UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

Departamento de Arquitectura de Computadores y Automática

BÚSQUEDA DE OBJETIVOS MÓVILES EN TIEMPO MÍNIMO SOBRE ENTORNOS CON INCERTIDUMBRE.

MINIMUM TIME SEARCH OF MOVING TARGETS IN UNCERTAIN ENVIRONMENTS.

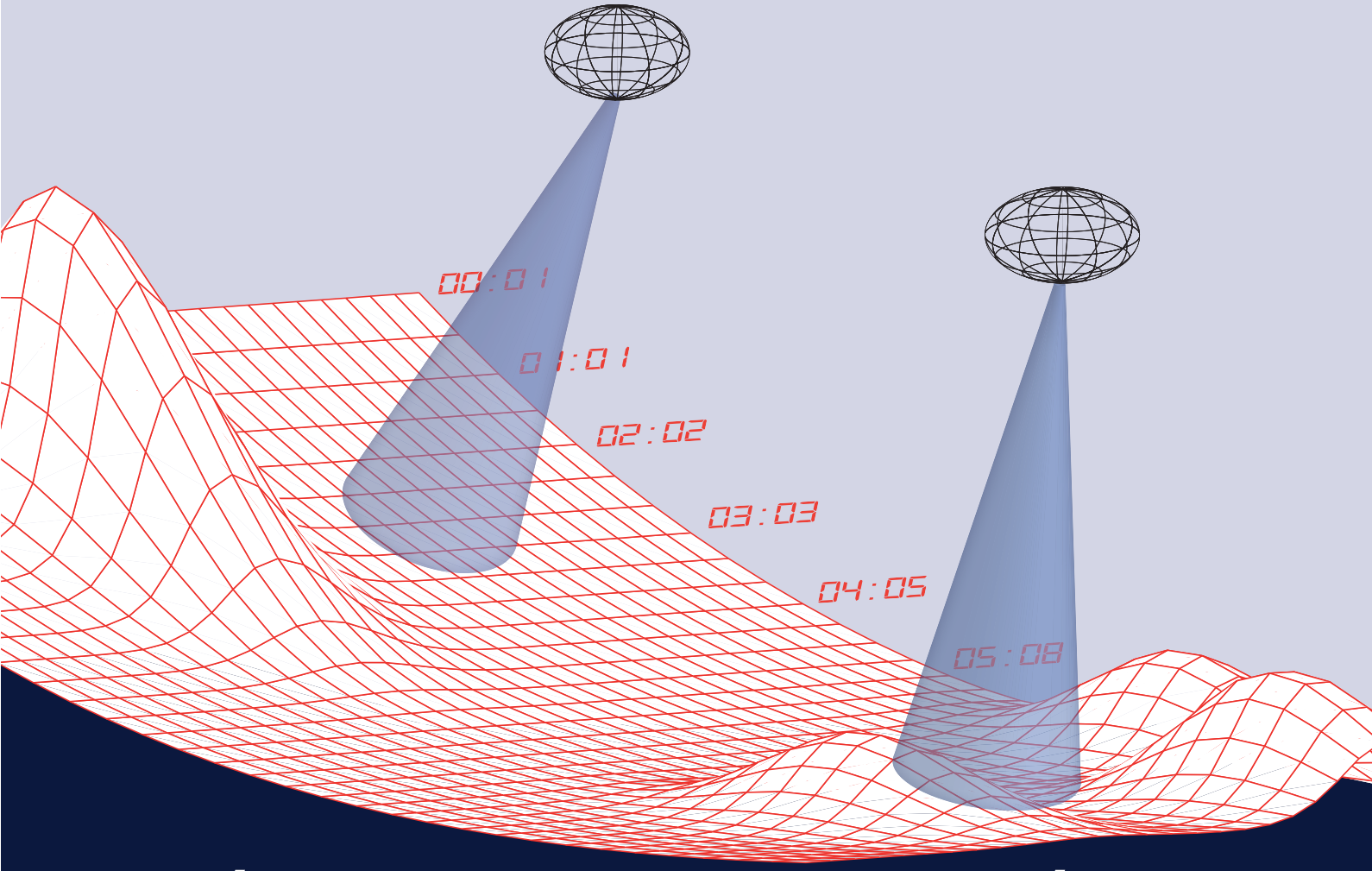
MEMORIA PARA OPTAR AL GRADO DE DOCTOR
PRESENTADA POR

Pablo Lanillos Pradas

Bajo la dirección de los doctores

Eva Besada Portas
Gonzalo Pajares Martinsanz
José Jaime Ruz Ortiz

MADRID, 2013



BÚSQUEDA DE OBJETIVOS MÓVILES EN TIEMPO MÍNIMO SOBRE ENTORNOS CON INCERTIDUMBRE

Minimum Time Search of Mobile Targets in Uncertain Environments

Pablo Lanillos Pradas

Departamento de Arquitectura de Computadores y Automática

Facultad de Informática

Universidad Complutense de Madrid

2013

UNIVERSIDAD COMPLUTENSE DE MADRID

MINIMUM TIME SEARCH OF MOVING TARGETS IN UNCERTAIN ENVIRONMENTS

by

Pablo Lanillos Pradas

A thesis submitted in partial fulfillment for the
degree of Doctor

Supervised by:
Eva Besada Portas
Gonzalo Pajares Martinsanz
José Jaime Ruz Ortiz

Systems Engineering, Control, Automatics and Robotics laboratory
Computer Architecture and System Engineering Department
Faculty of Computer Science

June 3, 2013

UNIVERSIDAD COMPLUTENSE DE MADRID

BÚSQUEDA DE OBJETIVOS MÓVILES EN TIEMPO MÍNIMO SOBRE ENTORNOS CON INCERTIDUMBRE

por

Pablo Lanillos Pradas

Tesis presentada para obtener el título de doctor

Dirigida por:
Eva Besada Portas
Gonzalo Pajares Martinsanz
José Jaime Ruz Ortiz

Laboratorio de Ingeniería de Sistemas, Control, Automtica y Robótica
Arquitectura de Computadores y Automática
Facultad de Informática

Ingeniera de Sistemas, Control, Automática y Robótica
Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid

Ph.D. Thesis
Minimum Time Search of Moving Targets in Uncertain Environments

Tesis Doctoral
Búsqueda de Objetivos Móviles en Tiempo Mínimo sobre Entornos con Incertidumbre

Author
Pablo Lanillos Pradas

Advisors
Eva Besada Portas
Gonzalo Pajares Martisanz
Jose Jaime Ruz Ortiz

ABSTRACT

This thesis is concerned with the development of an autonomous system to search a dynamic target in the minimum possible time in uncertain environments, that is, to solve the minimum time search problem, which is presented as an especial problem within the optimal search theory. This work proposes a Bayesian approach to find the target using several moving agents with constrained dynamics and equipped with sensors that provide information about the environment. The minimum time search involves two process: the target location estimation using the information collected by the agents, and the planning of the searching routes that the agents must follow to find the target.

The target location estimation is tackled using Bayesian techniques, more precisely, the recursive Bayesian filter. Moreover, an improved information filter, based on the extended Kalman filter, that deals with the team communication delays (i.e. out of sequence problem) is presented.

The agents trajectory planning is faced as a sequential decision making problem where, given the a priori target location estimation, the best actions that the agents have to perform are computed. For that purpose, three Bayesian strategies are proposed: minimizing the local expected time of detection, maximizing the discounted time probability of detection, and optimizing a probabilistic function that integrates an heuristic that approximates the expected observation.

To implement the strategies, three solutions are proposed. The first one, based on constraint programming, provides exact solutions in the discrete case when the target is static and the number of decision variables is small. The second one is an approximated algorithm stood on the cross entropy optimization method that tackles the discrete case for dynamic targets. The third solution is a gradient-based decentralized algorithm that achieves non-myopic solutions for the continuous case.

The minimum time search problems are found inside the core of many real applications, such as search and rescue emergency operations (e.g. shipwreck accidents) or pollution substances diffusion control (e.g. oil spill monitoring).

This thesis reveals how to reduce the searching time of a moving target efficiently, determining which searching strategies take into account the time and under which conditions are valid, and providing approximated polynomial algorithms to compute the actions that the agents must perform to find the target.

RESUMEN

Esta tesis aborda el desarrollo de un sistema autónomo para buscar un objetivo móvil en el menor tiempo posible sobre un entorno con incertidumbre, es decir, para resolver el problema de búsqueda de tiempo mínimo, que se presenta como un problema especial dentro de la teoría de búsqueda óptima. Se propone una solución Bayesiana para encontrar el objetivo utilizando varios agentes móviles con dinámica restringida provistos de sensores que proporcionan información del entorno. La búsqueda de tiempo mínimo involucra dos procesos: la estimación de la ubicación del objetivo a partir de la información recogida por los agentes que cooperan en la búsqueda, y el diseño de la planificación de las rutas que deben seguir los agentes para encontrar el objetivo.

La estimación de la ubicación del objetivo se aborda utilizando técnicas Bayesianas, más específicamente, el filtro recursivo Bayesiano. Además, se propone un filtro de información, basado en el filtro de Kalman extendido, que afronta el problema de los retrasos en la comunicación (problema de medidas desordenadas).

La planificación de las trayectorias de los agentes se plantea como un problema de decisión secuencial donde, a partir de la estimación de la ubicación del objetivo, se calculan las mejores acciones que los agentes tienen que realizar. Para ello se proponen tres estrategias Bayesianas: minimización del tiempo local de detección esperado, maximización de la probabilidad de detección descontada por una función dependiente del tiempo, y optimización de una función probabilística que integra una heurística que aproxima la observación esperada.

Para implementar las estrategias se proponen tres soluciones. La primera, basada en la programación con restricciones, ofrece soluciones exactas para el caso discreto cuando el objeto es estático y el número de variables de decisión pequeño. La segunda es un algoritmo aproximado construido a partir del método de optimización de entropía cruzada que aborda el caso discreto para objetos dinámicos. La tercera es un algoritmo descentralizado basado en el método del gradiente que calcula decisiones en un horizonte limitado, teniendo en cuenta el futuro, en el caso continuo.

Los problemas de búsqueda de tiempo mínimo se encuentran en el planteamiento de muchas aplicaciones reales, como son las operaciones de emergencia de búsqueda y rescate (p.e. rescate de náufragos en accidentes marítimos) o el control de la difusión de sustancias contaminantes (p.e. monitorización de derrames de petróleo).

Esta tesis muestra cómo reducir el tiempo de búsqueda de un objeto móvil de forma eficiente, determinando que estrategias de búsqueda tienen en cuenta el tiempo y bajo qué condiciones son válidas, y proporcionando algoritmos polinómicos que calculen las acciones que los agentes tienen que realizar para encontrar el objeto.

Para
María, porque con ella ya no queda nada por buscar
más que su sonrisa

Para
Mi familia porque siempre les encuentro

"It's not hard to make decisions
when you know what your values are"
Roy O. Disney

Acknowledgements

This thesis was supported by the Spanish Ministry of Education and Science (MEC) under the research grants DPI2006-15661-C02-01 and DPI2009-14552-C02-01. Some of the research has been hosted by the Aerospace Controls Laboratory (ACL) at the Massachusetts Institute of Technology (MIT), the Artificial Intelligence Laboratory (LIA) at the École Polytechnique Fédérale de Lausanne (EPFL) and the Australian Centre for Field Robotics (ACFR) at the University of Sydney.

I would first like to thank Dr. Eva Besada for being my adoptive advisor, partner and for your inestimable help at the problem discussion and paper writing and corrections. The sole person able to convert a black and white text into a full of colors art painting correction. Your obsession with notation, subscripts and superscripts has made the thesis grow up from the childhood to the mature age. Thanks to my co-advisors Dr. José Jaime Ruz and Dr. Gonzalo Pajares for your support, advises, ideas and for making me being down to earth during this intellectual trip. I would also like to thank the ISCAR research group director, Dr. Jesús Manuel de la Cruz, for giving me the opportunity to work within his lines and for your invaluable understanding and clever contributions.

When I first got into the Department of Arquitectura de Computadores y Automática (DACYA) as a graduate student the seed of an incredible crew of shavelings (mancebos) was spread. This prepared and curious team of scholars was timelessly formed by: the philosopher Carlos, the smiley David, the ironic Joaquin, the gadget Santiago, the hardware Hector, the multi-purpose David, the unknown occupation Chechu and myself. Now everyone is heading a different path due to the difficulties that the public university is going through, but all the lunches and life discussions are unpayable. For your friendship and for all the time spent without talking in front of our computers, thanks.

Thanks also to the DACYA professors and the ISCAR lab people for all the political-crisis debates and specially to: Juan Fran for your experience and life style that got to name a special sandwich, Bonifacio for his goodness and José Miguel for sharing the late afternoon working days at the office.

I am so grateful to the foreign research labs for taking me in with open hands. Thanks to the ACL lab at the MIT for showing me how high quality research can be done with money and good government intentions, and specially to Luca Bertuccelli for explaining me the real science method and for opening my eyes into this fascinating field in computer science. Thanks to the LIA lab at the EPFL for your kindness and support and showing me how people of different countries can work together to improve science, and specially thanks to Boi Faltings, the intelligent and kind LIA research director. Special thanks to Radoslav Szymanek for your knowledge and for your economics seminars. Finally I would like to thanks the ACFR lab at the Sydney University for being a second family.

Admirable work of Salah Sukkariesh piloting a ship full of incredible researchers and engineers. Seng Keat Gan (Jason), it was a pleasure to share the work, learn from your splendid knowledge and for being an excellent partner. It is right to mention, in order of appearance, all the collateral people that have accompanied me sentimentally after the sunset in those stays: the Swiss Till and Philipe, Lanie the artist; Lausanne crew Joseba, Laura and Julia; the Aussie family Ivan, Eloise littlefinger, my cubicle and football mates, etc.

Never forgetting, and they already know, one of the pillars, which helped me to get this thesis done: my friends. Thanks to the snotty gang for still singing in Sotillo de Ladrada. I also appreciate so much my regular mates to whom it is impossible to give them back all the good things they gave me. In gender order: the little girl of tazmania and the highschool-postschool inseparable sharing moments; black eyes chinese girl for the happiness, hits and hugs; multinamed first love letter for amazing trips although she went with the queen; Northern catwoman for making me a mature sentimental lover; little fish and her irrational support dancing; geopeople and their boring cartography; the micronova and biology girls for their understanding; stain human being and his friendship love everyday; indian surfer and the non-stop research; the funniest and his never lost proximity even being far away; cousin for being a cousin; hair lotion man, fixing boy and wevona, and the life lines that crossed our destinies; fuenla boys for the fresh air; Erasmo of Rotterdam and all the related, etc.

Special thanks to Ms. Ruggiero, you are my compass, my muse, my light, an outstanding person, who have guided me through this period of life. This thesis has a huge part of your love patience.

Finally I would like to thanks in a great manner my family for their genes and because they have been always encouraging me to be an improved person and to push my desires beyond limits. Within the classical definition of parental love, thanks to my mother for your support throughout my whole life, for feeding me and for loving me, and thanks to my father for teaching me everything that I know (well almost everything).

Contents

Preface

From the free will to engineering optimization

Wednesday, October 12th. The amygdala has started to work reacting to some contextual stimulus. The working memory has been partially activated and the cingulate cortex has begun to monitor the activity. The prefrontal cortex, with boasting technocrat authoritarianism, has taken the control. It is fusing the sensing, emotional and mnesic information giving way to make a motor decision. There is no doubt, I have lost the keys again, I'm late and I don't know where to start the search. I evaluate the alternatives. I usually leave them at the entrance hall, hanging from that screw that I put in a provisional eternal way. They can also be in the bedroom, inside the Cairo bowl or inside the jeans pocket, which I have used yesterday. Rationally, the most coherent action is to look first at the hall, but in the case that the keys are not there, I have to come back and that is a great time-wasting. I only have ten minutes left to pick up the high speed train to Barcelona. Therefore, I finally decide to watch in the bowl first and afterwards inside the jeans, and if I don't find them I will look at the screw, which is on my way out.

Already heading Barcelona, inside the train and without the keys, I think about my decision. I really don't know if my searching strategy has been optimal, neither if my reasoning has been determined by my subconscious brain. I would like to launch a question: do I have decided by reasoning or my brain had already the answer, conditioning my decision? Antonio Damasio and John-Dylan Haynes do not have doubts about this issue. The decision cognitive process "is just the tip of the iceberg". Therefore, my brain already knew which decision was optimal. My wandering inevitably took me to an obvious and realistic conclusion. The brain connections, developed during the years by means of learning, and using the senses as the interface with the world, are the ones that determine the most part of the decisions. I try to remind a high school biology class where the teacher explained this process in a simplified way. Of course, it can be demonstrated through pavlovian conditioning, or in other words with the stimulus-response theory. If a reward is obtained when an action is realized, thanks to repeatability we learn which is the most convenient decision alternative.

Already satisfied with my reasoning and served with a coffee at the dining car, I write down the different options that I had this morning to prove that I have made the correct decision. The problem started with the subjective information about the keys location: high probability of finding them at the hall and medium probability of finding them in the jeans and the bowl. If I can quantify that information, the only thing I have to determine is which searching path is the best. Thus, I need the actions sequence that gives maximum probability of finding the keys, which is the same to look for the

sequence that gives minimum probability of not finding them. I raise my gaze and I take pleasure of my own explanation. The train car is getting full of people, maybe because it is lunch time. I stare again at my sheet full of hieroglyph and I continue engrossed in my task. After doing some math, I check that ordering the places from greater to less probability and visiting them according to that order could be the best strategy. While I bite my green pen with anxiety, I discover that the method doesn't always work and more questions arrive without an answer: What happens if the time is critical? And what about if already choosing a decision the alternatives change?... But the fact that make me feel unease is to think that even making the right decisions, why haven't I found the damned keys?

Once in my seat, with the eyes wide shut and a little bit sleepy, I receive a message on my cellphone that says: - remember that I have taken the keys to water the plants. When you come back, please call me. Kisses-. With a half-smile I curl up in the seat.

The adrenaline has dropped and the somatotropin has risen. The synaptic connections used that day are reinforcing. There is no doubt, I am falling asleep.

Prefacio

Del libre albredío a la investigación operativa

Miércoles 12 de octubre. La amígdala se ha puesto en funcionamiento reaccionando a los estímulos contextuales. La memoria de trabajo se ha activado parcialmente y la corteza cingulada ha comenzado a monitorizar la actividad. En un alarde de autoritarismo tecnócrata, la corteza prefrontal ha tomado el mando. Está integrando la información sensorial, emocional y mnésica para dar paso a una decisión motora. No hay ninguna duda, otra vez he perdido las llaves, llego tarde y no sé por dónde empezar a buscar. Evalúo las alternativas. Normalmente las dejo en el recibidor, colgadas de aquel tornillo que puse de manera eternamente provisional. También podría estar en la habitación, en el cuenco del Cairo, reutilizado como llavero, y en los pantalones que use el día anterior. Razonadamente, lo más sensato es mirar en el recibidor, pero sino está allí, y tengo que volver, es una pérdida de tiempo enorme y me quedan 10 minutos para coger el tren de alta velocidad a Barcelona. Así que decido mirar en el cuenco primero, luego en los pantalones y si no lo he encontrado miro en el tornillo, que además, está de paso a la entrada de la casa. Ya de camino a Barcelona y sin llaves, reflexiono acerca de mi decisión. Mi duda no está solo en si la estrategia que he seguido era óptima, sino si mi razonamiento ha sido dictado por un determinista cerebro inconsciente. Lanzo una pregunta al aire: realmente he decidido de forma razonada o mi cerebro ya tenía la respuesta, condicionando mi decisión. Antonio Damasio y John-Dylan Haynes no dudarían en dar una respuesta. El proceso cognitivo de decisión consciente "es solo la punta del iceberg". Por lo tanto mi cerebro ya sabía cuál era la decisión óptima.

Mis divagaciones me llevan inevitablemente hacia una conclusión tan obvia como realista. Las conexiones cerebrales, formadas a lo largo de los años por medio del aprendizaje, y usando los sentidos como interfaz con el mundo, son quienes determinan la mayor parte de las decisiones. Intento recordar una clase de biología de segundo de bachiller donde explicaban ese proceso de forma sumamente simplificada. Por supuesto, se puede demostrar con el condicionamiento pavloviano, más conocido como comportamiento Estimulo-Respuesta. Si se obtiene recompensa por realizar una acción, gracias a la repetividad aprendemos que esa alternativa es la que más nos conviene.

Ya satisfecho con mi argumentación y acompañado de un café en el vagón restaurante, anoto en un papel las diferentes opciones que tenía por la mañana, para demostrar que realmente era la solución correcta. El problema partía de la información subjetiva de donde podrían estar las llaves: alta probabilidad de encontrarla en el recibidor y probabilidad media de encontrarla en los pantalones y en el cuenco. Si puedo cuantificar esa información, lo único que tengo que pensar es cuál de los caminos en las alternativas es el más indicado. Es decir, necesito la secuencia de acciones que me den máxima

probabilidad de encontrar las llaves. O lo que es lo mismo, las que me den mínima probabilidad de no encontrarlas. Levanto la mirada y me regodeo en mi propia explicación. El vagón se está llenando de gente, posiblemente es la hora de comer. Vuelvo a enfocar mi hoja llena de jeroglíficos y continúo ensimismado mi tarea. Después de hacer algunos números, compruebo que ordenar los lugares de mayor a menor probabilidad y visitarlos acordemente puede ser la mejor estrategia. Mientras muerdo el bolígrafo verde con ansiedad, descubro que ese método no siempre funciona y que hay varias preguntas todavía sin respuesta: ¿Y si el tiempo es crítico?, ¿Y si tomando una decisión las alternativas cambian?,... Pero lo que más desasosiego me produce es pensar que habiendo tomado una decisión óptima, por qué no he encontrado las malditas llaves.

Ya en la butaca con los ojos medio cerrados y un poco somnoliento recibo un mensaje que dice: -recuerda que te he cogido las llaves para regar las plantas. Cuando vuelvas llámame. Besos-. Con media sonrisa me recojo en el asiento.

La adrenalina ha bajado y la somatotrofina ha subido. Las conexiones sinápticas más utilizadas se están reforzando. No hay ninguna duda, me estoy durmiendo.

Thesis Structure

This thesis is written for anyone interested in learning decision making in probabilistic environments. In order to follow that purpose, there is no need to read extra-literature to understand the definitions and the algorithms presented here. All the information needed is included through these pages. The same philosophy is applied to each chapter, where the information is self-contained. This is useful for manuscript reusing, and for lazy or in a hurry readers that only want to deal with a specific chapter in depth. A full thesis reading will recognize some redundant definitions at each chapter, but with no doubt, this helps to consolidate the concepts and avoids undesirable jumping through references.

The first chapter is the well know introduction that presents the objective of the thesis, the approach followed to tackle the problem and the main contributions. Second chapter provides an historical background of the problem, analyzes the possible approaches depending on the research field, states the autonomous system design and contributes with a comparison between the related work to deal the Minimum Time Search (MTS) and the proposed in this thesis. Third chapter condenses the main thesis contributions: the definition of the MTS, applications and some examples; the demonstration of the MTS as a special problem of the optimal search; the framework and the items involved in the problem; the multiagent information decision filter for dynamic targets; the decision making strategies to tackle the time optimization; and the possible optimization approaches.

The following two chapters contribute with four algorithms, which complement and improve the current state of art methods, to solve the MTS. Chapter four focuses on the discrete MTS problem providing two algorithms: an optimal solution based on constraint programming and a stochastic algorithm approximation to deal with large scale discrete MTS and dynamic targets. Chapter five addresses the continuous MTS and proposes an online non-myopic optimization method and an algorithm that tackles the delays within the team communication. Both approaches are accompanied by an exhaustive result analysis.

Finally, chapter six concludes and summarizes the thesis, emphasizing the results achieved and the questions that are still opened at the Minimum Time Search Problem.

Figure 1 shows the thesis structure from the conceptual point of view.

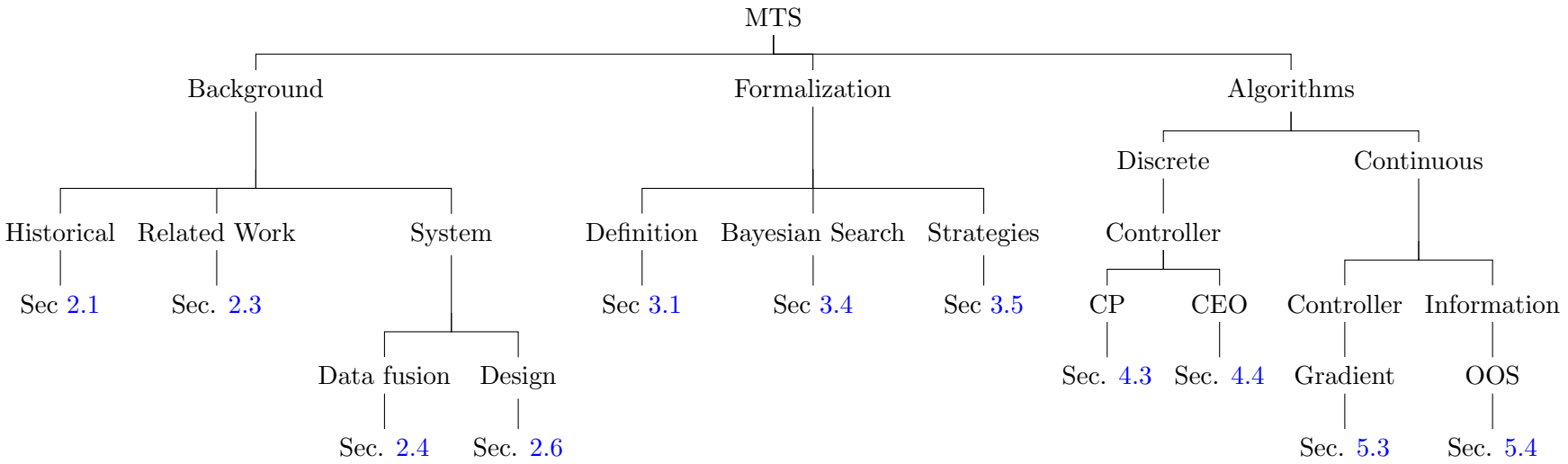


FIGURE 1: Thesis Structure Graph.

Chapter 1

INTRODUCTION

"In each action we must look beyond the action at our past, present, and future state, and at others whom it affects, and see the relations of all those things. And then we shall be very cautious" Blaise Pascal

1.1 Introduction

The 5th of June of 1968, the USS Scorpion submarine got lost in the Atlantic sea while doing tactical maneuvers [ISBA, 2009]. It sank in some location within the 4000 km that separates the Azores islands and Norfolk city in United States. The US navy searched the region for five months, unfortunately, without success. Meanwhile, the chief scientist of the special projects division and his crew developed a new interesting approach based on a previous searching method performed to find the H-bomb in Palomares (Spain). Using acoustic data and taking into account the probability of the possible submarine failures that caused the accident, they built a probability map of the possible locations. Figure 1.1 shows an illustrative example of this type of maps where each cell expresses the associated value probability by its color, that is, the more intense is the color the greater is the probability. Following that map, the submarine was found in only five days, surprisingly just 200 meters away from the location of greatest probability. This illustrative example permit us to glimpse the potentiality of using probabilistic information within a problem that involves uncertainties, although the search of a lost target implies much more than building the initial probability map.

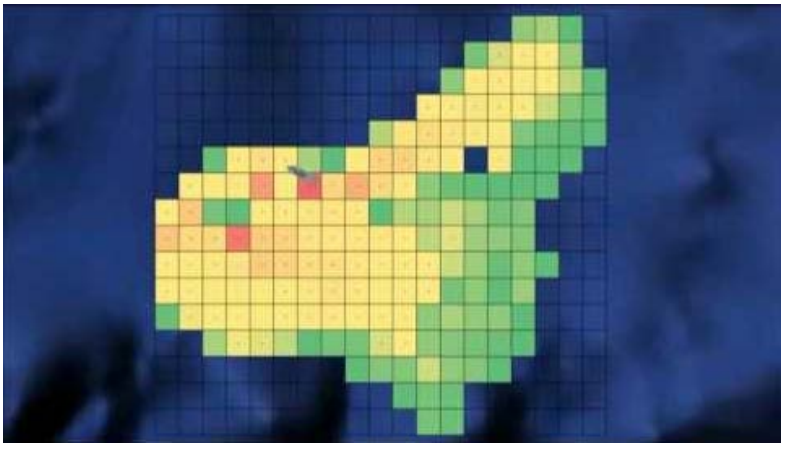


FIGURE 1.1: USS Scorpion submarine location probability map [ISBA, 2009].

To solve the whole problem, we also have to provide the optimal observation plan, that is, the visiting order of the possible locations to find the target as soon as possible. This problem is not naive and gets more complex when the searcher has spatial restrictions[1], we count with multiple searchers and, moreover, the target is moving.

Since the beginning of the human being, we have tried to perform any task faster and more efficiently, and with the arrival of the computers and the algorithmic theory, some of these tasks have been reproduced and even improved. None gets surprised when a computer chooses which stocks should be bought at the market, or if the plane trajectory

[1]When we talk about spatial or path restrictions we mean that we cannot observe any location when we want, because the location of the agent depends on the searcher dynamics. In this way, we differentiate the density problem [Stone, 1989] from the searching problem with constrained path [Eagle, 1984].

is computed by an algorithm [Pajares et al., 2008]. Nowadays, the searching of lost target task is still performed by humans (e.g. search and rescue sea operations [Kratzke et al., 2010]), but thanks to modern robotics, its miniaturization and cheaper sensors and actuators, we are close to automate the searching task. For instance, we can imagine the advantages of counting with autonomous vehicles helping in mountain or sea rescue operations.

If we want to transform the searching problem in a totally autonomous process we need first, an information management system that updates the target location knowledge and secondly, a decision system that chooses the best actions to find the target. Both systems have to deal with the uncertainty associated with the target location and its changes according to the time lapse [Jaynes, 2003], as well as, permit the incorporation of new information coming from the searchers observations. Moreover, when we have a team of searchers (agents[2]), we need to deal with the communication and information coupling between them, and we have to permit their cooperation to pursuit a common objective: finding the target in the minimum time possible.

Lets analyze the problem through an illustrative example to identify the elements involved and explain better the aim of this thesis. Imagine that we have lost the car keys (target) in our home and we are on a hurry because we have to drive to the airport to get a plane. Our objective is to find the keys in the shortest possible time and our searching region (world) is the house. We have a belief or prior information of the keys whereabouts: we usually leave them on the entry hall table or inside the trouser pocket that we keep in the bedroom. With this information we can build a probability map that we will use to drive the searching, and every time we observe any room of the house, we will include that information in our keys location knowledge. That is, if we look in the entrance and they are not there, we can discard that room and focus on the rest of the places. It can also happen that someone is cleaning the house in that precisely moment, changing the objects from one place to another, and therefore modifying the keys location. We should be able to predict those movements. Moreover we have restrictions in our searching path, like, to go from the living room to the entrance we have to cross the kitchen. To move from one place to another we perform actions like, for instance, from the living room go to the left to enter in the kitchen or to the right to access the bathroom. Besides, when there are several people searching, by sharing out the rooms and communicating our observations we will improve our success possibilities and we will also reduce the searching time. Taking into account all these elements, us, the agents, we should plan the optimal sequence of actions that make us find the keys in the minimum time possible.

This thesis concerns with the design of an intelligent and autonomous system with the aim of finding a target of unknown location in minimum time, or in other words, studies

[2]The word agents is used in this thesis to generically refer to the searcher, whether it is a vehicle, a person or a mobile sensor. To refer to many searchers we use the word multiagent.

and proposes a solution for the Minimum Time Search (MTS) problem with multiple agents. There are two main motivations for the realization of this work. The first one is framed within the projects DPI2006-15661-C02-01 and DPI2009-14552-C02-01 [DPI, 2006, 2009], where the cooperation of aerial and sea autonomous vehicles to obtain quick responses in sea emergencies is set up. Some scenarios are considered such as: diffusion control of polluting substances [Clark and Fierro, 2005; Lanillos et al., 2009] and shipwreck people rescuing [Bourgault et al., 2003]. The autonomous vehicles have the ability to communicate between them to collaborate in performing the tasks in the minimum time possible. The second motivation is risen by the lack, to the best of the authors knowledge, of previous effective decision making method to minimize the searching time when the agents dynamics are restricted and the target is moving. Therefore, in the present work, not only do we provide a solution to the MTS problem, but we also manifest the necessity of improving existing methods to fulfill our problem formulation.

To understand the difficulty to design efficient methods and why the time minimization is not contemplated among them, a little historical review of the problem is needed. The search of an object with uncertain location has been widely studied since 1946, when the work "Search and Screening" was published [Koopman, 1946]. In 1975 the researchers have already solved with beauty two complementary objectives of the search: detect the target with (1) minimum cost and (2) minimum time. However, this approach had one important drawback: they assumed that the space is infinitely divisible, i.e., they did not take into account the agents dynamics or the spatial restrictions to go from one localization to another. A few years later, a doctor in the naval engineering called Eagle published a mathematical model where he added the searching path restrictions [Eagle, 1984]. After this work, the approaches presented will be more oriented in accumulating the information about the target position than in minimizing the searching time. He stated, based on [Smallwood and Sondik, 1973], that the problem is a Partial Observable Markov Decision Process (POMDP) and therefore it can be solved using dynamic programming techniques. Although his solution was not scalable, he set out the grounds for designing some subsequent branch and bound algorithms [Eagle and Yee, 1990; Washburn, 1998]. Naturally, it was demonstrated that the algorithmic complexity was NP-complete or NP-hard depending on the problem instantiation [Trummel and Weisinger, 1986].

Some time after, and with the drones and unmanned vehicles fever [Ross, 2011], a new impulse appears on the optimal search, where instead of one searching agent, there are several [Yang et al., 2002; Bourgault et al., 2004; Clark and Fierro, 2005; Gan and Sukkarieh, 2010]. Then, the word agent begins to be used to design every mobile sensing platform involved in the search. Although, the multiagent setup rises the complexity to NEXP-complete [Bernstein et al., 2002], simplifying the communication and synchronization of the information that the team knows about the target, the problem is

partially solved when the target remains static [Mathews et al., 2007]. All these works approach the problem from the decentralization point of view, i.e., each mobile agent makes its own decisions cooperating as a team to find the target. Decentralization, apart from providing stability, modularity and redundancy [Bourgault et al., 2004], provides better searching results and adds robustness to the system. Despite of these advances the decision problem is not totally solved, a reason that motivates in part the approach that is proposed in this work.

On one hand, the majority of the mentioned research works forget one of the fundamental objectives of the original optimal search [Stone, 1975]: minimizing the target detection time. In fact in [Trummel and Weisinger, 1986] two theoretical formulations to approach the problem are mentioned: maximizing the probability of detection and minimizing the detection time. In this thesis we will see that, although both strategies are feasible for searching lost targets, they do not achieve the same result. Anyway, only the first strategy has been widely researched [Eagle and Yee, 1990; Bourgault et al., 2003; Yang et al., 2004; Lavis et al., 2008; Mathews et al., 2007; Gan and Sukkarieh, 2010], mostly due to the high complexity of the second [Bourgault et al., 2003]. Recent research works [Hollinger et al., 2009; Sarmiento et al., 2009] have approached the searching time reduction problem, although they have not efficiently solved the multiagent scheme, as it has been successfully achieved in ground surveillance tasks [Anisi et al., 2010]. These reasons motivate us to approach the multiagent searching problem from the time minimization point of view. In fact, we present the MTS as a special problem within the optimal search, which requires the development of novel decision strategies that minimize the time. Moreover, we analyze when the strategy of maximizing the probability of detection is feasible to minimize the searching time.

On the other hand, it exists an efficiency problem or too many assumptions in the methods found to compute the agent actions. Although it is true that, due to the high complexity, a tractable global optima method does not seem to exist, there are some approximations to tackle it such as: greedy algorithms [Bourgault et al., 2003; Yang et al., 2002, 2004], local optimizations [Gan and Sukkarieh, 2010; Mathews et al., 2007; Tisdale et al., 2009; Sarmiento et al., 2009] and enumeration of all the solutions [Hollinger et al., 2009]. The greedy methods and the local optimization are blind[3] with respect to future decisions. Moreover, most of the methods assume that the target is not moving or that there is only one agent searching. Therefore, in this work we also lay out the necessity of improving previous methods, and we propose algorithms and more intelligent strategies that reduce the solutions locality and provide more precise decision during the search, accepting multiple agents and targets with its own dynamics.

Summarizing, this thesis contributes to the multiagent searching problem with: 1) the identification of the MTS as a differentiated problem within the optimal search, as well

[3]The blind or myopic strategies are the ones that only take into account the actions effect in a bounded decision window, and therefore, obviate the effect that these actions could produce in the future.

as a totally probabilistic unified formalization and notation for evaluating the agents decisions; 2) the development of new strategies that actually minimize the time and reduce the myopicity; and 3) optimization algorithms that improve efficiency. Besides, we present the design of the general autonomous system to perform the MTS tasks.

1.2 Objective

The general objective of the thesis consists in designing an autonomous system that finds a lost object in the minimum time possible. The system is compounded by a team of mobile agents equipped with sensors that observe the world recollecting information about the target location. The information of the possible target location is used by the agents to plan the actions that guide them in the task achievement. In particular, these agents are aerial autonomous vehicles with vision perception modules able to detect the target that hover over the searching region. Figure 1.2 shows an schematic example of the problem, where a vehicles team observe the region and cooperate to choose the best searching paths. Meanwhile, the target, whose location is uncertain, moves within the world without considering the agents presence.

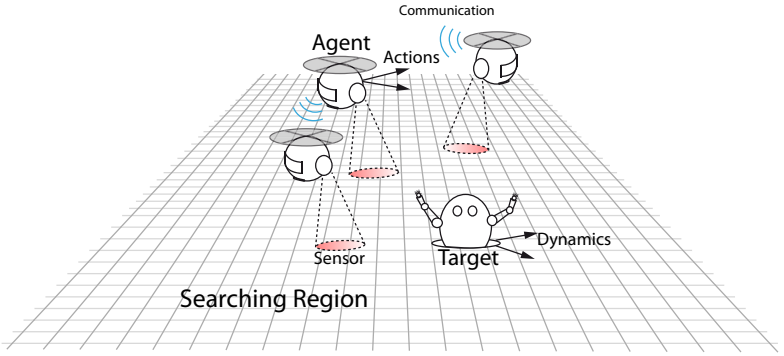


FIGURE 1.2: Decentralized MTS configuration.

Achieving this objective we approach the practical application proposed at the projects [DPI, 2006, 2009] that motivate this thesis. In the case of shipwreck rescue operations [Bourgault et al., 2004], the searching team are aerial vehicles (quadrotors) equipped with visual perception modules that permit the shipwreck person detection. The objective is to locate the survivors in minimum time for their posterior rescuing. Using the sea currents and the superficial wind information, and taking into account the initial accident location, we can build a probability map of the possible location and infer its movements. The vehicles, taking into account their dynamic restrictions, calculate the best trajectories to locate the shipwreck person and when they discover it, they send their position to a command center that is in charge of managing and coordinating the rescue mission. In [Lanillos et al., 2009], a similar strategy is applied to polluting

substances diffusion control, that is, instead of searching shipwreck survivors, like the previous application, we look for contaminated locations.

As it can be deduced from the previous applications, the autonomous system must, on one hand, predict and update the target location and, on the other hand, choose the best agents actions. Particularly, as well as the general searching problem, in the present work we tackle the decision making issue, or in other words, the planning of the best team actions, given the a priori target location information. Thus, the main objective of this work can be summarized as follows:

Determining the best sequence of actions to find a target (object) with unknown location in the minimum time possible.

Two different system configurations are considered depending whether the decision making is approached in a decentralized or centralized manner. In the decentralized case, schematized in Figure 1.2, the agents broadcast the information within the team and cooperate to compute the best actions. On one side, each agent is in charge of computing its own actions taking into account the information received from the rest of the team. On the other side, it broadcast its observations and the actions that it has planned in order to minimize the searching time of the team.

In a centralized system configuration, as it is shown in Figure 1.3, the mobile agents, represented as aerial vehicles, send the recollected information to a central system that updates and computes the best actions. Afterwards the actions are sent to the agents for their execution. The task finalizes when any of the agents detects the target. The system is, therefore, a central station that controls a team of agents able to observe the environment and communicate.

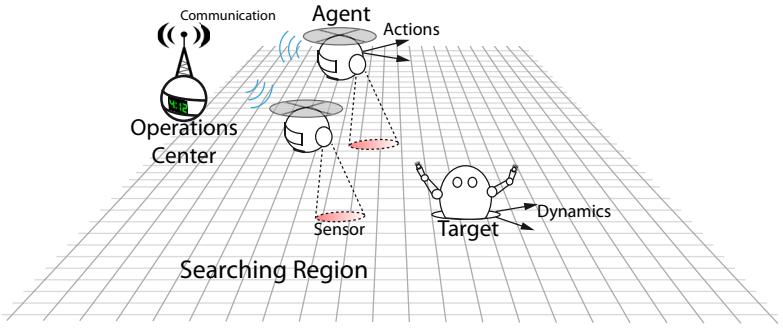


FIGURE 1.3: Centralized MTS configuration.

Challenge: Uncertain target location + moving target + multiples agents + optimal time.
Main Objective: Designing algorithms and strategies that decide the best team actions to detect an object with uncertain location in minimum time.

1.3 Approach

In this section we characterize the MTS with its elements and properties, we analyze the problem from the decision making framework and we explain the general design of the autonomous system and how we have approached its divers components.

At the MTS there are two partakers: the mobile agents that have the ability to observe and the searching object (target). For the agents, the target position and movements are uncertain. However, although the agents do not known the exact target location, they count with the information of the possible locations where it can be found initially (a priori information) and the probabilistic target motion model. Besides, the agents initial position and their actions (which make the agents displace in a deterministic way) are known. Finally, we consider that the searching region (world), where the target is placed and that the agents observe, is finite and delimited.

The MTS solution, from the decision making point of view, is the sequence of actions that the team of agents have to perform to detect the target in the minimum possible time. This decision making process has to take into account the following: 1) each time that an action is performed there is a new observation that changes the team target location belief, affecting future decisions; 2) the time and actions execution order matters; and 3) the search finalizes when the target is found (detected).

To perform the MTS task we propose an autonomous and intelligent system based on the works of [Yang et al., 2002; Bourgault et al., 2004; Mathews, 2008]. This system, described in an abstract way in Figure 1.4, distinguishes two differentiated layers: the sensory data fusion layer, which updates the target location belief with the new observations, and the controller layer[4], which computes the agents actions using the data fusion layer information, and the target motion and observation models. In the case of a centralized system, the data fusion and decision layers cannot be found at the agents but inside an operations center, and the agents only have the ability of moving, sense, and communicate (send the observations and receive the actions). Within a decentralized system, each agent counts with its own data fusion and decision layer that are synchronized in a transparent way. Therefore, both layers communicate with the rest of the team to update the target location and the actions planned by the rest of the agents. Thus, the decentralized system becomes an network of agents that cooperate in the search.

The system working process, in a simplified way, is the following: the sensors obtain measurements and send them to the data fusion layer, where the target location estimation is predicted and updated; the controller layer chooses the best actions to find the

[4]The controller layer is actually a planner that, given the target information, computes the best actions that each agent should perform. Because it works as a high level open loop controller [Mathews, 2008], we will use the words controller, planner and optimizer equally to refer to the decision layer.

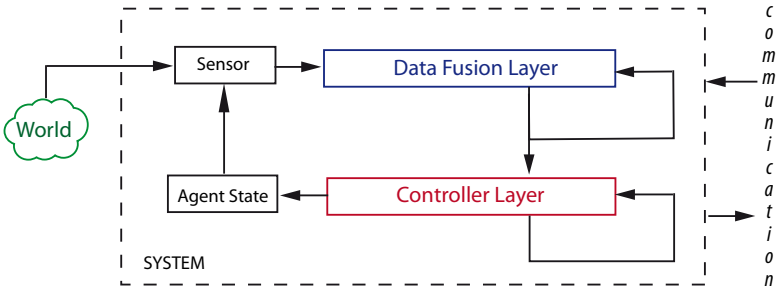


FIGURE 1.4: General system design to tackle the MTS.

target based on the target location estimation; the agents execute the actions, displace themselves over the searching region and make a new set of observations; finally the observations feed again the data fusion layer and the cycle is repeated.

The present work is fundamentally focused on the controller layer, although we also propose improvements at the data fusion layer when the observations (sensor measurements) arrive disordered [Bar-Shalom and Chen, 2005] due to the delays introduced by the communication network. These delays deteriorate the target location estimation and consequently make the agents decisions deficient. Therefore, we need an algorithm able to deal with this problem, improving the target location estimation and synchronization.

The design or definition of the controller layer is a complex problem that can be approached from different computer science disciplines such as: combinatorial optimization [Berger et al., 2009; Sarmiento et al., 2009], POMDPs [Eagle, 1984; Kaelbling et al., 1998; Hsu et al., 2008], control theory [Smallwood and Sondik, 1973; Bernstein et al., 2002; Furukawa et al., 2006], data fusion [Lavis et al., 2008; Bourgault et al., 2004], information theory [Yang et al., 2004; Kagan and Ben-Gal, 2006], and probability theory [Yang et al., 2002; El-Mane Wongy, 2005; Bertuccelli and How, 2006]. In this work we approach the problem from a probabilistic and Bayesian point of view [Jaynes, 2003]. On one hand, to exploit the previous information and, on the other hand, to support an intuitive design of the strategies that solve the MTS. We will always take into account the tractability and viability of the final strategies and algorithms for online execution.

We face the problem through two different representations: (1) discrete, where the searching region is described as a grid and the location and actions take discrete values; and (2) continuous, where the locations and actions take real values. Both problems are non-convex and they are solved by means of discrete combinatorial optimization and piece-wise continuous optimization respectively.

1.4 Principal Contributions

The principal contributions of the thesis are classified in three blocks. The first block is the problem formulation, where we state that the MTS is a problem, to the best of the

authors knowledge, not totally solved. In fact, the majority of the works cited in this chapter do not necessarily minimize the searching time, or handle the multiagent moving target problem. Moreover, we provide an unified Bayesian notation for the decisions evaluation. The second block tackles the design of strategies that really minimize the target detection time. Finally, the third block consists in algorithms that improve, in combination with the strategies, the cited methods to solve the MTS problem in either discrete and continuous representation.

1.4.1 Minimum Time Search Formulation

We propose a necessary condition that the strategies have to fulfill to minimize the target detection time when the searching path is constrained. This condition, not fulfilled by all the existing methods, makes us formalize the MTS as a special problem within the optimal search theory, with its own characteristics and strategies. In this sense, we propose a Bayesian unified formulation for multiagent systems that search dynamic targets. This contribution extends [Eagle and Yee, 1990; Yang et al., 2004; Bourgault et al., 2004; Mathews, 2008] and permits the development of novel Bayesian strategies that minimize the time.

These contributions are described in Sections 3.1, 3.3 y 3.4 of Chapter 3, and they have been published in the following articles: [Lanillos et al., 2012, 2013]

1.4.2 MTS Strategies

We contribute with three tractable strategies to tackle the MTS problem: minimizing the Local Expected Time (LET), where the expected time is reformulated as a receding horizon utility function with a bounded decision window; maximizing the Discounted Time Reward (DTR), where a discounted time function weights the agents accumulated information, given more importance to early decisions; and the Discounted Time Heuristic (DTH), which uses a novel heuristic modeled as a sensor to reduce any strategy myopicity. These three strategies improve the ones found in the literature [Eagle, 1984; Yang et al., 2002; Bourgault et al., 2003; Mathews, 2008; Sarmiento et al., 2009] in terms of minimizing the time and being more anticipative with respect to the future.

The decision strategies related contributions are described in Section 3.5.2 of Chapter 3, and they have been published in the following articles [Lanillos et al., 2012, 2013].

1.4.3 Algorithms

From the discrete optimization point of view we provide two algorithms to tackle the decision layer that, in combination with the strategies, improve previous approaches

[Eagle, 1984; Eagle and Yee, 1990; Bourgault et al., 2003, 2004; Yang et al., 2002, 2004] to solve the MTS problem. Within the continuous optimization and based on the works of [Mathews et al., 2007; Gan and Sukkarieh, 2010] we design an algorithm for long term MTS operations, and we refine the data fusion layer for systems with communication delays to obtain a more accurate target location estimation.

Therefore, we contribute with four solutions that tackle the MTS from different points of view:

- Express the MTS decision problem under the constraint programming with finite domains paradigm. Using this approach we obtain global optima solutions when the number of decision variables is small, i.e., we solve the discrete MTS in finite time.

  This solution is described in Section 4.3 of Chapter 4.

- An stochastic algorithm based on the Cross Entropy Optimization method (CEO) that tackle the MTS problem using the proposed strategies to search dynamic targets and supports any sensor model. By means of a tradeoff between optimality and computation time, this algorithm is able to compute the agents actions online.

  This solution is described in Section 4.4 of Chapter 4 and has been published in [Lanillos et al., 2012].

- A non-myopic gradient-based decentralized algorithm to solve the continuous MTS, which can be executed online. It optimizes the DTH strategy reducing the decisions locality and providing better long term decision without adding noticeable computational time.

  This solution is described in Section 5.3 of Chapter 5.

- An algorithm that solves, within the data fusion layer, the Out-Of-Sequence (OOS) problem when the target location is described by a Gaussian probability distribution. This contribution improves the target location estimation when there are communication delays as well as the searching performance of the agents.

  This algorithm is explained in Section 5.4 of Chapter 5, and has been published in [Besada-Portas et al., 2012].

Chapter 2

BACKGROUND

"Today's posterior distribution is tomorrow's prior" Lindley

2.1 Historical Background

To understand the subject tackled in this thesis, the minimum time search, it is interesting to make an historical review of the works related to this field. However, this overview focuses only on the researches that have motivated the thesis. Other surveys can be found in [Stone, 1989; Benkoski et al., 1991].

During the II World War the scientist and the engineers had already in mind the importance that the computers would have in many applications, but still the technology could not be as fast as the theory. Counted mechanical cases such the famous machine to decrypt "Enigma" went to the historical books as advances in computation. Names as Claude Shannon, Alan Turing, and Von Neumann were already involved in important research projects. This means that the period was the "primordial soup" for the computer science. The algorithmic theory started to intermingle with the classical mathematical models and applications appeared everywhere. One of the places for this evidence was in the navy, particularly at the submarine fleet. They were most of the time performing their tasks blindly and the idea of developing new mathematical methods to optimize the submarine trajectories was started. Anyway, we had to wait until the cold war for the nuclear submarines in the sixties, to see the first real application: finding the lost sea vessels of the USS Scorpion submarine [Stone, 1975]. A group of naval engineers and mathematicians had to work hard to look for a method that offered more guarantees of success in uncertain environments. Thanks to the intelligence or the expert information, they could develop a map of the most probable regions to search the objectives as vessels or other submarines. From that moment, the probabilistic theory and the search became inseparable friends. The probabilistic theory thanks to Bayes and their contemporary fellows, the algorithmic theory and a new discipline that had started during the war called operational research, which worked using mathematical methods to achieve better decisions, shaped what some later years will be called the optimal search. In the book "Theory of Optimal Search", written by Stone and published in 1975 [Stone, 1975], we can find the compendium of the mathematical models and algorithms used by the navy for searching lost targets.

Stone focused the problem on computing the optimal distribution of time that should be dedicated to each searching region. Few years later he would rename the problem as the density search problem to distinguish it from other setups in his overview of the optimal search published in 1989 [Stone, 1989]. Some contemporary authors realized that the search problem was just an instance of a more general one called Partially Observable Markov Decision Process (POMDP), as in the end, we observe partially an unknown state, the location of the target. The

search as a POMDP was first demonstrated for a small scenario [Dobbie, 1974] and later generalized in [Smallwood and Sondik, 1973]. At the same time, the naval engineer Eagle [Eagle, 1984] observed that if some restrictions in the searching path are introduced the methods provided by Stone did not work anymore. Thus, using POMDP resolution methods, that in practice are a modification of the original value iteration algorithm proposed by Bellman [Bellman, 1957] to work with belief states, he found a solution for the searching problem when the agent path is constrained. He faced then the inherent problem of solving a POMDP, the huge complexity due to the estate explosion, also called the curse of the dimensionality in [Powell, 2007]. For this reason and due to the low computation power of those days he only managed to solve small scenarios (i.e. a three by three grid). In 1991 the authors of the work in [Benkoski et al., 1991] affirmed:

"No general, effective algorithm for generating optimal searcher paths is known"

Simultaneously, some authors succeed to analyze the complexity of the searching problem when the path is constrained [Trummel and Weisinger, 1986], and demonstrated, by transforming it into a well-known problem that the complexity for only one agent is NP-hard or NP-complete depending on its instantiation.

A few years later, [Eagle and Yee, 1990; Washburn, 1998] designed branch and bound algorithms that solved some of the gaps in the Lagrange multipliers used by their predecessors and reduced the computational time by cutting the feasible estates. From all these studies, there was a fundamental idea, mentioned in [Eagle, 1984; Stone, 1989; Benkoski et al., 1991], that would be the key point for the new research lines. They set out that if the problem is to find a lost target, the task finishes when the target is discovered and therefore we can assume that the target is not detected during the whole decision plan. This little idea transforms the POMDP problem into a deterministic optimization, where there is just one type of observation: non-detection.

As a river that hides under the ground to reappear some kilometers farther, at the beginning of the XXI century and accompanied with other technological improvements, the miniaturization of sensing platforms and the swarms fever [Ross, 2011], a group of scientist working in the data fusion field, reactivated the research on optimal search. The researchers brought out the complexity of using the existent algorithms in real applications and particularly in multiagents frameworks. In fact Bernstein demonstrated in [Bernstein et al., 2002] that the multiagent search problem rises its complexity to NEXP. At this point, the researched lines were split into three. First, the generic POMDP algorithms followed their own way by offering new approximated algorithms [Kaelbling et al., 1998; Hsu et al., 2008]. Nevertheless they have a disadvantage that will be explained in the next section.

The second line tried to approach the problem in a discrete way with works related to the search [Yang et al., 2002, 2004] and external works related with more general problems [Blum et al., 2003]. The third line approached the problem from a continuous perspective and with a hard information theoretic base [Bourgault et al., 2004; Mathews et al., 2007; Lavis et al., 2008; Gan and Sukkarieh, 2010]. In this thesis we follow the works of the second and the third line that use the Bayesian approach.

But, how do they manage to solve such a complex problem in a reasonable time? The answer is straightforward: by using some approximations and assumptions. The most important approximation is to reduce the horizon or bound decision window. With uncertain information this is not a really bad idea, because it is possible that the information will lose value, but if we reduce the horizon we do not have the global view of the scenario, and our decisions are biased at the risk of getting trapped in a local optima. The majority of the works of the third line, [Bourgault et al., 2003, 2004; Mathews et al., 2007; Lavis et al., 2008; Gan and Sukkarieh, 2010], follow this approximation. One way to palliate the locality problem without losing tractability is to use heuristics as in [Yang et al., 2002]. From the multiagent point of view, another resource to make the problem tractable is to divide the searching system into a data fusion process and a controller that guides the agents. Assuming that there is information synchronization within the team of agents significantly simplifies the decision making controller [Bourgault et al., 2004; Mathews et al., 2007]. In this regard, data fusion techniques help a lot to tackle the optimal search problem.

From the search time minimization point of view, since Eagle's work [Eagle, 1984], the researches had abandoned the original objective of the optimal search: minimizing the time to find the target. The majority of the works, with exceptions such as [Sarmiento et al., 2009], used the other possible formulations defined in [Trummel and Weisinger, 1986], that is, maximizing the probability of detecting the target. Although, as we will see in the following chapter, this approach does not necessarily take into account the time. In fact, the authors of [Sarmiento et al., 2009] write:

"A more effective utility function balances the desire for a high probability of finding the object against the desire to minimize the search time"

The maximization of the probability of detecting the target strategy within the density problem, [Stone, 1975], has been demonstrated to minimize the time. However, when we constraint the agent path this is no longer true. To be fair, in some cases, when the necessary condition presented in Section 3.3 is satisfied, maximizing the probability is a feasible strategy to minimize the time.

Nowadays we can enumerate some baseline methods for the optimal search with constrained paths: an optimal not tractable algorithm [Eagle, 1984] with its approximation [Dell et al., 1996]; an approximation to the local expected time for a single agent [Sarmiento et al., 2009]; a discrete approach to the problem for multiple agents using heuristics that do not minimize the time directly [Yang et al., 2002]; and a cooperation multiagents algorithm [Mathews et al., 2007], which maximizes the probability of detecting an static target and provides local solutions.

This thesis holds on this historical background, where all the works mentioned have been included, complementing and ratifying this research. For this reason, this thesis does not only pretend to contribute with new time minimization strategies and methods for the optimal search, but it also formulates the multiagents Bayesian search theory under a unique notation and within a computational tractable methodology.

2.2 Formulations

Due to the interdisciplinary nature of the searching theory, we need to discuss three of the important existing formulations for the Minimum Time Search. The way to formalize mathematically and the notation used depends on the research field and affects implicitly the design of the solution. The first formulation is related to the operational research and is used nowadays by the artificial intelligence and machine learning community: Partial Observable Markov Decision Process (POMDP). The second one comes from the control field: Open Loop Controller (OLC). And finally, the third formulation appears from the data fusion community, where Bayesian inference is the core of the objective function. Although all formulations have many common concepts, sometimes is difficult to transform the problem from one notation to another.

This section just analyzes the different formulations and explains their main differences. Particularly this thesis will use a combination of OLC with the Bayesian formalization. We describe the main problem variables before analyzing the existing approaches.

τ - Target state location in a two dimensional space.

$\hat{\tau}$ - Estimated value of the target location.

b_τ - Belief that the team of agents have of the target location.

s - Agent state, defined by its location.

u - Action performed by the agent.

$v^k = \{u^k, \cdots, u^{k+N}\}$ - Set of planned actions for an horizon N.

$R(.)$ - Reward or cost of applying an action at a given state.

$V(.)$ and $J(.)$ - Value function and cost function. Although they are both related to the problem, we keep the notation of the different fields.

$P(.)$ - Probability of an event.

z - Observation (usually named o by the POMDP research community).

D - Target detection event. It has implicitly the observation $z = D$.

p - Decision tree of actions-observations.

$a(p)$ - The function that returns the root action of the decision tree.

$T(.)$ - State transition function.

γ - Discounted time parameter.

$E\{.\}$ - Expectation.

To develop the equations we assume deterministic actions and only two types of observations: detection D and non-detection $\overline{D}$. Besides, when a superscript appears like τ^k, it means the variable value at instant k, and when a subscript appears, like s_i^k, it refers to the value of the i-th agent.

2.2.1 POMDP

MTS can be formulated as a POMDP [Smallwood and Sondik, 1973], adding to the problem a definition of the reward $R([s^k, \tau^k], u^k)$ obtained by the agents given the extended state $[s^k, \tau^k]$ and action u^k. Under this approach, followed in [Kaelbling et al., 1998; Pineau et al., 2006], the expected reward $E\left[\sum_k \gamma^k R([s^k, \tau^k], u^k) \middle| b_\tau^0\right]$ is usually maximized. However, as the number of possible states of the agent grows, the problem becomes intractable for exact POMDPs algorithms, and it has to be tackled with approximated methods [Hsu et al., 2008; Amato et al., 2010].

The MTS objective function within the POMDP model is constructed by recursion following this equation [Kaelbling et al., 1998]:

$$V_p(s,\tau) = R(s,\tau,a(p)) + \gamma \sum_{\tau' \in T} P(\tau'|\tau) \sum_{z_i \in Z} P(z_i|s,a(p),\tau') V_{z_i(p)}(s',\tau') \qquad (2.1)$$

where $z_i(p)$ is the (k-1)-step policy subtree that has z_i at the top level of the k-step policy tree p. From this equation we can derive its belief MDP version that solves the problem.

The disadvantages of using this formulation can be summarized in three main ideas: the reward associated for each state and action should be defined as a constant value, scalability is poor for large scenarios, and it is not viable for online optimization [Amato et al., 2010].

An interesting formulation that falls between the POMDP and the information theoretic formalization described later appears in [Eagle, 1984]. Their authors exploit the binary detection/non-detection nature of the sensor model to tackle the problem for small scenarios using the same Dynamic Programming technique used to solve the exact belief MDP. By defining the value function as the remaining probability of detecting the target being in a specified state, and assuming that the agent just observes one state at each instant, the search can be solved by the following backward recursion:

$$V_k(s,b) = \underset{s^k \in S}{max} P(D^k|s^k)b(s^k) + (1 - P(D^k|s^k)b(s^k))V_{k-1}(s^k, T(b,s^k)) \tag{2.2}$$

where $P(D^k|s^k)b(s^k)$ is the probability of detecting the target in estate s^k and $T(b,s^k)$ is the update and predicted belief for the unsuccessful search in state s^k.

Although the method is not scalable, it improves the general POMDP formulation because the rewards are given by the target belief. This approach is a finite optimization (i.e. finite decision horizon) and its discounted infinite optimization version is still under research [Singh and Krishnamurthy, 2003].

2.2.2 Open Loop Controller

The optimal search has also been approached from the controller point of view. In fact, due to some properties later discussed in this thesis, it can be modelled as a open loop feedback controller [Mathews, 2008], where assuming that the observation is always non-detection the optimization becomes deterministic. Using an estimator to compute the estimated target location $\hat{\tau}^k$ the optimizer should solve the following objective function:

$$J = \sum_{j=1}^{\infty} R(u^k, \hat{\tau}^k) \tag{2.3}$$

where $R(u^k, \hat{\tau}^k)$ is the cost of performing the action u^k given the estimated state $\hat{\tau}^k$.

This infinity decision horizon optimization is still intractable, thus a Receding Horizon Controller (RHC) is needed. We will study extensively the MTS using a receding horizon controller in Section 3.5.

2.2.3 Information Theoretic

We believe that the best MTS formulation is the Bayesian approach, because it manages the problem uncertainty implicitly and exploits any prior information. In the probabilistic framework, the objective function is designed as an information filter. The thesis [Grocholsky, 2002] discusses this information theoretic approach widely. The Bayesian formulation provides an easy way to incorporate prior information about the target location, manages the multiagent information fusion and coupling, and permits the decision evaluation within a probabilistic problem.

Under this framework, information measures such us the system entropy [Grocholsky, 2002; Yang et al., 2002] and the joint probability of detection events [Bourgault et al., 2004] become important. The entropy measures the certitude that we have about an unknown variable and can be used as the utility function. This approach is good for learning an environment, but it lacks when we want to discover the target. This happens because searching a region to reduce the target location uncertainty does not imply that the target will be found by the agents. The joint probability of detection events, i.e., the probability of detecting the target during the decision plan provides more chances to succeed in the MTS. Following the Kalman notation we distinguish between the belief after the prediction from instant k to $k+1$ ($b_\tau^{k+1|k}$) and after the update of the new observations at $k+1$ ($b_\tau^{k+1|k+1}$).

The information theoretic approach for the optimal search is an algorithm that optimizes a probabilistic utility function or strategy. For instance, using the joint probability of detection events, given the agent trajectory and measurements, the strategy is to maximize the following function.

$$J = P(\bigcup_{j=1}^{N} D^{k+j} | s^{k+j}, z^{1:k+j-1}) \tag{2.4}$$

Both, the open loop controller and the information theoretic approach are the formalization that drive this thesis.

TABLE 2.1: Optimal search solutions

WORK	MTS	D/C	Constrained	Multiagent	Moving T.	Optimality(Horizon)	Optimization type
[Stone, 1975]	✓	D,C				Global	Lagrange
[Eagle, 1984]		D	✓		✓	Global(N)	DP (POMDP)
[Bourgault et al., 2003]		C	✓			Local(1)	Greedy
[Yang et al., 2002]		D	✓	✓	✓	Local(2)+Expectation	Heuristic+NN
[Bourgault et al., 2004]		C	✓	✓		Local(1)	Greedy
[Mathews et al., 2007]		C	✓	✓		Local(N)	Gradient
[Sarmiento et al., 2009]	✓	D,C	✓			Local Approx(N)	Limited DFS
[Gan and Sukkarieh, 2010]		C	✓	DEC		Local(N)	Explicit Gradient
Section 4.3	✓	D	✓	Central		Global(N)	CP
Section 4.4	✓	D	✓	Central	✓	Global Approx(N)+Heuristic	CEO
Section 5.3	✓	C	✓	DEC		Local(N)+Heuristic	Gradient

2.3 Closely Related Solutions

In the following, we show a comparison of the optimal search solutions more closely related to the MTS (including the algorithms presented in this thesis). This comparison does not pretend to be an exhaustive handbook of related works, but a compilation of the ones that have motivated this thesis. All the compared algorithms, except for [Eagle, 1984], are tractable, i.e., they compute a solution in an acceptable amount of time (polynomial depending on the horizon).

We compare these works using the seven characteristics summarized in Table 2.1:

MTS, the most important for us, indicates which algorithms solve directly the task of locating the target in the minimum possible time. [Eagle, 1984; Bourgault et al., 2003, 2004; Mathews et al., 2007; Gan and Sukkarieh, 2010] are based on detection maximization, although maximizing the detection does not mean that the time to locate the target is minimum. Indeed, [Bourgault et al., 2003] talks about the minimum expected time, but they decline to use the expression due to complexity issues. [Yang et al., 2002] is good in terms of reducing the uncertainty because they use the entropy, but that does not imply either that the target is located in minimum time. Only [Stone, 1975; Sarmiento et al., 2009] and the algorithms provided in this thesis solve the MTS directly. The solution provided in [Sarmiento et al., 2009], where a single agent searches for a static target using a local heuristic optimization, seems promising, although an analysis of the scalability to tackle the multiagent and dynamic target setup is still required. In our setup, the time to reach the next observation is the same for each action, and the local domination cut of the solutions that they propose is useless.

D/C, indicates if the search is modeled in continuous (C) or in discrete (D) form. Note that in continuous models, the control actions are piecewise linear. Besides, [Sarmiento et al., 2009] discretizes first the searching region into a visibility graph and then in a second phase it computes the continuous actions to fulfill the locations that should be visited.

Constrained, informs if the work solves the constrained path search problem, i.e., if it considers that the agent has spatial dynamic constraints.

Multi-agents, shows if the algorithm is applicable to several agents. [Yang et al., 2002; Bourgault et al., 2003, 2004; Mathews et al., 2007; Gan and Sukkarieh, 2010] are decentralize cooperative approaches, where the agents interchange the knowledge in a transparent way. Besides, in [Yang et al., 2002] the agents only interchange information when they are in adjacent cells. Although the methods proposed in this thesis are designed to be finally implemented in

decentralized form, the convergence study is not provided for all of them and therefore, only the continuous approach is considered as a decentralized method.

Moving Target, describes if the target can be moving during the decision stage. In [Stone, 1975; Bourgault et al., 2003, 2004; Mathews et al., 2007; Sarmiento et al., 2009; Gan and Sukkarieh, 2010] and in Section 4.3 and 5.3, the target is assumed to be static during the planning horizon. The remaining works assume target motion.

Optimality(Horizon), indicates if the solution is local or globally optimal and the horizon (how many steps ahead the algorithm optimizes). Most of the solutions are not global optima in the horizon window due to the non-convexity of the problem, so they get trapped in local optima (e.g. [Gan and Sukkarieh, 2010]). Besides, probabilistic approaches like the one in Section 4.4 are not globally optimal, because of their sampling mechanisms. Finally, the approaches in [Yang et al., 2002], Section 4.4 and Section 5.3 use the future expectation as an heuristic, to evaluate the solutions beyond the horizon.

Optimization Type, is more informative than comparative, and describes the algorithm used to solve the problem. [Mathews et al., 2007] and [Gan and Sukkarieh, 2010] are essentially the same work, but [Gan and Sukkarieh, 2010] uses an explicit gradient based algorithm, making the computation really fast. [Yang et al., 2002] uses an expectation as the heuristic and neural networks (NN) to learn the target location and [Eagle, 1984] implements a POMDP dynamic programming (DP) solution. The algorithms used in this thesis are presented in Sections 4.3, 4.4 and 5.3. The first method is based on Constraint Programming, the second one uses Cross Entropy Optimization (CEO) method and the third one is developed over a gradient-based method.

2.4 Probabilistic Theory in Uncertain Environments

The first important difficulty of the MTS is to manage the knowledge that the agents have about the target location, that is, the information that the system uses to choose between the actions. This subject has been largely studied in works such as [Furukawa et al., 2006].

In this section we describe the probabilistic information background used in this thesis. Because we have to handle subjective information with uncertainties, like the prior knowledge of the target distribution and the measurements provided by the sensors, the best way to tackle the uncertainty is to use Bayes inference [Jaynes, 2003]. In the following, we first go through a particular example (Section 2.4.1)

and we describe how to model the sensors (Section 2.4.2). Afterwards we describe the Bayesian inference for the optimal search (Section 2.4.3), where we provide: the general estimation method to solve the data fusion layer (Section 2.4.3.1) and the agents decision evaluation when the target is static (Section 2.4.3.2). Finally we study two particular cases: the Gaussian distribution (Section 2.5.1) and the out-of-sequence problem (Section 2.5.2).

2.4.1 Example: The Two Boxes Moving Animal

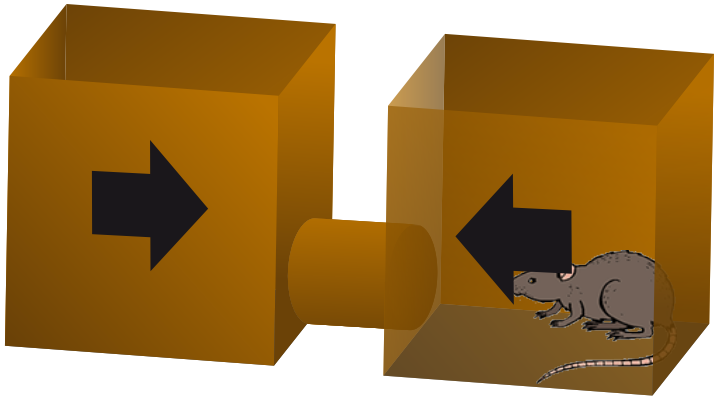


FIGURE 2.1: Two boxes animal example.

