr{E} : (v, e) \in E \Rightarrow \forall w \in \mathscr{C} : (v, w) \in E \Rightarrow \delta(w) \leq 2$, (i.e., given a corridor vertex $v$ attached to an entrance vertex $e$, all the neighbor corridor vertices $w$ should not have more than 1 other vertex, beyond $v$, attached to them).

The following result shows the estimate on the difference between the average performance of $\mathscr{S}_d$ and $\mathscr{S}_{dg}$ on a graph $G = (V, E)$ that belongs to $\mathscr{G}$. Given an arbitrary starting vertex $v_0 \in V$ (the root) and the set of leafs $J = \{v \in \mathscr{C} \mid \delta(v) = 1\}$, we call $\mathscr{C}^j$ the vertices in $\mathscr{C}$ that are on the shortest path from $v_0$ to the leaf $j \in J$ (the path is unique due to Property 1).

**Proposition 4.8.** *Consider a graph $G = (V, E)$ that belongs to $\mathscr{G}$, a starting vertex $v_0 \in \mathscr{C}$ with $\forall v \in N(v_0), \delta(v) = 2$ (where $N(v_0)$ is a function that returns the set of neighbor vertices of $v_0$), a goal percentage $\mathfrak{g} = 1$, and a sensor range $r = \epsilon$ ($\epsilon \to 0$). An estimate for the difference between the average performance of $\mathscr{S}_d$ and $\mathscr{S}_{dg}$ is*

$$\sum_{j \in J} B^j \cdot (|\mathscr{C}^j| - 2) \frac{q(\mathscr{C}^j)}{q(\mathscr{C}^j) + 1} - 2 \frac{|J| - q(J)}{|J|},$$

*where $B^j$ is the probability of $\mathscr{C}^j$ to be explored at the very end of the exploration process and $q()$ is a placement function.*

*Proof.* Looking at $\mathscr{S}_{dg}$, because of Property 4 and Property 5, while the robot explores a corridor, it explores every room it encounters before continuing on the corridor. A room, when chosen, is completely explored, due to Property 2 and Property 3 and to the fact that the starting vertex $v_0 \in \mathscr{C}$. Further, Property 2 guarantees that after exploring a room $\mathscr{R}_i$, the robot does not directly head to another room $\mathscr{R}_l$ ($i \neq l$) without first going back to a vertex in $\mathscr{C}$. This is clearly visible in Figure 4.5: no matter which starting vertex is taken, $\mathscr{S}_{dg}$, while exploring the corridor, will explore first the encountered rooms (i.e., choosing a vertex in the corridor has probability 0 when there are rooms attached to the current vertex). Also, due to Property 3, the mean tour cost $T(\mathscr{R}_i)$ for visiting a room $\mathscr{R}_i$ is the same for $\mathscr{S}_{dg}$ and

$\mathscr{S}_d$. Thus, the average complexity of $\mathscr{S}_{dg}$ is

$$E[AC_{dg}(v_0, G)] \approx \sum_{\mathscr{R}_i} T(\mathscr{R}_i) + \sum_{j \in J} B^j(|\mathscr{C}^j| + 2 \sum_{l \in J, l \neq j} |\mathscr{C}^l|) - 2\frac{q(J)}{|J|}. \tag{4.13}$$

$B^j$ is the product of the inverse of the corridor branching factors encountered while visiting vertices in $\mathscr{C}^j$.

Instead, for $\mathscr{S}_d$, while the robot explores a corridor, it could choose not to explore some encountered rooms. Hence, it could happen that, once the robot has reached a leaf, it should go back to explore rooms left behind (recall that corridor vertices form a tree because of Property 1). This can be seen in Figure 4.5, as $\mathscr{S}_d$, when the robot's current vertex is in the corridor and there are neighbor vertices labeled as 'room', has a probability strictly greater than $0$ to choose a vertex in the corridor, possibly leaving a room as the last area to explore. The complexity to visit the rooms is the same as $\mathscr{S}_{dg}$ because of Property 3. The only difference between $\mathscr{S}_d$ and $\mathscr{S}_{dg}$ is during the exploration of the last corridor vertices that are in $\mathscr{C}^j$. With $\mathscr{S}_d$ the robot could have to go back to some unexplored rooms left along the last $\mathscr{C}^j$ and so, differently from $\mathscr{S}_{dg}$, could end the exploration in a room.

Defining $\mathscr{R}^j_{last}$ as the unexplored room left along $\mathscr{C}^j$ which has the closest entrance vertex to the starting point $v_0$, let us call $\Delta^j$ the mean distance between $\mathscr{R}^j_{last}$ and the leaf $j$, and $\Lambda^j$ the difference between a tour and a path to explore a room $\mathscr{R}_i$ on $\mathscr{C}^j$, on average, weighing each difference with the probability that room $\mathscr{R}_i$ is $\mathscr{R}^j_{last}$. In summary, for $\mathscr{S}_d$ we have that the average complexity is

$$E[AC_d(v_0, G)] \approx \sum_{\mathscr{R}_i} T(\mathscr{R}_i) + \sum_{j \in J} B^j(|\mathscr{C}^j| + \Delta^j - \Lambda^j + 2 \sum_{l \in J, l \neq j} |\mathscr{C}^l|). \tag{4.14}$$

There should be also the term $-2$ multiplied by the probability for $\mathscr{S}_d$ to end in a room, but it is neglected because is close to $0$. The latter probability corresponds to the probability to explore all the cluster in $\mathscr{C}$ times the probability that a cluster is attached to a leaf $j$.

Note that $\Lambda^j$ (the average difference between a tour and a path to explore a room attached to a corridor vertex in $\mathscr{C}^j$) is always equal to 2 because of Property 3. Having no a priori knowledge about the size of the clusters, we approximate the probability to completely explore a given cluster $K_v$ of rooms (before going ahead along the corridor) as a constant value $p_K$, which can be seen as the mean of the probabilities to completely explore each cluster $K_v$. Under this hypothesis, the probability of not exploring a number $m$ of clusters $K_v$ along corridor vertices $\mathscr{C}^j$ is a binomial $S \sim \mathscr{B}(q(\mathscr{C}^j), 1 - p_K)$. We want to estimate $\Delta^j$ through an aleatory variable $D$. We first find $E[D|S]$. This conditional expectation can be approximated imagining the line as a continuous line and the relative position of the clusters as independent. The traveled distance to explore $S = m$ clusters left, is an aleatory variable $Z = \max(X_1, X_2, ..., X_m)$ where $X_v \sim U(0, |\mathscr{C}^j| - 2)$ (minus two because of the starting vertex constraint: $q(N(v_0) \cup v_0) = 0$) is the position of the cluster $K_v$ along $\mathscr{C}^j$. Thus, the cumulative distribution function of $Z$ is:

$$F_Z(t) = P(Z \leq t) =$$

(because of the definition of $Z$)

$$= P(X_1 \leq t \wedge X_2 \leq t \wedge ... \wedge X_m \leq t) =$$

(since we assumed the uniform distributions as indipendent)

$$= P(X_1 \leq t) \cdot P(X_2 \leq t) \cdot ... \cdot P(X_m \leq t) =$$

(substituting the cumulative distribution function for a uniform distribution)

$$= \left( \frac{t}{|\mathscr{C}^j| - 2} \right)^m .$$

Now we can compute the probability density function:

$$f_Z(t) = \frac{dF_Z(t)}{dt} = m \cdot \frac{t^{m-1}}{(|\mathscr{C}^j| - 2)^m}$$

and applying the definition of the expected value for a continuous aleatory variable we have

$$\begin{aligned} E[D \mid S = m] = E[Z] = & \\ & = \int_0^{|\mathscr{C}^j|} m \cdot \frac{t^{m-1}}{(|\mathscr{C}^j| - 2)^m} \cdot t \cdot dt = \\ & = \frac{m}{(|\mathscr{C}^j| - 2)^m} \int_0^{|\mathscr{C}^j|} t^m \cdot dt = \\ & = \frac{m}{m+1} (|\mathscr{C}^j| - 2). \end{aligned}$$

In the hypothesis that $p_K < 1$, an estimate for the mean gain is:

$$E\left[\Gamma_d(G, v_0) - \Gamma_{dg}(G, v_0)\right] = E\left[\Gamma_d(G, v_0)\right] - E[\Gamma_{dg}(G, v_0)] \approx$$

(substituting the expected values)

$$\approx \sum_{j \in J} B^j \left(\Delta^j - \Lambda^j\right) + 2\frac{q(J)}{|J|} =$$

(splitting the sum)

$$= \sum_{j \in J} B^j \Delta^j - \sum_{j \in J} B^j \cdot 2 + 2\frac{q(J)}{|J|} \approx$$

(because $\Delta^j \approx E[D]$)

$$\approx \sum_{j \in J} B^j E\left[D\right] - 2 + 2\frac{q(J)}{|J|}. \tag{4.15}$$

Thus, remembering that $\Delta^j \approx E[D] = E\left[E\left[D \mid S = m\right]\right]$ where $E\left[D \mid S = m\right]$ is the quantity found above and that $S \sim \mathscr{B}(q(\mathscr{C}^j), 1 - p_K)$, we have that Equation (4.15) becomes

$$\approx \sum_{j \in J} B^j \left[\sum_{m=0}^{q(\mathscr{C}^j)} \binom{q(\mathscr{C}^j)}{m} \left(p_K^{q(\mathscr{C}^j)-m}(1-p_K)^m \cdot \frac{m}{m+1}(|\mathscr{C}^j| - 2)\right)\right] +$$
$$- 2\frac{|J| - q(J)}{|J|} =$$

(simplifying this known sum)

$$= \sum_{j \in J} B^j \left[\frac{q(\mathscr{C}^j)(1-p_K) - p_K(1 - p_K^{q(\mathscr{C}^j)})}{(q(\mathscr{C}^j)+1)(1-p_K)}(|\mathscr{C}^j| - 2)\right] - 2\frac{|J| - q(J)}{|J|} \approx$$

(approximating $p_K(1 - p_K^{q(\mathscr{C}^j)}) \approx 0$)

$$\approx \sum_{j \in J} B^j \cdot (|\mathscr{C}^j| - 2)\frac{q(\mathscr{C}^j)}{q(\mathscr{C}^j)+1} - 2\frac{|J| - q(J)}{|J|}.$$

$\square$

This proposition provides insights on how graphs belonging to $\mathscr{G}$ are explored by a robot that employs either $\mathscr{S}_d$ or $\mathscr{S}_{dg}$ and quantify the difference of their performance in terms of number of edge traversals. In particular, looking at the average case, the result obtained shows that considering expected information gain in exploration strategies provides an advantage in a class of graphs that could model an indoor environment. This can be intuitively explained by the fact that the robot visits all rooms encountered without the need to go back to visit some rooms left behind while traversing the corridor.

This result is also supported by simulated experiments that have been conducted in randomly generated environments (the simulator has been implemented in Python). The experimental setting is as follows. The input parameters to generate an environment are the number of vertices labeled as 'corridor' $|\mathscr{C}|$ and the number of clusters $|K|$ of rooms to be placed along the corridors. We derive the number of clusters by using a placement function $g$ which depends only on the corridors length. More formally, let us define $I = N(v_0) \cup \{v_0\}$, $W = \mathscr{C} \setminus I$. For each set $A \subseteq W$, $q(A) = |A| \cdot |K|/|W|$ ($q(I) = 0$) and $d_K = |K|/|W|$, which represents the density of clusters in the corridors. For the experiments we considered the following values for $|\mathscr{C}| = \{50, 100, 150, 200, 250\}$ and $d_K = \{0.2, 0.3, 0.4\}$. To generate random environments, first trees composed of corridor vertices have been generated with a mean number of leaves equal to 5. Each cluster has a random number of rooms between 1 and 4 generated with a uniform probability. Then the clusters have been attached to the tree randomly (preserving the properties). Since the shape of each single room does not influence the gain of $\mathscr{S}_{dg}$ over $\mathscr{S}_d$ (as shown above), we decided to choose a full connected subgraph of three vertices each. Finally, the starting position has been chosen from those satisfying the constraint. See Figure 4.6 for an example of such random generated graph. For each of such generated environments and starting vertex, the exploration strategies $\mathscr{S}_d$ and $\mathscr{S}_{dg}$ are run 50 times and the difference in their performance is computed, together with the error of the estimate given in Proposition 4.8.

Our results (over 600 randomly generated graphs; see Table 4.2) suggest that $\mathscr{S}_d$ always performs worse than $\mathscr{S}_{dg}$, given this class of the environments. For example, considering $|V| = 150$ and a number of clusters $|K| \sim d_K|V|$, for $d_k = 0.4$ the difference between the mean traveled distances of $\mathscr{S}_d$ and $\mathscr{S}_{dg}$ is 63.3 (2.1 standard deviation) edge traversals. On average the gain seems to be almost independent of the number of rooms in each cluster. It appears to depend just on the the shape of the tree composed of corridor vertices. This makes the estimate fairly good, since there is no assumption about the clusters size. The error between the estimate on the gain of Proposition 4.8 and the real one seems to be limited. We notice that

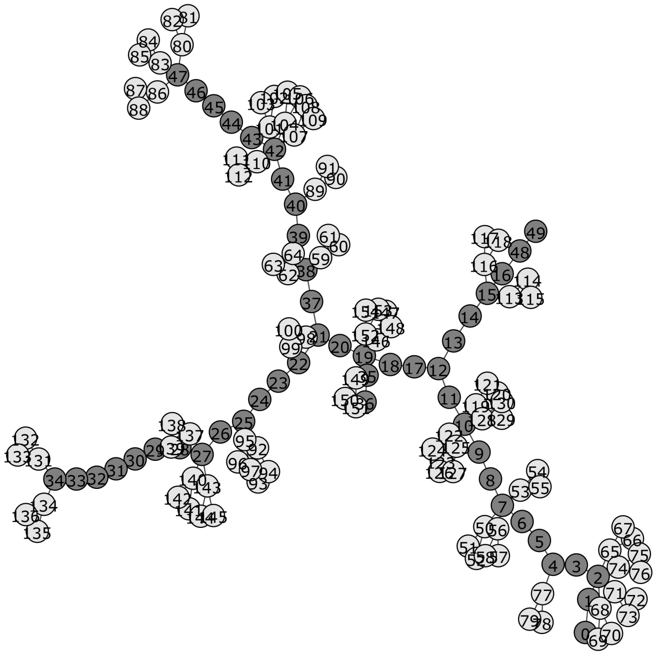


**Figure 4.6:** *An example of random $\mathscr{G}$ graph with room (light grey) and corridor (dark grey) vertices and $|\mathscr{C}| = 50$, $d_K = 0.3$.*

| | $d_K = 0.2$ | | $d_K = 0.3$ | | $d_K = 0.4$ | |
|---|---|---|---|---|---|---|
| $\lvert\mathscr{C}\rvert$ | Gain | Error | Gain | Error | Gain | Error |
| 50 | 13.8 (3.1) | -2.1 (3.8) | 15.4 (1.9) | -1.0 (2.2) | 16.9 (1.2) | 0.1 (1.2) |
| 100 | 14.7 (1.7) | 0.2 (2.2) | 15.9 (3.0) | -1.5 (3.3) | 21.8 (1.3) | 0.4 (1.3) |
| 150 | 17.0 (2.9) | -3.0 (4.2) | 28.8 (3.3) | 0.4 (3.3) | 63.3 (2.1) | 0.5 (2.1) |
| 200 | 66.1 (3.6) | -0.3 (3.7) | 57.1 (3.6) | 0.6 (3.6) | 66.5 (3.8) | 0.4 (3.8) |
| 250 | 70.1 (4.1) | -0.1 (4.1) | 76.7 (7.0) | -2.6 (7.4) | 104.5 (5.5) | -2.4 (6.0) |

**Table 4.2:** *Difference of traveled distance between $\mathscr{S}_d$ and $\mathscr{S}_{dg}$ (average and standard deviation) on randomly generated graphs belonging to $\mathscr{G}$ and the error between the estimate given in Proposition 4.8 and the actual difference (average and standard deviation with respect to $0$).*

the goodness of the estimate depends on the ratio between the cardinality of the set of leaves $|J|$ and that of corridor vertices $|\mathscr{C}|$. For example, in one of the experiments the error was $-2.1$ (3.8), over a gain of 13.8 (3.1), for $|\mathscr{C}| = 50$, $|J| = 5$, and $d_K = 0.2$. This could be explained by the fact that if $|J|/|\mathscr{C}|$ is high (in the example it is 0.1), the gain of $\mathscr{S}_{dg}$ over $\mathscr{S}_d$ is smaller and, because of the approximations in the computation of the estimate (shown in the proof), the percent error is higher. Intuitively, corridors are generally shorter when the ratio $|J|/|\mathscr{C}|$ is higher and so a robot employing $\mathscr{S}_d$ has to re-traverse less vertices to visit unexplored rooms. Instead, considering $|\mathscr{C}| = 200$ (with $|J| = 6$ and $d_K = 0.2$), $|J|/|\mathscr{C}| = 0.03$ and the percent error is smaller (namely, $-0.3$ (3.7) over a mean gain of 66.1 (3.6)). Further, we can notice that there is a slight overestimate, probably due to the approximations we made in the proof which, anyway, seems not to grow with the gain nor with the number of corridor vertices.

Thus, differently from the worst case, the obtained theoretical results in the average case (corroborated by experiments) show that taking into account also information gain in selecting the next destination location provides advantages over considering only distance on graphs modeling realistic indoor environments. This could be intuitively explained by the fact that using only distance does not fully take into account the structure of the environment, while the one with also information gain does. In the worst case, this benefit does not emerge as we have to consider worst-case environments, which are usually not realistic. Hence, the analysis on the average complexity provides some insights on the reason why in some indoor environments, exploration strategies that consider expected information gain besides distance as evaluation criteria perform well, as shown, for example, in experimental results performed, e.g., by Amigoni [2], Basilico and Amigoni [16].

Analyzing some exploration strategies in environments modeled as graphs allows to study the combinatorics of the exploration problem. By deriving the bounds in the worst and average cases, we contribute to shift from results obtained experimentally with real (and realistically simulated) exploring robots to a more theoretically-grounded justification of the latter, helping to answer the question: “which exploration strategy performs better in a given scenario?”.

# Part II
# Improve online exploration strategies

The second part of this dissertation shows a possible way to improve multirobot exploration systems. Our approach starts from considering the fact that, as presented in Section 2.2, most exploration strategies and coordination methods base their decisions on the current metric map built so far. Also, the common approach for coordinating robots is to send just one robot to a candidate location, following the single-robot-per-task paradigm, in order to spread robots over the environment. Given the increasing availability of reliable semantic mapping systems and the limited number of studies on their *use* for exploration, in this part, we propose a coordinated multirobot exploration system that operates in search and rescue settings and that exploits semantic labels to improve the performance of the system in terms of explored area in a given time interval. In the following, we present the formulation of the problem with the model of the robotic system employed and the details and the experiments on the proposed semantic-based multirobot exploration system.

# 5 Model and assumptions

In this (short) chapter we present the basic model and assumptions that are used for Chapters 6, 8, and 9, instantiating some of the relevant dimensions presented in Section 1.3. Note that the behavior of the robots is basically the one presented in Section 1.1. Also note that some of the parameters are specified within each chapter.

The robotic system used is composed of one or more homogeneous robots. Each robot is a differential wheeled robot, like a Pioneer P3AT (see Figure 5.1), equipped with some sensors, like a laser range scanner with a Field Of View (FOV) and angular resolution, and a sonar ring, that allow the robot to navigate and map the environment. Note that, if FOV is $360°$, two laser range scanners are mounted at the same height and back-to-back for covering a $360°$ around the robot. Each robot starts from an initial position in an initially unknown environment and builds a two-dimensional occupancy grid map of the explored environment. Each cell is either known, if the robot perceived the corresponding area, or unknown. Known cells can be free or occupied (by obstacles). So, the knowledge representation is discrete-based. The current map built so far is stored in the robot memory.

When more than one robot is employed, the global map of the environment is maintained by a base station, whose position is fixed in the environment, and to which robots send their maps every $2.5\,\mathrm{s}$. We assume that communication is error-free and unlimited in range and bandwidth (effects of more realistic communication models on exploration are discussed for example by Tuna et al. [128]). Our exploration system is largely independent of the mapping system employed to incrementally build the grid map. In our experiments, we use a simple scan matching method, inspired to that of Lu and Milios [79], in which a new acquired scan is aligned with the

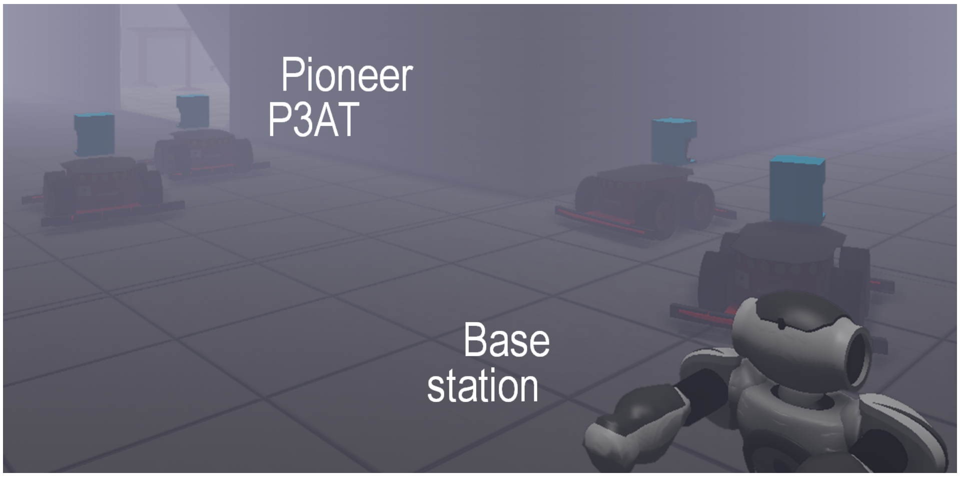


**Figure 5.1:** *Four P3ATs and a NAO that acts as base station in USARSim.*

current map (using odometry as initial guess) and the occupancy grid is updated correspondingly. Since we are not interested in analyzing the quality of the resulting map, we assume that the mapping module is error-free.

Given a map represented as above, to calculate candidate destination locations, we consider reachable free cells that are on the boundary between known and unknown cells. Then, a set of 8-adjacent boundary cells are grouped in a cluster, called frontier. The centroid of each cluster is considered as a *candidate destination location* to reach. The selection of such locations defines the motion model of the robotic system. Each candidate destination location is represented as a cell position in the grid. Note that when the angle covered by the FOV of the laser range scanner around the robot is less than $360°$, also the orientation that the robot should take once in the location is represented: the orientation is toward the unknown area along the perpendicular to the line tangent to the corresponding frontier and passing through the candidate destination location cell. Path planner uses A* on the grid map. Sonars are used for obstacle detection during navigation.

The criteria we consider for defining the exploration strategies include the following:

- $A(p)$ is the estimated amount of free area beyond the frontier of $p$ computed according to the length (in cells) of the frontier. The larger its value, the more information is expected to be acquired from $p$.
- $d_{L_2}(p, r)$ is the Euclidean distance between $p$ and current position of $r$. Using Euclidean distance instead of actual distance calculated by path

planner drastically reduces the computational effort in calculating this criterion without affecting too much the estimated utility $u(p, r)$, as some preliminary experiments we performed have shown.