In this problem we have two connected boxes and an animal that moves between them as depicted in Figure 2.1. The objective of the game is to know where the animal is at each instant k. The animal starts in box 2, although we do not know that. We can observe one box at a time and if the animal is inside we have the certitude that we are going to see it. Let τ^k be the location of the animal and b_τ^k the belief that we have at each instant k. The initial information that we have is that the animal can be in any of the boxes equally (i.e. a uniform distribution). Thus the initial probability of detecting the animal in each box is,

$$b_\tau^0 = P(\tau^0) = \{0.5, 0.5\}^T$$

Note that the total probability is 1 as $\sum b_\tau^0 = 1$.

Denoting which box we select to observe as the sensor state s^k, the detection likelihood is,

$$P(z^k = D|\tau^k, s^k) = \begin{cases} 1 & \text{if } \tau^k = s^k \\ 0 & \text{otherwise} \end{cases}$$

This means that if the animal is in the box that we are observing the detection is positive.

TABLE 2.2: Transition matrix (A) for the two boxes

Box	1	2
1	0.8	0.6
2	0.2	0.4

First of all we assume that the animal is not moving so $\tau^k = \tau^{k-1} = \tau$. At instant 1 we observe box 1 and as the animal is in box 2 we get a measurement $z^1 = \overline{D}$. The belief has to be actualized using the Bayesian update operation:

$$P(\tau = 1|z^1 = \overline{D}, s^1 = 1) = \\ = \frac{P(z^1 = \overline{D}|\tau = 1, s^1 = 1)P(\tau = 1)}{P(z^1 = \overline{D}|\tau = 1, s^1 = 1) \cdot P(\tau = 1) + P(z^1 = \overline{D}|\tau = 2, s^1 = 1)P(\tau = 2)} \tag{2.5}$$

Note that the first term on the numerator is the detection likelihood, which in this case is zero because if the animal is in box 1 it is impossible a non-detection observation. The second term of the numerator is the prior probability b^0_τ. If we solve the equation we get the following:

$$P(\tau = 1|z^1 = \overline{D}, s^1 = 1) = \frac{0 \cdot 0.5}{0.5 \cdot 0 + 0.5 \cdot 1} = 0$$

We use the same update rule to obtain the probability at the second box ($\tau = 2$),

$$P(\tau = 2|z^1 = \overline{D}, s^1 = 1) = \frac{0.5}{0.5} = 1$$

Thus, the general Bayes rule for step 1 is as follows:

$$b^1_\tau = P(\tau|z^1, s^1) = \frac{P(z^1|\tau, s^1) \cdot P(\tau)}{\sum_\tau P(z^1|\tau, s^1) \cdot P(\tau)}$$

Using a normalizer η to make $\sum b^1_\tau = 1$, we get the recursive Bayesian update expression:

$$b^1_\tau = \frac{1}{\eta} P(z^1|\tau, s^1) \cdot P(\tau) = \frac{1}{\eta} P(z^1|\tau, s^1) b^0_\tau$$

Now we focus on the probabilistic motion of the animal $P(\tau^k|\tau^{k-1})$, defined in *discrete worlds* as a transition matrix A_{ij}, where each element is the probability of the target moving from state j to i [Bertuccelli and How, 2006]. As we have two states (two boxes), A_{ij} is a 2×2 matrix built from the data extracted from Table 2.2.

The objective of the problem is to estimate the position of the animal at instant k. Lets assume that there are no observations, so the initial belief is only modified by the motion model of the animal. Thus the probability of finding the animal in box 1 is the probability of the animal being in box 1 multiplied for the probability of remaining there, plus the probability of being at box 2 multiplied for the probability of moving

from box 2 to box 1.

$$P(\tau^k = 1) = P(\tau^k = 1|\tau^{k-1} = 1)P(\tau^{k-1} = 1) + P(\tau^k = 1|\tau^{k-1} = 2)P(\tau^{k-1} = 2)$$

Using the same reasoning we can also calculate the probability of the animal being in box 2. Therefore, computing the probabilities for the first step, where we have a uniform location distribution of the animal, we get:

$$P(\tau^1 = 1) = 0.8 * 0.5 + 0.6 * 0.5 = 0.7$$

$$P(\tau^1 = 2) = 0.4 * 0.5 + 0.2 * 0.5 = 0.3$$

Thus the new belief is,

$$b_\tau^1 = \{0.7, 0.3\}^T$$

Generalizing, the prediction of τ^k (the belief b_τ^k) is the sum of the products of the probability motion model (distribution) and the previous belief b_τ^{k-1}, over all possible τ^{k-1},

$$b_\tau^k = \sum_{\tau^{k-1}} P(\tau^k|\tau^{k-1})b_\tau^{k-1}$$

Assuming that the probabilistic motion model does not change along time we can compute numerically the location belief at any instant using the transition matrix A,

$$b_\tau^k = A^k \cdot b_\tau^0$$

The infinite ahead prediction ($k \approx \infty$) or the steady state of τ, occurs when the system belief does not change: $b_\tau^k = Ab_\tau^{k-1}$. This is computed numerically by picking a high k value,

$$\lim_{k\to\infty} b_\tau^k = A^\infty b_\tau^0$$

The steady state computed for the example is,

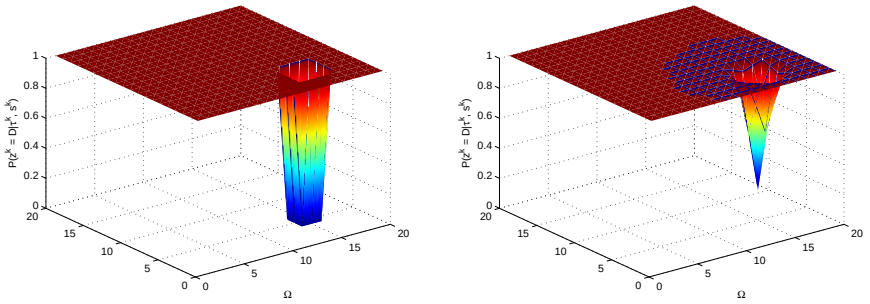
$$b_\tau^\infty = [0.75, 0.25]^T$$

That implies that in a long term the best decision is to observe box 1 first.

Finally, to solve the problem, we use the prediction and the update equation iteratively. The general equations are detailed in Section 2.4.3.

2.4.2 The Sensor

The sensor plays an important role in the MTS problem. It is the interface between the agent and the world. It is the agent eyes and, without it, the search is blind and



(a) Ideal Binary (b) Exponential

FIGURE 2.2: Sensor modeled as the observation likelihood with a range threshold of 3 in a searching region Ω of size 20×20.

depends too much on the prior information, which can be wrong or incomplete. Thanks to the sensor, the agent updates the knowledge about the target location. That is, the sensing agent makes observations z^k that are detection/non-detection measurements. The sensor model, which is generally a function of the sensor state s^k (i.e. agent) and the target state τ^k, can be characterized with the observation likelihood:

$$P(z^k = D|\tau^k, s^k) \tag{2.6}$$

This observation likelihood is the probability of detecting the target when the target position is τ^k and the sensor state is s^k. Assuming that the sensor and the agent have the same position, the sensor state is the agent location.

Using this approach we mainly contemplate two sensor models:

1. Ideal binary sensor model (Figure 2.2(a)) that returns 1 when the sensor "sees" the target.

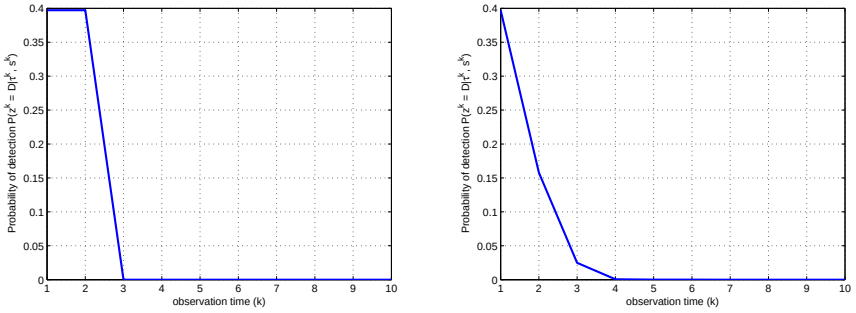
$$P(z^k = D|\tau^k, s^k) = \begin{cases} 1 & \text{if } \tau^k = s^k \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

2. Distance exponential model (Figure 2.2(b)). It is a continuous model that has been used largely in the literature of the optimal search [Stone, 1975; Gan and Sukkarieh, 2010].

$$P(z^k = D|\tau^k, s^k) = P_{d_{max}} e^{-\sigma\left(\frac{\|\tau^k - p^k\|}{d_{max}}\right)^2} \tag{2.8}$$

where $\|\tau^k - p^k\|$ is the Euclidean distance between the sensor and the target possible positions. It describes a sensor that is less certain in places far away from the sensor position.

Figure 2.3 shows an example of the temporal evolution of the target belief when the agent observe a location with initial probability of 0.4 and no target is present (i.e. the agent gets non-detection measurements). The blue curve describes the probability of detecting the target in that location as the time passes. Figure 2.3(a) describes this evolution when an ideal binary sensor observes the location at instant 3. In this moment the probability goes to zero. Figure 2.3(b) describes the behavior with the exponential sensor that reduces the uncertainty as the time passes. At instant 4 it is almost certain that the target is not at the observed location.



(a) Ideal Binary (b) Exponential

FIGURE 2.3: Sensor model certainty measure behavior when observing that there is not an object at the same location during k instants.

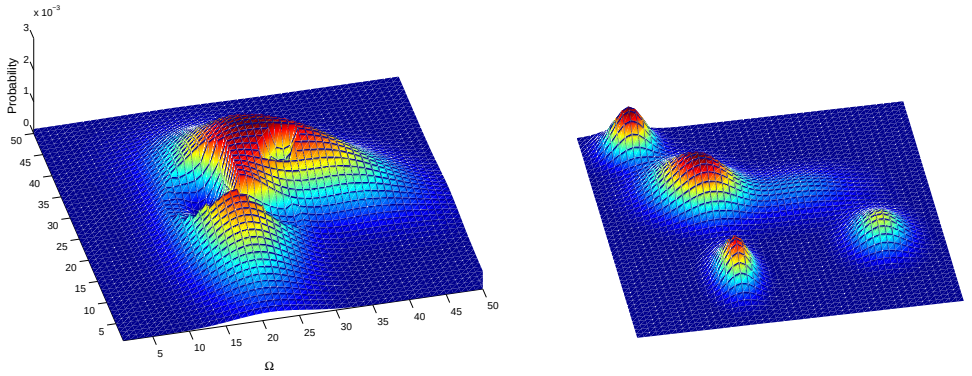
Finally, in Section 5.4 we employ a more sophisticated sensor model that instead of using the detection returns the relative distance information to the target. If the euclidean distance between the agent state and the estimated target location is $\|s_i^k - \tau^k\|$, the distance sensor measurement z_i^k is given by the following equation:

$$z_i^k = \begin{cases} \|s_i^k - \tau^k\| + \nu^k & \text{if } \|s_i^k - \tau^k\| \leq \xi \\ \text{no sensor measurement} & otherwise \end{cases} \tag{2.9}$$

The measurement error ν^k is modeled as a Gaussian distribution with zero mean and variance R, and the sensor only provides measurements then the target is close enough (i.e. when the distance is less than the threshold ξ).

2.4.3 Bayesian Inference

The Bayesian approach permits us to start from a prior target belief b_τ^0 and update the information after every successive sensor observation [Jaynes, 2003]. Also we can predict the target position according to the probabilistic motion model. This reasoning is the core of the information-theoretic searching algorithms because it manages the target location information at the data fusion layer and during the decision making process. Figure 2.4 shows two examples of the target location belief b_τ^k: a generic probability distribution and a multi Gaussians scenario. The height map represents the probability



(a) Generic Distribution (b) Multi Gaussian Distribution

FIGURE 2.4: Target probability of being in a region. The target location belief b_τ^k.

of the target to be in any location in the 2D space, where higher values imply bigger chances of finding the target.

2.4.3.1 Recursive Bayesian Estimation (RBE)

The Recursive Bayesian Estimation (RBE) [Furukawa et al., 2006] is an algorithm that permits us to estimate and predict the target location using the sensor measurement and the target transition model. It iterates over two steps: update and prediction. Using the Kalman filter notation we distinguish two beliefs:

1. $b_\tau^{k|k} = P(\tau^k|z^{1:k}, s^{1:k})$ is the probability location distribution of the target at instant k given all the observations and agent positions from the beginning $(z^{1:k}, s^{1:k})$. That is, the belief after the update step.
2. $b_\tau^{k|k-1} = P(\tau^k|z^{1:k-1}, s^{1:k-1})$ is the probability location distribution of the target at instant k given the previous observations and agent locations $(z^{1:k-1}, s^{1:k-1})$. That is, the belief after the prediction step.

We refer to the belief b_τ^k as the probability distribution after the prediction and the update step. Thus, $b_\tau^k = b_\tau^{k|k}$. The observation instant superscript can also be omitted when there are no observations because $b_\tau^{k|k} = b_\tau^{k|k-1}$.

Prediction Step

This step predicts the location of the target at instant k given the probability target

distribution of the target at $k-1$, according to the target probabilistic motion model.

$$b_\tau^{k|k-1} = P(\tau^k|z^{1:k-1}, s^{1:k-1}) = \tag{2.10}$$

$$= \int_{\tau^{k-1}} P(\tau^k, \tau^{k-1}|z^{1:k-1}, s^{1:k-1}) d\tau^{k-1} \tag{2.11}$$

$$= \int_{\tau^{k-1}} P(\tau^k|\tau^{k-1}) P(\tau^{k-1}|z^{1:k-1}, s^{1:k-1}) d\tau^{k-1} \tag{2.12}$$

$$= \int_{\tau^{k-1}} P(\tau^k|\tau^{k-1}) b_\tau^{k-1|k-1} d\tau^{k-1} \tag{2.13}$$

If the target is static, it remains in a fixed location along time. Thus, the target belief prediction step is not needed because:

$$b_\tau^{k|k-1} = P\left(\tau^k|z^{1:k-1}, s^{1:k-1}\right) = P\left(\tau^{k-1}|z^{1:k-1}, s^{1:k-1}\right) \tag{2.14}$$

$$= b_\tau^{k-1|k-1} \tag{2.15}$$

Therefore, we can write the state location of a static target without the superscript k: $\tau^k = \tau$.

Update Step

The following shows the general form of recursive Bayesian target belief estimation for a series of sensor observation events. We start with a prior target belief up to time step k, $P\left(\tau^k|z^{1:k-1}, s^{1:k-1}\right)$, conditioned on all previous sensor observations $z^{1:k-1}$ taken at sensor states $s^{1:k-1}$. The posterior target belief $b_\tau^{k|k}$ can be expressed using Bayes rule:

$$b_\tau^{k|k} = P\left(\tau^k|z^{1:k}, s^{1:k}\right) = \frac{P\left(z^k|\tau^k, z^{1:k-1}, s^{1:k}\right) P\left(\tau^k|z^{1:k-1}, s^{1:k}\right)}{P\left(z^k|z^{1:k-1}, s^{1:k}\right)} =$$

$$= \frac{P\left(z^k|\tau^k, z^{1:k-1}, s^{1:k}\right) P\left(\tau^k|z^{1:k-1}, s^{1:k-1}\right)}{P\left(z^k|z^{1:k-1}, s^{1:k}\right)} \tag{2.16}$$

Note that $P\left(z^k|\tau^k, z^{1:k-1}, s^{1:k}\right)$ is the sensor model or observation likelihood at instant k. $P\left(\tau^k|z^{1:k-1}, s^{1:k-1}\right)$ is the prior target belief $b_\tau^{k|k-1}$. Finally $P\left(z^k|z^{1:k-1}, s^{1:k}\right)$ depends on known information and therefore can be defined as the constant η that makes the sum of all probabilities equals to 1.

Thus, the update step is rewritten as:

$$b_\tau^{k|k} = P\left(\tau^k|z^{1:k}, s^{1:k}\right) = \frac{1}{\eta} P\left(z^k|\tau^k, s^k\right) b_\tau^{k|k-1}$$

On one hand, using the premise that the observation is always non-detection [Stone, 1989], $z^k = \overline{D}$, until the target is detected, we can, without loss of generality, ground

the RBE update as:

$$b_{\tau}^{k|k} = \frac{1}{\eta} P\left(\overline{D}^{k} | \tau^{k}, s^{k}\right) \hat{b}_{\tau}^{k|k-1} \tag{2.17}$$

In order to obtain the RBE algorithm for the searching task (Algorithm 1), we combine the prediction and update steps.

Algorithm 1 RBE Algorithm

Require: $b_{\tau}^{k|k}$

1: **for** $j = 1, N$ **do**
2: $\quad b_{\tau}^{k+j|k+j-1} \leftarrow \int_{\tau^{k+j-1}} P(\tau^{k+j}|\tau^{k+j-1}) b_{\tau}^{k+j-1|k+j-1} d\tau^{k+j-1}$
3: $\quad b_{\tau}^{k+j|k+j} \leftarrow \frac{1}{\eta} P(\overline{D}^{k+j}|\tau^{k+j}, s^{k+j}) b_{\tau}^{k+j|k+j-1}$
4: **end for**

2.4.3.2 Joint Probability of Non-Detection for the Static Target Scenario

Here we describe how to measure the goodness of a sequence of agent observations in a probabilistic framework. We follow the approach presented by [Bourgault et al., 2004; Mathews, 2008] for a static target. We want to derive the joint probability of non-detection events across a sensor action horizon of finite length. In here, s^k is the sensor state at time step k; v^k is an action vector that transits the sensor state from s^k to $\{s^{k+1}, \ldots, s^{k+N}\}$, where N is the number of step ahead actions; and $b_{\tau}^{k} = P(\tau | z^{1:k}, s^{1:k})$ is the posterior target belief at time step k.

For a piece-wise optimization decomposition of horizon N, the control actions are continuous but they change their value each discrete instant k, and there are a set of N observations ($z^{k+1:k+N}$). These actions are defined by the vector $v^k = \{u^k, \ldots, u^{k+N-1}\}$. We assume that the sensor state (location) transition model is deterministic, i.e. by applying v^k from any state s^k, the future sensor states $\{s^{k+1},\ldots,s^{k+N}\}$ can be explicitly obtained. We further assume that the target is static over the action horizon and that the sensor observations on these future states are independent. Therefore, we can write the joint conditional probability of not detecting the target over the action horizon as:

$$P(\bigcap_{j=1:N} z^{k+j} = \overline{D} | \tau, s^{k+1:k+N}) = P\left(z^{k+1:k+N} = \overline{D} | \tau, s^{k+1:k+N}\right) = \tag{2.18}$$

$$= \prod_{j=1}^{N} P\left(z^{k+j} = \overline{D} | \tau, s^{k+j}\right) \tag{2.19}$$

By marginalizing over the target location τ and multiplying with the target belief b_{τ}^{k}, we arrive at the utility function J_{nd} which is a scalar value describing the joint probability of non-detection event [Gan and Sukkarieh, 2010]:

TABLE 2.3: Initial target location probability mass function

0.1	0.25
0.2	0.45

$$
\begin{aligned}
J_{nd}(s^k, v^k, b_\tau) &= P(z^{k+1:k+N} = \overline{D} | z^{1:k}, s^{1:k+N}) \\
&= \int_\tau P\left(z^{k+1:k+N} = \overline{D}, \tau | z^{1:k}, s^{1:k+N}\right) d\tau \\
&= \int_\tau P\left(z^{k+1:k+N} = \overline{D} | \tau, s^{1:k+N}\right) P\left(\tau | z^{1:k}, s^{1:k+N}\right) d\tau \\
&= \int_\tau P\left(z^{k+1:k+N} = \overline{D} | \tau, s^{k+1:k+N}\right) P\left(\tau | z^{1:k}, s^{1:k}\right) d\tau \\
&= \int_\tau \prod_{j=1}^{N} P\left(z^{k+j} = \overline{D} | \tau, s^{k+j}\right) b_\tau^k d\tau
\end{aligned} \tag{2.20}
$$

In the derivation of the formula we also assume that the target location does not depend on the sensor future states: $P(\tau | z^{1:k}, s^{1:k+N}) = P(\tau | z^{1:k}, s^{1:k}) = b_\tau^k$

Lets show the effects of this formula through an example. The world Ω, described in Table 2.3, consists in 4 connected cells (states). The initial belief represented as a column vector is $b_\tau^k = [0.1, 0.25, 0.2, 0.45]^T$. Because Ω is discrete we substitute the integrals by the sum over all target possible states.

We fix the agent path length to $N = 2$ with the following sequential locations:

$$s^{k+1:k+N} = \{2, 4\}$$

We compute the non-detection joint probability (J_{nd}):

$$J_{nd} = \sum_\tau P(\overline{D}^{k+1} | \tau, s^{k+1} = 2) P(\overline{D}^{k+2} | \tau, s^{k+2} = 4) b_\tau^k$$

The detection likelihood for the ideal binary sensor is a column vector with zero at the agent location and ones in the rest locations. Therefore, the detection likelihood for the agent path is:

$$P(\overline{D}^{k+1} | \tau, s^{k+1} = 2) = [1, 0, 1, 1]^T$$

$$P(\overline{D}^{k+2} | \tau, s^{k+2} = 4) = [1, 1, 1, 0]^T$$

Substituting into J_{nd} we finally obtain the joint non-detection probability for the agent trajectory:

$$J_{nd} = \sum_{\tau} [1, 0, 1, 0]^T \cdot b_{\tau}^k = 0.3$$

Note that $\cdot$ is the element-wise product that multiplies each element of the vector.

2.4.3.3 Multi-Agent Extension

In the case that we have a team of q sensing agents we have a decision vector $v_{1:q}^k = \{v_1^k, \cdots, v_q^k\}$ and therefore the team states are $s_{1:q}^k$. The RBE just changes on the update step where we incorporate the information provided by all sensors at instant k ($z_{1:q}^k = \overline{D}$). Therefore Eq. 2.17 becomes:

$$b_{\tau}^{k|k} = \frac{1}{\eta} \prod_{i=1}^{q} P\left(\overline{D}_i^k | \tau^k, s_i^k\right) \hat{b}_{\tau}^{k|k-1} \tag{2.21}$$

Generalizing for a team of q agents the joint probability of non-detection events for static targets becomes:

$$J_{nd}(s_Q^k, v_Q^k, b_{\tau}^k) = \int_{\tau} \prod_{j=1}^{N} \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | \tau, s^{k+j}) b_{\tau}^k d\tau \tag{2.22}$$

2.4.3.4 Decentralized Extension

We can decentralize the utility function (Eq. 2.22) by cooperation [Mathews et al., 2007], communicating the partial products to the rest of the team. Therefore, we define the joint team belief that the agent i receive as ${}^i b_{\tau}^k$. Assuming an implicit synchronization of the team information or, in other words, a Decentralized Data Fusion (DDF) layer [Gan and Sukkarieh, 2010], that assures ${}^i b_{\tau}^k =^j b_{\tau}^k = b_{\tau}^k | \forall \{i, j\} \in (1:q)$, we can compute the joint non-detection for each individual agent as follows:

$${}^i b_{\tau}^k = \prod_{j=1, j \neq i}^{q} P(\overline{D}_j^{k+1:k+N} | \tau, s_j^{k+1:k+N}) b_{\tau}^k d\tau \tag{2.23}$$

Thus, the final joint probability of non-detection events in decentralized form is as follows [Mathews, 2008; Gan and Sukkarieh, 2010]:

$$J_{nd}(s_i^k, v_i^k, b_{\tau}^k) = \int_{\tau} \prod_{j=1}^{N} P(\overline{D}_i^{k+j} | \tau, s^{k+j}) {}^i b_{\tau}^k d\tau \tag{2.24}$$

Although we have the function in decentralized form, the computation of the best team actions, that it is our problem, needs first of all an optimization algorithm, and secondly a convergence proof. We also remind that while the RBE works for dynamic targets, the computation of the joint non-detection probability presented in this section is formulated for an static target.

2.5 Particular Cases

In this section we discuss two important particular cases that have importance within the data fusion problems. The first one is related to a particular representation of the target location belief, commonly used in applications like object tracking: the Gaussian distribution. Thanks to Kalman [Mutambara, 1998], we have the closed form of the optimal RBE to estimate the uncertain location state. Therefore this especial case needs a more deep analysis.

The second particular case is a problem that occurs in many multisensorial problems and is related to the order and the time of the observations arriving to the fusion center. This problem is known as the Out-Of-Sequence (OOS) [Bar-Shalom and Chen, 2005], and should be managed in multisensorial schemes, as this multiagent system, in order to have a good information synchronization.

2.5.1 Gaussian Distribution

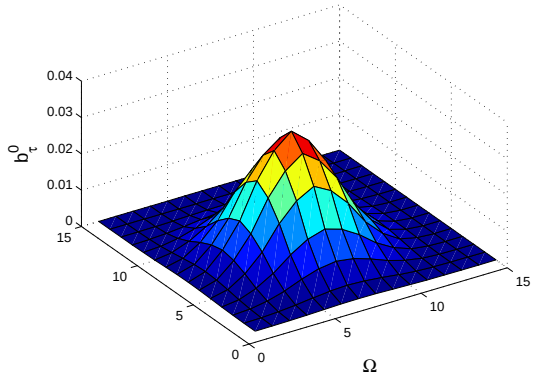


FIGURE 2.5: Target location represented as a Gaussian distribution (b^k_τ).

The Gaussian distribution is described by the estimated mean $\hat{\tau}^k$ and the covariance matrix Σ^k that represents the uncertainty of the target location. Figure 2.5 shows an example of a 2D Gaussian distribution with mean $\hat{\tau}^k = [7, 7]$ and variance $\Sigma^k = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$. The probability of the target being in any location of the space is given by the following probability density function:

$$b_{\tau}^{k} = \frac{1}{(2\pi)\sqrt{|\Sigma^k|}} e^{-\frac{1}{2}([x,y]-\hat{\tau}^k)(\Sigma^k)^{-1}([x,y]-\hat{\tau}^k)^T} \tag{2.25}$$

When we have a Gaussian model we can use the Kalman filter instead of the general RBE to predict and update the target location state. The Kalman Filter (KF) [Mutambara, 1998] provides an optimal way to update[1] and predict the target location estimation when there is additive Gaussian noise associated with the sensor measurements and the target dynamics. For some non-linear problems, the Extended Kalman Filter (EKF) [Mutambara, 1998] is a valid approximation. The KF is usually designed in the states space, but we can also represent it in the information space. Table 2.4 shows the variables used and the conversion equations for both domains. y^k is the information about the target location at k and Y^k the information matrix.

TABLE 2.4: State and information domains

NAME	STATE	INFORMATION	CONVERSION
Target Location	$\hat{\tau}^k$	y^k	$y^k = Y^k \tau^k$
Uncertainty	Σ^k	Y^k	$Y^k = (\Sigma^k)^{-1}$

By using the information space, the assimilation of the new observations of multiple sensors gets simplified by the addition properties of the information. When many measurements of the sensors arrive we only have to compute the information i_j^k and the information matrix I_j^k associated with each measurement j and sum them up to compute the total estimated sensor information vector and matrix. Given a set of q measurements provided by the agents, the total sensor information is calculated by the following equations:

$$y^k = \sum_{j=1}^{q} i_j^k \tag{2.26}$$

$$Y^k = \sum_{j=1}^{q} I_j^k \tag{2.27}$$

For the non-linear case we use the Extended Information Filter (EIF) [Mutambara, 1998] to manage the team information. The algorithm is an iterative process of prediction and updates/assimilations. Here we present an hybrid EKF and EIF where the prediction step uses the state space and the assimilation step exploits the information space properties.

[1] In Kalman literature the term assimilation is equivalent to the RBE update.

The prediction step (Algorithm 2) estimates the mean and the covariance matrix of the target location distribution for the next instant $k+1$, that in Kalman notation is defined as: $\hat{\tau}^{k+1|k}$, $\Sigma^{k+1|k}$. It uses the Jacobian F of the target motion model $f(\hat{\tau}^{k|k})$, and the target movement noise covariance matrix Q^k.

Algorithm 2 EKF prediction

Require: $\hat{\tau}^{k|k}$ ▷ Estimated and updated target location
Require: $\Sigma^{k|k}$ ▷ Target location error
Require: Q^k ▷ Target dynamics noise covariance
1: $\hat{\tau}^{k+1|k} \leftarrow f(\hat{\tau}^{k|k})$
2: $F \leftarrow \frac{\partial f(\hat{\tau}^{k|k})}{\partial(\hat{\tau}^{k|k})}$
3: $\Sigma^{k+1|k} \leftarrow F\Sigma^{k|k}F^T + Q^k$

The EIF assimilation (Algorithm 3) computes the updated Gaussian distribution of the target location, given the last belief parameters ($\hat{\tau}^{k+1|k}$, $\Sigma^{k+1|k}$) and the new team observations $z^k_{1:q}$. In this case z^k_i are target distances. First of all we transform the variables from the state space into the information space obtaining the information $y^{k+1|k}$ and its matrix $Y^{k+1|k}$. Then we correct the measurements getting zc^k_j and compute the information associated with each corrected measurement: i^k_j and I^k_j. Finally we calculate the new assimilated information $y^{k+1|k+1}$ by summing up all the measurements information and we transform again the variables into the states space. Note that each sensing agent provides a new observation at instant k and that we use the Jacobian H_j of the sensor model $g(s^k_j, \tau^k)$ for each agent to compute the information i^k_j contributed by the new observation.

Algorithm 3 EKF Assimilation

Require: $\hat{\tau}^{k+1|k}$ ▷ Controller predicted target location
Require: $\Sigma^{k+1|k}$ ▷ Controller predicted target location error
Require: $z^k_{1:q}$ ▷ Sensor measurements
Require: $R^k_{1:q}$ ▷ Measurement noise covariance
1: $Y^{k+1|k} \leftarrow \left(\Sigma^{k+1|k}\right)^{-1}$
2: $y^{k+1|k} \leftarrow Y^{k+1|k}\hat{\tau}^{k+1|k}$
3: **for** $j = 1$ to q **do**
4: $\quad H_j \leftarrow \frac{\partial g(s^k_j, \tau^k)}{\partial s}$
5: $\quad zc^k_j \leftarrow z^k_j - \|s^k_j - \hat{\tau}^k\| + H_j\hat{\tau}^k$
6: $\quad i^k_j \leftarrow H^T_j R^{-1}_j zc^k_j$
7: $\quad I^k_j \leftarrow H^T_j R^{-1}_j H_j$
8: **end for**
9: $y^{k+1|k+1} \leftarrow y^{k+1|k} + \sum_{j=1}^{q} i^k_j$
10: $Y^{k+1|k+1} \leftarrow Y^{k+1|k} + \sum_{j=1}^{q} I^k_j$
11: $\Sigma^{k+1|k+1} \leftarrow (Y^{k+1|k+1})^{-1}$
12: $\hat{\tau}^{k+1|k+1} \leftarrow \Sigma^{k+1|k+1}y^{k+1|k+1}$

2.5.2 Out Of Sequence Problem

Since the development of the KF [Mutambara, 1998] new difficulties have appeared associated with newer and more complex systems and scenarios. One of those challenging scenarios is related with the delay of the sensor measurement arrival to the fusion center. When the sensor information arrives delayed or disorder in time to the data fusion center we are talking about the Out-Of-Sequence Problem (OOSP) [Bar-Shalom and Chen, 2005]. To deal with the general OOSP we can find in the literature several approaches: discarding the delayed measurements [Smith and Seiler, 2003], postponing the estimation until the data is available [Lopez-Orozco et al., 2000], restarting the whole estimation from the oldest measurement time stamp [Kosaka et al., 1993], and doing a forward update approximation using just the new measurements [Nettleton and Durrant-Whyte, 2001; Zhang et al., 2003; Besada-Portas et al., 2009, 2011].

The agents that are sensing the region obtain information about the target that is communicated to the team. This process is always subject to delays and the information filtering can be modeled as an OOS data fusion [Bar-Shalom and Chen, 2005]. Figure 2.6 shows the OOS problem for multiple sensors. The network delays make the agents receive the measurements in a late instant $k+\delta$. In order to plan correctly the trajectory followed by the team of agents, we need the best updated target information without losing the algorithm speed for an online execution. Therefore, instead of adopting a discarding or full propagation strategy we use the one proposed in [Besada-Portas et al., 2011] where the new measurements are included properly (i.e. as if they have arrived on time), without a significant computation overload.

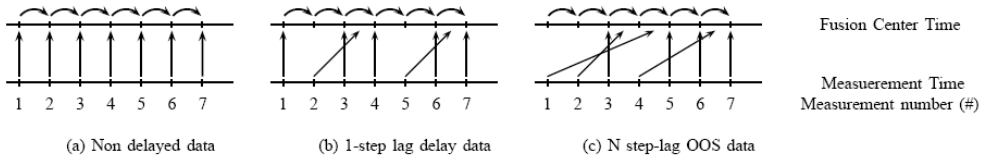


FIGURE 2.6: Out-Of-Sequence problem.

2.6 Search System Design

As we have seen, there are two processes involved in the searching: the management of the information or the data fusion for the team, and the decision making algorithm or controller. This two related layers have to be combined within the agent to design the searching system. This can be carried out, as proposed in [Mathews, 2008], with a top layer in charge of updating the information with the real measurements obtained by the sensors and a bottom layer that computes the agents actions using a prediction observation model to infer the target location. In practice, the data fusion layer is, for instance, driven by the standard RBE algorithm [Furukawa et al., 2006] (Section

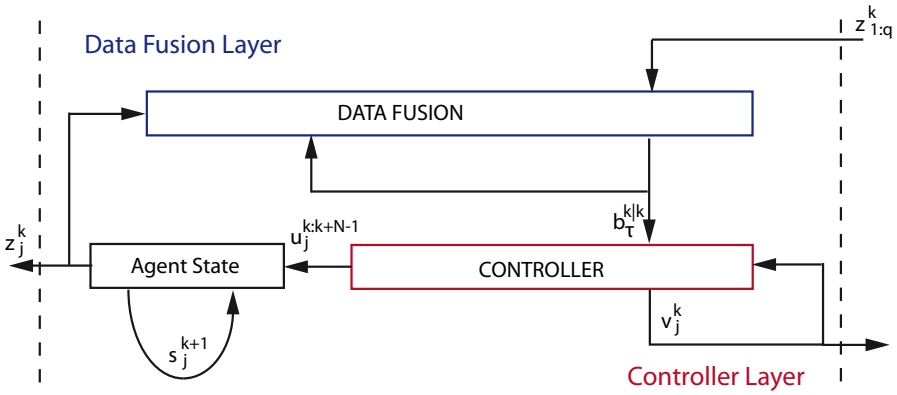


FIGURE 2.7: General search system design.

2.4.3.1) and the controller layer is administrated by an optimization algorithm. Figure 2.7 shows the abstract design of the searching system for an agent. The data fusion layer receives the new team observations, and updates and predicts the target location iteratively. This updated information feeds the controller layer that computes the best actions cooperating with the other agents. The output of the controller are the actions that the agent should apply. Executing the control actions each agent moves around the searching region obtaining new observations that feed the data fusion layer again, closing the system loop.

2.7 Summary

This chapter describes the thesis backdrop and explains some of the basic concepts used later. The historical review presents the MTS as a problem with great significance and rises up the importance of developing new online approaches that are able to minimize the time to find the target. We have presented several formulations that describe the MTS as a problem that falls in many research fields, and therefore, it can be approached from different points of view. For this thesis, and in order to exploit the MTS inner characteristics, we have chosen to design the strategies within the Bayesian theory and to approach the decision making process as a receding horizon controller. The related work comparison shows that there are still unsolved problems that this thesis tackles like multiagent searching for dynamic targets.

A complete study of the Bayesian probabilistic framework applied to the optimal search is also introduced. Facts such as the sensor model, the information inference and the probability of detection are explained. Besides, interesting particular problems as the out-of-sequence are analyzed.

Finally, the general system design used along this thesis is explained. This design details a data fusion layer that manages the information and a controller layer that optimizes the agent actions. With this scheme, dividing the MTS into two subsystems, proposing

the correct methods to fill both black boxes and defining the right connections, we will successfully tackle the problem.

Chapter 3

MINIMUM TIME SEARCH

"We must use time as a tool, not as a crutch." JFK

3.1 What is MTS

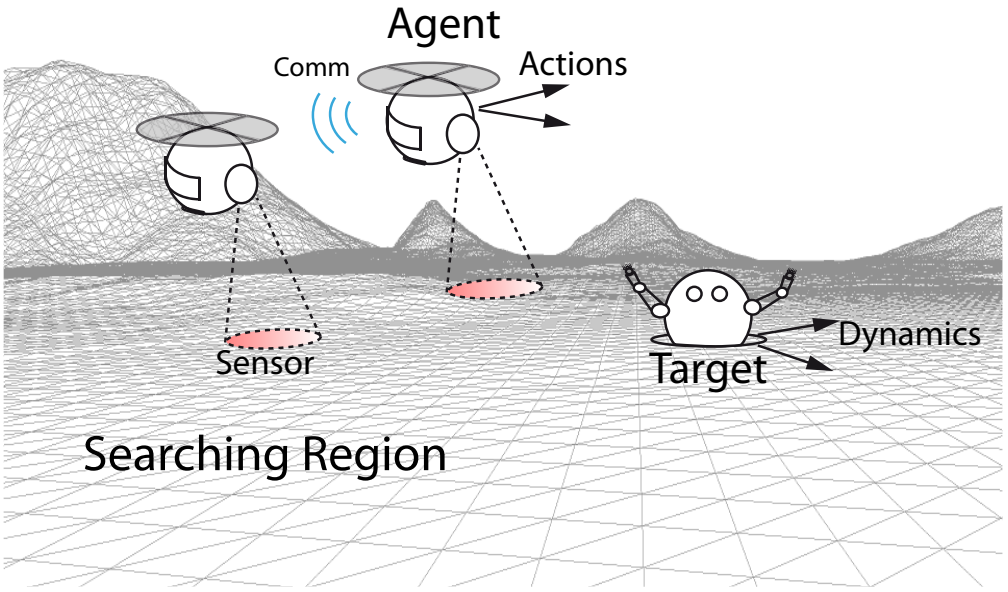


FIGURE 3.1: MTS general problem.

The Minimum Time Search (MTS) problem consists in finding an object, that is placed somewhere in the "space", in the minimum possible time. Underneath, it is about making the optimal decisions using the information that we have, taking into account the time. The object and the space concepts differ from one application to another (as Section 3.1.4 shows). Figure 3.1 displays a MTS illustrative example of two mobile agents looking for a moving target inside a delimited searching region. The team of agents has to optimize its actions to find the target as soon as possible.

The MTS problem is faced as a probabilistic information gathering task that involves two dynamic partakers: the mobile agents which have sensing capabilities, and the object that is searched. The object or target is contained within a closed and finite region, its location is uncertain and it is moving, but its dynamics do not depend on the searchers. The agents are able to move over the searching region, make observations, detect the target and communicate. The agents have an initial fixed position and count with an initial probabilistic information about the target dynamics and location.

The MTS solution is designed as a decision process, where: 1) each time an action is made there is a new observation that changes the target location belief, 2) there is temporal importance (i.e. the decisions order matters) and 3) the task finalizes when the target is detected.

Because the knowledge is probabilistic, one way to solve the search is to minimize the expected time, which is the mean time required to carry out the search. Other approach

is to think again about the information gathering process, where we accumulate information about the areas that we are searching. Observing a place during a period of time we can discover that the object is not there and, therefore, increment the possibility of finding it in other places. Thus, we should visit the places that have higher probability of finding the target. Hereby, the MTS objective becomes obtaining the maximum information in the minimum time possible.

The main challenges to tackle the MTS relay on the agent dynamics, that constrain the locations that we can observe [Eagle, 1984], and on the uncertainty associated with the target location and dynamics, and sensor observations. Besides, due to the problem complexity [Trummel and Weisinger, 1986], developing tractable solutions is a hard task, where the approximations should exploit the MTS inner characteristics.

The MTS is in essence an optimal search as defined in [Stone, 1975], where we can find a dual objective definition: minimizing the spent effort or the cost of looking in a determined place and minimizing the time that we employ to find the object. If we want to reduce the task time we can minimize the expected time to find the object. Defining the time to find the target as the random variable T, the solution to the MTS is given by the following equation:

$$\min(E\{T \leq k\}) \tag{3.1}$$

As we have one or more agents that make decisions using their knowledge about the world and making observations, the actions do not only modify the agent state but they also change the world belief, making every past action-observation affect the future decisions. This means that we are leading with a sequential problem. Therefore, the MTS is a decision making problem, because in order to minimize the time we have to choose the best sequence of team actions $v_{1:q}$. Thus, the optimization problem is:

$$\arg\min_{v_{1:q}}(E\{T \leq k\}) \tag{3.2}$$

Finally the MTS can also be considered a Partially Observable Markov Decision Process (POMDP), because we are observing the partial target location state [Smallwood and Sondik, 1973; Cassandra et al., 1996].

All MTS problems are probabilistic because the object or target location is unknown. Therefore if the location is, at any time, discovered, the problem is solved. This leads into the most important objective: we have to collect information about the target. As we look into the different places, our knowledge of the world increases and, in opposition, in a dynamic world, as the time passes, we have less certainty of the world. Thus, minimizing the time is similar to maximizing the accumulated information [Trummel and Weisinger, 1986]. The joint probability of detection, described previously in Section

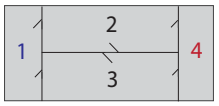
2.4.3.3, computes the accumulated information of a team path, but in this chapter we show that, in terms of time optimization, this function is not always appropriated, as the example in Proposition 3.1 illustrates. In fact only by maximizing the *increment* of information we can guarantee the time minimization.

Proposition 3.1. *Maximizing the probability of detecting the target of a sequence of observations (i.e. the agent path) does not usually minimize the time to discover the target.*

Proof. We use a counter example to demonstrate it. Given a searching region of $2x2$ connected locations, labeled as $\{1, 2, 3, 4\}$, assuming that we have an initial probability of locating the target in each location of $\{0, 0, 1, 0\}$ and that we are searching a static target, the agent following the path $\{1, 3, 2, 4\}$ will find the target sooner than following $\{1, 2, 3, 4\}$. This happens because the first path visits the target location with probability 1 before. Because both trajectories have the same joint probability of detection[1], we can state that the probability maximization does not necessarily minimize the time to find the target. □

A more detailed analysis of this proposition can be found in Section 3.3. Hereby, the MTS with constrained path should be considered as a differentiated problem within the optimal search with its own strategies, because previous existing ones are no longer valid.

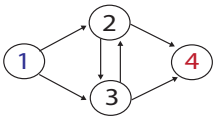
3.1.1 Example: The Connected Hallway



(a) Scenario

ROOM	Probability
1	0.1
2	0.5
3	0.2
4	0.2

(b) Target Location Belief



(c) Decision Graph

FIGURE 3.2: MTS example.

In this example we are looking for a non-moving object in a four locations world (searching space) as it is shown in Figure 3.2(a). There are two rooms with two doors and two hallways connected by a door. Starting in room 1 we can perform two actions: select the door on the left or on the right. In the hallway we can go back, to the other hallway or to the room 4 (back, left or right). We assume that when we are inside a place (room or hallway) we can see if the object is there. We also know, by experience, the target

[1] In this case the probability of detection is just the sum of the probabilities of the visited locations, that is 1 for both paths (Eq. 2.22).

location belief, which is the probability of the object being in each room (Figure 3.2(b)). We wonder what is the sequence of decisions to find the object in minimum time.

As we want to find the object, we need to visit all rooms, because the sum of all the probabilities of finding the object becomes 1. With one agent, two of the possible sequences of actions (v and v') that cover the entire world with only 3 actions are:

- v: From room 1 select the door on the left to hallway 2, then go to hallway 3 and finally end in room 4.
- v': From room 1 select the door on the right to hallway 3, then go to hallway 2 and finally end in room 4.

The solution alternatives are a tuple with the initial state (room) $s^0 = 1$ and the sequence of actions (doors) v. For a single-agent configuration the previous two alternatives are: $v = \{left, right, right\}$ and $v' = \{right, left, left\}$. As the agent is deterministic (i.e. starting in state 1 and performing action left it always arrives to hallway 2), we can also codify our sequence of actions by the visited states: $s^{0:3} = \{1, 2, 3, 4\}$ and $s'^{0:3} = \{1, 3, 2, 4\}$. It is important to note that if the time is not important or the probability to find the object is the same in each room, any of the paths will fulfill our objective. Otherwise the best decision, in this example, is to select first the hallway that has higher probability. Using the probability values in Figure 3.2(b) and assuming that each action consumes one time step, we compute the expected time of finding the object for each path with this equation:

$$\mu(v) = \sum_{k=1}^{\infty}(1 - P(\text{Detect the target before instant } k)) \tag{3.3}$$

$$\mu(v) = 1 + (1 - 0.1) + (1 - (0.1 + 0.5)) + (1 - (0.1 + 0.5 + 0.2)) = 2.5$$
$$\mu(v') = 1 + (1 - 0.1) + (1 - (0.1 + 0.2)) + (1 - (0.1 + 0.2 + 0.5)) = 2.8$$

which proves that the best sequence of actions is v (the one with lower expected time).

In the case that there are more than one agent (e.g., $q = 2$), we can improve the solution. There is only one optimal alternative starting in room 1: at the same time agent 1 makes $v_1 = \{left, left\}$, and agent 2 makes $v_2 = \{right, right\}$. The expected time to find the object using the two agents and actions $v_{1:2} = \{v_1, v_2\}$ is:

$$\mu(v_{1:2}) = 1 + (1 - 0.1) + (1 - (0.1 + 0.5 + 0.2)) + (1 - (0.1 + 0.5 + 0.2 + 0.2)) = 2.1$$

Note that the expected time is lower that in the single-agent best solution. However, it is worth highlighting that with the multi-agent configuration some problems appear

due to the coupling. For instance, when both agents are in the same room they have to join their observations.

3.1.2 Example: Search and Rescue (SaR)

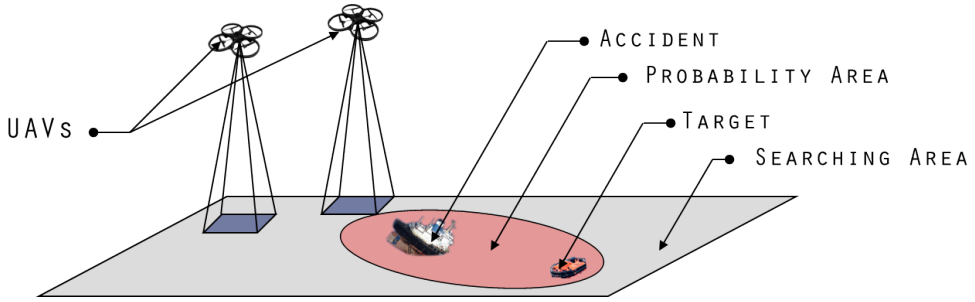


FIGURE 3.3: MTS search and rescue scenario.

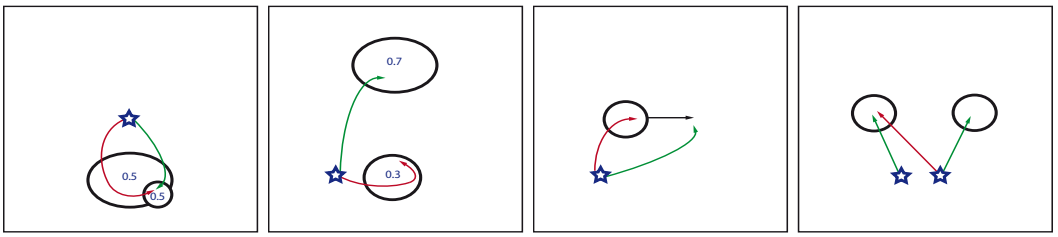
We are going to use this example through this thesis to clarify the concepts and apply the MTS algorithms, but the reader should be able to abstract the ideas into any of the possible problem setups. We have a ship wreck accident and a rubber raft lost in the sea with some survivors [Bourgault et al., 2003]. The ship has sent a stress signal before falling down into the ocean and the rescue operations center has to prepare a searching strategy to find the humans alive without health complications. The resources that they count with to accomplish the mission are a fleet of Unmanned Air Vehicles (UAVs). The time is critical and is given by the time elapsed between the fleet takeoff and the raft discovery. Thus, the objective is to plan a team trajectory for the aircrafts that finds the raft in minimum time. Figure 3.3 schematize the SaR example as a search in a 2D space.

3.1.3 Representative Scenarios in MTS

In this section we describe four important illustrative scenarios that we have to tackle in the MTS problem. With those examples we show several important facts: the importance of time, the drawbacks of using myopic solutions, the problems related with the dynamic targets and the benefits of using multiple agents.

Figure 3.4 shows the illustrative examples schematically. The blue stars are the agents and the red arrows that start from the agents represent bad trajectories, while the green arrows are the right MTS solutions. The black ellipses show the initial regions where the target can be (i.e. a location probability distribution) and the numbers inside these ellipses indicate the total probability of the region. Also, the black arrow starting at the ellipse (Figure 3.4(c)) summarizes the target dynamics.

The first example, Figure 3.4(a), presents a non-uniform target location belief with a huge abnormality with total probability of 0.5. Depending on the instant that we



(a) Minimum Time (b) Non-myopic (c) Dynamic Target (d) Multiple Agents

FIGURE 3.4: Illustrative scenarios.

observe the abnormality the detection time changes considerably, meaning that the observation order matters. The green trajectory is the best MTS trajectory because it accumulates faster the probabilities, while the red one shows a trajectory that finds the target later. Only strategies that take into account the time produce solutions like the green trajectory. Moreover we can see that greedy strategies also fail. In fact, all the solutions that only maximize the probability of detection like [Eagle, 1984; Bourgault et al., 2004; Lavis et al., 2008; Gan and Sukkarieh, 2010] produce wrong behaviors in this scenario.

The second example, Figure 3.4(b), represents the locality drawbacks. When we only use a limited horizon window to make the decisions, we cannot see the whole problem and therefore, our solutions are myopic. These local decisions could affect future decision in a wrong way. In this scenario, the agent initial location is near to a low probability region and if the decision horizon is small, it will think that the best decision is to observe the closer region, when the best choice is to go towards the further region because it has more probability. To reduce the myopicity we can either compute a longer decision horizon (ideally an infinite horizon) or use a function that computes the future expectation. Solutions like [Yang et al., 2002; Bourgault et al., 2003; Mathews et al., 2007; Gan and Sukkarieh, 2010] are not able to tackle properly these types of scenarios.

The third example, Figure 3.4(c), shows the problem arisen from searching a moving target. The best agent actions are the ones that anticipate the future target location. The red trajectory describes a solution that does not predict the target location in later instants, while the green one represents a better agent decision because it uses the target dynamics information to plan ahead. Therefore, the methods assuming an static target like [Blum et al., 2003; Mathews, 2008; Gan and Sukkarieh, 2010; Sarmiento et al., 2009] do not succeed in this scenario.

The last illustrative scenario, Figure 3.4(c), shows the benefits of using a team of agents. In this case we have two agents and two regions with the same total probability. The optimal MTS solution makes each agent go toward a different region splitting the effort and achieving sooner target detections. Methods like [Eagle and Yee, 1990; Sarmiento et al., 2009] do not handle properly these types of scenarios.

3.1.4 Applications

There are multiple applications for the MTS in the optimal search field such as:

- The boxes problem, where a set of boxes containing an object has to be found. We have to decide the opening order to find the object as soon as possible [Zimmerman, 1959; Stone, 1975].
- Search and Rescue (SaR), where there is a ship wreck accident and we know that there are survivors lost in the sea. As the time to find them is crucial, we need a decision system that guides us to find the survivors alive [Bourgault et al., 2003].
- Survey, where there is an unexplored region that should be observed or evaluated to collect information about the area. The task is to accumulate the maximum information in minimum time.
- Resources finding, where we need to find hidden water reserves in a dry area and we have a probability map of the most probable locations. The mission is to find the resources efficiently.
- Airplane crash,, where there is an airplane accident, and we want to find out which was the failure that caused the plane to fall down. For that purpose, we need to find and recover the black box with the flight recordings as soon as possible [Stone, 2011].
- Treasure hunting, where some of the pirate gold of central America has been lost after an environmental catastrophe. This gold has an important historical value for world community and has to be found quickly [Stone, 1992].
- Underwater missions: The uncertain nature of the localization in underwater environments offers numerous applications for the MTS. For instance the famous USS Scorpion submarine search was solved by determining the best scanning order of the possible locations [Koopman, 1980]. We can also mention beacons finding or tectonic plate crevices localization.
- Document search: Although searching a document is usually done by sequential opening or by indexing, we can develop a search that takes into account the probability of being in each folder and find the document in the minimum time.
- Network search: We can generalize the network searching task following the same ideas used in document search. The MTS decision solution is then to decide the network route from your computer to find what you are looking for.
- The lost cat: This application is just an example of looking for targets in indoor scenarios. The cat has disappeared and we need to find it as soon as possible. The solution is to find the right order of rooms to explore [Facebook, 2011].

- Earth-like planet classification: Assuming that we have the possibility to explore the universe with sensing spaceships, we need to determine the trajectory to find a habitable planet as soon as possible. Without spaceships, we can determine the space regions sequence that should be observed with telescopes and make an autonomous system that finds the planet for us.

The MTS outside the optimal search field is related with other applications such as:

- The impatient trader problem, where we have a trader that has to make a set of purchases that provide the best benefit in minimal time [Feillet et al., 2005]. The decision problem is to select the best purchases.
- Travel salesman problem with time decreasing benefits: At each city the salesman gets a reward but this reward decreases until he visits the city. The decision making problem is solved by finding the best city visiting order that makes the salesman reward maximum [Feillet et al., 2005].
- The package-delivery problem: We need to deliver a set of packages but the reward that we obtain for each packet depends on the delivery time. Thus we are looking for the delivering order that makes our benefits maximum [Savelsbergh and Sol, 1995].
- The orienteering problem: Assuming that we start in a fixed location we look for a path of fixed length that maximizes the reward obtained [Blum et al., 2003].

More applications are summarized in [Benkoski et al., 1991] such as: surveillance and reconnaissance, minerals search, glaucoma detection, industrial applications, and search for the remains of a Russian satellite.

3.2 Framework

We describe the main concepts involved at any MTS problem: the world, the agents, the sensor, the target, and the target belief. Some of the elements are already addressed in Chapter 2 and in the previous sections of this chapter, but here we redefine clearly the notation used in the rest of the thesis.

World. It is the space containing the targets and observed by the agents. In general, we consider the world as a convex region denoted by $\Omega \in \mathbb{R}^n$. In the case of searching a target on the ground the world is a 2D region ($\mathbb{R}^2$).

Agents. They are every independent sensing platform that takes active part in the search by making observations (i.e. using sensors). Each agent has mobility

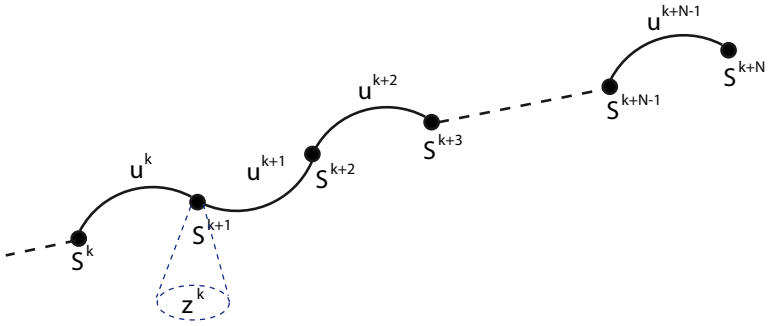


FIGURE 3.5: Agent decisions vector v^k starting at state s^k.

through the world and its dynamics are determined by the possible actions. The way of modeling the actions depends on the action state space: discrete (e.g. cardinal directions) or continuous (e.g. turn rate). Thus, the sensing agent will follow a trajectory generated by a decision making algorithm as an action vector v^k at each action planning instant k. During every action planning process, the action vector of each sensor platform is jointly optimized to fulfill the mission objective. As Figure 3.5 shows, we have an agent with an action vector composed of N actions $v^k = \{u^k, \cdots, u^{k+N-1}\}$, which moves the sensing platform from s^k into $s^{k+1}, \cdots, s^{k+N}$ as time increases. In the case of having a team of q agents, the joint action vector is $v^k_{1:q} = v^k_Q = \{v^k_1, \ldots, v^k_q\}$, and the agent states are defined by the following matrix:

$$s^{k:k+N}_{1:q} = s^{k:k+N}_{Q} = \begin{bmatrix} s^k_1 & \cdots & s^{k+N}_1 \\ \vdots & \ddots & \vdots \\ s^k_q & \cdots & s^{k+N}_q \end{bmatrix} \tag{3.4}$$

Target. It is the object that we are looking for. This item can be static (i.e. its location does not change along the time) or dynamic (i.e. it can move from one place to another). Its location τ^k and their movements are uncertain, although probabilistically modeled. In the static case, the target location is given by the location probability distribution over the world and, in the dynamic case, the target location is given by the initial location probability distribution $P(\tau^0)$ and the probabilistic motion model at instant k. The motion model is a function that states the probability of being at τ^k given the last location state τ^{k-1},

$$P(\tau^k|\tau^{k-1}) \tag{3.5}$$

In discrete worlds, the motion model can also be represented by a transition matrix A_{ij} that describes the probability of transiting from state j to state i.

Sensors. They are the items capable of observing and collecting information of the world. They are on board the agents and we assume that they have the same location as the agent, thus, their states match. The observations are used by the agents to update their belief and decide which actions are the best. The sensor model can be discrete (e.g. binary sensor) or continuous (e.g. exponential) and represents the likelihood of producing a target detection event D for the observation z_i at time step k, if the sensor is at state s_i^k and the target position is placed at τ^k.

$$P(z_i^k = D|\tau^k, s_i^k) \tag{3.6}$$

The complement of Eq. 3.6 is the sensor non-detection likelihood,

$$P(z_i^k = \bar{D}|\tau^k, s_i^k) = 1 - P(z_i^k = D|\tau^k, s_i^k) \tag{3.7}$$

To simplify the expressions, we use D_i^k and $\bar{D}_i^k$ to represent $z_i^k = D$ and $z_i^k = \bar{D}$ respectively.

Target belief. It is the subjective knowledge that the agents have about the target location and it is described by a probability density function. The location of the target at time k, τ^k, is uncertain and can be represented as the belief b_τ^k,

$$b_\tau^k = P(\tau^k|z_{1:q}^{1:k}, s_{1:q}^{1:k}) \tag{3.8}$$

describing the probability of the target being in a specific position, conditioned on all the previous sensor observations $z_{1:q}^{1:k}$ made at sensor states $s_{1:q}^{1:k}$. Therefore, each agent starts with a prior knowledge (b_τ^0) about the target that is updated using all the observations along the time.

Note that when we refer to the variables without underscripts (s^k instead of s_i^k) we are talking about the single agent case.

3.3 Minimum Time Optimality Analysis of The Maximum Detection Strategy: The Necessary Condition

We have proved in Section 3.1, by a counter example, that maximizing the probability of detecting the target of a sequence of observations does not necessarily minimize the time to discover the target. In this section we show mathematically in which cases this is true and we provide the necessary condition to assure that maximizing the detection implies minimizing the time.

To prove this condition, on one hand we define $P(k, v)$ as the probability of detection up to instant k when the agents decisions plan is v. When the decision plan is optimal

v^* the probability of detection is optimal: $P(k, v^*)$. Therefore, for a decision plan of N steps this probability is maximal comparing with the rest of the solutions:

$$P(N, v^*) \geq P(N, v) \tag{3.9}$$

On the other hand the expected time of detection $\mu(N, v)$ in the interval $(1, N)$ for a given decision plan v is defined by the following equation [Stone, 1975]:

$$\mu(N, v) = E(1 \leq T \leq N) = \sum_{k=1}^{N}(1 - P(k, v)) \tag{3.10}$$

When the decision plan is optimal its value is minimal comparing with any other solution:

$$\sum_{k=1}^{N}(1 - P(k, v^*)) \leq \sum_{k=1}^{N}(1 - P(k, v)) \tag{3.11}$$

We want to analyze when maximizing the probability of detection minimizes the time. Therefore, we want to evaluate when the following equation is true:

$$P(N, v^*) \geq P(N, v) \stackrel{?}{\Longrightarrow} \sum_{k=1}^{N}(1 - P(k, v^*)) \leq \sum_{k=1}^{N}(1 - P(k, v)) \tag{3.12}$$

In order to do that, we first operate the right side of Eq. 3.12 by regrouping terms,

$$N - \sum_{k=1}^{N} P(k, v^*) \leq N - \sum_{k=1}^{N} P(k, v) \tag{3.13}$$

We next eliminate N and rearrange the expression,

$$\sum_{k=1}^{N} P(k, v^*) \geq \sum_{k=1}^{N} P(k, v) \tag{3.14}$$

We next separate the last term (N) from both summations,

$$P(N, v^*) + \sum_{k=1}^{N-1} P(k, v^*) \geq P(N, v) + \sum_{k=1}^{N-1} P(k, v) \tag{3.15}$$

And finally, we regroup terms again, obtaining a necessary condition for minimizing the expected time:

$$P(N, v^*) - P(N, v) \geq \sum_{k=1}^{N-1} P(k, v) - \sum_{k=1}^{N-1} P(k, v^*) \tag{3.16}$$

Note that the left side of the inequality is the difference between the optimal and the non optimal probability of detection.

If Eq. 3.16 and 3.9 hold then Eq. 3.12 is satisfied. Thus we analyze two different subcases:

1. If $\sum_{k=1}^{N-1} P(k,v) \leq \sum_{k=1}^{N-1} P(k,v^*)$ the left side of Eq. 3.16 is negative and due to Eq. 3.9, Eq. 3.12 is true.
2. If $\sum_{k=1}^{N-1} P(k,v) > \sum_{k=1}^{N-1} P(k,v^*)$ we can only guaranty the satisfactibility of Eq. 3.12 if Eq. 3.16 holds. Otherwise, Eq. 3.12 is false.

Following the second subcase, if Eq. 3.16 is not satisfied Eq. 3.12 is not true:

$$P(N,v^*) - P(N,v) < \sum_{k=1}^{N-1} P(k,v) - \sum_{k=1}^{N-1} P(k,v^*) \implies \neg\text{Eq. 3.12} \tag{3.17}$$

Therefore, we can state that maximizing the probability of detection does not imply minimizing the expected time of detection.

One way to force Eq. 3.12 is to find an algorithm that satisfies the following sufficient condition [Stone, 1975]:

$$P(k,v^*) \geq P(k,v) \quad \forall k \in (1,N) \tag{3.18}$$

Meaning that if all increments along the agent optimal path are higher than any other solution we are minimizing the time. For instance, the Gaussian target location distribution is a case where following the maximum slope is always the best strategy for the single agent case because the probability always incremented towards the gradient direction. Therefore in MTS single agent instances with Gaussian location distribution maximizing the detection is a good approach.

In the case that the agent paths are not constrained by the agent dynamics the MTS becomes the density problem [Stone, 1989] where we can use the algorithm in [Stone, 1975] that satisfies Eq. 3.18 and consequently minimizes the time.

Corollary 3.2. *The general necessary condition that should be satisfied in order to minimize the time is:*

$$\forall i \in (1,N) \quad \sum_{k=N-i+1}^{N} P(k,v^*) - \sum_{k=N-i+1}^{N} P(k,v) \geq \sum_{k=1}^{N-i} P(k,v) - \sum_{k=1}^{N-i} P(k,v^*) \tag{3.19}$$

In the MTS where the agents have restricted dynamics, making the path constrained, we cannot guarantee that this condition is satisfied when maximizing the probability because it depends on the scenario. Therefore we conclude that, if $\exists i$ when maximizing $P(N,v)$ that makes Eq. 3.19 not true, maximizing the probability does not minimize the detection time.

3.4 Multiagent Bayesian Search for Dynamic Targets

In Section 2.4.3.2 of the background chapter, we have shown how to compute the joint probability of non-detection of a multiagent set of trajectories for a static target. Here we generalize it for the dynamic target case by showing the general expressions to calculate the probability of detecting the target at any future time step. We also provide some nice mathematical properties of the probability functions that are useful for obtaining these expressions and for designing the MTS decision strategies (i.e. utility functions).

3.4.1 Bayes Basic Concepts

In this section we revise some important Bayesian properties to derive the joint probability of detection events, that in the next section will be used to design the MTS strategies.

Table 3.1 presents the necessary basic probability operations. The first operation is the conditional probability Bayes rule. The marginalization operator, second row, shows how to include the event C into the probability of occurring A conditioned by the event B. The third operation express that the probability of the union events that occurs in the instants interval $(1, N)$ is the sum of the union of the events that happen in each instant when the previous events have not occurred. The fourth one is the Morgan law, where the negation of the events union is the intersection of the negated events. Finally the last operation defines that the probability of the events union is equal to one minus the negation of that union.

TABLE 3.1: Basic probability operations

Operators	**Expressions**
Bayes Rule	$P(A,B\|C) = P(A\|B,C)P(B\|C)$
Marginalization	$P(A\|B) = \int P(A,C\|B)dC$
$\cup \rightarrow \sum$	$P(\bigcup_{i=1:q,j=1:N} X_i^j) = \sum_{j=1:N} P(\bigcup_{i=1:q} X_i^j, \overline{X}_{1:M}^{1:j-1})$
$\cup \rightarrow \cap$	$P(\overline{\bigcup_{j=1:N} X^j}) = P(\bigcap_{j=1:N} \overline{X}^j)$
Complementary	$P(\bigcup_{j=1:N} X^j) = 1 - P(\overline{\bigcup_{j=1:N} X^j})$

3.4.2 Joint Probability of Detection Events

We describe the natural probability function to evaluate the searching trajectory of a team of agents. In this section we present the joint detection probability and in the next section the non-detection joint probability. Although they are equivalent, both are useful to design the MTS strategies.

Proposition 3.3. *The probability of not detecting a target (J_{nd}) in N instants is probabilistically complementary to the probability of detecting the target (J_d) at any N consecutive instants: $J_{nd} = 1 - J_d$*

Proof. By definition, J_d is the union of the detection events along the agents trajectory:

$$J_d \triangleq P(\bigcup_{j=1:N} D_{1:q}^{k+j} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) \tag{3.20}$$

And J_{nd} is the intersection of the non-detection events:

$$J_{nd} \triangleq P(\bigcap_{j=1:N} \overline{D}_{1:q}^{k+j} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) \tag{3.21}$$

Applying the complementary and the negation operation sequentially we obtain the proof:

$$\begin{aligned} J_d &= P(\bigcup_{j=1:N, i=1:q} D_i^{k+j} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) = \\ &= 1 - P(\overline{\bigcup_{j=1:N, i=1:q} D_i^{k+j}} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) = \\ &= 1 - P(\bigcap_{j=1:N, i=1:q} \overline{D}_i^{k+j} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) = 1 - J_{nd} \end{aligned} \tag{3.22}$$

□

This equivalence is important in order to simplify the probabilistic equations because the computation of the union of events is more expensive than the computation of the intersection when there is coupling involved[2].

[2] $P(A \cup B) = P(A) + P(B) - P(A, B)$ while $P(A, B) = P(A)P(B)$ when A and B are independent.

The multiagent joint probability of detection from instant k up to $k+N$ is computed as follows[3]:

$$J_d(s_{1:q}^{k+1:k+N}) = P(\bigcup_{j=1:N,i=1:q} D_i^{k+j} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) = \tag{3.23}$$

Applying $\cup \rightarrow \sum$

$$= \sum_{j=1}^{N} P(\bigcup_{i=1:q} D_i^{k+j}, \overline{D}_{1:q}^{k+1:k+j-1} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) = \tag{3.24}$$

Marginalizing over τ^{k+j}

$$= \sum_{j=1}^{N} \int_{\tau^{k+j}} P(\bigcup_{i=1:q} D_i^{k+j}, \tau^{k+j}, \overline{D}_{1:q}^{k+1:k+j-1} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) d\tau^{k+j} = \tag{3.25}$$

Bayes rule

$$= \sum_{j=1}^{N} \int_{\tau^{k+j}} P(\bigcup_{i=1:q} D_i^{k+j} | \tau^{k+j}, s_{1:q}^{k+j}) P(\tau^{k+j}, \overline{D}_{1:q}^{k+1:k+j-1} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) d\tau^{k+j} = \tag{3.26}$$

Defining $\tilde{b}_\tau^{k+j|k+j-1} = P(\tau^{k+j}, \overline{D}_{1:q}^{k+1:k+j-1} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k})$

$$= \sum_{j=1}^{N} \int_{\tau^{k+j}} P(\bigcup_{i=1:q} D_i^{k+j} | \tau^{k+j}, s_{1:q}^{k+j}) \tilde{b}_\tau^{k+j|k+j-1} d\tau^{k+j} = \tag{3.27}$$

Applying the complementary of the union of events

$$= \sum_{j=1}^{N} \int_{\tau^{k+j}} \left[1 - P(\overline{D}_{1:q}^{k+j} | s_{1:q}^{k+j}, \tau^{k+j})\right] \tilde{b}_\tau^{k+j|k+j-1} d\tau^{k+j} = \tag{3.28}$$

Assuming measurement independence

$$= \sum_{j=1}^{N} \int_{\tau^{k+j}} \left[1 - \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s_i^{k+j}, \tau^{k+j})\right] \tilde{b}_\tau^{k+j|k+j-1} d\tau^{k+j} \tag{3.29}$$

Note that we have used similar notation for the term that we have substituted in Eq. 3.27, $\tilde{b}_\tau^{k+j|k+j-1}$, to the target location belief $b_\tau^{k|k-1}$ at the RBE algorithm (Section 2.4.3.1). This is because it is closely related to the belief $b_\tau^{k|k-1} = P(\tau^k | z^{1:k-1}, s^{1:k-1})$. In fact, we will show next how computing $\tilde{b}_\tau^{k+j|k+j-1}$ is analogous to do Bayesian filtering, with the difference that there is not normalization at the update step.

The calculation of $\tilde{b}_\tau^{k+j|k+j-1}$ is effectuated by computing recursively its value from $j=2$ up to $j=N$, using Eq. 3.34, considering that $\tilde{b}_\tau^{k+1|k+0} = b_\tau^{k+1|k}$ (i.e. the predicted target belief at instant $k+1$ with all the observations assimilated up to k).

[3]We remind that the target movement is defined by the transition probability $P(\tau^{k+1}|\tau^k)$ and the sensor probability of detection is described by $P(D^k|s^k, \tau^k)$.

$$\tilde{b}_{\tau}^{k+j|k+j-1} = P(\tau^{k+j}, \overline{D}_{1:q}^{k+1:k+j-1} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) = \tag{3.30}$$

Marginalizing over τ^{k+j-1}

$$= \int_{\tau^{k+j-1}} P(\tau^{k+j}, \tau^{k+j-1}, \overline{D}_{1:q}^{k+1:k+j-1} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) d\tau^{k+j-1} = \tag{3.31}$$

Applying Bayes rule assuming independence of τ^{k+j} on any variable except τ^{k+j-1}

$$= \int_{\tau^{k+j-1}} P(\tau^{k+j} | \tau^{k+j-1}) P(\tau^{k+j-1}, \overline{D}_{1:q}^{k+1:k+j-1} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) d\tau^{k+j-1} = \tag{3.32}$$

Bayes rule assuming independence of $\overline{D}_{1:q}^{k+j-1}$ on everything but τ^{k+j-1} and $s_{1:q}^{k+j-1}$

$$= \int_{\tau^{k+j-1}} P(\tau^{k+j} | \tau^{k+j-1}) P(\overline{D}_{1:q}^{k+j-1} | \tau^{k+j-1}, s_{1:q}^{k+j-1}) P(\tau^{k+j-1}, \overline{D}_{1:q}^{k+1:k+j-2} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) d\tau^{k+j-1} = \tag{3.33}$$

Assuming that the measurements are independent on other measurements and agents

$$= \int_{\tau^{k+j-1}} P(\tau^{k+j} | \tau^{k+j-1}) \prod_{i=1}^{q} P(\overline{D}_{i}^{k+j-1} | \tau^{k+j-1}, s_{i}^{k+j-1}) \tilde{b}_{\tau}^{k+j-1|k+j-2} d\tau^{k+j-1} \tag{3.34}$$

In the particular case of a static target ($\tau^{k+1} = \tau^k = \tau$) the prediction disappears and we only have to marginalize with respect to τ. Therefore Eq. 3.29 becomes:

$$\begin{aligned} J_d(s_{1:q}^{k+1:k+N}) &= \sum_{j=1}^{N} \int_{\tau} \left[1 - P(\overline{D}_{1:q}^{k+j} | \tau, s^{k+j})\right] \prod_{l=1}^{j-1} P(\overline{D}_{1:q}^{k+l} | \tau, s^{k+l}) b_{\tau}^{k} d\tau = \\ &= \int_{\tau} \sum_{j=1}^{N} \left[1 - P(\overline{D}_{1:q}^{k+j} | \tau, s^{k+j})\right] \prod_{l=1}^{j-1} \prod_{i=1}^{q} P(\overline{D}_{i}^{k+l} | \tau, s^{k+l}) b_{\tau}^{k} d\tau \end{aligned} \tag{3.35}$$

3.4.3 Joint Probability of Non-Detection Events

Analogously we can derive the joint non-detection probability for a team of agents along their trajectory, by computing the intersection of the non-detection events:

$$J_{nd}(s_{1:q}^{k+1:k+N}) = P(\overline{D}_{1:q}^{k+1:k+N}|s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) = \tag{3.36}$$

Separating the last non-detection event

$$= P(\overline{D}_{1:q}^{k+N}, \overline{D}_{1:q}^{k+1:k+N-1}|s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) = \tag{3.37}$$

Marginalizing over τ^{k+N}

$$= \int_{\tau^{k+N}} P(\overline{D}_{1:q}^{k+N}, \tau^{k+N}, \overline{D}_{1:q}^{k+1:k+N-1}|s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) d\tau^{k+N} = \tag{3.38}$$

Bayes Rule

$$= \int_{\tau^{k+N}} P(\overline{D}_{1:q}^{k+N}|s_{1:q}^{k+N}, \tau^{k+N}) P(\tau^{k+N}, \overline{D}^{k+1:k+N-1}|s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) d\tau^{k+N} \tag{3.39}$$

Defining $\tilde{b}_{\tau}^{k+N|k+N-1} = P(\tau^{k+j}, \overline{D}_{1:q}^{k+1:k+N-1}|s_{1:q}^{1:k+N}, z_{1:q}^{1:k})$

$$= \int_{\tau^{k+N}} P(\overline{D}_{1:q}^{k+N}|s_{1:q}^{k+N}, \tau^{k+N}) \tilde{b}_{\tau}^{k+N|k+N-1} d\tau^{k+N} = \tag{3.40}$$

Assuming measurement independence

$$= \int_{\tau^{k+N}} \prod_{i=1}^{q} P(\overline{D}_{i}^{k+N}|s_{i}^{k+N}, \tau^{k+N}) \tilde{b}_{\tau}^{k+N|k+N-1} d\tau^{k+N} \tag{3.41}$$

Note that substituting N by any instant j we are computing the non-detection joint probability up to j.

In the particular case of the static target the non-detection function becomes the one presented by [Bourgault et al., 2004; Mathews, 2008] and already shown in Section 2.4.3.2:

$$J_{nd}(s_{1:q}^{k+1:k+N}) = \int_{\tau} \prod_{j=1}^{N} \prod_{i=1}^{q} P(\overline{D}_{i}^{k+j}|s_{i}^{k+j}, \tau) b_{\tau}^{k} d\tau \tag{3.42}$$

3.5 Tractable MTS Decision Making Strategies

This section concerns about the developing of tractable strategies or utility functions suitable to the MTS problem. In practice these strategies are used by the controller layer to compute the optimal agents actions. Our starting point is the work done by [Mathews, 2008] that postulates that assuming that the agent never detects the target [Stone, 1989] we can predict the uncertain target location in a deterministic way using a Bayesian inference and therefore, simplify the general (POMDP) problem into a deterministic one. Thus, we use a model predictive controller [Bertsekas, 1995] as the solution for the controller layer. Figure 3.6 shows the target location belief along the decision process, where we can see that there is only one path on the belief graph, so we do not need to compute the expectation.

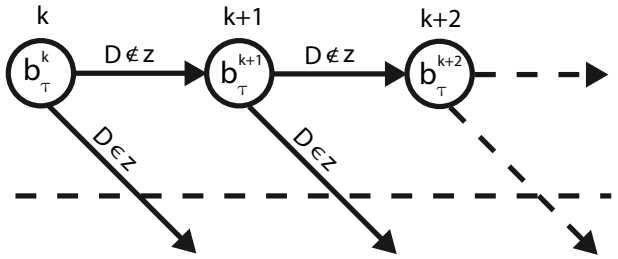


FIGURE 3.6: Belief generation along the agent decisions (extracted from [Mathews, 2008]).

To design the model predictive model we start from the open loop controller defined by the following function: the expectation of the accumulated rewards over the beliefs,

$$J(s^k, v^k, b_\tau^k) = E_{b_\tau^k, \cdots, b_\tau^\infty} \left\{ \sum_{j=k}^{\infty} R(u^j, b_\tau^j) \right\} \tag{3.43}$$

The solution are the optimal actions that maximize the reward obtained:

$$v^{k*} = \arg\max_{v^k} J(s^k, v^k, b_\tau^k) \tag{3.44}$$

Instead of computing all the actions we can approximate this scheme by using a Receding Horizon Controller [Bertsekas, 1995]. Let the horizon be N, the utility function becomes:

$$J(s^k, v^k, b_\tau^k) = E_{b_\tau^k, \cdots, b_\tau^{k+N}} \left\{ \sum_{j=k}^{k+N} R(u^j, b_\tau^j) + E_{b_\tau^{k+N}} \left\{ \hat{H}^{k+N}(b_\tau^{k+N}) \right\} \right\} \tag{3.45}$$

where $\hat{H}^{k+N}$ is the estimated reward-to-go from the selected horizon N to the future.

Now assuming that it is a deterministic decision problem because of the unique observation (non-detection), as shown in Figure 3.6, we can predict the target belief $\tilde{b}_\tau^k$ depending on the decision chosen, and evaluate the agents decisions using the equations presented in Section 3.4. Thus, the tractable utility function for the model predictive controller is the following expression:

$$J(s^k, v^k, b_\tau^k) \triangleq \sum_{j=k}^{k+N} R(u^j, \tilde{b}_\tau^j) + \hat{H}^{k+N}(\tilde{b}_\tau^{k+N}) \tag{3.46}$$

Using this approximation we design a tractable decision model that depends on the system dynamics and the prior target distribution.

Let us define the MTS as a time decision making problem. Due to the uncertainty at the target location, the time spent to finish the task, i.e. to find the target, is a random variable T with density function $P(T \leq k)$, that is the probability of detecting the target before instant k. The value of T can be computed by its expectation.

We define the expected time to find a target for an action plan $v^k_{1:q}$ as $\mu(v^k_{1:q})$.

$$\mu(v^k_{1:q}) = \sum_{k=1}^{\infty} \left[1 - P(\text{Joint detections before } k \text{ using plan } v^k_{1:q})\right] \tag{3.47}$$

We are looking for the optimal plan $v^{k*}_{1:q}$, that satisfies the following condition:

$$\mu(v^{k*}_{1:q}) \leq \mu(v^k_{1:q}) \tag{3.48}$$

This can be done by minimizing $\mu(v^k_{1:q})$ but also we can maximize the increment of the probability of detection (see Section 3.3). Therefore, the strategy that we are looking for is the one that makes the team observe sooner the regions with high probability of finding the target. But to solve the constrained path MTS there is not a global optima tractable algorithm. First of all, computing the expected time up to infinity (Eq. 3.47) is untraceable [Bourgault et al., 2003] and secondly, Eq. 3.19 is not guaranteed because of the agent constraints.

3.5.1 Decision Model

We present a time decision tractable model to deal with the MTS problem with location uncertainty when the agent has spatial constraints. This approach avoids solving the whole problem, and assumes limited rationality and that the information loses importance/credibility as the time passes [Simon et al., 2008]. Within this framework, the MTS problem becomes a cognitive process where different evaluation systems take active part using the filtered information b^k_τ. As the reward evaluation system can be divided in three subsystems [Brocas and Carrillo, 2008], in the following we present the expressions of the immediate reward (R_I), the short term reward (R_N) and the future or long term reward (R_H)[4]. Figure 3.7 shows the decision systems that takes active part on the MTS.

Stating these ideas as a Model Predictive Controller we have the following utility function:

$$J(s^k, v^k, b^k_\tau) = R_I(s^k, b^k_\tau) + R_N(s^k, v^k, b^k_\tau) + R_H(s^{k+N}, \tilde{b}^{k+N}_\tau) \tag{3.49}$$

[4]In works like [Mathews, 2008; Bertsekas, 1995] R_N and R_H terms are defined as the loss and the terminal functions.

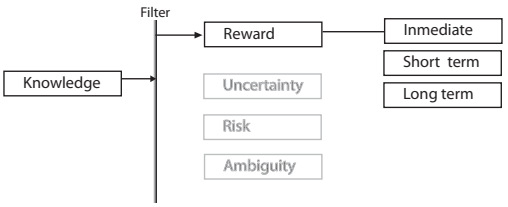


FIGURE 3.7: Decision model based on rewards (adapted from [Brocas and Carrillo, 2008]).

Figure 3.8 shows visually the agent actions and states (i.e trajectory) and the action range of each of the reward terms.

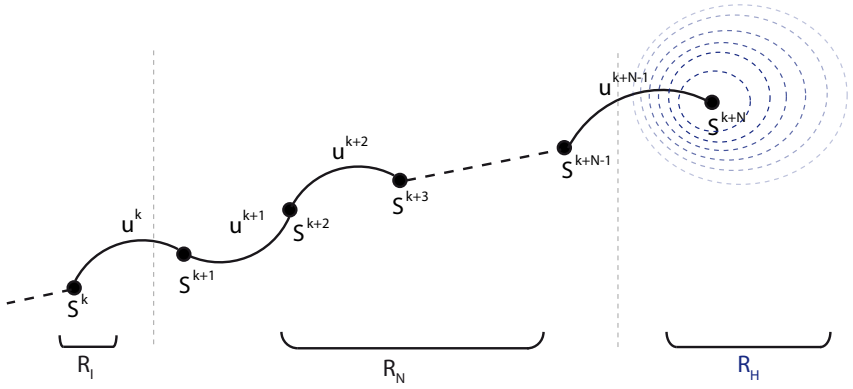


FIGURE 3.8: Decision model with the three rewards system: R_I, R_N and R_H.

It should be noted that the immediate reward does not depend on the time, just on the estimated observation given at the state s^k with the prior location distribution b_τ^k, while the short reward term depends on the N decisions taken and therefore N instants. R_H is the future expectation and can be approximated heuristically. When an algorithm just uses R_I is considered greedy or myopic and when it only uses R_N is a receding horizon controller with no terminal cost. Using the R_H term the algorithm becomes heuristically informed. Depending on the quality of this three functions and the model precision the decision model will be better. The system or world dynamics also have an impact in the design of the utility function. For instance, systems with low valuable information or hardly changing unknown dynamics need a short term strategy or even greedy, because the information used to compute the plan is quickly outdated. Also, for really stable systems with no long range dependencies R_H can be set to zero [Mathews, 2008; Bertsekas, 1995], because the myopicity is solved by replanning. Finally we should take into account how fast we want a solution. For instance, if we are designing an online algorithm, we need strategies with low delay answer.

3.5.2 Strategies

In the following sections we propose three MTS strategies that use the tractable decision making model. The first one shows a finite version of the expected time that can be used as the R_N term to build a receding horizon controller. The second one exploits the idea of discounted time functions applied to the probability of detection to build an alternative function for R_N. Finally, the third one raises the heuristic approach that can be used to build a non-myopic algorithm. Moreover we discuss two other particular strategies that are interesting for a Gaussian target location model.

All these strategies complement the existing joint probability of detection/non-detection for the optimal search, focusing on the time reduction.

3.5.2.1 Minimizing the Local Expected Time

If we want to reduce the task time, one of the approaches is to minimize the expected time to find the object. Defining the time to find the target as the random variable T, we can compute the density function $P(T \leq k)$ for any instant $k \geq 1$. The Expected Time (ET) or the expected value of T is [Papoulis and Pillai, 2002; Stone, 1975]:

$$E\{T\} = \sum_{k=1}^{\infty} (1 - P(T \leq k)) \tag{3.50}$$

The computation of ET for infinite terms is intractable [Bourgault et al., 2003], but taking into account the property described by [Feller, 1966] where Eq. 3.50 can be used for density functions that sum less than one, we can still apply Eq. 3.50 for a limited horizon N[5]. Therefore, we can compute the Local Expected Time (LET) as:

$$LET\{1 \leq T \leq N\} = \sum_{k=1}^{N} (1 - P(1 \leq T \leq k)) \tag{3.51}$$

In fact, the decision plan of length N with minimum LET is the optimal policy for the decision horizon N, because the utility function shape conserves the minimum property.

We have a team of agents that make decisions using its knowledge about the world $b_\tau^{k|k}$ and making observations. For a piecewise actions optimization, the decision plan of horizon N is $v_{1:q}^k = \{u_{1:q}^k, \cdots, u_{1:q}^{N-1}\}$. By definition, the probability of finding the target

[5]Other expectation functions for the optimal search, do not have that property and they need that the density function sums 1. For instance the following expressions, extracted from [Trummel and Weisinger, 1986; Bourgault et al., 2003], cannot compute LET:

$$E\{T\} = \sum_{k=1}^{\infty} k\,(P(T \leq k) - P(T \leq k-1)) = \sum_{k=1}^{\infty} kP(T = k)$$

.

before instant $k+j$, $P(k \leq T \leq k+j)$, is equivalent to the probability of detecting the target in the time interval $(k+1, k+j)$, that is, the probability of the union of the detection events:

$$P(k \leq T \leq k+j) = P(\bigcup_{l=k}^{k+j} \bigcup_{i=1}^{q} D_i^{k+l} | s_{1:q}^{1:k}, u_{1:q}^{k:k+j-1}, z_{1:q}^{1:k}) \tag{3.52}$$

Therefore, we compute the LET for a given decision plan $v_{1:q}^k$ as the mean time $\mu(v_{1:q}^k)$:

$$\mu(s_{1:q}^k, v_{1:q}^k) = \sum_{j=1}^{N} \left(1 - P(\bigcup_{l=k}^{k+j} \bigcup_{i=1}^{q} D_i^{k+l} | s_{1:q}^{1:k}, u_{1:q}^{k:k+j-1}, z_{1:q}^{1:k})\right) \tag{3.53}$$

Assuming deterministic actions we have that $\{s_{1:q}^k, u_{1:q}^{k+1:k+N}\}$ is equivalent to $\{s_{1:q}^{k+1:k+N}\}$. Also, instead of using the detection even D_i^k we can apply the complement to use the non-detection event $\overline{D}_i^k$ obtaining:

$$\mu(s_{1:q}^k, v^k) = \sum_{j=1}^{N} P(\overline{D}_{1:q}^{k+1:k+j} | s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) \tag{3.54}$$

Assuming non-detection inside the horizon plan we can compute recursively the probability of non-detecting the target substituting in Eq. 3.41 N by j. Therefore, the non-detection probability becomes:

$$P(\overline{D}_{1:q}^{k+1:k+j} | s^{1:k+N}, z_{1:q}^{1:k}) = \int_{\tau^{k+j}} \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s_i^{k+j}, \tau^{k+j}) \tilde{b}_\tau^{k+j|k+j-1} d\tau^{k+j} \tag{3.55}$$

Note that by unrolling the equation we arrive to a recursive prediction and assimilation of a Bayesian filter without normalization factor (Section 3.4), where we compute $\tilde{b}_\tau^{k+j|k+j-1}$, yielding to the final utility function:

$$LET = \mu(s_{1:q}^k, v_{1:q}^k, b_\tau^{k|k}) = \sum_{j=1}^{N} \int_{\tau^{k+j}} \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s_i^{k+j}, \tau^{k+j}) \tilde{b}_\tau^{k+j|k+j-1} d\tau^{k+j} \tag{3.56}$$

To compute $\tilde{b}_\tau^{k+j|k+j-1}$ we use the following equations (see Section 3.4 for the full derivation):

$$\tilde{b}_\tau^{k+j|k+j-1} = \int_{\tau^{k+j-1}} P(\tau^{k+j} | \tau^{k+j-1}) \tilde{b}_\tau^{k+j-1|k+j-1} d\tau^{k+j-1} \tag{3.57}$$

$$\tilde{b}_\tau^{k+j|k+j} = \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s_i^{k+j}, \tau^{k+j}) \tilde{b}_\tau^{k+j|k+j-1} \tag{3.58}$$

where the prior target location belief is used at the first step of the recursion: $\tilde{b}_{\tau}^{k|k} = b_{\tau}^{k|k}$.

The solution (i.e. the team actions $v_{1:q}^{k*}$ to the MTS using the LET strategy given the agents initial state $s_{1:q}^{k}$ and the prior target location distribution $b_{\tau}^{k|k}$ is the following equation:

$$v_{1:q}^{k*} = \arg\min_{v^k} \mu(s_{1:q}^{k}, v_{1:q}^{k}, b_{\tau}^{k|k}) \tag{3.59}$$

Using Eq. 3.56, we build the time decision model for the LET where the MTS solution is the minimization of the rewards.

The immediate reward system, R_I, is what the agent does not observe at instant k.

$$R_I = \int_{\tau^k} \prod_{i=1}^{q} P(\overline{D}_i^k | s_i^k, \tau^k) \tilde{b}_{\tau}^{k|k-1} d\tau^k \tag{3.60}$$

The short term reward system, R_N, is what the agent obtains in a time window of N instants. It is instantiated as a predictive model receding horizon controller with utility function:

$$R_N = \sum_{j=1}^{N} \int_{\tau^{k+j}} \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s_i^{k+j}, \tau^{k+j}) \tilde{b}_{\tau}^{k+j|k+j-1} d\tau^{k+j} \tag{3.61}$$

Finally, the future reward system R_H is the expectation of the future value of detection.

$$R_H = E\left\{ \sum_{j=k+N+1}^{\infty} \int_{\tau^j} P(\overline{D}_{1:q}^{j} | s_{1:q}^{j}, \tau^j) b_{\tau}^{j|j-1} d\tau^j \right\} \tag{3.62}$$

The computation of this expectation is as complex as the MTS itself, and some authors like [Mathews, 2008] set R_H to zero assuming that the system is unstable. If R_H is obviated this strategy becomes a receding horizon solution with no terminal cost. This approximation suffers from myopicity for long term decisions, but we will see in Section 3.5.2.3 how to reduce this drawback by including a heuristic. Other interesting approaches can be found in [Yang et al., 2002; Bertsekas, 1995].

3.5.2.2 Maximum Discounted Time Reward

Apart from minimizing the expected time, we want also to include the possibility of modeling the time. We know by Proposition 3.1 that maximizing the joint probability of detection does not necessary optimize the time. Moreover, we want to value more visiting regions with high probability of finding the target earlier. Thus, we can weight the rewards with a discounted time function as used in Bellman equation [Bellman, 1957] or

in [Blum et al., 2003]. Using this approach, the computation of the optimal action plan becomes the Discounted Time Reward (DTR) problem. The discounted time function can be any decreasing function that models the time importance and the reward is the joint probability of detection. Figure 3.9 shows how the target probability of detection

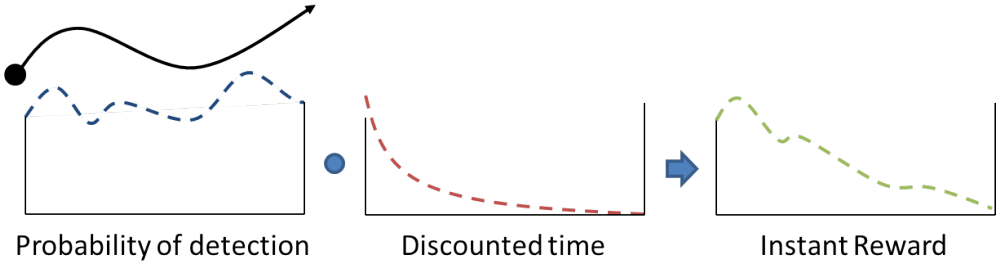


FIGURE 3.9: DTR strategy.

for an action plan (represented by the black dot and the curved black arrow) is modified by the discounted time function. On the left we can see the probability of detection computed along the agent trajectory. The discounted time function, on the middle image, weights the probability obtaining the right image where the rewards decrease along with the time. The accumulated discounted instant reward is the objective function that should be maximized.

Defining the discounted time function $f(k)$ as a discount parameter that decreases exponentially with the time k:

$$f(k) = \lambda^k \mid 0 \leq \lambda \leq 1 \tag{3.63}$$

And using the joint probability of detection (Eq. 3.29). The multiagent DTR strategy is formalized as:

$$DTR = J_{dtr}(s^k_{1:q}, v^k_{1:q}, b^{k|k}_\tau) = \sum_{j=1}^{N} \lambda^{j-1} \int_{\tau^{k+j}} \left[1 - P(\overline{D}^{k+j}_{1:q} | s^{k+j}_{1:q}, \tau^{k+j})\right] \tilde{b}^{k+j|k+j-1}_\tau d\tau^{k+j} = \tag{3.64}$$

$$= \sum_{j=1}^{N} \lambda^{j-1} \int_{\tau^{k+j}} \left[1 - \prod_{i=1}^{q} P(\overline{D}^{k+j}_{i} | s^{k+j}_{1:q}, \tau^{k+j})\right] \tilde{b}^{k+j|k+j-1}_\tau d\tau^{k+j} \tag{3.65}$$

Maximizing J_{dtr} is not an uniformly optimal plan, but it guarantees an optimal plan constrained by the discounted time function for an horizon N. Note that in this strategy we have to maximize DTR (i.e. the probability of detection discounted by $f(k)$), while in LET (Eq. 3.77) we minimize the sum of the accumulated non-detection at each instant.

We design the time decision model for the DTR, whose solution is the maximization of the rewards.

The immediate reward system, R_I, is what the agent accumulates making next observation:

$$R_I = \int_{\tau^{k+1}} \left[1 - \prod_{i=1}^{q} P(\overline{D}_i^{k+1} | s_{1:q}^{k+1}, \tau^{k+1}) \right] b_\tau^{k+1|k} d\tau^k \tag{3.66}$$

The short term reward system, R_N, or the model predictive receding horizon utility function [Mathews, 2008], is calculated as follows,

$$R_N = \sum_{j=1}^{N} \lambda^{j-1} \int_{\tau^{k+j}} \left[1 - P(\overline{D}_{1:q}^{k+j} | \tau^{k+j}, s^{k+j}) \right] b_\tau^{k+j|k+j-1} d\tau^{k+j} \tag{3.67}$$

It is interesting that if we use $\lambda = 1$ the R_N becomes the detection function (Eq. 3.29). Therefore, the tuning parameter λ permits us to decide indirectly how fast we want to find the target or in other words, the importance of the actions that the agent will take in the future.

Finally, the future reward system R_H is the expectation of the future value of detection.

$$R_H = E \left\{ \sum_{j=k+N+1}^{\infty} \lambda^{j-1} \int_{\tau^j} \left[1 - P(\overline{D}_{1:q}^{j} | \tau^j, s^j) \right] b_\tau^{j|j-1} d\tau^j \right\} \tag{3.68}$$

The expectation is, as in the LET strategy too complex. We can set it to zero assuming fast information value loss. However, it can be approximated following the strategy presented in next section.

3.5.2.3 Discounted Time Heuristic

This strategy focuses on approximating the future expectation of the objective function R_H by using a heuristic. This approach can be used in combination with the previous strategies to reduce the myopicity. The idea is to compute R_N with any probabilistic function and include a time dependent heuristic as the terminal cost. Within the MTS, R_H is the expected observation (or the expected non-detection observation).

Given any actions set $v_{1:q}^k$, the sensor state transits from $s_{1:q}^k$ to $s_{1:q}^{k+N}$ through its motion model, providing the information of the short term reward. We incorporate the future rewards into the optimization method by computing the expected observation at the terminal state s_i^{k+N}. Thus, we need a function that evaluates the reward of choosing $v_{1:q}^k$ and a function that estimates the goodness of being in $s_{1:q}^{k+N}$. The estimation relies on inferring how much detection the platform can get from a state. The heuristic

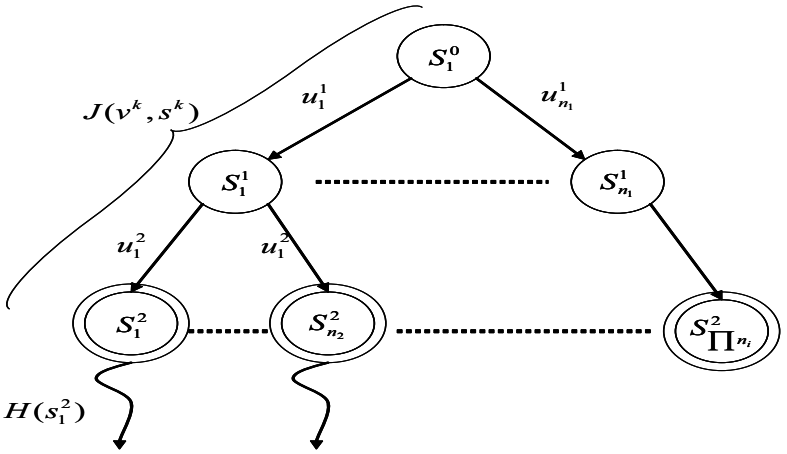


FIGURE 3.10: Decision tree with terminal nodes with heuristic associated. The utility function $J(v^k)$ is extended by a cost-to-go/future-rewards-estimation $H(s^{k+N})$.

proposed, approximates the future observation assuming that the target is static after instant $k+N$.

First of all, we model the expected observation with the classical approach [Flint et al., 2002; Bellingham et al., 2002], that incorporates the heuristic by building the utility function as the addition of the short term reward and the expected reward. This method is intractable due to the intense computation of the expected reward. Thus, afterwards we propose a method that includes the expected observation without increasing the computational time, by modeling the heuristic as a sensor associated with the final state of the sensing platform.

Expected Reward The expected reward or expected observation is defined as the reward that the sensor will expect to obtain in the future for being at state $s_{1:q}^{k+N}$ with target belief $\tilde{b}_\tau^{k+N}$. Accurately representing this expected reward is as hard as solving the original general problem itself. Hence, we use a heuristic to approximate this expected reward. The classical approach is to develop a heuristic $H(s_{1:q}^k, v_{1:q}^k, b_\tau^k)$ that evaluates the sensor state and add it to the short reward utility function, as described in Figure 3.10.

In order to build the heuristic we can approximate the forward reachable set space of the sensor platform into some regular shapes [Yang, 2005]. Figure 3.11 shows three different approximated shapes of these regions. In practice, the triangle shape is less effective for this problem compared to the others as it has a very limited field of view and the reliable min-max turn shape is too complex for real time computation and not worthy in terms of search performance improvement. Hence we choose the circular shape due to its good approximation to the real sensor dynamics and its more convenient mathematical description.

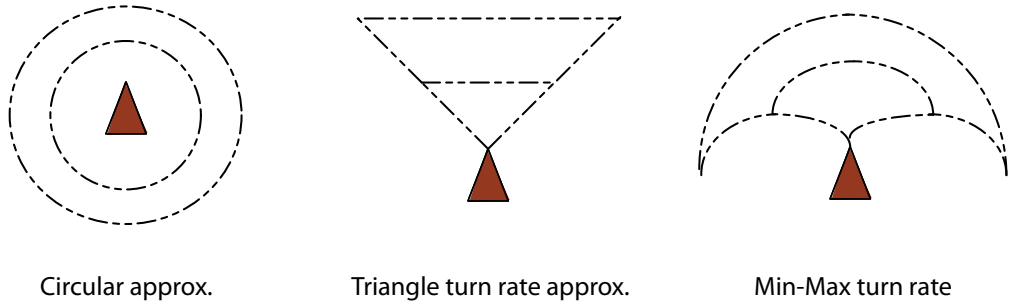


FIGURE 3.11: Possible region shapes for the heuristic.

We now define the probability of the non-detection event with one step ahead from states $s_{1:q}^{k+N}$ and belief b_τ^{k+N}, conditioned on a possible future sensing states $s_{1:q}^{k+N+1}$:

$$P(\overline{D}_{1:q}^{k:k+N+1}|s_{1:q}^{k+N+1}, z_{1:q}^{1:k}, s_{1:q}^{1:k+N}) = \int_\tau P(\overline{D}_{1:q}^{k+N+1}|s_{1:q}^{k+N+1}, \tau)\tilde{b}_\tau^{k+N} d\tau \tag{3.69}$$

where $s_{1:q}^{k+N+1}$ could be any possible state of each agent sensor (i.e any location in Ω). We emphasize again that $z_{1:q}^{1:k+N}$ and $s_{1:q}^{1:k+N}$ are known and included in $\tilde{b}_\tau^{k+N}$. We can compute the expected value of non-detection in $k+N+1$ by integrating over the possible states $s_{1:q}^{k+N+1}$, obtaining the expected reward formulation:

$$\begin{aligned} &H(s_{1:q}^k, v_{1:q}^k, b_\tau^k) = P(\overline{D}_{1:q}^{k:k+N+1}|s_{1:q}^{k:k+N}, z_{1:q}^{1:k}) = \\ &= \int_{\tau^{k+N+1}} \int_{s_1^{k+N+1}} \cdots \int_{s_q^{k+N+1}} \prod_{i=1}^{q} P(\overline{D}_i^{k+N+1}|s_{1:q}^{k+N+1}, \tau^{k+N+1}) P(s_i^{k+N+1}|s_i^{k+N}) \tilde{b}_\tau^{k+N} ds_{1:q}^{k+N+1} d\tau^{k+N+1} \end{aligned} \tag{3.70}$$

In here, $P(s_i^{k+N+1}|s_i^{k+N})$ is the probability of the state transition from s_i^{k+N} to s_i^{k+N+1}. Eq. 3.70 is effectively weighting the probability of non-detection event by the probability of reaching that sensing state. The modeling of $P(s_i^{k+N+1}|s_i^{k+N})$ depends on the shape of the forward reachable set that we have described earlier. For a simple circular region, a discounted time function can be used:

$$P(s_i^{k+N+1}|s_i^k) = \frac{1}{\eta}\beta^{(\|s_i^{k+N} - s_i^{k+N+1}\|/V_i)} \tag{3.71}$$

where $\|s_i^{k+N} - s_i^{k+N+1}\|$ is the Euclidean distance between s_i^{k+N} and s_i^{k+N+1}, η is a probability normalizer, β is the discounted time factor ($0 \leq \beta \leq 1$) and V_i is the agent velocity.

The extended utility function from short term reward with expected reward is thus:

$$J_{DTH}(s^k_{1:q}, v^k_{1:q}, b^k_\tau) = R_N + R_H = J(s^k_{1:q}, v^k_{1:q}, b^k_\tau) + H(s^k_{1:q}, v^k_{1:q}, b^k_\tau) \tag{3.72}$$

This utility function is computationally intense because the expectation operator in Eq. 3.70 has to go through all possible sensing states $s^{k+N+1:\infty}_{1:q}$. Therefore, instead of using this approach, we propose a heuristic that captures the effectiveness of the expected reward by using an infinite range sensor model at state s^{k+N}_i and combine it within the framework of short term reward formulation.

Infinite Range Sensor Heuristic Avoiding the classical expectation heuristic, we can model an infinite range sensor model that reaches the whole world (target belief), as the expected observation. This approach solves the intractability of computing the expectation. We design the expected observation heuristic $\hat{H}(s^k_i)$ as a sensor and we use it at the terminal state s^{k+N}_i. This heuristic sensor is restricted to problems where the platforms have sensors based on range. To apply it to other type of sensing platforms, a different sensor heuristic should be designed, but the method remains the same.

The infinite range sensor model $\hat{H}(s^{k+N}_i)$ is designed to include the time dependency by using a discounted function:

$$\hat{H}(s^{k+N}_i) = \eta\beta^{\|\tau - s^{k+N}_i\|/V_i} \tag{3.73}$$

where $\|\tau - s^{k+N}_i\|$ is the Euclidean distance from the sensor to the target location, V_i is the agent velocity and β controls the importance of the probability of finding the target at further regions from state s^{k+N}_i. When the parameter η makes $\sum_{s^{k+N}_i} \hat{H}(s^{k+N}_i) = 1$ the heuristic is normalized. However, we can build unnormalized heuristics making $\eta = 1$ in order to increment the contribution of the future rewards.

The complementary version of the heuristic, that is the expected non-observation is:

$$\overline{\hat{H}}(s^{k+N}_i) = 1 - \eta\beta^{\|\tau - s^{k+N}_i\|/V_i} \tag{3.74}$$

In Figure 3.12 we can see the shape of Eq. 3.74, where the value is 1 when we expect no observation and approaches to 0 as we get close to the agent location s^{k+N}_i.

DTH Decision Model The sensing platform uses the sensor model for obtaining the sensor observations at states $s^k_{1:q}, \cdots, s^{k+N}_{1:q}$ and Eq. 3.73 for expected observation from the last state $s^{k+N}_{1:q}$. Implicitly it uses the circular region approximation explained at Figure 3.11.

Because we are modeling the heuristic as a sensor, the contribution to the multiagent utility function is:

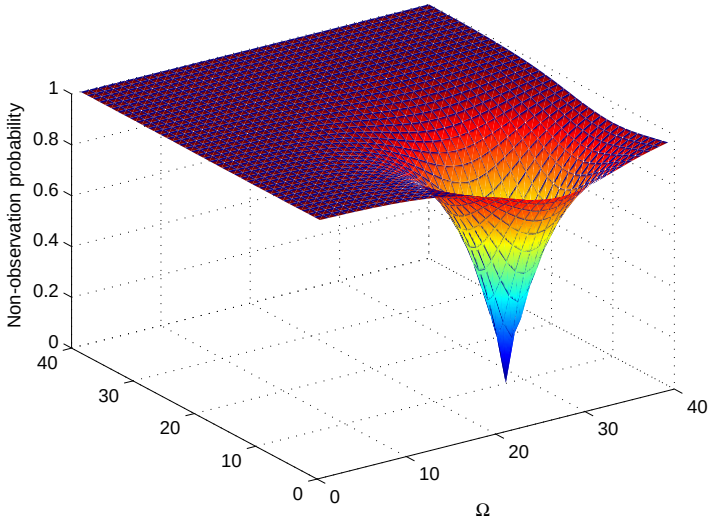


FIGURE 3.12: Infinite range sensor model $\overline{\hat{H}}(s^{k+N})$ shape when $\beta = 0.8$, $V = 0.5$ and $s^{k+N} = [28, 10]^T$.

$$R_H = \int_{\tau} \prod_{i=1}^{q} \overline{\hat{H}}(s_i^{k+N}) \tilde{b}_{\tau}^{k+N} d\tau \tag{3.75}$$

Now we can design the final strategy, but in this case, depending on the short term utility function R_N, the heuristic is included differently. For instance, combined with the joint non-detection probability (Eq. 3.41) we have:

$$J_{NDH}(s_{1:q}^k, v_{1:q}^k, b_{\tau}^k) = \int_{\tau^{k+N}} \prod_{i=1}^{q} \overline{\hat{H}}(s_i^{k+N}) \tilde{b}_{\tau}^{k+N|k+N} d\tau^{k+N} \tag{3.76}$$

Note that the heuristic behaves like a sensor inside the function.

Analogously, using LET (Eq. 3.61) as the short term utility function:

$$\begin{aligned} LETH(s_{1:q}^k, v_{1:q}^k, b_{\tau}^k) = \sum_{j=1}^{N} \int_{\tau^{k+j}} \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s_i^{k+j}, \tau^{k+j}) \tilde{b}_{\tau}^{k+j|k+j-1} d\tau^{k+j} + \\ + \int_{\tau^{k+N}} \prod_{i=1}^{q} \overline{\hat{H}}(s_i^{k+N}) \tilde{b}_{\tau}^{k+N|k+N} d\tau^{k+N} \end{aligned} \tag{3.77}$$

Finally using DTR (Eq. 3.67) as the short term reward the strategy becomes:

$$DTRH(s^k_{1:q}, v^k_{1:q}, b^k_\tau) = \sum_{j=1}^{N} \lambda^{j-1} \int_{\tau^{k+j}} \left[1 - \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | \tau^{k+j}, s_i^{k+j})\right] b_\tau^{k+j|k+j-1} d\tau^{k+j} + \\ + \lambda^N \int_{\tau^{k+N}} \left[1 - \prod_{i=1}^{q} \overline{\hat{H}}(s_i^{k+N})\right] \tilde{b}_\tau^{k+N|k+N} d\tau^{k+N} \tag{3.78}$$

By including this simple expected observation heuristic, we reduce the myopicity of any local strategy without the addition of appreciable computation time.

Other Heuristics Another possible heuristic is the average discounted probability of detection [Yang et al., 2002]. In this section we only comment how to compute it for a single agent because this heuristic increases the computational complexity a lot, and therefore, in this thesis we focus on the infinite range heuristic previously explained.

We start from an agent state s^{k+N} and we estimate the possible observations that are related with the probability of being in a state s' at instant $j > k+N$ and the probability of detecting the target in s'. An important aspect of this heuristic is that it overestimates the utility value.

Defining the region $\Gamma(j, s^{k+N})$ as the one that can be visited at time k from state s^{k+N} and β as the time horizon, the discounted time heuristic can be designed as follows [Yang et al., 2002]:

$$\hat{H}(s^{k+N}) = \frac{1}{\beta} \sum_{j=1}^{\beta} \frac{\beta^j}{Area(\Gamma(j, s^{k+N})} \sum_{s' \in \Gamma(j, s^{k+N})} \int_\tau P(D^{k+N+j} | \tau, s') b_\tau^{k+N} d\tau \tag{3.79}$$

3.5.2.4 Other Useful Strategies

Here we describe other two useful strategies when the target location is described by a Gaussian distribution. The first strategy, Maximum Slope (MS), exploits the geometrical structure of the Gaussian assuming that the agents will not have sensor measurements, which is the worst possible case. The second approach, Minimum-Entropy (ME), is based on assuming that our estimate of the target is correct and the agents will have measurements with covariance noise dependent on the distance.

We define the target location belief as the Gaussian $b_\tau^k \sim N(\hat{\tau}^k, \Sigma^k)$. b_τ^k represents the last updated target location distribution given by the information layer. We also refer to $\tilde{b}_\tau^k \sim N(\tilde{\tau}^k, \tilde{\Sigma}^k)$ as the target location predicted by the controller when applying the control actions $u^k_{1:q}$. Because we are using a Gaussian representation of the target variable, and modeling its movement and the sensors with non-linear Gaussian probabilities, we can substitute the RBE by an extended Kalman filter.

Maximum Slope The first strategy is based on the optimal density solution for a Gaussian distribution for one agent [Stone, 1975], consistent on reaching the local maxima of the probability density function and then describing circles incrementing the radius. Due to the gradient shape of the Gaussian, this is conceptually equivalent to accumulate the maximum probability using a single agent. The agent dynamic constraints introduce the restrictions to the optimization. We have to model which measurements the agent will obtain and therefore in this strategy we adopt the worst case situation, that is, the agents do not receive any new measurements. Note that in a Kalman filter framework, this means that there is no assimilation step, but we still have to predict the target location for each future instant.

The utility function J_{MS} for this strategy (Eq. 3.80) is the sum of the values of the Gaussian distribution along the agent states, because due to the distribution structure we drive the agent through the maximum incremental slope. Note that this utility function is independent for each agent because there are no observations.

$$J_{MS}(s_i^k, v_i^k, b_\tau^k) = \sum_{j=1}^{N} e^{-\frac{1}{2}(s_i^{k+j} - \tilde{\tau}^{k+j})^T (\tilde{\Sigma}^{k+j})^{-1} (s_i^{k+j} - \tilde{\tau}^{k+j})} \tag{3.80}$$

This function evaluates the agent state with the value of the Gaussian distribution of the target location. In order to compute $\tilde{\tau}^{k+j}$ and $\tilde{\Sigma}^{k+j}$ we use an Extended Kalman Filter (EKF). This strategy will be explained extensively in Section 5.4.4.1 of Chapter 5.

This strategy, which does not use communication among the agents during the planning process, has one drawback: all agents pursuit the same peak yielding to less optimal search when the agents are close to each other. In the case of a single agent this suboptimal behavior does not exist.

Minimum Entropy Instead of driving the agents towards the mean of the Gaussian, this approach minimizes the uncertainty that the team has about the target location, represented as the entropy at the last estimated states of the agents:

$$J_{ME}(s_{1:q}^k, v_{1:q}^k, b_\tau^k) = \frac{1}{2} \ln(2\pi e)^2 |\tilde{\Sigma}^{k+N}| \tag{3.81}$$

The computation of $\tilde{\Sigma}^{k+N}$ is done using a modified version of the EKF. For a detailed explanation please refer to Section 5.4.4.2 of the Chapter 5. However, it is worth noting that this strategy does not necessary optimize the searching time.

TABLE 3.2: Strategies summary

Strategy	Description	Equation	opt
LET	Average expected time.	3.61	min
DTR	Discounted time probability of detection.	3.67	max
DTH	Future expectation approximation modeled as a heuristic sensor. It can be combined with rest.	3.76	min/max
LETH	Non-myopic variant of LET by combining with DTH.	3.77	min
DTRH	Non-myopic variant of DTR by combining with DTH.	3.78	max
MS	Select the maximum slope.	3.80	max
ME	Minimize the entropy at instant $k+N$.	3.81	min
D	Joint detection events.	3.29	max
ND	Joint Non-Detection events.	3.41	min
NDH	Non-myopic variant of ND by combining with DTH.	3.76	min

3.5.2.5 Summary of strategies

Table 3.2 summarizes all strategies presented in this section, by assigning each one to its formula and collecting the most representative characteristics.

3.6 Algorithms

Once we have the MTS strategies, we need algorithms that compute the optimal actions of the team. Given the strategy J, the intial states of the agents $s^k_{1:q}$ and a prior target location distribution $b^{k|k}_\tau$ provided by the data fusion layer, these algorithms have to compute the action plan $v^{k*}_{1:q}$, that optimizes (**opt** $= max/min$) the strategy function.

Therefore we look for algorithms that compute:

$$v_{1:q}^{k*} = \arg \underset{v_{1:q}^k}{\mathbf{opt}} J(s_{1:q}^k, v_{1:q}^k, b_\tau^{k|k}) \tag{3.82}$$

Depending on the belief and the actions representation, i.e. discrete or continuous, the optimization space changes and therefore we need two types of algorithms.

In this section we first present some simple algorithms such as the greedy search [Bourgault et al., 2003] and the classical Limited Depth First Search [Russell et al., 1996] applied to the MTS. Afterwards we describe the constraint programming approach that frames the first algorithm proposed in Chapter 4 and the iterative stochastic algorithms that we use to solve the MTS in discrete form in Chapter 4 too. Finally we discuss the gradient-based general method to solve the MTS in continuous form used in Chapter 5.

3.6.1 Naive Algorithms

In this section we analyze the MTS from the classical artificial intelligence point of view. We approach the problem as a complete state search discussing its drawbacks. The two algorithms presented in this section use a discrete representation of the actions.

First of all we describe the 1-step greedy algorithm already proposed by [Bourgault et al., 2003] because when we only use one step team actions, optimizing LET and DTR is equivalent to maximize the probability of detection. We define the possible actions from state s_i^k as $\alpha(s_i^k) \in U$ where $U = \{a_1, \ldots, a_l\}$. For the multiagent case we have the set of states $s_{1:q}^k$ and the possible actions become the combination of all actions: $\alpha(s_{1:q}^k) = \bigcup \alpha(s_i^k)$. The set of forward states $s_{1:q}^{k+1}$ are obtained deterministically applying the set of actions $u_{1:q}^k$: $s_{1:q}^{k+1} = \beta(s_{1:1}^k, u_{1:q}^k)$. We also define the last updated location belief of the target $b_\tau^{k|k}$ as the prior belief.

The MTS greedy algorithm (Algorithm 4) selects the agents actions that provides the maximum instant reward (i.e. the maximum probability of detection[6]). The information of the target location is updated using the RBE (Algorithm 1). It is important to highlight that the algorithm is defined to maximize the probability of detection.

Algorithm 4 is totally myopic, but for target locations that follow a Gaussian distribution is a good approach.

The second algorithm in this section is an improved version of the greedy one, where we compute N-steps team decisions. It performs a complete search up to a limit level, where the final nodes are substituted by a terminal cost (i.e. the final nodes are evaluated using an heuristic). The algorithm is called Limited First Depth Search (LDFS) [Russell et al., 1996] and the most simple iterative version is presented in Algorithm 5. It uses a queue

[6]Maximizing the probability of detection is the same as minimizing the probability of non-detection.

Algorithm 4 MTS greedy algorithm

Require: $b_\tau^{k|k}$ ▷ Prior information
Require: $s_{1:q}^k$ ▷ Initial agents state
$\tilde{b}_\tau \leftarrow b_\tau^{k|k}$
while Not detected target **do**
  $list = \alpha(s_{1:q}^k)$ ▷ list stores all possible agent actions from state $s_{1:q}^k$
  $reward \leftarrow 0$
  $\tilde{b}_\tau \leftarrow RBEprediction(P(\tau^{k+1}|\tau^k), \tilde{b}_\tau)$ ▷ Alg. 2
  for $i = 1, |list|$ **do**
    $u_{1:q}^k \leftarrow list(i)$
    $s_{1:q}^{k+1} \leftarrow \beta(s_{1:q}^k, u_{1:q}^k)$
    $v \leftarrow \int_{\tau^{k+1}} \left[1 - \prod_{j=1}^q P(\overline{D}_j^{k+1}|\tau^{k+1}, s_j^{k+1})\right] \tilde{b}_\tau d\tau^{k+1}$
    if $v > reward$ **then**
      $reward \leftarrow v$
      $u_{1:q}^{k(*)} \leftarrow u_{1:q}^k$
      $s_{1:q}^{k+1(*)} \leftarrow s_{1:q}^{k+1}$
    end if
  end for
  $\tilde{b}_\tau \leftarrow RBEupdate(s_{1:q}^{k+1}, \tilde{b}_\tau)$ ▷ Alg. 3
end while

to store the feasible solution nodes, which can be operated by the following functions: *head* returns the last inserted node, *pop* deletes the last inserted node and *push* inserts nodes into the queue.

We describe the strategy with the function $J(s_{1:q}^k, v_{1:q}^k, b_\tau^{k|k})$ but, because $\beta(s_{1:q}^k, u_{1:q}^k)$ is a deterministic function, we assume that the actions are implicit and we omit them for clarity. Therefore the utility function for an horizon N is $J(s_{1:q}^{k+1:k+N}, b_\tau^{k|k})$. Besides, the heuristic function is described by $\hat{H}(s_{1:q}^N, \tilde{b}_{k+N|k+N})$. The nodes in Algorithm 5 represent a partial team trajectory, and the function *generate_adjnodes* generates the new locations of the agents by applying all possible actions to the last agents state. The algorithm maximizes the utility function.

The branching factor is exponential: for l number of actions for each state, at each level j the number of opened nodes is l^j and the total number of open nodes is $\sum_j l^j$ (e.g. for an horizon $N = 10$, and actions $l = 8$, the number of nodes that should be opened is $\approx 1.2 \cdot 10^9$). Therefore, this algorithm is only tractable for a small horizon problem (e.g. $N = 3$).

We can improve Algorithm 5 by computing the partial rewards while opening the nodes, and designing bounds yielding to a branch and bound method as [Eagle and Yee, 1990; Dell et al., 1996]. Also, we can specify the horizon limit to $N \log N$, select the best solution and continue from there the optimization. For instance [Sarmiento et al., 2009] uses this approximation. Therefore, domination solution test or bounds and good heuristics are needed for this algorithm to achieve good performance.

Algorithm 5 Limited depth-first heuristic search algorithm

Require: $b_\tau^{k|k}$ ▷ Prior information
Require: $s_{1:q}^k$ ▷ Initial agents state
Require: N ▷ Horizon, Limited Depth
$node = \left[s_{1:q}^k\right]$
$queue \leftarrow [node]$
$reward \leftarrow 0$
$j \leftarrow 1$ ▷ Nodes level index (instant)
while Not empty *queue* **do**
  $s_{1:q}^{k:k+j} \leftarrow head(queue)$
  $pop(queue)$
  $adjnodes \leftarrow generate\_adjnodes(s_{1:q}^{k:k+j})$ ▷ adjnodes is a list of nodes: $s_{1:q}^{k+j+1}$
  if $empty(adjnodes) || j \geq N$ **then**
    $h \leftarrow heuristic(s_{1:q}^{k+j}, \tilde{b}_\tau^{k+N|k+N})$
    $g \leftarrow J(s_{1:q}^{k+1:k+j}, b_\tau^{k|k})$
    $r \leftarrow h + g$
    if $r > reward$ **then**
      $reward \leftarrow r$
      $s_{1:q}^{k:k+j(*)} \leftarrow s_{1:q}^{k:k+j}$
    end if
  else
    $push(queue, adjnodes)$
  end if
end while

3.6.2 Constraint Programming

Constraint Programming (CP) is useful to find global solutions for hard problems when the restrictions provide implicit information about the problem. The way to proceed is to model the problem and then leave the optimization to the solver, which uses algorithms such as simplex or depth first search. The solvers are usually complete, thus the solution provided is optimal for the defined problem.

The general continuous MTS problem is very simple, the optimization of an utility function where the actions are constrained by a minimum and a maximum value:

$$\begin{aligned} &\underset{u_{1:q}^{k:k+N-1}}{\textbf{opt}} \quad J(s_{1:q}^k, u_{1:q}^{k:k+N-1}, b_\tau^k) \\ &\text{s.t.} \\ &\quad s_{1:q}^k \text{ fixed} \\ &\quad u_{min} \leq u_{1:q}^{k:k+N-1} \leq u_{max} \end{aligned} \tag{3.83}$$

We cannot see a lot of restrictions in Eq. 3.83 that will make the solver to cut a lot of solutions, but if instead of the actions $u_{1:q}^{k:k+1}$ we use the agents states $s_{1:q}^{k+1:k+N}$ we have:

$$\begin{aligned} &\underset{s_{1:q}^{k+1:k+N}}{\textbf{opt}} \; J(s_{1:q}^{k+1:k+N}, b_\tau^k) \\ &\text{s.t.} \\ &\quad s_{1:q}^k \text{ fixed} \\ &\quad s_i^k \text{ and } s_i^{k+1} \text{ are feasible (connected)} \end{aligned} \tag{3.84}$$

In this form, where the MTS model is approached as a discrete optimization, we have a lot of constraints in the path that will help the optimization computation. For instance, this approach can be used in the mixed integer linear programming (MILP) or non linear programming paradigm (NLP) [Amato et al., 2010], by using an state binary selector $x_{j,j',i,k}$, that takes value 1 if agent i, moves from cell j at time $k-1$ to cell j' at time k, and 0 otherwise [Dell et al., 1996]. In finite domains we can use a function that associates the value to each state depending on the instant that is visited.

Instead of using MILP or NLP, in this thesis we propose a CP for finite domains model in Section 4.3 of Chapter 4 to solve the discrete MTS for static targets where we exploit the structure of the constraints within the problem. The MTS for this CP method is modeled under the perspective of finite domains.

3.6.3 Estimation Of Distribution Algorithms

We are looking for tractable algorithms that permit searching for a dynamic target and exploit the implicit information of the problem. From the discrete actions point of view, a good approach is to use stochastic sampling algorithms, that learn the implicit distribution of the solution samples and then use it for optimization purposes. These methods are called Estimation of Distribution Algorithms (EDAs) [Larrañaga, 2002].

Instead of finding the best actions, within theses methods, we look for the distribution of the actions that optimizes the utility function. At each instant k the agent i has a probability of making an action $u_i^k \in U$. Then, the problem becomes the estimation of the action distribution p that maximizes the MTS strategy. Afterwards, when we have the optimal distribution p^*, we can extract the solution as:

$$v_{1:q}^{k*} = \arg \textbf{opt} p* \tag{3.85}$$

Assuming that we have the same number of actions l for each agent state and fixing the horizon N, the actions distribution p^* that we want to learn is a matrix of size $l \times N \times q$. Each element p_{aji} is defined as the probability of doing action a at instant j for agent i. We can learn the optimal distribution iteratively with an EDA.

An advantage of using these iterative algorithms is that we are improving the solution at each iteration. Thus, they can be used online, by trading off the optimality versus the computation time. Besides, they are suitable for any type of sensor models, even the non-differentiable ones.

Algorithm 6 describes the general EDA for the MTS. The superscript p^t is used to indicate the iteration step.

Algorithm 6 EDA algorithm

Require: $b_\tau^{k|k}$ ▷ Prior information
Require: $s_{1:q}^k$ ▷ Initial agents state
1: $t \leftarrow 0$
2: $p^t \leftarrow$ initialize the matrix action distribution
3: **while** Not terminated **do**
4: $X \leftarrow$ Generate Solutions by sampling p^t: $X_i \sim p^t$
5: $J \leftarrow$ Evaluate each X_i with the MTS strategy $J(s_{1:q}^k, v_{1:q}^k, b_\tau^{k|k})$ where $v_{1:q}^k = X_i$
6: $X' \leftarrow$ Select the best solutions of X (better J)
7: $p^{t+1} \leftarrow$ fit the new distribution of actions to the selected samples X'
8: $t \leftarrow t + 1$
9: **end while**
10: $\{p^* = p^t\}$
11: $u_{1:q}^{k:k+N-1} \leftarrow \arg\max p^*$

Assuming that the variables p_{aji} are independent, we can use the Cross Entropy Optimization method (CEO) [Rubinstein and Kroese, 2004] as an EDA[7]. This algorithm is widely explained in Section 4.4 of Chapter 4.

3.6.4 Gradient-based Algorithms

From a continuous action point of view, the piece wise optimization method is just an approximation [Bourgault et al., 2004] where each control parameter is maintained over a constant interval of time Δk. Once we have split each trajectory into N control actions, we can use a general optimization method [Furukawa, 2002]. If the sensor model and the strategy function is continuous, we can apply a gradient-based optimization algorithm [Mathews et al., 2007; Gan and Sukkarieh, 2010]. An advantage of the gradient methods for multiagents systems, is that they can be decentralized [Bertsekas, 1995]. Besides, recently the convergence in asynchronous form has been demonstrated, just using an adaptive step parameter that takes into account the coupling between the measurements of the different agents [Mathews et al., 2007].

The general centralized gradient-based method for the MTS is presented in Algorithm 7. The superscript $v_{1:q}^t$ indicates the iteration step. We assume that we have the dynamic

[7] Although CEO algorithm comes from other research line, which assumes that the optimal solution have a small probability to be sampled, in practice, CEO can be considered an EDA.

model of the agent as a set of differential equations $\dot{s}$ that we can integrate ($\int \dot{s}dk$) through the function $\beta(s^k_{1:q}, v^k_{1:q})$ to obtain the forward states of the agents $s^{k+1:k+N}_{1:q}$. It is important to highlight that Algorithm 7 is written as a minimization.

Algorithm 7 Gradient-based algorithm

Require: $b^{k|k}_\tau$ ▷ Prior information
Require: $s^k_{1:q}$ ▷ Initial agents state
$t \leftarrow 0$ ▷ Initial iteration
$v^t_{1:q} \leftarrow$ initialize the actions vector
while Not terminated **do**
  $s^{k+1:k+N}_{1:q} \leftarrow \beta(s^k_{1:q}, v^t_{1:q})$ ▷ Forward states
  $\gamma \leftarrow$ Compute step parameter
  $v^{t+1}_{1:q} \leftarrow v^t_{1:q} - \gamma \frac{\partial J(s^{k+1:k+N}_{1:q}, b^{k|k}_\tau)}{\partial v^t_{1:q}}$
  $t \leftarrow t + 1$
end while

Due to the non-convexity of the MTS problem, the solutions obtained are locally optimal. Although its convergence is proved to work for static targets, it can be used as well for dynamic targets, but the Jacobian computation and the possibles local maxima complicate the convergence proof. We explain deeply this method in Chapter 5

3.7 Summary

In this chapter, the MTS has been defined showing the elements involved in the problem and their properties. For a more clear comprehension, two examples are described, as well as many applications that motivate this thesis. The presented necessary conditions for optimizing the expected time by optimizing the probability of detection show why the MTS is a new challenging problem not solved yet. Moreover, the evaluation of the team trajectory as a probabilistic formulation for searching dynamic targets has been discussed, offering a set of mathematical resources to derive MTS utility functions.

The decision making theoretic base for the MTS has been defined and a tractable decision model compounded of three reward functions (immediate, short term and long term) is the core of the open loop controller that guides the agents to find the target in minimum time. Three MTS strategies that tackle the MTS have been proposed: minimizing the local expected time (LET, using a function that allow us to compute partially the expected time without loosing the minimization properties), maximizing the discounted probability of detection (DTR, including a discounted time function as λ^k to simultaneously minimize the time and maximize the probability of detection) and optimizing the discounted time heuristic (DTH, computing the expected observation heuristically to combine with the short term utility functions). Besides, combining the DTH with other probabilistic strategies we have designed some informed strategies such as: DTRH, LETH and DH.

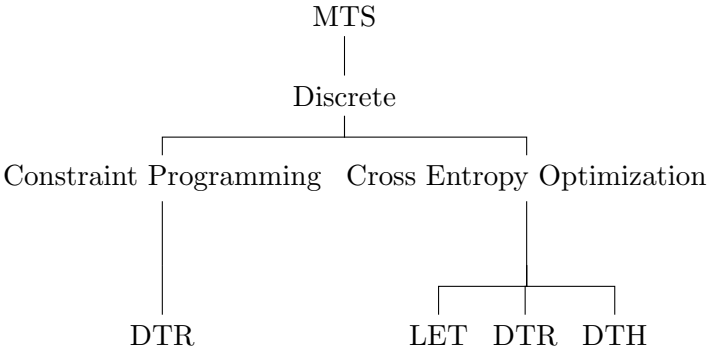
Finally, three types of non-convex optimization algorithms have been discussed to solve the MTS using the proposed strategies. The first type corresponds to the constraint programming paradigm and is suitable for static targets in a discrete model. Still, its scalability for large scenarios is limited. The second types of algorithms are stochastic sampling methods that are suitable to solve the general discrete MTS for a specified horizon. The third and final approach tackles approximately the continuous MTS as a piecewise gradient-based optimization.

This chapter groups and summarizes the whole theoretic contribution of the thesis that will be used and extended in the following chapters, where we apply the strategies and algorithms to synthetic and real applications in order to provide results that validate our proposals.

Chapter 4

DISCRETE APPROACH

"I do engineering, not religion" Daniel J. Bernstein



4.1 Contents

We analyze the MTS for a discrete point of view, where the region to explore is discretized into cells, making the distribution that describes the target location become a Probability Mass Function (PMF), which indicates the probability of the target existence in each cell. The aim of the chapter is to contribute with algorithms that work as the agents team controller to detect the target in the minimum possible time.

Once the world is represented by cells we need to define their adjacency (i.e. which cells are connected), thus, we finally transform the state space into a graph where the vertexes are the cells and the edges are the connections. By using the graph abstraction, the agents location is a vertex of the graph and their possible actions at each instant are the output edges of that vertex. The target also moves within the graph, so its dynamics are defined by the probability of going from one vertex to another.

We look for algorithms that find the global optima for N decisions and due to intractability issues we present 1) a global optima algorithm that assumes that the target is static during the planning horizon[1] (Section 4.3) and 2) a global suboptimal approximation that deals with the dynamic target and any type of sensor model (Section 4.4).

The first algorithm is based on constraint programming and finds the exact solution providing an optimality proof, but it has some disadvantages. It is only able to tackle static targets and ideal binary agent sensor models. The method only exploits the Discounted Time Reward (DTR) strategy (Section 3.5.2.2) and obtains the MTS solution when the number of decision variables is small. Also, this approach gives us the first view of the model predictive receding horizon controller bringing out the complexity issues of the multiagent MTS. In this solution we use the centralized approach.

The second algorithm of this chapter tackles the MTS when the target is dynamic and the agent sensor could be of any kind. The algorithm presented is able to compute the best N actions with the three strategies formalized in Section 3.5.2: the local expected time (LET), the discounted time reward (DTR), and the discounted time heuristic (DTH). This proposal is an stochastic suboptimal algorithm that makes a trade-off between optimality and computation time. The implementation is based on the Cross Entropy Optimization (CEO) method [Rubinstein and Kroese, 2004]. Although decentralization versions of CEO could be applied, we have not analyzed their use due to the lack of optimality proof. Thus we have focused on applying the algorithm into a centralized multiagent configuration.

In this chapter, we first present the framework used in the general *discrete* MTS (Section 4.2) and afterwards we present the two algorithms (Sections 4.3 and 4.4). The description of both algorithms is organized similarly by, starting with the strategies representation,

[1] Note that the static target assumption is a widely used approximation in the literature [Mathews et al., 2007; Gan and Sukkarieh, 2010].

continuing with the explanation of the optimization method itself and finalizing with some performance results. Moreover, at the end of the chapter we show an exhaustive analysis of the algorithms for the different strategies (Section 4.5). The chapter is concluded with the contribution highlights (Section 4.6).

4.2 Modeling the Problem

In a search problem, we have sensing agents capable of maneuvering freely and gathering information about the targets existence in a mission defined work space. This discrete approach allows us to make high level decisions (i.e. cardinal directions). In this section, we state the problem using a discrete approach of the world, agent dynamics and sensor model definition, and describe the location target belief by means of a probability mass distribution. Besides, we assume that the target is not evading from the searcher.

The objective of the optimization is to compute the best action plan at instant k, taking into account the observations. Each action u^k makes the agent transit to another state or position over the search space Ω. As we assume that the sensor position is the same as the agent position, we have the action plan $v_i^k = \{u_i^k, \cdots, u_i^{k+N-1}\}$ that moves the i-th agent/sensor along the states $\{s_i^{k+1}, \cdots, s_i^{k+N}\}$ making an observation z_i^k at each s_i^k state. As we have explained in Section 2.6, the data fusion layer uses the real agent observations and the action planning layer predicts the observations using a sensor model. Therefore it is important to distinguish between the data fusion layer and the algorithms presented here that implement the controller layer. The information layer implemented as a RBE algorithm (Section 2.4.3.1) is described briefly when we explain the target model.

Figure 4.1 illustrates the MTS problem: there is a delimited searching region Ω that is discretized into cells that are 8-connected (using cardinal directions) except for the region frontiers; the agents and the target are contained in Ω; the agent, starting at s_i^k, moves over the grid to make observations z_i^k and detect the target; and depending on the actions (u_i^{k+j}) the agent will observe different cells; The solution is the sequence of actions $v_{1:q}^{k*} = \{u_{1:q}^k, \cdots, u_{1:q}^{k+N-1}\}$ over the grid that makes the agents locate the target in minimum time.

4.2.1 The World

Lets us define the delimited mission search space as $\Omega \in \mathbf{R}^2$ where the agents and the targets are contained. The world in our approach is discretized into a two dimensional grid with $w_x \times w_y$ cells. This grid is described mathematically as a graph $G = (V, E)$ by assigning each cell to a node, and defining the edges of the graph as the adjacency among cells.

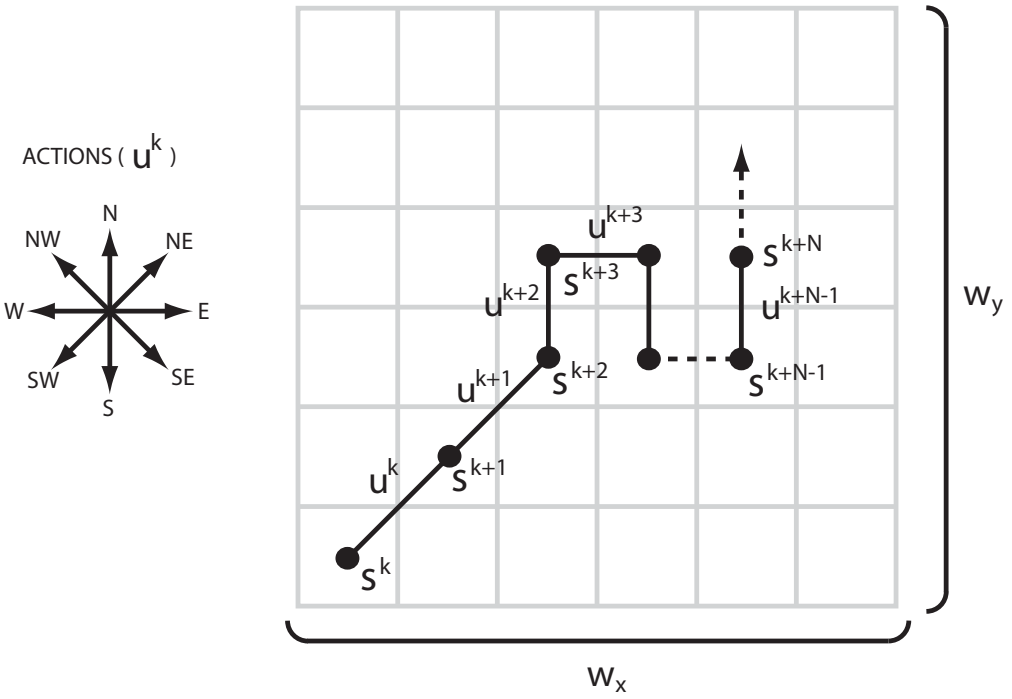


FIGURE 4.1: Scenario for the searching problem. The solution is a sequence of actions that drives the agent to detect the target in minimum time. The searching region is a set of cells that contain the agent and the target. The target starts in s^k with an a priori information of the target (i.e. the probability of the target being at each cell) that it will be updated with the observations made at $\{s^k, \cdots, s^{k+N}\}$. The agent actions $\{u^k, \cdots, u^{k+N-1}\}$ are the cardinal directions that correspond to the cells adjacency.

4.2.2 Agent Motion Model

We consider an agent with discrete actions dynamics that are restricted by the edges of the graph G induced by the two dimensional grid. As each cell surrounded by 8 cells is accessible, the possible action values (u_i^k) for each instant k are the eight cardinal directions: N, NE, E, SE, S, SW, W and NW. We have an agent i with an action vector composed of N discrete actions $v_i^k = \{u_i^k, \cdots, u_i^{k+N-1}\}$, which makes the sensing agent transit from state s_i^k into $\{s_i^{k+1}, \cdots, s_i^{k+N}\}$ as time increases. In other words, s_i^k is the cell of the agent i in the grid at instant k and v_i^k is the action plan computed for the next N steps.

4.2.3 Sensor Model

The sensor model represents the likelihood of producing a target detection event D from the observation z_i^k at time step k, if the sensor is at state s_i^k and the target is at position τ^k. Although the second algorithm presented in this chapter allows any type of sensor, for simplification purposes, the sensor model or observation likelihood used in this chapter is defined as (Section 2.4.2),

$$P(z_i^k = D|\tau^k, s_i^k) = \begin{cases} 1 & \text{if } \tau^k = s_i^k \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

This model implies that, only if the agent is at the same location s_i^k as the target τ^k, the object is discovered. We remind that D_i^k and $\overline{D_i^k}$ stand for $z_i^k = D$ and $z_i^k = \overline{D}$ respectively.