- $d_{PP}(p, r)$ is the distance between $p$ and current position of $r$; given $p$ and $r$, this criterion is calculated using the path planner that returns the length of the path.
- $b(p, r)$ is an estimate of the energy spent by $r$ for reaching $p$; the larger its value, the smaller the amount of residual energy in the battery (0 = full, 1 = empty); this criterion is calculated considering a very simple model in which the power consumption is related to the time required for reaching $p$, computed according to the path that $r$ should follow and according to linear and angular velocities of the robots.
- $o(p, r)$ is the cost related to the heading change that the robot should perform, computed according to the difference between the orientation required at $p$ and the current orientation of $r$.
- $P(p)$ is the probability that a robot, once reached $p$, will be able to transmit information (e.g., the perceived data or the locations of victims) to the base station (whose position in the environment is known), this criterion depends on the distance between $p$ and the base station.

All these criteria can be calculated from the robots' status and from the metric grid map. We assume that the exploration solution is not a tour, but a path, namely it is not required that robots (although they could revisit some already explored portion of the environments) return to the starting point when the termination criterion is met. As said, this kind of solution is often considered, for example, in mapping or search and rescue scenarios where returning to the starting point can be performed in a second stage after the more important exploration task is completed. Experimental results shown in the next chapters are evaluated by optimality criteria depending on the task, as we discuss next.

6

# A semantically-informed multirobot system

Some robotic exploration systems that exploit semantic information have been presented in literature (see Section 2.2), finding out that use of semantic information can reduce the time required to cover a given amount of area and can increase the *total* amount of area mapped by robots in a given time interval. In this chapter, we extend these results by showing that semantic knowledge can also be used to significantly improve the exploration of *relevant areas* of indoor environments. We assume that *a priori* and reliable information about the areas of the environment that are considered relevant is available, for example, provided by humans. This assumption is of interest in realistic scenarios. In a search and rescue setting, the *a priori* information could be the possible location of victims or the preferred areas to search first, given by human rescuers. For example, if a disaster happens in a building during office hours, victims are most likely located in the offices, and, thus, robots should focus on searching small-size rooms. If it happens during lunch time, robots should head to large-size rooms, like a canteen. In the following of this chapter, we consider the following *a priori* information about victims location: either victims are in small rooms or victims are in big rooms.

Our system is composed of multiple robots that operate according to the behavior model we showed before in Section 1.1 and with capabilities presented in Chapter 5. Note that we are assuming that, as typically done in autonomous mobile robotics, the perception and decision models are event-based. We originally address the following problem: to what extent is it possible and convenient to exploit semantic information to efficiently explore areas (of an initially unknown environment) that are considered relevant? The main original contribution of the proposed approach is thus a system that exploits semantic information to improve exploration. In particular, in the following, we propose a method for evaluating candidate locations, which is a variant of that presented by Basilico and Amigoni [16], and a method for allocating robots to candidate locations. In this chapter, we aim at showing that, when *a priori* knowledge on victims' locations is available (i.e., preferred areas to visit are specified), the use of semantic information could improve the performance of exploration of relevant areas of the environment, besides the total one, differently from the works presented in Section 2.2 (like [121] or [24]) that use semantic information to improve the total explored area of the environment. Also, we attempt to overcome the ST-SR assumption by allocating more robots to the same candidate locations according to a multi-robot tasks (MR) paradigm. (Recall that the ST-SR assumption means that a single task (ST), namely a single candidate destination location, is assigned to each robot, and a single robot (SR) is assigned to each task.) Specifically, we aim at showing that semantic information enables the possibility to determine the ideal number of robots to send to a specific area so that exploration can proceed faster and more effectively.

Note that, since in this chapter we are assuming that some *a priori* information is available about relevant areas, the works in literature including semantic information are not directly comparable with our approach, in terms of the relevant areas explored. However, in our experimental simulated activities, we will compare the proposed exploration strategy with that proposed by Basilico and Amigoni [16], which has been experimentally proven to perform well in search and rescue scenarios, but does not consider any semantic information.

## 6.1 MCDM-based exploration strategy

Exploration strategies use several criteria to evaluate the goodness of a candidate location $p$ for a robot $r$ with the evaluation function $u(p, r)$. In particular, we consider the criteria $A, d_{L_2}, b$ that can be computed on the basis of the metric map (see Chapter 5). Additionally, we assume that the robotic system presented in the previous chapter has a semantic map that labels each free cell of the grid map with its room type (i.e., 'corridor', 'small room', 'medium room', 'big room') and with the number of doorways present in the room in which the cell is located. This semantic map can be built exploiting any available method (e.g., [89]). Thus, the kind of knowledge, besides being metric-based, is also semantic-based and the following additional evaluation criteria are considered:

- $S(p)$ is the relevance of $p$ (from 0, not relevant, to 1, relevant), calculated according to the semantic label of $p$ and the *a priori* knowledge on victims' locations. For example, if it is known that victims are most likely in big rooms, and $p$ is in a big room, $S(p) = 1$, while if $p$ is in a small room, and under the same hypothesis about the location of victims, $S(p) = 0$. If $p$ is in a corridor, regardless the hypothesis on victims' locations, $S(p) = 0.15$, as corridors are usually important to reach relevant rooms. The values for $S(p)$ have been manually set to obtain good performance after experiments with different combinations of values. Different value combinations (e.g., range $[0.10, 0.50]$ for $S(p)$ with $p$ in corridors), that maintain relevance of corridors and of rooms according to the hypothesis on victims' location, have been experimentally demonstrated to have similar performance.

- *ND*$(p)$ is the number of doors in the room where $p$ is located. This criterion evaluates the connectivity of a room with other rooms. The idea is that a highly-connected room should be visited to ease finding relevant rooms.

We assume that semantic labeling used to calculate the criteria $S(p)$ and *ND*$(p)$ is perfect. This assumption will be relaxed later to experimentally verify the robustness of the approach.

We define exploration strategies using the Multi-Criteria Decision Making (MCDM) approach introduced by Basilico and Amigoni [16] (see Section 2.2), to combine criteria. We selected the MCDM approach because it is theoretically grounded and allows to easily integrate several criteria in a utility function. Below we show the criteria adopted and the weights defined.

For our semantically-informed exploration strategy (*S-MCDM*), we use the criteria $N = \{A, d_{L_2}, b, S, \mathit{ND}\}$ defined above and the weights reported in Table 6.1 (left). The weights of the subsets of criteria not reported in the table are calculated by summing the weights of the individual criteria. Note that in selecting these weights, we have chosen values reasonably (e.g., criteria $d_{L_2}$ and $A$ have the same importance, so their weights are equal). Moreover, criteria $d_{L_2}$ and $b$ are redundant, since both prefer candidate locations close to the robot and a candidate location satisfies both criteria well or both not well, and so $\mu(\{d_{L_2}, b\}) < \mu(\{d_{L_2}\}) + \mu(\{b\})$. Intuitively, both measure the cost to reach a candidate location: the more distant a candidate location is, the more battery is used to reach it. Criteria $A$ and $d_{L_2}$ are instead synergic (one prefers candidate locations on long frontiers while the other one prefers candidate locations close to the robot

| | criteria | $\mu()$ | criteria | $\mu()$ | criteria | $\mu()$ |
|---|---|---|---|---|---|---|
| **S-MCDM** | $A$ | 0.09 | $d_{L_2}, b$ | 0.09 | $A, b$ | 0.15 |
| | $d_{L_2}$ | 0.09 | $d_{L_2}, S$ | 0.8 | $d_{L_2}, b, S$ | 0.8 |
| | $b$ | 0.02 | $b, S$ | 0.6 | $A, S$ | 0.65 |
| | $S$ | 0.5 | $A, d_{L_2}, b$ | 0.3 | $A, d_{L_2}, b, S$ | 0.8 |
| | *ND* | 0.3 | $A, d_{L_2}, S$ | 0.8 | | |
| | $A, d_{L_2}$ | 0.3 | $A, b, S$ | 0.65 | | |

| | criteria | $\mu()$ |
|---|---|---|
| **D-MCDM** | $A$ | 0.4 |
| | $d_{L_2}$ | 0.4 |
| | $b$ | 0.2 |
| | $A, d_{L_2}$ | 0.95 |
| | $A, b$ | 0.7 |
| | $d_{L_2}, b$ | 0.4 |

**Table 6.1:** *Weights of MCDM-based exploration strategies.*

and a candidate location can satisfy one criterion well and the other one not well) and so $\mu(\{A, d_{L_2}\}) > \mu(\{A\}) + \mu(\{d_{L_2}\})$. Values of weights have been set to obtain good performance, according to criteria importance and relations [16]. Slightly varying the selected weights values ($\pm 10\%$), we experimentally obtained similar performance. Principled methods for selecting weights are discussed by Basilico and Amigoni [16].

For comparing the performance of S-MCDM, we chose a state-of-the-art exploration strategy. Specifically, we defined another MCDM-based exploration strategy (*D-MCDM*), whose criteria set is $N = \{A, d_{L_2}, b\}$, similarly to the work of Basilico and Amigoni [16], and with weights reported in Table 6.1 (right). As discussed in Section 2.2, existing exploration strategies that exploit semantic information focus on improving the total explored area and not the relevant one. Furthermore, the work of Calisi et al. [24] is not easily configurable in our setting, as PROLOG rules should be set. Nevertheless, the D-MCDM exploration strategy has been shown by Basilico and Amigoni [16] to be very effective in exploring environments (in particular, it outperformed the exploration strategies proposed by Visser and Slamet [131] and Amigoni and Caglioti [3]).

## 6.2 ST-MR coordination method

The coordination method we use is market-based [140]. The base station regularly sets up auctions in which candidate locations (generated on current frontiers as discussed before) are auctioned to the robots, which bid on them. This process allocates candidate locations $p$ to robots $r$ attempting to maximize the sum of utilities $u(p, r)$. In our system, the coordination method can allocate multiple robots (MR) to the same candidate locations. For example, allocating two robots to the same candidate location in a big room could speed up the exploration of the room, overcoming potential negative effects due to the initially overlapping views of the two robots.

We employ a fuzzy-based function $i(p)$ that computes the ideal number of robots (1, 2, or 3, in our experiments) that should be assigned to a candidate location $p$, according to the semantic label given to $p$ and to some other features. In particular, if $p$ is located in a room ('small room', 'medium room', or 'big room'), the features considered are the room area, the free area percentage of the total area in the room (visibility), the number of doors, and the already perceived area of the room. Note that an estimate of the already perceived area of a room can be computed by having a knowledge base that associates the semantic labels of rooms to the corresponding average area (see, e.g., [80]). Figure 6.1 shows the membership functions for the input features and for the output we have used for experiments of the next section. When slightly varying the selected fuzzy values ($\pm 10\%$), we experimentally obtained similar performance. Given $p$, if the room in which $p$ is located is large, the number of its doors is large, its visibility is large, and the amount of already perceived area is small, then more robots are allocated to $p$. Other examples of the rules for determining the ideal number of robots $i(p)$ to be allocated to $p$ (in a room) are reported in Algorithm 6.1.

```
1 if RoomSize is SMALL and #Doors is HIGH and Visibility is LOW and
    AlreadyPercArea is MEDIUM then #Robots is MEDIUM;
2 if RoomSize is BIG and #Doors is LOW and Visibility is LOW and
    AlreadyPercArea is HIGH then #Robots is LOW;
3 if RoomSize is BIG and #Doors is MEDIUM and Visibility is MEDIUM and
    AlreadyPercArea is LOW then #Robots is HIGH;
```

**Algorithm 6.1:** Sample of rules for calculating the ideal number of robots that can be allocated to $p$ (in a room).

Similarly, if $p$ is located in a corridor (label 'corridor'), the features considered are the length of the corridor, the number of doors, the number of intersecting corridors, and the already perceived area of the corridor. The membership functions and the rules are similar to those for the room case, as shown in Figure 6.2 and in Algorithm 6.2.

Each robot $r$ evaluates all candidate locations $p$, as auctioned by the base station every $5\,\mathrm{s}$ or when requested by a robot that has reached its assigned location, according to the exploration strategy and submits bids $u(p, r)$ accordingly. We propose two coordination methods executed by the base station to allocate candidate locations to robots. The first coordination method (*MRv1*) works as reported in Algorithm 6.3. Basically, MRv1 greedily allocates the best pair $(p^*, r^*)$, avoiding to allocate $p^*$ to more than $i(p^*)$ robots.

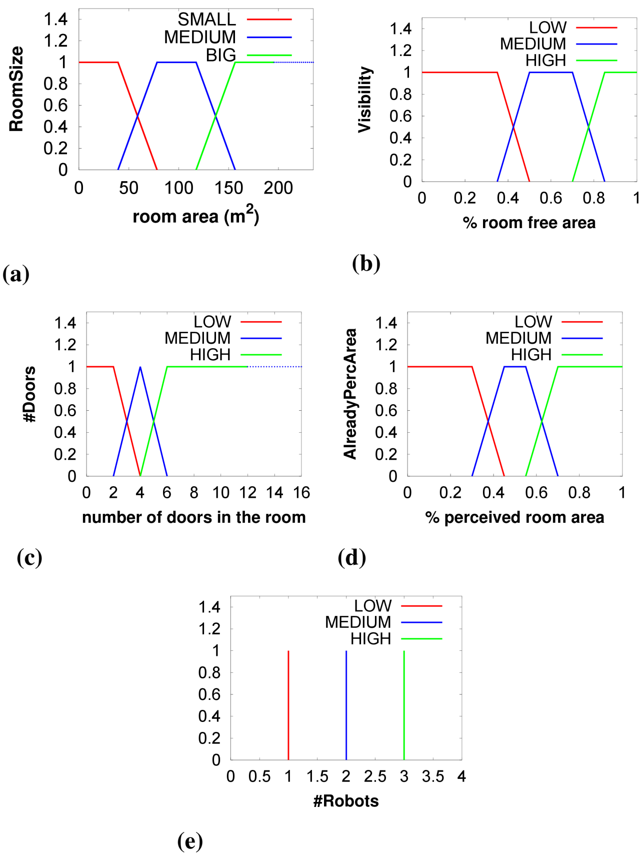
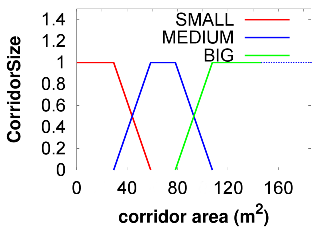


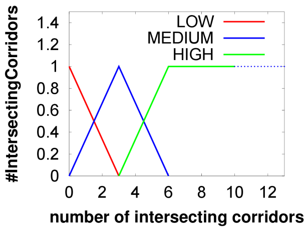
**Figure 6.1:** *Membership functions for the input features (a-d) and for the output (e), when $p$ is in a room.*

The second coordination method, called *MRv2*, is similar to MRv1, but, after each allocation of a robot to a $p^*$ (step 4), it discounts the utility of $p^*$ for other robots, according to the number of robots already allocated to $p^*$ (similarly to the work of Stachniss et al. [121]). Figure 6.3 shows the discount factor that decreases linearly until the number of allocated robots is less than or equal to $i(p^*)$, and then decays exponentially. The rationale is that assigning to $p^*$ less robots than $i(p^*)$ could be a necessity (e.g., there are not enough robots) and that assigning to $p^*$ more robots than $i(p^*)$ is not useful to speed up exploration.

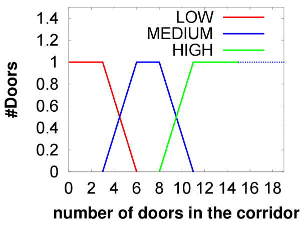
The two proposed ST-MR coordination methods are experimentally compared to a standard coordination method (ST-SR) [140], which allocates just one robot to a candidate location in a greedy fashion. Namely, it runs MRv1 with $i(p) = 1$ for every $p$.



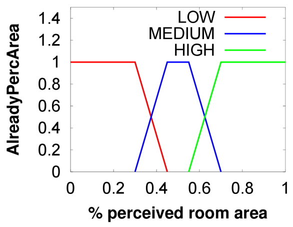
(a)



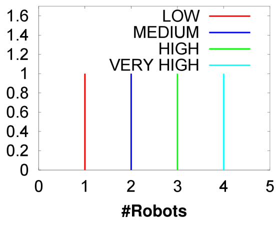
(b)



(c)



(d)



(e)

**Figure 6.2:** *Membership functions for the input features (a-d) and for the output (e), when $p$ is in a corridor.*

## 6.3 Simulation results

In order to perform replicable tests under controlled conditions, we use a robot simulator. We selected USARSim [27], because it is a realistic and reliable 3D robot simulator. Note that, despite the fact that we are just focusing on the decisions about the next locations to reach (represented in 2D), the use of a 3D simulator, which simulates laws of physics that influence the behavior of robot sensors and actuators, allows to get results that could be closer to the ones that could be obtained with real robots. We developed the multirobot system controller software that operates with that simulator[1].

[1]The code and the experimental data are publicly available at `http://sourceforge.net/projects/polimirobocup`.

```
1 if CorridorSize is SMALL and #Doors is HIGH and #IntersectingCorridors is
  MEDIUM and AlreadyPercArea is MEDIUM then #Robots is LOW;
2 if CorridorSize is SMALL and #Doors is MEDIUM and #IntersectingCorridors is
  LOW and AlreadyPercArea is LOW then #Robots is MEDIUM;
3 if CorridorSize is MEDIUM and #Doors is MEDIUM and #IntersectingCorridors
  is MEDIUM and AlreadyPercArea is LOW then #Robots is HIGH;
4 if CorridorSize is BIG and #Doors is HIGH and #IntersectingCorridors is
  MEDIUM and AlreadyPercArea is LOW then #Robots is VERY_HIGH;
```

**Algorithm 6.2:** Sample of rules for calculating the ideal number of robots that can be allocated to $p$ (in a corridor).

```
1 collect bids $u(p, r)$, which are calculated using (Equation (2.8));
2 while $\exists$ robot $r$ not allocated and candidate location $p$ do
3     find the pair $(p^*, r^*)$: $(p^*, r^*) = \arg\max_{p,r} u(p, r)$;
4     allocate $p^*$ to $r^*$;
5     if $i(p^*)$ is equal to the number of robots already assigned to $p^*$ then
6         eliminate $p^*$;
      end
7     eliminate robot $r^*$;
  end
```

**Algorithm 6.3:** MRv1.

We report simulated experiments conducted in two indoor environments, called *office* and *mall* (Figure 6.4), where robots start from fixed starting locations without any initial knowledge about the structure of the environment. The cells of the test environments are labeled as 'corridor', 'small room', 'medium room', or 'big room' according to the size of the rooms they belong to. Label distributions are reported in Tables 6.2 and 6.3. The first test environment is part of the "vasche_library_floor1", taken by Radish repository [63], and is characterized mainly by the presence of small and medium rooms (as we can see from Table 6.2, the number of small and medium rooms is almost the 86% of the total number of rooms in the environment). The second one is a floor of a (real) mall, and is characterized by the presence of very big rooms. Table 6.3 shows that the number of big rooms is almost the 12% of the total number of rooms in the environment, but they occupy 41% of the total area of the environment. Some obstacles (shown as short line segments in Figure 8.1) have been added to the rooms to simulate furniture and to make the exploration task more realistic. We consider structured indoor environments because many semantic maps have been built for indoor environments and search and rescue scenarios are often indoor (like those of the Virtual Robot Competition of the RoboCup

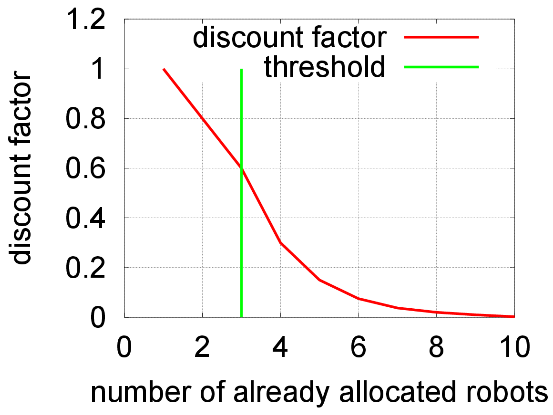


**Figure 6.3:** *Discount factor vs. the number of robots already allocated to $p$, if $i(p) = 3$.*

| Type | Number of cells | % of the environment area | Number of rooms |
|---|---|---|---|
| Corridors | 2493 | 30% | - |
| Small rooms | 756 | 9% | 21 |
| Medium rooms | 2835 | 34% | 42 |
| Big rooms | 2304 | 27% | 10 |

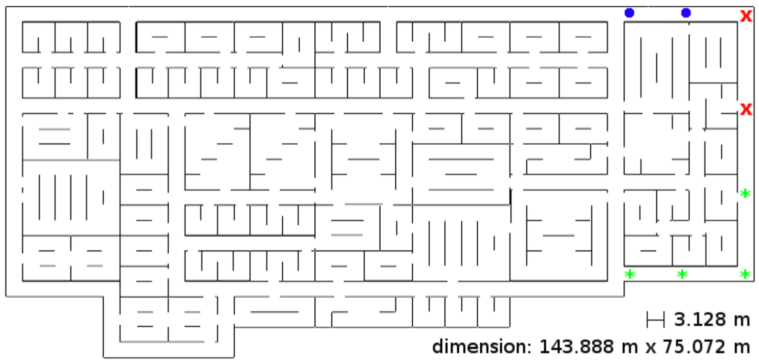
**Table 6.2:** *Number of cells, percentage of the area of the environment, and number of rooms of each semantic label (room type) for the office environment.*

Rescue Simulation League).

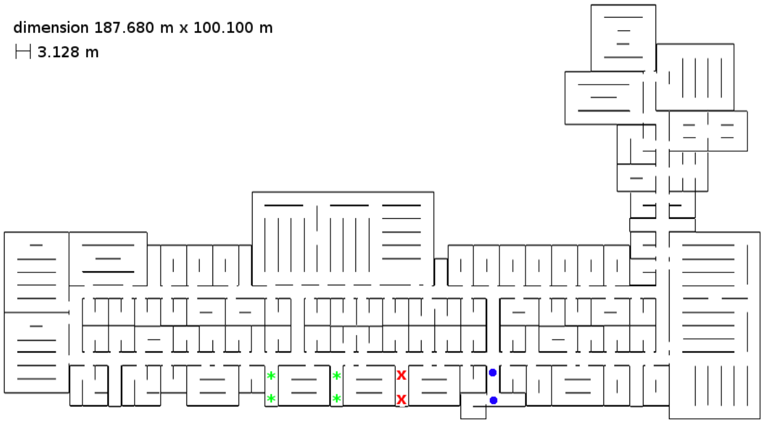
We consider teams of 4, 6, and 8 robots and two *a priori* hypotheses (assumed to be correct) on victims' location, namely victims in big rooms and in small rooms. We define a setting as an environment (office or mall), a number of robots (4, 6, or 8), an exploration strategy (D-MCDM or S-MCDM), a coordination method (SR, MRv1, or MRv2), and an hypothesis on the victims' location (big or small rooms). For each setting, we execute 10 runs of 20 minutes each.

| Type | Number of cells | % of the environment area | Number of rooms |
|---|---|---|---|
| Corridors | 1449 | 18% | - |
| Small rooms | 1332 | 16% | 37 |
| Medium rooms | 2088 | 25% | 31 |
| Big rooms | 3393 | 41% | 9 |

**Table 6.3:** *Number of cells, percentage of the area of the environment, and number of rooms of each semantic label (room type) for the mall environment.*



**(a)** *Office*



**(b)** *Mall*

**Figure 6.4:** *Test environments. Green stars represent initial positions for the robots in the settings with 4 robots, red crosses refer to the addition of two robots (6 robots), and blue points to the addition of further two robots (8 robots).*

In a search and rescue setting, the goal is to explore an initially unknown environment for finding the largest number of human victims within a short time. Assuming *a priori* knowledge about the relevant area in which victims are supposed to be, and assuming that victims are uniformly distributed in the relevant areas, the problem of maximizing the number of victims found in a given time interval is equivalent to the problem of maximizing the *amount of relevant area* covered by robots' sensors in the same interval. Thus, we assess our system performance by measuring the amount of relevant area (area of small or of big rooms, according to the victims' location hypothesis) explored, every 1 minute of exploration. We typically report data at the end of runs (after 20 minutes), but, for some settings, we report graphs of data over 20 minutes. This measure is particularly relevant in the context of search and rescue, as time is limited, and thus, we want to explore as quickly as possible the relevant parts of an environment. We report also some results about the total explored area so that it is possible to compare our proposed method with other approaches that do not consider relevant area.

We start with some preliminary experiments we performed with state-of-the-art exploration strategies. Specifically, we compared D-MCDM with other two exploration strategies, namely, a random one (Random), which performs a random selection according to a uniform distribution over the current frontiers, and one that only considers the distance as criterion (Distance), as in the work of Wurm et al. [134]. In all cases, the coordination method is SR. Simulation results confirm that, similarly to Basilico and Amigoni [16], D-MCDM performs better than the other two exploration strategies. For example, Figure 6.5 shows that, in the case of office environment, 6 robots, SR coordination method, D-MCDM outperforms Random and performs relatively better than Distance, in terms of total explored area (measured in $m^2$). This provides a justification on the choice of using D-MCDM as baseline exploration strategy for comparing our proposed exploration strategy.

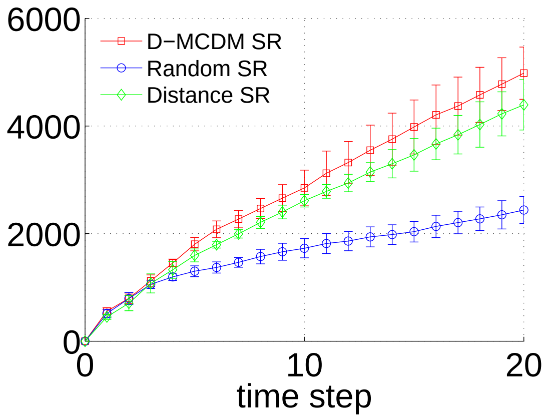


**Figure 6.5:** *Total explored area ($m^2$) over 20 minutes, in office environment, by* 6 *robots, with Random, Distance, and D-MCDM exploration strategies and SR coordination method.*

Table 6.4 reports experimental results for the office environment. The values reported in each entry are the average and the standard deviation (in parentheses) over the 10 runs of the corresponding setting.

Considering the various settings, with all the three coordination meth-

| **Office** | | **Exploration** | | | |
|---|---|---|---|---|---|
| | **Coord.** | D-MCDM (B) | S-MCDM (B) | D-MCDM (S) | S-MCDM (S) |
| **#Robots = 4** | SR | 747.3(112.5) | 1473.9(202.7) | 80.4(22.4) | 276.2(83.9) |
| | MRv1 | 953.8(111.4) | 1729.9(81.9) | 103.1(22.7) | 224.6(73.1) |
| | MRv2 | 921.6(130.3) | 1773.8(65.4) | 112.9(24.4) | 272.1(52.1) |
| **#Robots = 6** | SR | 1024.6(220.7) | 1603.2(59.1) | 127.6(38.5) | 235.7(43.1) |
| | MRv1 | 1123.6(143.6) | 1851.3(7.8) | 148.6(21.2) | 400.5(100.4) |
| | MRv2 | 1164.9(94.2) | 1856.7(36.3) | 162.5(22.2) | 429.0(83.8) |
| **#Robots = 8** | SR | 1222.6(133.0) | 1653.5(129.2) | 170.3(43.0) | 312.0(28.1) |
| | MRv1 | 1379.8(122.8) | 1877.6(62.6) | 186.2(31.7) | 496.3(101.8) |
| | MRv2 | 1284.9(144.5) | 1854.3(80.3) | 185.6(42.7) | 454.3(125.2) |

**Table 6.4:** *Results (average and standard deviation) of explored relevant area ($m^2$) for the office environment, after 20 minutes of exploration. B indicates victims most likely are in big rooms, S in small rooms.*

| **Office** | | **Exploration** | | | |
|---|---|---|---|---|---|
| | **Coord.** | D-MCDM (B) | S-MCDM (B) | D-MCDM (S) | S-MCDM (S) |
| **#Robots = 4** | SR | 2372.7(240.1) | 3424.4(316.1) | 2372.7(240.1) | 3956.6(556.5) |
| | MRv1 | 2753.4(199.9) | 3305.3(268.9) | 2753.4(199.9) | 3667.1(597.4) |
| | MRv2 | 2855.0(285.2) | 3616.0(219.4) | 2885.0(285.2) | 3683.0(441.0) |
| **#Robots = 6** | SR | 3060.3(408.1) | 3958.0(187.9) | 3060.3(408.1) | 3895.1(528.1) |
| | MRv1 | 3361.4(339.3) | 4333.1(187.0) | 3361.4(339.3) | 5094.0(194.8) |
| | MRv2 | 3536.0(158.2) | 4408.0(282.2) | 3536.0(158.2) | 5200.4(278.9) |
| **#Robots = 8** | SR | 3612.8(324.0) | 4350.6(313.5) | 3612.8(324.0) | 4677.6(526.5) |
| | MRv1 | 3996.5(267.9) | 4856.2(524.0) | 3966.5(267.9) | 5424.5(264.6) |
| | MRv2 | 3853.8(358.2) | 4786.8(486.7) | 3853.8(358.2) | 5251.2(528.2) |

**Table 6.5:** *Results (average and standard deviation) of total explored area ($m^2$) for the office environment, after 20 minutes of exploration. B indicates victims most likely are in big rooms, S in small rooms.*

| **Mall** | | **Exploration** | | | |
|---|---|---|---|---|---|
| | **Coord.** | D-MCDM (B) | S-MCDM (B) | D-MCDM (S) | S-MCDM (S) |
| **#Robots = 4** | SR | 265.4(212.9) | 1868.0(160.1) | 567.0(66.6) | 737.3(61.9) |
| | MRv1 | 517.3(164.6) | 1785.2(232.5) | 615.9(24.2) | 836.4(94.4) |
| | MRv2 | 380.3(67.7) | 1780.7(233.8) | 633.4(37.7) | 809.3(104.2) |
| **#Robots = 6** | SR | 634.4(209.3) | 1978.6(200.0) | 638.1(74.4) | 888.0(72.6) |
| | MRv1 | 574.2(73.8) | 2151.7(228.8) | 702.3(52.5) | 1018.3(93.3) |
| | MRv2 | 545.2(129.4) | 2105.3(241.4) | 701.7(18.8) | 983.9(76.7) |
| **#Robots = 8** | SR | 768.6(190.1) | 2050.0(189.3) | 708.8(56.4) | 953.3(97.4) |
| | MRv1 | 606.5(207.5) | 2304.5(161.6) | 755.9(44.0) | 1149.5(95.9) |
| | MRv2 | 540.7(139.5) | 2336.9(214.5) | 751.8(56.8) | 1046.5(80.4) |

**Table 6.6:** *Results (average and standard deviation) of explored relevant area ($m^2$) for the mall environment, after 20 minutes of exploration. B indicates victims most likely are in big rooms, S in small rooms.*

| Mall | | Exploration | | | |
|---|---|---|---|---|---|
| | **Coord.** | D-MCDM (B) | S-MCDM (B) | D-MCDM (S) | S-MCDM (S) |
| **#Robots = 4** | SR | 2480.5(463.5) | 5229.9(114.4) | 2480.5(463.5) | 2605.9(251.1) |
| | MRv1 | 2668.0(156.7) | 4501.1(541.3) | 2668.0(156.7) | 2802.9(427.3) |
| | MRv2 | 2542.8(54.3) | 4765.6(546.2) | 2542.8(54.3) | 2797.7(530.4) |
| **#Robots = 6** | SR | 3164.2(478.1) | 5222.3(508.8) | 3164.2(478.1) | 3355.1(451.2) |
| | MRv1 | 2990.1(262.1) | 5449.9(349.4) | 2990.1(262.1) | 3706.4(436.7) |
| | MRv2 | 2941.1(198.6) | 5214.6(581.0) | 2941.1(198.6) | 3536.6(363.2) |
| **#Robots = 8** | SR | 3756.9(425.0) | 5625.6(243.2) | 3756.9(425.0) | 3956.6(465.4) |
| | MRv1 | 3446.9(406.9) | 5846.5(235.7) | 3446.9(406.9) | 4255.4(406.0) |
| | MRv2 | 3326.0(324.7) | 5765.0(269.8) | 3326.0(324.7) | 3903.4(379.6) |

**Table 6.7:** *Results (average and standard deviation) of total explored area ($m^2$) for the mall environment, after 20 minutes of exploration. B indicates victims most likely are in big rooms, S in small rooms.*

ods, S-MCDM behaves better than D-MCDM, and differences are statistically significant, according to an ANOVA analysis with a threshold for significance $p$-value $< 0.05$ [101]. For example, the difference between the relevant area mapped at 20 minutes with S-MCDM and D-MCDM, in the case of victims in big rooms, with SR and 6 robots, is statistically significant ($p$-value$= 2.42 \cdot 10^{-7}$). Figure 6.6 illustrates the evolution of the explored relevant area over 20 minutes in the setting just discussed. We can observe that at the beginning the trend is almost the same for both exploration strategies. This could be explained by the fact that the 6 robots start from positions that are close to some big rooms and so also D-MCDM chooses candidate locations in big rooms. After 10 minutes, S-MCDM outperforms D-MCDM, indicating that, when there are more candidate locations in different rooms that could be selected by the robots, the benefits of using a semantic-based exploration are more evident.
Note that similar trends are also valid for the hypothesis of victims in small rooms and, also in this case, the difference between the relevant area mapped at 20 minutes with the two exploration strategies is statistically significant (e.g., with SR and 6 robots, $p$-value$= 1.34 \cdot 10^{-5}$).

For both exploration strategies, MRv1 and MRv2 appear to perform relatively better than SR, and differences are statistically significant (for instance, for MRv2 vs. SR, $p$-value$= 9.24 \cdot 10^{-10}$, with S-MCDM, considering the hypothesis of victims in big rooms and 6 robots). Figure 6.7 shows the explored relevant area considering the latter setting over 20 minutes. We can observe that MRv1 and MRv2 have similar trends and that there are some points in which their two curves intersect with the curve of SR. This can be explained by the fact that there are some initial drawbacks in sending more robots to the same candidate location, due to sensing over-

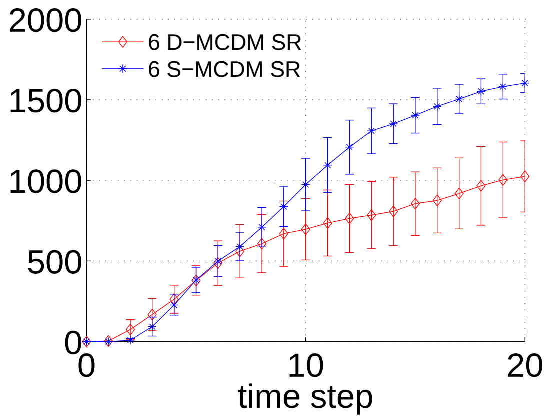


**Figure 6.6:** *Explored relevant area ($m^2$) over 20 minutes, in office environment, by 6 robots, with SR coordination method, in the case of victims in big rooms.*

laps. However, in the long term, there seems to be a benefit; indeed, after 10 minutes, curves of MRv1 and MRv2 are well above the one of SR.

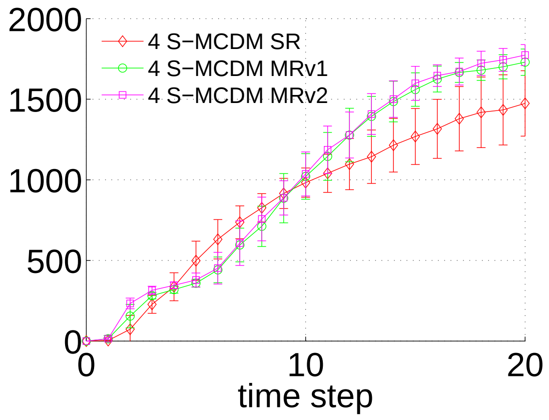


**Figure 6.7:** *Explored relevant area ($m^2$) over 20 minutes, in office environment, by 4 robots, with S-MCDM, in the case of victims in big rooms.*

Only considering $4$ robots, in the case of victims in small rooms, SR seems to have better results than MRv1 and MRv2, even if not statistically significant (e.g., in this setting, with S-MCDM, for SR vs. MRv2, $p$-value$=$ $0.80$). This similar performance of SR and MRv1/MRv2 can be explained noting that, when the number of robots is small, the exploration becomes unbalanced if more robots are assigned to the same candidate location.

Another consideration from Table 6.4 is that, as expected, increasing the number of robots, the amount of explored relevant area increases (apart from one degenerate case with SR and S-MCDM considering victims in small rooms and increasing robots from 4 to 6), even if the increase is not statistically significant. Note that the standard deviation of the results in Table 6.4 is high in the case of victims in small rooms. This could be due to the fact that, since robots should focus on small rooms, the space in which robots can move is small and, so, errors in the movement of the robots have greater influence in these experiments. Indeed, we observed in the experiments that, for example, robots spend some time to enter in a small room.

Table 6.5 shows the *total* amount of explored area (as opposite to the amount of relevant area considered so far) for the office environment. The total amount of explored area increases from D-MCDM to S-MCDM in the case of victims in big rooms. For example, with 6 robots and SR, the total amount of explored area changes from 3115.6 (367.0) m$^2$ to 3958.0 (187.9) m$^2$, with a statistically significant difference ($p$-value= $5.91 \cdot 10^{-6}$). Figure 6.8 shows the trend over 20 minutes of such setting. This performance increase could be due to the fact that robots are pushed to big rooms, where it is possible to easily explore large portions of the environment.

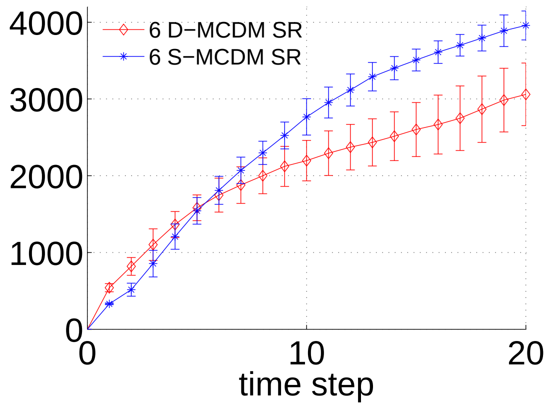


**Figure 6.8:** *Explored total area (m$^2$) over 20 minutes, in office environment, by 6 robots, with SR coordination method, in the case of victims in big rooms.*

In the case of victims in small rooms the total amount of explored area is more or less the same for D-MCDM and S-MCDM. The total amount of explored area is similar for all coordination methods. Note that the traveled distance by the robots does not change much over all the experiments (see, for example, Figure 6.9). This fact shows that the difference in the amount of (relevant or total) explored area does not depend on the fact that the robots may be stuck, but almost exclusively on the exploration strategy and the coordination methods adopted.

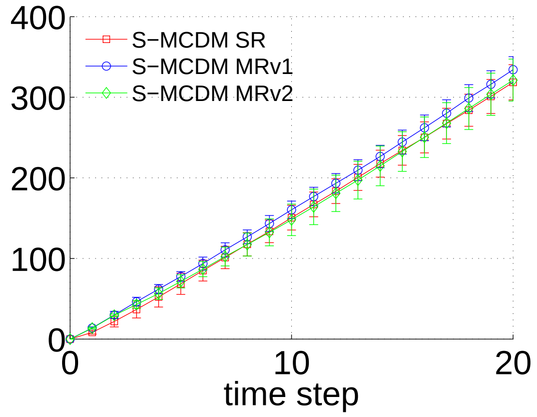


**Figure 6.9:** *Sum of the traveled distances (m) over 20 minutes, in office environment, by 8 robots, considering S-MCDM, in the case of victims in big rooms.*

The difference in the performance of the exploration strategies can be further analyzed by looking at how they evaluate candidate locations in different rooms. As explained in Section 6.1, this evaluation changes according to the semantic labels of the cells and the hypothesis on the victims locations for S-MCDM (criterion $S$), while D-MCDM evaluates candidate locations in different rooms more uniformly. Figure 6.10 illustrates this behavior in the case of 6 robots, SR coordination method, and victims most likely located in big rooms. This different evaluation of the candidate locations determines the number of assigned candidate locations in different rooms for D-MCDM and S-MCDM. Including semantic information in the exploration strategy effectively allows the robots to focus on candidate locations in the relevant areas, neglecting those in the irrelevant ones. Figure 6.11a shows that, in the case of 6 robots, SR coordination method, and victims most likely located in big rooms, the number of candidate locations in big rooms assigned to the robots using S-MCDM is greater than the same number in the case of D-MCDM. Figure 6.11b illustrates that, in the same last setting, almost no candidate locations in small rooms are assigned to the robots in the case of S-MCDM.

Tables 6.6 and 6.7 show experimental results for the mall environment and report the explored relevant area and explored total area, respectively. All the above observations hold also in this setting. The only difference is

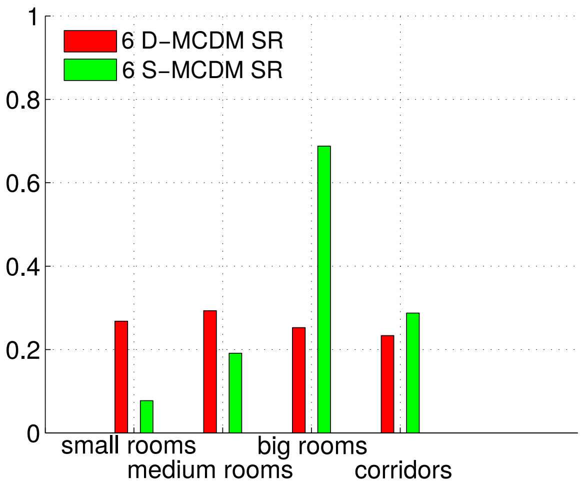


**Figure 6.10:** *Evaluation of the candidate locations (on a relative scale, average over all the candidate locations evaluated by the robots over 20 minutes) that are located in small, medium, big rooms and corridors, in office environment, with 6 robots, considering SR and the hypothesis of victims in big rooms.*

relative to the case of D-MCDM and victims in big rooms, for which the results obtained by MRv1 and MRv2 worsen with respect to SR, and only with $8$ robots the difference between SR and MRv2 is statistically significant ($p$-value$= 0.01$). This could imply that the joint use of a coordination method that uses semantic information and an exploration strategy that does not can be inefficient.

We also experimentally verified that our results are still valid varying starting locations and the number of the robots (10 or 12). For example, Figure 6.12 shows that increasing the number of robots, the explored relevant area increases, as expected. As shown in the figure, the trends for the different combinations of exploration strategy/coordination method are rather similar.

We now relax the assumption of perfect semantic information, as our system strongly relies on it. Specifically, we consider two imperfect semantic mapping modules, which make errors in assigning labels to rooms (and to cells within rooms):

1. randomly according to an error rate (0.1 or 0.2 of the number of classifications), as in the work of Stachniss et al. [121];

2. depending on the percentage of the area actually discovered. If a candidate location $p$ is located in a room, whose fraction of already explored area is less than a pre-defined threshold ($0.2$ or $0.4$), the seman-

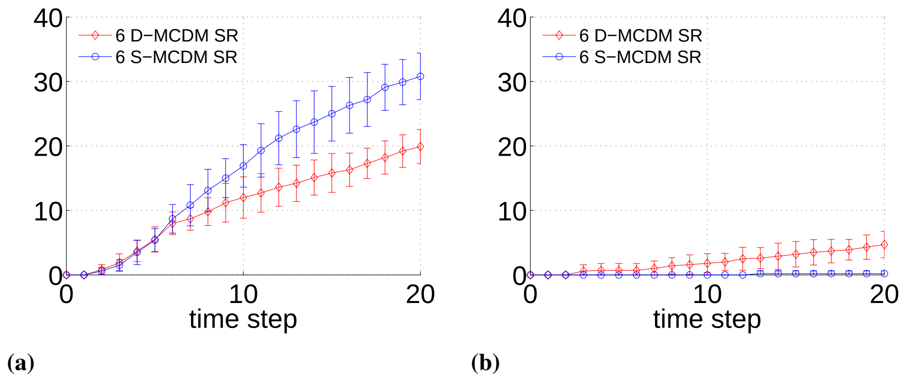


**Figure 6.11:** *The number of assigned candidate locations in big rooms (a) and in small rooms (b) over 20 minutes, in office environment, to 6 robots, considering SR and the hypothesis of victims in big rooms.*

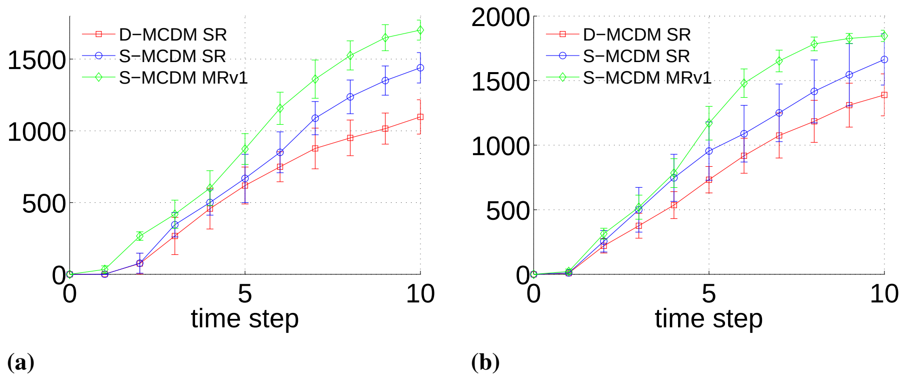
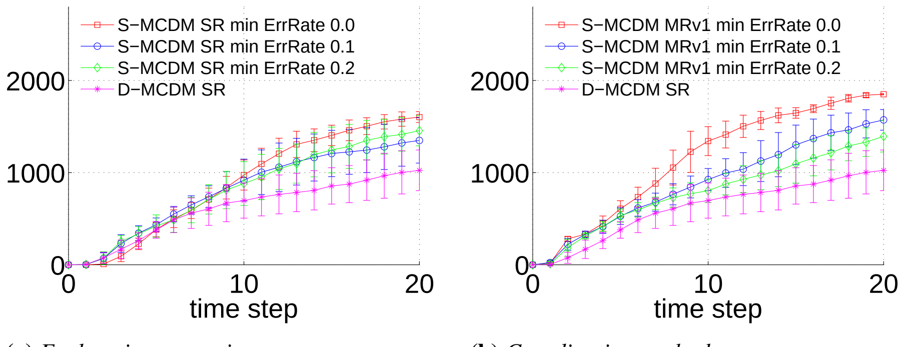


**Figure 6.12:** *The explored relevant area ($m^2$) over 10 minutes, in office environment, by 10 (a) and 12 (b) robots, with the hypothesis of victims in big rooms.*

tic mapping module classifies $p$ randomly (with uniform probability) over the semantic labels available. Otherwise, the semantic mapping module correctly classifies $p$.