4.2.4 Target Model

The dynamics and location (τ^k) of the target are uncertain, so we define our knowledge about the target location probabilistically. We define the target motion model as $P(\tau^k|\tau^{k-1})$, which in our discrete space approach, describes the probability of the target to go from one cell to another. We define the target location in the probabilistic framework as the belief $b_\tau^k : V \to \mathbb{R}_{[0,1]}$ that is the probability of the target being in a location inside the region. This belief is represented as a Probability Mass Function (PMF), and therefore, the total probability of the belief is $\sum_{c\in V} b_\tau^k(c) = 1$. We refer to the whole region belief at instant k as b_τ^k and the probability of locating the target in a single cell $c \in V$ as $b_\tau^k(c)$. Note that V is the grid discretization of the region Ω.

The Bayesian approach permits us to start from a prior target belief b_τ^0 and update the information after every successive sensor observation. Also, we can predict the target position according to the probabilistic motion model. For the data fusion layer we use the RBE (Section 2.4.3.1) to update and predict the information inside the system.

Prediction Step This step allows us to predict the location of the target at instant k given the probability target distribution of the target at $k-1$, according to the target probabilistic motion model.

$$b_\tau^{k|k-1} = \sum_{\tau^{k-1}\in V} P(\tau^k|\tau^{k-1}) b_\tau^{k-1|k-1} \tag{4.2}$$

Update Step The following shows the general form of recursive Bayesian target belief estimation for a series of sensor observation events. We start with a prior target belief up to time step k, $P\left(\tau^k|z_{1:q}^{1:k-1}, s_{1:q}^{1:k-1}\right)$, conditioned on all previous sensor observations $z_{1:q}^{1:k-1}$ taken at sensor states $s_{1:q}^{1:k-1}$. At time step k, the new sensor observations $z_{1:q}^k$ at sensor state $s_{1:q}^k$ is $\overline{D}$. The posterior target belief $b_\tau^{k|k}$ can be expressed using Bayes rule:

$$b_\tau^{k|k} = \frac{1}{\eta} \prod_{i=1}^{q} P\left(\overline{D}_i^k|\tau^k, s_i^k\right) b_\tau^{k|k-1} \tag{4.3}$$

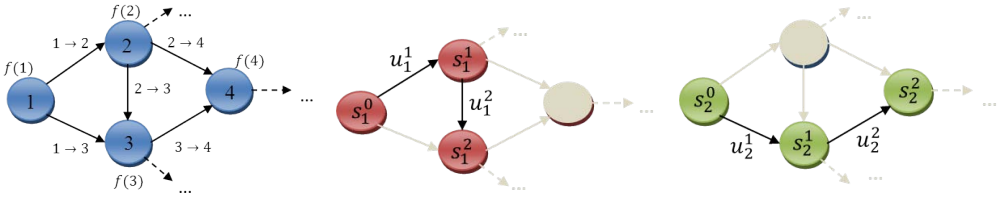
where η is the normalization constant that forces $\sum_{c \in V} b_\tau^{k|k}(c) = 1$. Note that the update step assumes that the target is not observed for the action planning purpose, because when the target is detected the task finalizes.

4.3 Global Deterministic Solution: Constraint Programming (CP-MTS)

- *Problem Model:* Discrete
- *Target Information Representation:* General PMF
- *Optimization:* Global
- *Horizon:* N
- *Sensor:* Ideal
- *Target:* Static
- *Data Fusion Layer:* RBE
- *Multiagent controller:* Centralized Constraint Programming Optimization

Our first approach to the MTS is based on modeling the decision problem with the constraint programming paradigm that permits codifying in a straight way the agent dynamic restrictions. Remember that the MTS is a sequential decision making process where the chosen sequential actions construct the path. The objective of this section is to provide a global optima algorithm to compute the best actions to detect a target with unknown location in minimum time. Due to complexity issues some hard assumptions are made yielding to important drawbacks that are the price of looking for global optimality. In other words, we achieve the optimal solution by solving a special instance of the MTS problem.

First of all, the sensor model is restricted to the ideal and binary one. Secondly the target is assumed to be static during the decision plan, which implies that for fast target dynamics this algorithm is just an approximation. Moreover, as we are using an ideal sensor, when two agents are visiting the same cell we can handle the coupling directly. On the positive side, the flexibility of this algorithm to define the agent constraints supports the generalization of our approach into other well-known applications apart from the MTS as the impatient trader or the discounted reward travel salesman problem [Blum et al., 2003].



(a) World Graph (b) Agent 1 Decisions (c) Agent 2 Decisions

FIGURE 4.2: DTR Example. Figure 4.2(a) shows a decision graph with the states with their associated scalar reward $f(c)$ and the possible actions $\{1 \to 2, 1 \to 3, 2 \to 3, 2 \to 4, \cdots\}$. Figures 4.2(b) and 4.2(c) show a solution of this instance of the problem for two agents, which perform different actions and visit different states.

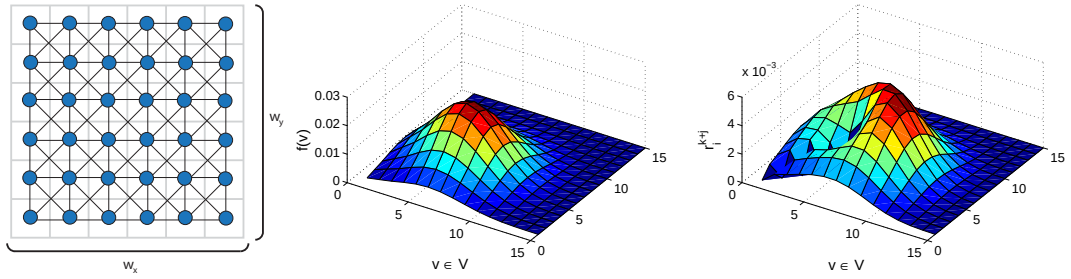
In this section, we have focused on the Discounted Time Reward (DTR) strategy that is one of the proposed utility functions to solve the MTS in Section 3.5.2.2. This function evaluates the agent actions giving more value to the places with high probability of detecting the target visited earlier. Besides it considers that the instant reward that the agents obtain for observing a location is the probability of detection discounted by a time dependent function.

An example of the MTS decision solution is presented in Figure 4.2. Defining the searching region as the graph showed in Figure 4.2(a), we have 4 possibles cell/locations to observe, represented as the blue circles. The actions that the agents can perform at each cell are the arrows that connect the vertexes. Because the target is static, the belief of the target location associated to each vertex $c_i \in V$ is the scalar $b_\tau^k(c)$. A solution to this particular example, when we have two agents starting in vertex 1 is described in Figures 4.2(b) and 4.2(c). Agent 1 performs the actions $\{u_1^1, u_1^2\}$ visiting the states $\{s_1^0, s_1^1, s_1^2\}$ and agent 2 executes $\{u_2^1, u_2^2\}$ transiting through the sequence of states $\{s_2^0, s_2^1, s_2^2\}$. Agent 1 starts at instant 0 in state $s_1^0 = 1$ and makes $u_1^1 = 1 \to 2$ observing state 2. Agent 2 selects at instant 1 the action $u_2^2 = 3 \to 4$ observing the state 4.

Before presenting the solution using a constraint programming model for finite domains, we define the optimization of the DTR strategy for discrete domains.

4.3.1 Optimization

On one hand, we have the world representation as the directed graph $G =< V, E >$ (see Figure 4.3(a)). Each $c \in V$ represents a possible location and each $e \in E$ represents a possible action that connects the location $c \in V$ to $c' \in V$. On the other hand we have a set of q agents with a fixed initial location $s_i^k \in V$ and a prior location distribution of the target b_τ^k, that is defined as a PMF that gives a probability value for each vertex.



(a) Grid Graph (b) b_τ^k (c) Discounted Time Reward

FIGURE 4.3: Figure 4.3(a) shows a grid decision graph of size 6×6, Figure 4.3(b) shows the belief associated with each vertex b_τ^k and Figure 4.3(c) shows the discounted time reward from vertex 65, i.e. vertex $(5, 5)$ in matrix notation, when the states $\{1, 17, 33, 49\}$ have already been visited.

Figure 4.3(b) shows the $b_\tau^k(c)$ of the grid graph as a height map, where the z axis values are the atemporal reward values and axis x and y index the possible states of the graph[2].

We want to plan N steps ahead, obtaining the best set of sequential actions v_i^{k*} for each agent i: $v_i^{k*} = \{u_i^k, \cdots, u_i^{k+N-1}\}$, where $u_i^k \in E$. Thus, the agent performs N actions starting in s_i^k and following the set of sequential states $\{s_i^k, ..., s^{k+N}\}$ and making the observations $\{z_i^{k+1}, \cdots, z_i^{k+N}\}$. The multiagent MTS solution must solve this optimization function:

$$\arg\max_{v_{1:q}^k} J(s_{1:q}^k, v_{1:q}^k, b_\tau^k) \tag{4.4}$$

Using the multiagent DTR strategy for an static target the utility function for the discrete domain is as follows (Section 3.5.2.2):

$$J_{dtr}(s_{1:q}^k, v_{1:q}^k, b_\tau^k) = \sum_{j=1}^{N} \lambda^j \sum_{\tau \in V} \left[1 - P(D_{1:q}^{k+j} | \tau, s^{k+j})\right] \prod_{i=1}^{j-1} P(\overline{D}_{1:q}^{k+i} | \tau, s^{k+i}) b_\tau^k \tag{4.5}$$

We can simplify the equation exploiting the advantages of using an ideal sensor: when two agents visit the same cell at instant k only one of them accumulates the reward; and the reward of visiting any already visited location is 0, because we always have non-detection. Therefore we can define the instant reward obtained by each agent r_i^k as:

$$r_i^{k+j} = \begin{cases} \lambda^{j-1} b_\tau^k(s_i^{k+j}) & \text{If } \left(s_i^{k+j} \neq s_m^l, \forall l < k+j, \forall m \in (1, q)\right) \wedge \left(\nexists n | s_n^{k+j} = s_i^{k+j} \wedge n < i\right) \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

[2]The vertexes of a grid graph of size $w_x \times w_y$ can be expressed in matrix form as $V = (x, y)$, where $x \in (1, w_x)$ and $y \in (1, w_y)$, or in vector form as $V = \{1, \cdots, w_x w_y\}$ by concatenating all the rows of the grid.

The expression states that the agent i gets 0 if the vertex s_i^{k+j} has been visited previously or is already visited by another agent at the same time step and $\lambda^j b_\tau^k(s_i^{k+j})$ if it is the first visit.

Using Eq. 4.6 we take into account the coupling and the product of all the previous observations of Eq. 4.5. Thus, the final simplified equation that we should optimize is:

$$J_{DTR}(s_{1:q}^k, v_{1:q}^k, b_\tau^k) = \sum_{j=1}^{N} \sum_{i=1}^{q} r_i^k \tag{4.7}$$

Figure 4.3(c) shows an example of Eq. 4.7 starting in vertex $s_i^{k+3} = 65$ when a single agent has already visited the states $\{1, 17, 33, 49\}$. The reward associated with the visited states are zero and the remaining rewards are weighted by the discount function that pushes down the belief accordingly with the time.

The aim of the optimization problem is to compute the team action vector that maximizes the joint discounted reward. Assuming that the agent actions are deterministic (i.e. if an agent in state $s \in V$ performs action $u \in E$, it always arrives to the same state $s' \in V$) we can describe the MTS solution in the states domain instead of in the actions domain. Thus, the MTS solution is the set S^k of sequential states that lets the agents obtain the highest joint discounted time reward. S^k can be defined as the following matrix of size $N \times q$:

$$S^k = \begin{bmatrix} s_1^{k+1} & \dots & s_1^{k+N} \\ \vdots & \dots & \vdots \\ s_q^{k+1} & \dots & s_q^{k+N} \end{bmatrix} \tag{4.8}$$

Analogously, we can also define the agent individual obtained reward after carrying out the actions as:

$$R^k = \begin{bmatrix} r_1^{k+1} & \dots & r_1^{k+N} \\ \vdots & \dots & \vdots \\ r_q^{k+1} & \dots & r_q^{k+N} \end{bmatrix} \tag{4.9}$$

Therefore, the optimal solution S^{k*} for this instance[3] of the MTS problem in the state space is the set of sequential states that maximize Equation 4.10:

$$S^{k*} = \arg \max_{s_{1:q}^{k+1:k+N}} J_{DTR}(s_{1:q}^{k:k+N}, b_\tau^k) \tag{4.10}$$

[3]We remind that the MTS subproblem tackle in this section has a discrete representation, uses an ideal sensor and optimizes the DTR strategy.

4.3.2 Constraint Programming Algorithm (CP-MTS)

In this section we present the CP-MTS algorithm that solves, using the DTR strategy, the MTS with the characteristics presented in Section 4.3. We have decided to use the CP paradigm because CP approaches have been proved to be very competitive and deal with many NP-hard problems and we are looking for a global optima and tractable algorithm. We remind that the tractable related approaches in Table 2.1 are approximations with local optimization. We have selected the JaCoP open source solver [Kuchcinski, 2003] due to its flexibility and facility to include new constraints and recode the search method. To achieve a tractable and efficient algorithm, our CP model includes state-of-the-art constraints and new specific constraints that work as an upper and lower bound of the problem. Moreover, we have designed an efficient search and labeling that exploits the MTS characteristics reducing considerably the computation time. One of the important features of using this algorithm inside the JaCoP solver is that we have proof of optimality, when the search of JaCoP is complete.

We present the CP-DTR algorithm in the following two sections: the CP model in Section 4.3.2.1, with the variables, the constraints, and the optimization function; and the search and labeling procedure in Section 4.3.2.2.

4.3.2.1 Constraint Programming Model

To model the problem in CP we determine the decision variables, define the function to optimize, and describe the used constraints.

The decision variables are the agent states S^k (Eq. 4.8) and the rewards R^k (Eq. 4.9). Although those variables are related, we treat them separately to improve the search (see Section 4.3.2.2 for a justification) and use a constraint to link their values. Therefore, if q and N are the number of agents and decisions respectively, the number of decision variables becomes $2qN$.

The multi-agent DTR optimization function is:

$$\max J(s_{1:q}^{k:k+N}) = \max \sum_{i=1}^{q} \sum_{j=1}^{N} r_i^{k+j} \tag{4.11}$$

The CP constraints that the DTR have to fulfill are:

1. Fixed initial vertex for all agents:

$$s_i^k = c | c \in V$$

2. The states must be valid, i.e., the vertices inside a path $\{s_i^k, \cdots, s_i^{k+N}\}$ should be connected by edges. This restriction that exploits the repetitive structure of the grid is easily codified by a Multivalued Decision Diagram (MDD) [Cheng and Yap, 2010] that allows us to define the connections of the graph as a pair table (Figure 4.4).

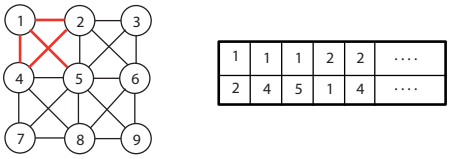


FIGURE 4.4: Directed grid graph edges codified as a MDD table as node origin and node destination pair.

3. The rewards have to be weighted by a discounted time function. Therefore, we build a function table that links the vertex $c \in V$ with its real reward value according to the instant j:

$$g(c, j) = \lambda^j b_\tau^k(c) \quad \forall c \in V \quad \forall j \in (1, N)$$

Then we map the agent state s_i^{k+j} with the reward r_i^{k+j} by using the element constraint:

$$element(s_i^{k+j}, j, r_i^{k+j} := g(s_i^{k+j}, j))$$

where depending on the value that is grounded on s_i^{k+j} and on the temporal parameter j, it selects the corresponding term of the function table $g(s_i^{k+j}, j)$.

We also introduce additional constraints to improve the algorithm performance:

1. *Upper an lower bound constraint.* The maximization problem in Eq. 4.11 is an addition and therefore, the total reward, defined as the sum of benefit variables, is the variable that the search will actualize every time a solution is found. The sum is a weak constraint because it does not provoke failures in propagation. Thus, to improve the optimization, we have developed a specific constraint for the problem that makes the solver prune faster and more guided than using just the sum. The new constraint is formed by a lower bound and an upper bound that tight progressively the possibles values of the reward variables. The upper and the lower bounds are the maximum and the minimum value that the total reward $J(s_{1:q}^{k:k+N})$ can have. To obtain this constraint, we take into account that during the search we can distinguish two types of variables: fixed/grounded (i.e. variable that have been labeled with an integer value) and not grounded (i.e. variables whose domain is a set of values). Besides, the search grounds, first of all, the reward variable r_i^{k+j} and, afterwards, the state variable s_i^{k+j}.

We define the lower bound R_{min} as the sum of the rewards variables associated with the state variables already grounded ($grounded(s_i^{k+j})$):

$$R_{min} = \sum_{i=1}^{q} \sum_{j \in grounded(s_i^{k+j})} r_i^{k+j} \tag{4.12}$$

We define the upper bound as the maximum reward that the solution can achieve. This is computed by a greedy heuristic based on the maximum values of the variables not grounded yet:

$$\hat{R}_{max} = \sum_{i=1}^{q} \sum_{j \notin grounded(s_i^{k+j})} \max r_i^{k+j} \tag{4.13}$$

Finally, the objective function is rewritten as:

$$J(s_{1:q}^{k:k+N}) = R_{min} + \hat{R}_{max} \tag{4.14}$$

This constraint is only executed when a state variable s_i^{k+j} is grounded, to avoid repetitive calculus and to make computation less intensive. It improves the search in two ways. On one hand, it estimates the future total reward value of an open branch when a variable is grounded making the prune more effective because the solutions with less $\hat{R}_{max}$ than the benefit necessary to improve the last solution are removed. On the other hand, combined with the graph connections coded with the MDD it introduces the grid restrictions into the rewards before labeling the variables, helping the prune.

2. To improve the algorithm efficiency we can include the following approximated constraint (AllDifferent): there should not be repeated states except for the starting vertex that can be the same for all agents:

$$\text{alldifferent}(s_{1:q}^{k+1:k+N})$$

Using this constraint the optimization time is reduced considerably but the solution is only optimal when the best solution does not include repeated locations. For this reason, the inclusion of this constraint makes this CP solution an approximated optimizer of the DTR function (Eq. 4.5). At the end of this chapter, at the results section (Section 4.5), we analyze in detail the advantages and drawbacks of including this constraint.

4.3.2.2 Searching and Labeling

Once the problem is modeled as CP, we can design an efficient search of the possible variables values. The CP solver is based on a first depth search that builds a decision tree whose root starts with all possible values of the variables and continues propagating the constraints and assigning values to the variables (labeling). There are three important properties that can help us to find the optimal solution. First, the solution should be found as soon as possible, because it will prune the decision tree branches that have a worse solution (i.e. for a maximization, any branch that cannot have more than the already obtained reward is pruned). Second, the branches that are opened inside the search should be close to the optimal value, because if the solutions founded are good enough, the CP solver will prune more branches and guide the search better. Finally, the values associated to the variables should help the propagation of the constraints eliminating as soon as possible not feasible solutions.

Analyzing the problem, we observe that we can search in the states variables (s_i^{k+j}) or in the benefit variables (r_i^{k+j}). If we find a feasible sequence (i.e. a solution satisfying all constraints) with the benefits ordered in decreasing order we will find the optimal solution. Thus, the most efficient approach for this problem is to search in the benefits and try to order them from the highest value to the smallest one. On the contrary, searching in the states domain gives many solutions with low benefit making the prune inefficient.

Searching in r_i^{k+j}, the solver prunes possible values of the variable domain forming sets of domains like $dom(r_i^{k+j}) = \{C_1, \cdots, C_p | C_m \cap C_n = \emptyset, \forall m \neq n, m \in (1,p), n \in (1,p), C_m \subset \mathbb{Z}, C_n \subset \mathbb{Z}\}$. The search opens a new node using a variable selector that chooses the new domain to try. The used variable selector picks the domain C_b that has the highest value:

$$\{C_b \quad | c \in C_b, c = \max(\cup C_n), \forall C_n \in dom(r_i^{k+j}), C_b \in dom(r_i^{k+j})\}$$

where max returns the maximum integer value contained in a set.

At the labeling process the decision variable r_i^{k+j} is set to one of the specified value of the selected set C_b. The value selector used chooses the highest value possible:

$$c = \max(C_b)$$

However, there is a drawback about searching in the domain of reward variables without taking into account the states: the solver finds a lot of infeasible solutions due to path restrictions (i.e. graph edges). That is, if the search first fixes all the reward values and then it starts to try the state values, there are too much backtracking caused by failures

at the MDD constraint. These failures can happen at a deep tree level, making the optimization untraceable. To solve it, we do a search by pairs of type $< r_i^{k+j}, s_i^{k+j} >$. That is, we force that every time a reward variable r_i^{k+j} is labeled, the associated state variable s_i^{k+j} is labeled as well, forcing the failures to take place before, reducing the backtracking considerably and maintaining potential feasible solutions.

The variable selector for the state variables (s_i^{k+j}) is designed to guide the search towards a solution as soon as possible by selecting the subdomain H_b of the variable with less elements:

$$\{H_b \quad | \quad |H_b| \geq |H_n|, \forall m \neq n, H_b \in dom(s_i^{k+j}), \forall H_n \in dom(s_i^{k+j})\}$$

where $|H_n|$ is the number of elements inside the set and $dom(s_i^{k+j}) = \{H_1, \cdots, H_p | H_m \cap H_n = \emptyset, \forall m \neq n, m \in (1,p), n \in (1,p), H_m \subset V, H_n \subset V\}$

The value selector for the state variable chooses the first value on the domain list:

$$h = first(H_b)$$

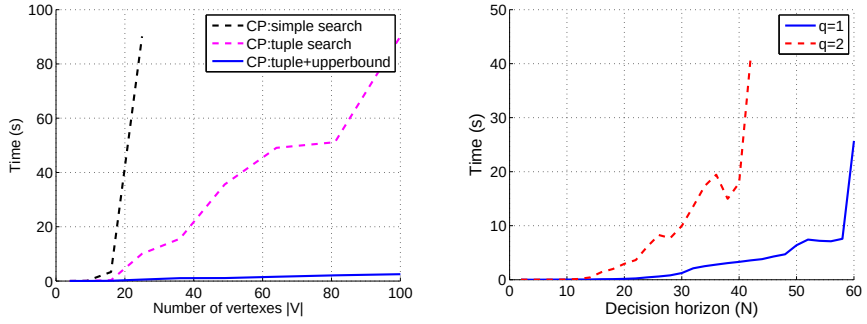
where $first(H_b)$ returns the first value contained in H_b.

4.3.3 Results

In this section we evaluate the CP-DTR performance by studying the computational time dependency on the number of vertexes and on the decision horizon N, and show how the designed constraints speed up the algorithm. All executions of the algorithm presented here incorporate the AllDifferent constraint approximation, that forbids to visit repeated locations. Complementary results, which include an analysis of the effects of the AllDifferent constraint are placed in Section 4.5. In this section we also provide a synthetic simulation of the MTS using the algorithm as a receding horizon controller. The CP solver is JaCoP [Kuchcinski, 2003], running in a Java virtual machine over a Pentium Core 2 duo 2.2 GHz with 2GB of RAM.

To analyze the computation time required by the algorithm to find the optimal set of agent states, we test the CP model with different improvements on different grid graph sizes ($|V| = 2^2, \cdots, 11^2\}$), using as the prior target location belief, a Gaussian distribution with global maxima at the center of the grid.

Figure 4.5 shows the computation time analysis of the CP-DTR algorithm with $\lambda = 0.8$ and the horizon $N = 40$ for one agent. First, in Figure 4.5(a), we study how the computation time required to give an optimal solution changes when we increment the searching region size. The time needed for CP-DTR without improvements (black dotted line) is too high, making the optimization intractable, while the time needed for the



(a) Improvements Comparative (b) Horizon Dependency

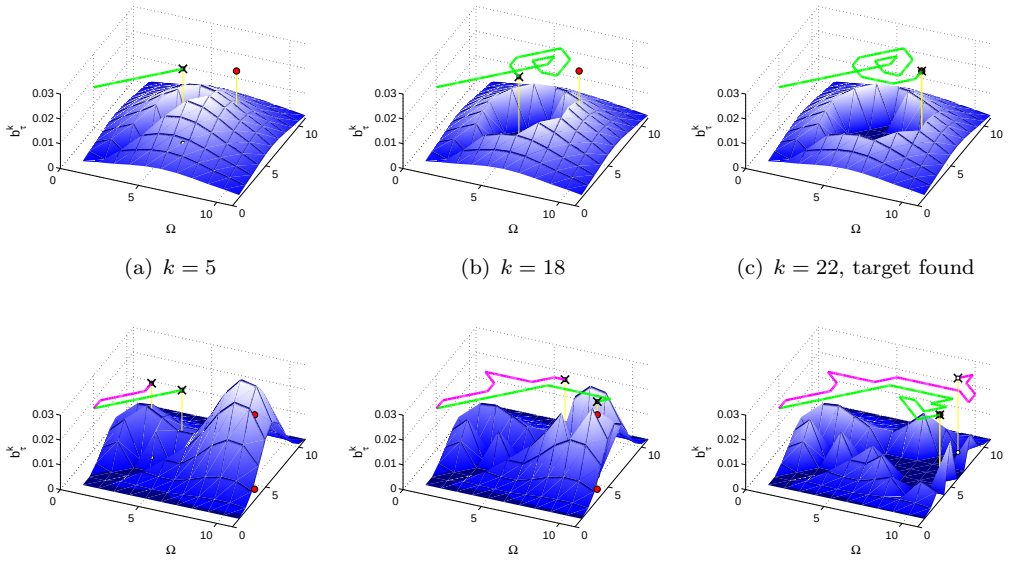
FIGURE 4.5: Figure 4.5(a) shows that with $\lambda = 0.8$ the computation time to get the optimal solution for a fixed horizon of $N = 40$ is exponential on the number of vertexes $|V|$ using the standard CP search on the reward variables, behaves linearly using the proposed tuple search, and has a drastically reduced computation time using the upper bound constraint. Figure 4.5(b) shows the time spent by the CP-DTR algorithm to compute a solution in a 10×10 grid graph, with one and two agents, when we vary the decision horizon N.

tuple search (magenta dotted line) is significantly slower. The inclusion of the proposed upper/lower bound constraints (blue line) speeds up drastically the algorithm, achieving solutions for graphs up to 121 vertexes in feasible time.

Figure 4.5(b) shows how the decision horizon N affects the computational time. For a grid graph of size 11×11 we compute the optimal solution for the horizons $N = \{2, \cdots, 44\}$ for two agents starting at the same cell and $N \in \{2, \cdots, 60\}$ for a single agent. These horizon ranges are selected because if we choose $N > 44$ for $q = 2$ and $N > 60$ for $q = 1$ the CP-DTR becomes intractable due to the non repeating vertexes hard constraint (AllDifferent). Nevertheless, the inclusion of the AllDifferent constraint dramatically reduce the computational time as we will see in Section 4.5. The results also show that the CP-DTR algorithm is tractable for two agents for an horizon up to 40 steps, which requires the computation of 160 decision variables.

In conclusion, the number of decision variables is the real bottleneck in the MTS problem due to the exponential estate expansion although the CP-DTR algorithm provides tractable solutions for small decision horizons instances of the problem.

Now we show an illustrative application for the CP-DTR algorithm: find a lost crashed airplane. Imagine that there has been an aircraft accident and the authorities have received the stress call from the plane crew a few minutes ago. Afterwards they have lost contact. The trasponder is not working, so we do not know the exact location of the plane. There can be survivors, thus the time is critical. Our search and rescue Department has two Unmanned Air Vehicles (UAVs) close to the accident area and wants to use them to find the plane as soon as possible. This is an application of the MTS problem that we solve here with our CP-DTR algorithm working as a centralized receding horizon controller.



(a) $k = 5$ (b) $k = 18$ (c) $k = 22$, target found

(d) $k = 6$ (e) $k = 10$ (f) $k = 19$, target found

FIGURE 4.6: The UAVs trajectory computed by solving MTS with the CP-DTR algorithm for a static target. In the first example (top row) the target is placed following a Gaussian distribution with global maxima at the center of the region. In the second example (bottom row) it is placed following a Gaussian mixture Figures 4.6(a), 4.6(b) and 4.6(c) show the simulation of the first example with a single UAV; while 4.6(d), 4.6(e) and 4.6(f) show the task solved by two UAVs for the second example.

Figure 4.6 shows some sequential instants of the MTS mission using one (first row of graphics) and two UAVs (second row), whose final positions at a given k are represented by black asps. The green and red lines are used to represent the UAVs trajectories. The plane is the red dot and the location target belief, which is initialized with a Gaussian in the first example and as a mixture of Gaussians with different local maxima in the second, is shown as a height map.

Figures 4.6(a), 4.6(b) and 4.6(c) show the results of the first simulation, where the UAV starts to hover the area from the south west, searches the target based on the optimal action sequence, and finds the plane at instant 22.

Figures 4.6(d), 4.6(e) and 4.6(f) show the results of the second simulation using two UAVs, where there are two regions with high probability of finding the plane. The UAVs initially split their trajectory towards the two regions and later they join forces to search in the most probable region until they find the target.

4.4 Global Approximated Solution: Cross Entropy Optimization (CEO-MTS)

- *Problem Model:* Discrete
- *Target Information Representation:* General PMF
- *Optimization:* Global approximation
- *Horizon:* N+Heuristic
- *Sensor:* Any
- *Target:* Dynamic
- *Data Fusion Layer:* RBE
- *Multiagent controller:* Centralized Cross Entropy Optimization

The second algorithm proposed to solve the MTS decision problem in discrete form is based on a sampling methodology called Cross Entropy Optimization (CEO) [Rubinstein and Kroese, 2004]. The objective of this section is to provide a global optima approximated algorithm that works as the agents controller layer and that improves previous approaches in terms of tractability. This proposal deals with any kind of sensor, target dynamics and tackles the information coupling. This is possible due to its sampling nature, where each sample is a possible solution that is evaluated by any utility function. For large MTS instances the number of samples needed to get the optimal solution is high and therefore we need a tradeoff between optimality and computation time. Theoretically, if we have enough samples, we can arrive to the global optima [Boer et al., 2002], but the quantity of required examples depends on the number of variables to optimize. Thus, in terms of scalability we will have suboptimal solutions for large MTS problems.

The algorithm has an iterative learning nature, where instead of computing the best actions, we learn the best actions distribution. Therefore, we do not have anymore a set of sequential actions as the solution, we have the discrete distribution of actions that better fit an optimization function. The method, in practice, minimizes the discrepancy between two distributions using the KullbackLeibler divergence. These distributions are the expected optimal action distribution and the best distribution of actions found by the algorithm at each optimization iteration.

In short, this formulation lets us propose a new discrete receding horizon controller, which provides the best N actions for a team of agents to locate a moving target with uncertain location in the minimum possible time. Figure 4.7 shows the MTS-CEO

controller connected with the information layer and the variables that are shared. The MTS-CEO controller takes as the input the last updated belief $b_\tau^{k|k}$ and starts an iterative process where it progressively improves the actions solution. Afterwards, the solution is sent to the mobile agent that uses the control actions to transit to other states where it makes new observations. Those observations update the target location belief using the RBE filter and the process is started again.

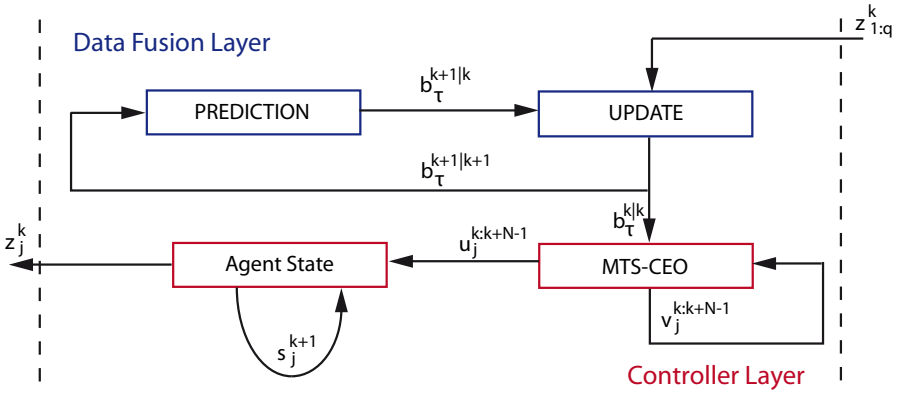


FIGURE 4.7: MTS system with the MTS-CEO controller.

We remind that we are considering q agents that cooperate in a team, and we want to compute the joint action vector $v_{1:q}^k$ at each instant k defined as: $v_{1:q}^k = [v_1^k, v_2^k, \cdots, v_q^k]$. We also assume that there is a Data Fusion (DF) layer that synchronizes the target belief at every agent (such that ${}^i b_\tau^k =^j b_\tau^k = b_\tau^k \forall \{i,j\} \in q$) and that feeds the controller (Figure 4.7).

4.4.1 Strategies in Discrete Form

Here we describe shortly the three MTS strategies in discrete form and the baseline strategy [Gan and Sukkarieh, 2010] used to compare the performance. By comparing the MTS strategies with the one widely used in the literature (baseline), with the same proposed CEO algorithm, we show that the algorithm is suitable for the optimal search and that the proposed strategies reduce more the detection time than the strategy used in the literature. To see a deeper derivation of the MTS strategies, please refer to Chapter 3, Section 3.5.2. To formalize the strategies in discrete form we have to take into account that the target possible locations are discrete $\tau^k \in \Omega$ and therefore to compute the instant probability of detection we need to sum all the probabilities instead of computing the integral over the target location τ^k.

4.4.1.1 Local Expected Time

One of the approaches to reduce the task time is to minimize the local expected time (LET) to find the object (Section 3.5.2.1). The LET of a team action vector $v_{1:q}^k$ given

the initial agents positions $s^k_{1:q}$ and the prior target location belief $b^{k|k}_\tau$ is:

$$\mu(s^k_{1:q}, v^k_{1:q}, b^{k|k}_\tau) = \sum_{j=1}^{N} \sum_{\tau^{k+j}} \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s^{k+j}_{1:q}, \tau^{k+j}) \tilde{b}^{k+j|k+j-1}_\tau \tag{4.15}$$

To compute $\tilde{b}^{k+j|k+j-1}_\tau$ we use the following equations (See Section 3.5.2.1 for the full derivation):

$$\tilde{b}^{k+j|k+j-1}_\tau = \sum_{\tau^{k+j-1}} P(\tau^{k+j} | \tau^{k+j-1}) \tilde{b}^{k+j-1|k+j-1}_\tau \tag{4.16}$$

$$\tilde{b}^{k+j|k+j}_\tau = \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s_i^{k+j}, \tau^{k+j}) \tilde{b}^{k+j|k+j-1}_\tau \tag{4.17}$$

where the prior target location belief is used at the first step of the recursion: $\tilde{b}^{k|k}_\tau = b^{k|k}_\tau$

Therefore, the algorithm has to optimize the following equation:

$$v^{k*}_{1:q} = \arg\min_{v^k} \mu(s^k_{1:q}, v^k_{1:q}, b^{k|k}_\tau) \tag{4.18}$$

4.4.1.2 Discounted Time Reward

Another strategy proposed in this thesis for the MTS is the Discounted Time Reward (DTR, Section 3.5.2.2) where we use a discounted time function to give more importance to the actions made in early instants. For the discounted time function we use a exponential function $f(k)$, that reduces the possible rewards (i.e. probability of detecting the target) as the time passes:

$$f(k) = \lambda^k \;\mid\; 0 \leq \lambda \leq 1 \tag{4.19}$$

The tuning parameter λ permits us to decide indirectly how fast we want to find the target or in other words, to model how important actions that the agent will take in the future are.

We define the multiagent DTR in discrete form as the following utility function:

$$J_{DTR}(s^k_{1:q}, v^k_{1:q}, b^{k|k}_\tau) = \sum_{j=1}^{N} \lambda^{j-1} \sum_{\tau^{k+j}} \left[1 - \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s^{k+j}_{1:q}, \tau^{k+j}) \right] \tilde{b}^{k+j|k+j-1}_\tau \tag{4.20}$$

where $\tilde{b}^{k+j|k+j-1}_\tau$ is computed in the same way as in the LET using the prediction (Eq. 4.16) and the assimilation without normalization (Section 4.17).

The solution (i.e. the team actions $v^{k*}_{1:q}$) to the MTS using the DTR strategy given the agents initial state $s^k_{1:q}$ and the prior target location distribution $b^{k|k}_\tau$ is the following equation:

$$v^{k*}_{1:q} = \arg\max_{v^k_{1:q}} J_{DTR}(s^k_{1:q}, v^k_{1:q}, b^{k|k}_\tau) \tag{4.21}$$

4.4.1.3 Discounted Time Heuristic

The heuristic versions of LET and DTR incorporates the DTH strategy where we include the heuristic sensor into Eq.4.15 and 4.20 as explained in Section 3.5.2.3.

$$\begin{aligned} J_{LETH}(s^k_{1:q}, v^k_{1:q}, b^k_\tau) = \sum_{j=1}^{N} \sum_{\tau^{k+j}} P(\overline{D}^{k+j}_{1:q} | s^{k+j}_{1:q}, \tau^{k+j}) \tilde{b}^{k+j|k+j-1}_\tau + \\ + \sum_{\tau^{k+N}} \prod_{i=1}^{q} \overline{\hat{H}}(s^{k+N}_i) \tilde{b}^{k+N|k+N}_\tau \end{aligned} \tag{4.22}$$

$$\begin{aligned} J_{DTRH}(s^k_{1:q}, v^k_{1:q}, b^k_\tau) = \sum_{j=1}^{N} \lambda^{j-1} \sum_{\tau^{k+j}} \left[1 - P(\overline{D}^{k+j}_{1:q} | \tau^{k+j}, s^{k+j}_{1:q})\right] b^{k+j|k+j-1}_\tau d\tau^{k+j} + \\ + \lambda^N \sum_{\tau^{k+N}} \left[1 - \prod_{i=1}^{q} \overline{\hat{H}}(s^{k+N}_i)\right] \tilde{b}^{k+N|k+N}_\tau \end{aligned} \tag{4.23}$$

The heuristic $\overline{\hat{H}}(s^{k+N}_i)$, with $0 < \beta \leq 1$ is as follows:

$$\hat{H}(s^{k+N}_i) = \eta \beta^{\|\tau - s^{k+N}_i\|/V_i} \tag{4.24}$$

where V_i is the agent velocity and η is used to either define a normalized heuristic ($\eta = 1/\sum_{s^{k+N}_i} \hat{H}(s^{k+N}_i)$) or unnormalized one ($\eta = 1$).

4.4.1.4 Baseline: Probability of Detection

We generalize the probability of detection utility function used in the literature for a team of agents and a dynamic target. This strategy does not minimize the time to detect the target (see Section 3.4), although using a short horizon we can reduce the detection time. Remember that two paths with the same joint probability of detection do not have to find the target at the same time.

The probability of detection until instant $k+N$ in discrete form is (Eq. 3.29):

$$J_d(s^k_{1:q}, v^k_{1:q}, b^{k|k}_\tau) = \sum_{j=1}^{N} \sum_{\tau^{k+j}} P(\bigcup_{i=1}^{q} D^{k+j} | s^{k+j}_{1:q}, \tau^{k+j}) \tilde{b}^{k+j|k+j-1}_\tau = \tag{4.25}$$

$$= \sum_{j=1}^{N} \sum_{\tau^{k+j}} \left[1 - \prod_{i=1}^{q} P(\overline{D}^{k+j}_i | s^{k+j}_i, \tau^{k+j}) \right] \tilde{b}^{k+j|k+j-1}_\tau \tag{4.26}$$

4.4.2 Cross Entropy Optimization

As a black box, our algorithm works as follows: given the starting agent states $s^k_{1:q}$ and the prior target location belief $b^{k|k}_\tau$ the algorithm returns the best N sequential actions $v^{k*}_{1:q}$. In practice, Eq. 4.18, 4.21, 4.22, 4.23 and 4.26 are non-convex discrete optimization problems. We propose the cross entropy optimization approach [Rubinstein and Kroese, 2004], due to its good performance in solving countable problems [Boer et al., 2002].

The idea behind CEO is to learn a probability distribution that lets us sample the optimal action plan ($v^{k*}_{1:q} = u^{k:k+N-1}_{1:q}$). CEO learns this probability iterating two steps. In the first, it samples the solutions from the probability distribution obtained so far and selects the set with the best ones. "Best" is defined according to the problem optimization criteria: minimum local expected time (LET, Eq. 4.15), the maximum time discounted reward (DTR, Eq. 4.20), minimum LETH (Eq. 4.22), maximum DTRH (Eq. 4.23) or maximum detection (Eq. 4.26). In the second, it obtains the parameters of the distribution from the samples minimizing the cross entropy between the obtained distribution and the optimal one, that will let us compute, using importance sampling, the percentage of the best solutions.

To solve the MTS using CEO we have to transform the action solution $v^k_{1:q}$ into a probability distribution $\hat{p}$ (i.e. the probability of taking the action u at instant k). The $\hat{p}$ is written with a hat because CEO will estimate its value. Also we have to design the samples as action solution instances. A sample X is a binary matrix $(8 \times N \times q)$, representing which action u is taken at instant k for each agent i. An example for one agent, a 2×2 grid world and action sequence E, S, W is showed in Figure 4.8. The agent starts at cell 1 and then it takes the actions (East, South and West), arriving to cell 3.

Finally we have to identify the utility function J that will be used to evaluate the samples. Every iteration j, CEO computes an estimation of the utility function used ($\hat{J}^j$) and, when it converges, it is the optimal solution.

Before presenting the method, we introduce some additional notation:

- $\hat{p}_{uki}$, learned probability of taking the action u at instant k for agent i

$$X = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

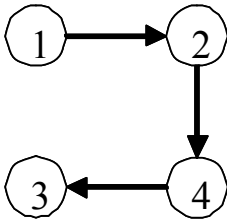


FIGURE 4.8: Action-time-agent representation to code a sample solution of the problem. On the left the binary matrix representing the action u is taken at time k. On the right it is show the translation to a path over the graph if the agent starts in cell one.

- X, binary matrix $(8 \times N \times q)$ whose cell x_{uki} represents that action u is taken at instant k for agent i. That is:

$$x_{uki} = \begin{cases} 1 & \text{if action u is taken at instant k by agent } i \\ 0 & \text{otherwise} \end{cases} \tag{4.27}$$

  Note that x_{uki} is an action-time-agent representation while $u_{1:q}^{k+1:k+N}$ is a sequence of actions.

- $\aleph = \{X_1, \ldots, X_M\}$, set with M samples for the CEO algorithm. To index the values x_{uki} for each X_e of the set we use this notation: $x_{uki,e}$.

- I_C, indicator function with condition C, that returns 1 when the condition is achieved and 0 otherwise. It is used to indicate which samples fulfill the condition.

- $\hat{J}$, reward of the best policy in the set $\aleph$.

- ρ, rare event occurrence probability.

- Finally we use the superscript j in q_{uki}^j, X_i^j, $x_{uki,e}^j$ and $\aleph^j$ to denote the j-th iteration number.

The optimal MTS solution is a vector $(s_{1:q}^k, u_{1:q}^{k:k+N-1})$ that contains the starting point and a set of sequential actions. To compute $u_{1:q}^{k:k+N-1}$ we learn the probability distribution $\hat{p}_{uki}^*$, which will let us sample the optimal actions from it. In order to learn $\hat{p}_{uki}^*$ we generate action-time samples and estimate $\hat{p}_{uki}$ as a function that fits the samples that satisfies that J (i.e. the utility function) is better than a specified value $\hat{J}^j$ (whose value is also estimated as we explain later). $\hat{p}_{uki}^j$ is estimated iteratively. First of all, we need to initialize its values and generate the samples following the $\hat{p}_{uki}^j$ distribution. Later on, elite "rare" samples are selected, the reward $\hat{J}^j$ is estimated and $\hat{p}_{uki}^{j+1}$ is calculated based on cross entropy minimization. We loop again generating more samples with the new distribution $\hat{p}_{uki}^{j+1}$ until convergence, where $\hat{p}_{uki} = p_{uk}^*$. Finally we extract the solution: the optimal actions for the agent starting from a fixed vertex $u_{1:q}^{k:k+N-1^*} = \arg\max \hat{p}_{uki}^*$.

4.4.2.1 Algorithm

Algorithm 8 shows the CEO method written for maximization purposes. For minimizing the objective function, the inequality $\geq$ and the function *max* should be substituted by $\leq$ and *min* respectively. The algorithm stages are explained as follows:

- Line 1: Initialize the probabilities of taking each action at each time step $\hat{p}^0_{uki}$.
- Line 3: Generate a set of action-time binary matrices $\aleph^j = \{X^j_1, \ldots, X^j_M\}$ sampling their values according to $\hat{p}^j_{uki}$.
- Line 6: Update the best estimated reward $\hat{J}^j$ using the samples $\aleph^j$.
- Line 7: Consider that the best solutions are rare events with low probability (ρ) and select them $\aleph^j_{elite}$.
- Line 8: Learn $\hat{p}^{j+1}_{uki}$ as the function that better fits the rare samples $\aleph^j_{elite}$.
- Line 2: If the stop condition is satisfied, then $p^* = \hat{p}^{j+1}_{uki}$, else $j = j + 1$ and the algorithm goes to line 3.
- Line 10: Finally, extract the sequence of actions: $v^{k*}_{1:q} = \arg\max p^*$.

Algorithm 8 CEO algorithm

1: $\hat{p}^0 \leftarrow \mathbf{1}_{|A|\times N\times q} \cdot \frac{1}{|A|}$
2: **while** $|\hat{J}^j - \hat{J}^{j-N_I}| < \epsilon$ **do**
3: $\quad \{X^j_1, \ldots, X^j_M\} \sim \hat{p}^{j-1}$
4: $\quad \{w^j_1, \ldots, w^j_M\} \leftarrow \{J(X^j_1), \ldots, J(X^j_M)\}$
5: $\quad W_{ord} \leftarrow sort(\{w^j_1, \ldots, w^j_M\}$
6: $\quad \hat{J}^j \leftarrow W_{ord}(\lceil \rho \cdot M \rceil)$
7: $\quad \aleph_{elite} \leftarrow \{X^j_i | J(X^j_i) \geq \hat{J}^j\}$
8: $\quad \hat{p}^j \leftarrow \alpha \left(\frac{\sum_{X_i \in \aleph_{elite}} x^j_i}{|\aleph_{elite}|} \right) + (1-\alpha)\hat{p}^{j-1}$
9: **end while**
10: $v^* \leftarrow \arg\max \hat{p}^j$

4.4.2.2 Probabilities Initialization

The initial probability of taking an action u at time k is $\frac{1}{|A|}$, where $|A|$ is the number of possible actions.

$$\hat{q}^0_{uki} = \frac{1}{|A|} = 1/8 \tag{4.28}$$

4.4.2.3 Action-Time Matricial Samples Generation

In order to generate the set $\aleph^j$ of action-time-agent binary matrices we generate their values from the estimated distribution $\hat{p}^j_{uki}$ using multinomial sampling [Candy, 2009]. According to [Boer et al., 2002] the number of samples to generate must be at least $c \cdot |A| \cdot N \cdot q$ where $|A|$ is the number of actions per location, N is the decision horizon, q is the number of agents and $1 \leq c \leq 10$.

Multinomial sampling

- At each time k compute the cumulative sum vector of the probability distribution of the possible actions:

$$S_{m,k,i} = \sum_{l=1}^{m} \hat{q}_{lki}, \quad m = 1, \ldots, |A|$$

- Generate a uniform random number r
- Pick the action $u^k_i = \{m | S_{m,k,i} < r \leq S_{m+1,k,i}\}$.
- Establish the new vertex by[4] $s^{k+1} = \beta(s^k, u^k)$ and if there are actions left repeat the process.

4.4.2.4 Reward Estimation and Rare Event

Using the utility function J we look for samples that optimize[5] this reward by using the probability of finding a sample with $J(s^k, v^k, b^k_\tau) \geq \hat{J}$. Sampling a solution with optimal reward function is a rare event so we know that the probability of a sample with reward equal to the optimal reward $\hat{J}^*$ is really low. It is not possible to compute $\hat{J}^*$ in a close form but it is possible to estimate it assuming that it belongs to the samples with higher rewards. So $\hat{\gamma}$ is computed using the probability of finding a sample over a specified value and taking into account that it is a rare event. By definition, X_i codifies the same information as $u^{k+1:k+N}_{1:q}$ using the action-time binary matrix form instead of the action sequence form. Thus we can redefine the utility function for a sample as $J(X_i) = J(s^k_{1:q}, v^k_{1:q}, b^{k|k}_\tau)$ and we can rewrite the inequality $J(s^k_{1:q}, v^k_{1:q}, b^{k|k}_\tau) \geq \hat{J}$ as $J(X_i) \geq \hat{J}$. The probability of finding a sample with equal or higher rewards to $\hat{J}$ is:

$$P(J(X) \geq \hat{J}) = E\left\{I_{\{J(X) \geq \hat{J}\}}\right\} \tag{4.29}$$

[4] $\beta(s^k, u^k)$ is the function that maps the state and the action into the next state.

[5] Although the optimization can be either a maximization or a minimization depending on the strategy, we only describe the algorithm in maximization form.

For each iteration $\hat{J}^j$ is estimated adaptively as a tuning parameter. As we are maximizing, the reward $\hat{J}^j$ will be higher until it cannot be improved anymore. To compute $\hat{J}$, using a rare event, we select the worst reward of the ordered samples set of quantile (ρ), where ρ is a factor that sets the probability of the occurrence of the rare event. In a probability density function the rare events are far away from high frequency events, so the optimal sample can be found at zones with less accumulation than 0.01. Thus, a good choice of the parameter ρ will be 0.01. Let be $\aleph^j = \{X_1^j, \ldots, X_M^j\}$ be the set of samples, the update sequence is:

1. Evaluate samples with the utility function

$$\{w_1^j, \cdots, w_M^j\} \leftarrow J(X_1^j), \cdots, J(X_M^j)$$

2. Order the values in increasing order (for maximization purposes)

$$W_{ord} \leftarrow sort(\{w_1^j, \cdots, w_M^j\})$$

3. Set the estimated reward $\hat{J}^j$ as the first value of the samples that is inside the percentile $(\rho \cdot M)$.

$$\hat{J}^j \leftarrow W_{ord}(\lceil \rho M \rceil)$$

4. Select the elite samples $\aleph_{elite}^j$ as the ones that have the same or higher reward than $\hat{J}^j$:

$$\aleph_{elite}^j = \left\{ X_i^j | J(X_i^j) \geq \hat{J}^j \right\}$$

These elite samples are next used to update the learning policy.

4.4.2.5 Policy Update

Once $\hat{J}^j$ has been computed and the new elite samples $\aleph_{elite}^j$ have been selected, it is possible to learn a new $\hat{p}_{uki}^{j+1}$ fitting the value of its variables to the probabilities defined by the elite samples. If each variable of the sample is consider an independent Bernoulli variable, the new probability can be estimated as follows:

$$\hat{p}_{uki}^j = \frac{\sum_{i=1}^N I_{\{J(X_i^j) \geq \hat{J}^j\}} I_{\{x_{uki,e}^j = 1\}}}{I_{\{J(X_i^j) \geq \hat{J}^j\}}} = \frac{\sum_{X_i \in \aleph_{elite}^j} x_{uki,e}^j}{|\aleph_{elite}^j|} \tag{4.30}$$

This function says that to estimate the probability of each action at each time step for each agent we have to count the number of times that it happens in the sample set and divide this number by its cardinality, because the sum of the indicator function over the

samples with condition higher reward than $\hat{J}^j$ is exactly the number of elite samples $\aleph^j_{elite}$.

In order to reduce the fluctuations of the estimated probability distributions in different iterations and avoid local maxima, we smooth the updated $\hat{p}^j_{uki}$. That is, we estimate the probability that adjusts the elite samples with Eq. 4.30 and then we smooth it with the last probability computed $\hat{p}^{j-1}_{uki}$ using Eq. 4.31.

$$\hat{p}^j_{uki} = \alpha \cdot \hat{p}^j_{uki} + (1 - \alpha) \cdot \hat{p}^{j-1}_{uki} \tag{4.31}$$

where $0 \leq \alpha \leq 1$ is the smooth parameter.

4.4.2.6 Stop Condition

There are two easy ways to test if the algorithm has converged: analyzing the degeneration of the samples (i.e. if all $x^k_{uki,e}$ have the same value) and testing if the utility estimation, $\hat{J}$, changes from one iteration to another. However, degeneration is not appropriate, because there are usually many variables that never reach an steady state. Therefore, the best way to check if the optimization has converged is to test if $\hat{J}$ has reached a fixed value. Due to the relation $J(X) \geq \hat{J}$ the stabilization of $\hat{J}$ implies that there are little chances to find a better solution. Two consecutive $\hat{J}$ are considered equal when:

$$\hat{J}^j = \hat{J}^{j-1} \Rightarrow |\hat{J}^j - \hat{J}^{j-1}| < \epsilon$$

The optimization is considered to converge when there are N_I iterations where $\hat{J}^j$ value does not change. The stop condition is therefore:

$$|\hat{J}^j - \hat{\gamma}^{j-N_I}| < \epsilon$$

4.4.3 Results

In this section we present some performance results to evaluate the CEO-MTS algorithm. We compare the performance of the following strategies: LET (Eq. 4.18), DTR (Eq. 4.21), Detection (Eq. 4.26), the greedy strategy[6], and the random walk. Moreover, we show an illustrative application of the algorithm by showing the team behavior within a search and rescue mission. A more detailed analysis, which includes the studies of the heuristic informed strategies and multiagent scenarios, is presented in Section 4.5.

We evaluate the algorithms performance by statistically analyzing the probability of detecting the target at different instants taking into account the probabilistic nature of

[6]The greedy strategy only considers the immediate reward, which in discrete form, is selecting the adjacent cell with maximum probability (Section 3.6, Algorithm 4).

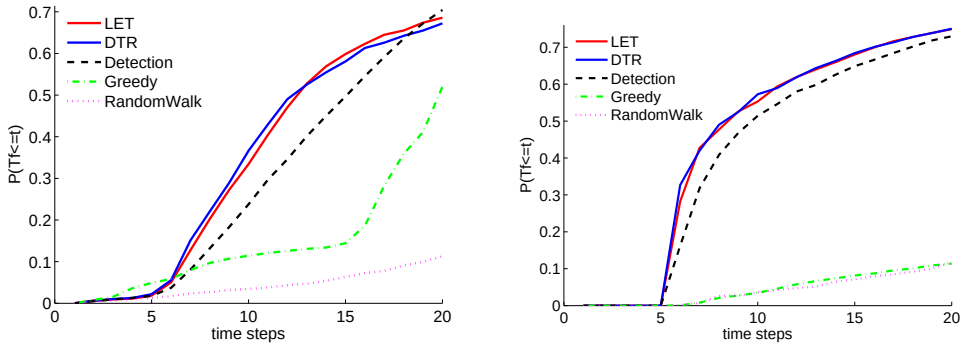
the problem (there is uncertainty in the target location and dynamics) and also, the stochastic nature of CEO (the solutions provided are non deterministic).

In order to obtain the data of this analysis we use two different single agent scenarios (with static and dynamic target), compute $N_a = 30$ actions plans using each of the algorithms for both scenarios, and run $N_s = 10000$ target search simulations for each scenario and precomputed action plan (i.e. we run $N_s * N_a$ simulations for each scenario and optimization strategy). Additionally,

- In the two scenarios used, the searching area is a square of dimensions $w_x = 8, w_y = 8$ and the starting location of the agent s^0 is fixed in the lower-left cell. The initial target location belief b_τ^0 of the static scenario is generated by propagating for a while a two peaks belief with a random transition matrix. The b_τ^0 for the dynamic case is builded as a zero vector of size $w_x w_y$ where the 48 position is set to 1 ($b_{\tau=48}^0 = 1$), meaning that the target starts in the opposite side of the square region. The target dynamics model $P(\tau^k|\tau^{k-1})$ is the transition matrix that spreads the belief probabilities for the dynamic case. In our dynamic scenario, the target and the agent velocity are equal (i.e., at every time step the agent and the target move from one cell to another), forcing the prediction step of the Bayesian filter to be computed every agent movement. The target dynamics $P(\tau^k|\tau^{k-1})$ is generated by extracting an 8×8 region of the Matlab wind database [MathWorks, 2010] and building a normalized transition matrix from it. We have chosen these dynamics due to their non-uniform properties, because our approaches improve better the performance with asymmetric transition matrices, where two paths have different reward. With this setup, the two scenarios only differ at the target dynamics $P(\tau^k|\tau^{k-1})$ and the prior location distribution b_τ^0.

- The CEO parameters used to compute each of the $N_a = 30$ actions plans for the scenarios and approaches are: $\rho = 0.01$, $\alpha = 0.6$. Besides, for the DTR approach the λ parameter is fixed to 0.8. In all the approaches the decision making horizon is $N = 20$.

- In each $N_s = 10000$ simulations run for each of the $N_a = 30$ actions plans obtained by each approach for each scenario, the target starts in any cell $c \in V$ of the searching region Ω with non zero probability in the initial belief b_τ^0. In the static case, it remains in that position, while in the dynamic scenario, it follows the selected target dynamics $P(\tau^k|\tau^{k-1})$. In both cases, for each simulation we store the time t_s that the agent spends to detect the target (calculated as the number of cells that the agent covers until it detects the target) and use it to compute, for each action plan and approach, the probability of detecting the target along the time $P(T \leq k)$. This probability is a cumulative density function and is computed as follows:



(a) Static target (b) Dynamic target

FIGURE 4.9: Cumulative probability of detecting the target along the time using five different strategies: DTR, LET, detection, greedy and random walk. In (a) the target is static and in (b) the target is moving.

$$P(T \leq k) = \frac{1}{N_s} \sum_{i=1}^{N_s} I_c(t_s <= k) \tag{4.32}$$

Using a non-optimized Matlab implementation of CEO running on a Intel Core 2 Duo 2Ghz, the dynamic scenario solution is found in 2 seconds generating 1600 samples per iteration and 550 seconds using 25600 samples. This yields into a trade-of between optimality and computation time. However in this example the solution improvement achieved by increasing the number of samples is only 1%.

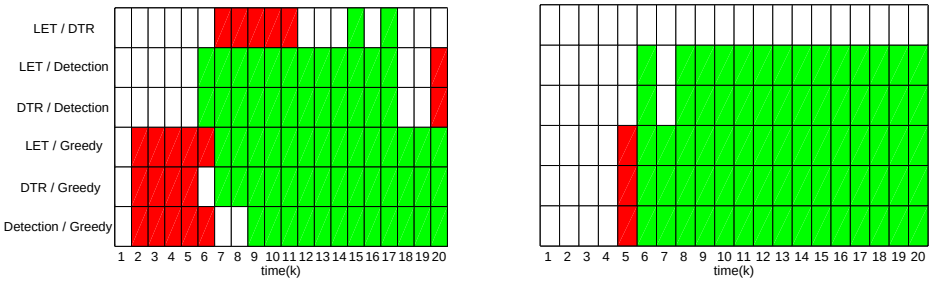
We use Eq. 4.32 to measure the performance of the algorithms, because the best strategy is the one that detects the target earlier, which is statistically equivalent to be the one that has a higher probability of finding the target in the first instants. That is, if the certainty increases faster the decision is better.

Based on this observation, we compare the solutions for each scenario in two ways:

1. Plotting the average (over the $N_a = 30$ actions plans) accumulated probability of detecting the target at each instant k.

2. Showing if the solutions provided by one algorithm usually dominate (are better in terms of the $P(T \leq k)$) the solutions provided by others, using the Wilcoxon test, that determines if there is a significant difference at each instant k.

Figure 4.9 shows the average accumulated probability of finding the target $P(T \leq k)$ for the action plans obtained by CEO with three utility functions (LET Eq. 4.15, DTR Eq. 4.20, and Detection Eq. 4.26), the greedy max algorithm and the random walk strategy. It should be noted that the standard detection (black dotted line) and the greedy algorithm (green dotted line) are subcases of the DTR, where $\lambda = 1$ and $\lambda = 0$



(a) Static target (b) Dynamic target

FIGURE 4.10: Dominance analysis of the algorithms in two different scenarios for the 20 first instants: static and dynamic. The algorithms are compared in pairs as showed at the y axis labels (algorithm 1 / algorithm 2). The figure follows a three color legend: the white means that there are not a significant difference; the green means that the algorithm 1 beats algorithm 2; the red means that the algorithm 1 is beaten by the algorithm 2.

respectively. As the accumulated probability of detecting the target, at lower values of k, is higher in the LET and DTR cases than in the detection and random walk strategies, we can conclude that in average, the solutions obtained by the two proposed strategies (red and blue continuous lines) have higher chances of finding the target sooner. The greedy strategy achieves in the static case (Figure 4.9(a)) better performance on the first instants because in the selected scenario there is a peak close to the initial location of the agent but, this blind behavior yields to bad performance in future instants. It is worth highlighting that the performance has a hard dependency on the scenario, as the performance differences between the static (Figure 4.9(a)) and dynamic scenario (Figure 4.9(b)) shows. This happens because the dynamic case spreads the probabilities and therefore the gain at each instant is lower.

Figure 4.10 shows the dominance study among the best algorithms. The dominance test lets us identify the algorithms that, when initialized with any random initial conditions, obtain improved or at least as good results as the others. The cumulative probability at each instant of the 30 CEO solutions is compared for each pair of algorithms by using a Wilcoxon test, that says if both distributions have the same median with a significance of 0.05. In the case that the median of the two algorithms are different, we say that the algorithm with better mean dominates (is better than) the other. The results describe the comparative between pairs of algorithms as it is shown at the y axis labels (algorithm 1 / algorithm 2) of Figure 4.10. Each row is a dominance test between two algorithms, and each column shows the comparison at one instant k. If the algorithm 1 dominates the algorithm 2 at instant k, the cell color is green. If the algorithm 1 is dominated by the algorithm 2, the cell color is red. Finally if there is not significant difference between the performance of the algorithms, the color is set to white.

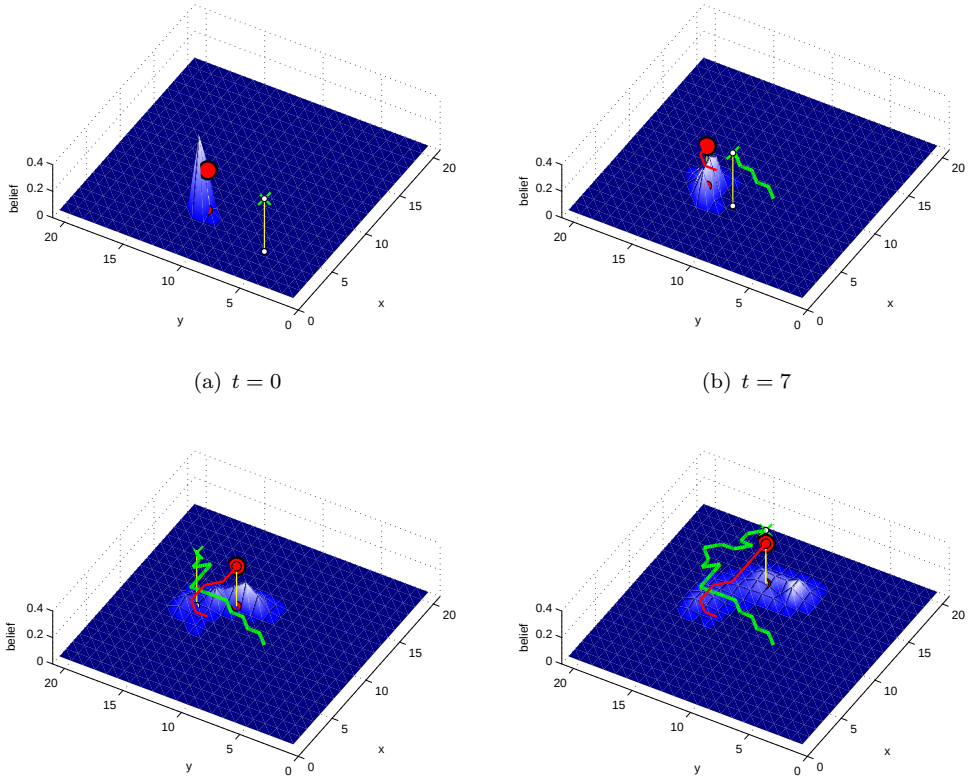
The dominance figures (Figure 4.10) show that LET and DTR achieve the best performance in terms of time reduction (i.e. more accumulated probability of detection in each

instant). In the static target case (Figure 4.10(a)) LET and DTR beat the Detection strategy in the majority of the instants. The greedy strategy, due to its myopic nature, dominates the first five instants but then it loses in the rest of the decision window. In the dynamic target case (Figure 4.10(b)) the dominance of the strategies proposed is even clearer than in the static case. After $k = 5$ LET and DTR beat the Detection and the greedy strategy. For its part, the Detection strategy beats the greedy one. In conclusion, LET and DTR approaches dominate the Detection one in both scenarios showing the improvement of our proposed strategies in terms of reducing the detection time.

Although Detection (4.26) has good performance, our approaches beat it, because they relay on a different objective. The difference between LET and DTR is that we can decide, in the second approach, the importance of finding the target at each instant, which means, for instance, that we can try to force the decision making algorithm to find the target in the first ten instants, no matters what happens afterwards. This is reflected in Figure 4.10(a) where at some instants DTR dominates LET. It is also interesting to highlight that with low values of λ DTR forces the agent to accumulate faster (with $\lambda = 0$ we have the greedy strategy where the policy does not take into account future instants) and with high values of λ the agent has more time to explore the region (with $\lambda \approx 1$ we have the Detection strategy).

We also show the CEO-MTS working as a controller within a synthetic search and rescue operation. A boat has sunk in the ocean in a known location and the crew has only time to inflate a rubber raft and get into the sea. We need to find them as soon as possible in order to maximize the chances of finding the people alive. We assume that the dynamics of the target (rubber raft) are given by the wind currents, so a probabilistic transition matrix is developed according to a wind database. The agent is an Unmanned Air Vehicle (UAV). For simulation purposes we assume that the UAV is 5 times faster than the target and we use the wind database provided by [MathWorks, 2010].

Figure 4.11 shows some sequential instants of the search until the UAV discovers the survivors. The UAV is represented by a green asp and a green line is used for its trajectory. The rubber draft is the red dot and its trajectory is also red. The probability map of the target location is shown as a height map. At instant 0 (Figure 4.11(a)), the UAV has started to hover the area from the south west. While the target is moving according to the wind, the UAV searches the target based on the preplanned action sequence. At instant 7 (Figure 4.11(b)) the UAV arrives to the area where the probability distribution indicates that the target could be found. At instant 15 (Figure 4.11(c)) the target moves again going away from the UAV. Finally the UAV finds the target at instant 27 (Figure 4.11(d)).



(a) $t = 0$ (b) $t = 7$

(c) $t = 15$ (d) $t = 22$, target found

FIGURE 4.11: The UAV trajectory computed by solving the MTS with CEO. The probabilistic model of the target dynamics is extracted from a wind map.

4.5 General Results

In this section we analyze the MTS problem from the discrete point of view. First of all, we study the optimization performance achieved by comparing the two algorithm proposed: CP-MTS (Section 4.3) and CEO-MTS (Section 4.4). Afterwards, we show how the strategies proposed (DTR and LET, Section 3.5.2) improve the maximum detection one (D, Section 3.4.2) in terms of minimizing the time to detect the target. Finally we study the impact of using the heuristic (DTH strategy, Section 3.5.2.3) at the myopicity of the algorithm. The study is based on three measures:

- Strategy (J) vs optimization time (s): This measure shows how the algorithm improves the solution quality according to the time spent to compute the solution (e.g. Figure 4.14(a)). It describes also the tractability of the algorithm and the time needed to converge to the optimal solution.
- Information gain (IG) vs step (k): It represents the accumulated probability of detection before instant k: $P(T \leq k)$ (e.g. Figure 4.17(a)). This distribution is

related with the theoretical expected time to detect the target and therefore higher values of the IG in previous instants imply finding the target before.

- Dominance study at each k: The dominance study analyzes the mean and the standard deviation showing which algorithm is better stochastically speaking (e.g. Figure 4.17(d)). Using the Wilcoxon test, that determines if there is a significant difference between two distributions we show which method wins according to the median disparity at each instant k. It uses a color code, where green means that the first method wins the second one, red means that the second method beats the first one and white expresses that there is no significant difference.

The whole study is done over the scenarios described in Figure 4.12. The selected scenarios differ in, the initial agents location (represented by the red stars) and the initial target location belief b_τ^k (which is no zero inside the black shapes) and target motion model (static or dynamic, the last following the black arrows). The same grid size (20×20) and sensor model (ideal) is used in all the scenarios. Finally, in the same graphs we also show with the red arrows the beginning of the best agent trajectories. The first row of Figure 4.12 are scenarios where the target is static and the second row are scenarios where the target is dynamic. The initial belief or probability map b_τ^0 for each scenario is displayed in Figure 4.13. To visualize the dynamic scenarios prior distribution instead of showing the first instant we show the probability distribution after 8 instants ($k = 8$) to provide a clear picture of the target movement.