Both of them, when errors are made, environment regions are classified randomly according to a uniform distribution over the possible label set. We tested the system with imperfect semantic information in the office environment, with 6 robots, coordination method SR, and victims located in big rooms. Figure 6.13 shows the amount of relevant area explored over 20 minutes, with the imperfect semantic mapping module (a). The explored relevant area diminishes compared to the case of a perfect semantic mapping module. However, the combination of our proposed exploration strategy and coordination method allows to have a better performance compared to the state-of-the-art combination of exploration strategy and coordination method (at the end of 20 minutes, this difference between S-MCDM + MRv1 and D-MCDM + SR is statistically significant with $p$-value$=1.02 \cdot 10^{-4}$). Comparing trends of the results obtained by using MRv1, we can observe that the performance degrades, when the error rate increases. This can be explained by the fact that our proposed coordination method assigns more robots to a candidate location in a big room or a corridor, but, with an imperfect oracle, the risk is to assign more robots to areas that could be explored by only one robot.

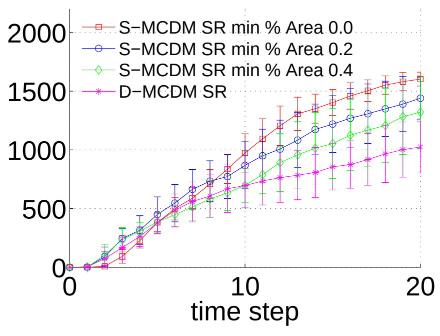
Figure 6.14a shows the amount of relevant area explored over 20 minutes, with the more realistic semantic mapping module (imperfect semantic mapping module (b)) that assigns a random label to a room if it is explored less than a threshold. The performance does not degrade very much with respect to the performance obtained by our system with perfect semantic information, and S-MCDM still performs better than D-MCDM. For example, at 20 minutes, with S-MCDM and realistic semantic mapping with threshold $0.4$, the explored relevant area is 1321.8 (310.2) m$^2$, while with D-MCDM and perfect semantic information, the explored relevant area is 1024.6 (220.7) m$^2$ ($p$-value$=0.02$). The same trend is observed considering coordination methods (see Figure 6.14b). The setting of S-MCDM and MRv1 with threshold $0.4$ is still better than D-MCDM and SR with perfect semantic information (1629.9 (120.8) m$^2$ vs. 1024.6 (220.7) m$^2$, $p$-value$=5.0 \cdot 10^{-7}$).



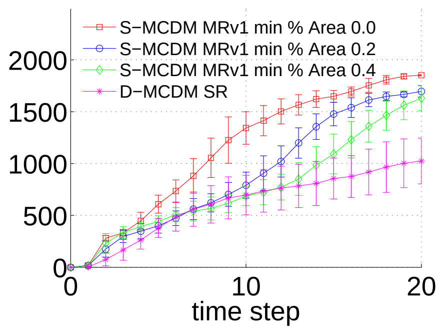
**(a)** *Exploration strategies.* **(b)** *Coordination methods.*

**Figure 6.13:** *Explored relevant area (m$^2$) over 20 minutes, in office environment, by 6 robots with random semantic mapping.*

Finally, we tested the performance of our system by setting as termination criterion a given percentage of relevant area to be mapped (instead of



**(a)** *Exploration strategies.*



**(b)** *Coordination methods.*

**Figure 6.14:** *Explored relevant area ($m^2$) over 20 minutes, in office environment, by 6 robots with realistic semantic mapping.*

the 20 minutes timeout considered before), as in the work of Wurm et al. [134]. In this case, the system performance could be evaluated according to the time spent for accomplishing the mission. This experiment was carried out on a portion of the mall environment, with 8 robots with the goal of mapping 90% of the relevant area (victims located in big rooms). Figure 6.15 shows that S-MCDM (and MRv1) terminates earlier (around 20 minutes) than the state-of-the-art combination of D-MCDM and SR (around 29 minutes).

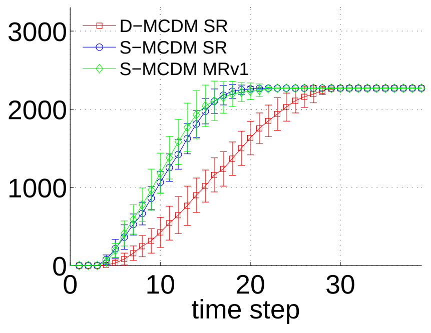


**Figure 6.15:** *Explored relevant area ($m^2$) over 20 minutes, in mall environment, by 8 robots with a different termination criterion.*

In summary, results show that our semantically-informed exploration strategy largely outperforms a state-of-the-art exploration strategy in discovering areas of interest in the office and the mall environments. This can be explained by the fact that the exploration strategies that do not consider semantic information evaluate candidate locations only according to their metric features, independently of their interest for the possible presence of victims. Another relevant result is that both MRv1 and MRv2 have better performance compared to SR. This behavior is more evident with the hypothesis of victims in big rooms, because MRv1 and MRv2 directly accelerate the exploration of big rooms, as more robots are sent to such rooms. The result is valid in the hypothesis of victims in small rooms as well but, in this case, the reason is that MRv1 and MRv2 send more robots in corridors, to which several rooms are connected and can be easily accessed. However, no statistically significant trend can be observed when comparing MRv1 and MRv2. In addition, our experimental results suggest that the coordination method has comparatively less impact on the performance than the exploration strategy. This is in line with the results obtained by our analysis in [7], for different search and rescue settings and reported later in Chapter 9. Note also that our semantically-informed approach generally performs better than traditional approaches independently of the percentage of relevant area over total area. However, with few relevant areas (e.g., big rooms in office, Figure 6.4a), the advantage in using semantic information in coordination is more evident. With many relevant areas that are easily accessible from the starting positions of the robots (e.g., small rooms in mall, Figure 6.4b), using semantically-informed coordination is less effective (robots can be simply spread using traditional approaches with good chances of visiting relevant areas). Finally, our system proved to be enough robust to errors in semantic labeling of the areas of the test environments. The proposed semantically-based multirobot exploration system contributes to one of the objectives of this dissertation, namely to improve exploration by exploiting also semantic information.

# Part III
# Evaluation

Recognizing the lack of a mature experimental methodology for autonomous mobile robots (see Section 2.3), the third part of this dissertation contributes to answering the question 'which approach performs better in a given setting?'. Also, it contributes to understand through experimental analyses how some of the factors that are, to some degree, controllable by the robotics designers impact the performance of exploration.

Specifically, we show how the approach provided in Chapter 3 that calculates the optimal offline exploration path can be used as a tool to calculate the (approximated) competitive ratio so that exploration strategies can be compared against an optimal behavior. Further, we experimentally analyze what impact the perception/decision timing has on exploration. We also quantitatively assess the relative influence of exploration strategies and coordination methods on the performance of multirobot exploration systems for search and rescue.

For performing experimental analyses, despite the importance of testing methods on real robots, we used two different simulators to obtain repeated results with a good significance level. The reason is that we searched for existing simulators and controllers, for which the timings of perception and decision can be easily changed and some coordination methods are already available. As a consequence, we selected the Stage simulator [130] for evaluating the impact of perception/decision timing and USARSim [27] for evaluating the relative influence of exploration strategies and coordination methods on the exploration performance. Also note that we selected existing controllers so that we can focus only on exploration strategies and on coordination methods (when more than one robot is employed), exploiting existing and tested methods for navigation, localization, and mapping.

The evaluation results discussed here could serve to aid robotic designers to better tune some of the parameters of the systems in order to enhance their performance.

# 7 Evaluation of upper bounds and competitive ratio

In this (short) chapter, using the simulation results presented in Section 3.4 and relative to optimal offline exploration paths, we first show that some of the (theoretical) upper bounds relative to exploration algorithms proposed in literature are usually very far from the optimal exploration paths results obtainable in specific environments. The reason lies in the definition of upper bounds: those bounds are calculated considering different types of environments, including (unrealistic) worst-case environments, like star-shape environments. Hence, looking only at upper bounds does not allow to answer the question which exploration algorithm performs better in a specific setting. Second, we compare our results with online exploration algorithms analyzed by Amigoni [2], showing how our approach can be used to compute their competitive ratio for specific environments.

As we noted in Chapter 2, different works in literature deal with the optimal exploration problem from an online perspective, assuming that the environment is initially unknown. The contributions provided by this class of algorithms can be roughly divided in two types. Theoretical results typically involve the definition of guaranteed bounds on the performance that a particular algorithm can achieve on some classes of environments. Experimental contributions, on the other hand, are typically focused on assessing the performance of exploration algorithms on the field, evaluating them on some specific test environments. We now show how our approach can provide interesting insights for both these types of results, helping to answer questions on how tight a bound is in practice or how effectively an algorithm can compete against an optimal exploring robot.

## 7.1 Upper bounds

Let us consider the bounds presented by Ghosh et al. [55] which apply to the number of steps. Authors introduce an online algorithm that, given an initial position of the robot in a polygon $P$, finds an exploration path that starts by following the boundary of $P$ and then continues over the boundary between the explored and unexplored areas, until all the polygon is covered. In such work, finite sensor range and time-discrete perception are considered in an arbitrary environment. Authors found that the upper bound of their proposed algorithm is (using authors' notation) $\text{UB}_{\#} = \frac{8 \times A(E_f)}{3 \times R^2} + \frac{\text{Perimeter}(E_f)}{R} + r + h + 1$, where $A(E_f)$ is the environment free space area, Perimeter$(E_f)$ is the environment perimeter, $R$ is the sensor range, $r$ is the number of reflex vertices, $h$ is the number of holes.

For the traveled distance, a work that provides an upper bound to the proposed algorithm is that of Gabriely and Rimon [51], which also considers finite sensor range and time-discrete perception, in an arbitrary environment. Specifically, authors extend the offline algorithm we briefly presented in Section 2.1 to the online case, in which the environment is initially unknown, by incrementally building the spanning tree according to the current knowledge of the environment. The upper bound is (using authors' notation) $\text{UB}_{\mathscr{D}} = (n+m)D$, where $n = \frac{A(E_f)}{D^2}$ is the total number of $D$-size cells (the continuous work-area is approximated by a discrete grid of D-size cells), $m = \frac{\text{Perimeter}(E_f)}{D} \leq n$ is the number of cells sharing a point with the grid boundary, and $D$ is the size of the tool. Table 7.1 shows that,

| | Perimeter (units) | Area (units$^2$) | $h$ | $r$ | # OF STEPS [55] | DISTANCE [51] | # OF STEPS | DISTANCE |
|---|---|---|---|---|---|---|---|---|
| Indoor | 1226 | 22295 | 0 | 8 | 218.9 | 1783.4 | 25.3 | 538.1 |
| Openspace | 1398 | 70944 | 1 | 5 | 549.9 | 3171.6 | 74.1 | 1686.9 |
| Obstacles | 1992 | 18280 | 14 | 72 | 308.5 | 2449.0 | 27.7 | 607.3 |

**Table 7.1:** *Environments settings for calculating upper bounds for the number of steps [55] and for traveled distance [51], with sensor range $R = 20$ ($D = 20$) units (last two columns show the costs of the solutions found by our algorithm considering $r = 20$, $e = 1$, $\mathfrak{g} = 0.95$).*

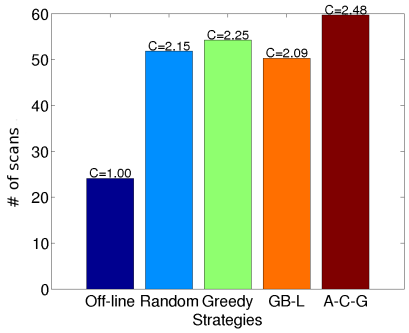
in case of both optimality criteria, the bounds provided by Ghosh et al. [55] and Gabriely and Rimon [51] are (very) loose for the realistic environments of Figure 3.9. The reason is that those bounds are calculated considering different types of environments, including (unrealistic) worst-case environments like, for example, weird star-shaped ones. Thus, having worst-case bounds does not seem to provide useful insights for establishing which algorithm performs better in a specific environment.

## 7.2 Competitive ratio

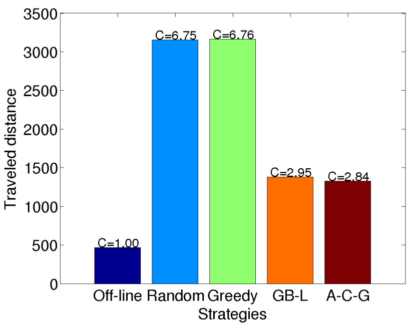
The work presented by Amigoni [2] performs experimental evaluations and comparisons of some online exploration strategies. The model assumed by the author is basically the same as the one presented in Chapter 5. Specifically, Amigoni [2] assumes to have an holonomic mobile robot that can move in a two-dimensional environment and that is equipped with two laser range scanners, so that the covered area is $360°$ and a configurable range $r$. It is assumed that the perception and decision models are time-discrete. Differently from our approach, which considers a grid map, the environment is represented with a line-segment-based map, which stores two lists of line segments: *obstacle list* that stores segments representing obstacles and *free edge list* that stores segments representing the boundaries between known and unknown portions of the environment. The candidate destination locations are generated randomly on the line segments belonging to the free edge list. The author evaluates four exploration strategies. Two of them are proposed in the literature and are called GB-L [56] and A-C-G [3] (see Equation (2.1) and Equation (2.4) in Section 2.2). Another one is called Greedy exploration strategy, which evaluates a candidate destination location $p$ only on the basis of its utility $u(p) = A(p)$, namely the expected information gain once reached $p$. The last evaluated exploration strategy acts as “bottom line” and is called Random, as it selects next destination locations randomly. These exploration strategies are tested in the same environments depicted in Figure 3.9. The considered metrics by the author are the number of steps and the traveled distance. Recall that a step (or scan) in the author’s work (and here) refers to map updates for exploration purposes and not to reading from sensors data at a given frequency. This measure provides an idea about the computational effort required to build the map of the whole environment (assuming that the localization process takes less computational time).

With the results presented in Section 3.4, we are able to calculate an approximated competitive ratio for these strategies in the three environments shown in Figure 3.9, using the approximation of the optimal exploration path returned by our approach. Below, we show the computed competitive ratio considering $r = 20$ and $\mathfrak{g} = 0.95$ in the three environments of Figure 3.9. Note that similar considerations hold also for the other values of $r$ and $\mathfrak{g}$ used in Section 3.4.

Figures 7.1a and 7.1b show the results for the indoor environment with

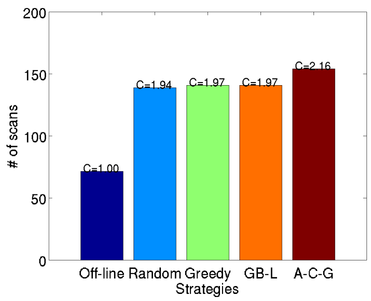


**(a)** *Number of steps*

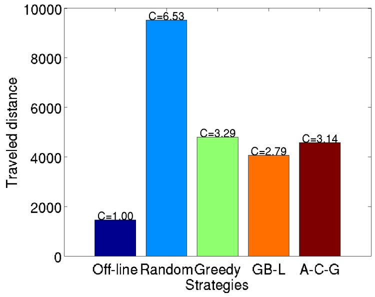


**(b)** *Traveled distance*

**Figure 7.1:** *Evaluations of some online exploration strategies in indoor environment,* $r = 20$, $\mathfrak{g} = 0.95$.



**(a)** *Number of steps*

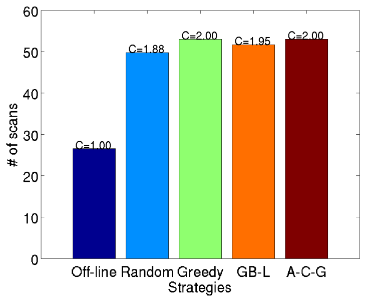


**(b)** *Traveled distance*

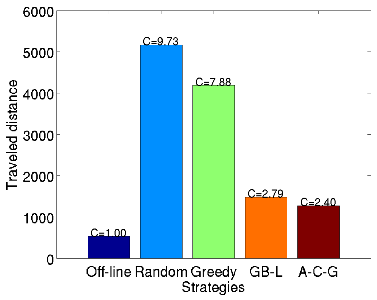
**Figure 7.2:** *Evaluations of some online exploration strategies in openspace environment,* $r = 20$, $\mathfrak{g} = 0.95$.

$r = 20$ and $\mathfrak{g} = 0.95$. It can be noted that, considering number of scans as optimality criterion, the online exploration strategies have more or less the same competitive ratio. This could be explained by the fact that the candidate locations are considered on the boundary between known and unknown part of the environment. So, at each step, the robot can always perceive some unexplored area of the environment. Also, GB-L, A-C-G, and Greedy strategies do not account for the number of steps as criterion to evaluate candidate locations. Significant difference can be observed considering traveled distance as optimality criterion. Indeed, the more sophisticated exploration strategies (GB-L and A-C-G) are almost 3-competitive, while the simpler ones (random and greedy) are almost 7-competitive.

Similar considerations hold in the other two environments, as shown in Figure 7.2 and Figure 7.3.



**(a)** *Number of steps*



**(b)** *Traveled distance*

**Figure 7.3:** *Evaluations of some online exploration strategies in obstacles environment,* $r = 20$, $\mathfrak{g} = 0.95$.

This kind of findings are some of the insights that our approach presented in Chapter 3 enables, and thus contributing to the improvement of the experimental assessment of proposed methods for exploration. Clearly for real environments that are initially unknown, our approach allows to calculate competitive ratios only *a posteriori*, that is, after the environments have been actually mapped.

# 8 Impact of perception and decision timing

As said in Section 1.1, two fundamental aspects of exploration are the integration of perceived data into the current map of the environment (Step (b)) and the decision about where to move next in a partially known environment (Step (c)). These two activities are performed by robots either in an event-based (e.g., when a destination location is reached) or in a frequency-based (i.e., periodically) fashion, as we presented in Section 1.3. Usually, as shown in Section 2.3 and Table 2.5, papers consider only a single combination of perception and decision modalities. This makes difficult to assess the impact of the perception and decision timing on the performance of an exploring robotic system. On the other hand, these parameters should be set by designers when developing exploring robot systems.

In the following, we provide a *quantitative* analysis of the impact of timing of perception and decision on the performance of a single exploring mobile robot. We mainly focus on works employing a single robot, but our considerations hold also for multi-robot exploration (e.g., [22, 126, 131]). We consider these parameters as particularly important for autonomous exploration because they are mostly related to the robot control software, while other parameters, like robot speed, sensor speed, and sensor range are, although controllable to some degree, more related to the hardware equipment. The motivation of our work is to provide designers with insights on perception and decision frequencies to develop better exploring robotic systems.

## 8.1 Simulation setting

Following the model outlined in Chapter 5, for this analysis, the robotic platform used in the simulations is a Segway-RMP robot equipped with a SICK LMS200 laser range scanner, with a maximum range of $8\,\mathrm{m}$, $180°$ FOV, and angular resolution at $1°$. The grid map has a resolution of $0.2\,\mathrm{m}$. Exploration is performed as a sequence of movements to destination locations, selected according to criteria $N = \{A, d_{PP}, o\}$ and the following exploration strategies.

The first exploration strategy we consider is based on a *weighted average* of the individual criteria (as for example in the work of Burgard et al. [22]):

$$u(p,r) = w_A A(p) - w_{d_{PP}} d_{PP}(p,r) - w_o o(p,r) \tag{8.1}$$

where $w_j$ indicates the weight associated to criterion $j$.

The second exploration strategy is called *MCDM strategy* and combines the criteria of the set $N = \{A, d_{PP}, o\}$ (see Chapter 5) using the Multi-Criteria Decision Making (MCDM) approach, presented in Section 2.2.

We use the weights reported in the following tables for weighted average (left) and MCDM (right) exploration strategies, which have been set, after some preliminary experiments, in order to obtain good performance (we also experimentally verified that slightly different values provide similar performance).

| criteria | $w_{()}$ |
|---|---|
| $A$ | 1.0 |
| $d_{PP}$ | 0.005 |
| $o$ | 0.0 |

| criteria | $\mu()$ | criteria | $\mu()$ |
|---|---|---|---|
| $A$ | 0.5 | $A, d_{PP}$ | 0.9 |
| $d_{PP}$ | 0.3 | $A, o$ | 0.7 |
| $o$ | 0.1 | $d_{PP}, o$ | 0.4 |

**Table 8.1:** *Weights of weighted average (left) and MCDM (right) exploration strategies.*

We consider these two exploration strategies because weighted average is less computationally expensive than MCDM. In this way, we can evaluate if the computational cost of making decisions has an impact on the exploration performance, given perception and decision timing.

Decisions about destination locations can be made either in an event-based or in a frequency-based fashion, in this last case with a frequency $f_d$ that can be set by the designer. During navigation, the robot perceives the surrounding environment either in an event-based or in a frequency-based fashion, also in this last case with a configurable frequency $f_p$. Recall that we refer to perception as relevant for Step (b), namely the integration of the sensor data in the map. We use six values for perception/decision frequencies $f_p$ and $f_d$, namely $0.2$, $0.4$, $0.6$, $1.0$, $2.0$ and $4.0$ Hz (smaller values make exploration too slow and larger values require too computational effort or are not feasible because of the sensor scanning frequency). We also use the combination of event-based perception and event-based decision.

The termination criterion is to map $90\%$ of the free area of the environments. This criterion is of interest, for example, in rescue applications, for which knowing the general structure of an environment is more important than exploring completely the few last posts [118].

In order to perform repeated tests under controlled conditions, we developed a ROS [105] package[1] for experimentally evaluating exploration strategies with different perception and decision timings using the Stage simulator [130]. The performed experiments consider simulated robots with realistically noisy odometry that affects both locomotion and sensing capabilities. Our package mainly depends on the following other ROS packages (some of which have been adapted or extended):

- *explore*. It implements frontier-based exploration (we added MCDM exploration strategy to the default implemented strategy, based on weighted average). Next best frontier is selected at frequency $f_d$ or when the current frontier is reached.
- *move_base*. It implements the action of movement to a destination location by following the trajectory returned by the planner.
- *gmapping*. It provides laser-based SLAM (Simultaneous Localization and Mapping) using a grid map. Map of the environment is updated when a perception is acquired (at frequency $f_p$ or when the current frontier is reached).
- *costmap_2d*. It implements a two-dimensional costmap which takes in sensor data from the world, builds an occupancy grid from the data, and assigns costs to cells.
- *stageros*. It implements two-dimensional robot simulation using Stage.

## 8.2 Simulation results

For our experimental evaluation, we consider three indoor environments: *maze*, *fort*, and *open* (Figure 8.1, the unit in the figure is 3 m), which

[1]Our developed package is publicly available at `http://sourceforge.net/projects/explorationeval`.

are all publicly available as part of the ROS *bosch_common* package, of the *fort ap_hill_07b*, and of the *acapulco_convention_centre* data sets at Radish [63], respectively. The three environments present several challenges for exploration, including dead-end corridors and intersections where the best decision about where to go next is not obvious if the environment is only partially known. Moreover, two environments are rather structured while the last one presents an empty large area.

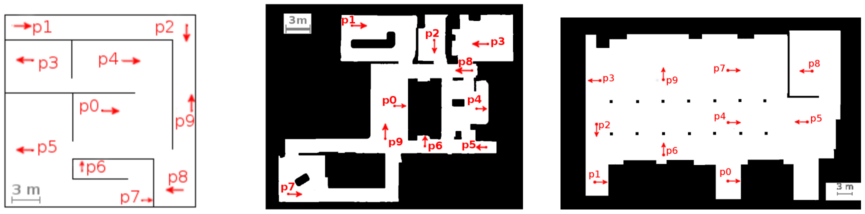


**Figure 8.1:** *Maze (left), fort (center), and open (right) environments.*

We refer to a combination of exploration strategy and perception/decision modalities as setting. For each environment, and for $10$ randomly selected initial robot poses (shown by arrows in Figure 8.1), we performed $5$ runs per setting. For the runs correctly terminated, we measured the average (over initial locations and runs, namely over $50$ values) traveled *distance* (in m) and the *time* required (in s) to meet the termination criterion.

Figures 8.2, 8.3, and 8.4 show results[2] for the maze, fort, and open environment, respectively. For each environment, the two graphs at the top are relative to weighted average exploration strategy, while the two graphs at the bottom are relative to MCDM. Moreover, for each environment and exploration strategy, the left-hand graph shows the average traveled distance with respect to the perception and decision frequencies, while the right-hand graph shows the average time to complete the exploration with respect to perception and decision frequencies. For ease of reading we discretize the values of traveled distance and exploration time in five bins (the darker the better). Some interesting trends emerge from the above results and are discussed below.

**Distance vs. perception frequency.** Given a decision frequency, there is an optimal interval of perception frequencies with respect to traveled distance. Increasing the number of perceptions in a given time interval reduces the traveled distance because the robot sees the environments at a higher pace. For example, in the maze environment, with MCDM and $f_d = 1.0$ Hz, average distance changes from $151.0\,\text{m}$ to $113.8\,\text{m}$ when $f_p$

[2]Raw data are available at `http://sourceforge.net/projects/explorationeval`.

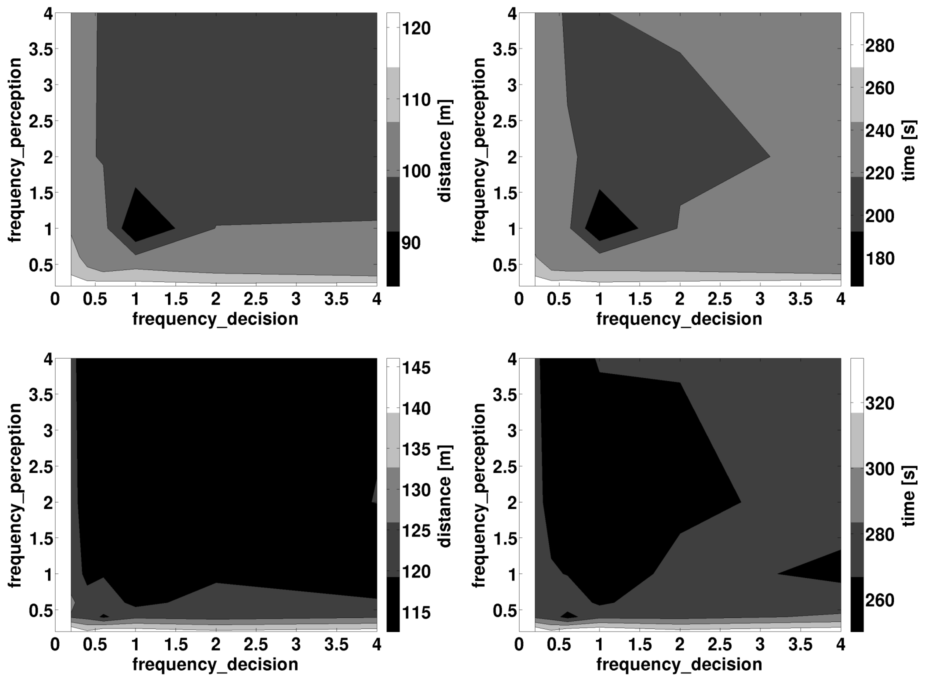


**Figure 8.2:** *Results for the maze environment, using weighted average (two graphs at the top) and MCDM (two graphs at the bottom).*

changes from $0.2\,\mathrm{Hz}$ to $1.0\,\mathrm{Hz}$. This difference is statistically significant ($p$-value=$2.37 \cdot 10^{-11}$) according to an ANOVA analysis with a threshold for significance $p$-value $< 0.05$ [101]. In all the environments, the reduction of traveled distance tends to reach a plateau when the perception frequency grows, suggesting that there is some "optimal" traveled distance for an environment [104]. For example, in the open environment, with MCDM and $f_d = 1.0$ Hz, average distance changes from $299.8\,\mathrm{m}$ to $293.3\,\mathrm{m}$ when $f_p$ changes from $2.0\,\mathrm{Hz}$ to $4.0\,\mathrm{Hz}$, a difference that is not statistically significant ($p$-value=$0.77$).

**Distance vs. decision frequency.** Given a perception frequency, the traveled distance generally decreases when the decision frequency increases. For example, in the maze environment, with MCDM and $f_p = 1.0$ Hz, average distance changes from $123.3\,\mathrm{m}$ to $113.8\,\mathrm{m}$ when $f_d$ changes from $0.2\,\mathrm{Hz}$ to $1.0\,\mathrm{Hz}$ ($p$-value=$0.044$). In a way, this is an expected behavior, because the more frequently decisions are revised, the better their outcome. Moreover, at a deeper level of analysis, this result also shows that the robot have not any "schizophrenic" behavior, namely it does not change destina-

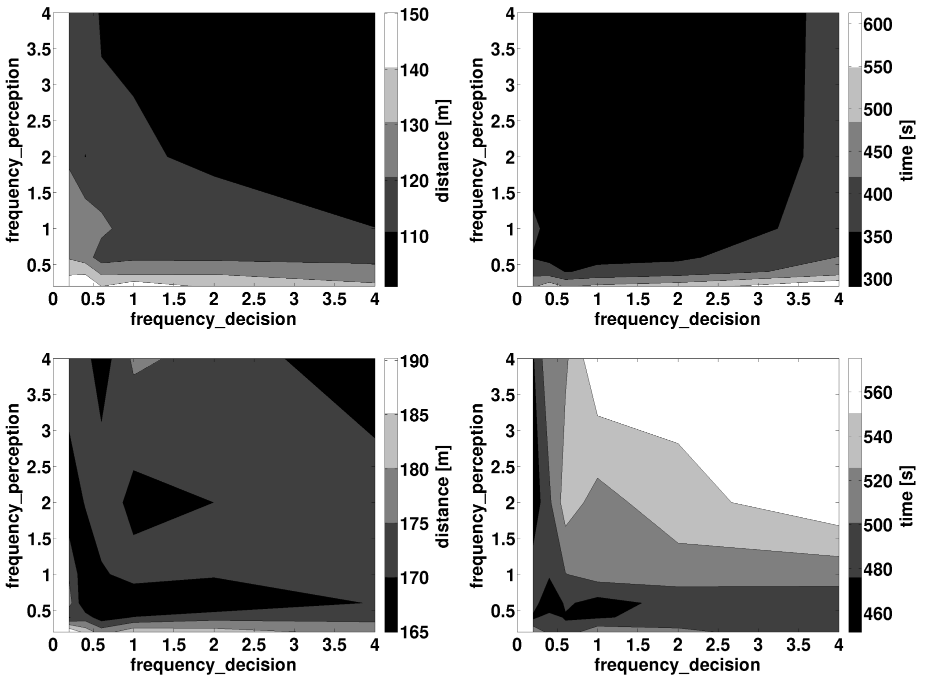


**Figure 8.3:** *Results for the fort environment, using weighted average (two graphs at the top) and MCDM (two graphs at the bottom).*

tion location every time a new decision is taken, at least in the maze and fort environments. Indeed, if that was the case, the distance would have shown an increase with growing decision frequency. In other words, the two exploration strategies we tested have the nice property of estimating enough accurately the goodness of a destination location in the maze and fort environments and this estimate is usually not changed if more data about the environment is collected. In the open environment and considering MCDM (Figure 8.4, the graph at the bottom left), the graph shows that there is a region in which increasing decision frequency can worsen the traveled distance, even if this is not statistically significant. For example, with $f_p = 2.0$ Hz, average distance changes from $263.9$ m to $297.6$ m when $f_d$ changes from $2.0$ Hz to $4.0$ Hz ($p$-value=$0.93$).

**Time vs. perception frequency.** When the perception frequency increases, the robot collects data about the environment at a higher rate and the exploration time decreases. For example, in the maze environment, with weighted average and $f_d = 1.0$ Hz, average time changes from $306.8$ s to $233.9$ s when $f_p$ changes from $0.2$ Hz to $1.0$ Hz ($p$-value=$1.34 \cdot 10^{-34}$).

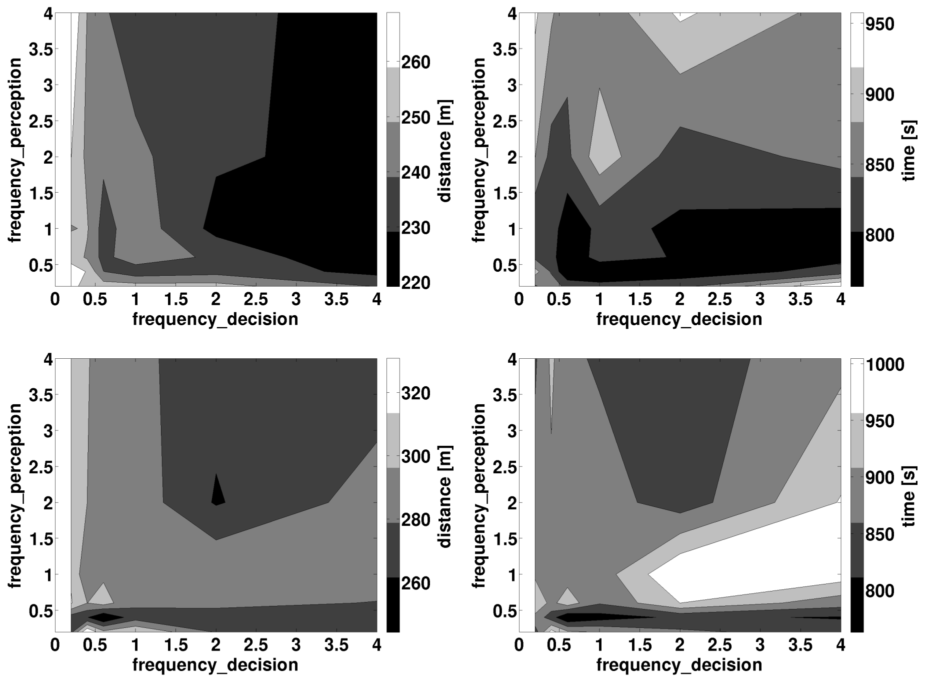


**Figure 8.4:** *Results for the open environment, using weighted average (two graphs at the top) and MCDM (two graphs at the bottom).*

However, in some settings, when the perception frequency becomes too high, the robot spends a significant amount of time in updating the map and the exploration time slightly increases even if not significantly. For example, in the open environment, with weighted average and $f_d = 2.0$ Hz, average time changes from $804.2$ s to $966.2$ s when $f_p$ changes from $1.0$ Hz to $4.0$ Hz ($p$-value=$0.002$).

**Time vs. decision frequency.** A similar behavior can be observed when looking at the impact of the decision frequency on the exploration time. With a very high decision frequency, the robot spends time in decision making (i.e., in evaluating the candidate destination locations) and the exploration time tends to increase with MCDM. This is more evident in the fort environment, which is more complicated than the maze environment, in which the exploration path is almost fixed. For example, in fort environment, with weighted average and $f_p = 2.0$ Hz, when $f_d$ changes from $1.0$ Hz to $4.0$ Hz average time changes from $326.6$ s to $385.4$ s ($p$-value=$8.06 \cdot 10^{-4}$), whereas, in maze environment, it changes from $225.8$ s to $234.8$ s ($p$-value=$0.034$). With weighted average, instead, the increase of the decision frequency does not seem to affect too much the exploration time. This could be explained by the fact that MCDM exploration strategy requires more computational effort than weighted average exploration strategy. In the open environment, exploration time has different trends according to different perception frequencies for both exploration strategies. With weighted average, it seems that increasing the decision frequency does not affect too much the exploration time. Instead, with MCDM, for low perception frequencies, high decision frequencies can worsen the exploration time, while for high perception frequencies, settings with high decision frequencies obtain results similar to those with low decision frequencies. These results in the open environment suggest that decision making in unstructured environments needs an updated map to reliably choose a candidate location.

**Distance and time vs. perception and decision frequencies.** Looking at the impact of combined frequencies on traveled distance and time, it emerges that, in general, there is an interval of frequencies that guarantee the best performance. However, there are some differences related to exploration strategies and environments, as discussed below.

**Weighted average and MCDM exploration strategies vs. environments.** With weighted average, the traveled distance and the exploration time tend to reach a wide plateau, at almost the optimal "height", when the perception and decision frequencies increase. With MCDM, both the optimal traveled distance and the exploration time are obtained for a narrower interval of frequency values, in the fort and open environment. For example, in open environment, an optimal combination of decision and perception frequencies seems to be $f_p = 0.4$ Hz and $f_d = 0.6$ Hz, with average distance of $244.3$ m and average time of $763.0$ s. For small changes in the values of the frequencies, results worsen, especially regarding exploration time. This can be explained by the higher computational effort required by MCDM, degrading the performance when decision frequency increases too much. In the open environment, the optimal traveled distance and the exploration time are in a narrower interval of frequency values compared to the case of maze and fort environments, even for the weighted average. This could be explained by considering that, in structured environments, the movements of the robot are "forced" by the presence of walls, lessening the impact of chosen frequencies values, while, in unstructured environments, the robot can possibly go in any direction and so carefully tuning frequencies of perception and decision is more critical.

Finally, we present some results about event-based perception and decision. Figure 8.5 shows results relative to the maze environment for event-based perception and decision, for frequency-based perception ($f_p = 5.0\,\text{Hz}$) and event-based decision, and for frequency-based perception and decision ($f_p = 1.0$ Hz and $f_d = 1.0$ Hz). The left-hand graph shows the average traveled distance, while the right-hand graph shows the average exploration time for each of the above combinations.

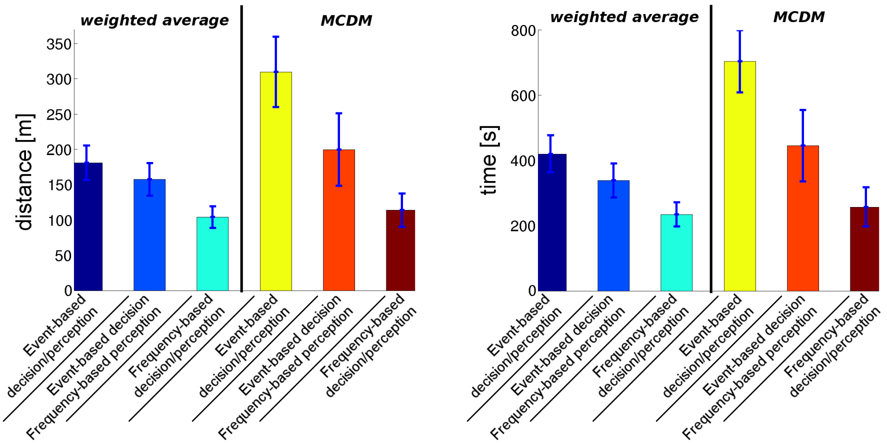


**Figure 8.5:** *Results for the maze environment.*

Results confirm, as expected, that better performance is obtained with frequency-based perception and decision. For example, for MCDM, changing from event-based to frequency-based perception and decision, average distance changes from $309.4\,\text{m}$ to $113.8\,\text{m}$ ($p$-value=$4.65 \cdot 10^{-44}$) and average time changes from $703.0\,\text{s}$ to $256.6\,\text{s}$ ($p$-value=$5.51 \cdot 10^{-48}$). Note that MCDM performs worse than weighted average. This could be explained since the maze environment is rather simple and does not require any complex exploration strategy.

The experimental analysis presented above confirms the intuitive idea that the best performance is obtained with fast-paced perceptions and decisions, but also suggest some trade-offs for the values of perception and decision frequencies in some settings. This quantitative analysis provides some insights to robotic designers on how the frequencies of decisions and perceptions should be set, especially when robots have low-power capability on-board. Also, this experimental analysis remarks the good practice on the fact that parameters should be reported in the descriptions of the experiments, so that they can be easily reproducible by other researchers and the related results can be thoroughly analyzed, as some parameters could really affect the performance of the system.

# 9 Impact of exploration strategies vs. coordination methods

Prior work evaluates exploration strategies and coordination methods mostly in a separated way, making it difficult to assess their relative effects on exploration, as shown in Section 2.3. In this chapter, we contribute to fill this gap. First, we evaluate, relatively to some coordination methods, the exploration strategies proposed by Visser and Slamet [131] and Basilico and Amigoni [16], as representative samples of *ad hoc* and theoretically-grounded exploration strategies, respectively. Also, we evaluate, relatively to some exploration strategies, some variants of the coordination method employed by Visser and Slamet [131], which produces the same allocation of the market-based coordination method of Simmons et al. [118]. Our results complement those of Zlot et al. [140], by considering more complex ways for generating the locations allocated to robots. We selected a search and rescue application for our experimental assessment, because there is an international competition, namely the RoboCup Rescue Simulation League Virtual Robot Competition[1], which provides a simulated common ground (e.g., metrics and software tools) for assessing the performance of exploring multirobot systems, enabling comparison and reproduction of results. We contribute to answer the following question: With limited computing or time resources, should developers spend more efforts on developing an effective exploration strategy or coordination method?

[1]http://www.robocuprescue.org/virtualsim.html

## 9.1 Simulation setting

In this section, instantiating some of the dimensions presented in Section 1.3, we describe the search and rescue setting in which we investigated the relative impact of exploration strategies and coordination methods on performance of exploring multirobot systems. In our setting, the goal is to explore an initially unknown indoor environment for finding the largest number of human victims within a given time. Assuming no *a priori* knowledge about the possible locations of the victims, the problem of maximizing the number of victims found in a given time interval is equivalent to the problem of maximizing the amount of area covered by robots' sensors in the same interval (the optimality criterion we evaluate). The termination criterion is determined by the time interval of 15 minutes. We first describe the adopted simulation environment and robot controller. Then, we describe the exploration strategies and the coordination methods we consider.

In order to perform repeated tests under controlled conditions, we use a robot simulator. We selected USARSim [27] because it is a high fidelity 3D robot simulator and it is employed in the Virtual Robot Competition.

From an analysis based on availability of code and performance obtained in the Virtual Robot Competition, we selected the controller developed by the Amsterdam and Oxford Universities (Amsterdam Oxford Joint Rescue Forces, AOJRF[2]) for the 2009 competition [132]. The main reason for using an existing controller is that we can focus only on the exploration strategies and on the coordination methods, exploiting existing and tested methods for navigation, localization, and mapping. The controller manages a team of robots. The model of the robot is basically the same as presented in Chapter 5 with some extensions: three occupancy grids are used to represent the knowledge about the environment. The first one is obtained with a small-range (3 m) scanner and constitutes the *safe area*, i.e., the area where the robots can safely move. The second one is obtained from maximum-range scans (20 m) and constitutes the *free area*, i.e., the area which is believed to be free but not yet safe. Moreover, a representation of the *clear area* is maintained as a subset of the safe area that has been checked for the presence of victims (this task is accomplished with simulated sensors for victim detection). The set of candidate locations is computed on the boundaries between safe and free regions.

The set of criteria considered is $N = \{A, P, d, b\}$ (see Chapter 5). Note that the criterion $d$, as shown below, can be computed using $d_{L_2}$ or $d_{PP}$. We define two exploration strategies. The first one is a slight variation of the

[2]`http://www.jointrescueforces.eu/`

strategy proposed by Visser and Slamet [131] and is called *AOJRF strategy*. It integrates the above criteria in an *ad hoc* utility function:

$$u(p, r) = \frac{A(p)P(p)}{d(p, r)^{b(p,r)}}. \tag{9.1}$$

The second exploration strategy is again *MCDM strategy* and combines the criteria of the set $N = \{A, P, d, b\}$ using the Multi-Criteria Decision Making (MCDM) approach. The same consideration about the evaluation criterion $d$ for the AOJRF strategy holds also for MCDM. We use the weights reported in the following table, which have been manually set in order to obtain good performance (according to the guidelines of Basilico and Amigoni [16]).

| criteria | $\mu()$ |
|---|---|
| $A$ | 0.4 |
| $d$ | 0.3 |
| $P$ | 0.05 |
| $b$ | 0.25 |

| criteria | $\mu()$ |
|---|---|
| $A, d$ | 0.75 |
| $A, P$ | 0.55 |
| $A, b$ | 0.55 |
| $d, P$ | 0.4 |
| $d, b$ | 0.32 |
| $P, b$ | 0.28 |

| criteria | $\mu()$ |
|---|---|
| $A, d, P$ | 0.9 |
| $A, d, b$ | 0.8 |
| $A, P, b$ | 0.85 |
| $d, P, b$ | 0.4 |

**Table 9.1:** *Weights used for the MCDM strategy.*

The two exploration strategies have been selected because they are representative of the two main classes of strategies that have been proposed for exploration of unknown environments (see Section 2.2). In particular, the AOJRF strategy represents *ad hoc* strategies, while the MDCM strategy represents more theoretically-grounded strategies.

While exploration strategies evaluate the goodness of a candidate location $p$ for a robot $r$, coordination methods are used to assign candidate locations to robots. We define three coordination methods for allocating candidate locations to robots. They start from a set of candidate locations (generated as discussed in Chapter 5 on the boundary between the safe and the clear area) and a set of robots, and their goal is to assign a location to each robot.

The first coordination method, shown in Algorithm 9.1, is executed by each robot independently, knowing (from the base station) the current map and the positions of the other robots, and is derived directly from the work of Visser and Slamet [131].
This first coordination method is called *AOJRF original coordination*. The reason behind the utility update of Step 4 of Algorithm 9.1 is that com-

```
1 compute the global utility $u(p, r)$ of allocating each candidate $p$ to each robot $r$
  (using Equation (9.1) or Equation (2.8)) where $d(p, r)$ is calculated using the
  Euclidean distance $d_{L_2}$ (namely using an underestimate of the real distance);
2 while $\exists$ robot $r$ not allocated and candidate location $p$ do
3     find the pair $(p^*, r^*)$ such that the previously computed utility is maximum,
      $(p^*, r^*) = \arg\max_{p,r} u(p, r)$;
4     re-compute the distance between $p^*$ and $r^*$ using $d_{PP}$ with the path planner
      (namely considering the real distance) and update the utility of $(p^*, r^*)$ using
      such exact value instead of the Euclidean distance;
5     if $(p^*, r^*)$ is still the best allocation then
6         allocate location $p^*$ to robot $r^*$;
7         eliminate robot $r^*$ and candidate $p^*$;
      end
  end
```

**Algorithm 9.1:** AOJRF original coordination method.

puting $d_{PP}$ requires a considerable amount of time. Calculating it for all the candidate locations and all robots would be not affordable in the rescue competition, since a maximum exploration time is enforced. Although in pathological cases all pairs $(p, r)$ could be re-evaluated using the above algorithm, in practice this is done only for few of them. Note that, being $d_{L_2}$ an underestimate of the real distance to be traveled and being Equation (9.1) and Equation (2.8) monotonically decreasing with $d$, the method is guaranteed to select the best pair $(p^*, r^*)$ according to $u()$ calculated with $d_{PP}$.