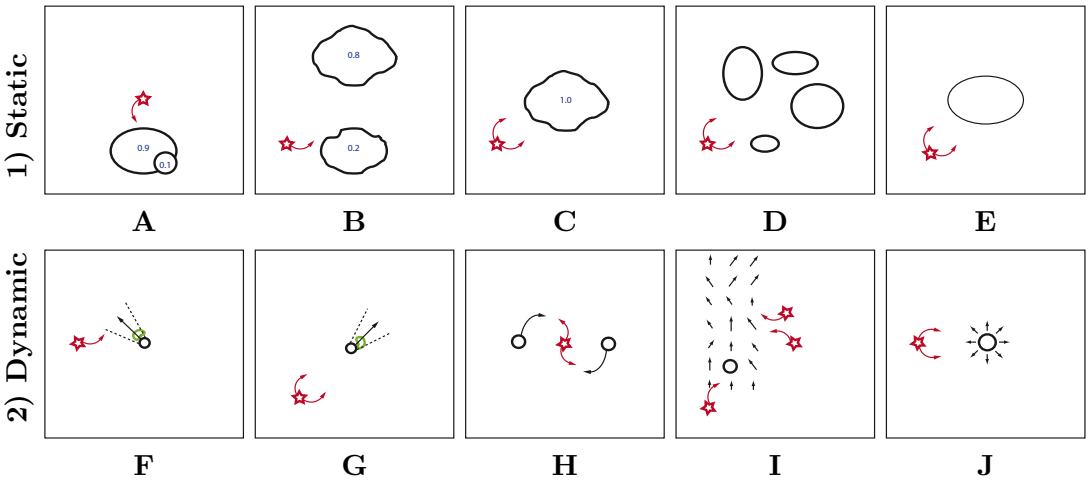


FIGURE 4.12: Scenarios schematically

The main characteristics of each scenario are:

Scenario A. The target location belief is condensed in a smooth ellipsoid region with one abnormality with 0.1 probability. The agent initial location is the center of the searching region. The purpose of this scenario is to illustrate the difference between maximizing the accumulated probability of detection and the strategies proposed in this thesis.

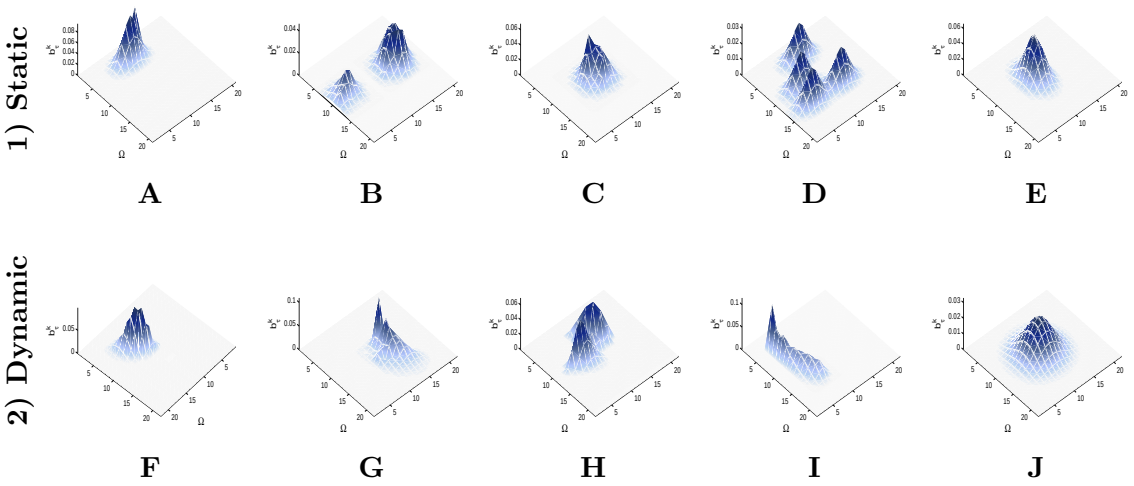


FIGURE 4.13: Scenarios prior distribution. For the static target scenarios we show b_τ^0 and for the dynamic target ones we show b_τ^8 assuming that each instant the target moves.

Scenario B. The target can be in two delimited regions generated by applying a random transition matrix 4 times to a 0.8 and 0.2 probability cell respectively. Defining the transition matrix as A the belief used is $b_\tau^k = A^4 b_\tau^0$. The agent starts close to the region with less condensed probability to test if the algorithms get stuck in local optima when the decision horizon is small. This scenario is used to illustrate if the algorithms are able to escape from the local optima and to show the effects of using the proposed heuristics.

Scenario C. The target probability distribution is generated by setting the center cell probability of the searching region to 1 and then applying n-steps of a random transition matrix. The agents start close to the probability mass. This scenario is used to study the agents decisions when the belief is random and therefore there is not a pattern underneath. Optimizers will have troubles to reach the optimum.

Scenario D. The target location distribution is a set of disjoint Gaussians. With this scenario we can study the MTS when the target can be in different regions.

Scenario E. The target location is described by a Gaussian distribution placed in the middle of the searching region. This scenario is a special case where maximizing the detection behaves similar to the proposed strategies, due to the gradient properties of the Gaussian probability distribution (where the necessary condition presented in Section 3.3 is satisfied).

Scenario F. The target starts in the center of the searching region and is moving towards north west, approaching the agents. The target location gets more uncertain as the time passes.

Scenario G. The target, starting in the center of the searching region, is moving away from the agents heading in the north east direction. This scenario shows the team behavior when the high probability regions are observed in later instants.

Scenario H. Two probability masses are turning around the agents until the target location believe becomes a uniform distribution. In this scenario the initial actions affects considerably the final results. Thus, the target dynamics should drive the team.

Scenario I. This scenario is the more realistic one because, instead of using a simple target dynamics, we have a transition matrix built from a wind map [MathWorks, 2010]. Moreover, we have a team of three agents with two starting in the same area and the other in a totally different place.

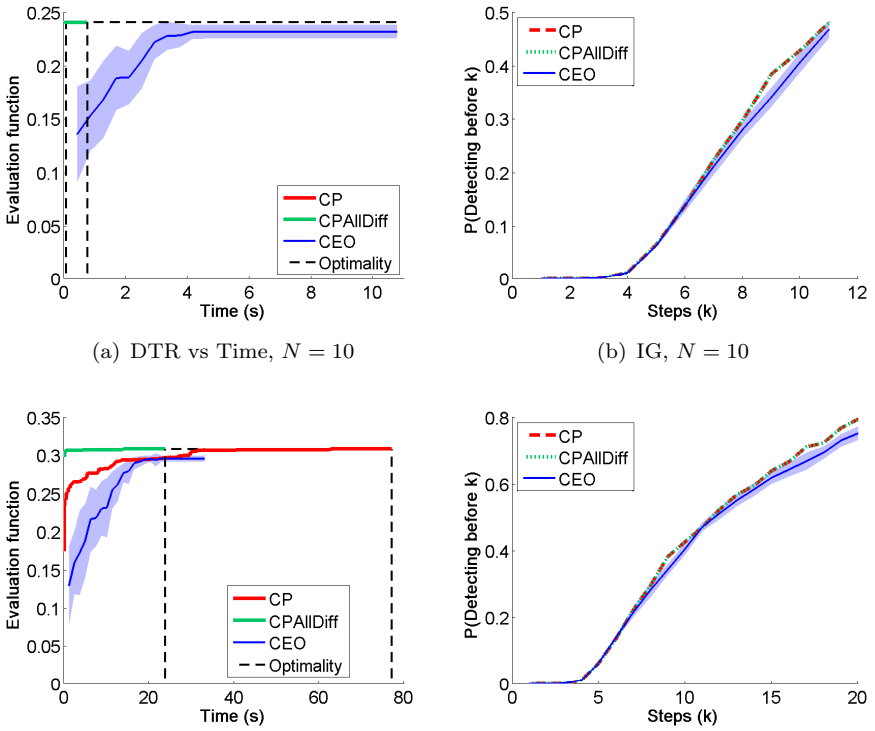
Scenario J. The target dynamics are uniform (i.e equally probable in all directions), spreading the probabilities away from the initial location. This configuration is used to show the team response when the target location is represented by a Gaussian distribution.

4.5.1 CP-MTS vs CEO-MTS

In this section we compare the performance of CP-MTS and CEO-MTS under the same scenarios. Due to the objective function and types of problems optimized by CP-MTS we need to select scenarios with static targets and ideal sensors, and we have to use DTR as the utility function. In the analysis we show if it is worthy to compute the optimum with CP-MTS instead of approximate its solution with CEO-MTS by measuring the quality of the solution and the time invested by both algorithms. Due to the deterministic nature of CP-MTS we only compute one solution with this approach for each scenario. Due to the stochastic nature of CEO-MTS, we execute this algorithm 50 times and then compute its mean and standard deviation values.

First we study in detail the optimality difference between the algorithms at scenario A of Figure 4.12. Afterwards we present the results for the rest of static scenarios. Figure 4.14 shows the comparison between the CP and CEO algorithms when optimizing the DTR strategy. Figure 4.14(a) and 4.14(b) shows the DTR function value depending on the optimizing time and the IG respectively for a horizon $N = 10$. Figure 4.14(c) and Figure 4.14(d) are the counterparts of Figure 4.14(a) and 4.14(b) for $N = 20$. In all figures we compare: CP, CP with the AllDifferent constraint (where the agents cannot visit the same location twice) and CEO with DTR. Moreover in the optimality figures (Figure 4.14(a) and 4.14(c)) we represent when the optimality proof is obtained by CP with a dotted black line.

With $N = 10$ the two CP algorithm versions are quite fast in finding the optimal solution and the optimality proof is achieved in less than 1 second (Figure 4.14(a)). The



(a) DTR vs Time, $N = 10$

(b) IG, $N = 10$

(c) DTR vs Time, $N = 20$

(d) IG, $N = 20$

FIGURE 4.14: CP-MTS vs CEO-MTS for scenario A.

CP (red line) is hidden by the CPAllDiff (green line) that takes some extra milliseconds to propagate the constraints. The CEO (blue line) takes more time to achieve its best solution (5 seconds) and we can see an offset with respect to the optimal solution, meaning that we are trapped into a local optima. This difference yields into a smaller increment of IG and therefore later expected detection time as it can be seen at step 8 in Figure 4.14(b).

With $N = 20$ we observe that CP with the AllDifferent constraint is very fast and gives optimal solution with optimality proof (Figure 4.14(c)). This is because the CP algorithm behaves better when we include more constraints and we do not loose optimality in this scenario because the optimal solution does not need to visit the same cell twice. In fact the CP and the CPAlldiff find the same solution. Although CP reaches the optimal solution in 63 seconds, it takes much time to visit all the possible solutions and proof optimality due to the states expansion. The CEO algorithm have again a negative offset from the optimal solution as it can be seen in Figure 4.14(c). The CPAllDiff (green line) proof the optimality at the same time that the CEO starts stabilizing (blue line) below both CP versions. It is important to highlight that there is no much difference between CEO solution at convergence time (24 seconds) to CP but, afterwards, CP is able to escape from the local maxima. Figure 4.14(d) shows the optimality difference in terms of IG, where we can see that the CEO curve is always below the CP ones.

In conclusion, when the number of variables is small we can use CP online to be sure that it achieves the optimal solution, and when the number of variables rises up we can use the AllDifferent constraint approximation or stop the algorithm at a certain time although we do not have optimality proof. CEO provides good enough solutions but it takes more time to converge and it has an offset comparing with the optimal utility function value.

Now we analyze the optimality in the same way for the rest of the static target scenarios in Figure 4.15, showing the tendency of the facts observed in scenario A. The label on the left side of the figures shows the scenario tested, the number of agents q and the decision horizon N.

The first row (B) shows the algorithms behavior in scenario B using 1 agent with a decision horizon of $N = 20$. The CPAllDiff (green line) has the best IG performance because it escapes from the local maxima produced by the low probability region close to the agent. CPAllDiff (green dotted line) reaches a solution that covers the high probability mass, without accumulating probability until step 8 avoiding the nearest region. CP (red and red dotted line) and CEO (blue line) achieve similar suboptimal solutions. CP is slow due to the states expansion and therefore it reaches the time out of the solver, settled to 400 seconds, although it will eventually find the optimal solution. CEO gets trapped in the local optima due to its sampling nature, because early steps decisions bias the optimization update. This drawback can be reduced by including a heuristic as we will study later.

The second row (C) studies scenario C with a team of $q = 2$ agents and a decision horizon of $N = 10$. It shows the algorithm performance when dealing with a pseudo-random probability mass. Although there is the same number of decision variables than in the previous scenario, CP reaches the optimal solution, showing two facts: longer decision paths are less tractable than using more agents and, the optimization is scenario dependent. CPAllDiff finds the optimal solution really fast prevailing over the other algorithms. It is interesting to highlight that there is not only one solution with the same DTR value as it is showed at the IG plot of Figure 4.15.

The third and the forth rows analyze the algorithm tractability when we increase the decision horizon using a team of two agents. Executing the optimization over scenario D we see that for $N = 10$ (D1), CP and CPAllDiff achieve optimality quickly and beat CEO. But for $N = 20$ (D2), the states explosion makes CP really slow. We have to remind that CEO and CP solve the same problem, while CPAllDiff does not allow to observe repeated locations. From this perspective, CEO beats CP considerably because CP constraints propagation is not enough restrictive to reduce the solution space. Although, its important to note that CPAllDiff gets better solutions really soon. One of the CP drawbacks is that when the agents start in the same location we have symmetrical solution subtrees that make harder the branching.

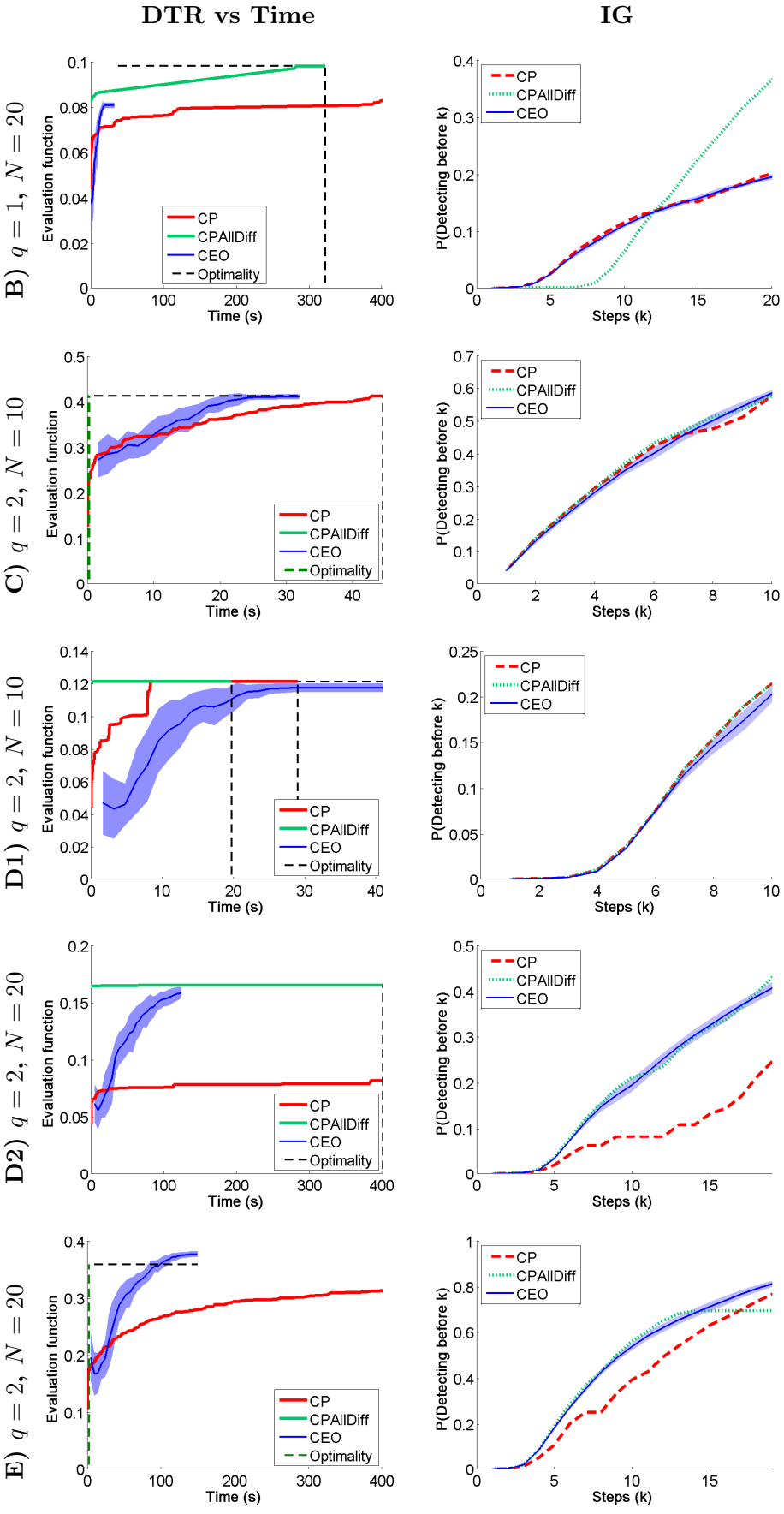


FIGURE 4.15: CP vs CEO.

Finally, the fifth row (E) shows an example where visiting repeated states provides better solutions. The results of the algorithms executed over scenario E show that CPAllDiff reaches its optimal solution really fast, but CEO beats it in both metrics because it allows repeated states. The quickly termination of CPAllDiff is due to: the power of the AllDifferent constraint propagation and the Gaussian distribution properties that make the upperbound to find the solution fast. Moreover, this scenario shows that CEO is a reliable approximated algorithm to solve the discrete MTS.

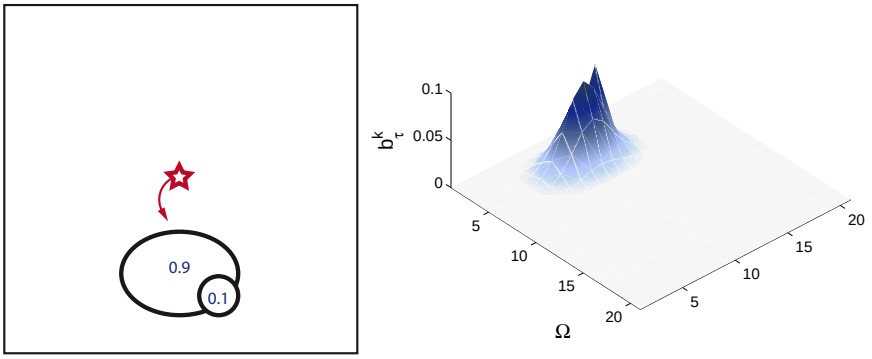
In conclusion we can state that CP is tractable with small number of variables and CPAllDiff is a fast way to provide near-optimal solutions. CPAllDiff achieves the optimum when the optimal solution does not visit repeated locations. In deed, CPAllDiff solutions can feed CP or other optimization methods to avoid its limitation when the optimal solution needs to visit repeated states. CEO has been showed to be a good and fast polynomial approximation with two main drawbacks: it can get stuck in a local optima and it has an offset with the optimal solution. Both drawbacks can be reduced by refinements like heuristics or improving the samples selection.

4.5.2 MTS Strategies vs Detection

As we have already explained, the majority of authors maximizes the probability of detection, but this does not necessary minimize the expected detection time. This section shows statistically that the proposed strategies fix this problem. Moreover, we show that when adding the heuristic to the uninformed utility functions, the time minimization objective is not affected. For this purpose, we compare the two strategies proposed (DTR and LET) and the detection baseline (D), and their heuristic versions for a fixed horizon N.

First we analyze deeply an illustrative example that shows clearly the difference between the strategies and afterwards we present the solutions over different scenarios to show the results tendency. For all scenarios we analyze the IG measure and we provide a dominance study. The IG study shows how fast the probability of detection is accumulated and gives us a way to compare all algorithms no matter which utility function is optimized (Figure 4.17(a)), as solutions with higher IG in previous steps have higher chances to detect the target sooner.

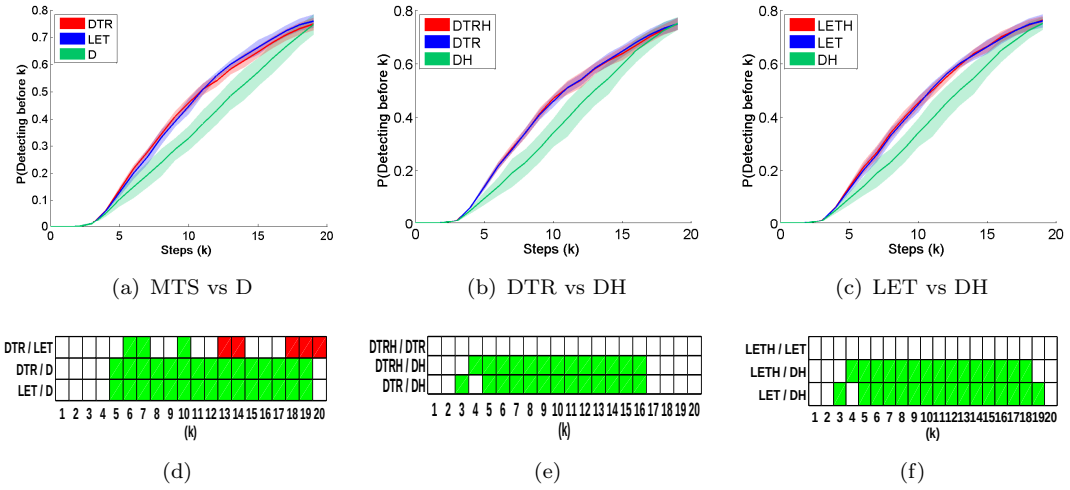
The illustrative example described in Figure 4.16, is used to show the advantages of using the proposed MTS strategies. The initial probability mass has high probability peak hidden inside a more spread probability region. With a long decision horizon (e.g $N = 20$), the agent is able to observe the whole probability region, and therefore, only strategies that visit the hidden peak before will reduce the detection time considerably. We will show that proposed MTS strategies are able to compute the right agent trajectory, because they take into account the locations visiting order. Figure 4.16(a) presents



(a) Scenario (b) b^k_τ

FIGURE 4.16: Illustrative example: scenario A.

the scenario schematically, where the agent (red star) starts in location $(10, 10)$ of the searching region and the initial target location belief is described by the circles, where the irregularity accumulates a probability of 0.1. Figure 4.16(b) shows the real initial belief represented by a height map.



(a) MTS vs D (b) DTR vs DH (c) LET vs DH

(d) (e) (f)

FIGURE 4.17: Information gain and dominance study.

Figure 4.17 shows a detailed study of scenario A (Figure 4.16). It is clear that the LET and the DTR strategies outperform D yielding to find the target in less time. In Figure 4.17(a), DTR (red line) and LET (blue line) gain faster the information as they have a higher IG value in early instants than using the D strategy. As the dominance study in Figure 4.17(d) reveals, DTR is better in the early instants and LET in the latest. In fact LET and DTR optimize different objectives: the average time and the discounted detection probability.

In case of using the heuristic strategies versions (DTRH, LETH and DH), as shown in Figure 4.17(b) and 4.17(c), we have the same behavior as using the non heuristic ones

(DTR, LET, D) because the impact of the heuristic appears when the team replans. Figure 4.17(b) shows that both, the DTR and the DTRH strategies, minimize the time more than DH and Figure 4.17(c) shows the same tendency in the results of LET, LETH and DH. It is also interesting to highlight that in some scenarios, the heuristic strategies will perform worse in the short term but later on, they will enhance the searching. A more complete analysis of the effects of the heuristic when replanning occurs is presented in Section 4.5.3.

Now we study the performance of the approaches over the set of dynamic target scenarios described in Figure 4.12, by showing the IG and dominance study. We use the dynamic target ones because their complexity is much higher than the static cases. In these experiments it has been stated that the agents are twice faster that the target, which means that the agent is able to observe two locations every time the target moves. In these experiments we have to take into account that due to the stochastic nature of the algorithm we are introducing disturbances that alters the studied measures. For that reason, the figures show the response of the tandem strategy-algorithm.

Figure 4.18 condenses all the results for the dynamic scenarios, each scenario in a row, the first column showing the IG performance and the second column representing the dominance study.

The first row presents the results for scenario F, where we have two agents with $N = 20$. The first figure on the left shows that at the instants 5, 6, and 7 the MTS (LET and DTR) strategies improves considerably the IG and therefore beats the D strategy in terms of detecting the target before. Afterwards, in later instants we can observe an undesired effect where LET and DTR have more troubles to accumulate more probability. The effort of acquiring information sooner can make more difficult to increment the probabilities in later instants, when the target location gets more uncertain (i.e. the probabilities are more spread in later instants in this scenario). This problem combined with the algorithm suboptimality (because it is harder to infer good actions when all have mostly the same reward), makes D dominate DTR in the later instants, although the difference is really small in terms of IG.

The second row, which analyzes scenario G with two agents and a decision window of $N = 10$, shows that in the first 6 instants there is no difference between strategies, because the agents are accumulating the tale of the probability mass. Once the team reaches the mass, DTR and LET have better behavior than D, because the rewards provided by the observations are time dependent. In this sense, LET achieves the best IG and even dominates DTR in instant 7 and 8.

The third row, where scenario H is optimized using 3 agents with a horizon $N = 15$, shows a drastic difference between the MTS strategies and D. This scenario is really bad for the D strategy because there are many solutions with high probability of detection at instant $k+N$, due to the probabilities condensation in a small region. Thus, the order of

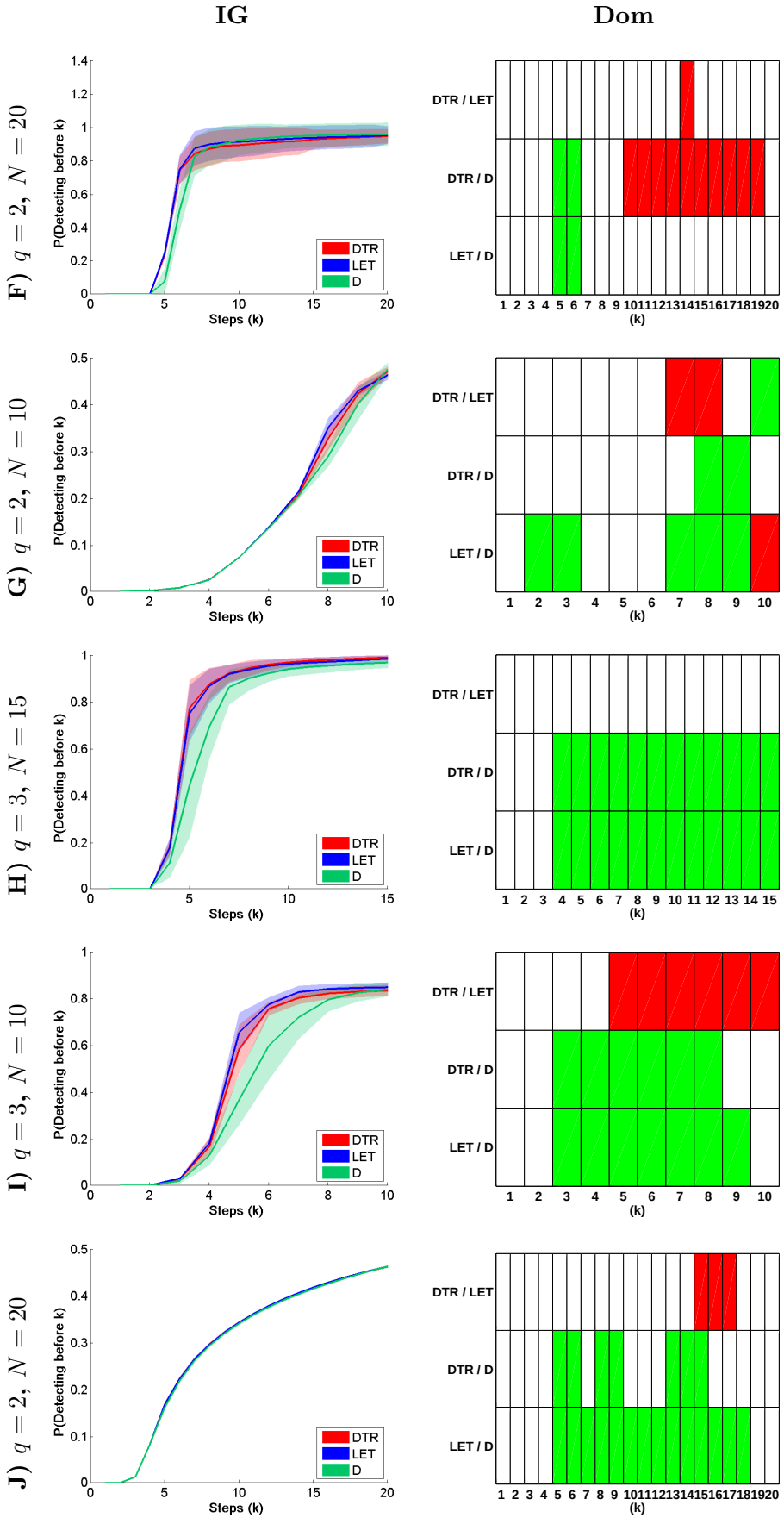


FIGURE 4.18: MTS strategies vs detection.

the observed locations makes the difference. Both, the IG measure and the dominance study, shows that LET and DTR reduce considerably the time to find the target.

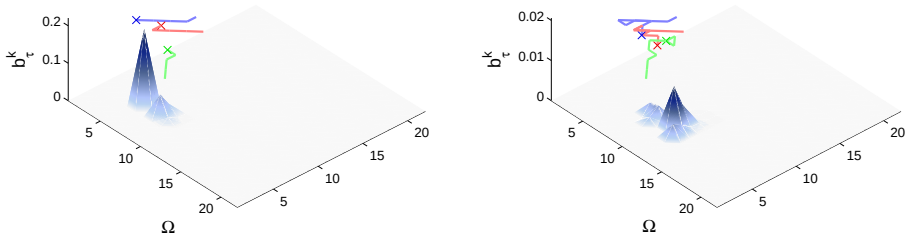
The fourth row, scenario I, is the most complex in terms of target dynamics because its model is driven by a wind map. In this case, a team of $q = 3$ agents computes a team plan of horizon $N = 10$. The IG figure shows a great improvement when using the proposed MTS strategies. In fact at instant 5, LET ameliorates the probability of detecting the target by 55% comparing with D (i.e. LET obtains 0.7 and D 0.38). This means that using LET we have 30% more chances to discover the target sooner than using D. The domination study shows that LET and DTR outperform D and also that LET behaves better than DTR.

The fifth row, scenario J with $q = 2$ and $N = 20$, shows the team response to a Gaussian distribution created by an uniform target dynamic model. We can see that this is a special case where the necessary condition to minimize the time by maximizing the probability of detection holds (Section 3.3). Thus, all strategies have the same time detection performance as it is shown in the IG plot. However, we can see at the domination plot, that LET and DTR dominates D, due to the distribution disturbances induced by the observations until the target moves.

In conclusion, this stochastic study corroborates the statements presented in Chapter 3. The results show that the Bayesian approach, in combination with the CEO algorithm, is able to tackle the multiagent MTS with dynamic targets. They also show that the proposed strategies (LET and DTR) improve previous ones (D) in terms of time detection optimization, and in some scenarios they outperform drastically the detection strategy. In the case that the necessary condition is satisfied, all strategies have a similar performance, thus, for scenarios where the target is modeled as a Gaussian distribution, using the detection/non-detection strategy is also a good choice. Finally we infer a slightly better response using LET than DTR.

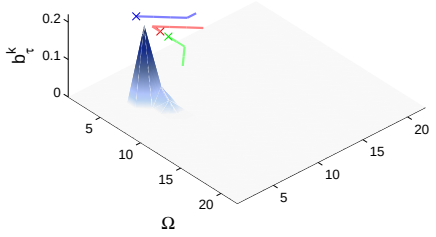
To visualize the differences between the three strategies in terms of team trajectory, we show in Figure 4.19 one simulation of scenario I (Figure 4.12) with a team of three agents. This simulation helps to understand the results provided in Figure 4.18 for scenario I. Figures 4.19(a), 4.19(c) and 4.19(e) show the agents after 5 steps of simulation for the DTR, LET and D strategies respectively. Figures 4.19(b), 4.19(d) and 4.19(f) show the agents after 10 steps. Note that the probability mass is moving as well as the agents are updating their belief. At instant $k = 5$ the target has moved twice and the agents start to observe locations with high probability. Note that in Figure 4.19(e), related with the D strategy, all the agents are looking really close locations waiting to accumulate all the probability, while using the MTS strategies they are already observing high probability locations. Finally, at instant $k = 10$ we can see in Figure 4.19(d) that LET has reduced high probability peaks while the other two strategies still have important regions to observe. Because LET and DTR care about when the high probability locations are

observed and D only takes into account the total information gained, LET and DTR are better strategies. In fact, we can see that while in LET and DTR the three agents cover a different region (i.e. they split the searching area), in D all agents go to the same region because they get a high reward.

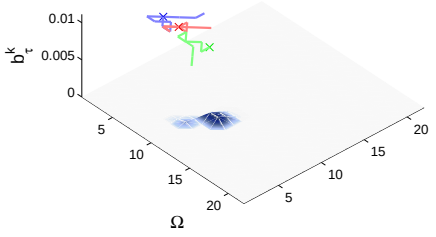


(a) DTR k=5

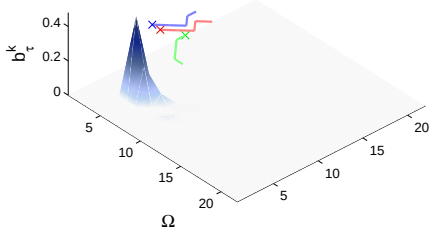
(b) DTR k=10



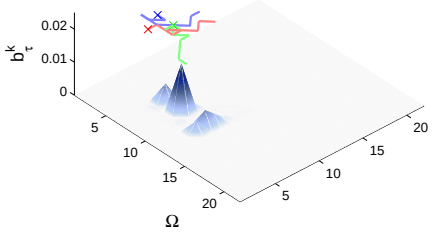
(c) LET $k = 5$



(d) LET $k = 10$



(e) D $k = 5$



(f) D $k = 10$

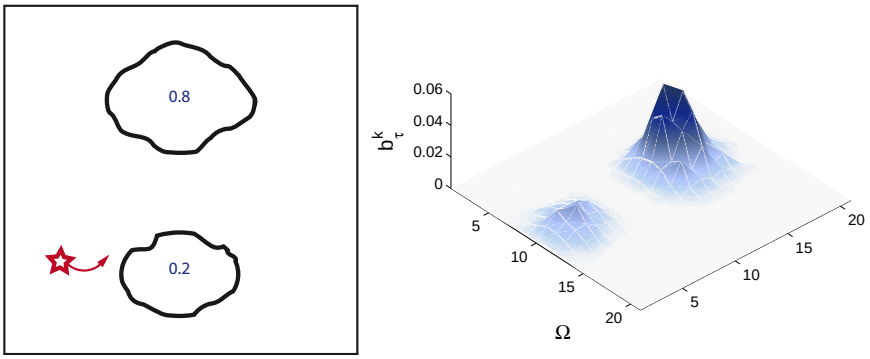
FIGURE 4.19: Simulations of scenario I with $q = 3$ and $N = 10$.

4.5.3 With Heuristic vs Without Heuristic

In this section we study the effect of using the DTH strategy proposed in Section 3.5.2.3 in terms of reducing the myopicity of the team decisions. When we only compute the decisions up to an horizon N, it is probable that we lose information about the future,

yielding to a myopic algorithm that can be trapped into local optima. Thus, we include a sensorial heuristic that mitigates this behavior.

The analysis is done by comparing LET and DTR strategies with and without the heuristic. The team computes its N joint actions using its current location and target belief. Afterwards, it executes the actions while taking measurements. Finally it replans again from the new agents location and target belief, which is calculated by using the agents measurements within the RBE information layer. The aim of this section is to show that algorithms with the proposed heuristic are less myopic: the agents visit early further away regions with high probability of detecting the target, and therefore reduce the detection time.

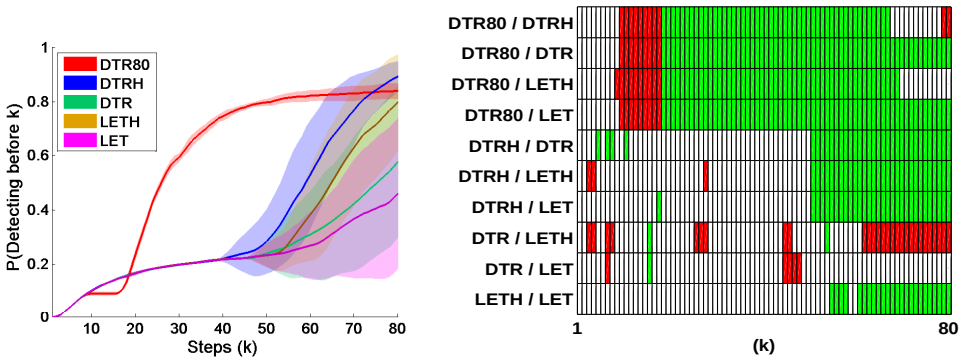


(a) Scenario (b) b^k_τ

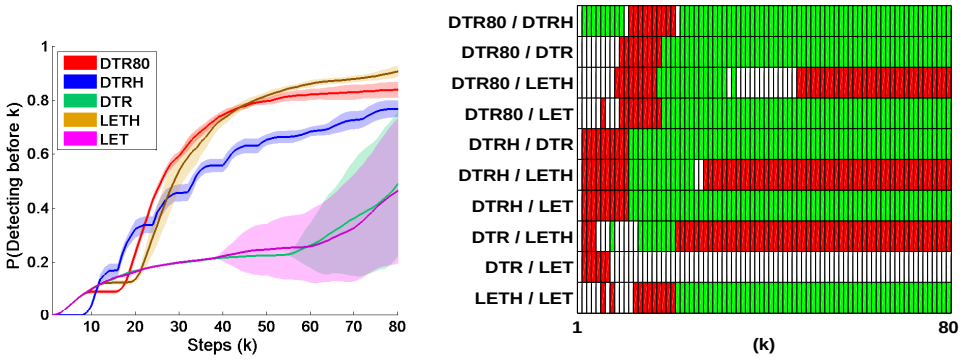
FIGURE 4.20: Illustrative example: scenario B.

We analyze the heuristic through a representative example where the locality of the algorithms without heuristic are trapped into a low probability area. The example showed in Figure 4.20(a) is an agent that has a low probability region really close and further a big probability mass. When we do not use the heuristic, the agent covers the closer region until it discovers the other region by chance. In the case that the other region is enough far, the agent never visits it. When the agents use the heuristic, further regions with high probability affect their behavior before and the agents avoid wandering over small probability masses. In this example, visiting the big probability mass sooner implies that the detection time is reduced. Figure 4.20(b) shows the real initial target location belief.

We compare the strategies DTR and LET and their respective heuristic versions (DTRH and LETH) with a decision horizon $N = 8$ executed 10 sequential times, with DTR computed for an horizon $N = 80$ (DTR80), which is the same problem where the actions are totally computed instead of obtained as a sequence of partial solutions. Because the optimization algorithm is stochastic we run 50 simulations of each strategy and then we compute the mean and the variance values.



(a) IG Hnorm (b) Dominance Hnorm



(c) IG H (d) Dominance H

FIGURE 4.21: With heuristic vs without heuristic for scenario B.

Figure 4.21 shows the analysis using the proposed heuristic where: in the first row, the heuristic is normalized (Hnorm, $\eta = 1/\sum_{s_i^{k+N}} \hat{H}(s_i^{k+N})$) and in the second row, the heuristic is not normalized (H, $\eta = 1$). Both results are provided to show that the heuristic impact is determinant to make future decisions.

The first study, in Figure 4.21(a), shows the strategies response for scenario B when using the normalized heuristic. The color code is the following: DTRH blue, LETH brown, DTR green, LET purple and DTR80 red. The lines are the mean IG value for all runs and the transparency region of the same color is the standard deviation. We can see that the strategies using the heuristic behave better that the ones without it. Still, DTR80 achieves the best IG because it visits the high probability region before. We can infer that the sensivity to the heuristic is higher in DTRH than in LETH, because DTRH leaves before the low probability region. In mean, DTRH in the step 80, has more than 80% chances to find the target while DTR and LET has just 50% and 40% respectively. In consequence the heuristic reduces the strategy myopicity. The dominance study of Figure 4.21(b) corroborates this analysis. We remind the color code: green means the first strategy wins the second one, red tells us that the second strategy wins the first one and white states that both strategies are stochastically similar. The dominance is showed for every step up to 80. It is clear that DTR80 beats the others except for the

first instants where the agent is traveling towards the high probability region (Figure 4.22(a)). It is also shown that strategies with the normalized heuristic are better in long term decisions due to the myopycity of the non heuristic utility functions.

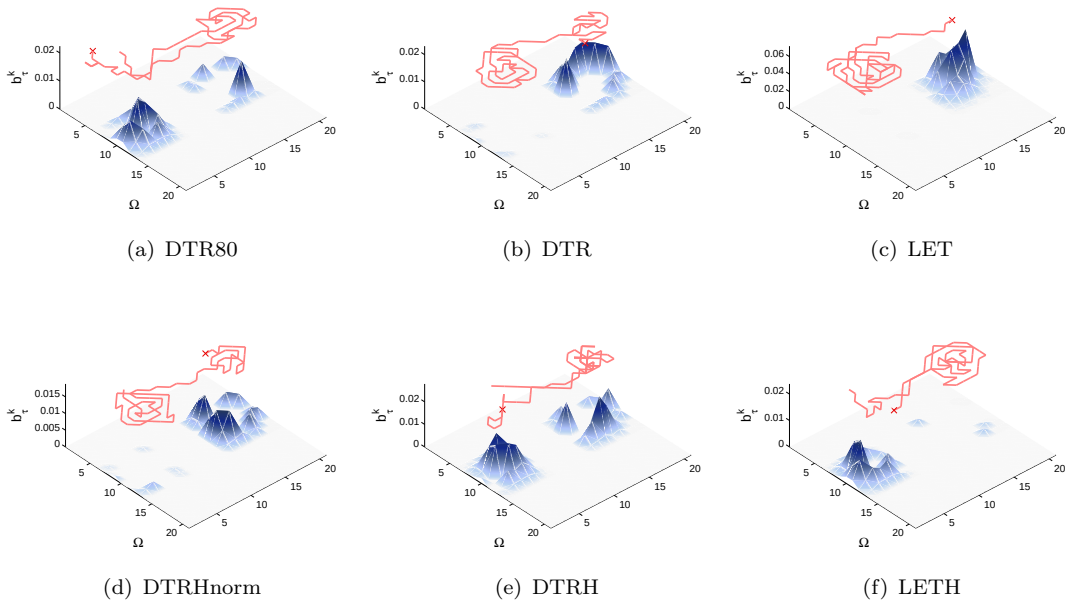
Figure 4.21(c) and 4.21(d), shows the comparison using the non-normalized heuristic (H). On one hand DTR and LET optimize locally the detection time and they do not take into account future decisions, achieving very bad solutions for a long term MTS task. On the other hand, DTRH and LETH drive the agent towards the high probability mass obtaining good solutions close to the DTR optimized for the whole path (DTR80). For instance, LETH achieves better probability increments than DTR80 from step 50 up to 80. Therefore, it is clear that the strategies that use the heuristic behave better than without it.

It is also important to note that DTRH is more sensitive to the heuristic that LETH because of the construction of the utility function. The agent using DTRH goes directly to the high probability mass and is blind to closer probability locations, while using LETH it goes directly to the high probability region passing through the low probability one. We can also see that DTRH suffers from the normalization of the belief at each new optimization (described as bumps in the blue line in Figure 4.21(c)). This analysis gives us the feeling that the heuristic should be tuned depending on the strategy and the searching region scale. In the case of the DTRH, the heuristic should conserve the proportions with its own discount parameter.

The domination study (Figure 4.21(d)) shows that all the strategies with H win the ones without it. Although at first instants, DTR and LET start accumulating faster the probability because they are observing the closer non null probability region, afterwards they get stuck. Meanwhile, their heuristic informed versions win the rest of the steps because they are observing the further high probability area. DTR80 is similar to LETH but wins DTRH, because of the heuristic sensitivity. Anyway, for a fast online algorithm we cannot compute the whole solution and the heuristic turns out as a good approximation.

Figure 4.22 shows simulated examples of the analyzed strategies. The examples describe visually the results of the statistical study (Figure 4.21). The agent following DTR80 visits first the high probability mass and afterwards it comes back to observe the low probability area.Using DTR and LET, the agent gets stuck observing the low probability region until, by chance, it discovers the further one. On the contrary, the heuristic strategies push the agent to visit the high probability mass before. Depending on the function sensibility to the heuristic, the agent is more or less impatient to travel towards the further region. DTRH is really sensitive, and therefore, the agent goes straight away to the high probability region, but suddenly, it comes back. This oscillation behavior appears when the normalization of the probability distribution increases the probability of unvisited locations and means that the heuristic contribution should be tuned.

The agent following the LETH strategy, due to its smaller sensibility, goes towards the high probability area traversing the low one that is revisited at the end when the high probability region is observed enough.



(a) DTR80 (b) DTR (c) LET

(d) DTRHnorm (e) DTRH (f) LETH

FIGURE 4.22: Simulations w heuristic vs w/o heuristic for scenario B.

In conclusion, using the heuristic we reduce the strategy myopiciy achieving better response in the future and the agents avoid getting stuck in local optima solutions. However, the heuristic should be tuned according to the scale of the region and the strategy used. The experiments shows that the DTR is more sensitive to the heuristic by construction, yielding to undesired behaviors when the normalization of the target location distribution takes place. To solve this issue, we make the discount factor fit the values proportions of the DTR function. LET is less sensitive to the heuristic achieving good behaviors when the heuristic value is high. In practice, using the heuristic is a nice and fast approximation for an online decision algorithm within the MTS problem.

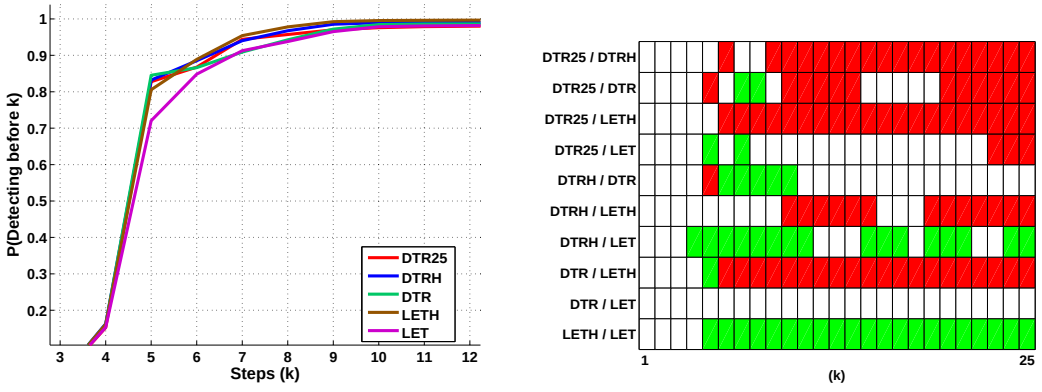
We also analyze, in Figure 4.23, the heuristic response in the dynamic target scenario I, where we have a wind map that describes the target movements and 3 searching agents. We show again the results for the LET and DTR versus their heuristic versions (LETH and DTRH) and the DTR that computes the whole decision horizon (DTR25). In this case each agent computes a decision of horizon $N = 5$, and we perform the search for 25 instants. We use as the heuristic the non-normalized sensor and we fix the sensibility of DTR to the heuristic by weighting it with λ^N.

First, we study the IG plot in Figure 4.23(a). This figure is zoomed to show the region of interest and visualize properly the differences between strategies, because after 12 instants there is no much difference between them. Besides, we are only showing

the mean IG values for each strategy because the standard deviations make the figure unintelligible and do not contribute with relevant information. However, the variances are indirectly contemplated in the dominance study (Figure 4.23(b)). We have to focus on the instant $N = 5$ because is when the first replanning takes place. We can see that LET is worse than the others and that DTR is the best one, with the rest really close. This a clear example of a local decision, where DTR puts too much effort in acquiring information in the first 5 instants but then it loses performance in the future. In fact LETH and DTRH have a lower performance at instant 5 because they infer the future, knowing that losing some information in early instants, they will later achieve better performance. In fact, after instant 5, DTRH and LETH outperform the rest of the strategies and even beat DTR25 because is harder to compute a good solution when the horizon is longer.

Secondly, we analyze the dominance results (Figure 4.23(b)). We can see that all the heuristic strategies win in the majority of the instants, and again LETH wins DTRH. We can also infer that DTR is a good approximation in this scenario because it wins DTR25. This means that computing the best solution using DTR for $N = 25$ is hard and that the normalization helps to give too much importance to the future rewards. However, we have to read this plot combined with the IG one, where we can see that when DTR dominates DTR25, the IG difference is really small, while in the instants in which DTR25 dominates DTR ($k = (7, 8)$) the IG difference is high, telling us that DTR25 is slightly better.

Studying the whole picture that both Figures, 4.23(a) and 4.23(b), proportionate, we can conclude that by using the DTH strategy combined with DTR and LET we obtain a really good performance at the MTS. We reduce the locality and we improve the computation time. Therefore, optimizing DTRH and LETH with CEO is a good solution for the controller layer when we have discrete agent actions.



(a) IG (b) Dominance

FIGURE 4.23: With heuristic vs without heuristic for the scenario I (dynamic target).

4.6 Summary

This chapter approaches the MTS as a discrete optimization problem, where the agent actions can get a finite number of values (e.g. the cardinal directions) and the searching region is modeled as a connected grid graph.

Two approaches are proposed to compute the best N joint team actions to search a no-evading target in the minimum possible time. The first one (CP-MTS), based on constraint programming, is only applicable for searching static targets with DTR and the second one, an stochastic sampling method (CEO-MTS), solves the general MTS for a dynamic target.

The CP solution uses the multiagent DTR strategy over a decision graph computing the accumulated discounted joint probability of detection for a fixed horizon. The approach is global optima within the horizon and tractable for small size problems and could be approximated for medium size problems. Its validity has been showed through simulation.

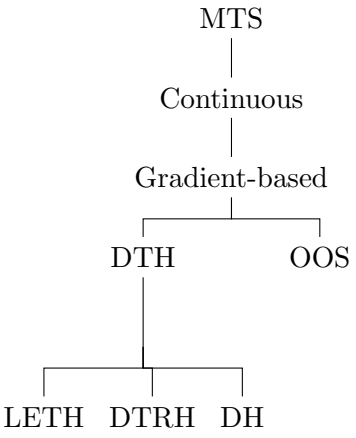
The second algorithm (CEO-MTS) accepts any kind of sensor model and any of the MTS strategies proposed in Chapter 3. The CEO method achieves tractable suboptimal solutions to search a dynamic target. For large MTS problems the computation complexity depends on the trade-off between the number of samples computed and the optimality needed.

A large and detailed analysis of the MTS strategies and the MTS solutions is also presented. The results show that CEO is a valid approximation for an online controller layer although it achieves suboptimal solutions. The experiments also corroborate that the proposed MTS strategies, DTR and LET, minimize the time to detect the target, improving previous approaches, and that the heuristic versions of these strategies reduce considerably the locality of the decisions. Besides, the combination of the recursive Bayesian estimator as the data fusion layer and the CEO-MTS as the controller layer has been showed as a good solution for the MTS autonomous system.

Chapter 5

CONTINUOUS APPROACH

"Information: the negative reciprocal value of probability" Claude Shannon



5.1 Contents

This chapter analyzes the continuous MTS, where the actions take real values and the target location distribution is represented as a Probability Density Function (PDF). We approximate the optimization by transforming the problem into a piecewise linear optimization, where the same control action is used during a constant interval of time. Thus, we have N observations and N control actions for each agent.

We provide two algorithms: 1) a combination of the gradient method with the Discounted Time Heuristic (DTH) strategies that reduces the target detection time and the locality of the decisions at the controller layer, improving previous works; and 2) an algorithm that improves the information layer to tackle the delayed observation measurements of the agents when the target location follows a Gaussian distribution.

The first approach, based on the works [Mathews et al., 2007; Gan and Sukkarieh, 2010], is fast, suitable for online execution, suboptimal and non-myopic. This solution uses a gradient-based method to compute the best team actions. In order to minimize the target detection time, this approach optimizes the heuristic informed strategies (DTH) presented in Section 3.5.2.3 and assumes that the target is static during the decision process [Mathews, 2008] due to the complexity of computing the Jacobian of the utility function when the target is moving. Initially we face the problem in centralized form showing how to solve the continuous MTS using all the proposed strategies. Afterwards, we explore the possibility of decentralizing the algorithm for one of the utility functions to develop an efficient searching autonomous system.

The second approach deals with a real problem that occurs in sensing networks: the delayed and disordered arrival of measurements to the data fusion center. This problem, known as the Out Of Sequence Problem (OOSP, [Bar-Shalom and Chen, 2005]) has to be especially handled by the Bayesian estimator and causes difficulties to the decision controller. However, the OOSP can be tackled by modifying the Bayesian filter. To do it, we model the target uncertain location as a Gaussian distribution, and modify the Extended Kalman Filter (EKF) to let it accept delayed measurements and achieve better updated information than in the case where the measurements are not available. In this case, we also use a gradient-based controller, but we also test two other strategies apart from DTH, which are useful for Gaussian representations: Maximum Slope (MS, Section 3.5.2.4) and Minimum Entropy (Section 3.5.2.4).

The chapter organization is as follows. After formalizing the continuous MTS and its framework, we describe the first algorithm proposal (Gradient-MTS) where we design the expected observation as a heuristic. Using this heuristic we build non-myopic utility functions to evaluate the agents actions and show how to solve this non-convex optimization with a gradient-based algorithm in centralized form. Afterwards we present the decentralized solution for one of the strategies. Finally we present the OOS-MTS

algorithm by explaining the OOS problem and the Gaussian distribution as the target representation. Besides, we provide an statistical analysis and simulations of the results obtained by both algorithms.

5.2 Modeling the Problem

Our multi-sensor coordinated search problem considers a team of mobile sensing platforms (e.g. a team of agents) capable of maneuvering freely and gathering information about the existence of targets within a defined work space. In this section, we start by introducing the dynamic and sensor models of the sensing platforms, and follow by describing the probabilistic representation of the target location belief.

We define the delimited mission search space where the agent and the targets are contained as $\Omega \subseteq \mathbb{R}^2$. Each sensing agent follows a trajectory generated by a receding horizon controller as an action vector v_i^k at each planning instant k. During every action planning process, the action vector of each sensor platform is jointly optimized to minimize the target detection time. As Figure 5.1 shows, we have an agent with an action vector composed of N piecewise linear actions $v_i^k = \{u_i^k, \ldots, u_i^{k+N-1}\}$, which makes the sensing platform state transit from s_i^k into $\{s_i^{k+1}, \ldots, s_i^{k+N}\}$ as time increases. To build the utility function we distinguish two rewards: the short term reward $J(v_i^k)$ and the future expected reward $\hat{H}(s_i^{k+N})$ (see Section 3.5.1). The short term reward is the reward obtained from the sensor observations at states s_i^{k+1} to s_i^{k+N} when the action vector v_i^k is applied. The future reward, described as the concentric circles in Figure 5.1, is an estimation of the possible cumulative rewards that are obtainable in the future from state s_i^{k+N} given that v_i^k has been executed.

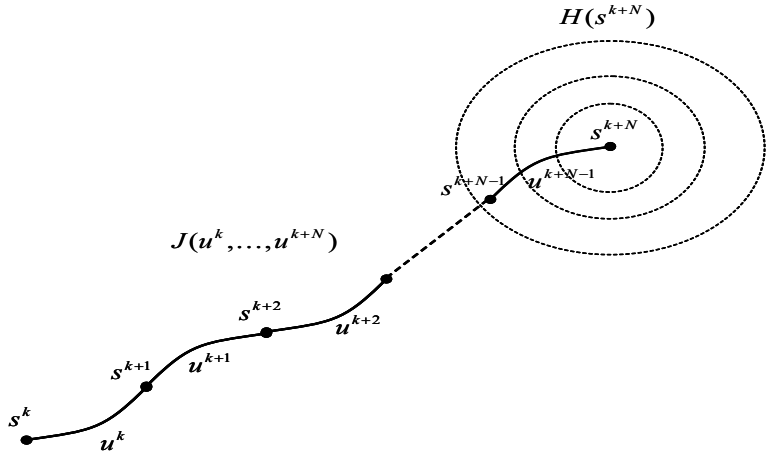


FIGURE 5.1: Continuous MTS searching trajectory generation using the DTH strategy. The solution is a sequence of actions v_i^k that drives the agent to find the target. With the expected observation $\hat{H}(s_i^{k+N})$, we estimate the goodness of the sensor platform being at state s_i^{k+N} according to some future expected detections.

5.2.1 Agent Motion Model

We assume that the agent has pitch and roll stabilization and that it operates at a constant altitude. The resulting simplified kinematic (Eq. 5.1) is a two dimensional constant velocity motion model where the state of the agent s_i is characterized solely by its position $p_i = [x_i, y_i]^T$ and the heading angle ψ_i. The agent is traveling at constant velocity V_i and its heading is controlled through an instantaneous turn rate command u_i. The kinematic equations are:

$$\dot{s_i} = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\psi}_i \end{bmatrix} = \begin{bmatrix} V_i \cos\psi_i \\ V_i \sin\psi_i \\ u_i \end{bmatrix} \tag{5.1}$$

Note that in this chapter we eliminate the concern of sensor state (location) uncertainty by employing the assumption that the state of each sensor platform is known through some high accuracy navigation module (e.g. GPS or IMU). We further assume that the sensor is attached at the platform center of mass thus the sensor state is the same as the agent state.

The forward states of applying an action u_i^k are computed recursively by integrating the differential kinematics over the time interval $\int \dot{s_i}\Delta k$ [Gan and Sukkarieh, 2010]:

$$s_i^{k+1} = \begin{bmatrix} x_i^k \\ y_i^k \\ \psi_i^k \end{bmatrix} + \begin{bmatrix} \big(sin(\psi_i^k + u_i^k\Delta k) - sin(\psi_i^k)\big) \cdot V_i/u_i^k \\ \big(-cos(\psi_i^k + u_i^k\Delta k) + cos(\psi_i^k)\big) \cdot V_i/u_i^k \\ u_i^k\Delta k \end{bmatrix} \tag{5.2}$$

5.2.2 Sensor Model

Each agent has a sensor that makes observations along the path. It should be noted that each agent can have a different sensor model, but to simplify the problem formulation, we have chosen the exponential sensor model for all the sensing platforms. This sensor model represents the likelihood of producing a target detection event D from the observation z_i^k at time step k, if the sensor is at state s_i^k and the *static* target[1] is placed at position τ. This sensor model (a.k.a. observation likelihood) can be defined as,

$$P(z_i^k = D|\tau, s_i^k) = P_{d_{max}} e^{-\sigma\left(\frac{\|\tau - p_i^k\|}{d_{max}}\right)^2} \tag{5.3}$$

where $\|\tau - p_i^k\|$ is the euclidean distance between the sensor i position $p_i^k = \left[x_i^k, y_i^k\right]^T$ and target position τ. This exponential model is shaped by the parameters: $P_{d_{max}}, \sigma$

[1] Note that as the target is static τ is without the superscript k.

and d_{max}. It is a smooth and differentiable function that decreases when the distance increases. The complement of Eq. 5.3 is the sensor non-detection likelihood,

$$P(z_i^k = \overline{D}|\tau, s_i^k) = 1 - P(z_i^k = D|\tau, s_i^k) \tag{5.4}$$

We remind that we use D_i^k and $\overline{D}_i^k$ to refer to $z_i^k = D$ and $z_i^k = \overline{D}$ respectively.

5.2.3 Target Belief

The target is assumed to be static in the field while its position, τ, is uncertain and can be represented as a belief $b_\tau^k = P(\tau|z_{1:q}^{1:k}, s_{1:q}^{1:k})$, describing the probability of the target being in a specific 2D position, conditioned on all the previous sensor observations $z_{1:q}^{1:k}$ made at sensor states $s_{1:q}^{1:k}$. We consider that the observations at each instant are always non–detection: $z_i^k = \overline{D}$. The initial target believe b_τ^0 is known a priori and it will be updated after every successive sensor observation using the update operation of the Recursive Bayesian Estimation (RBE, Algorithm 1), according to Eq. 5.5.

$$b_\tau^k = \frac{1}{\eta} \prod_{i=1}^{q} P\left(\overline{D}_i^k|\tau, s_i^k\right) b_\tau^{k-1} \tag{5.5}$$

There is no prediction step because the target is assumed to be static. To see the complete derivation of this filter refer to Section 2.4.3.1.

5.3 Continuous Approximated Solution: Gradient-Based Optimization (Gradient-MTS)

- *Model:* Continuous
- *Target Information Representation:* General PDF
- *Optimization:* Local
- *Horizon:* N + Heuristic
- *Sensor:* Differentiable
- *Target:* Static
- *Data Fusion Layer:* Decentralized / Centralized RBE
- *Multiagent controller:* Decentralized / Centralized Gradient-based

One of the most important aspects of a decision making algorithm is its ability to run in real-time. It is critical to balance the algorithm computational time and myopicity during the action optimization process. The myopic characteristic defines how far we see into the future to select the best decision. In discrete algorithms such as the one in Section 4.4, one possible approach is using a sampling-based coarse-grain resolution that trades off between action optimality and computational time. Another approach is the discrete algorithm in [Yang et al., 2002], where they introduce heuristics based on the discounted expected reward to reduce the action myopic disadvantage. Discrete optimization has the disadvantage that the decision variables cannot have continuous values, making the controller more complex to design when the input signals do not have the same granularity. In current state-of-the-art continuous action space optimization there are no algorithms that take into account future estimated rewards, thus the optimal actions are constrained by the length of the action horizon.

This section proposes a non-myopic solution for the continuous MTS problem when we can assume that the target is static during the decision process. Figure 5.2 shows the continuous MTS, where a team of two agents coordinate their trajectories, to find the target in minimum time. The subjective information about the targets location, described as a PDF and shown as a colored height map, is known a priori, and it is the information that the team uses to compute their joint action plan. This information or belief is updated every time an agent makes an observation, making the team estimate the location of the target and therefore helping the team to do more precise decisions.

This approach is developed as a real time receding horizon controller that optimizes the actions in the continuous space, taking into account the expected team observation. Using a receding horizon controller with no terminal cost [Gan and Sukkarieh, 2010; Mathews, 2008] when the horizon is too short, makes the decisions local, yielding to low performance in worst case situations. To fill this gap, we estimate the goodness of being at the future state arrived after executing those decisions, taking into account the rewards that can be obtained in the future from that state. This section exploits the concept of expected observation presented in Section 3.5.2.3, which is a heuristic that infers the amount of information about a target existence that the agents can possibly collect in the future. The heuristic proposed for the expected observation is designed as a modified sensor model easy to incorporate into the team utility function and to use with a gradient-based optimization algorithm to find the optimal solution. The algorithm provides less myopic solutions reducing the time to detect the target. Its good scalability makes it useful for large scale environments too.

We also extend this non-myopic approach for a decentralized multiagent configuration incorporating the advantages of the decentralized continuous optimization proposed in [Gan and Sukkarieh, 2010; Mathews et al., 2007; Bourgault et al., 2004]. Therefore, this work complements [Mathews, 2008; Gan and Sukkarieh, 2010] by presenting the

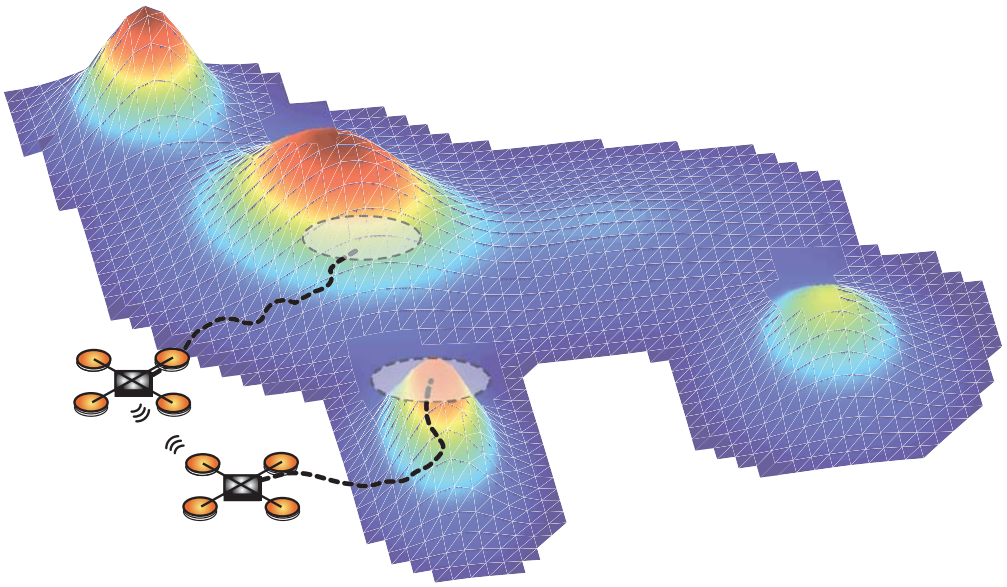


FIGURE 5.2: The continuous MTS coordinated search in a probabilistic scenario using a team of agents.

advantages of using the expected observation to eliminate the deficiencies that local optimizers have, and shows how to include it easily with a low computational cost.

The sections organization is as follows: section 5.3.1 presents the expected observation heuristic and the strategies in centralized form; section 5.3.2 describes the gradient-based optimization algorithm and section 5.3.3 details its extension to multi-agent decentralized coordination; finally, section 5.3.4 presents the strategies study within the continuous MTS and shows the improvements that our approach brings to complex probabilistic target configurations in large scale scenarios through the statistical analysis of simulated results.

5.3.1 Strategies with Expected Observation in Centralized Form

The team actions are computed using the MTS heuristic informed strategies presented in Section 3.5.2.3 for static targets. Therefore, in this section we summarize the DTH strategies (LETH, DTRH and NDH) simplified for the static case. First, we show the expected non-observation heuristic used to evaluate the future reward and afterwards we present the static target version of the strategies.

The expected non-observation heuristic $\overline{\hat{H}}(s_i^k)$ is an infinite range sensor, which is applied at the terminal agent state s_i^{k+N}. It is computed using the following expression:

$$\overline{\hat{H}}(s_i^k) = 1 - \eta\beta^{\|\tau - p_i^k\|/V_i} \tag{5.6}$$

where $\|\tau - p_i^k\|$ is the Euclidean distance from the sensor i to the target location, $0 < \beta \leq 1$ and η is the heuristic normalizing factor. Besides, the parameter β controls the

importance of the probability of finding the target at further regions from state s^{k+N} and belief b_τ^{k+N}.

Simplifying the LETH strategy for a static target we obtain the following utility function:

$$J_{LETH}(s_{1:q}^k, v_{1:q}^k, b_\tau^k) = \sum_{j=1}^{N} \int_\tau \prod_{l=1}^{j} \prod_{i=1}^{q} P(\overline{D}_i^{k+l}|\tau, s_i^{k+l}) b_\tau^k d\tau + \\ + \int_\tau \prod_{i=1}^{q} \overline{\hat{H}}(s_i^{k+N}) \prod_{l=1}^{N} \prod_{i=1}^{q} P(\overline{D}_i^{k+l}|\tau, s_i^{k+l}) b_\tau^k d\tau \tag{5.7}$$

In the DTRH strategy case we arrive to:

$$J_{DTRH}(s_{1:q}^k, v_{1:q}^k, b_\tau^k) = \sum_{j=1}^{N} \lambda^{j-1} \int_\tau \left[1 - \prod_{i=1}^{q} P(\overline{D}_i^{k+j}|\tau, s_i^{k+j})\right] \prod_{l=1}^{j-1} \prod_{i=1}^{q} P(\overline{D}_i^{k+l}|\tau, s_i^{k+l}) b_\tau^k d\tau + \\ + \lambda^{k+N} \int_\tau \left[1 - \prod_{i=1}^{q} \overline{\hat{H}}(s_i^{k+N})\right] \prod_{l=1}^{N} \prod_{i=1}^{q} P(\overline{D}_i^{k+l}|\tau, s_i^{k+l}) b_\tau^k d\tau \tag{5.8}$$

And finally using the Non-Detection heuristic strategy (NDH):

$$J_{NDH}(s_{1:q}^k, v_{1:q}^k, b_\tau^k) = \int_\tau \prod_{j=1}^{N} \prod_{i=1}^{q} P(\overline{D}_i^{k+j}|\tau, s_i^{k+j}) \prod_{i=1}^{q} \overline{\hat{H}}(s_i^{k+N}) b_\tau^k d\tau \tag{5.9}$$

It is important to highlight that when we refer to ND and NDH we are also talking about their complementary strategies Detection (D) and DH, because they provide the same response. It is also important to note that the non-heuristic utility functions can be obtained from the heuristic ones by eliminating the heuristic contribution. In LETH and DTRH we only have to omit the last sum that includes the heuristic and in DTH we have to rip out the product of the q heuristic sensors.

5.3.2 Gradient-based Optimization

The aim of the optimization process is to obtain the action vector that optimizes the utility function. Written in minimization form, the MTS decision solution is:

$$v_{1:q}^{k*} = \arg\min_{v_{1:q}^k} J(s_{1:q}^k, v_{1:q}^k, b_\tau^k) \tag{5.10}$$

The above optimization problem can be solved using a gradient-based approach, which only requires the partial derivatives of the utility function with respect to each action in

the action vector ($\frac{\partial J}{\partial v_{1:q}^k}$) and which iteratively updates the action vector with a steepest descent approach [Mathews et al., 2007; Gan and Sukkarieh, 2010].

Let $v_{1:q}^{k,t}$ be the actions vector of the agents team at instant k and t be the optimization iteration step, the solution improvement is given by the following expression:

$$v_{1:q}^{k,t+1} = v_{1:q}^{k,t} - \gamma \frac{\partial J}{\partial v_{1:q}^{k,t}} \tag{5.11}$$

where γ is the step size that tunes how fast the algorithm converges [Mathews et al., 2007].