The AOJRF original coordination method produces the same results of the market-based mechanism proposed by Simmons et al. [118] and applied considering Equation (9.1) or Equation (2.8) to calculate utilities for bids. Both methods first select the pair $(p^*, r^*)$ with the largest utility $u()$, then, among the pairs left after elimination of those involving $p^*$ and $r^*$, they select the pair $(p^{**}, r^{**})$ with the largest utility, until a candidate destination location is allocated to each robot.

The second coordination method, called *AOJRF simplified coordination*, is similar to the previous one, but does not re-compute the distance in the Step 4 of Algorithm 9.1. It selects the best pair $(p^*, r^*)$ only on the basis of the Euclidean distance.

In the third coordination method, called *no coordination*, each robot selfishly selects its best candidate location, without considering the presence of other robots. This means that Steps 1-6 of the AOJRF original coordination method are performed only for one robot $r^*$ (the robot that is running the method) and that Step 7 is skipped. Note, however, that Step 4 is executed and distance re-computed.

The three coordination methods are in decreasing order of "optimality" in allocating locations to the robots, with the AOJRF original coordination method producing the best allocation and the no coordination method the worst. The last two methods can end up with sub-optimal allocations in which a robot is assigned a location that is supposed to be close but is actually far (AOJRF simplified coordination method) or in which two robots are assigned the same location (no coordination method).

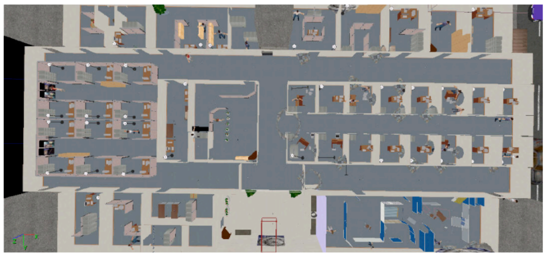
To have a baseline in comparing the results, we consider a *random coordination* method that randomly assigns robots to candidate locations, without evaluating them. We expect this random method to perform worse than other combinations of exploration strategies and coordination methods.

## 9.2 Simulation results

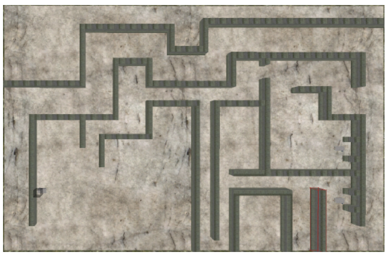
We consider teams of two and three robots (plus the base station) deployed in the "DM-compWorldDay4b_250" and "DM-VMAC1" environments[1], called *office* and *open* environments, respectively (see Figure 9.1). Both the environments are indoor with the office environment (about $800\,\mathrm{m}^2$) presenting an intricate cluttered structure and the open environment (about $1300\,\mathrm{m}^2$) presenting more open spaces. We define a setting as an environment, a number of robots, an exploration strategy, and a coordination method. For each setting, we execute $5$ runs (with randomly selected starting locations for the mobile robots such that they are separated by about $20$ meters) of $15$ minutes each. We assess performance by measuring the amount of free, safe, and clear area every $30$ seconds of the exploration. We report only data on safe area at the end of runs, as free area is less significant and clear area is similar to the safe area. Of course, the larger the mapped safe area within $15$ minutes, the better the performance. Under the assumption that victims are uniformly spread in the environment, this metric is basically equivalent to the metric that counts the number of victims found. Experiments have been run in real-time as in the competition, to realistically account for time spent in movements and in computation.

Table 9.2a shows results for the office environment. As said, every point is the average between $5$ runs. Looking at Figure 9.2, it can be noted that in the first part of the exploration, and for all the coordination methods, the curves associated to the two exploration strategies are very similar. They start to differentiate after about $5$ minutes from the start. This is because at the beginning of the exploration the selection of candidate locations is

[1]Maps can be found at `https://sourceforge.net/projects/usarsim/files/Maps/3.31/`.



**(a)** *Office*

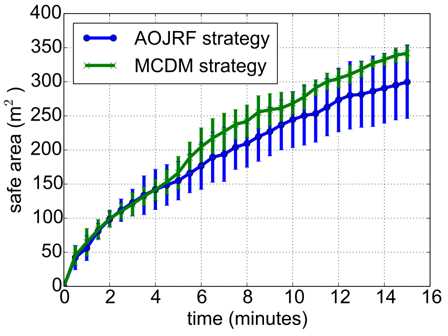


**(b)** *Open*

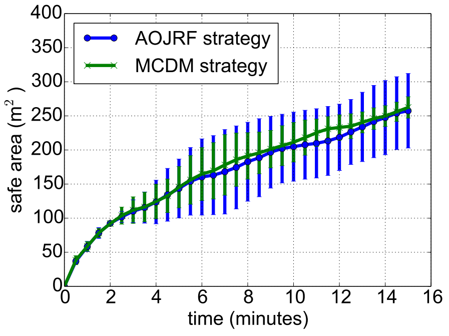
**Figure 9.1:** *The office environment (a) and the open environment (b).*

rather straightforward (the environment is largely unknown and all choices are almost equally good in increasing the mapped safe area), while it becomes more challenging as exploration continues, evidencing differences between the exploration strategies. With all the first three coordination methods, the MCDM strategy seems to behave better than the AOJRF strategy, although differences are not statistically significant, according to an ANOVA analysis with a threshold for significance $p$-value $< 0.05$ [101]. The difference between the safe area mapped at the end of the $15$ minutes is more evident with the AOJRF original coordination method. Conversely, the difference between the two exploration strategies is less evident with the AOJRF simplified coordination method. These results can be explained by saying that MCDM better exploits the more precise information used with the AOJRF original coordination method (a precise distance value obtained with path planning procedures instead of an approximate Euclidean distance value). Multirobot exploration introduces some benefits, as shown by the settings with three robots that consistently outperform those with two robots, although the difference is not statistically significant in most of the cases (e.g., MCDM strategy, AOJRF original coordination, 2 and 3 robots, $p$-value$= 0.17$). Finally, the random method has, as expected, the worst performance (we tested it only with two robots). Note also that, in Figure 9.2d, there is a single curve because, being candidate locations randomly assigned to robots, no exploration strategy is used to evaluate these locations.

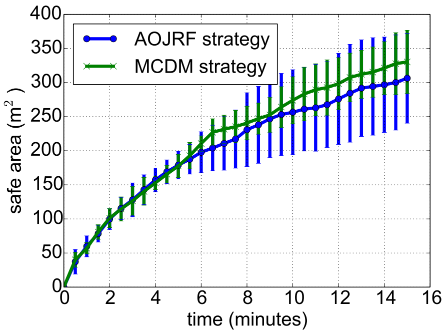
As can be noted in Figure 9.3, the performance of the first three coordination methods are very similar at the beginning of the exploration and differ as the exploration proceeds. For both the exploration strategies, we can clearly identify three behaviors. The performance of the AOJRF original coordination method and that of the method without coordination are very similar and better than that of the AOJRF simplified coordination method.



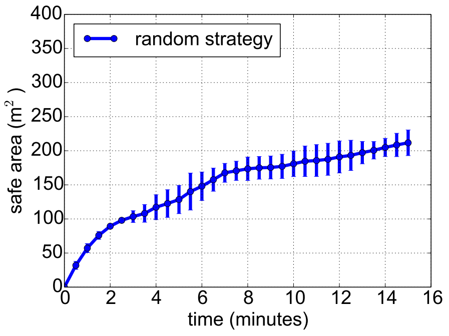
**(a)** *AOJRF original coordination method*



**(b)** *AOJRF simplified coordination method*



**(c)** *no coordination*



**(d)** *random*

**Figure 9.2:** *Performance of exploration strategies with respect to coordination methods in the office environment.*

| | **2 robots** | | **3 robots** | |
|---|---|---|---|---|
| | AOJRF strategy | MCDM strategy | AOJRF strategy | MCDM strategy |
| AOJRF original coordination | 299.77(53.60) | 341.95(12.54) | 341.58(98.62) | 387.41(66.67) |
| AOJRF simplified coordination | 257.53(54.65) | 262.43(15.62) | 320.40(63.71) | 325.14(42.21) |
| no coordination | 306.36(65.91) | 330.27(46.38) | 332.58(42.03) | 374.28(40.31) |
| random | 211.68(18.86) | | | |

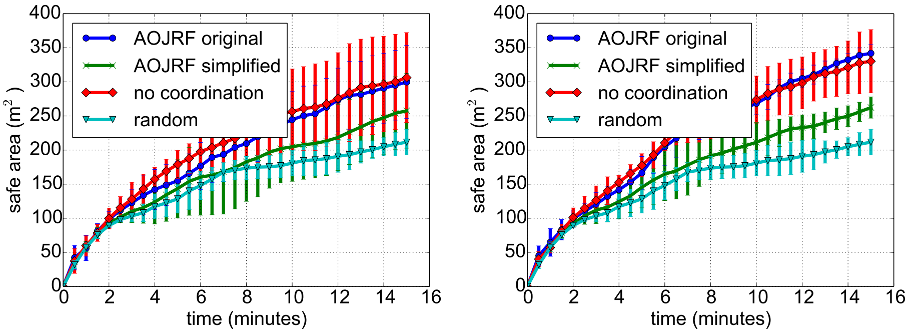
**(a)** *office environment*

| | **2 robots** | | **3 robots** | |
|---|---|---|---|---|
| | AOJRF strategy | MCDM strategy | AOJRF strategy | MCDM strategy |
| AOJRF original coordination | 430.18(78.86) | 498.45(51.12) | 483.46(130.14) | 511.83(118.35) |
| AOJRF simplified coordination | 586.77(72.16) | 678.27(48.77) | 673.48.77(85.61) | 690.16(36.69) |
| no coordination | 356.92(65.97) | 425.05(99.01) | 458.55(80.30) | 498.08(81.03) |
| random | 472.71(115.48) | | | |

**(b)** *open environment*

**Table 9.2:** *Average safe area (and standard deviation) mapped after* 15 *minutes (units are* $\mathrm{m}^2$*).*

The difference between the safe area mapped at $15$ minutes with the AOJRF original coordination and with the AOJRF simplified coordination methods is statistically significant for the MCDM strategy ($p$-value$= 2.05 \cdot 10^{-5}$ for two robots and $p$-value$= 0.04321$ for three robots), but not for the AOJRF strategy ($p$-value$= 0.25$ for two robots and $p$-value$= 0.6972$ for three robots). Similarly, the difference between the no coordination and the AOJRF simplified coordination methods is statistically significant for the MCDM strategy ($p$-value$= 0.0147$ for two robots and $p$-value$= 0.0485$ for three robots), but not for the AOJRF strategy ($p$-value$= 0.23797$ for two robots and $p$-value$= 0.7304$ for three robots).



**(a)** *AOJRF strategy* **(b)** *MCDM strategy*

**Figure 9.3:** *Performance of coordination methods with respect to exploration strategies.*

Table 9.2b shows the results for the open environment. Also in this case, the MCDM strategy seems to behave better than the AOJRF strategy with all the first three coordination methods, although differences are not statistically significant. The difference between the safe area mapped at the end of the $15$ minutes is more evident with the no coordination method, suggesting that a theoretically-grounded exploration strategy like MCDM can be more effective in limiting the problems of uncoordinated robots in the open environment.

In the open environment, the AOJRF simplified coordination method outperforms the other methods. The difference between the safe area mapped at $15$ minutes with the AOJRF simplified coordination and with the AOJRF original coordination methods is statistically significant both for the MCDM strategy ($p$-value$= 5.00 \cdot 10^{-4}$ for two robots and $p$-value$= 0.0123$ for three robots) and for the AOJRF strategy ($p$-value$= 0.0113$ for two robots and $p$-value$= 0.02594$ for three robots). Similarly, the difference between the AOJRF simplified coordination and the no coordination methods is statistically significant both for the MCDM strategy ($p$-value$= 9.00 \cdot 10^{-4}$ for two robots and $p$-value$= 1.31 \cdot 10^{-3}$ for three robots) and for the AOJRF strategy ($p$-value$= 8.00 \cdot 10^{-4}$ for two robots and $p$-value$= 3.5 \cdot 10^{-3}$ for three robots).

The results for the office environment are rather surprising: coordinately allocating tasks to robots and allocating tasks without any coordination lead to the same performance. Although the initial separation of robots could help to decompose the problem, this observation can be explained by saying that what is predominantly important in exploring the highly structured office environment is the quality of the information used to evaluate the candidate locations (like the distance returned by path planning procedures instead of the Euclidean distance). This result does not contradict previous results that concluded that coordinated robots perform better than uncoordinated robots (see Section 2.2). It seems rather to complement previous works, which considered much simpler exploration strategies than those used in this chapter. The use of exploration strategies, like MCDM and AOJRF strategies, that efficiently exploit good quality information to select observation locations effectively balances computational effort and accuracy of information. Indeed, although obtaining more accurate information (i.e., planning a path between the current location of the robot and the candidate location) requires more time and could represent a problem with the $15$ minutes deadline, the resulting selection of a good observation location has a global benefit in highly structured environments.

The results for the open environment suggest that coordination becomes more important when the environment is less structured. This can be explained by noting that, in the office environment, robots can choose from many candidate locations and the intricate structure of the environment "pushes" robots to spread, while, in the open environment, the number of candidate locations is smaller and robots need to be coordinated to effectively spread across the environment and map it. Accordingly, in the open environment, the worst performance is obtained with the no coordination method, which is outperformed also by the random method, suggesting that assigning candidate locations randomly to robots is more effective than letting robots independently choosing their best candidate locations. In the open environment, the quality of information seems not so important (AOJRF simplified coordination method using Euclidean distance outperforms AOJRF original coordination method using distance returned by path planning procedures), mainly because obtaining accurate information requires some efforts, thus leaving less time to exploration, which can be performed very quickly in uncluttered open environments.

In conclusion, we can say that the best performance is obtained for the setting with the MCDM strategy and the AOJRF original coordination method (or with the no coordination method), namely with the setting that uses the most precise information (AOJRF original coordination or no coordination) and combines it in the most effective way (MCDM strategy). The quality of information used to make decisions appears to have a stronger influence on the performance of exploration than the coordination of the robots. This experimental analysis could provide some insights on what robotic designers should focus on when designing a robotic exploration system. Also, it could highlight the fact that, as good experimental practice, the analysis of a component (e.g., exploration strategy) should be isolated to other components (e.g., coordination method), so that it becomes clearer what factors really impact on the exploration process.

# Part IV
# Use of some of the contributions

In this part, we show how the search-based approach introduced in Chapter 3 to find the optimal exploration strategy in an environment can be employed in other contexts to explore abstract state spaces.

Specifically, we select the domain of pursuit-evasion, where a pursuer tries to capture an evader, which, in turn, tries to actively escape. The models used in this context are of interest in robotics applications, because they can represent situations like those in patrolling and search and rescue. In the following, we show an approach to find a solution to a version of the pursuit-evasion problem and, in particular, to find the optimal pursuing strategy by exploring the belief space.

# 10 Exploration of an abstract state space

Pursuit-evasion games are of fundamental importance to model several robotics applications such as surveillance and search and rescue [32]. Typically, these games involve two players: a pursuer and an evader. The pursuer (or sometimes a team of pursuers) tries to capture the evader. Due to its practical importance, there has been significant interest in studying and solving such games in complex environments and with realistic assumptions regarding players' sensing capabilities.

The literature on pursuit-evasion can be divided into two main categories with respect to the environment: discrete space where the environment is topologically represented as a graph (e.g., [99]), and continuous space, where the game happens in a geometric space (e.g., [113]). Furthermore, various assumptions have been made about the capabilities of the players: evader arbitrarily faster than pursuer (e.g., [59]) vs. same bounded velocity for both players (e.g., [125]); full visibility for both players (e.g., [18]) vs. limited and full visibility for pursuer and evader, respectively (e.g., [53]). Moreover, different variants of the goal of the game have been considered. For example, a typical goal is to physically capture the evader, namely the position of the evader should coincide or be within a certain distance from the position of the pursuer (e.g., [18]). Other works, like that of [59], consider a game in which the goal is just to find the evader. Another game considered is the one whose goal is to maintain visibility of the evader over time (e.g., [90]).

In this chapter, we focus on a version of the pursuit-evasion game played in simply-connected polygons (i.e., without obstacles), in which the two players have *a priori* knowledge about the environment, the same speed, and line-of-sight visibility: the two players can see each other only if the line segment connecting them lies inside the polygon. The goal of the pursuer is to capture the evader that tries to actively escape. These assumptions are realistic when considering for example a practical scenario in which a pursuer and an evader move inside an environment of limited size relying just on their vision. In such a case, the players have a belief on the other player's position when they are not visible to each other. Thus, it is necessary to explore the belief space so that the pursuer can find a strategy that guarantees capture no matter what the evader does.

In literature, the term visibility-based pursuit-evasion game is used for the game in which one or more pursuers with a visibility sensor try to detect an evader which is arbitrarily faster and whose position is unknown [77]. In contrast, the lion-and-man game is played by a pursuer and an evader that have the same speed and global visibility [113]. Our problem can be considered as lion-and-man game with visibility constraints. See [32] for an overview of recent results on these two games. Here we show a brief overview of some works dealing with the lion-and-man game with visibility constraints. Stiffler and O'Kane [122] present an algorithm that computes a shortest path to find the evader in a simply-connected polygon. Experimental results show the effectiveness of such approach. The work of Isler et al. [65] presents a randomized strategy to solve the problem of detecting an evader by a single pursuer with line-of-sight visibility and the problem of capturing it by two pursuers in simply-connected polygons. The expected time to capture the evader is $O(nT_1^2 + T_2 \cdot (n^2 \ln n))$, where $n$ is the number of vertices and $T_1$ and $T_2$ are the time it takes for the two pursuers to travel the diameter of the polygon, respectively. The authors also show a modification of the proposed strategy so that a single pursuer can capture the evader, but the expected capture time may significantly increase compared to that with two pursuers. The work of Noori and Isler [94] deals with the lion-and-man problem, where both have equal speeds and the lion has a line-of-sight visibility. The authors show that the lion can capture the man in any monotone polygon with a deterministic strategy and with a capture time of $O(n^7 D^{13})$, where $n$ is the number of vertices of the polygon and $D$ is the diameter of the polygon (i.e., the distance between the two furthest points in the polygon). Klein and Suri [70] study how many pursuers are necessary to capture an evader in an environment with obstacles in the lion-and-man game assuming the visibility-based discrete-time model. In a general hole-free polygon of $n$ vertices, the authors show that in the worst-case $\Theta(n^{1/2})$ pursuers are both necessary and sufficient. In an environment with $h$ holes, the upper bound on the number of pursuers is $O(n^{1/2}h^{1/4})$, for $h \leq n^{2/3}$, and $O(n^{1/3}h^{1/2})$ otherwise. All the above works dealing with lion-and-man game with visibility constraints assume that the evader has global visibility.

A general question that might arise from these works is whether there are other classes of environments in which a single pursuer can capture the evader with a deterministic strategy. In the contribution described in this chapter, by devising a search-based approach that explores the belief space of the players and reasons on an efficient state-space representation informed by the geometry for a simply-connected polygon, we investigate whether a pursuer is sufficient to capture an evader in realistic indoor environments. Specifically, we formulate an adversarial search problem to calculate the optimal capture path (if any), measured according to the number of time steps. We solve this problem by using a search method based on min-max and inspired to that of Chapter 3. However, the complexity of representing the information available to the players at a given time makes solving pursuit-evasion games with sensing limitations difficult if care is not taken in limiting the state space. Considering a high-resolution regular grid partition as usually done in path planning is not viable due to the high number of states to consider (this is further justified in our simulations). Instead, we partition the environment by connecting mutually visible vertices of the polygon with a line, embedding a visibility information on the deriving dual graph. In this way, we significantly circumvent the problem of the high computation, as the number of cells that derive from the partition we propose is drastically reduced. We show that, given our discretization, the solutions are complete for a rash evader model in which the evader can hide from the pursuer but does not move from one hiding location to another when the pursuer is not visible. We also present some techniques in order to further reduce the computational effort. Extensive simulated activities show the viability of our approach also for general evader models in games that take place in realistic indoor environments taken from a robotics repository.

## 10.1 Problem formulation

We study the following pursuit-evasion game that takes place in a simply-connected polygon $\mathscr{P}$, which is *a priori* known to both the pursuer and the evader.

We consider them as points in the plane. This is without loss of generality for translating robots, if we enlarge the polygon boundary to account for the real size of the two players, as usually done in path planning [76].

We assume that time $t$ is discretized and players move in turns of a unit time step. We denote $p(t)$ and $e(t)$ as the pursuer's and the evader's location at time step $t$, respectively. In each turn the players can move along a line segment of length at most 1 to a point visible to themselves.

Both players have line-of-sight visibility, namely, from a point $q \in \mathscr{P}$ they can see all the points that can be joined to $q$ with a line segment that completely lies in $\mathscr{P}$. We call $\mathit{VR}(q)$ the set of all points visible from a point $q \in \mathscr{P}$, the *visibility region*. This implies that at a time step $t$ the pursuer in position $p(t)$ and the evader in position $e(t)$ can see each other if and only if $p(t) \in \mathit{VR}(e(t))$ (or, equivalently, if $e(t) \in \mathit{VR}(p(t))$). Given that both players have a limited visibility, at a time step $t$, they have knowledge about the cleared region so far $\mathit{CR}(p(t))$ ($\mathit{CR}(e(t))$) which is a subset of $\mathscr{P}$ and includes the visibility region $\mathit{VR}(p(t))$ ($\mathit{VR}(e(t))$), but also regions that have been cleared at previous time steps and not possibly recontaminated (similarly to the events the work of Guibas et al. [59] considers). Thus, given their current position at a time step $t$, we define $E(t) = \{e(t) | e(t) \in \mathscr{P} \wedge e(t) \notin \mathit{CR}(p(t))\}$ as the current knowledge of the pursuer about the possible positions of the evader, when it is not visible. Similarly, we have the same kind of knowledge for the evader over the current possible positions of pursuer $P(t) = \{p(t) | p(t) \in \mathscr{P} \wedge p(t) \notin \mathit{CR}(e(t))\}$. Clearly, if the two players are visible to each other, we have singletons $E(t) = \{e(t)\}$ and $P(t) = \{p(t)\}$. Note that initially at $t = 0$, $E(0)$ and $P(0)$ correspond to $\mathscr{P} \setminus \mathit{VR}(p(0))$ and $\mathscr{P} \setminus \mathit{VR}(e(0))$, respectively.

The pursuer wins the game when, at a finite time $t^*$ (that it tries to minimize), the pursuer's location is the same as the evader (that tries to maximize $t^*$), namely, $p(t^*) = e(t^*)$. Otherwise, the evader wins the game if it can escape forever.

We call *general evader* one allowed to move at every turn: if the two players are not visible to each other, the evader can stay put an amount of time (which could be given by the distance between the last known position of the pursuer and the point where the evader disappeared), and then moves to an arbitrary vertex of $\mathscr{P}$ not in the current visibility region. In addition to general evader strategies, we will study a *rash evader* model. In this model, the evader can run and hide in an arbitrary hiding location which is not visible by the pursuer. However, the evader does not move from one hiding location to another unless the pursuer becomes visible. We will show that assuming the rash evader model allows us to efficiently solve the game in a complete fashion.

In general, the rash model is not unreasonable. Suppose the scenario in which the evader goes around a corner and the pursuer loses sight of the evader. Would it make sense for the evader to hide behind some vertex $v$

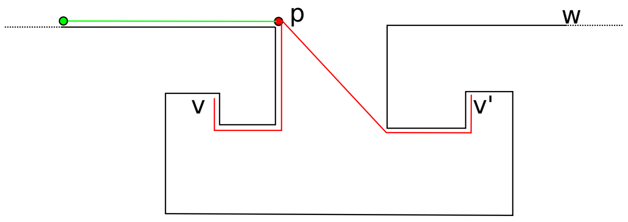


**Figure 10.1:** *Case where for the evader (red point), in the worst case, there is no gain in being general.*

for a while and then move to another vertex $w$ before the pursuer shows up? Intuitively, since the pursuer is not visible throughout this time, the evader could go to $w$ in the first place. The rash model captures this intuition as it allows the evader to choose an arbitrary vertex to hide behind but does not allow further movement until the pursuer becomes visible.

At this point, we do not know whether rash strategies are as powerful as general strategies. In the above scenario, could it be that there are two branches toward vertices $v$ and $v'$ and after hiding behind $v$ for some time $t$, the evader infers that the pursuer moved toward $v'$ and decides to attempt an escape toward $w$? The answer is not clear because the pursuer can prevent such inferences by waiting $t$ steps before it commits to $v$ or $v'$.

In most practical environments, the rash model seems to be as powerful as the general model. For example in cases similar to the one shown in Figure 10.1, we have the situation described above, in which the pursuer is following the evader along a corridor and then the evader hides in one of the two pockets in the hall. By hiding there, as the evader does not have knowledge about the current position of the pursuer, if the evader tries to get again to the corridor, it risks to be seen and possibly captured by the pursuer.

In the simulation results (shown in the next two sections), we can see that in realistic indoor environments, it does not seem that the general evader is significantly more powerful than the rash one.

## 10.2 Solving the game

In this section, we present the pursuit-evasion game formulated as an adversarial search problem and an efficient state-space representation using a visibility-based decomposition of the environment. We show that the solutions are complete for a rash evader model.

### 10.2.1 Formulation as a search problem

For a given state-space representation, the pursuit-evasion game presented in the previous section can be solved as a search problem. See, e.g., [112, Chapter 5] for further information on this classical approach.

In our case, the state $s(t)$ is a triple $((p(t), E(t)), (e(t), P(t)), cp)$ composed of the current positions $p(t)$ and $e(t)$ of pursuer and evader, of their current knowledge about the opponent $E(t)$ and $P(t)$, and of the current player $cp \in \{p, e\}$ that is playing at turn $t$.

For now, let us assume that the representation of the state space is powerful enough so that any subset of $\mathscr{P}$ can be represented. Our search problem for solving the pursuit-evasion problem is formulated as follows.

**Initial state**. The initial state $s(0) = ((p(0), E(0)), (e(0), P(0)), p)$ is given by the initial positions of the two players and their initial knowledge about the possible positions of the opponent.

**Player**. We assume that the pursuer starts the game, and so consequently for a state at an even round the pursuer plays, whereas at an odd round the evader plays.

**Actions**. From a state $s(t) = ((p(t), E(t)), (e(t), P(t)), cp)$, applicable actions for a player *cp* at time step $t$ are to move to any reachable point $q' \in \mathscr{P}$ given its maximum speed and to perceive the environment from the new point $q'$. A point $q'$ is reachable from $p(t)$ or $e(t)$ if there is a safe path (namely, inside the polygon and not colliding with the boundary of the environment). Player $\overline{cp} = \{p, e\} \setminus \{cp\}$ that does not play in $s(t)$ has no action, but can perceive.

**Transition function**. The new state of the game resulting from performing an applicable action during the player *cp* turn "*cp* moves to $q'$" is a new state $s(t + 1)$, where the position of the player *cp* is updated to $q'$ and also their knowledge is updated. Note that the number of new states resulting from the transition function depends on the state-space representation. If it is too fine (e.g., by using a fine-grained grid over the environment) we get too many states, while if it is too coarse, then the new state can become inaccurate.

**Terminal-test**. A terminal test checks whether (a) $p(t) = e(t)$ in a state $s(t) = ((p(t), \{e(t)\}), (e(t), \{p(t)\}), cp)$ or (b) there has been a loop in the path from the initial state to the current state or (c) $t > T$, where $T$ is the maximum time of the game.

**Utility**. The utility (or payoff) function $u(s(t))$, which evaluates the terminal states $s(t)$, if the pursuer wins, returns $g(s(t))$ (where $g(s(t))$ computes the number of time steps $n$ to reach $s(t)$) and, if the evader wins (in the case of (b) or (c)), returns $+\inf$.

A solution to the above search problem is a finite sequence of states $S = \langle s(0), s(1), \ldots, s(n) \rangle$ such that $s(0)$ is the initial state and $s(n)$ is a state that satisfies the terminal test. An optimal solution is a solution $S^*$ that is optimal in terms of the number of time steps $n^*$ (that the pursuer tries to minimize, while the evader tries to maximize), that is no player has any gain by changing an action in the solution.

### 10.2.2 Solution of the search problem

Given the search problem defined above, the solution efficiency is related to the representation of the state space. We present an efficient way of representing the environment by providing a decomposition that partitions the polygon $\mathscr{P}$ in a set of disjoint cells $C$ ($\bigcup_{c \in C} c = \mathscr{P}$) and returns a subset of points $D = \{d \mid d \text{ belongs to a } c \in C\}$, which is needed for the solution and show the completeness for the rash evader model under such representation.

Given the subset $D$ of points derived from a decomposition representing an environment $\mathscr{P}$, we have the following state representation $s_D(t)$, which is a triple $((p(t), E(t)), (e(t), P(t)), cp)$, where $p(t), e(t) \in D$. Also the actions change: from a current point $q \in D$ that belongs to a cell $c \in C$, we have that next points $q' \in D$ are in neighbor cells $c'$.

In robotics, it is common to tile $\mathscr{P}$ with a finite regular grid of cells $G^e$ such that each point $q \in \mathscr{P}$ belongs to a cell of the grid, as we did in Chapter 3. Cells are identical squares with edge $e$ and can be either free or occupied by the boundaries of the environment. The positions encoded in a state is given by the center of a cell and the next possible actions are the movement to the center of the neighbor cells. However, depending on $e$, the number of cells can be really high and such decomposition does not encode any visibility information.

Now, we present a decomposition informed by the geometry of the environment, which allows to largely reduce the number of possible cells (and so the number of possible actions and states). Specifically, we consider the cells that derive from drawing a line (that we call *inducing line*) between each pair of visible vertices of the polygon and that lies inside the environment (see Figure 10.2). Each of these lines by intersecting with other lines and the boundary of the polygon induces some cells. We refer to the resulting partition as *visibility-based decomposition.* By construction, this decomposition generates convex cells. All the points inside a cell are informatively equivalent:

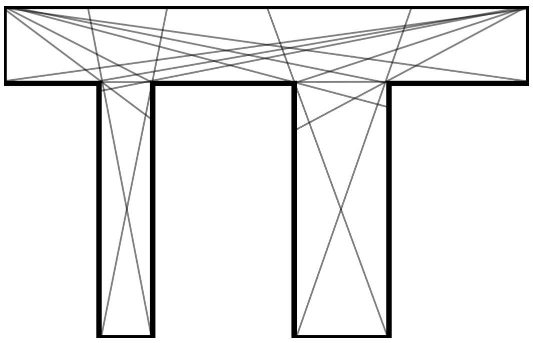


**Figure 10.2:** *Polygonal decomposition of an environment.*

**Proposition 10.1.** *Let $x$ and $y$ be two points of a cell obtained by the visibility-based decomposition. Let $z$ be a third point. $x$ and $z$ are mutually visible if and only if $y$ and $z$ are mutually visible.*

In our representation, when the players are not visible, rather than discretizing the entire space, we will keep track of only the entry and the exit points to and from a cell by discretizing the cell boundaries. More formally, to keep track of the entry and exit points we define as cells $C$ the edges of the visibility-based cells. On each of such cell, we have a set of points $D$ that belong to each cell $c \in C$. The points $D$ are obtained by discretizing the edge corresponding to a cell $c$. The neighbor cells that determine the next possible actions for the players are those in which the shortest path between the current point that belongs to a cell $c$ to a next point that belongs to a cell $c'$ does not intersect with any other cell $c'' \in C$. Thus, the positions encoded in the state are the points along the edges of a visibility-based cell. We refer to the resulting state representation as the *visibility-based representation.* Note that at a time step $t$ it is not necessary to reason on all of the cells, but just on those whose inducing lines are not entirely contained in the visibility region of a player. The lines that lie in one's visibility region do not change the information set of the player.

Instead, when the players are visible to each other, the current knowledge of a player about the opponent is the opponent's actual position. As the game evolves in a turn-based fashion, when reasoning on the successor states, we need to take into account the reachable points from their current position in a time step. In particular, those points are found online at each time step on the shortest path between a player and the points on the edges of the cells. We call *turn representation* the information the players can have and the resulting points which the search algorithm can reason on.

We can prove that this decomposition allows to preserve the completeness of a search algorithm that reasons on such state space.

**Theorem 1.** *A strategy to capture the rash evader exists if and only if a complete search algorithm can find a solution in the state-space of the visibility-based and turn representation up to a given discretization.*

*Proof. Only if:* This direction is trivial because if the search algorithm finds a solution, then a capture strategy exists.

*If:* Now, suppose that there exists a strategy to capture the evader. We must show that a strategy still exists when we restrict the representation of the information available to the players to the visibility-based and turn decomposition. Then the theorem follows from the completeness of min-max search.

Notice that when the players see each other their information state is the opponent's location. In this case, the two representations are identical and the solution is complete up to a given resolution level which is encoded by the turn representation.

When the players are not visible, under the rash model, the pursuer's information can be represented as possible vertices the evader can be hiding behind. The evader, when in hiding, does not move. What affects his strategy is where the pursuer enters its line of sight. By Proposition 10.1, this event corresponds to crossing a cell boundary in the visibility-based decomposition. Therefore, the representation captures it up to discretization of the entrance point.

Hence, in all case, the information available is captured by the visibility-based and turn decomposition. □

We use a min-max approach with branch-and-bound as search algorithm, for which we provide a sketch about how it operates. Starting from the initial state, the terminal test is applied. If it is true, then the game ends. Otherwise, the successor states are recursively found in a depth-first way according to the current state and player turn until a leaf node is encountered. When it is encountered, the utility function shown in the previous section is applied to compute the values for the players and is propagated to the non-leaf nodes. Other branches are then evaluated and possibly pruned if their current value is worse than the one already found (either for the pursuer or the evader). The worst-case computational complexity of our min-max-based approach is exponential in the number of time steps needed to reach a terminal state. However, as the results of the next section show, our approach can solve pursuit-evasion games for realistic indoor environments in reasonable time. In order to further improve the efficiency of our approach, we introduce also some speedup techniques that are expected to reduce the computational effort, preserving the quality of the solutions.

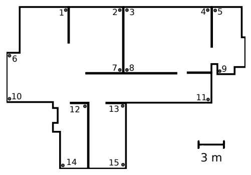
**Dominated actions.** When the players are mutually visible, we consider a subset of possible actions available to the evader: we discard the actions that would lead to points $q$ where the pursuer is or that it can reach in one time step (in case no action is available, the evader stays put in its location). We have that the existence of the solution is preserved, as if the evader moves to a point that is guarded by the pursuer, then at the next time step it would be captured.

**Duplicate states.** When selecting a state to expand, duplicates that have been already expanded can be discarded: considering the next player move, a state $s(\tau) = ((p(\tau), E(\tau)), (e(\tau), P(\tau)), cp)$ can be safely discarded if $\exists s(\tau') = ((p(\tau'), E(\tau')), (e(\tau'), P(\tau')), cp)$ already expanded ($\tau' < \tau$) such that $p(\tau) = p(\tau')$, $e(\tau) = e(\tau')$, and, in the case of the pursuer, $E(\tau') \subseteq E(\tau)$ and $g(s(\tau')) > g(s(\tau))$, while in the case of the evader $P(\tau') \subseteq P(\tau)$ and $g(s(\tau')) < g(s(\tau))$ (recall that $g(s(\tau))$ is the cost to reach the state $s(\tau)$ from the initial state $s(0)$). The algorithm is still complete and optimal as duplicate states will eventually lead to the same states already expanded, and just those states with a cost greater for the pursuer (less for the evader) than the cost of at least one already expanded duplicate is not considered.

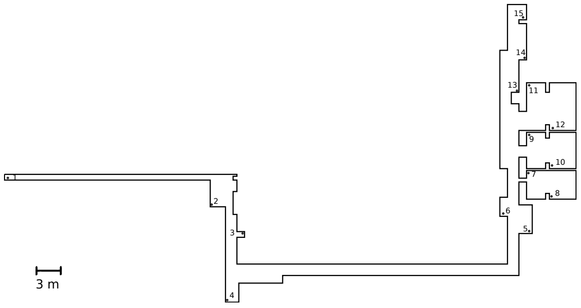
## 10.3 Simulation results for rash evader

In order to assess the validity of our approach in solving pursuit-evasion games with visibility, extensive simulated experiments have been conducted in indoor environments available in Radish repository [63]. As in the repository environments are mainly represented with grid maps, we manually converted them in simple polygons. We report here two representative environments: one (*albert-b-laser*) with a unique short corridor with rooms attached to it (Figure 10.3a) and the other one (*utk-claxton*) with long corridors (Figure 10.3b) (boundaries of environments are considered as obstacles). The line segment reported in the figure measures $3\,\mathrm{m}$, so the size of the environments is approximately $30\,\mathrm{m} \times 24\,\mathrm{m}$ and $140\,\mathrm{m} \times 80\,\mathrm{m}$, respectively (note that the size of the environments has been guessed by considering $10\,\mathrm{cm}$ the size of the grid cell, as this information is not reported in the data sets: this seems to be reasonable in terms of the average size of a room).

Table 10.1 shows the number of cells that derive from imposing a high-resolution grid on these environments and from partitioning them with the



**(a)** *Indoor with rooms.*



**(b)** *Indoor with long corridors.*

**Figure 10.3:** *Selected indoor environments where points indicate the initial positions of the players.*

| Environment | Number of grid cells | Number of visibility-based cells |
|---|---|---|
| Indoor with rooms | 71628 | 4604 |
| Indoor with long corridors | 117625 | 2964 |

**Table 10.1:** *Number of grid cells (where the size of the cell edge is* $10\,\mathrm{cm}$*) and of cells deriving from the visibility-based decomposition.*

visibility-based decomposition. It can be noted that the number of cells is greatly abated in both cases, but especially for the second environment, because of the presence of less rooms.

We call setting a combination of initial positions (chosen randomly so that they could cover the whole environment; see Figure 10.3) and environment. For each setting we ran our approach on a computer equipped with a $3.50\,\mathrm{GHz}$ i7-4710HQ processor and $8\,\mathrm{GB}$ RAM to find the optimal pursuit strategy that guarantees capture of the evader. We have developed our own implementation of the simulator and the search algorithm in C++ (the state space is generated online), as no generic graph library, like BGL [17], is easily adaptable to our case. We used CGAL library [29] for handling simple polygons and arrangements deriving from the partition of the environment.

We set a timeout of $5$ hours for each run. For all of the runs that found a solution, we report the average and standard deviation of the traveled distance (in $1$ time step they are assumed to travel of $1\,\mathrm{m}$).

Table 10.2 reports experimental results for the two environments and rash evader. In all experiments we considered the speedup techniques we presented in the previous section. The values reported in each entry are the average and the standard deviation (in parentheses) over the $15 \times 14$

| | Number of time steps | Computation time (s) |
|---|---|---|
| Indoor with rooms | 65.8 (20.9) | 196.5 (419.0) |
| Indoor with long corridors | 309.1 (21.7) | 774.1 (1059.6) |

**Table 10.2:** *Results (average and standard deviation over the results for each setting) for the indoor environments under the rash evader model.*

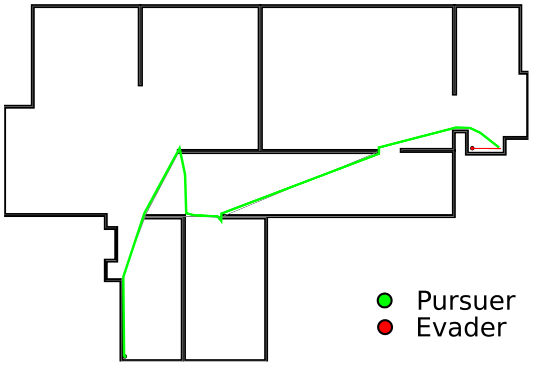


**Figure 10.4:** *Simulation instance.*

combinations of initial positions for the corresponding environment (without considering the same initial position for both players). It emerges that in both environments the proposed method was able to find a solution. The relatively high standard deviation could be explained by the fact that the number of time steps to capture the evader is highly dependent on the initial positions of the two players. For example, in the first environment, from positions 12 (pursuer) and 14 (evader), both players can see each other from the start of the game and the evader is basically trapped in that room, while from positions 14 (pursuer) and 9 (evader) the pursuer has to clear some of the contaminated area before reaching the area where the evader is located. This is reflected also on the computational time for finding solutions, which varies greatly with the setting: the more time steps required, the higher the computational time. Under the rash evader model it seems that there is a roughly linear relation between them.

Figure 10.4 shows an instance of the solution found with the starting position of the two players. As can be noted, the pursuer tries to clear each contaminated region at a time considering the worst case position of the evader, in such a way that the evader can be trapped and cannot recontaminate the cleared region. Finally, when the evader is found, it is trapped in a corner, as the pursuer is able to guard the line that prevents the evader from escaping that area.

We also ran some experiments considering a grid decomposition and removing the speedup techniques, however, most of the simulated runs did not terminate within the timeout and so the results are not reported.

In the second indoor environment, the number of time steps to capture the evader increases because of the presence of very long corridors. Furthermore, given the fact that a corridor could be thought of as a line, if the pursuer is in the middle, it has first to go to one side, and, if the evader is not in that area, it has to go to the other side.

So, in realistic indoor environments, it seems that is always possible to find a deterministic solution against the rash evader model in a reasonable computation time. A possible explanation could be that, as the environment is highly structured, usually contaminated regions are disjoint, and passages between rooms and corridors are narrow, allowing the pursuer to protect the area cleared so far.

## 10.4 Simulation results for general evader

We also ran some experiments considering the general evader model and the results are reported in Table 10.3. In the simulation settings considered, the pursuer is able to capture the evader. This could be explained by the fact that the evader has limited visibility and the lack of knowledge about the pursuer strategy does not allow to purposefully escape. Furthermore, given the low number of pockets where the evader can hide in such indoor environments, it seems not to be able to hide itself to the pursuer. The general evader model could require more time steps for the capture than the rash model. However, there is not a significant difference according to an ANOVA analysis with a $p$-value $< 0.05$ [101]. For example, in the first environment, we have $p$-value=$0.1$. The fact that the difference is not evident could be due the fact that once the pursuer reaches the corridor, it can always keep the corridor clear. Note that the corridor is the only way to get to other rooms. The computation time, however, greatly increases with the general evader model and the difference is statistically significant ($p$-value=$1.4 \cdot 10^{-10}$). The more time steps required, the higher the computational time, but it seems that the relation here is sublinear.

Our conjecture is that even for a general evader model the search algorithm under the visibility-based decomposition can find a solution.

In this chapter, we have presented a method for finding the optimal pursuit path in a simply-connected polygon when the pursuer and the evader have line-of-sight visibility. There is no closed-form solution known for

| | Number of time steps | Computation time (s) |
|---|---|---|
| Indoor with rooms | 81.3 (31.3) | 3430.3 (621.4) |
| Indoor with long corridor | 338.4 (18.3) | 4528.4 (1134.0) |

**Table 10.3:** *Results (average and standard deviation over the results for each setting) for the indoor environments under general evader model.*

this game. Our results provide a computational search-based method for solving it, by using an efficient representation of the state of the game using a visibility-based decomposition of the environment. The extensive simulated activities showed the viability of the proposed approach. More in general, it seems that search-based approaches, like the one in Chapter 3, can explore abstract state spaces.

# 11 Conclusions

Exploration strategies and coordination methods are fundamental components for autonomous mobile robots whose task is to discover features of initially unknown environments. They can really lead to a breakthrough in autonomous mobile robotics, as they foster robots autonomy in the decisions on where to go and on who goes where to accomplish a given mission, like search and rescue. There are several gaps in the current research on coordinated multirobot exploration. An important one is the fact that either the theoretical approaches are far from real applicability, while the practical ones are rather specific and cannot be plainly used in contexts different from those where experiments and tests were carried out. Also, the experimental methodology for multirobot exploration systems is still not fully mature. This slows down research and makes the transfer of existing research results to market products rather difficult.

The contributions presented in this dissertation advance the filling of these gaps in the exploration problem. They are summarized according to the three general goals stated in the introduction.

From a more theoretical point of view, we first considered the problem of calculating the optimal offline exploration (constrained coverage) path, followed by a robot with time-discrete and limited perception in grid environments. A search-based approach is proposed and simulation results show the viability of our approach for realistic environments. The main advantage of this approach is that it is general enough to be computed on any environment given as input. This contribution enables the comparison of online exploration strategies with the optimal behavior in a given environment. Second, using a graph-based model of the environment, we provided worst-case and average-case analysis on the number of edge traversals required by exploration strategies that include distance and expected information gain as criteria to evaluate candidate destination locations. The obtained theoretical results show that, in the worst case, taking into account also information gain in selecting the next destination location does not provide any advantage over considering only distance, while it does in the average case on graphs modeling realistic indoor environments. This contribution is a step towards making some of the exploration strategies used in practice and obtained results more theoretically-grounded.

On the practical side, we built a multirobot exploration system which employs exploration strategies and coordination methods that use information coming from semantic maps. Simulation results show the significant benefits of using semantic information on the exploration of both total explored area and, prominently, relevant areas, when *a priori* information about these relevant areas is available, e.g., from human users.

About good experimental methodologies, besides providing a tool to compute an optimal reference, against which exploration strategies can be compared in specific settings, we showed that one of the aspects that have been overlooked so far is the full reproducibility and robustness of the experiments, given by the fact that, for example, parameters are usually not fully reported in the description of the experiments. Thinking that this should become best practice, we presented detailed experimental results on some factors (i.e., perception/decision timings and exploration strategies vs. coordination methods) that can impact on the performance of exploration. These quantitative analyses provide insights on settings some important parameters for an exploration system and further remark the importance of reporting and studying the values of these parameters in the description of the experiments.

Moreover, we showed the applicability of our search-based approach to explore belief state spaces in pursuit-evasion games, and the simulation results confirmed the viability of our approach in that context too.

Several issues of the exploration problem are worth further investigation both from theoretical and practical points of view. From a theoretical point of view, more steps should be made to study exploration problems where assumptions are more realistic. For example, it could be interesting to study:

- Exploration strategies that use a function that aggregates different criteria to understand how to normalize the two metrics.
- Constrained exploration, in which a robot should, for example, stay within a certain distance from the base station [39]. This can model

a practical scenario where a robot has a limited communication range with a base station or is plugged to a power socket in order to extend its autonomy in terms of battery.

- Multirobot exploration, in which multiple robots are employed to efficiently accomplish exploration task [20].

Both cases can be analyzed using artificial intelligence techniques to find the optimal strategies on graph- or geometrically-represented environments and to derive bounds on the performance of exploration strategies typically used in real settings.

On the possible extensions of the semantically-informed exploration system, future work could address several research directions. First, for improving our proposed multirobot exploration system, one important aspect is to find an automated and theoretically-grounded way to compute some of the parameters used in our system, particularly focusing on:

- The membership functions that model the features of rooms and corridors used by the proposed coordination method can be set according to the specific building type (e.g., being a school), on the basis of the results we had in [80], or according to the robots' capabilities (e.g., sensor range).