As the utility function for this search problem is non-convex in nature, gradient-based approaches can only guarantee local optima solutions. The partial derivatives $\frac{\partial J}{\partial v_{1:q}^{k,t}}$ can be computed directly by using the explicit formulation previously developed in [Gan and Sukkarieh, 2010].

Gradient-based optimization requires an initial solution proposal. This initial action vector prior to the gradient-based optimization loop is obtained using a rapid sampling-based algorithm, in our case, the Rapidly Random Tree (RRT) algorithm [Lavalle, 1998; Yang et al., 2010].

5.3.3 Decentralized Multiagent Extension through Coordination

In this section we focus on coordination by negotiation using the NDH utility function (Eq. 5.9). NDH is chosen for four reasons: it permits anonymous coordination because the agents do not need to know the sensor model neither the dynamics of the rest of the team; it is easy to implement; there is a convergence proof [Mathews et al., 2007]; and the results show that for small decision horizons it obtains good search performance.

To decentralize the decision process we consider that every agent has a controller layer able to communicate with the rest of the team. The initial target location belief b_τ^k have to be the same for each controller to start the optimization from the same initial conditions. Thus, we assume that there is a Decentralized Data Fusion (DDF) layer (Section 2.4.3.4) that synchronizes the target belief at every agent, such that ${}^i b_\tau^k = {}^j b_\tau^k = b_\tau^k \; \forall \; \{i, j\} \in (1, q)$. Besides, we also assume that the sensor observations of each individual agent are independent.

Each controller optimizes iteratively, by means of the gradient-based method, the individual agent utility function $J(s_i^k, v_i^k, b_\tau^k)$ to obtain the best actions vector v_i^k. Within the optimization, the controller uses the communicated information from the team, which is the target non-detection likelihood. At each iteration, the agent sends to the team the non-detection likelihood that it would obtain if it performs v_i^k. This communicated

information is fused by the other controllers to recompute their actions vector solution. After some iterations this asynchronous negotiation process converges. Therefore, the coordination is done by sharing the likelihood of non-detecting the target within the team.

Because each agent communicates their non-detection likelihood to other agents in the network, the NDH utility function (Eq. 5.9) for a single agent i in decentralized form is as follows:

$$J_{NDH}(s_i^k, v_i^k, b_\tau^k) = \int_\tau \prod_{j=1}^{N} P(\overline{D}_i^{k+j}|\tau, s_i^{k+1:k+N})\overline{\hat{H}}(s_i^{k+N})\,{}^i\tilde{b}_\tau^k \, d\tau = \tag{5.12}$$

$$= \int_\tau P_i^k \overline{\hat{H}}(s_i^{k+N})\,{}^i\tilde{b}_\tau^k \, d\tau \tag{5.13}$$

where

$${}^i\tilde{b}_\tau^k = \prod_{j=1,j\neq i}^{q} \prod_{l=1}^{N} P(\overline{D}_j^{k+l}|\tau, s_j^{k+l}) \prod_{j=1,j\neq i}^{q} \overline{\hat{H}}(s_j^{k+N})\; b_\tau^k = \tag{5.14}$$

$$= \prod_{j=1,j\neq i}^{q} P_j^k \prod_{j=1,j\neq i}^{q} \overline{\hat{H}}(s_j^{k+N})\; b_\tau^k \tag{5.15}$$

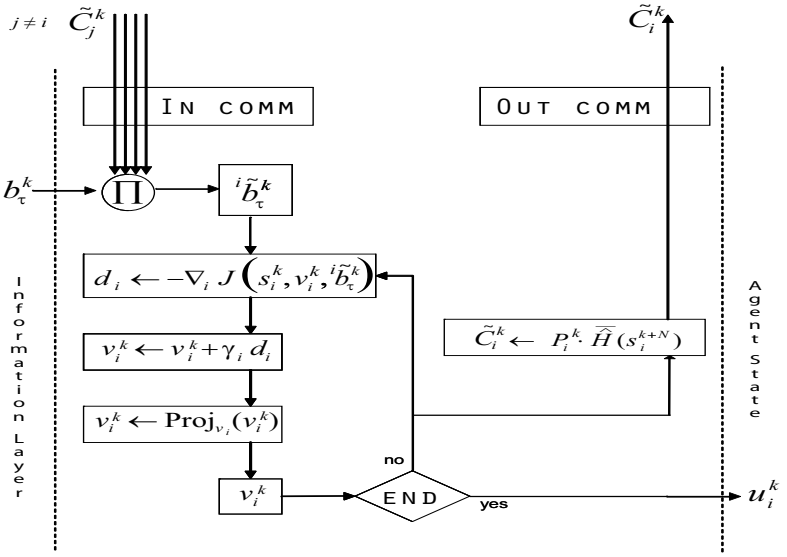
is the belief of agent i modified by communicated information $P_j^k \triangleq \prod_{l=1}^{N} P(\overline{D}_j^{k+l}|\tau, s_j^{k+l})$ and $\overline{\hat{H}}(s_j^{k+N})$ from other agent $\{j \in (1,q), j \neq i\}$.

Using the DDF as the information layer, the team has the target location belief (b_τ^k) synchronized. Thus, each agent i communicates at each optimization iteration the following data:

$$\tilde{C}_i^k = P_i^k \overline{\hat{H}}(s_i^{k+N}) \tag{5.16}$$

Each agent is in charge of fusing the information to construct ${}^i\tilde{b}_\tau^k$. In practice, this operation is done by multiplying the team communicated information $\tilde{C}_j^k$ and the target location belief b_τ^k, provided by the information layer.

The decentralized receding horizon controller with expected observation is explained in Algorithm 9. This algorithm is based on the work presented in [Mathews et al., 2007] modified to achieve the inclusion of the expected observation. It is worthy to say that the mentioned figure describes the controller layer of an individual agent. As we can see, it is connected internally with the information layer and the agent state, and externally with the team by communication. The algorithm flow is as follows:



ALGORITHM 9: Decentralized receding horizon controller with expected observation algorithm.

1. The initial target location belief b_τ^k is modified by the non-detection likelihood received from the rest of the agents (IN COMM) generating ${}^i\tilde{b}_\tau^k$.

2. The actions vector v_i^k is computed using the gradient-based method. The step size γ should be calculated using [Mathews et al., 2007] approximation.

3. If the algorithm has converged, the controller sends the optimized actions to be performed by the agent. Otherwise, it communicates (OUT COMM) the non-detection likelihood that embeds the expected non-observation heuristic ($\tilde{C}_i^k$) and continues improving the solution.

The proposed decentralized MTS solution integrates the non-detection joint probability with the expected non-observation. It takes into account future rewards, reduces the drawback of a finite lookahead action horizon, and reduces the time to find the target. The expected observation heuristic is embedded inside the team utility, by computing the joint team expected observation, reducing the complexity of other methods and maintaining a similar computational time.

5.3.4 Results

In this section we study the MTS from the continuous point of view. First of all we analyze the decision algorithm exhaustively as a centralized optimization in two ways (Section 5.3.4.1): 1) we show how the strategies proposed (DTR and LET) improve the minimum non-detection one (ND) [Eagle, 1984; Bourgault et al., 2003, 2004; Mathews, 2008; Gan and Sukkarieh, 2010] in terms of minimizing the time to detect the target; 2) we study the impact of using the heuristic (DTH strategy) at the myopicity of the

algorithm. Afterwards we show the MTS system working in decentralized form and compare it with previous works as [Mathews et al., 2007; Gan and Sukkarieh, 2010] (Section 5.3.4.2).

We remind that, due to the complexity to compute the Jacobian of the utility function for a dynamic target and in order to compare our proposal with previous methods, the target is assumed to be static during the planning horizon. Moreover, the heuristic within the DTH strategy is also defined for an static target.

5.3.4.1 Centralized Optimization Study

This study shows the advantages of using the Gradient algorithm (Section 5.3.2) combined with the MTS strategies proposed in terms of minimizing the time to detect the target. The optimization is done in a centralized manner, where all the decision variables are computed in the same processor.

We focus on the quality of the actions computed using each strategy. The quality is measured using the Information Gain (IG), which measures the theoretical probability of detecting the target at instant k. In practice the IG is the cumulative probability of detection obtained by the agents during the decision horizon. Plotting the IG measure for each k is a nice way to visualize the expected time of detection because higher values of IG in previous instants means finding the target before.

Because the gradient algorithm is sensitive to the parameters initialization we fix them for all experiments, using the following values: sensor ($dmax = 1$, $\sigma = 0.7$, and $Pdmax = 0.8$), heuristic ($\beta = 0.8$ and $\eta = 1$), initial agent actions ($v_i^k = \mathbf{1} \cdot 0.01$), and algorithm iterations ($N_I = 5000$).

The results are organized in two parts. In the first part, we show that DTR and LET reduce the target detection time more than ND. In the second part, we show how using the DTH strategy we enhance the search by reducing the myopicity of the decisions.

MTS strategies vs Non-Detection In this study we show that the theoretical results of Section 3.3 to minimize the detection time using the MTS strategies also apply to the continuous MTS. For that purpose we compare the strategies presented in Section 5.3.1 without using the heuristic term: LET, DTR and Non-Detection (ND). The utility functions are optimized with the Gradient-based algorithm described in Section 5.3.2.

The study is realized over five representative scenarios described in the first column of Figure 5.3. For each scenario we compute the best team actions using each strategy and then we calculate their IG (second column of Figure 5.3). In Figure 5.3 each row presents an scenario and its respective IG plot. The labels that appears at the left of the scenarios define the number of agents (q) and the optimization horizon (N). As

in the previous chapters, the scenarios are described schematically, where the red stars define the agents initial position, the red arrows the initial agents direction and the black contours the regions with non zero probability. The elliptical contours represent Gaussian distributions and the irregular shaped ones are generated by first assigning to their central location a fixed probability value and by next applying to this probability four random transitions. Note that the IG plots do not show the variances as in previous chapters results because the optimization is deterministic.

We also provide the visualization of the team trajectory (Figure 5.4) for each strategy and scenario to understand the IG results properly. Note that the simulation plots in this study only show the planned trajectory over the initial target location belief.

For the right comprehension of the study we are going to analyze each scenario separately linking the IG results of Figure 5.3 with the visualization of the team trajectory shown in Figure 5.4.

Scenario A The first row of Figure 5.3 shows the IG obtained by one agent with a decision horizon of $N = 20$ for the three strategies. In this scenario, the initial target location distribution hides a probability peak of 0.1. The IG plot shows clearly that both DTR and LET improve ND, achieving an increment of 10% at the chances to find the target in some instants (e.g. $k = 11$). The first row of Figure 5.4 explains this improvement: the agent trajectory following the actions calculated with LET and DTR visit the peak sooner than when using ND because they take into account the locations observation order.

Scenario B The second row of Figure 5.3 shows an scenario where the agents could get trapped in a local minima because there is a low probability region close to them. The team is compound by two agents with a decision horizon of $N = 20$. In this case the IG plot shows that all strategies get trapped due to myopicity issues because the accumulated probability is really low. The second row of Figure 5.4 confirms this behavior where the team only observes the low probability area. Anyway, DTR and LET reduce more the expected time than ND. We also see at the IG plot that, at instant $k = 20$, the DTR strategy loses performance compared to the rest of the strategies. This happens because, due to the DTR construction, later decisions have small rewards and therefore the gradient optimization gets more complex. Moreover, in Figure 5.4 we appreciate that following the actions produced by DTR and LET yields to similar trajectories.

Scenario C The third row of Figure 5.3 shows a non-Gaussian target location distribution where two agents start in the middle of the probability mass. The decision horizon is fixed to $N = 20$. Because there are many possible trajectories that accumulate much probability, ND strategy does not minimize the time, while DTR and LET do it. Again we can see similar trajectories using DTR and LET. In

comparison to ND, they are more spread and there is no trajectory crossing (i.e. the agents do no visit the same locations).

Scenario D The fourth row of Figure 5.3 shows a prior target location compounded by multiple disjoint Gaussian distributions. In this case there are three agents with a decision horizon of $N = 20$. The IG figure shows that DTR beats LET and LET beats ND. In this case, DTR takes advantage of the algorithm initial conditions. Figure 5.4 shows that DTR is better than the others because the agent represented by the red line visits a peak that is not observed with the other strategies.

Scenario E The last row of Figure 5.3 presents an scenario with b_τ^0 represented by a Gaussian distribution centered in the searching region. In this case the actions are computed for four agents that start in different locations with decision horizon $N = 15$. Theoretically, strategy ND would be a good approximation for this problem, but due to information coupling, the agents dynamics and the suboptimality makes DTR and LET obtain again better results in terms of time minimization. It seems that for a long term simulation LET and ND will have good performance because in the last row of Figure 5.4 the agents are turning around.

In conclusion, the MTS strategies have greater reduction of the target detection time than the non-detection one. Moreover, all strategies present locality problems. The suboptimality of the Gradient-based optimization interferes with the quality of the team behavior producing, in some scenarios, wrong trajectories. For instance, in scenario B, using DTR, further actions are not completely optimized. Also, in scenario D, LET and ND do not achieve the optimal solution. The results also show that DTR and LET, in spite of optimizing different functions, usually have similar behaviors. Therefore, both strategies are suitable to solve the MTS problem.

With Heuristic vs Without Heuristic In this study we show the advantages of using the DTH strategy combined with local strategies as DTR, LET or ND to achieve better performance in the search and to reduce the target detection time. Besides, it shows how the heuristic strategies reduce the myopicity. Moreover, they constitute a good approximation to solve large scale problems by using short horizons when the computation power is constrained. Finally, we also analyze the computational time required to obtain the best solution with the gradient-based optimization.

To perform the study, we use a decision horizon of $N = 8$ to force a high myopicity at the strategies. With this horizon MTS local strategies like DTR and LET do not have enough information to minimize the time correctly and therefore their behavior is close to ND's.

Figure 5.5 shows the IG analysis for the scenarios represented in the first column of Figure 5.3, which have already been used for the previous study of Section 5.3.4.1.

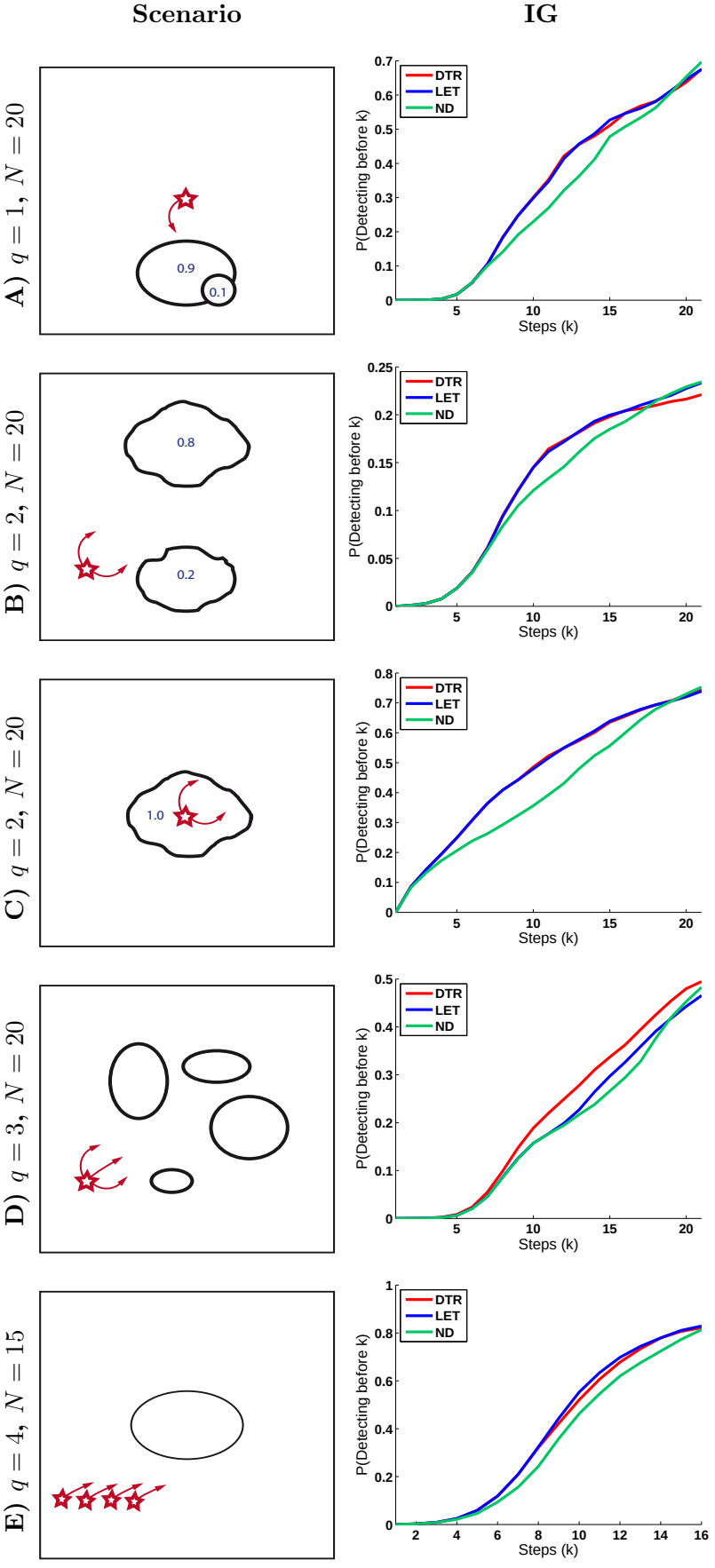


FIGURE 5.3: MTS strategies vs detection.

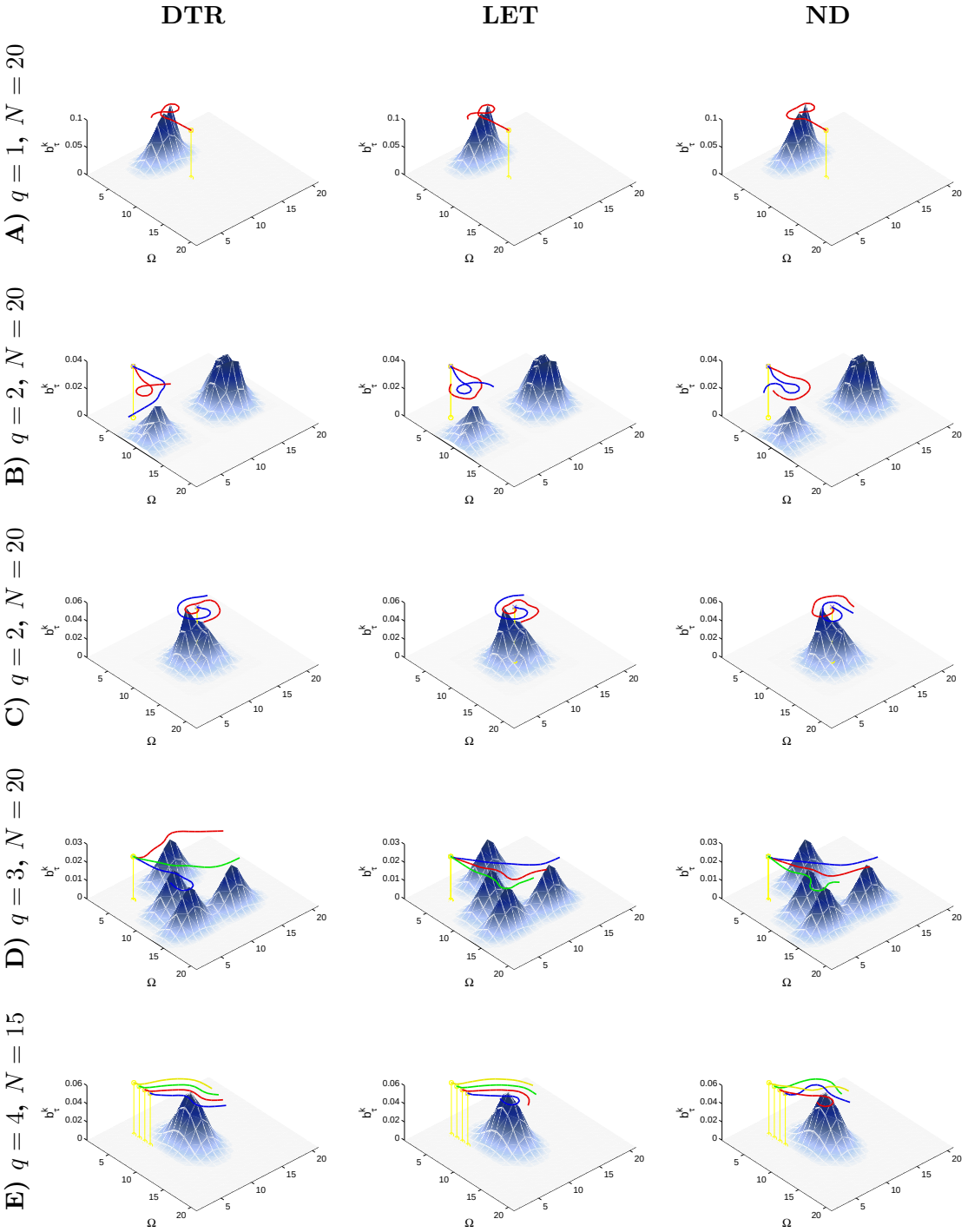


FIGURE 5.4: Team computed trajectory for each scenario.

The title of each graph shows the number of agents q, the decision horizon N and the number of sequential executions N_E. In this study we compute the team actions using the following strategies:

- The whole solution: LET40 (LET ALL). We compute LET for a decision horizon of $N = 40$ to compare with the heuristic strategies.
- Heuristic informed strategies: DTRH (Eq. 5.8), LETH (Eq. 5.7) and NDH (Eq. 5.9) with decision horizon of $N = 8$ executed $N_E = 5$ times sequentially (i.e. during $8 \cdot 5 = 40$ simulations instants).
- Local strategies: DTR, LET and ND (the previous utility functions without the heuristic), with decision horizon $N = 8$ executed $N_E = 5$ times.

Figure 5.6 and 5.7 show the team trajectory simulation using all the strategies for the most representative scenarios: B, D, E. In this study, the height map shown at each plot is the updated target belief with the agents observations. The labels on the left side of the figure display the strategy used and labels on the top of the figure define the scenario. Thus, each column contains a scenario and each row shows the solution for a different strategy.

To explain the results for each scenario we focus on the IG plots (Figure 5.5) and we only refer to the simulated graphics (Figure 5.5) when there is something useful to highlight.

Scenario A Figure 5.5(a) shows that all trajectories have similar behaviors although LET40 (LET ALL) is the only strategy that visits the hidden peak before, yielding to the best performance in all instants. The heuristic strategies also help the team to increment the IG but not enough to beat LET40. The non heuristic utility functions have an slightly worse performance until they observe the high probability peaks, where they catch the other strategies. We can see that LET and DTR have lost their ability to reduce the time due to the small horizon.

Scenario B Figure 5.5(b) describes the most illustrative example to test the heuristic. The team trajectory for LET40 (LET ALL), DTRH, DTR, LETH and LET is displayed at the first column of Figure 5.6 and NDH and ND are shown in Figure 5.7. The IG plot is totally related to the team trajectory, in fact, we can see that all the heuristic strategies visit both probability regions while the local ones just visit the closer probability area. This yields into a greater time detection reduction when using DTRH, LETH and NDH. It is also interesting to analyze the sensibility to the heuristic contribution that each utility function have. In this case, NDH has the fastest reaction and one of the agents goes straight to the further region. To tune correctly the heuristic within the NDH function, the heuristic sensor should be normalized. LETH appears as the less sensitive strategy and therefore, the

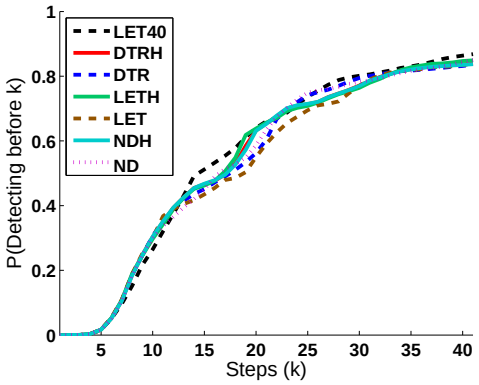
team visits the high probability region later than NDH and DTRH. It is clear that the local strategies like DTR, LET and ND, get trapped due myopicity and only their heuristic versions tackle correctly this scenario.

Scenario C Figure 5.5(c) shows an example where the strategies give too much importance to the heuristic. We have already shown that NDH and DTRH are highly affected by the expected observation (heuristic) and in this case, due to the small horizon, the team behaves too greedy. LETH, on the contrary, have the best performance achieving even better performance than the total horizon optimization (LET40). According to the decision problem complexity, as the number of agents increases, LET40 gets harder to optimize. Analyzing the local strategies, DTR is the best one followed closely by ND and the worst performance is given by LET.

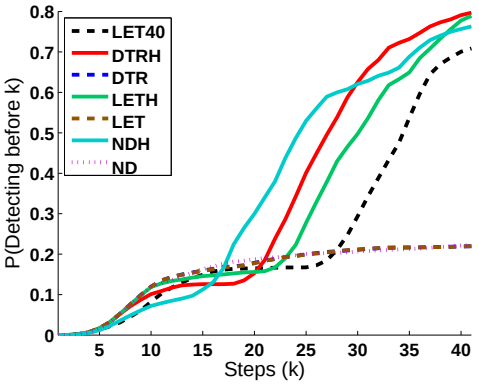
Scenario D Figure 5.5(d) shows the IG plot with separated probability regions. To see the team trajectory for each strategy please refer to the second column of Figures 5.6 and 5.7. Again we observe that the heuristic strategies are able to reduce the time considerably compare to the local ones. In this case DTRH achieves the best performance because it spreads the agents to observe all probability regions. NDH oscillates due to its sensibility to the heuristic sensor but still gets a really good performance. LETH, due to its conservative nature, is a bit slower to visit all regions. Anyway, the local strategies are worse than the others. Besides, the gradient method is not able to optimize LET40 with 5000 iterations. This is visualized in Figure 5.6 at the LET ALL row, where the agents trajectory goes away from the searching region.

Scenario E In the last scenario, shown in Figure 5.5(e), due to the small horizon and the properties of the Gaussian distribution, we have similar IG results for all strategies. It is interesting to highlight the good performance achieved by ND strategy in these types of scenarios. In fact, in the continuous MTS problem with Gaussian target location distribution and using a small decision horizon, ND appears as a very good greedy approximation. Anyway, the best performance is obtained by DTRH and LETH.

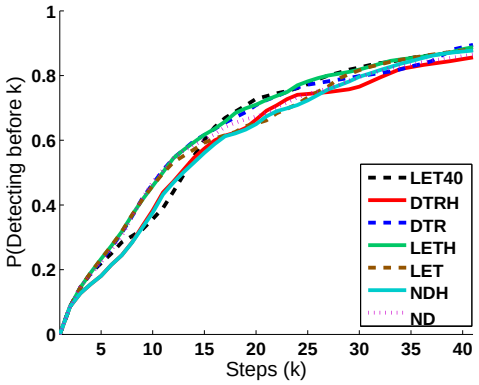
We also want to analyze specifically the team behavior difference between using the local non-detection (ND) strategy used in the literature comparing with its heuristic version (NDH). For that purpose we present in Figure 5.7 the trajectories computed with ND, NDH and ND ALL (i.e. ND strategy computed for the whole horizon $N = 40$). Regarding the trajectories and the IG results presented in Figure 5.5 we want to infer if using the DTH strategy is better than using ND. Both, the IG plots and the trajectories simulations show that using the heuristic within ND reduces the time to find the target because the solution is less myopic. The first column of Figure 5.7 shows clearly that NDH beats ND. Because the inclusion of the heuristic does not significantly increase the computational cost of the optimization and conserves the same properties as the



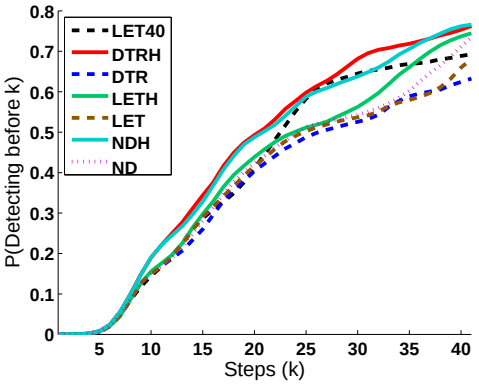
(a) A. $q = 1$, $N = 8$, $NE = 5$



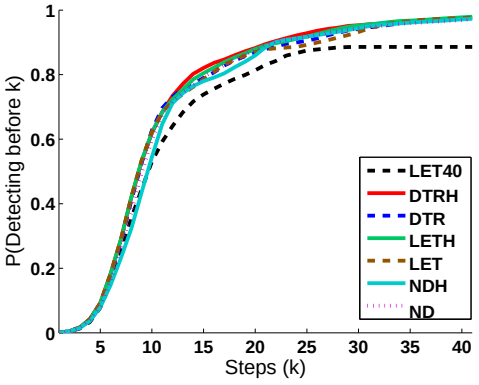
(b) B. $q = 2$, $N = 8$, $NE = 5$



(c) C. $q = 2$, $N = 8$, $NE = 5$



(d) D. $q = 3$, $N = 8$, $NE = 5$



(e) E. $q = 4$, $N = 8$, $NE = 5$

FIGURE 5.5: Strategies with heuristic vs without heuristic.

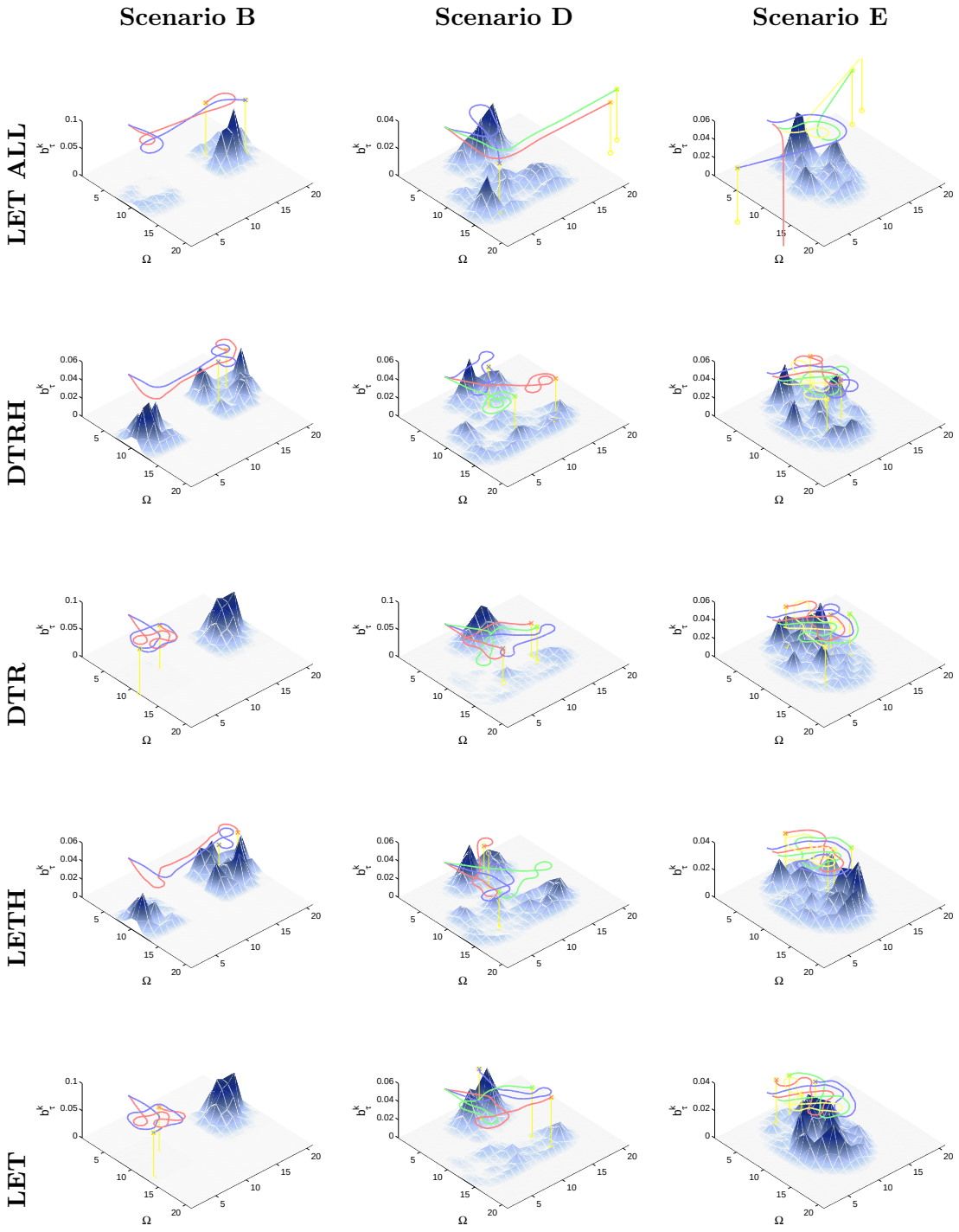


FIGURE 5.6: w H vs w/o H team search simulation.

non-detection utility function is a good choice for designing the decentralized controller, as we will see in the next result section.

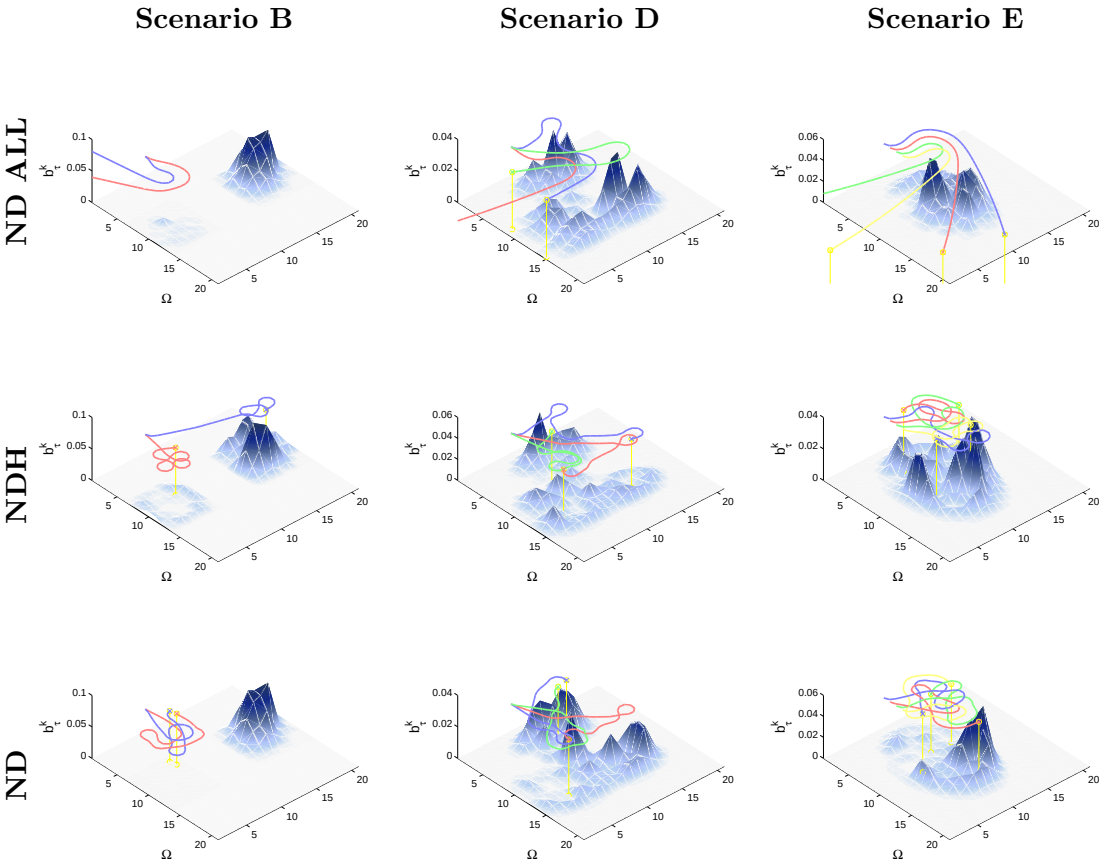


FIGURE 5.7: DTH strategy vs ND team search simulation.

Finally, we study the computation time that the Gradient-MTS using the NDH strategy takes to plan the best actions. We run the algorithm under Matlab environment on a Core 2 Duo 2 Ghz CPU for different decision horizon lengths. The number of agents is fixed to $q = 5$ and the initial agents location and target location distribution are the same as in scenario A (see first row of Figure 5.3). Figure 5.8 shows the computation time in seconds that the algorithm requires to calculate the agents actions for the horizons $N = (5, \cdots, 30)$. We observe that the time is increased polynomially with the decision horizon. The peak at $N = 17$ occurs because the complexity of the problem depends on the scenario and the horizon.

Summarizing the results we infer that:

- The heuristic strategies versions (DTRH, LETH, NDH) improve the long term team performance avoiding the myopicity of the non heuristically informed strategies, being a nice approximation when the horizon should be small due to tractability issues. The scenarios where the heuristic improves the search considerably are the ones that have disjointed probability regions.

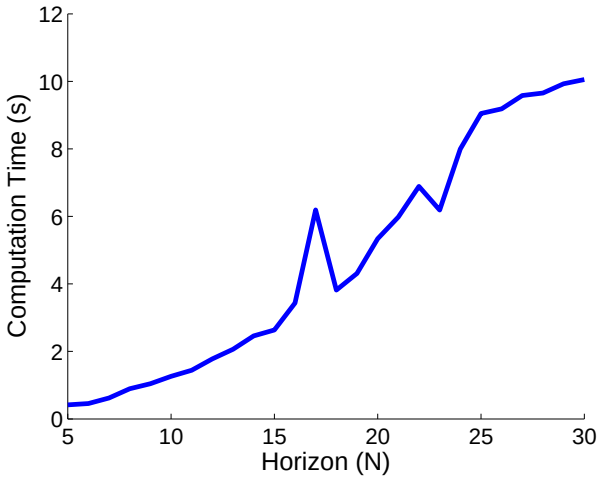


FIGURE 5.8: Computation time of the Gradient-MTS using NDH.

- For a small horizon, LET and DTR do not have a great impact in the time reduction because there is not enough information. Thus, they obtain a similar behavior than using ND.

- When using the heuristic, DTRH and NDH have similar performance due to the small horizon and although, LET and LETH get better performance with long decision horizons, with small ones as $N = 8$ LET and LETH have slightly worse results.

- Analyzing the utility function sensibility to the heuristic we can order the strategies from higher to smaller sensibility. DTRH is the most sensitive one because the obtained rewards at each instant are discounted by the time. To homogenize the rewards contribution and solve the excessive sensibility, we weight the heuristic reward with λ^{k+N}. NDH is the next strategy more sensitive because the total contribution of the observations is at most 1. Thus, any heuristic contribution that rises that value makes oscillate the team trajectory. We can solve this discrepancy by normalizing the heuristic sensor. LETH is the less sensitive strategy due to its construction, that makes the heuristic contribution really small compare to the total reward within the decision horizon. This converts LETH into a more robust strategy for dense regions but it loses performance when it has to visit spatially separated probability regions.

- ND seems to work quite well at the continuous MTS when the decision horizon must be small due to tractability issues, and when the target location distribution has a Gaussian shape.

- NDH is a suboptimal strategy with good results in time reduction.

- The selection of the decision horizon for the MTS problem should take into account the tractability (computation time) and complexity of the belief space, and be large enough to let LET and DTR work properly.

5.3.4.2 Decentralized Coordination

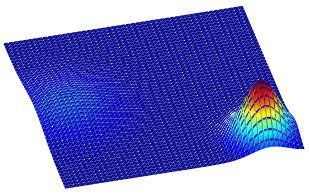
In this section we study the performance of the whole system in decentralized form by testing it in two different synthetic scenarios. We focus on the decentralized NDH strategy (Eq. 5.12) and we compare it with its non heuristically informed version, the decentralized maximum non-detection probability (ND, Eq. 2.24) used in works like [Mathews et al., 2007; Gan and Sukkarieh, 2010]. We want to show the improvements at the MTS problem when computing the expected observation heuristic within a decentralized team of agents.

Figure 5.9 compares the search performance and behavior for a team of coordinating agents. Two scenarios are selected: a two regions scenario (Figure 5.9(a)) and a complex scenario with multiple local maxima (Figure 5.9(b)). The results show that, using the expected observation, the team has a higher search performance than when using the non-heuristic utility function for both scenarios.

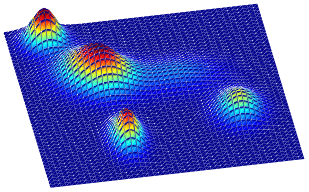
The simulation Figures 5.9(c), 5.9(d), 5.9(e) and 5.9(f) show the team behavior at a fixed instant. Each agent is represented by a color where: the thin line represents the trajectory already followed, the dotted line describes the rest of the trajectory planned and the continuous thick line is the new planning. The gray scale 2D map is the probability of detecting the target updated by the team observations (e.g. the bottom peak showed in Figure 5.9(b) has been already flatten by the team at Figure 5.9(d) and 5.9(f)). Figures 5.9(g) and 5.9(h) shows the information gain obtained by the team at each instant.

In the first scenario, the team that uses the expected observation utility function (Eq. 5.12, Figure 5.9(e)) avoids getting stuck in the low probability region as it happens with the pure short term local controller (Figure 5.9(c)), and the agents explore higher probability areas outside the range of the action horizon window. This results in a faster rate of accumulating probability of target detection (Fig 5.9(g)). Indeed, using the expected observation, in $400s$, the target is detected with 100% of certitude while without the expected observation the probability of detecting the target is less than 50% (Figure 5.9(g)).

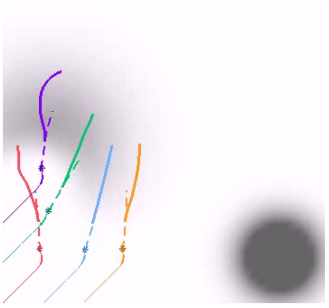
In the second scenario, the agents guided by the proposed algorithm (Figure 5.9(f)) spread more towards the state of higher probability of detecting targets in the future, improving search performance in future decisions. At the simulation instant presented in Figures 5.9(d) and 5.9(f), the agents have observed the closer local maxima, updating the probabilities of that region. Afterwards, while in Figure 5.9(f) the yellow agent is already searching the most right probability peak, in Figure 5.9(d), the yellow agent is



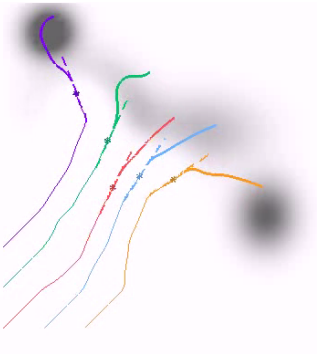
(a) Two gaussians scenario b_τ^0



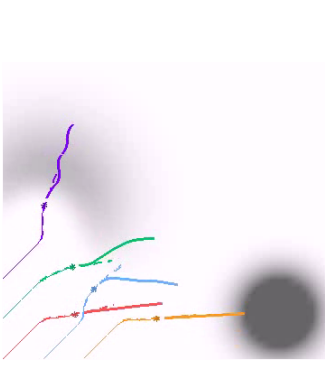
(b) Complex scenario b_τ^0



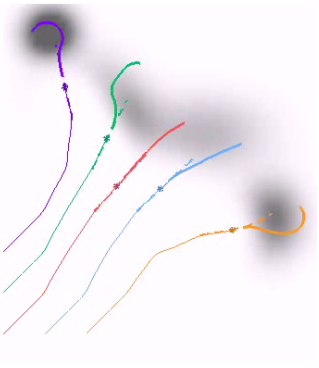
(c) No Expected Observation



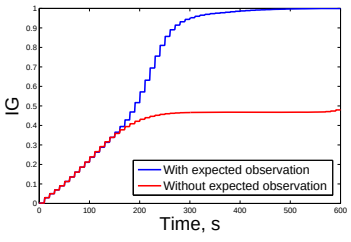
(d) No Expected Observation



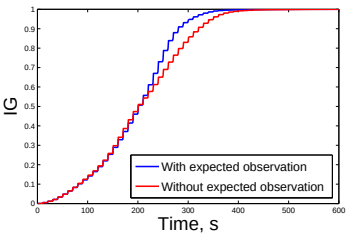
(e) Expected Observation



(f) Expected Observation



(g) Two gaussians scenario IG



(h) Complex scenario IG

FIGURE 5.9: Multi-agent coordination in two different scenarios. The search performance is compared between the non-detection for decentralized coordination [Gan and Sukkarieh, 2010] (red line) and non-detection using expected observation (Eq. 5.12) (blue line). The number of agents used for this simulation is 5.

still far away from that peak, meaning that the information gain is incremented faster with the proposed heuristic method as showed in Figure 5.9(h).

In conclusion, the presented approach improves the performance of a team coordination for the MTS and the target search problems in the following ways: the team takes into account the future, anticipating decisions for a better reward in the next planning steps; it avoids getting trapped in local regions due to the short action horizon, exploring further high probability areas; depending on the platform velocity and the temporal parameter β we can model the importance of the decisions along the time; and the algorithm does not increase the computational time nor the communicational load. The results show that the expected observation improves the target search process in many ways and does not incorporate any disadvantages.

5.4 Out Of Sequence Solution for the Gaussian Special Case

- *Problem Model:* Continuous
- *Target Information Representation:* Gaussian Distribution
- *Optimization:* Local
- *Horizon:* N
- *Sensor:* Continuous
- *Target:* Dynamic
- *Data Fusion Layer:* OOS EKF
- *Multiagent controller:* Gradient-based

The team of agents, as a sensing network, is subject to delay problems. When an agent observes the searching region and sends the measurement to the network, the message could arrive disordered and delayed to the data fusion center. This affects the quality of the target location prediction. In the literature, this problem is known as the Out Of Sequence Problem (OOSP) [Bar-Shalom and Chen, 2005]. As commented in [Bourgault et al., 2004], improving the data fusion center, the cooperation between agents is enhanced. Therefore, dealing with the OOSP our target estimation will have more accuracy and the search and tracking of the target will be enhanced.

Our approach to the OOS-MTS is focused on target locations modeled as a Gaussian density function. In this scenario the team knows the expected location and variance of the target, and its dynamic probabilistic model. Figure 5.10 shows two agents (black lines with dots), planning the best trajectory to search and track correctly the target (blue box under the height map). The setup counts with the agents model, including the sensor, the target model and the probabilistic location information (height map). Although we consider that the agents know their own location without error, we have taken into account the following system sources of noise: the agents measurements errors and the target movements uncertainty. Within a Gaussian representation and in order to estimate the target location, the general RBE is substituted by the Extended Kalman filter (EKF) [Mutambara, 1998].

The proposed solution for the OOSP is to use, as the information layer, a modified EKF that uses the delayed measurements into the estimation process. To deal with the general OOSP we can find in the literature four main approaches: discarding the delayed measurements [Smith and Seiler, 2003], postponing the estimation until the data

is available [Lopez-Orozco et al., 2000], restarting the whole estimation from the oldest measurement time stamp [Kosaka et al., 1993], and doing a forward update approximation using just the new measurements [Nettleton and Durrant-Whyte, 2001; Zhang et al., 2003; Besada-Portas et al., 2009, 2011]. In order to plan correctly the trajectory followed by the team of agents, we need the best updated target information without losing the algorithm computational speed for an online execution. Therefore, the best approach for the MTS is to adapt the fourth method [Besada-Portas et al., 2011] for a distributed multisensor scheme.

We first describe the framework when using a Gaussian distribution to represent the target location (Section 5.4.1) and explain briefly the system design (Section 5.4.2). Afterwards, we propose the OOS data fusion layer (Section 5.4.3). Then we discuss three team strategies to test the OOS layer (Section 5.4.4). Finally, we study statistically the target location estimation improvement when using our approach (Section 5.4.5).

5.4.1 Particular Framework

In this case, the searching region Ω is not bounded and we distinguish two main elements that participate in the task: the team of agents and the target. For simplification purposes, each agent is assumed to have the same dynamics and sensor. Besides, we use a Gaussian non linear model based on the distance for the sensor and a Gaussian density function to model the target location.

The agents broadcast the sensor measurements, which are received by the data fusion layer with a random delay δ that is bounded by the minimum delay δ_{min} (i.e. its own sensor delay) and the maximum network delay δ_{max}. Thus, any measurement broadcasted at instant k arrives to the information layer within the time interval $k + \delta_{min} \leq k + \delta \leq k + \delta_{max}$.

Each agent has a controller that computes the best turn rate at each instant using the uncertain target information. As shown in Figure 5.10, this is computed as a piecewise linear optimization, where we calculate as set of N sequential actions $v_i^k = \{u_i^k, \cdots, u_i^{k+N}\}$ that will transit the agent through the states $\{s_i^k, \cdots, s_i^{k+N}\}$. The computed turn rate u_i^k is applied during the whole stretch between the two states s_i^k and s_i^{k+1}.

Sensor Each agent has a distance sensor that provides the relative distance information to the target. If the euclidean distance between the agent state and the target location is $\|s_i^k - \tau^k\|$, the sensor measurement z_i^k is given by the following equation:

$$z_i^k = \begin{cases} \|s_i^k - \tau^k\| + \nu^k & \text{if } \|s_i^k - \tau^k\| \leq \xi \\ \text{no sensor measurement} & otherwise \end{cases} \tag{5.17}$$

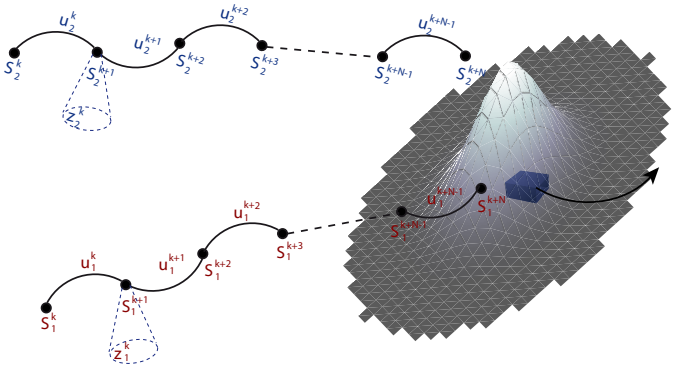


FIGURE 5.10: Multiagent search and tracking using a Gaussian distribution.

In this expression, the measurement error ν^k is modeled as a white noise with zero mean and variance R^k and the sensor only provides measurements when the target is close enough (i.e. when the distance is less than the threshold ξ).

Target Dynamics The real target is defined by its location $\tau^k = (p_x, p_y)$ and the target motion:

$$\tau^{k+1} = f(\tau^k) = \begin{bmatrix} p_x + \Delta k V_\tau \\ p_y + \Delta k V_\tau \end{bmatrix} + \begin{bmatrix} w_x^k \\ w_y^k \end{bmatrix} \tag{5.18}$$

where V_τ is the constant velocity of the target, Δk is the time interval and (w_x^k, w_y^k) are Gaussian noise with zero mean and covariance Q^k.

Gaussian Target Location Belief The team knowledge about the target is uncertain and is modeled probabilistically. Thus, the target location belief b_τ^k is represented by a Gaussian model where the mean represents the estimated location $\hat{\tau}^k$ and the covariance matrix Σ^k represents the uncertainty. Therefore, the probability of finding the target in any point (x, y) in the space is given by the following equation:

$$b_\tau^k = \frac{1}{(2\pi)\sqrt{|\Sigma^k|}} e^{-\frac{1}{2}([x,y]^T - \hat{\tau}^k)^T (\Sigma^k)^{-1} ([x,y]^T - \hat{\tau}^k)} \tag{5.19}$$

We can also refer to the belief as the Gausian distribution with mean $\hat{\tau}^k$ and covariance Σ^k: $b_\tau^k \sim N(\hat{\tau}^k, \Sigma^k)$ and from here, this belief will represent the last updated information that the team has. This believe is updated with the team sensor measurements using the OOS information layer that we describe in Section 5.4.3.

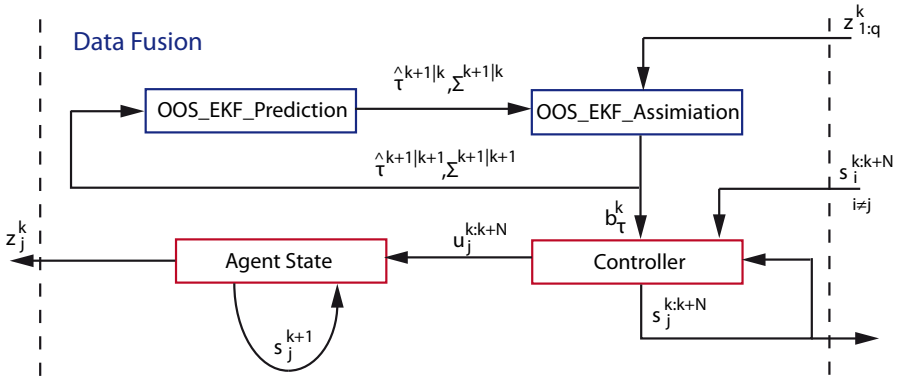


FIGURE 5.11: Search And tracking algorithm.

5.4.2 Search and Tracking Algorithm

The search and tracking algorithm design depicted in Figure 5.11 is based on the system design presented in Section 2.6, where we have two independent and parallel layers that share information: the fusion layer and the controller layer.

The first one is the information management layer that synchronizes the target location belief of the team. Its core is the OOS information filter proposed in the following section, which is compounded by two algorithms[2]: the prediction that estimates the target location and the uncertainty for the next instant $(\hat{\tau}^{k+1|k}, \Sigma^{k+1|k})$ and the assimilation which fuses the information provided by the sensors to update the target location $(\hat{\tau}^{k+1|k+1}, \Sigma^{k+1|k+1})$. Note that the assimilation is executed every time a new measurement arrives while the prediction step is computed for each instant (i.e. due to delays it could happen that at instant k the agent has no observations so it only carries out the prediction step). The OOS layer is always waiting for new measurements sent by the rest of the agents to maintain the belief synchronization and provides the controller layer the target location information to compute the best agent actions. This layer is explained in Section 5.4.3.

The second layer is the controller (Figure 5.11). It gets the last update information about the target location b_τ^k from the OOS DDF layer as a Gaussian with mean $\hat{\tau}^k$ and covariance matrix Σ^k and starts a gradient-based optimization. This method has been previously explained in Section 5.3. Each agent controller improves iteratively its solution using the states calculated by the rest of the team $s_{l\neq j}^{k:k+N}$. The optimization outputs the best actions $u_j^{k:k+N}$ that make the agent j transit towards the new state s_j^{k+1}. In the new state the agent makes a new observation z_j^{k+1} that feeds the OOS layer, closing the process circle.

[2]We use the Kalman notation $k+1|k$ and $k+1|k+1$ where the first index indicates the instant of belief without observation and the second index describes the instant of the last assimilation. See Section 5.4.3 for explanation.

5.4.3 Information Layer: Out of Sequence Data Fusion

The agents that are sensing the region obtain information about the target that is communicated to the team. This process is always subject to delays that make the agents receive the measurements in a late instant $k + \delta$. To tackle these delays we implement the information filter as an OOS data fusion algorithm [Bar-Shalom and Chen, 2005].

This filter works as the data fusion layer and manages the arrival of measurements that have been acquired in early instants. The OOS algorithm presented here is an extension of the EKF and deals with non-linear systems modeled with a Gaussian distribution where we have a hidden variable and multiple sensors. The filter has two iterative steps: the prediction (Algorithm 10) and the assimilation (Algorithm 11).

The agents communicate their measurements z_i^k as well as their time stamps k. At each instant the agent checks if there are new measurements to assimilate and predicts the new state of the target. In order to update the target location belief $b_\tau^{k|k}$ with the OOS measurements provided by the agents, we carry out the assimilation in the information space, following the information filter equations. To predict the new position and its variance we follow the state space prediction equations of the EKF.

The prediction algorithm (Algorithm 10) estimates the mean and the covariance matrix of the target location distribution for the next instant $k + 1$, that in Kalman notation is defined as: $\hat{\tau}^{k+1|k}$, $\Sigma^{k+1|k}$. It uses the Jacobian F of the target motion ($f(\hat{\tau}^{k|k})$, Eq. 5.18) and the target movement noise covariance matrix Q^k.

Algorithm 10 OOS EKF prediction

Require: $\hat{\tau}^{k|k}$ ▷ Estimated and updated target location
Require: $\Sigma^{k|k}$ ▷ Target location error
Require: Q^k ▷ Target dynamics noise covariance

1: $\hat{\tau}^{k+1|k} \leftarrow [p_x + \Delta k V_\tau + w_x^k, p_y + \Delta k V_\tau + w_y^k]^T$
2: $F \leftarrow \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
3: $\Sigma^{k+1|k} \leftarrow F\Sigma^{k|k}F^T + Q^k$

For the assimilation step (Algorithm 11) we need to define a time window $l = \delta_{max}$ that determines during how many instants we store the states and variances to be able to assimilate the out of sequence measurements. Ideally, this parameter is set to the maximum network delay (δ_{max}), that is, the maximum time that an agent needs to receive a measurement from the team. Once we have the time window defined, each agent has to store the prediction target location mean and the covariance matrix from instant $k - l$ to the actual instant k (i.e. $\{\hat{\tau}^{t|t-1}, \Sigma^{t|t-1} | \forall t \in (k - l, k)\}$). It also has to store the already assimilated measurements projected in the information space (i.e.

$\{i^t, I^t | \forall t \in (k-l, k)\}$). The assimilation step updates the target location distribution with the new measurements returning: $\hat{\tau}^{k+1|k+1}$, $\Sigma^{k+1|k+1}$.

Algorithm 11 OOS EIF assimilation

Require: l ▷ Memory time window δ_{max}
Require: $\{\hat{\tau}^{t|t-1}, \Sigma^{t|t-1} | \forall t \in (k-l, k)\}$ ▷ Predicted target location memory
Require: $\{i^t, I^t | \forall t \in (k-l, k)\}$ ▷ Projected measurements memory
Require: R^k ▷ Sensor noise covariance
if New measurements (Z)? **then**
  $t = MinTimeStamp(Z)$
  for $j = t, k$ **do**
    $Y^{j|j-1} \leftarrow (\Sigma^{j|j-1})^{-1}$
    $y^{j|j-1} \leftarrow Y\hat{\tau}^{j|j-1}$
    $Z^j \leftarrow GetMeasurementsAtK(Z, j)$
      ▷ Z^j is a set of z_a^j where a is the agent and j is the time stamp
    for $z_a^j \in (Z^j)$ **do**
      $H = (\hat{\tau}^{j|j-1} - s_a^j)/\|s_a^j - \hat{\tau}^{j|j-1}\|$
      $zc_a^j \leftarrow z_a^j - \|s_a^j - \hat{\tau}^{j|j-1}\| + H\hat{\tau}^{j|j-1}$ ▷ Corrected Measurement
      $i^j \leftarrow i^j + H^T R^{-1} zc_a^j$
      $I^j \leftarrow I^j + H^T R^{-1} H$
      Store new values of i^j, I^j
    end for
    $Y^{j|j} \leftarrow Y^{j|j-1} + I^j$
    $y^{j|j} \leftarrow y^{j|j-1} + i^j$
    $\Sigma^{j|j} \leftarrow (Y^{j|j})^{-1}$
    $\hat{\tau}^{j|j} \leftarrow \Sigma^{j|j} y^{j|j}$
    $(\hat{\tau}^{j+1|j}, \Sigma^{j+1|j}) \leftarrow Prediction(\hat{\tau}^{j|j}, \Sigma^{j|j})$ ▷ Alg. 10
    Store new values of $\hat{\tau}^{j+1|j}, \Sigma^{j+1|j}$
  end for
end if

The set of measurements Z that arrives at instant k to the agent are structures that provide information about the time that the measurement was taken j (i.e. time stamp) and the agent that took it a, that is, elements of the form z_a^j. $MinTimeStamp(Z)$ returns the time stamp that has the oldest measurement and $GetMeasurementsAtK(Z, j)$ returns the measurements taken at instant j of the set Z: Z^j. Therefore, every time the agent receives measurements it checks first which measurement is the oldest, which is the measurement with minimum time stamp. With that information we can approximate the estimation from there. Because the algorithm admits non-linear measurements, we need to do a measurement correction of z_a^j defined by the following equation:

$$zc_a^j = z_a^j - \|s_a^j - \hat{\tau}^{j|j-1}\| + H\hat{\tau}^{j|j-1}$$

where H is the sensor Jacobian, i.e., the partial derivative of the euclidean distance with respect to the agent location.

Once we have the corrected measurement we compute the information i_a^j and the information matrix I_a^j associated with that observation. We do that for every measurement

delayed with time stamp j. Then, we sum all the informations to get the updated information about the target location $y^{j|j}$ at instant j and we predict the new target location using Algorithm 10. We continue assimilating the earlier measurements and predicting the next target state until there are no new measurements.

Executing the prediction and the assimilation algorithms we have an approximated OOS synchronization of the target location information in the team. This process is computed in parallel to the controller, that will use the information to plan the best actions for the search and tracking task.

5.4.4 Cooperation Strategies

In this section we discuss three strategies to compute the best trajectories of the team when the target is described as a Gaussian distribution. These strategies are the core of the controller that drives the team of agents. The first strategy, Max-Slope (MS) (i.e the maximum probability accumulation for a single agent), exploits the geometrical structure of the Gaussian density function assuming that the agents do not have sensor measurements during the decision process. The second approach, Min-Entropy (ME), is based on assuming that our estimate of the target is correct and that the agents have measurements with covariance noise dependent on the distance to the target. Finally, the third strategy, Non-Detection with expected observation (NDH[3]), relies on transforming the Gaussian distribution into a general PDF representation. The controller layer optimizes these utility functions using the Gradient-based algorithm (Section 5.3.2).

We refer to $b_\tau^k \sim N(\hat{\tau}^k, \Sigma^k)$ as the last updated target location distribution given by the information layer and to $\tilde{b}_\tau^k \sim N(\tilde{\tau}^k, \tilde{\Sigma}^k)$ as the target location predicted by the controller when applying the control actions $u_{1:q}^k$.

5.4.4.1 Maximum Slope

The first strategy, already introduced in Section 3.5.2.4, follows the ideas of the optimal density solution for a Gaussian distribution [Stone, 1975], where the best strategy for one agent is to reach the local maxima $\tilde{\tau}^k$ and then describe circles incrementing the radius. The behavior that we are looking for is to reach the maximum value of the Gaussian distribution, meaning that we want to follow the maximum slope. This strategy is designed to work without communication, thus, the optimization is done by each agent without knowing the rest of the team actions. Besides, we assume that during the decision horizon, the agents do not receive any measurements, which is the worst case situation.

[3]This strategy has been presented in Section 3.5.2.3 and has been studied for the continuous MTS in this chapter at Section 5.3.

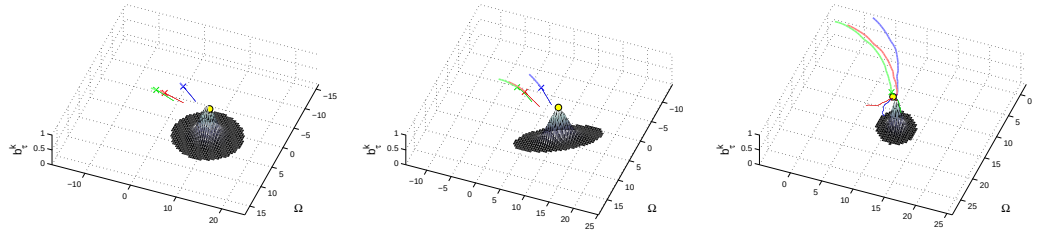
The utility function J for this strategy is the sum of the values of the Gaussian distribution along the agent states, because due to the distribution structure, the sum drives the agent through the maximum incremental slope.

$$J_{MS}(s_i^k, v_i^k, b_\tau^k) = \sum_{j=1}^{N} e^{-\frac{1}{2}(s_i^{k+j} - \tilde{\tau}^{k+j})^T (\tilde{\Sigma}^{k+j})^{-1} (s_i^{k+j} - \tilde{\tau}^{k+j})} \tag{5.20}$$

This function assigns to the agent location the value of the Gaussian distribution at that position. The optimization is convex and optimal for a single agent. In order to compute $\tilde{\tau}^{k+j}$ and $\tilde{\Sigma}^{k+j}$ we use the EKF prediction step (Algorithm 10). We only apply the prediction step because, within the EKF framework, when there are no measurements the update step is not needed.

As this strategy does not use communication during the planning process it has one drawback: all agents pursuit the same peak yielding to suboptimal coordinated behaviors when the agents are close to each other. In the case of a single agent this suboptimal behavior does not exist.

Figure 5.12 shows the behavior of a team of three agents searching and tracking a lost target using this strategy in different instants k. The thick colored lines represent the agents trace and the thin ones define the optimal plan computed by the team. The yellow circle shows the 2D real position of the target that is moving with constant velocity. The height map represents the synchronized belief of the target location b_τ^k. Figure 5.12(a) describes the initial configuration and the three figures sequence shows the drawback of using this strategy, as the agents converge to the same point because the maximum slope ends in the same location. Figure 5.12(b) shows that the agents have started to get measurements, most of them delayed, thus the uncertainty is reduced along the agents direction. Finally, in Figure 5.12(c) the agents are over the target doing the tracking.



(a) $k = 1$ (b) $k = 10$ (c) $k = 20$

FIGURE 5.12: Team behavior using the maximum slope strategy.

5.4.4.2 Minimum Entropy

This strategy, already introduced in Section 3.5.2.4, minimizes the uncertainty that the team has about the target location, represented as the target location entropy at instant $k+N$:

$$J_{ME}(s_i^k, v_i^k, b_\tau^k) = \frac{1}{2}\ln(2\pi e)^2 \det(\tilde{\Sigma}^{k+N}) \tag{5.21}$$

In order to know the predicted covariance matrix at instant $k+N$ ($\tilde{\Sigma}^{k+N}$) we have to make predictions and updates of the Gaussian. We assume that each agent has a sensor that provides the distance to the objective (Section 5.2). The computation of $\tilde{\Sigma}^{k+N}$ is based on an iterative three steps algorithm: 1) prediction (Algorithm 10), 2) generate estimated measure using Eq. 5.22 and 3) assimilation (Algorithm 12). The prediction algorithm is the EKF prediction (Algorithm 10) and the update is the information filter assimilation (Algorithm 12) with measurements and noise according with the strategy assumptions.

In the previous strategy (Section 5.4.4.1) we have assumed the worst case (i.e. no measurements), but using the entropy we need observations to be able to evaluate correctly the agent actions. In fact, without measurements any action is correct from the entropy point of view. Therefore we assume that:

- Our estimate of the target location is correct and the predicted measurement at each agent state is the distance to the mean after the prediction step $\tilde{\tau}^{k+1|k}$:

$$\tilde{z}_j^k = \|s_j^k - \tilde{\tau}^{k+1|k}\| \tag{5.22}$$

- The predicted observation noise is modeled as a zero mean Gaussian noise with covariance $\tilde{R}^k$. Using a constant $\tilde{R}^k$ we obtain good results for one agent (Figure 5.13(a)), but for multiple agents we get an undesired behavior: the agents, in order to minimize the variance, try to be as far as possible from the rest of the team to achieve a better triangulation (Figure 5.13(b)). To solve this problem we make the covariance $\tilde{R}^k$ directly dependent on the distance, i.e., the bigger the distance the higher the uncertainty. The function that follows $\tilde{R}^k$ is:

$$\tilde{R}^k = \|\tilde{\tau}^k - s_j^k\|^2 / K; \tag{5.23}$$

  where K is a constant that makes the obtained values proportional to the size of the searching region.

Algorithm 12 describes the Kalman assimilation for the predicted measurements using the information space representation for a team of q agents with a distance sensor (Eq.

5.17). The algorithm computes the updated Gaussian distribution of the target location, given the last belief ($\tilde{\tau}^{k+1|k}$, $\tilde{\Sigma}^{k+1|k}$) and the predicted team observations $\tilde{z}^k_{1:q}$. First of all we transform the variables from the state space into the information space obtaining the information $y^{k+1|k}$ and its matrix $Y^{k+1|k}$. Then we correct the measurements[4] and compute the information associated with each corrected measurement: $\tilde{i}_j$ and $\tilde{I}_j$. Finally we calculate the new assimilated information $\tilde{y}^{k+1|k+1}$ by summing up all the measurements information and we transform again the variables into the state space.