- The number of robots to send to the same candidate location in order to speed up the exploration should be determined by a theoretically-grounded method, drawing some insights given by the work of Nieto-Granda et al. [93].

This would make the design of the system more reliable, robust, and general. Furthermore, the exploration system could be extended in a more broader perspective, by studying how to integrate and analyzing the impact of some other *a priori* information that could be available in realistic scenarios, including:

- Information about relevant areas derived or changed at runtime, imagining a scenario where robots interact with humans during the exploration process.

- Distributions of probability over the possible locations of the victims, to guide the exploration process, starting from results of Aydemir et al. [14].

Moreover, as we extended the coordination method to the multi-robot (MR) paradigm, another direction of interest is the investigation of multi-task

(MT) coordination methods (i.e., each robot plans how to reach a sequence of candidate locations) or path/trajectory optimization, starting from results of Tovar et al. [126] and Prestes and Engel [103].

The experimental results obtained using realistically-simulated robotic systems, although highly informative, are far from being exhaustive or definitive and validation with real robots is an issue of paramount importance. There are several other parameters that can be changed to evaluate autonomous robotic systems for exploration. For example, quality of the resulting map and the amount of area mapped over time could be included in the adopted metrics. Other settings should be considered in further experiments, which include other environments (e.g., outdoor areas) and additional map building methods.

In general, future work in all the directions (i.e., theoretical and practical) could address the further assessment of proposed methods considering real robots, where real-world noise is included, for example in communication and sensing, and it could be interesting to investigate the use of Markov Decision Processes to model exploration in order to account for uncertainty, like in [85].

Looking at exploration, thus, the general future research direction is to define a framework for the theoretical and practical development and evaluation of exploration strategies and coordination methods for increasing autonomy of mobile robots.

In a broader perspective, we can generalize exploration to a problem in which robots can “explore” other features of the environment. An interesting domain is that of environmental monitoring, where some techniques similar to that of exploration strategies could be applied, as done, for example, in [98]. Other examples include continuous monitoring of temporal-spatial phenomena in partially known environments, like ocean monitoring [119], olfactory exploration [83], and tactile exploration of object properties [102]. In this direction, the objective is to design an even more general framework in order to define navigation strategies and coordination methods that are general and largely application-independent, so that the development of a robotic system is eased and can be ideally carried out by instantiating the general framework in a specific setting.

# Bibliography

[1] S. Albers, K. Kursawe, and S. Schuierer. Exploring unknown environments with obstacles. *Algorithmica*, 32:123–143, 2002.

[2] F. Amigoni. Experimental evaluation of some exploration strategies for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2818–2823, 2008.

[3] F. Amigoni and V. Caglioti. An information-based exploration strategy for environment mapping with mobile robots. *Robotics and Autonomous Systems*, 5(58):684–699, 2010.

[4] F. Amigoni and V. Schiaffonati. Good experimental methodologies and simulation in autonomous mobile robotics. In L. Magnani, W. Carnielli, and C. Pizzi, editors, *Model-Based Reasoning in Science and Technology*, volume 314 of *Studies in Computational Intelligence*, pages 315–332. Springer, 2010.

[5] F. Amigoni and V. Schiaffonati. Autonomous mobile robots as technical artifacts: A discussion of experimental issues. In L. Magnani, editor, *Model-Based Reasoning in Science and Technology*, volume 8 of *Studies in Applied Philosophy, Epistemology and Rational Ethics*, pages 527–542. Springer, 2014.

[6] F. Amigoni, M. Reggiani, and V. Schiaffonati. An insightful comparison between experiments in mobile robotics and in science. *Autonomous Robots*, 27(4):313–325, 2009.

[7] F. Amigoni, N. Basilico, and A. Quattrini Li. How much worth is coordination of mobile robots for exploration in search and rescue? In *Proceedings of the International RoboCup Symposium*, pages 106–117, 2012.

[8] F. Amigoni, A. Quattrini Li, and D. Holz. Evaluating the impact of perception and decision timing on autonomous robotic exploration. In *Proceedings of the European Conference on Mobile Robots (ECMR)*, pages 68–73, 2013.

[9] F. Amigoni, N. Basilico, and A. Quattrini Li. Moving from 'how to go there?' to 'where to go?': Towards increased autonomy of mobile robots. In A. Rodic, D. Pisla, and H. Bleuler, editors, *New Trends in Medical and Service Robots*, pages 345–356. Springer, 2014.

[10] E. Arkin, S. Fekete, and J. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry: Theory and Applications*, 17(1-2):25–50, 2000.

[11] E. Arkin, M. Bender, E. Demaine, S. Fekete, J. Mitchell, and S. Sethia. Optimal covering tours with turn costs. *SIAM Journal on Computing*, 35(3):531–566, 2005.

[12] Y. Asahiro, E. Miyano, S. Miyazaki, and T. Yoshimuta. Weighted nearest neighbor algorithms for the graph exploration problem on cycles. *Information Processing Letters*, 110:93–98, 2010.

[13] B. Awerbuch, M. Betke, R. Rivest, and M. Singh. Piecemeal graph exploration by a mobile robot. *Information and Computation*, 152 (2):155–172, 1999.

[14] A. Aydemir, A. Pronobis, M. Göbelbecker, and P. Jensfelt. Active visual object search in unknown environments using uncertain semantics. *IEEE Transactions on Robotics*, 29(4):986–1002, 2013.

[15] N. Basilico and F. Amigoni. On evaluating performance of exploration strategies for an autonomous mobile robot. In *IROS Workshop on Performance Evaluation and Benchmarking for Intelligent Robots and Systems*, 2008.

[16] N. Basilico and F. Amigoni. Exploration strategies based on multi-criteria decision making for searching environments in rescue operations. *Autonomous Robots*, 31(4):401–417, 2011.

[17] BGL. The Boost Graph Library, 2014. URL `http://www.boost.org/`.

[18] D. Bhadauria, K. Klein, V. Isler, and S. Suri. Capturing an evader in polygonal environments with obstacles: The full visibility case. *The International Journal of Robotics Research*, 31(10):1176–1189, 2012.

[19] A. Bottino and A. Laurentini. A nearly optimal algorithm for covering the interior of an art gallery. *Pattern Recognition*, 44(5):1048–1056, 2011.

[20] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao. Multirobot tree and graph exploration. *IEEE Transactions on Robotics*, 27(4):707–717, 2011.

[21] W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 476–481, 2000.

[22] W. Burgard, M. Moors, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–378, 2005.

[23] D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi. Multi-objective exploration and search for autonomous rescue robots. *Journal of Field Robotics*, 24(8-9):763–777, 2007.

[24] D. Calisi, L. Iocchi, D. Nardi, G. Randelli, and V. Ziparo. Improving search and rescue using contextual information. *Advanced Robotics*, 23:1199–1216, 2009.

[25] S. Carlsson, B. Nilsson, and S. Ntafos. Optimum guard covers and m-watchmen routes for restricted polygons. In F. Dehne, J.-R. Sack, and N. Santoro, editors, *Algorithms and Data Structures*, volume 519 of *Lecture Notes in Computer Science*, pages 367–378. Springer, 1991.

[26] S. Carlsson, H. Jonsson, and B. Nilsson. Finding the shortest watchman route in a simple polygon. *Discrete & Computational Geometry*, 22:377–402, 1999.

[27] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. USARSim: A robot simulator for research and education. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1400–1405, 2007.

[28] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. Sorrenti, and P. Taddei. Rawseeds ground truth collection systems for indoor self-localization and mapping. *Autonomous Robots*, 27(4):353–371, 2009.

[29] CGAL. Computational Geometry Algorithms Library, 2014. URL `http://www.cgal.org/`.

[30] W.-P. Chin and S. Ntafos. Optimum watchman routes. *Information Processing Letters*, 28(1):39–44, 1988.

[31] H. Choset. Coverage for robotics: A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):113–126, 2001.

[32] T. Chung, G. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics – A survey. *Autonomous Robots*, 31(4):299–316, 2011.

[33] R. Cipolleschi, M. Giusto, A. Quattrini Li, and F. Amigoni. Semantically-informed coordinated multirobot exploration of relevant areas in search and rescue settings. In *Proceedings of the European Conference on Mobile Robots (ECMR)*, pages 216–221, 2013.

[34] B. Clark, C. Colbourn, and D. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1–3):165–177, 1990.

[35] S. Das, P. Flocchini, S. Kutten, A. Nayak, and N. Santoro. Map construction of unknown graphs by multiple agents. *Theoretical Computer Science*, 385(1–3):34–48, 2007.

[36] J. de Hoog, S. Cameron, and A. Visser. Autonomous multi-robot exploration in communication-limited environment. In *Proceedings of the Conference Towards Autonomous Robotics Systems (TAROS)*, pages 68–75, 2010.

[37] X. Deng and C. Papadimitriou. Exploring an unknown graph. *Journal of Graph Theory*, 32(3):265–297, 1999.

[38] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment I: The rectilinear case. *Journal of the ACM*, 45 (2):215–245, 1998.

[39] C. Duncan, S. Kobourov, and A. Kumar. Optimal constrained graph exploration. *ACM Transactions on Algorithms*, 2(3):380–402, 2006.

[40] T. Edlinger and E. von Puttkamer. Exploration of an indoor-environment by an autonomous mobile robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1278–1284, 1994.

[41] J. Faigl, M. Kulich, and L. Preucil. Goal assignment using distance cost in multi-robot exploration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3741–3746, 2012.

[42] P. Fazli, A. Davoodi, and A. Mackworth. Multi-robot repeated area coverage. *Autonomous Robots*, 34(4):251–276, 2013.

[43] S. Fekete and C. Schmidt. Polygon exploration with time-discrete vision. *Computational Geometry*, 43(2):148–168, 2010.

[44] S. Fekete, J. Mitchell, and C. Schmidt. Minimum covering with travel cost. In *Proceedings of the International Symposium on Algorithms and Computation (ISAAC)*, pages 393–402, 2009.

[45] A. Felner, R. Stern, A. Ben-Yair, S. Kraus, and N. Netanyahu. PHA*: Finding the shortest path with A* in an unknown physical environment. *Journal of Artificial Intelligence Research*, 21:631–670, 2004.

[46] K. Forster and R. Wattenhofer. Directed graph exploration. In R. Baldoni, P. Flocchini, and R. Binoy, editors, *Principles of Distributed Systems*, volume 7702 of *Lecture Notes in Computer Science*, pages 151–165. Springer, 2012.

[47] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325–1339, 2006.

[48] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. *Theoretical Computer Science*, 345(2–3):331–344, 2005.

[49] L. Freda and G. Oriolo. Frontier-based probabilistic strategies for sensor-based exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3881–3887, 2005.

[50] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31:77–98, 2001.

[51] Y. Gabriely and E. Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Computational Geometry: Theory and Applications*, 24(3):197–224, 2003.

[52] B. Gerkey and M. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23:939–954, 2004.

[53] B. Gerkey, S. Thrun, and G. Gordon. Visibility-based pursuit-evasion with limited field of view. *The International Journal of Robotics Research*, 25(4):299–315, 2006.

[54] S. Ghosh and R. Klein. Online algorithms for searching and exploration in the plane. *Computer Science Review*, 4(4):189–201, 2010.

[55] S. Ghosh, J. Burdick, A. Bhattacharya, and S. Sarkar. Online algorithms with discrete visibility – exploring unknown polygonal environments. *IEEE Robotics and Automation Magazine*, 15(2):67–76, 2008.

[56] H. Gonzáles-Baños and J.-C. Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21(10-11):829–848, 2002.

[57] H. González-Baños. A randomized art-gallery algorithm for sensor placement. In *Proceedings of the Annual Symposium on Computational Geometry*, pages 232–240, 2001.

[58] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[59] L. Guibas, J.-C. Latombe, S. LaValle, D. Lin, and R. Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry & Applications*, 9(4-5):471–494, 1999.

[60] J. Hawley and Z. Butler. Hierarchical distributed task allocation for multi-robot exploration. In A. Martinoli, F. Mondada, N. Correll, G. Mermoud, M. Egerstedt, A. Hsieh, L. Parker, and K. Støy, editors, *Distributed Autonomous Robotic Systems*, volume 83 of *Springer Tracts in Advanced Robotics*, pages 445–458. Springer, 2013.

[61] Y. Higashikawa, N. Katoh, S. Langerman, and S. Tanigawa. Online graph exploration algorithms for cycles and trees by multiple searchers. *Journal of Combinatorial Optimization*, 28(2):480–495, 2014.

[62] D. Holz, N. Basilico, F. Amigoni, and S. Behnke. A comparative evaluation of exploration strategies and heuristics to improve them. In *Proceedings of the European Conference on Mobile Robots (ECMR)*, pages 25–30, 2011.

[63] A. Howard and N. Roy. The robotics data set repository (Radish). `http://radish.sourceforge.net/`, 2003.

[64] V. Isler. Theoretical robot exploration. Technical report, Computer and Information Science, University of Pennsylvania, 2001.

[65] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *IEEE Transactions on Robotics*, 21(5): 875–884, 2005.

[66] A. Itai, C. Papadimitriou, and J. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

[67] M. Julia, A. Gil, and Ó. Reinoso. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33(4):427–444, 2012.

[68] B. Kalyanasundaram and K. Pruhs. Constructing competitive tours from local information. *Theoretical Computer Science*, 130:125–138, 1994.

[69] M. Keidar and G. Kaminka. Robot exploration with fast frontier detection: theory and experiments. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 113–120, 2012.

[70] K. Klein and S. Suri. Capture bounds for visibility-based pursuit evasion. In *Proc. SOCG*, pages 329–338, 2013.

[71] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3232–3238, 2003.

[72] S. Koenig, C. Tovey, and Y. Smirnov. Performance bounds for planning in unknown terrain. *Artificial Intelligence*, 147(1-2):253–279, 2003.

[73] A. Kröller, T. Baumgartner, S. Fekete, and C. Schmidt. Exact solutions and bounds for general art gallery problems. *ACM Journal of Experimental Algorithmics*, 17(1):1–23, 2012.

[74] B. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8:47–63, 1981.

[75] N. Kwak, G.-W. Kim, S.-H. Ji, and B.-H. Lee. A mobile robot exploration strategy with low cost sonar and tungsten-halogen structured light. *Journal of Intelligent Robot Systems*, 51:89–111, 2008.

[76] S. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[77] S. LaValle, D. Lin, L. Guibas, J.-C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 737–742, 1997.

[78] M. Lopez-Sanchez, F. Esteva, R. Lopez de Mantaras, C. Sierra, and J. Amat. Map generation by cooperative low-cost robots in structured unknown environments. *Autonomous Robots*, 5:53–61, 1998.

[79] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18(3):249–275, 1997.

[80] M. Luperto, A. Quattrini Li, and F. Amigoni. A system for building semantic maps of indoor environments exploiting the concept of building typology. In *Proceedings of the International RoboCup Symposium*, pages 504–515, 2013.

[81] A. Makarenko, S. Williams, F. Bourgault, and H. Durrant-Whyte. An experiment in integrated exploration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 534–539, 2002.

[82] R. Mannadiar and I. Rekleitis. Optimal coverage of a known arbitrary environment. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5525–5530, 2010.

[83] A. Marjovi and L. Marques. Multi-robot topological exploration using olfactory cues. In A. Martinoli, F. Mondada, N. Correll, G. Mermoud, M. Egerstedt, A. Hsieh, L. Parker, and K. Støy, editors, *Distributed Autonomous Robotic Systems*, volume 83 of *Springer Tracts in Advanced Robotics*, pages 47–60. Springer, 2013.

[84] A. Marjovi, J. Nunes, L. Marques, and A. de Almeida. Multi-robot exploration and fire searching. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1929–1934, 2009.

[85] L. Matignon, L. Jeanpierre, and A. Mouaddib. Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2017–2023, 2012.

[86] N. Megow, K. Mehlhorn, and P. Schweitzer. Online graph exploration: New results on old and new algorithms. *Theoretical Computer Science*, 463:62–72, 2012.

[87] J. Mitchell. Geometric shortest paths network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook on Computational Geometry*, pages 633–702. Elsevier Science, 2000.

[88] S. Miyazaki, N. Morimoto, and Y. Okabe. The online graph exploration problem on restricted graphs. *IEICE Transactions on Information Systems*, E92-D(9):1620–1627, 2009.

[89] O. Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using AdaBoost. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1742–1747, 2005.

[90] R. Murrieta-Cid, R. Monroy, S. Hutchinson, and J.-P. Laumond. A complexity result for the pursuit-evasion game of maintaining visibility of a moving evader. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2657–2664, 2008.

[91] A. Nash. *Any-angle Path Planning*. PhD thesis, University of Southern California, 2012.

[92] P. Newman, M. Bosse, and J. Leonard. Autonomous feature-based exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1234–1240, 2003.

[93] C. Nieto-Granda, J. Rogers, and H. Christensen. Coordination strategies for multi-robot exploration and mapping. *The International Journal of Robotics Research*, 33(4):519–533, 2014.

[94] N. Noori and V. Isler. Lion and man with visibility in monotone polygons. *The International Journal of Robotics Research*, 33(1): 155–181, 2014.

[95] S. Ntafos. Watchman routes under limited visibility. *Computational Geometry: Theory and Applications*, 1(3):149–170, 1992.

[96] J. O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press, 1987.

[97] P. Panaite and A. Pelc. Impact of topographic information on graph exploration efficiency. *Networks*, 36(2):96–103, 2000.

[98] L. Parker, R. Coogle, and A. Howard. Estimation-informed, resource-aware robot navigation for environmental monitoring applications. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1041–1046, 2013.

[99] T. Parsons. Pursuit-evasion in a graph. In Y. Alavi and D. Lick, editors, *Theory and Applications of Graphs*, volume 642 of *Lecture Notes in Mathematics*, pages 426–441. Springer, 1978.

[100] M. Peniak, D. Marocco, and A. Cangelosi. Autonomous robot exploration of unknown terrain: a preliminary model of Mars Rover robot. In *10th ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA2008)*, 2008.

[101] W. Pestman. *Mathematical Statistics: an Introduction*. de Gruyter, 1998.

[102] Z. Pezzementi, E. Plaku, C. Reyda, and G. Hager. Tactile-object recognition from appearance information. *IEEE Transactions on Robotics*, 27(3):473–487, 2011.

[103] E. Prestes and P. Engel. Exploration driven by local potential distortions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1122–1127, 2011.

[104] A. Quattrini Li, F. Amigoni, and N. Basilico. Searching for optimal off-line exploration paths in grid environments for a robot with limited visibility. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 2060–2066, 2012.

[105] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[106] I. Rekleitis. Single robot exploration: Simultaneous localization and uncertainty reduction on maps (SLURM). In *Proceedings of the Conference on Computer and Robot Vision (CRV)*, pages 214–220, 2012.

[107] I. Rekleitis. Multi-robot simultaneous localization and uncertainty reduction on maps (MR-SLURM). In *Proceedings of the IEEE International Conference on Robotics and Biomimetics ROBIO*, pages 1216–1221, 2013.

[108] I. Rekleitis, V. Dujmović, and G. Dudek. Efficient topological exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 676–681, 1999.

[109] I. Rekleitis, A. New, E. Rankin, and H. Choset. Efficient boustrophedon multi-robot coverage: an algorithmic approach. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):109–142, 2008.

[110] A. Riva, A. Quattrini Li, and F. Amigoni. Some performance bounds on strategies for graph exploration. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, 2015.

[111] D. Rosenkrantz, R. Stearns, and P. Lewis. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal of Computation*, 6(3):563–581, 1977.

[112] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 2010.

[113] J. Sgall. Solution of David Gale's lion and man problem. *Theoretical Computer Science*, 259(1-2):663–670, 2001.

[114] T. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

[115] I. Shnaps and E. Rimon. Online coverage by a tethered autonomous mobile robot in planar unknown environments. *IEEE Transactions on Robotics*, 30(4):966–974, 2014.

[116] R. Siegwart, I. Nourbakhsh, and D. Scaramuzza. *Introduction to Autonomous Mobile Robots*. MIT Press, 2011.

[117] R. Sim and G. Dudek. Effective exploration strategies for the construction of visual maps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3224–3231, 2003.

[118] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 852–858, 2000.

[119] R. Smith, M. Schwager, S. Smith, D. Rus, and G. Sukhatme. Persistent ocean monitoring with underwater gliders: Towards accurate reconstruction of dynamic ocean processes. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1517–1524, 2011.

[120] C. Stachniss and W. Burgard. Exploring unknown environments with mobile robots using coverage maps. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1127–1134, 2003.

[121] C. Stachniss, O. Mozos, and W. Burgard. Efficient exploration of unknown indoor environments using a team of mobile robots. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):205–227, 2008.

[122] N. Stiffler and J. O'Kane. Shortest paths for visibility-based pursuit-evasion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3997–4002, 2012.

[123] S. Tadokoro. *Rescue Robotics*. Springer, 2010.

[124] S. Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*, pages 1–35. Morgan Kaufmann, 2002.

[125] B. Tovar and S. LaValle. Visibility-based pursuit – Evasion with bounded speed. *The International Journal of Robotics Research*, 27 (11-12):1350–1360, 2008.

[126] B. Tovar, L. Munoz, R. Murrieta-Cid, M. Alencastre, R. Monroy, and S. Hutchinson. Planning exploration strategies for simultaneous localization and mapping. *Robotics and Autonomous Systems*, 54(4): 314–331, 2006.

[127] C. Tovey and S. Koenig. Improved analysis of greedy mapping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3251–3257, 2003.

[128] G. Tuna, K. Gulez, and V. Gungor. The effects of exploration strategies and communication models on the performance of cooperation exploration. *Ad Hoc Networks*, 11(7):1931–1941, 2012.

[129] J. Urrutia. Art gallery and illumination problems. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. North-Holland, 2000.

[130] R. Vaughan. Massively multiple robot simulations in Stage. *Swarm Intelligence*, 2(2-4):189–208, 2008.

[131] A. Visser and B. Slamet. Including communication success in the estimation of information gain for multi-robot exploration. In *Proceedings of the International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 680–687, 2008.

[132] A. Visser, G. de Buy Wenniger, H. Nijhuis, F. Alnajar, B. Huijten, M. van der Velden, W. Josemans, B. Terwijn, R. Sobolewski, H. Flynn, and J. de Hoog. Amsterdam Oxford Joint Rescue Forces – Team description paper - RoboCup 2009. In *Proceedings CD of the RoboCup Symposium*, 2009.

[133] D. Wolf and G. Sukhatme. Semantic mapping using mobile robots. *IEEE Transactions on Robotics*, 24(2):245–258, 2008.

[134] K. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1160–1165, 2008.

[135] A. Xu, C. Viriyasuthee, and I. Rekleitis. Efficient complete coverage of a known arbitrary environment with applications to aerial operations. *Autonomous Robots*, 36(4):365–381, 2014.

[136] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 146–151, 1997.

[137] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the International Conference on Autonomous Agents*, pages 47–53, 1998.

[138] B. Zalik and G. Clapworthy. A universal trapezoidation algorithm for planar polygons. *Computers & Graphics*, 23(3):353–363, 1999.

[139] Q. Zhang, D. Whitney, F. Shkurti, and I. Rekleitis. Ear-based exploration on hybrid metric/topological maps. In *Proceedings of the*

*IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3081–3088, 2014.

[140] R. Zlot, A. Stentz, M. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3016–3023, 2002.