Algorithm 12 ME Assimilation

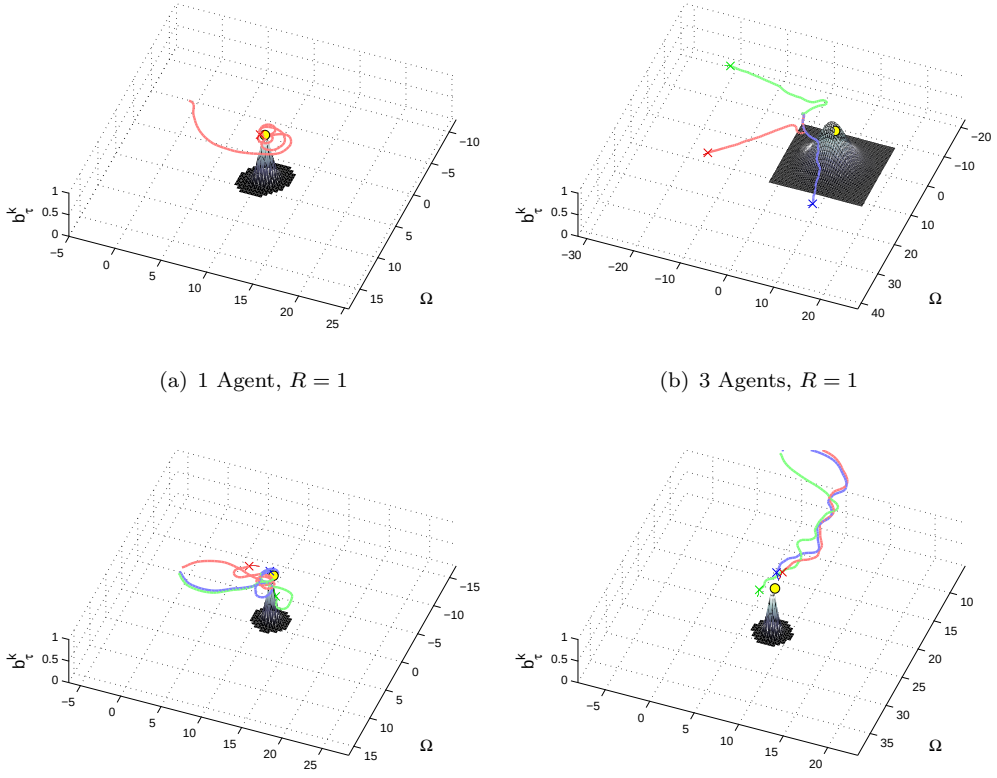
Require: $\tilde{\tau}^{k+1|k}$ ▷ Controller predicted target location
Require: $\tilde{\Sigma}^{k+1|k}$ ▷ Controller predicted target location error
Require: $\tilde{z}^k_{1:q}$ ▷ Modeled sensor measurements

$\tilde{Y}^{k+1|k} \leftarrow \tilde{\Sigma}^{k+1|k}$
$\tilde{y}^{k+1|k} \leftarrow Y^{k+1|k}\tilde{\tau}^{k+1|k}$
for $j = 1,q$ **do**
  $H_j \leftarrow \frac{(\tilde{\tau}^k - s^k_j)}{\|s^k_j - \tilde{\tau}^k\|}$
  $\tilde{zc}^k_j \leftarrow H_j\tilde{\tau}^k$
  $\tilde{i}_j \leftarrow H_j^T(\tilde{R}^k)^{-1}\tilde{zc}^k_j$ ▷ $\tilde{R}^k$ is defined by Eq. 5.23
  $\tilde{I}_j \leftarrow H_j^T(\tilde{R}^k)^{-1}H_j$
end for
$\tilde{y}^{k+1|k+1} \leftarrow \tilde{y}^{k+1|k} + \sum_{j=1}^{q} i_j$
$\tilde{Y}^{k+1|k+1} \leftarrow \tilde{Y}^{k+1|k} + \sum_{j=1}^{q} I_j$
$\tilde{\Sigma}^{k+1|k+1} \leftarrow (\tilde{Y}^{k+1|k+1})^{-1}$
$\tilde{\tau}^{k+1|k+1} \leftarrow \Sigma^{k+1|k+1}\tilde{y}^{k+1|k+1}$

The three steps are executed sequentially N times to obtain the estimated target location covariance matrix $\tilde{\Sigma}^{k+N}$ and to compute the strategy utility function Eq. 5.21.

Figure 5.13 shows the team behavior using the entropy minimization strategy. First of all, in Figure 5.13(a) and Figure 5.13(b) we show the drawback of using a constant noise covariance ($\tilde{R}^k = 1$): although the trajectory obtained minimizing the variance is correct for one agent, in the multiagent case the agents try to separate from each other to triangulate better the information. In the trajectory in Figure 5.13(c), obtained using Eq. 5.23 as the noise covariance $\tilde{R}^k$, we can see that the agents spread to minimize the entropy while approaching the estimated target location. Finally, Figure 5.13(d) shows the team behavior when the target is moving. It is really interesting to observe the locations of that the team trajectories obtained by this strategy: one agent seeks the probability peak (i.e. $\hat{\tau}^k$) while the remaining agents enclose the Gaussian distribution reducing the uncertainty.

[4] The assumed measurements $\tilde{z}^k_j$ are the distance from the agent to the estimated target location $\tilde{\tau}^k$, and therefore, the correction equation $\tilde{z}^k_j - \|s^k_j - \tilde{\tau}^k\| + H_j\tilde{\tau}^k$ becomes $H_j\tilde{\tau}^k$.



(a) 1 Agent, $R = 1$

(b) 3 Agents, $R = 1$

(c) 3 Agents, $R =$ Eq. 5.23

(d) 3 Agents, $R =$ Eq. 5.23, Dynamic Target

FIGURE 5.13: Team behavior using the minimum entropy strategy.

5.4.4.3 Discounted Time Heuristic

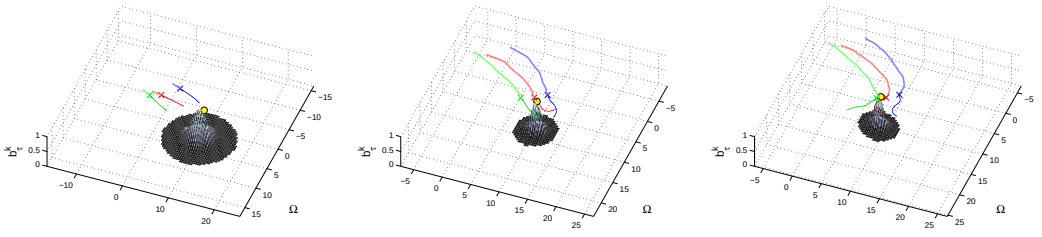
The third strategy discussed is the one explained in Section 3.5.2.3 and used in 5.3. It deals with the information coupling and makes more effective the agents cooperation. To use the NDH strategy we transform the Gaussian location distribution $b_\tau^k = N(\hat{\tau}^k, \Sigma^k)$ into a general PDF. While the controller layer uses the general PDF, the target location distribution remains as a Gaussian at the data fusion layer. The strategy assumes that the target is not moving during the planning horizon, and therefore it has the drawback that the prediction of the target movement is not taken into account. Thus, during the planning $\tau^{k+j} = \tau^k \triangleq \tau$.

Assuming that we have a common decision target belief b_τ^k, the NDH utility function for each agent is defined by the following joint non-detection probability (note that we are using its complementary formulation NDH):

$$J_{NDH}(s_{1:q}^k, v_{1:q}^k, b_\tau^k) = \int_\tau \prod_{j=1}^{N} \prod_{i=1}^{q} \left(P(\overline{D}_i^{k+j} | s^{k+j}, \tau) \overline{\hat{H}}(s_i^{k+N}) \right) b_\tau^k d\tau \tag{5.24}$$

where $\overline{D}_j^k$ is the non-detection event and $\overline{\hat{H}}(s_i^{k+N}) = 1 - \eta\beta^{\|\hat{\tau}-s_i^{k+N}\|/V_i}$ is the heuristic that estimates the future observation designed as a sensor (where V_i is the agent constant velocity, $0 \leq \beta \leq 1$ and η is the heuristic normalization parameter).

Figure 5.14 shows this strategy working in the target search and tracking task. We can see that, comparing with the MS strategy, the agents spread due to the sensor coupling. When the agents start to obtain measurements (Figure 5.14(b)) the error (i.e. the covariance matrix) is contained, and when the agents reach the target they try to push down the probabilities. For long horizon planning or fast target dynamics the solution is just an approximation because it does not take into account the target prediction step during the optimization. The heuristic sensor helps the team to not lose the probability distribution if the agents are far away.



(a) $k = 1$ (b) $k = 10$ (c) $k = 20$

FIGURE 5.14: Team behavior using the NDH strategy.

5.4.5 Results

In this section, we study the performance of the proposed OOS approach by evaluating the target location estimation accuracy. For that purpose we use stochastic metrics [Gorji et al., 2011] to test, for all the strategies discussed, how good is the target position estimation. On one hand, we present the benefits of using the proposed OOS filter (Section 5.4.3). On the other hand, we show which planning strategy improves the target estimation accuracy.

We compare three filter approaches at the data fusion layer: 1) the proposed OOS EKF (OOS); 2) a standard EKF where the delayed measurements are discarded (Discard); and 3) an EKF with no delayed measurements. We assume that the third approach, where there are not network delays, is the best case scenario. Theoretically, the OOS EKF performs better than the second filter and should have a behavior approximated to the third approach. Each simulation is subject to random noise produced by the measurements and the target movements. Therefore, in order to compare the filters performance, we compute $N_S = 50$ Monte Carlo simulations for each strategy presented in Section 5.4.4 (MS, ME and NDH) with each filter approach (OOS, Discard and EKF without delays). For instance, for the MS strategy we compare: MS, DiscardMS and

OOSMS. Besides, the measurement delays are fixed and equal for all simulations, and precomputed by selecting random numbers between 0 and δ_{max}.

The estimation accuracy metrics to compare the approaches are based on the error between the real target location and the estimated one, the estimation entropy and the target detection ratio.

Let the error between the true value of the target location random variable and its estimated value at instant k for simulation n be $r_n^k = ||\tau^k - \hat{\tau}^k||$. Then, the metrics used for N_S montecarlo simulations are the following:

Root Mean Squared Error. It is the expected value of the error and measures the squared difference between the true location and the estimated one for each instant k.

$$RMSE^k = \sqrt{(1/N_S)\sum_{n=1}^{N_S}(r_n^k)^2} \tag{5.25}$$

Entropy. It measures the entropy of the target location information:

$$E^k = (1/N_S)\sum_{n=1}^{N_S}\frac{1}{2}\ln(2\pi e)^2\det(\Sigma_n^k) \tag{5.26}$$

where Σ_n^k is the target location variance computed by the information filter at each instant. High values of this measure means that the uncertainty is elevated.

Average Target Detection. It measures the mean target detections at each instant k. It help us to infer how close to the target the agents are. We count a target detection event when the probability of detecting the target is higher than a threshold ζ. Using the detection likelihood defined for each agent i by the following function:

$$P(z_i^k = D|\tau^k, s_i^k) = P_{d_{max}} e^{-\sigma\left(\frac{\|\tau^k - s_i^k\|}{d_{max}}\right)} \tag{5.27}$$

The detection event is given by:

$$P(z_i^k = D|\tau^k, s_i^k) \geq \zeta \tag{5.28}$$

Therefore, the average target detection (AD) is:

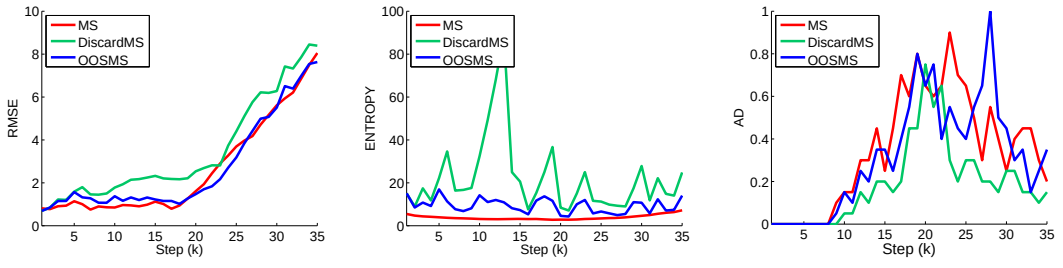
$$AD^k = (1/N_S)\sum_{n=1}^{N_S} I(\exists i|P_n(z_i^k = D|\tau^k, s_i^k) \geq \zeta) \tag{5.29}$$

Where I is an indicator function that returns one when the condition holds and P_n is the detection likelihood for each simulation n.

The initial configuration is common for all simulations: simulation time $T = 35$; receding horizon window $N = 8$; time interval between observations $\Delta k = 1$; number of agents $N_a = 3$; agents initial states $s_1^0 = [3, 0, 0]^T$, $s_2^0 = [0, 3, 0]^T$ and $s_3^0 = [1, 2, 0]^T$; agents constant linear velocity $V_i = 0.5$; target initial state and estimated state $\hat{\tau}^0 = \tau^0 = [10, 1]^T$; target initial covariance $\Sigma^0 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$; target velocity $V_\tau = 0.3$; sensor measurement noise covariance $R = 0.3$; target dynamics noise covariance $Q = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$; sensor measurement threshold distance $\xi = 10$; and maximum team network delay $\delta_{max} = 5$. The detection threshold is $\zeta = 0.6$ and sensor parameters for the AD measure computation and the NDH strategy are: $P_{d_{max}} = 1$, $d_{max} = 1$ and $\sigma = 0.7$.

5.4.5.1 Filter Analysis

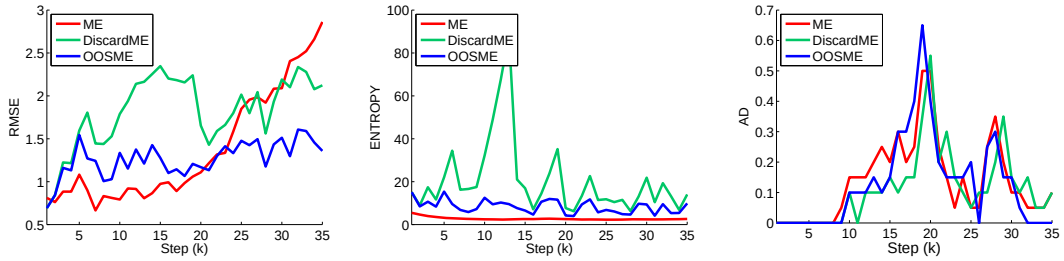
Figure 5.15 presents the search and tracking analysis using the MS strategy (Section 5.4.4.1). The quadratic estimation error in Figure 5.15(a) shows that the proposed OOS layer (OOSMS) estimates the target location in a similar way as the ideal approach (MS). On the contrary, discarding the delayed measurements (DiscardMS), the estimation error grows considerably. It is important to notice that the filter stops tracking the target at instant $k = 25$. This is corroborated because while the estimation error continues growing, the entropy (Figure 5.15(b)) is contained. The solution to this problem is out of the scope of this thesis. The entropy also shows that OOSMS is more pro-active than DiscardMS in reducing the uncertainty, because it has low entropy values in all instants, approximating the ideal MS. The high entropy peaks presented in DiscardMS are produced by the communication delays, which prevent the team from having new measurements, making the target location uncertainty increase. Then the entropy is reduced again when the fusion layer receive new measurements. Nevertheless, OOSMS manages to solve the delays problem. Finally, the average detection plot (Figure 5.15(c)) shows high detection ratios, even achieving a 100% of detections at instant $k = 28$ with OOSMS. This means that the agents are hovering over the center of the Gaussian distribution. Again, the DiscardMS gets the worst results.



(a) RMSE (b) E (c) AD

FIGURE 5.15: OOS study using the MS strategy.

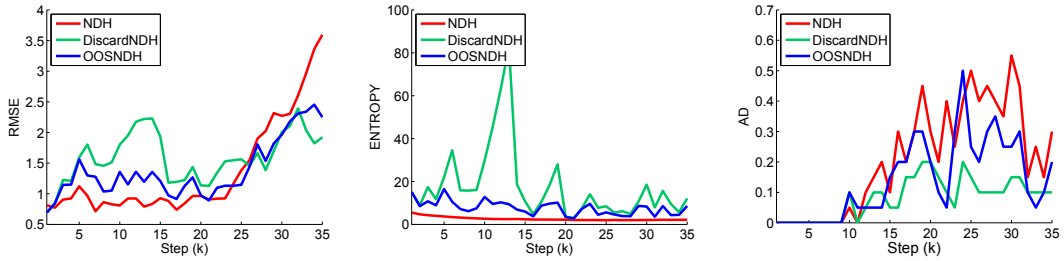
Figure 5.16 shows the study when using the ME strategy (Sec 5.4.4.2). The RMSE (Figure 5.16(a)) shows that OOSME has a really good performance despite of the delayed measurements. In fact, it approaches the behavior obtained by ME, which is the ideal scenario with no communication delays. On the contrary, DiscardME provides worse estimation than OOSME. Unfortunately, using ME, the team loses the target after instant $k = 25$ as explained at the MS results. The entropy analysis (Figure 5.16(b)) shows that the ideal ME maintains the uncertainty perfectly contained with low entropy values and that OOSME deals really well with the measurements delays. Besides, DiscardME is totally affected by the networks delays. Finally, the AD study (Figure 5.16(c)) shows that all filters behave similar with high detection ratios at some particular instants and low ratios at the rest of the simulation.



(a) RMSE (b) E (c) AD

FIGURE 5.16: OOS study using the ME strategy.

Figure 5.17 presents the results when using the NDH strategy. In this case, the RMSE study (Figure 5.17(a)) shows the same undesired behavior as in previous results where the team starts losing the target. Anyway, while the target is correctly tracked, OOSNDH reduces more the estimation error than DiscardNDH. The entropy (Figure 5.17(b)) and the AD (Figure 5.17(c)) plots show again the tendency observed in previous results, where OOSNDH has better performance than DiscardNDH.



(a) RMSE (b) E (c) AD

FIGURE 5.17: OOS study using the NDH strategy.

The team trajectories for just one simulation of the 50 used for the statistical analysis are displayed in Figure 5.18 for all approaches. The labels on the left side of the figure defines the strategy used and the labels on the top specify the filter approach. The agent paths are the red, green and blue lines, and the target trajectory is represented

by the yellow line. The last instant agents planned trajectory is represented by the thin line with the same color as the agent (red, green, blue). The crosses represent the agent location and the yellow circle describes the target position. Finally, the colored height map represents the target location belief (i.e. the Gaussian distribution). From the target estimation point of view the best team trajectory is the one where the Gaussian PDF is more concentrated (e.g. OOSNDH or ME with no delays).

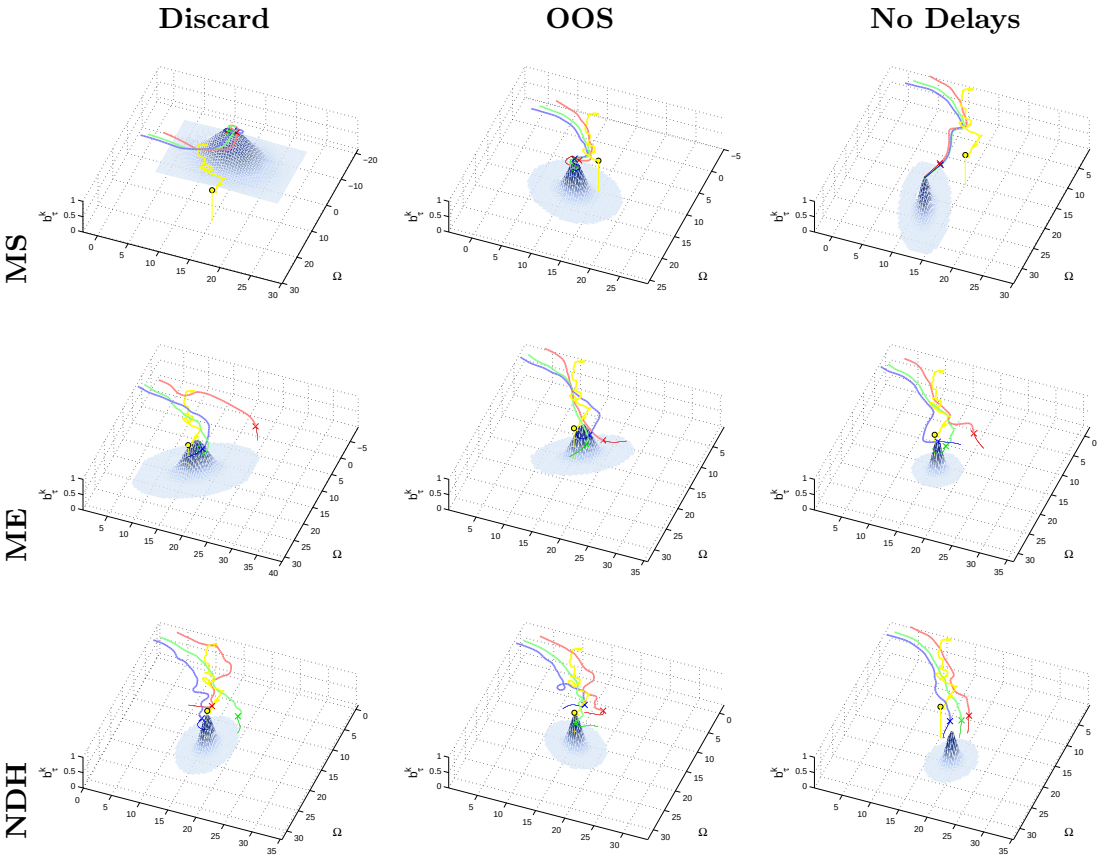


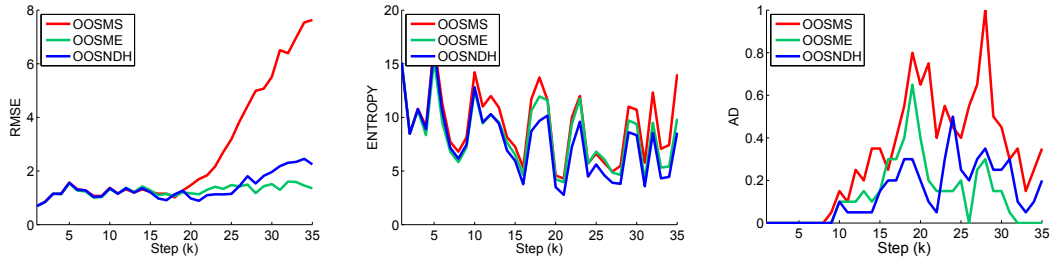
FIGURE 5.18: Team trajectories simulation using the different data fusion filter approaches.

5.4.5.2 Strategies Analysis

We also study which strategy of the ones discussed in this section improves the target estimation accuracy. For that purpose, using the same simulation data as in the previous analysis, we compare all strategies using the proposed OOS filter with measurements delays in Figure 5.20 and with no delays in Figure 5.20.

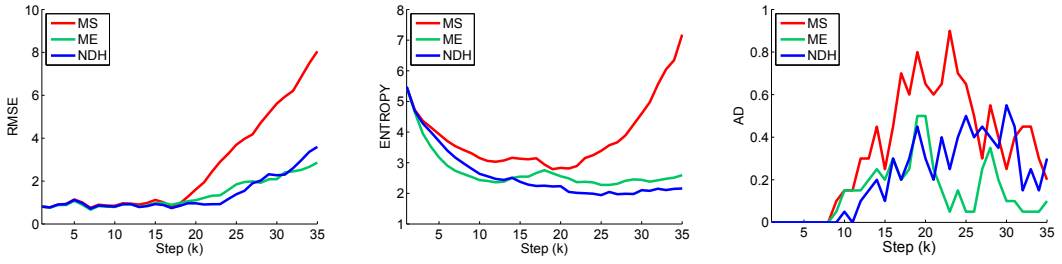
In terms of the target location estimation error, the RMSE metric shows similar performance for all strategies independently of the delays, although OOSMS loses the target tracking earlier (Figures 5.19(a) and 5.20(a)). The entropy study for the OOS case, represented in Figure 5.19(b), shows that OOSMS has the worst performance in reducing the uncertainty. This happen because when using OOSNDH and OOSME the agents are

more spread than when using MS, due to the observation coupling and the information triangulation respectively. We can see the same tendency at the no delayed measurements results showed in Figure 5.20(b). Besides, we observe that the entropy is highly correlated with the delayed measurements because all curves in Figure 5.19(b) have the same shape. However, there is a slightly entropy reduction improvement using the OOSNDH. Finally, the average detection measure shows higher target detection rates using the MS strategy than the other two (Figure 5.19(b) and 5.20(b)). This is totally coherent, because using MS, the agents try to stay at the Gaussian peak, which is the target location estimation. In the case of using ME, the agents tend to separate from the rest of the team to minimize the entropy, a behavior that yields in less detection ratios. Also, one of the three agents sometimes hovers close to the target producing the high AD peaks of ME in Figure 5.19(c). Finally, OOSNDH and NDH tend to have more uniform detection ratio along the time, as showed in Figure 5.19(c) and 5.20(c).



(a) RMSE (b) E (c) AD

FIGURE 5.19: Strategies comparative using the OOS filter.



(a) RMSE (b) E (c) AD

FIGURE 5.20: Strategies comparative without network delays.

5.5 Summary

In this chapter we have approached the MTS in continuous form by approximating it to a piece-wise linear optimization, where the actions take real values, the sensors are continuous and differentiable and the target location is modeled as a PDF. First, we have solved the MTS decision problem using a gradient-based algorithm, whose kernel are the different strategies proposed in this thesis, and secondly, we have tackled the

OOSP (i.e. the arrival of delayed measurements to the data fusion center) by modifying the information layer when the PDF is a Gaussian.

We have presented a non-myopic solution to approach the continuous MTS decision problem. We have combined the gradient-based optimization with the DTH strategy to compute the team actions.

In centralized form, the results show that the MTS strategies combined with the gradient-based algorithm are more effective in terms of time detection reduction. Moreover, the DTH strategies, which incorporate the expected observation, reduce the myopicity of the solution obtained by the local MTS strategies. The results also show that the local MTS strategies (DTR and LET) with a short decision horizon behave similar comparing to the detection strategy and that the strategies that use the heuristic have better performance. Therefore the strategy horizon should be as long as the computational power permits and the expected observation should be used.

We have also presented a non-myopic team coordination solution using a decentralized gradient-based negotiation approach. This solution works as a real time receding horizon controller in the continuous action space suitable for multiagents coordination for searching in a probabilistic environment. The decentralized simulation results and their analysis show that this method improves previous works in complex target scenarios, where the probability of target detection is accumulated at a faster rate.

Moreover we have developed an autonomous multiagent system that tackles the OOSP for the search and tracking task when the target state can be modeled as a Gaussian distribution. The system is compounded by an information management layer that uses an OOS Extended Kalman Filter and a controller layer designed as a model predictive receding horizon algorithm that drives the team towards a common objective, using three strategies: a simple coordination that exploits the Gaussian structure, a cooperative minimization of the information entropy, and a general distribution non-detection approach. The results show the benefits of using the proposed OOS information filter in the data fusion layer and the coordinated strategies as ME and DH in the controller.

Chapter 6

CONCLUSIONS

"Is it probable that probability brings certainty?" Blaise Pascal

This thesis tackles the multiagent Minimum Time Search with constrained paths. It states that, although there are several ways to approach the problem, the Bayesian formalization permits to fully exploit the intrinsic uncertainty of the MTS. In this sense, it studies one of the most frequently used strategies in the literature: maximizing the detection along the agents path. This thesis also proves that in the constrained path case, the detection method does not necessarily minimize the time to find the target. By means of determining the necessary condition that has to be hold to let the maximization of the detection optimize the time to detect the target, the thesis shows that there are many scenarios where maximizing the probability does not minimize the time to find the target. This has permitted us to define the MTS as a special case within the optimal search that needs, to solve it, new strategies that take into account the time directly.

For that purpose we have first developed two local strategies that minimize the target detection time. The first one, LET, is based on the classical definition of the expected time, which can be approximated to a finite horizon without losing its feasibility. The second one, DTR, uses a discounted time function that permits to parametrize the importance of the decisions depending on the instant that they are made. This work shows that both strategies achieve better response to minimize the target detection time than previous existing approaches.

This thesis also shows that the strategies that have a limited decision horizon (receding horizon) cause myopicity problems in future decisions. For this reason, during the development of this work, a third strategy, DTH, has been proposed. This strategy incorporates a heuristic that approximates the expected future agents observation when a set of decisions have already been chosen. It reduces the locality problems of the myopic solutions by anticipating where the promising probability regions to find the target in the future are. The heuristic has been designed in the way that the problem complexity does not rise and that allows its inclusion into the proposed receding horizon probabilistic strategies easily. This is achieved thanks to model the heuristic as a sensor, and therefore, its insertion inside LET or DTR is straightforward. The thesis shows by statistical analysis how DTH reduces the myopicity considerably.

To characterize all the strategies, a decision model for the utility functions is defined. It identifies three types of rewards: the instantaneous reward that appears in the greedy strategies, the short term reward that includes all the receding horizon strategies without terminal cost like LET and DTR, and the future reward, which integrates the methods that estimate the expected observation, such as DTH.

To optimize the proposed strategies, three solutions to be implemented in the autonomous system controller layer have been developed. For the discrete MTS the thesis proposes, on one hand, a global optima solution based on a constraint programming model for finite domains that can only deal with static targets problems with a small number of decision variables and, on the other hand, a stochastic method that, in spite

of finding suboptimal solutions, its results are competitive for searching dynamic targets. For the continuous approach the thesis proposes a gradient-based method to compute the agents actions in either centralized or decentralized form.

Besides, this work improves the data fusion layer, for those problems where the target is described by a Gaussian distribution, by means of an out of sequence information filter that tackles the communication delays.

The simulation results presented in the thesis show how the control layer solutions connected with the fusion layer provide an autonomous intelligent system able to find a moving target with uncertain location in the minimum expected time.

6.1 Summary of Contributions

In this section we describe one by one the thesis contributions from the conclusions point of view. Besides, we provide a short discussion of the results obtained within this work. The main contributions are condensed in Chapter 3 where we describe the MTS formulation and its properties, and the proposed Bayesian strategies. Moreover, we have contributed with some MTS solutions at Chapters 4 and 5, where we can find three different algorithms for the decision layer (CP, CEO and Gradient-based) and one algorithm for the information layer (OOS).

6.1.1 General Formulation of the MTS

The MTS is presented as a complex problem motivated by many applications where the time to locate the object is important. The conceptual definition and examples describe the MTS as a mixture between the data fusion and the decision making theory. Looking at the MTS as an information gathering process, the Bayesian theory turns up as an intuitively and powerful way to approach the problem.

This thesis verifies mathematically, by means of a necessary condition, that maximizing the probability of detection when the agents have constrained dynamics does not necessary minimize the detection time. In fact, we cannot guarantee that this condition holds for any setup and therefore, the MTS problem needs new strategies that optimize the time directly in any case. Although previous researchers had considered that minimizing the expected time was too complex, this thesis shows how to deal efficiently with the time importance. These theoretical results, presented in Chapter 3, are posteriorly corroborated by experimentation in Chapters 4 and 5.

6.1.2 Multiagent Bayesian Search for Dynamic Targets

The decision evaluation is formalized under the Bayesian theory for a multiagent sensing system that searches a moving target, using a unified probabilistic notation, to define several utility functions. These functions, which work as an information filter, permit the developing of the data fusion kernel, as well as, the design of strategies to compute the agents actions.

6.1.3 Tractable Time Optimization Decision Making Strategies under Uncertainty

Based on previous works, as the optimal search book [Stone, 1975] and the thesis [Mathews, 2008], the agents decision layer is modeled as an open loop high level controller that, using the information provided by the data fusion layer, optimizes the actions that the agents must perform. To make the actions calculation computationally tractable, the optimization is approximated limiting the decision horizon and assuming that we know the probabilistic dynamic target model to estimate the target location. Once the controller has been simplified, three strategies have been proposed to solve the MTS decision problem:

- Local Expected Time (LET): by using an expected time detection equation, the thesis presents a receding horizon utility function that conserves the minimum property, i.e., the agents paths with less LET find the target sooner. This strategy minimizes the average time within the decision horizon.
- Discounted Time Reward (DTR): a receding horizon utility function is designed by using a discounted time function (i.e. a decreasing function with dependency on the time) that weights the detection probabilities. Maximizing this function the agents reduce directly the time to find the target. This happens because early decisions that gives high rewards are more valued. Therefore, the agents paths with highest DTR find the target sooner.
- Discounted Time Heuristic (DTH): incorporating a sensor model that approximates the expected value of observing the target (long term reward), the myopicity drawbacks of short term strategies like LET and DTR are reduced. In fact, eliminating the locality we improve the minimization searching time in the long term significantly. Moreover, because the heuristic is modeled as a sensor, it does not increase substantially the complexity of the myopic functions neither the time required to optimize them.

The simulation results shows that LET and DTR reduce the time to find the target and improve previous strategies. LET presents a better behavior because it minimizes the

average time, but in some scenarios, the possibility of tuning the time importance that offers DTR has achieved better solutions.

In terms of myopicity it is worthy to say that, because LET and DTR are computed for a fixed horizon without taking into account the future, they present locality drawbacks. Indeed, accumulating faster the probabilities could make losing good solutions in the future. For instance, in dynamic target scenarios, being too anxious could leave us into a local minima. In this sense, DTR is, according to the results, more sensitive to myopicity.

One way to reduce this problem and, in addition to improve future decisions, is to use the DTH strategy. The present work shows how the heuristic, combined with LET or DTR, or even with the maximum detection strategy (D), avoids locality and enhances the expected time minimization. The analysis of the results shows that DTR and D are more sensitive to the heuristic because it breaks the utility function proportions, while LET uses the heuristic just as a "suggestion". Therefore, the heuristic should be tuned for each utility function. In spite of this fact, the thesis shows that heuristic versions of LET, DTR and D are good approximations to develop controllers that guide the agents within the searching task.

Besides, the results study also shows that some scenarios can satisfy the necessary condition to minimize the time by maximizing the detection probability. In fact, when the target location is described by a Gaussian distribution or by a really smooth distributions, the maximum detection (minimum non-detection) probability strategy combined with the DTH strategy is a good approximation to deal with the MTS problem.

6.1.4 Optimal Constraint Programming Discrete MTS Solution for Static Targets

This thesis provides an optimal algorithm for small scenarios with static targets by modeling the MTS under the CP paradigm for finites domains. Using an open source solver we have implemented the optimization of the DTR strategy. However, this method only solves a subinstance of the MTS problem, as it assumes that the target is static during the decision plan and that the sensor is ideal.

An approximate MTS CP model, which assumes that the agents cannot visit the same location twice, is also presented to solve MTS instances with greater number of decision variables.

The simulation results collected in this thesis show that for small MTS instances, the CP method is a good online choice. Besides, the approximation of restricting the solution to no repeated locations gives really good performance in terms of optimization time and

solution quality. The results show that this approximation is reliable for medium size MTS instances.

6.1.5 Suboptimal Stochastic Discrete MTS Solution for Dynamic Targets

This thesis proposes an approximated algorithm to solve the general MTS for dynamic target scenarios and any type of sensor. This method, which provides suboptimal solutions, is a tractable approach to work as the agents controller. The algorithm has been tested for all the proposed MTS strategies.

The optimality analysis provided for static target scenarios shows that this algorithm approximates quite good the global optima (computed with the contribution of Section 6.1.4), although it presents a little offset and suffers from locality, due to the non-convexity of the problem.

From the strategies point of view, the results show that the algorithm combined with DTR and LET is able to minimize the time to detect the target within the horizon window. Besides, combining these strategies with the DTH one, the myopicity problems are reduced.

6.1.6 Non-myopic Gradient-based Continuous MTS Solution

The thesis proposes a solution for the continuous MTS based on a gradient descent algorithm. The method is local and suboptimal, and optimizes the proposed MTS strategies providing the team actions.

A novel way to design the future reward heuristic, modeled as a sensor, permits the designing of informed methods to solve the MTS when the expected observation is too complex to be computed in reasonable time. The results provided in the thesis show that the team behavior improvements are drastic comparing to the easiness of the heuristic utility function.

It is important to highlight that the proposed solution for the continuous case obtains really quick responses, although it assumes that the target is static. Assuming that the target is static simplifies the estimation problem and the Jacobian computation for the gradient method, but in many real world applications, the target is moving. Therefore, this method can be applied for targets with slow motion, which do not change their location during the agents decision negotiation process. Ideally, we can also recompute the best trajectory at each instant interval, adding in a implicit way, the target dynamics. Otherwise, this algorithm is an approximation.

Finally, this work presents a decentralized non-myopic continuous MTS solution by combining the works [Mathews et al., 2007; Gan and Sukkarieh, 2010] with the DTH strategy.

6.1.7 Out of sequence MTS Solution

When the target location can be modelled with a Gaussian distribution, this work proposes an improved information data fusion layer to tackle the sensor delays within the team. The simulation results shows that, when there are delays in the agents communication, the target searching and tracking is affected reducing the agents searching performance. By dealing with the OOS problem, the team has a more precise estimation of the target location and obtains more accurate MTS decisions. The results show that using the OOS information filter as the data fusion layer, the searching time is reduced and the target tracking is improved.

6.2 Future Work and Open Problems

What do we expect in the following years? Digging into the possible future lines, it is clear that the optimal search research goes towards finding a unified information theoretic controller in a decentralized scheme and its final implementation in real platforms. Also, as long as there is no polynomial algorithm to solve the problem demonstrating that $NP = P$, the researchers will have to develop better approximated algorithms. The algorithms improving direction can be defined as a vector of several components: sensor types, targets dynamics, myopicity, optimality, task decentralization and computational efficiency. Once all these properties are considered within the autonomous system the problem will be successfully tackled.

In terms of the controller layer, we have to differentiate the discrete from the continuous approach.

In the discrete case, we have seen that sampling methods provide suboptimal solutions because of the non-convexity of the problem. Therefore one improvement direction is to design more effective sampling generation where we can cut the solutions using bounds. Decentralization of the discrete approach with convergence proof can contribute with faster online cooperation algorithms. In this sense we can modify the CP model into a decentralized CP one and solve it with some recent solvers that are totally distributed. In the case of CEO we can either exploit its parallelization, develop a decentralized CEO algorithm, or follow the approach used in the gradient-based algorithm with its convergence proof.

In the continuous approach, the improvements should be focused on dealing with moving targets with a decentralized multiagent configuration. The probabilistic strategies with

dynamic targets use an estimation of the future states that introduces a big complexity inside the gradient-based optimization. The Jacobian computation of the utility function becomes computationally expensive, thus an explicit derivation could be a way to approach it for an online solution. Besides, the decentralized version for the dynamic target search needs a convergence proof.

It would also be interesting to try some recent optimization approaches to compute the team actions. On one hand, more sophisticated estimation methods such as the Bayesian optimization algorithm can improve the optimization by learning the decision variables dependencies. On the other hand, novel approaches that mix non-linear constraint programming within the POMDP formalization seem to be acceptable to face the MTS.

Eliminating the myopicity of the algorithm is another complicated issue. Calculating the exact expected reward is still intractable within the optimization because computing the possible detection over all the possible reached states is too expensive. Moreover, the coupling should be taken into account. A really good improvement will be a more accurate heuristic that approximates the expected reward when the target is moving.

In terms of the data fusion layer we can generalize the OOS algorithm for a general target location distribution. Moreover, from the real implementation point of view, communication between agents has to be optimized to provide robustness.

Anyway, we think that the real improvement, that will change the search paradigm is related to the data fusion representation within the algorithms. For instance, we can think about a time-information structure that condenses the knowledge and the time component in an optimized way. This will improve the agents cooperation and coordination, the communication between the agents and the optimization methods. Besides, it will speed up the controllers drastically.

In the future we will have a self organized and coordinated (or cooperative), active, heterogeneous, and flexible sensor networks that communicate their world perception and act following a common objective. The aim will include learning in the environment, minimize the task time and maximize the team reward.

Finally, the implementation on real platforms and consequently the searching task performed by a team of autonomous vehicles is the final objective of this research. Besides, the generalization of the theoretical models into new applications could be an interesting research field too.

Resumen en Español

Chapter 1

INTRODUCCIÓN

"In each action we must look beyond the action at our past, present, and future state, and at others whom it affects, and see the relations of all those things. And then we shall be very cautious" Blaise Pascal

1.1 Introducción

El 5 de Junio de 1968, el submarino USS Scorpion se perdió en las aguas del océano Atlántico mientras realizaba maniobras tácticas [ISBA, 2009]. Se hundió en algún lugar entre los 4000 km que separan las islas Azores de la ciudad de Norfolk en Estados Unidos. La marina estadounidense estuvo buscando en la zona durante cinco meses pero, desafortunadamente, no tuvo éxito. Mientras tanto, el jefe científico de la división de proyectos especiales y sus colaboradores, desarrollaron un nuevo e interesante método basado en la estrategia previamente utilizada para buscar la bomba H en Palomares (España). Utilizando datos acústicos y teniendo en cuenta la probabilidad de cada uno de los fallos que se pudieran haber dado en el submarino, crearon un mapa de probabilidades de su posible localización. La Figura 1.1 muestra un ejemplo ilustrativo de este tipo de mapas donde cada celda expresa el valor de la probabilidad asociada en función de su color, de suerte que cuanto más intenso es el color, mayor es la probabilidad de encontrar el objetivo. Siguiendo dicho mapa el submarino fue encontrado en cinco días, sorprendentemente a tan sólo 200 metros de la ubicación de máxima probabilidad. Este ilustrativo ejemplo permite vislumbrar la potencialidad de utilizar la información probabilística en un problema con incertidumbres involucradas, si bien, la búsqueda de un objeto perdido implica mucho más que construir el mapa de probabilidad inicial.

Para resolver el problema completo, también tenemos que obtener el plan óptimo de observación, es decir, el orden en el que hay que visitar las posibles localizaciones (celdas) para encontrar el objeto lo antes posible. Este problema no es nada sencillo y se

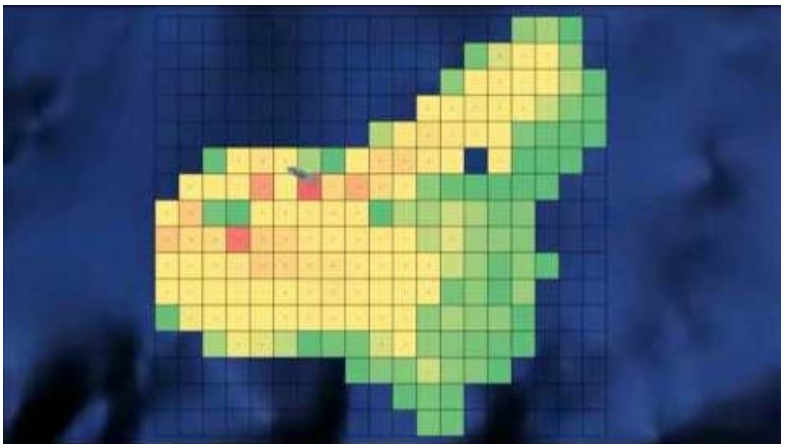


FIGURE 1.1: Mapa de probabilidad de la ubicación del submarino USS Scorpion [ISBA, 2009].

vuelve más complicado cuando el buscador tiene restricciones espaciales[1], contamos con múltiples buscadores y además el objetivo se está moviendo.

Desde el comienzo de la historia, hemos intentado realizar las tareas más eficiente y rápidamente y, con la llegada de los computadores y la teoría algorítmica, algunas de las tareas humanas han sido reproducidas e incluso mejoradas por los ordenadores. Nadie se sorprende cuando un algoritmo elige qué acciones de la bolsa hay que comprar o vender, o si la trayectoria de un vehículo aéreo no tripulado (Unmanned Air Vehicle, UAV) es calculada por un computador [Pajares et al., 2008]. Actualmente, la búsqueda de objetos perdidos sigue siendo realizada por humanos (p.e. tareas de salvamento marítimo [Kratzke et al., 2010]), pero gracias a la robótica moderna, su miniaturización y a sensores y actuadores más baratos, estamos cerca de poder automatizar la búsqueda. Por ejemplo, imaginemos la ventaja que sería contar con vehículos autónomos que ayudaran en operaciones de rescate marítimo o de montaña.

Si queremos trasformar el problema de la búsqueda en un proceso totalmente autónomo, necesitamos primero un sistema que maneje la información para poder actualizar el conocimiento que tenemos de la posición objetivo, y segundo un sistema de decisión que elija las mejores alternativas para encontrar el objetivo. Los dos sistemas tienen que tratar la incertidumbre asociada con la posición del objetivo y sus cambios conforme va pasando el tiempo [Jaynes, 2003], a la vez que permitir que se pueda incorporar nueva información que proviene de la observación del buscador. Además, cuando se dispone de un equipo de buscadores (agentes[2]), necesitamos abordar la comunicación y el acoplamiento de la información entre ellos, y tenemos que permitir su cooperación para conseguir un objetivo común: encontrar el objetivo en el menor tiempo posible.

[1] Cuando hablamos de restricciones en el camino o espaciales, nos referimos a que no podemos observar cualquier localización cuando queramos, sino que la localización depende de la dinámica del buscador. Así, diferenciamos el problema de densidad [Stone, 1989] del problema de búsqueda con camino restringido [Eagle, 1984].

[2] La palabra agente se utiliza en esta tesis para referirse genéricamente al que busca, ya sea un vehículo, una persona o un sensor móvil. Para referirnos a varios buscadores se utilizará la palabra multiagente.

Analicemos el problema mediante un ejemplo ilustrativo para identificar los elementos involucrados y explicar mejor la finalidad del trabajo que se presenta. Imaginemos que hemos perdido las llaves del coche en nuestra casa y que tenemos prisa por llegar al aeropuerto para coger un avión. Nuestro objetivo es encontrar las llaves en el menor tiempo posible y nuestra región de búsqueda (mundo) es la casa. Tenemos una creencia o información inicial del posible paradero de las llaves: normalmente las dejamos en la mesa de la entrada o en el bolsillo del pantalón, que guardamos en el dormitorio. Con esta información inicial podemos crear un mapa de probabilidades que utilizaremos para guiar la búsqueda y cada vez que observemos una estancia de la casa, incluiremos esa información en nuestro conocimiento de la ubicación de las llaves. Es decir, si miramos en la entrada y no están allí, podemos descartar esa estancia y centrarnos en los demás lugares. También puede ocurrir que alguien esté limpiando la casa en ese mismo momento cambiando los objetos de lugar y por lo tanto modificando la ubicación de las llaves, así que tenemos que poder predecir esos movimientos. Además tenemos restricciones en nuestro camino de búsqueda, como para ir del salón a la entrada tenemos que cruzar la cocina. Para movernos de un sitio a otro realizamos acciones, como por ejemplo, desde el salón ir a la izquierda para entrar en la cocina o a la derecha para acceder al baño. Además, en el caso de ser varias personas buscando, repartiéndonos los lugares y comunicándonos nuestras observaciones mejoraremos el éxito de la misión y también reduciremos el tiempo de búsqueda. Teniendo en cuenta todos esos elementos, nosotros, los agentes, debemos planear la secuencia óptima de acciones que nos hacen encontrar las llaves en el menor tiempo posible.

Esta tesis plantea el diseño de un sistema inteligente y autónomo con el fin de encontrar un objetivo de ubicación desconocida en mínimo tiempo, o en otras palabras, estudia y plantea una solución de la Búsqueda de Tiempo Mínimo (Minimum Time Search, MTS) con múltiples agentes. Existen dos motivaciones principales para la realización de este trabajo. La primera se enmarca dentro de los proyectos [DPI, 2006, 2009], donde se expone la cooperación de vehículos autónomos, aéreos y marinos, con la finalidad de conseguir una rápida respuesta en emergencias marítimas. En estos proyectos se consideran escenarios como el control de difusión de sustancias contaminantes [Clark and Fierro, 2005; Lanillos et al., 2009] y el rescate de náufragos [Bourgault et al., 2003]. Los vehículos autónomos tienen la posibilidad de comunicación entre ellos para colaborar en la realización de las tareas en el menor tiempo posible. La segunda motivación se refiere a que, hasta donde hemos podido investigar, no hemos encontrado ningún método efectivo de toma de decisiones que minimice el tiempo de búsqueda cuando la dinámica de los agentes está restringida y el objetivo está a la vez en movimiento. En el presente trabajo se pone de manifiesto la necesidad de mejorar los métodos existentes y proponemos una solución viable al problema del MTS.

Para entender la dificultad para diseñar métodos eficientes y el motivo por el que no

se contempla la minimización del tiempo en ellos es necesario realizar una pequeña revisión histórica del problema. La búsqueda de un objeto con localización incierta ha sido ampliamente estudiada desde que en 1946 el trabajo "Search and Screening" fue publicado [Koopman, 1946]. En 1975 los investigadores ya habían resuelto con brillantez dos objetivos complementarios de la búsqueda [Stone, 1975]: detectar el objetivo con (1) mínimo coste y (2) mínimo tiempo. Si bien, este enfoque tenía un inconveniente principal ya que asumían que el espacio era infinitamente divisible, es decir, que no se contemplaba la dinámica de los agentes o las restricciones espaciales para ir de una localización a otra. Unos años más tarde, un doctor del ámbito de la ingeniería naval llamado Eagle publicó un modelo matemático del problema donde añadía restricciones en el camino del buscador [Eagle, 1984]. A partir de esta aproximación al problema los investigadores se centrarán más en acumular información sobre la posición del objetivo que en la minimización del tiempo de búsqueda. Eagle establecía, basándose en [Smallwood and Sondik, 1973], que el problema es un Proceso de Decisión de Markov Parcialmente Observable (Partial Observable Markov Decision Process, POMDP) y que por lo tanto, se podía resolver con técnicas de programación dinámica. Pese a que el algoritmo no era nada escalable, sentó las bases para plantear algunos algoritmos de ramificación y poda posteriores [Eagle and Yee, 1990; Washburn, 1998]. Por supuesto, se demostró que la complejidad algorítmica era la de un problema NP-completo o NP-duro dependiendo de cómo se instancie la búsqueda [Trummel and Weisinger, 1986].

Trascurrido el tiempo, y con el auge de los enjambres y los aviones no tripulados [Ross, 2011], aparece un nuevo impulso en la investigación de la búsqueda óptima donde, en vez de un único buscador aparecen varios [Yang et al., 2002; Bourgault et al., 2004; Clark and Fierro, 2005; Hollinger et al., 2009]. Entonces se empieza a utilizar la palabra agente para designar a cada plataforma móvil involucrada en la búsqueda y equipada con elementos sensoriales. Utilizando la configuración multiagente se muestra que la complejidad del problema se eleva a NEXP-Completo [Bernstein et al., 2002], pero simplificando las comunicaciones y sincronizando la información que se conoce del objetivo, el problema se resuelve parcialmente cuando el objetivo permanece estático [Mathews et al., 2007]. Todos estos trabajos abordan el problema de la descentralización, es decir, cada agente móvil toma sus propias decisiones cooperando en equipo para encontrar el objetivo. La descentralización, aparte de proveer de escalabilidad, modularidad y redundancia [Bourgault et al., 2004], proporciona mejores resultados en la búsqueda y añade la robustez al sistema. A pesar de estos avances, el problema de decisión no está totalmente resuelto, razón que motiva en parte el planeamiento que se propone en este trabajo.

En la mayoría de las investigaciones mencionadas se olvidan de uno de los dos objetivos fundamentales en la búsqueda óptima original [Stone, 1975]: minimizar el tiempo de detección. De hecho, en [Trummel and Weisinger, 1986] se mencionan teóricamente dos formulaciones para el problema: maximizar la probabilidad de detección y minimizar

el tiempo de detección. En esta tesis veremos que, pese a que ambas son estrategias válidas para buscar objetos perdidos, no consiguen exactamente el mismo resultado. En cualquier caso, sólo la primera estrategia ha sido ampliamente investigada [Eagle and Yee, 1990; Bourgault et al., 2003; Yang et al., 2004; Lavis et al., 2008; Mathews et al., 2007; Gan and Sukkarieh, 2010], en parte debido a la alta complejidad de la segunda [Bourgault et al., 2003]. Investigaciones recientes [Hollinger et al., 2009; Sarmiento et al., 2009] han intentado acercarse al problema de la reducción del tiempo de búsqueda, aunque no han logrado resolver eficientemente el problema multiagente como se ha conseguido en tareas de vigilancia terrestre [Anisi et al., 2010]. Por este motivo, en esta tesis abordamos el problema de la búsqueda desde el punto de vista de la minimización del tiempo en sistemas o situaciones en las que intervienen más de un agente. De hecho, exponemos que el MTS es un problema especial dentro de la búsqueda óptima, que necesita el desarrollo de estrategias de decisión que minimicen el tiempo. Además, analizamos cuándo es factible usar la estrategia de maximizar la probabilidad de detección del objetivo para minimizar el tiempo de búsqueda.

Por otra parte, existe un problema de eficiencia o de demasiadas suposiciones en los métodos encontrados para calcular las acciones de los agentes. Si bien es cierto que, debido a la alta complejidad, no parece existir un método óptimo global tratable, existen algunas aproximaciones como: algoritmos voraces [Bourgault et al., 2003; Yang et al., 2002], optimización local [Gan and Sukkarieh, 2010; Mathews et al., 2007; Tisdale et al., 2009; Sarmiento et al., 2009], reducción del espacio de búsqueda mediante discretización [Yang et al., 2004; Sarmiento et al., 2009] y enumeración de todas las soluciones [Hollinger et al., 2009]. Tanto los algoritmos voraces como los de optimización local son miopes[3] con respecto a decisiones futuras, además, muchos de los métodos asumen que el objetivo no se mueve o que sólo existe un agente buscando. Por eso, en este trabajo se plantean estrategias y algoritmos que se acercan al problema de forma más general y eficiente, y se proponen métodos para reducir la localidad de las soluciones.

En resumen, esta tesis contribuye al problema de la búsqueda con varios agentes (multiagente) con: 1) la identificación del MTS como un problema diferenciado dentro de la búsqueda óptima, además de una formalización y notación unificada totalmente probabilística para la evaluación de las decisiones de los agentes; 2) el desarrollo de nuevas estrategias que sí minimizan el tiempo y eliminan la miopía; y 3) nuevos algoritmos de optimización que mejoran la eficiencia de la toma de decisiones. Aparte, mostramos el diseño general de sistema autónomo para realizar las tareas de MTS.

[3]Las estrategias miopes o ciegas son aquellas que sólo tienen en cuenta los efectos de las acciones en una ventana de decisión limitada, y que por lo tanto obvian el efecto que éstas pueden ocasionar en el futuro.

1.2 Objetivo

El objetivo general de la tesis consiste en diseñar un sistema autónomo que encuentre un objeto perdido en el menor tiempo posible. El sistema está formado por un equipo de agentes móviles equipados con sensores que observan el mundo analizando la información sobre la localización del objetivo. La información obtenida por los sensores es utilizada por los agentes para actualizar el mapa de probabilidad sobre la posible localización del objetivo. En función de ese mapa, los agentes planean las mejores acciones para éxito de la tarea. En particular, estos agentes son vehículos aéreos autónomos, equipados con sistemas de visión artificial capaces de detectar el objetivo, que sobrevuelan la zona de búsqueda. La Figura 1.2 muestra un ejemplo esquemático del problema, donde un grupo de vehículos observan la región de interés y cooperan para elegir el mejor camino de búsqueda. Mientras tanto, el objeto con posición desconocida se mueve por el mundo sin tener en cuenta la presencia de los agentes.

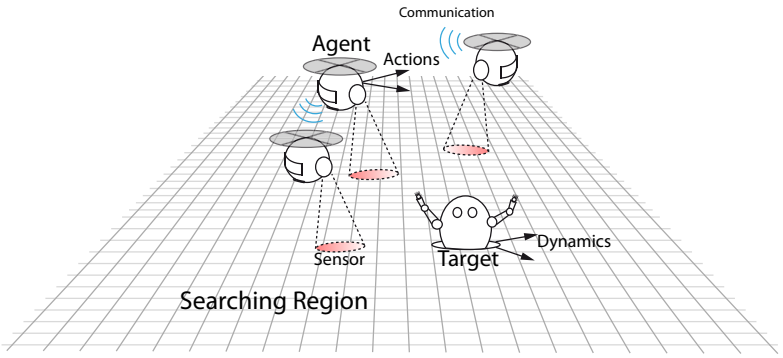


FIGURE 1.2: MTS Configuración Descentralizada.

Consiguiendo este objetivo podemos aproximarnos a resolver las aplicaciones prácticas propuestas en los proyectos [DPI, 2006, 2009] que motivan esta tesis. En el caso de operaciones de rescate de náufragos en el mar [Bourgault et al., 2004], el equipo de búsqueda está constituido por vehículos aéreos (cuatrirotores) equipados con un módulo de percepción visual que permiten la detección de los náufragos. El objetivo es localizar los supervivientes en el menor tiempo posible para su posterior rescate. Utilizando la información sobre las corrientes marinas y el lugar del accidente podemos construir un mapa de probabilidad de su posible ubicación e inferir sus movimientos. Los vehículos, teniendo en cuenta sus restricciones dinámicas, calculan la mejor trayectoria para localizar a los náufragos y cuando los descubren envían su posición a un centro de mando que se encarga de gestionar y coordinar el rescate. En [Lanillos et al., 2009] se aplica una estrategia parecida para el control de sustancias contaminantes, pero en vez de náufragos como en el caso anterior, se buscan áreas contaminadas.

Tal y como se desprende de las aplicaciones anteriores, el sistema autónomo debe, por un lado, predecir y actualizar la localización del objetivo y por otro elegir las mejores

acciones de los agentes. El presente trabajo se centra en la toma de decisiones, o en otras palabras, la planificación de las mejores acciones del equipo de agentes, dada una información a priori de la localización objeto. Así pues, el objetivo principal de este trabajo se puede resumir como sigue:

Cuál es la mejor secuencia de acciones para encontrar en el menor tiempo posible un objetivo ("target" u objeto) del que no se conoce su ubicación.

Se consideran dos configuraciones diferentes del sistema, dependiendo de si la toma de decisiones se plantea de forma descentralizada o centralizada. En el caso de ser descentralizada, esquematizado en la Figura 1.2, los agentes difunden la información entre los miembros del equipo y cooperan para elegir las mejores acciones. Por un lado, cada agente se ocupa de calcular sus acciones teniendo en cuenta la información que le llega del resto del equipo, y por otro, difunde sus observaciones y las acciones que está planeando. Debe cumplirse que las decisiones tomadas por el grupo de agentes minimice el tiempo de búsqueda.

En la configuración centralizada del sistema, que se muestra en el esquema de la Figura 1.3, los agentes móviles, representados por vehículos aéreos, envían la información recolectada a un sistema central que actualiza el mapa de probabilidad de la localización del objetivo y calcula las mejores acciones, que son enviadas a los agentes para su ejecución. El objeto, mientras tanto, representado por la figura del robot, se mueve independientemente de las acciones de los agentes. La tarea finaliza cuando alguno de los agentes detecta el objetivo. El sistema consiste básicamente en una estación central que controla un equipo de agentes capaces de observar el entorno y de comunicarse con él.

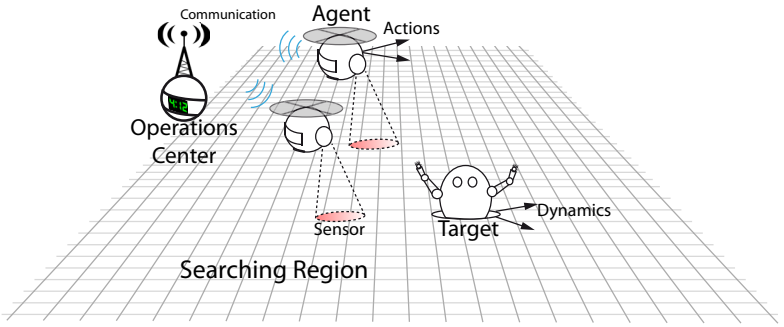


FIGURE 1.3: MTS Configuración Centralizada.

Desafío: Posición del objetivo incierta + objetivo móvil + múltiples agentes + tiempo óptimo. **Objetivo Principal**: Diseñar algoritmos y estrategias que decidan las mejores acciones para detectar un objeto con localización desconocida en mínimo tiempo.

1.3 Enfoque

En esta sección caracterizamos el MTS con sus elementos y propiedades, analizamos el problema desde el marco de la toma de decisiones y explicamos el diseño general del sistema autónomo y cómo hemos enfocado sus diversas componentes.

En el MTS participan esencialmente dos entidades: los agentes móviles que tienen capacidad de observar y el objeto que se busca (objetivo). Por un lado, la posición inicial y las acciones que pueden realizar los agentes son conocidas. Estas acciones hacen que los agentes se muevan de manera determinista por la región de búsqueda. Por otro lado, la posición del objetivo es incierta y, aunque no conocemos su posición exacta, tenemos una información previa sobre dónde puede situarse (información a priori). Además, el objetivo se mueve durante la búsqueda siguiendo una dinámica incierta. Los agentes cuentan con el modelo probabilístico de la dinámica del objetivo para tomar sus decisiones. Finalmente, consideramos que la región de búsqueda (mundo), donde se encuentra el objetivo y que los agentes observan, es finita y está delimitada.

La solución del MTS, desde el punto de vista de decisión, coincide con la secuencia de acciones que el equipo de agentes debe realizar para detectar el objetivo en el menor tiempo posible. Este proceso de decisión tiene que tener en cuenta lo siguiente: 1) cada vez que se realiza una acción se produce una nueva observación que cambia la creencia que tenemos sobre la localización del objetivo, afectando a futuras decisiones; 2) el tiempo es importante, es decir, el orden de las acciones es relevante; y 3) la búsqueda finaliza cuando se encuentra el objetivo.

Para realizar la tarea de MTS proponemos un sistema inteligente y autónomo basado en los trabajos de [Yang et al., 2002; Bourgault et al., 2004; Mathews, 2008]. En este sistema, descrito en la Figura 1.4 de forma abstracta, se distinguen dos capas diferenciadas: la capa de fusión sensorial, que actualiza la información con las nuevas observaciones y la capa de control[4], que calcula las acciones de los agentes utilizando la información de la capa de fusión y un modelo de observación. En el caso de que el sistema sea centralizado, las capas de fusión sensorial y de decisión no se encuentran en los agentes sino en un centro de operaciones, y el agente sólo posee la capacidad de moverse, sentir y comunicarse (enviar las observaciones y recibir las acciones). En el caso de tratarse de un sistema descentralizado, cada agente cuenta con su propia capa de fusión y decisión que se sincronizan de forma trasparente [Bourgault et al., 2004; Gan and Sukkarieh, 2010], es decir, ambas capas se comunican con el resto de agentes para actualizar la localización del objetivo y las acciones que realizarán los demás agentes. Así, el sistema descentralizado se convierte en una red de agentes que cooperan en la búsqueda.

[4]La capa de control es realmente un planificador, que dada la información del objetivo, calcula las mejores acciones que cada agente debe realizar. Al funcionar como un controlador de alto nivel en lazo abierto [Mathews, 2008], utilizaremos las palabras controlador, planificador y optimizador indistintamente para referirnos a la capa de decisión.

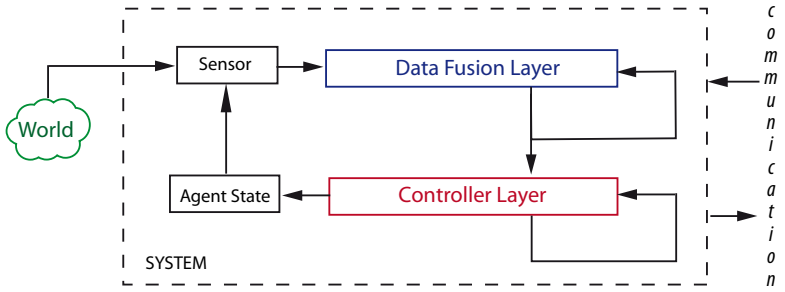


FIGURE 1.4: Diseño del sistema para resolver el MTS.

El proceso de funcionamiento del sistema, de forma simplificada, es el siguiente: los sensores toman medidas y las envían a la capa de fusión sensorial, donde se actualiza y se predice la localización del objetivo; con esta información, el controlador elige las mejores acciones para encontrar el objeto; el agente ejecuta esas acciones desplazándose por la región de búsqueda realizando una serie de observaciones; finalmente las observaciones alimentan de nuevo la capa de fusión y se repite el ciclo.

El presente trabajo se centra fundamentalmente en la capa del controlador, aunque también proponemos mejoras en la capa de fusión cuando las observaciones (medidas de los sensores) llegan con retraso o desordenadas [Bar-Shalom and Chen, 2005]. Cuando contamos con un equipo de agentes móviles con capacidad sensorial, que están enviando información constantemente, puede ocurrir que existan retrasos en la red de comunicación. Estos retrasos empeoran la estimación de la localización del objetivo y en consecuencia hacen que las decisiones sean deficientes. Por lo tanto, necesitamos un algoritmo capaz de tratar con ese problema y así mejorar la sincronización del proceso de estimación del estado/posición del objetivo.

El diseño o definición de la capa de control es un problema muy complejo que puede ser abordado desde diferentes disciplinas de la computación como son: optimización combinatoria [Berger et al., 2009; Sarmiento et al., 2009], POMDPs [Eagle, 1984; Kaelbling et al., 1998; Hsu et al., 2008], teoría del control [Smallwood and Sondik, 1973; Bernstein et al., 2002; Furukawa et al., 2006], fusión sensorial [Lavis et al., 2008; Bourgault et al., 2004], teoría de la información [Yang et al., 2004; Kagan and Ben-Gal, 2006], y teoría de la probabilidad [Yang et al., 2002; El-Mane Wongy, 2005; Bertuccelli and How, 2006]. En este trabajo nos acercamos al problema desde un punto de vista probabilístico y Bayesiano [Jaynes, 2003]. Por un lado, para explotar la información previa y por otro lado, porque permite el diseño intuitivo de estrategias que resuelven el MTS. Al diseñar las estrategias y los algoritmos tendremos siempre en cuenta que deben ser tratables y viables para su ejecución en línea, es decir, que el tiempo de cálculo de las acciones tiene que ser rápido en comparación con la dinámica de los agentes para que el algoritmo permita la replanificación.

Nos enfrentamos al problema mediante dos representaciones diferentes: (1) discreto, donde las acciones de los agentes toman valores discretos, la región de búsqueda y por lo

tanto la información del objetivo está representada por una rejilla y (2) continuo, donde el conocimiento de la localización y el sensor son funciones continuas y derivables, y las acciones toman valores reales. Ambos problemas son no-convexos y se resuelven mediante optimización combinatoria discreta y mediante optimización continua a tramos.

1.4 Contribuciones Principales

Las contribuciones principales de la tesis se clasifican en tres bloques. El primer bloque es la formulación del problema, donde establecemos que el MTS es un problema, según nuestro conocimiento, sin resolver, al menos totalmente. En efecto, los trabajos citados previamente no minimizan necesariamente el tiempo de búsqueda o son ineficientes. Además, proporcionamos una notación Bayesiana unificada para la evaluación de las decisiones. El segundo bloque aborda el diseño de estrategias que sí minimizan el tiempo de detección del objetivo. Finalmente, el tercer bloque consiste en algoritmos que mejoran, en combinación con las estrategias, métodos previamente utilizados para resolver el MTS tanto en discreto como en continuo.

1.4.1 Formulación de la Búsqueda de Tiempo Mínimo

Verificamos, mediante una condición necesaria, qué estrategias previas utilizadas para la búsqueda óptima no minimizan el tiempo de detección del objetivo cuando el camino está restringido. Esto significa que el MTS debe ser formalizado como un problema especial dentro de la teoría de búsqueda óptima, con sus características y estrategias propias. En este sentido, proponemos una formulación Bayesiana unificada para multiagentes y objetivos dinámicos. Esta contribución extiende [Eagle and Yee, 1990; Yang et al., 2004; Bourgault et al., 2004; Mathews, 2008] y permite el desarrollo de innovadoras estrategias Bayesianas que minimizan el tiempo.

Estas contribuciones se encuentran descritas en las Secciones 3.1, 3.2 y 3.3 del Capítulo 3, y han sido publicadas en los siguientes artículos: [Lanillos et al., 2012, 2013].

1.4.2 Estrategias MTS

Contribuimos con tres estrategias para abordar el MTS: minimizar el Tiempo Esperado Local (Local Expected Time, LET), donde el tiempo esperado para detectar el objetivo se reformula como una función objetivo con horizonte limitado (la ventana de decisión es finita); maximizar la Recompensa de Tiempo Descontado (Discounted Time Reward, DTR), donde una función de tiempo descontado valora, asignando pesos específicos, la información acumulada por los agentes dándole más importancia a las decisiones que se toman antes; y la Heurística de Tiempo Descontado (Discounted Time Heuristic, DTH),

que utiliza una innovadora heurística modelada como un sensor que reduce la miopía de las estrategias. Las estrategias propuestas mejoran las encontradas en la literatura [Eagle, 1984; Yang et al., 2002; Bourgault et al., 2003; Mathews, 2008; Sarmiento et al., 2009] en términos de minimizar el tiempo y en ser más anticipativas con respecto al futuro.

Las contribuciones relacionadas con las estrategias de decisión se describen en la Sección 3.4.1 del Capítulo 3, y han sido publicadas en los siguientes trabajos [Lanillos et al., 2012, 2013].

1.4.3 Algoritmos

Desde el punto de vista de la optimización discreta proporcionamos dos algoritmos para la capa de decisión, que en combinación con las estrategias, mejoran ostensiblemente el MTS, en comparación con soluciones previas como las propuestas en los siguientes trabajos [Eagle, 1984; Eagle and Yee, 1990; Bourgault et al., 2003, 2004; Yang et al., 2002, 2004]. Dentro de la optimización continua, basándonos en los trabajos de [Mathews et al., 2007; Gan and Sukkarieh, 2010] diseñamos un algoritmo que reduce la miopía y por lo tanto mejora la búsqueda del objetivo, y refinamos la capa de fusión sensorial cuando hay retrasos en la comunicación, obteniendo una estimación de la localización del objetivo más precisa.

Por lo tanto, contribuimos con cuatro soluciones que abordan el MTS desde diferentes puntos de vista:

- Expresar el problema de decisión del MTS bajo el paradigma de la programación con restricciones con dominios finitos. Utilizando esta aproximación se obtienen soluciones globales y óptimas cuando el número de variables de decisión es pequeño. Es decir resolvemos el MTS discreto en tiempo finito.

  Esta solución se describe en la Sección 4.2 del Capítulo 4.

- Un algoritmo estocástico basado en el método de optimización de entropía cruzada (Cross Entropy Optimization, CEO) que resuelve el problema de MTS utilizando las estrategias propuestas para objetivos dinámicos y que soporta cualquier modelo de sensor. Llegando a un consenso entre optimalidad y tiempo de computación, este algoritmo puede calcular en línea las acciones de los agentes.

  Este algoritmo se encuentra descrito en la Sección 4.3 del Capítulo 4 y ha sido publicado en [Lanillos et al., 2012].

- Un algoritmo no miope y descentralizado basado en el método de gradiente que resuelve el MTS continuo y que puede ser ejecutado en línea. Optimiza la estrategia DTH mejorando los algoritmos existentes reduciendo la localidad y proporcionando mejores decisiones a largo plazo sin añadir tiempo significativo de cálculo.

Esta solución se encuentra descrita en la Sección 5.2 del Capítulo 5.

- Un algoritmo que resuelve, dentro de la capa de fusión sensorial, el problema de las medidas desordenadas (Out Of Sequence, OOS) cuando la posición del objetivo está descrita por una distribución de probabilidad Gaussiana. Esta contribución mejora la estimación de la localización del objetivo, cuando existen retrasos en la comunicación. Por lo tanto consigue una mejor respuesta del sistema de agentes en la búsqueda.

  Este algoritmo se explica en la Sección 5.3 del Capítulo 5 y ha sido publicado en el artículo [Besada-Portas et al., 2012].

Chapter 2

CONTEXTO

"Today's posterior distribution is tomorrow's prior" Lindley

2.1 Antecedentes Históricos

Para entender el principal objeto de estudio que abarca esta tesis, la *búsqueda de tiempo mínimo*, resulta interesante hacer una revisión histórica de aquellos trabajos relacionados con dicho campo de estudio. Sin embargo, esta revisión está enfocada específicamente a los estudios que han motivado la tesis. Otras investigaciones pueden encontrarse en [Stone, 1989; Benkoski et al., 1991].

Durante la Segunda Guerra Mundial, los científicos e ingenieros ya tenían en mente la importancia que los computadores tendrían en diversas aplicaciones, pero la tecnología no se desarrolló tan rápido como la teoría. Numerosos casos de estudio mecánicos como la máquina para desencriptar el famoso "Enigma" fueron a parar a los libros de historia como avances de la computación. Nombres como Claude Shannon, Alan Turing o Von Neumann ya estaban embarcados en proyectos de investigación de gran relevancia. Esto significa que aquel periodo puede considerarse como la "sopa primigenia" de las ciencias computacionales. La teoría algorítmica comenzó a entremezclarse con modelos matemáticos clásicos y las aplicaciones aparecieron por doquier. Dichas aplicaciones se evidenciaron en la marina, particularmente en las flotas submarinas, que realizaban, la mayor parte del tiempo, tareas sin visibilidad. A raíz de esto, comenzó la idea de desarrollar nuevos modelos matemáticos para optimizar las trayectorias submarinas. De todos modos, hubo que esperar hasta la utilización de submarinos nucleares durante la Guerra Fría, en los años sesenta, para que se pusiera en práctica la primera aplicación real: encontrar la embarcación perdida del submarino USS Scorpion [Stone, 1975]. Un

grupo de ingenieros navales y matemáticos tuvo que trabajar intensamente para encontrar un método que ofreciera la mayor garantía de éxito posible en ambientes desconocidos. Gracias a la inteligencia o la información de expertos, se pudieron desarrollar un mapa con las regiones de mayor probabilidad para la búsqueda de objetivos tales como embarcaciones u otros submarinos. En aquel momento, la teoría probabilística y la búsqueda llegaron a ser compañeros inseparables. La teoría probabilística desarrollada por Bayes y sus contemporáneos, la teoría algorítmica y una nueva disciplina que había comenzado durante la Guerra, conocida como investigación operacional, que funcionaba utilizando métodos matemáticos para optimizar la toma de decisiones, dieron forma a lo que pasó a conocerse más tarde como búsqueda óptima. En el libro "Teoría de Búsqueda Óptima", escrito por Stone y publicado en 1975 [Stone, 1975], podemos encontrar un compendio de modelos matemáticos y algoritmos utilizados por la marina para buscar objetivos perdidos.

Stone se centró en el problema de calcular la distribución óptima del tiempo que debería dedicarse a cada región de búsqueda. Algunos años después, en una revisión de los métodos de búsqueda óptima [Stone, 1989], el propio Stone renombraría ese problema como el problema de densidad de búsqueda para distinguirlo de otras instancias del problema. Algunos autores contemporáneos se dieron cuenta de que la búsqueda era un subproblema de otro más general conocido como Procesos de Decisión Parcialmente Observables de Markov (POMDP, según las siglas en inglés), ya que al final, observamos parcialmente un estado desconocido, es decir, la localización del objetivo. La búsqueda como POMDP se realizó en primer lugar para un escenario reducido [Dobbie, 1974] y fue generalizada posteriormente [Smallwood and Sondik, 1973]. Al mismo tiempo, el ingeniero naval Eagle [Eagle, 1984] advirtió que, al introducir restricciones en los caminos de búsqueda, los métodos proporcionados por Stone dejaban de funcionar. Entonces, desarrolló un método para resolver el problema de la búsqueda cuando el camino está restringido, basándose en algoritmos de programación dinámica para resolver los POMDP. En la práctica, estos métodos son una modificación del algoritmo de iteración del valor original propuesto por Bellman [Bellman, 1957] para trabajar con estados de creencia. Se enfrentó entonces al problema inherente de solucionar un POMDP, la gran complejidad debida a la explosión del número de estados, también conocido como la maldición de la dimensionalidad [Powell, 2007]. Por esta razón y dada la baja capacidad de computación de aquel momento histórico, tan solo se resolvieron problemas con escenarios pequeños (p.e. una malla de tres por tres celdas). En 1991 los autores del trabajo en [Benkoski et al., 1991] afirmaron:

"No se conoce ningún algoritmo general y efectivo para generar caminos óptimos de búsqueda"

Simultáneamente, algunos autores analizaron con éxito la complejidad del problema de búsqueda cuando el camino está restringido [Trummel and Weisinger, 1986] y demostraron, mediante su transformación a un problema conocido que la complejidad para un único agente puede ser la de un problema NP-duro o NP-completo dependiendo de su instanciación.

Años más tarde, [Eagle and Yee, 1990; Washburn, 1998] diseñaron algoritmos de ramificación y poda que solucionaban algunas de las lagunas existentes en los multiplicadores de Lagrange utilizados por sus predecesores y redujeron el tiempo de computación limitando el número de estados viables. De todos estos estudios se deriva una idea fundamental, mencionada en [Eagle, 1984; Stone, 1989; Benkoski et al., 1991], que sería central para futuras líneas de investigación. Establecen que si el problema es encontrar un objetivo perdido, entonces dicha tarea termina una vez que el objetivo es encontrado y se puede asumir que el objetivo no es detectado durante todo el plan de decisión. Esta idea convierte el problema POMDP en una optimización determinista, donde sólo hay un tipo de observación: la no detección.

Como un río que se esconde en el subsuelo para reaparecer unos kilómetros aguas abajo, la investigación de la búsqueda óptima se reactivó al comienzo del siglo XXI gracias a un grupo de científicos que trabajaba en el campo de la fusión sensorial, y gracias también a otras mejoras tecnológicas como la miniaturización de plataformas sensoriales y los vehículos no tripulados [Ross, 2011]. Los investigadores se enfrentaron a la complejidad de utilizar los algoritmos existentes en aplicaciones reales y particularmente en configuraciones multiagente. De hecho, Bernstein mostró que el problema de búsqueda multiagente eleva la complejidad del problema a NEXP [Bernstein et al., 2002; Mathews, 2008]. Llegado este punto, las líneas de investigación se dividieron en tres. Primero, los algoritmos genéricos POMDP siguieron su camino ofreciendo nuevos algoritmos de aproximación [Kaelbling et al., 1998; Hsu et al., 2008]. Sin embargo, estos algoritmos presentan desventajas que se explicarán en la siguiente sección. La segunda línea de investigación intentó aproximarse al problema de forma discreta con trabajos relacionados con la búsqueda [Yang et al., 2002, 2004] y trabajos relacionados con problemas más generales [Blum et al., 2003]. La tercera línea se aproximó al problema desde una perspectiva continua y con una fuerte base teórica de fusión sensorial [Bourgault et al., 2004; Mathews et al., 2007; Lavis et al., 2008; Gan and Sukkarieh, 2010]. La metodología utilizada en esta tesis se basa en los trabajos de la segunda línea y de la tercera, que utilizan una aproximación Bayesiana.

Pero, cómo consiguen estos métodos solucionar un problema tan complejo en un intervalo de tiempo de computación razonable? La respuesta es sencilla: utilizando ciertas aproximaciones y suposiciones. La aproximación más importante consiste en reducir el horizonte o limitar la ventana de decisión. Con información incierta

esto no es una mala idea, ya que es posible que la información pierda valor en el futuro, pero si se reduce el horizonte no tenemos una visión global del escenario, y por lo tanto, nuestras decisiones estarán sesgadas y los agentes correrán el riesgo de terminar atrapados en un óptimo local. La mayoría de los trabajos de la tercera línea, [Bourgault et al., 2003, 2004; Mathews et al., 2007; Lavis et al., 2008; Gan and Sukkarieh, 2010], siguen esta aproximación. Un modo de paliar el problema de la localidad, sin perder tratabilidad computacional, es utilizar métodos heurísticos como en [Yang et al., 2002]. Desde el punto de vista multiagente, otra posibilidad para hacer el problema tratable es dividir el sistema de búsqueda en un proceso de fusión sensorial y en un controlador que guía a los agentes. Asumir que existe una sincronización de la información sobre la posición del objetivo entre el equipo de agentes simplifica significativamente la toma de decisiones del controlador [Bourgault et al., 2004; Mathews et al., 2007]. En este sentido, las técnicas de fusión de datos ayudan en gran medida al problema de búsqueda óptima.

Desde el punto de vista de la minimización del tiempo de búsqueda, desde el trabajo de Eagle [Eagle, 1984] hasta hoy, los investigadores han abandonado el objetivo original de la búsqueda óptima: minimizar el tiempo para encontrar el objetivo. La mayoría de los trabajos, con excepciones como [Sarmiento et al., 2009], utilizaron una de las posibles formulaciones definidas en [Trummel and Weisinger, 1986] para la búsqueda, que es, maximizar la probabilidad de detectar el objetivo. Aunque como veremos en el siguiente capítulo, esta aproximación no tiene en cuenta el tiempo necesariamente. De hecho, los autores de [Sarmiento et al., 2009] escriben:

"Una función de utilidad más efectiva equilibra el deseo de una alta probabilidad de encontrar el objeto frente al deseo de minimizar el tiempo de búsqueda"

[Stone, 1975] demostró que la estrategia de maximización de la probabilidad de detectar el objetivo dentro del problema de densidad también minimiza el tiempo de búsqueda. Pero cuando restringimos el camino de búsqueda, dicha afirmación deja de ser cierta. Para ser justos, en algunos casos, cuando se da la condición necesaria presentada en la Sección 3.2, maximizar la probabilidad es una estrategia factible para minimizar el tiempo.

Actualmente podemos enumerar algunos métodos de referencia para la búsqueda óptima con caminos restringidos: un algoritmo de óptimo no tratable [Eagle, 1984] con su aproximación [Dell et al., 1996]; una aproximación al tiempo local esperado para un único agente [Sarmiento et al., 2009]; una aproximación discreta al problema de múltiples agentes utilizando métodos heurísticos que no minimizan el tiempo directamente [Yang et al., 2002]; y un algoritmo de cooperación multiagente

[Mathews et al., 2007], que maximiza la probabilidad de detectar un objetivo estático y que proporciona soluciones locales.

Esta tesis se apoya en estos antecedentes históricos, incluyendo todos los trabajos citados de modo que complementen y justifiquen la presente investigación. Por esta razón, esta tesis no sólo contribuye con nuevas estrategias y métodos de búsqueda óptima para minimizar el tiempo, sino que también formula la teoría de búsqueda Bayesiana multiagente mediante una notación única y una metodología computacionalmente viable.

2.2 Soluciones Relacionadas con la Búsqueda Óptima

En esta sección mostramos la comparación de las soluciones más cercanas al problema de decisión dentro del MTS, incluyendo los algoritmos que se proponen en esta tesis. Esta comparación no pretende ser un manual exhaustivo de todos los trabajos relacionados, sino una recopilación de los que realmente han motivado el presente trabajo. Todos los algoritmos comparados, con excepción de [Eagle, 1984], son computacionalmente tratables, es decir, calculan la solución en un tiempo aceptable (tiempo polinómico dependiente del horizonte).

Para comparar los trabajos, analizamos las siete características resumidas en la Tabla 2.1:

MTS, es la característica más importante e indica qué algoritmos resuelven directamente la búsqueda del objetivo en el menor tiempo posible. [Eagle, 1984; Bourgault et al., 2003, 2004; Mathews et al., 2007; Gan and Sukkarieh, 2010] están basados en la maximización de la detección que no implica que el tiempo que se precisa para encontrar el objetivo sea mínimo. De hecho, [Bourgault et al., 2003] comenta la posibilidad de optimizar el tiempo esperado, pero lo rechazan por problemas de complejidad. En [Yang et al., 2002] se reduce la incertidumbre ya que los autores utilizan la entropía para evaluar las decisiones de los agentes, pero esto no implica tampoco que el objetivo se encuentra en mínimo tiempo. Sólo [Stone, 1975; Sarmiento et al., 2009] y las soluciones proporcionadas en esta tesis resuelven el problema del tiempo mínimo. Sin embargo, [Stone, 1975] no tiene en cuenta las restricciones en el camino y [Sarmiento et al., 2009] sólo contempla el problema de la búsqueda de un objetivo estático con un único agente. Aún así, el trabajo de [Sarmiento et al., 2009] es interesante, ya que propone la utilización de una condición de dominancia para elegir las soluciones, aunque no se conoce su escalabilidad para abordar la configuración multiagente y objetivos dinámicos.

D/C, indica si la búsqueda esta modelada de forma continua (C) o discreta (D). En los modelos continuos, las acciones del controlador son continuas por intervalos. [Sarmiento et al., 2009] discretiza primero la región de búsqueda en un grafo de visibilidad y después, en una segunda fase, calcula las acciones continuas que garantizan la visita de esas posiciones.

Restringido, informa si el problema resuelve el problema cuando el camino de búsqueda está restringido, es decir, si el método considera que los agentes tienen restricciones dinámicas espaciales.

Multiagente, muestra si el algoritmo es aplicable para varios agentes. [Yang et al., 2002; Bourgault et al., 2003, 2004; Mathews et al., 2007; Gan and Sukkarieh, 2010] son soluciones descentralizadas cooperativas (DEC), donde los agentes intercambian información durante la etapa de decisión de forma transparente. En [Yang et al., 2002] los agentes sólo intercambian la información cuando se encuentran en celdas adyacentes, mientras que en los demás la información es transmitida en todo momento. Aunque los métodos propuestos en esta tesis están diseñados para ser finalmente implementados de forma descentralizada, no se proporciona una prueba de convergencia para todos ellos y por lo tanto, únicamente se considera la descentralización de la propuesta de la Sección 5.2.

Objeto Móvil, describe si el objetivo puede estar desplazándose durante la etapa de decisión. En [Stone, 1975; Bourgault et al., 2003, 2004; Mathews et al., 2007; Sarmiento et al., 2009; Gan and Sukkarieh, 2010] y en las Secciones 4.2 y 5.2, el objetivo se asume estático durante el horizonte de decisión. El resto contempla el movimiento del objeto buscado.

Optimalidad(Horizonte), indica si la solución es local o globalmente óptima y el horizonte de decisión, es decir, cuántos pasos hacia delante el algoritmo optimiza. La mayoría de las soluciones no son globalmente óptimas en el horizonte de decisión ya que se estancan en óptimos locales (p.e. [Gan and Sukkarieh, 2010]). Los métodos estocásticos como el presentado en la Sección 4.3 no son óptimos globalmente porque utilizan mecanismos de muestreo. Finalmente, las soluciones de [Yang et al., 2002], Sección 4.3 y Sección 5.2 utilizan una aproximación del valor esperado como heurística para evaluar las decisiones más allá del horizonte de decisión.

Tipo de Optimización, es una característica más informativa que comparativa, y describe qué método algorítmico se utiliza para resolver el problema. [Mathews et al., 2007] y [Gan and Sukkarieh, 2010] utilizan básicamente el mismo método, aunque [Gan and Sukkarieh, 2010] usa la formulación explícita de gradiente para disminuir el coste computacional. [Yang et al., 2002] utilizan el valor esperado como heurística y una red neuronal (Neural Network, NN) para aprender la posición del objetivo, y [Eagle, 1984]

implementa técnicas de programación dinámica para resolver los POMDP. Los algoritmos presentados en esta tesis se encuentran en las Secciones 4.2, 4.3 y 5.2. El primer método se basa en la programación con restricciones (Constraint Programming, CP), el segundo utiliza la optimización de entropía cruzada (Cross Entropy Optimization, CEO) y el tercer método se ha diseñado sobre una estrategia de optimización de gradiente.

TABLE 2.1: Soluciones en la Búsqueda Óptima

TRABAJO	MTS	D/C	Restringido	Multiagente	O. Móvil	Optimalidad(Horizonte)	Tipo de Optimización
[Stone, 1975]	✓	D,C				Global	Lagrangiano
[Eagle, 1984]		D	✓		✓	Global(N)	DP (POMDP)
[Bourgault et al., 2003]		C	✓			Local(1)	Voraz
[Yang et al., 2002]		D	✓	✓	✓	Local(2)+Heurística	Informado+NN
[Bourgault et al., 2004]		C	✓	✓		Local(1)	Voraz
[Mathews et al., 2007]		C	✓	✓		Local(N)	Gradiente
[Sarmiento et al., 2009]	✓	D,C	✓			Aproximación Local(N)	DFS Limitado
[Gan and Sukkarieh, 2010]		C	✓	DEC		Local(N)	Gradiente Explícito
Sección 4.2	✓	D	✓	Central		Global(N)	CP
Sección 4.3	✓	D	✓	Central	✓	Aprox. Global(N)+Heurística	CEO
Sección 5.2	✓	C	✓	DEC		Local(N)+Heurística	Gradiente

2.3 Notación

A continuación se presenta la notación empleada para definir cada elemento involucrado en el problema de MTS.

τ - Posición del objetivo.

$\hat{\tau}$ - Posición estimada del objetivo.

b_τ - Creencia que el equipo de agentes tiene sobre la posición del objetivo

s - Estado del agente definido por su posición. Por simplificación, se asume que la posición del sensor coincide con la posición del agente.

u - Acción realizada por el agente.

$v^k = \{u^k, \cdots, u^{k+N-1}\}$ - Serie de acciones planeadas con horizonte N, comenzando en el instante k.

$R(.)$ - Recompensa obtenida por los agentes.

$J(.)$ - Función de utilidad.

$P(.)$ - Probabilidad de que ocurra un evento.

z - Observación.

D - Evento de detección del objetivo. Tiene implícita la observación $z = D$.

$E\{.\}$ - Valor esperado.

Asumimos que las acciones que realizan los agentes son deterministas y que sólo existen dos tipos de observaciones: detección (D) y no-detección ($\overline{D}$). Además, cuando aparece un superíndice como en τ^k, significa que es el valor de esa variable en el instante k, y cuando se trata de un subíndice como en s_i^k, se refiere al valor del i-ésimo agente.

La Figura 2.1 muestra la trayectoria de un agente que comienza en la posición s^k y que realiza las acciones v^k para desplazarse.

2.4 Teoría de la Probabilidad en Entornos con Incertidumbre

La primera dificultad para resolver el problema de MTS es gestionar el conocimiento que los agentes tienen sobre la posición del objetivo, es decir, la información que el

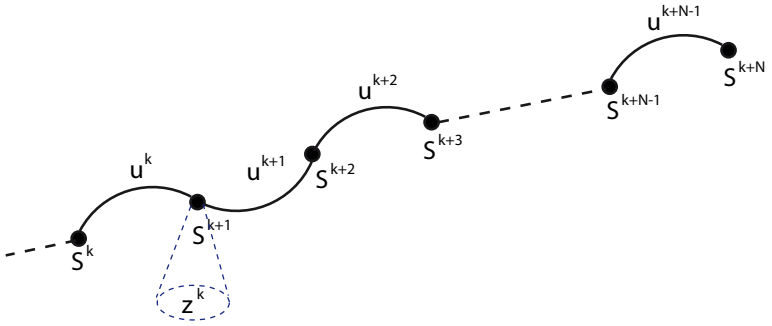
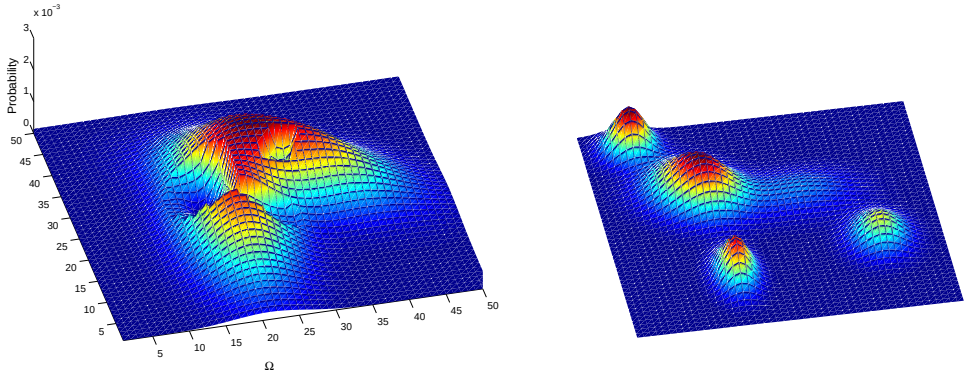


FIGURE 2.1: Vector de acciones para un agente v^k comenzando en el estado s^k.

sistema utiliza para luego elegir las acciones. Este problema ha sido ampliamente estudiado en diversos trabajos, destacando especialmente el de [Furukawa et al., 2006].

En esta sección se describen brevemente los conceptos relativos a la teoría probabilística y de la información. Su introducción se debe a que es necesario usar la información subjetiva sobre el objetivo, es decir, la información a priori de su localización y movimientos, y las medidas proporcionadas por los sensores. La mejor forma para abordar el problema es utilizar la inferencia Bayesiana [Jaynes, 2003].



(a) Distribución Genérica (b) Distribución multi-Gaussiana

FIGURE 2.2: Probabilidad de que el objetivo se encuentre en la región: la creencia de la localización del objetivo b_r^k.

La teoría Bayesiana nos posibilita comenzar con una creencia previa de la posición del objetivo y actualizarla según van llegando las observaciones de los sensores. Además, nos permite predecir la posición del objetivo de acuerdo con su modelo de movimiento probabilístico. Este razonamiento Bayesiano, es el núcleo de los sistemas de búsqueda, ya que gestiona la información tanto de la capa de fusión

sensorial como de la de decisión. La Figura 2.2 muestra dos ejemplos de la creencia de la posición del objetivo b_τ^k: (a) una función de densidad probabilística genérica y (b) un escenario con múltiples Gaussianas. El mapa de altura representa la probabilidad de que el objetivo se encuentre en cualquier posición del espacio de búsqueda, donde valores más altos implican mayores posibilidades de encontrar el objetivo.

Primero analizamos el método utilizado por la capa de fusión sensorial para estimar la posición del objetivo cuando ésta está descrita por una función de distribución genérica, y segundo, la forma en que se evalúan las decisiones de los agentes utilizando funciones probabilísticas.

2.4.1 Estimador Bayesiano Recursivo (RBE)

El Estimador Bayesiano Recursivo (RBE), [Furukawa et al., 2006], es un algoritmo que permite estimar y predecir la posición del objetivo utilizando las medidas de los sensores y el modelo de movimiento del objeto. El método itera dos pasos: actualización y predicción. Vamos a utilizar la notación de Kalman para definir la distribución o creencia; $b_\tau^{k|k}$ es la creencia actualizada con las medidas de los sensores y $b_\tau^{k+1|k}$ es la posición estimada en el siguiente instante.

Dado que las observaciones son siempre de no-detección ($z^k = \overline{D}$, [Stone, 1989]) hasta que el objetivo es descubierto y que el modelo probabilístico del movimiento esta definido por la función $P(\tau^k|\tau^{k-1})$, definimos el filtro RBE según se muestra en el Algoritmo 13.

Algorithm 13 Algoritmo RBE

Require: $b_\tau^{k|k}$

1: **for** $j = 1, N$ **do**
2: $\quad b_\tau^{k+j|k+j-1} \leftarrow \int_{\tau^{k+j-1}} P(\tau^{k+j}|\tau^{k+j-1}) b_\tau^{k+j-1|k+j-1} d\tau^{k+j-1}$
3: $\quad b_\tau^{k+j|k+j} \leftarrow \frac{1}{\eta} P(\overline{D}^{k+j}|\tau^{k+j}, s^{k+j}) b_\tau^{k+j|k+j-1}$
4: **end for**

2.4.2 Probabilidad de No-Detección en la Búsqueda de un Objetivo Estático

En esta sección se describe cómo evaluar una secuencia de observaciones y acciones de los agentes dentro de un marco probabilístico. Seguimos el planteamiento propuesto por [Bourgault et al., 2004; Mathews, 2008] para un objetivo estático. Queremos derivar la probabilidad conjunta de eventos de no-detección a lo largo de un horizonte finito de observaciones.

Para una optimización a tramos de horizonte N, las acciones de control cambian en cada instante discreto k y por lo tanto, empezando en el instante k tenemos una secuencia de observaciones $z^{k+1:k+N}$. Las acciones se definen por el vector de decisión $v^k = \{u^k, \ldots, u^{k+N-1}\}$. Asumimos que el objetivo es estático durante el horizonte de decisión y que las observaciones son independientes. La probabilidad conjunta de eventos de no-detección en el horizonte de decisión es:

$$P(\bigcap_{j=1:N} z^{k+j} = \overline{D}|\tau, s^{k+1:k+N}) = P\left(z^{k+1:k+N} = \overline{D}|\tau, s^{k+1:k+N}\right) = \\ = \prod_{j=1}^{N} P\left(z^{k+j} = \overline{D}|\tau, s^{k+j}\right) \tag{2.1}$$

Marginalizando sobre la posición del objetivo τ y multiplicando por la creencia del objetivo b_τ^k, obtenemos la función de utilidad J_{nd} que es un escalar que describe la probabilidad conjunta de no-detección [Gan and Sukkarieh, 2010]:

$$J_{nd}(s^k, v^k, b_\tau) = P(z^{k+1:k+N} = \overline{D}|z^{1:k}, s^{1:k+N}) = \\ = \int_\tau \prod_{j=1}^{N} P\left(z^{k+j} = \overline{D}|\tau, s^{k+j}\right) b_\tau^k d\tau \tag{2.2}$$

2.4.3 Extensión a un Sistema Multiagente

Cuando se dispone de un equipo de q agentes el vector de decisión es $v_{1:q}^k = \{v_1^k, \cdots, v_q^k\}$ y por tanto, los estados de los agentes son $s_{1:q}^k$. La fase de actualización del RBE se transforma en:

$$b_\tau^{k|k} = \frac{1}{\eta} \prod_{i=1}^{q} P\left(\overline{D}_i^k|\tau^k, s_i^k\right) b_\tau^{k|k-1} \tag{2.3}$$

La probabilidad conjunta de no-detección para múltiples agentes es:

$$J_{nd}(s_Q^k, v_Q^k, b_\tau^k) = \int_\tau \prod_{j=1}^{N} \prod_{i=1}^{q} P(\overline{D}_i^{k+j}|\tau, s^{k+j}) b_\tau^k d\tau \tag{2.4}$$

2.4.4 Extensión a un Sistema Descentralizado

Es posible descentralizar el RBE, así como la función de utilidad siguiendo los métodos propuestos por [Bourgault et al., 2004; Furukawa et al., 2006; Mathews, 2008; Gan and Sukkarieh, 2010].

2.5 Casos Especiales

Existen dos casos particularmente interesantes relacionados con la fusión multisensorial. El primero es cuando la posición del objetivo puede modelarse por una distribución Gaussiana. Gracias a Kalman [Mutambara, 1998], es posible encontrar una solución óptima en el caso lineal y una solución subóptima en el no-lineal, para estimar la posición del objetivo. El segundo caso es un problema que ocurre cuando las observaciones de los agentes llegan con retraso a la capa de fusión sensorial. Este problema, conocido como el de las medidas desordenadas (Out-Of-Sequence, OOS) [Bar-Shalom and Chen, 2005], debe ser considerado en cualquier problema multisensorial como en el MTS que se plantea en esta tesis.

2.6 Diseño del Sistema de Búsqueda

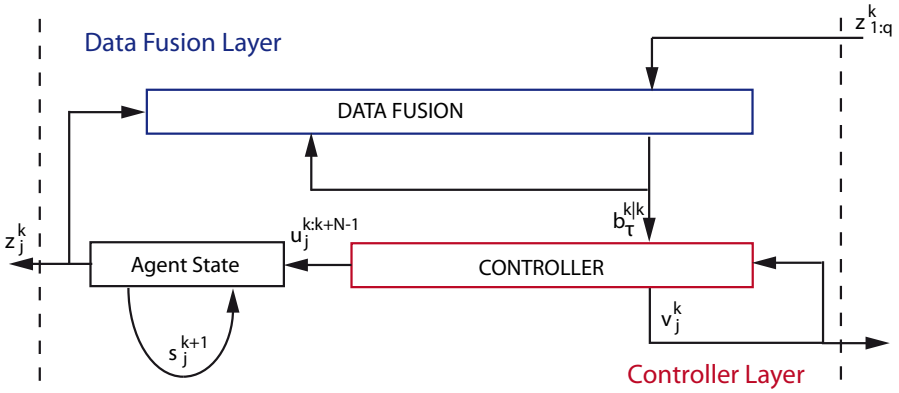


FIGURE 2.3: Diseño General del Sistema.

Como se ha expuesto previamente, existen dos procesos involucrados en la búsqueda: la gestión de la información o la capa de fusión sensorial, y el algoritmo de decisión o controlador. Estas dos capas relacionadas deben combinarse con el agente para el diseño del sistema de búsqueda. Esto se puede realizar como se propone en [Mathews, 2008] donde una capa se encarga de actualizar la información real proporcionada por los sensores y otra capa calcula las acciones de los agentes utilizando un modelo de predicción para inferir la posición del objetivo. En la

práctica, la capa de fusión sensorial está implementada, por ejemplo, por el algoritmo RBE [Furukawa et al., 2006] (Sección 2.4.1) y la capa de control está administrada por un algoritmo de optimización. La Figura 2.3 muestra el diseño abstracto del sistema autónomo para un agente. La capa de fusión sensorial recibe las nuevas observaciones de los agentes, y actualiza y predice la posición del objetivo iterativamente. Esta información actualizada alimenta la capa de control que calcula las mejores acciones cooperando con los demás agentes. La salida del controlador son las acciones que el agente debe ejecutar. Realizando estas acciones cada agente se mueve en la región de búsqueda obteniendo nuevas observaciones que vuelven a alimentar la capa de fusión sensorial, cerrando el ciclo del sistema.

2.7 Resumen

Este capítulo describe los elementos sobre los se sustenta este trabajo de investigación, explicando algunos conceptos básicos que serán utilizados a lo largo de la tesis. La revisión histórica muestra el MTS como un problema de gran importancia y resalta la necesidad de desarrollar nuevas estrategias y métodos capaces de minimizar el tiempo de detección del objetivo. El análisis de los trabajos relacionados muestra que todavía existen problemas no resueltos que se plantean en esta tesis, como la búsqueda de objetos en movimiento en el menor tiempo posible utilizando múltiples agentes provistos de sensores.

Se ha explicado brevemente cómo se predice y se actualiza la información del objetivo y cómo se pueden evaluar las acciones de los agentes. Además se han mencionado casos particulares interesantes como el problema de las medidas desordenadas (OOS).

Finalmente se ha mostrado el diseño general del sistema utilizado como modelo en el presente trabajo. Este diseño contempla una capa de fusión sensorial que maneja la información y un controlador que optimiza las acciones de los agentes.

Chapter 3

BÚSQUEDA DE TIEMPO MÍNIMO

"We must use time as a tool, not as a crutch." JFK

3.1 Qué es MTS

Como se ha indicado previamente, la búsqueda de tiempo mínimo (MTS) consiste en encontrar un objeto que está posicionado en algún lugar del "espacio", en el menor tiempo posible. En el fondo, se trata de tomar las decisiones óptimas utilizando la información con la que contamos teniendo en cuenta que se quiere minimizar el tiempo. Es decir, el equipo de agentes tiene que optimizar sus acciones para encontrar el objetivo dentro de un tiempo con tendencia al mínimo.

Nos enfrentamos al MTS como una tarea de recolección de información donde toman parte dos elementos dinámicos: los agentes móviles con capacidades sensoriales y el objeto que se busca. El objeto u objetivo está contenido en una región finita y delimitada, su posición es incierta y se mueve independientemente de las acciones de los agentes. Los agentes son capaces de moverse, de acuerdo con su dinámica, por la región de búsqueda, realizar observaciones, detectar el objetivo y comunicarse. Los agentes se encuentran en una su posición inicial fija y cuentan con una información probabilística inicial sobre la dinámica y la posición del objetivo.

La solución al MTS se ha diseñado como un proceso de decisión donde: 1) cada vez que se realiza una acción se produce una nueva observación que cambia la creencia de la posición del objetivo; 2) la ocurrencia de las acciones en el tiempo

importa, es decir, el orden de las acciones es importante; y 3) la tarea termina cuando el objetivo se detecta.

Definiendo el tiempo para encontrar el objetivo como una variable aleatoria T [Stone, 1975], la solución al problema del MTS consiste en minimizar el tiempo esperado para encontrar el objetivo antes del instante k:

$$\min(E\{T \leq k\}) \tag{3.1}$$

El problema de optimización para un equipo de agentes es encontrar la mejor secuencia de acciones $v_{1:q}$ que cumplen:

$$\arg\min_{v_{1:q}}(E\{T \leq k\}) \tag{3.2}$$

Como se muestra en [Trummel and Weisinger, 1986], minimizar el tiempo esperado es similar a maximizar la probabilidad de detección. Sin embargo, cuando el camino de búsqueda está restringido [Eagle, 1984] esto no es siempre cierto. La proposición 3.1 ilustra un ejemplo en el que la equivalencia no se cumple. De hecho sólo maximizando el *incremento* de la probabilidad podemos garantizar la minimización del tiempo.

Proposition 3.1. *Maximizar la probabilidad de detección del objetivo en una secuencia de observaciones (el camino del agente) no minimiza el tiempo para descubrir el objetivo.*

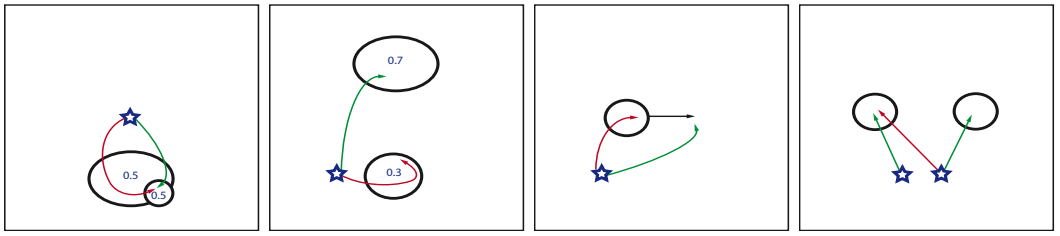
Proof. Utilizamos un contraejemplo para demostrarlo. Dada una región de búsqueda de 2×2 posiciones adyacentes, etiquetadas como $\{1, 2, 3, 4\}$, asumiendo que la probabilidad inicial de la posición del objetivo en cada posición es $\{0, 0, 1, 0\}$ y que buscamos un objetivo estático, el agente que sigue el camino $\{1, 3, 2, 4\}$ encontrará el objeto antes que siguiendo el camino $\{1, 2, 3, 4\}$. Esto ocurre porque eligiendo el primer camino se visita la posición con probabilidad 1 antes. Ya que ambos caminos tienen la misma probabilidad de detección[1], podemos afirmar que maximizar la probabilidad no minimiza necesariamente el tiempo para detectar el objetivo. □

Un análisis más detallado de esta proposición se puede encontrar en la Sección 3.2. Por lo tanto, el MTS con camino restringido debe ser considerado como un problema diferenciado dentro de la búsqueda óptima con sus propias estrategias, ya que los métodos previos no son válidos.

[1] En este caso, la probabilidad de detección es 1) la suma de las probabilidades de los lugares visitados y 2) igual a 1 para ambos caminos.

3.1.1 Escenarios Representativos en MTS

En esta sección describimos cuatro escenarios ilustrativos cuyo objetivo estriba en poder resolver el problema del MTS. Con estos ejemplos mostramos algunos de los aspectos que deben ser abordados en esta tesis: la importancia del tiempo, los problemas de utilizar estrategias miopes, los problemas relacionados con el movimiento del objetivo, y los beneficios de utilizar múltiples agentes.



(a) Tiempo Mínimo (b) No-miope (c) Objeto Dinámico (d) Múltiples Agentes

FIGURE 3.1: Escenarios Ilustrativos.

La Figura 3.1 muestra los ejemplos ilustrativos de forma esquemática. Las estrellas azules representan los agentes y las flechas rojas que nacen en ellos muestran trayectorias malas, mientras que las flechas verdes representan las trayectorias correctas. Las elipses negras muestran las regiones iniciales donde el objetivo puede estar, es decir, las zonas de probabilidad no nula, y los números dentro de las elipses indican la probabilidad total de esa región. Además, la flecha negra que comienza en la elipse (Figura 3.1(c)) especifica el movimiento del objetivo.

El primer ejemplo, Figura 3.1(a), describe una distribución no uniforme con una anomalía grande cuya probabilidad total es de 0.5. Dependiendo del instante en que observemos la anomalía, el tiempo de detección varía considerablemente. Por lo tanto, el orden de las acciones es importante. La trayectoria verde es la mejor de las posibles para el MTS ya que acumula antes las probabilidades, mientras que la roja muestra una trayectoria que encuentra el objetivo más tarde. Sólo las estrategias que tienen en cuenta el tiempo producen soluciones como la trayectoria verde. Además, podemos observar que las estrategias voraces también fallarían. De hecho todas las soluciones que maximizan la probabilidad de detección como [Eagle, 1984; Yang et al., 2002; Bourgault et al., 2004; Lavis et al., 2008; Gan and Sukkarieh, 2010] producen comportamientos erróneos en este escenario.

El segundo ejemplo, Figura 3.1(b), explica los problemas de localidad de las soluciones. Cuando se utiliza un horizonte de decisión limitado para calcular las decisiones, no podemos ver el problema completo y por lo tanto, nuestras soluciones son miopes. En este escenario, la posición inicial del agente está cerca de una región de baja probabilidad; por tanto, si el horizonte de decisión es pequeño, el

agente pensará que lo mejor es observar la región más próxima, cuando la mejor opción es ir hacia la región lejana, ya que se acumulará más probabilidad. Para reducir la miopía podemos calcular horizontes más largos (idealmente horizontes infinitos) o utilizar una función que calcula el valor esperado futuro. Soluciones como [Bourgault et al., 2003; Mathews et al., 2007; Gan and Sukkarieh, 2010] no son capaces de abordar este tipo de escenarios.

El tercer ejemplo, Figura 3.1(c), muestra el problema que aparece al buscar un objetivo que se mueve. Las mejores acciones son aquellas que anticipan la posición futura del objetivo. La trayectoria roja describe una solución que no predice la posición del objetivo en instantes futuros, mientras que la verde describe una decisión mejor, ya que utiliza la dinámica del objetivo para planear hacia adelante. Los métodos que asumen un objetivo estático como [Blum et al., 2003; Mathews, 2008; Gan and Sukkarieh, 2010; Sarmiento et al., 2009] no tienen éxito en este tipo de escenarios.

El último escenario, Figura 3.1(d), muestra los beneficios de utilizar un equipo de agentes. En este caso tenemos dos agentes y dos regiones con igual probabilidad de encontrar al objetivo. La solución óptima para el MTS es que cada agente se desplace para observar una región diferente, dividiendo el esfuerzo de búsqueda y consiguiendo detectar el objetivo antes. Métodos como [Eagle and Yee, 1990; Sarmiento et al., 2009] no pueden manejar este tipo de escenarios.

3.2 Análisis de Optimalidad de la Estrategia de Máxima Detección para Minimizar el Tiempo: La Condición Necesaria

En esta sección mostramos matemáticamente en qué casos maximizar la probabilidad de detección si minimiza el tiempo de búsqueda y aportamos la condición necesaria para que esto ocurra.

Para este análisis, definimos $P(k, v)$ como la probabilidad de detección hasta el instante k cuando el plan de decisión es v. Cuando el plan de decisión es óptimo v^* la probabilidad de detección es también óptima $P(k, v^*)$. Entonces, para un plan de N pasos, esta probabilidad es máxima cuando se compara con el resto de soluciones:

$$P(N, v^*) \geq P(N, v) \tag{3.3}$$

Por otra parte, el tiempo esperado de detección $\mu(N, v)$ en el intervalo $(1, N)$ para un plan dado v está definido por la siguiente ecuación [Stone, 1975]:

$$\mu(N, v) = E(1 \leq T \leq N) = \sum_{k=1}^{N}(1 - P(k, v)) \tag{3.4}$$

Cuando el plan es óptimo, su valor es mínimo comparado con cualquier otra solución:

$$\sum_{k=1}^{N}(1 - P(k, v^*)) \leq \sum_{k=1}^{N}(1 - P(k, v)) \tag{3.5}$$

Queremos analizar cuándo el hecho de maximizar la probabilidad de detección minimiza el tiempo, por tanto, queremos evaluar cuándo la siguiente ecuación es cierta:

$$P(N, v^*) \geq P(N, v) \overset{?}{\Longrightarrow} \sum_{k=1}^{N}(1 - P(k, v^*)) \leq \sum_{k=1}^{N}(1 - P(k, v)) \tag{3.6}$$

Para hacerlo, primero operamos la parte derecha de la ecuación 3.6 agrupando los términos,

$$N - \sum_{k=1}^{N} P(k, v^*) \leq N - \sum_{k=1}^{N} P(k, v) \tag{3.7}$$

A continuación eliminamos N y reorganizamos la expresión,

$$\sum_{k=1}^{N} P(k, v^*) \geq \sum_{k=1}^{N} P(k, v) \tag{3.8}$$

Luego separamos el último término (N) de ambas sumas,

$$P(N, v^*) + \sum_{k=1}^{N-1} P(k, v^*) \geq P(N, v) + \sum_{k=1}^{N-1} P(k, v) \tag{3.9}$$

Y finalmente reagrupamos los términos de nuevo obteniendo una condición necesaria para minimizar el tiempo esperado,

$$P(N, v^*) - P(N, v) \geq \sum_{k=1}^{N-1} P(k, v) - \sum_{k=1}^{N-1} P(k, v^*) \tag{3.10}$$

Conviene resaltar que la parte izquierda de la desigualdad es la diferencia entre la probabilidad de detección óptima y la no óptima.

Si las ecuaciones 3.10 y 3.3 se cumplen, entonces la ecuación 3.6 se satisface. Así que analizamos los diferentes subcasos:

1. Si $\sum_{k=1}^{N-1} P(k,v) \leq \sum_{k=1}^{N-1} P(k,v^*)$ la parte derecha de la ecuación 3.10 es negativa y como la ecuación 3.3 es una premisa, la ecuación 3.6 es cierta.
2. Si $\sum_{k=1}^{N-1} P(k,v) > \sum_{k=1}^{N-1} P(k,v^*)$ sólo podemos garantizar la satisfacibilidad de la ecuación 3.6 si la ecuación 3.10 se cumple. En cualquier otro caso la ecuación 3.6 es falsa.

Siguiendo el segundo subcaso, si la ecuación 3.10 no se satisface, la ecuación 3.6 no es cierta:

$$P(N,v^*) - P(N,v) < \sum_{k=1}^{N-1} P(k,v) - \sum_{k=1}^{N-1} P(k,v^*) \implies \neg\text{ecuación 3.6} \tag{3.11}$$

Por lo tanto, podemos afirmar que maximizar la probabilidad de detección no implica minimizar el tiempo esperado de detección.

Corollary 3.2. *La condición necesaria general que debe satisfacerse para minimizar el tiempo esperado es:*

$$\forall i \in (1,N) \quad \sum_{k=N-i+1}^{N} P(k,v^*) - \sum_{k=N-i+1}^{N} P(k,v) \geq \sum_{k=1}^{N-i} P(k,v) - \sum_{k=1}^{N-i} P(k,v^*) \tag{3.12}$$

En el MTS donde los agentes tienen dinámica restringida (camino restringido) no podemos garantizar que esta condición se cumpla maximizando la probabilidad, ya que depende del escenario. Por lo tanto concluimos que, si $\exists i$ tal que al maximizar $P(N,v)$ la ecuación 3.12 no se cumple, no se minimiza el tiempo de detección maximizando la probabilidad de detección.

3.3 Búsqueda Multiagente Bayesiana para Objetivos Dinámicos

En la Sección 2.4.2 del capítulo anterior, hemos mostrado cómo calcular la probabilidad conjunta de no-detección para las trayectorias de los múltiples agentes que buscan un objeto estático. Aquí, generalizamos el proceso para un objetivo dinámico mostrando las expresiones probabilísticas generales que obtienen la probabilidad de detectar/no-detectar el objetivo en cualquier instante. Estas funciones nos permitirán diseñar las estrategias específicas para minimizar el tiempo.

Recordamos que el movimiento del objetivo esta definido por la probabilidad de transición $P(\tau^{k+1}|\tau^k)$ y que la probabilidad de que el sensor detecte el objetivo está descrita por $P(D^k|s^k,\tau^k)$.

3.3.1 Probabilidad Conjunta de Eventos de Detección

Describimos la función natural para evaluar la trayectoria de búsqueda para un grupo de agentes. La probabilidad conjunta de detección para q agentes desde el instante k al $k+N$ se calcula de la siguiente forma:

$$J_d(s_{1:q}^{k+1:k+N}) = P(\bigcup_{j=1:N,i=1:q} D_i^{k+j}|s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) = \tag{3.13}$$

$$= \sum_{j=1}^{N}\int_{\tau^{k+j}}\left[1-\prod_{i=1}^{q}P(\overline{D}_i^{k+j}|s_i^{k+j},\tau^{k+j})\right]\tilde{b}_\tau^{k+j|k+j-1}d\tau^{k+j} \tag{3.14}$$

El cómputo de $\tilde{b}_\tau^{k+j|k+j-1}$ se realiza calculando recursivamente su valor desde $j=2$ hasta $j=N$, utilizando la ecuación 3.16 y considerando que $\tilde{b}_\tau^{k+1|k+0} = b_\tau^{k+1|k}$ (la creencia de la posición del objetivo en el instante $k+1$ con todas las observaciones asimiladas hasta k).

$$\tilde{b}_\tau^{k+j|k+j-1} = P(\tau^{k+j}, \overline{D}_{1:q}^{k+1:k+j-1}|s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) = \tag{3.15}$$

$$= \int_{\tau^{k+j-1}} P(\tau^{k+j}|\tau^{k+j-1})\prod_{i=1}^{q}P(\overline{D}_i^{k+j-1}|\tau^{k+j-1}, s_i^{k+j-1})\tilde{b}_\tau^{k+j-1|k+j-2}d\tau^{k+j-1} \tag{3.16}$$

En el caso particular de que el objetivo sea estático ($\tau^{k+1}=\tau^k=\tau$) la predicción desaparece y entonces la ecuación 3.14 se convierte en:

$$J_d(s_{1:q}^{k+1:k+N}) = \int_\tau \sum_{j=1}^{N}\left[1-P(\overline{D}_{1:q}^{k+j}|\tau, s^{k+j})\right]\prod_{l=1}^{j-1}\prod_{i=1}^{q}P(\overline{D}_i^{k+l}|\tau, s^{k+l})b_\tau^k d\tau \tag{3.17}$$

3.3.2 Probabilidad Conjunta de Eventos de No-Detección

Análogamente, podemos calcular la probabilidad de no-detección para un grupo de agentes, en este caso computando la intersección de los eventos de no-detección:

$$J_{nd}(s_{1:q}^{k+1:k+N}) = P(\overline{D}_{1:q}^{k+1:k+N}|s_{1:q}^{1:k+N}, z_{1:q}^{1:k}) = \tag{3.18}$$

$$= \int_{\tau^{k+N}} \prod_{i=1}^{q} P(\overline{D}_i^{k+N}|s_i^{k+N}, \tau^{k+N})\tilde{b}_\tau^{k+N|k+N-1} d\tau^{k+N} \tag{3.19}$$

Nótese que sustituyendo N por cualquier instante j, estamos calculando la no-detección conjunta hasta j.

En el caso de que el objetivo sea estático la función se convierte en la propuesta por [Bourgault et al., 2004; Mathews, 2008], ya mostrada en la Sección 2.4.2:

$$J_{nd}(s_{1:q}^{k+1:k+N}) = \int_\tau \prod_{j=1}^{N} \prod_{i=1}^{q} P(\overline{D}_i^{k+j}|s_i^{k+j}, \tau) b_\tau^k d\tau \tag{3.20}$$

3.4 Estrategias de Decisión Tratables para el MTS

Esta sección trata sobre el desarrollo de estrategias y funciones de utilidad computacionalmente tratables[2] para solucionar el problema de decisión del MTS. En la práctica, estas estrategias serán utilizadas por la capa del controlador, representada en la Figura 2.3, para calcular las acciones óptimas de los agentes. Nuestro punto de partida es el trabajo realizado por [Mathews, 2008] quien postula, que asumiendo que el agente nunca detecta el objetivo [Stone, 1989], podemos predecir la posición incierta del objetivo de forma determinista utilizando inferencia Bayesiana, y por tanto, simplificar el problema general (POMDP) en un problema determinista. Así, usaremos un controlador de modelo predictivo [Bertsekas, 1995] como la solución para la capa de control. La Figura 3.2 muestra el proceso de detección durante la toma de decisiones, donde vemos que sólo hay un camino para la evolución de la posición del objetivo. Por lo tanto, no tenemos que calcular el valor esperado de la función a optimizar.

Asumiendo que se trata de un proceso de decisión determinista podemos estimar el estado del objetivo $\tilde{b}_\tau^k$ en función de las decisiones tomadas, y evaluar las decisiones de los agentes utilizando las ecuaciones presentadas en la Sección 3.3. La función de utilidad, computacionalmente tratable para el controlador, es la siguiente:

[2] El concepto computacionalmente tratable se refiere a que la toma de decisiones se realiza en un tiempo polinómico independientemente de la escala del problema. Es decir, que el cálculo de las acciones de los agentes es eficiente en términos de tiempo de computo. En caso contrario nos referiremos a computacionalmente intratable.

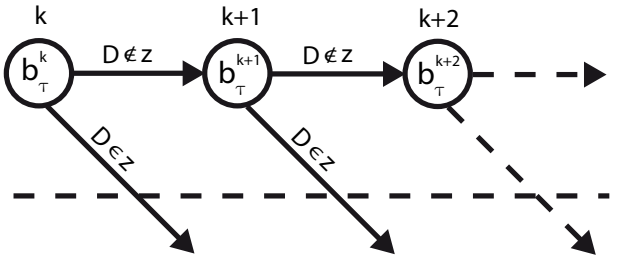


FIGURE 3.2: Creencia de la posición del objetivo a lo largo de las acciones del agente (extraído de [Mathews, 2008]).

$$J(s^k, v^k, b_\tau^k) \triangleq \sum_{j=k}^{k+N} R(u^j, \tilde{b}_\tau^j) + \hat{H}^{k+N}(\tilde{b}_\tau^{k+N}) \tag{3.21}$$

Donde $R(u^k, \tilde{b}_\tau^j)$ es la recompensa que se obtendría en cada instante y $\hat{H}^{k+N}(\tilde{b}_\tau^{k+N})$ es el valor esperado de la función después del instante $k+N$. Utilizando esta aproximación podemos diseñar un modelo de decisión tratable que depende de la dinámica del sistema y la distribución inicial.

Para solucionar el MTS buscamos estrategias que hagan que los agentes observen cuanto antes las regiones con alta probabilidad de encontrar el objetivo, es decir, que minimicen el tiempo que los agentes emplean para encontrar el objetivo.

3.4.1 Estrategias

Proponemos tres estrategias principales para abordar el problema del MTS. La primera utiliza una versión de horizonte limitado del tiempo esperado. La segunda utiliza la idea de aplicar una función de descuento a la probabilidad de detección que depende del tiempo. La tercera y última, plantea como modelar una heurística para crear estrategias no miopes. Aparte, comentamos otras dos estrategias útiles cuando la posición del objetivo está modelada por una distribución Gaussiana.

3.4.1.1 Minimizar el Tiempo Esperado Local

Para reducir el tiempo de la búsqueda, podemos minimizar el tiempo esperado para encontrar el objetivo. Definiendo el tiempo para encontrar el objeto como una variable aleatoria T, podemos calcular la función de densidad $P(T \leq k)$ para

cualquier instante $k \geq 1$. El tiempo esperado, o el valor esperado de T es [Papoulis and Pillai, 2002; Stone, 1975]:

$$E\{T\} = \sum_{k=1}^{\infty} (1 - P(T \leq k)) \tag{3.22}$$

El cálculo del tiempo esperado para infinitos términos es computacionalmente intratable, pero teniendo en cuenta la propiedad descrita por [Feller, 1966], donde la ecuación 3.22 puede ser utilizada para funciones de densidad que sumen menos de uno, podemos seguir utilizándola para una ventana de decisión con un horizonte limitado de N instantes[3]. Por lo tanto, podemos calcular el Tiempo Esperado Local (Local Expected Time, LET) como:

$$LET\{1 \leq T \leq N\} = \sum_{k=1}^{N} (1 - P(1 \leq T \leq k)) \tag{3.23}$$

Para calcularlo, nótese que tenemos un grupo de agentes que toman decisiones utilizando su conocimiento sobre el mundo $b_{\tau}^{k|k}$ y que realizan observaciones. El plan de decisión para un horizonte N es $v_{1:q}^{k} = \{u_{1:q}^{k}, \cdots, u_{1:q}^{N-1}\}$. Por definición, la probabilidad de encontrar el objetivo antes del instante $k+j$, $P(k \leq T \leq k+j)$, es equivalente a la probabilidad de detectar el objeto en el intervalo de tiempo $(k+1, k+j)$, es decir, la probabilidad de la unión de los eventos de detección:

$$P(k \leq T \leq k+j) = P(\bigcup_{l=k}^{k+j} \bigcup_{i=1}^{q} D_i^{k+l} | s_{1:q}^{k}, u_{1:q}^{k:k+j-1}, z_{1:q}^{1:k}) \tag{3.24}$$

Por lo tanto, calculamos LET para un plan de decisión dado $v_{1:q}^{k}$ como la media $\mu(v_{1:q}^{k})$:

$$\mu(s_{1:q}^{k}, v_{1:q}^{k}) = \sum_{j=1}^{N} \left(1 - P(\bigcup_{l=k}^{k+j} \bigcup_{i=1}^{q} D_i^{k+l} | s_{1:q}^{k}, u_{1:q}^{k:k+j-1}, z_{1:q}^{1:k}) \right) \tag{3.25}$$

Asumiendo que las acciones son deterministas tenemos que $\{s_{1:q}^{k}, u_{1:q}^{k+1:k+N}\}$ es

[3] Otras funciones para el calculo de la esperanza aplicadas para la búsqueda óptima, no tienen esa propiedad, así que necesitan que la función de densidad sume 1. Por ejemplo, las siguientes funciones, extraídas de [Trummel and Weisinger, 1986; Bourgault et al., 2003], no pueden calcular el LET:

$$E\{T\} = \sum_{k=1}^{\infty} k \left(P(T \leq k) - P(T \leq k-1) \right) = \sum_{k=1}^{\infty} k P(T = k)$$

.

equivalente a $\{s_{1:q}^{k+1:k+N}\}$. Además, en vez de utilizar el evento de detección, podemos usar el evento de no-detección $\overline{D}_i^k$, que es su complementario, obteniendo:

$$\mu(s_{1:q}^k, v^k) = \sum_{j=1}^{N} P(\overline{D}_{1:q}^{k+1:k+j} | s_{1:q}^{k+1:k+N}, z_{1:q}^{1:k}) \tag{3.26}$$

Asumiendo que siempre se produce no-detección dentro del horizonte de decisión podemos calcular recursivamente la probabilidad conjunta de no-detección, marginalizando con respecto a los estados del objetivo $\tau^{k+1:k+N}$ (Sección 3.3):

$$P(\overline{D}_{1:q}^{k+1:k+j} | s^{1:k+N}, z_{1:q}^{1:k}) = \int_{\tau^{k+j}} \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s_i^{k+j}, \tau^{k+j}) \tilde{b}_\tau^{k+j|k+j-1} d\tau^{k+j} \tag{3.27}$$

Desarrollando esta ecuación llegamos al mismo resultado que aplicar recursivamente la predicción y asimilación Bayesiana sin factor de normalización (Sección 3.3), donde calculamos $\tilde{b}_\tau^{k+j|k+j-1}$, y así llegamos a la función de utilidad final para calcular el LET:

$$LET = \mu(s_{1:q}^k, v_{1:q}^k, b_\tau^{k|k}) = \sum_{j=1}^{N} \int_{\tau^{k+j}} \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s_i^{k+j}, \tau^{k+j}) \tilde{b}_\tau^{k+j|k+j-1} d\tau^{k+j} \tag{3.28}$$

Para calcular $\tilde{b}_\tau^{k+j|k+j-1}$ utilizamos las siguientes ecuaciones:

$$\tilde{b}_\tau^{k+j|k+j-1} = \int_{\tau^{k+j-1}} P(\tau^{k+j} | \tau^{k+j-1}) \tilde{b}_\tau^{k+j-1|k+j-1} d\tau^{k+j-1} \tag{3.29}$$

$$\tilde{b}_\tau^{k+j|k+j} = \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s_i^{k+j}, \tau^{k+j}) \tilde{b}_\tau^{k+j|k+j-1} \tag{3.30}$$

donde la creencia a priori de la posición del objetivo se utiliza en el primer paso de la recursión ($\tilde{b}_\tau^{k|k} = b_\tau^{k|k}$).

La minimización de LET proporciona las acciones para que los agentes encuentren el objetivo antes.

3.4.1.2 Máxima Recompensa de Tiempo Descontado

Aparte de minimizar el tiempo esperado, también queremos incluir la posibilidad de modelar el tiempo. Sabemos por la Sección 3.2 que maximizando la probabilidad conjunta de detección no se optimiza necesariamente el tiempo. Además,

queremos valorar más el hecho de que los agentes visiten regiones con alta probabilidad cuanto antes. Así que podemos asignar pesos a las recompensas (probabilidades) obtenidas por los agentes por medio de una función de descuento, al igual que se realiza en las ecuaciones de Bellman [Bellman, 1957] o en [Blum et al., 2003]. De esta forma, el cálculo del plan óptimo de acciones se convierte en el problema de máxima Recompensa de Tiempo Descontado (DTR). La función de descuento puede ser cualquier función decreciente que modele la importancia del tiempo y la recompensa es la probabilidad conjunta de detección en cada instante.

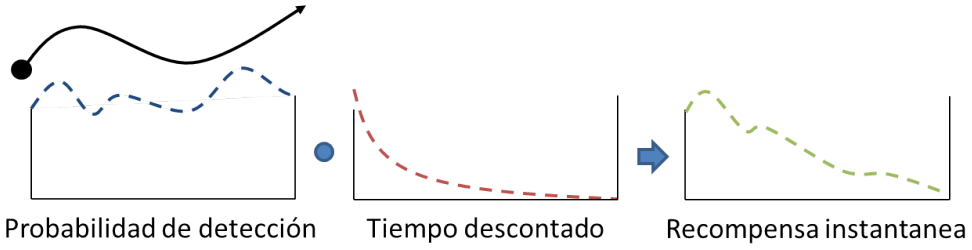


FIGURE 3.3: Estrategia DTR.

La Figura 3.3 muestra una representación gráfica sobre cómo la probabilidad de detección del objetivo para un plan de acciones o trayectoria (representado por el punto de color negro y la flecha curvada) se modifica por la función de descuento. En la imagen izquierda, podemos observar la probabilidad de detección calculada a lo largo de la trayectoria. La función de descuento, en la parte central, asigna pesos específicos obteniendo la representación mostrada en la parte derecha, donde las recompensas decrecen con el tiempo. La recompensa descontada acumulada corresponde a la función de utilidad que debe ser maximizada.

Definiendo la función de tiempo descontado $f(k)$ como un parámetro de descuento λ, fijado previamente, que decrece exponencialmente con el tiempo k:

$$f(k) = \lambda^k \;\mid\; 0 \leq \lambda \leq 1 \tag{3.31}$$

Utilizando la probabilidad conjunta de detección (ecuación 3.14) la estrategia multiagente DTR se formula de la siguiente forma:

$$\begin{aligned} DTR &= J_{dtr}(s^k_{1:q}, v^k_{1:q}, b^{k|k}_\tau) = \\ &= \sum_{j=1}^{N} \lambda^{j-1} \int_{\tau^{k+j}} \left[1 - \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s^{k+j}_{1:q}, \tau^{k+j}) \right] \tilde{b}^{k+j|k+j-1}_\tau d\tau^{k+j} \end{aligned} \tag{3.32}$$

Maximizar DTR garantiza un plan óptimo restringido por la función de descuento para un horizonte N.

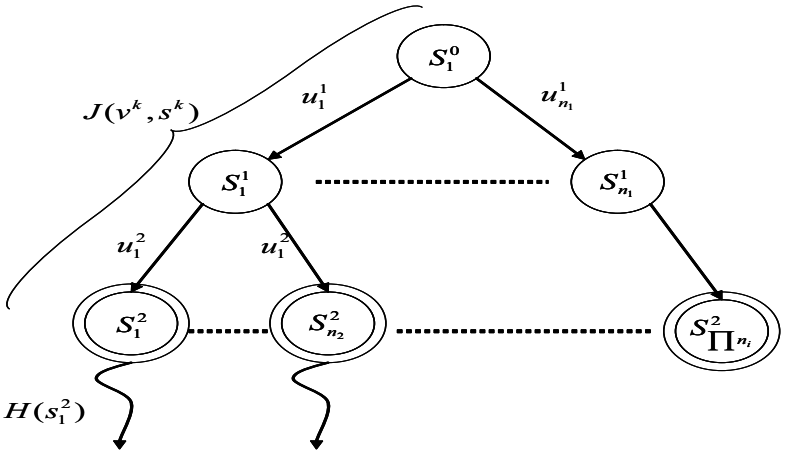


FIGURE 3.4: Árbol de decisión con una heurística asociada a los nodos terminales. La función de utilidad $J(v^k)$ se hace informada utilizando una función que estima la recompensa futura $H(s^{k+N})$.

3.4.1.3 Heurística de Tiempo Descontado

Esta estrategia se basa en aproximar el valor futuro esperado de la función de utilidad por medio de una heurística. Esta solución es fácilmente combinable con las estrategias previamente explicadas para reducir la miopía. La idea es calcular la recompensa a corto plazo con cualquier función probabilística e incluir una heurística que dependa del tiempo como recompensa terminal. Dentro del MTS, el valor esperado de la función en el futuro es la observación esperada (o la no-detección esperada).

Dado cualquier plan de acciones $v_{1:q}^k$, los agentes transitan del estado $s_{1:q}^k$ al $s_{1:q}^{k+N}$ por medio de su modelo de movimiento, proporcionando la información para calcular recompensa a corto plazo. Incorporamos las recompensas futuras en el proceso de optimización calculando la observación esperada en el estado terminal s_i^{k+N}. Por tanto, necesitamos una función que evalúe la recompensa de elegir $v_{1:q}^k$ y una función que estime la recompensa futura de estar en $s_{1:q}^{k+N}$. Esta estimación se basa en inferir cuánta probabilidad de detección los agentes podrían obtener estando en ese estado. Por lo tanto, la heurística propuesta aproxima la observación futura de los agentes. Para calcularla se asume que el objetivo es estático después del instante $k + N$.

La recompensa esperada o la observación esperada está definida como la recompensa que los agentes esperan obtener en el futuro por terminar en el estado $s_{1:q}^{k+N}$ con una creencia de la posición del objetivo $\tilde{b}_\tau^{k+N}$. Calcular este valor esperado es tan costoso como resolver el problema original. Por ello, utilizamos una heurística que aproxima este valor. El planteamiento consiste en desarrollar la heurística

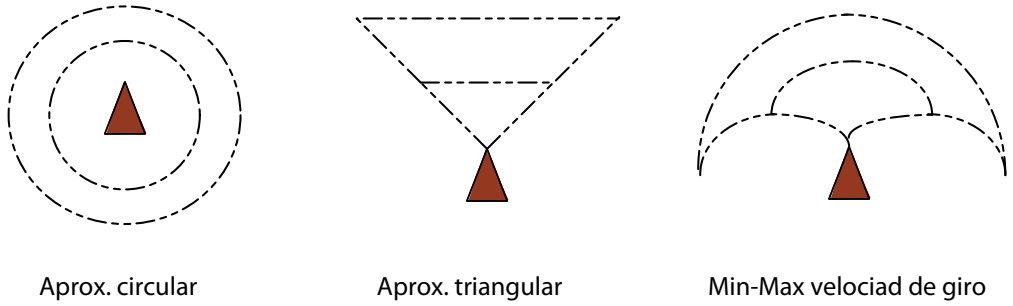


FIGURE 3.5: Posibles regiones de alcance para la heurística.

$H(s^k_{1:q}, v^k_{1:q}, b^k_\tau)$ que evalúa el estado del agente y añade esta contribución a la función de utilidad a corto plazo, como se describe en la Figura 3.4.

Para construir la heurística podemos aproximar el espacio alcanzable por el agente en formas geométricas regulares [Yang, 2005]. La Figura 3.5 muestra tres aproximaciones diferentes de representación del espacio. En la practica, la forma triangular es menos efectiva para este problema ya que permite un campo de visión muy limitado. Por otro lado el min-max velocidad de giro es bastante complejo para calcular en tiempo real y tampoco ofrece una mejora sustancial. Por lo tanto, elegimos la forma circular ya que aproxima bastante bien la dinámica del agente y su descripción matemática es más conveniente.

Entonces modelamos la observación esperada como un sensor de alcance infinito que observa todo el mundo (la creencia de la posición del objetivo). Este planteamiento soluciona el problema de la complejidad a la hora de calcular el valor esperado. Así, diseñamos la heurística de observación esperada $\hat{H}(s^k_i)$ como un sensor que aplicamos en el estado terminal s^{k+N}_i. Hay que tener en cuenta que este sensor, que funciona como una heurística, sirve sólo para problemas donde los agentes tienen sensores basados en la distancia. Para poder aplicarlos a otro tipo de plataformas sensoriales, se necesita diseñar un nuevo sensor heurístico, pero la estrategia no varía.

El modelo del sensor de alcance infinito $\hat{H}(s^{k+N}_i)$ está diseñado de tal forma que incluye el tiempo por medio de una función de descuento:

$$\hat{H}(s^{k+N}_i) = \eta\beta^{\|\tau - s^{k+N}_i\|/V_i} \tag{3.33}$$

donde $\|\tau - s^{k+N}_i\|$ es la distancia euclídea desde el agente a la posición del objetivo y V_i es la velocidad del agente. Cuando el parámetro η hace que $\sum_{s^{k+N}_i} \hat{H}(s^{k+N}_i) = 1$ la heurística está normalizada. También podemos construir heurísticas no normalizadas haciendo que $\eta = 1$ para incrementar la contribución de las recompensas futuras.

Su versión complementaria, que es la no-observación esperada, se define mediante la siguiente expresión:

$$\overline{\hat{H}}(s_i^{k+N}) = 1 - \eta\beta^{\|\tau - s_i^{k+N}\|/V_i} \tag{3.34}$$

En la Figura 3.6 podemos ver la forma de la ecuación 3.34, donde el valor es la unidad lejos de la posición del agente s_i^{k+N} y tiende a a cero según se acerca a s_i^{k+N}.

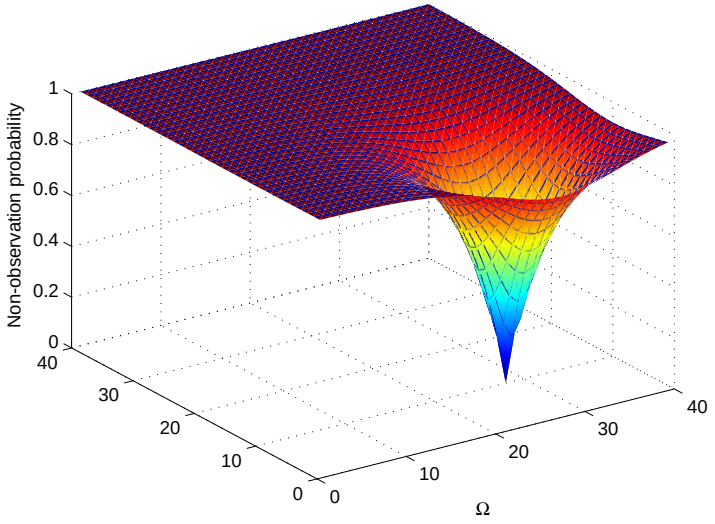


FIGURE 3.6: Modelo del sensor de alcance infinito $\overline{\hat{H}}(s^{k+N})$ cuando β =0.8, V =0.5 y $s^{k+N} = [28, 10]^T$.

La plataforma sensorial utiliza el modelo del sensor para obtener las observaciones en los estados $s_{1:q}^k, \cdots, s_{1:q}^{k+N}$ y la ecuación 3.33 para la observación esperada en el último estado $s_{1:q}^{k+N}$. El parámetro β controla la importancia de la probabilidad de encontrar el objetivo en las regiones lejanas desde el estado $s_{1:q}^{k+N}$ con creencia $\tilde{b}_\tau^{k+N}$. Implícitamente utiliza la aproximación de forma circular mostrada en la Figura 3.5.

Ya que estamos modelando la heurística como un sensor, la contribución a la función de utilidad multiagente es:

$$\int_\tau \prod_{i=1}^{q} \overline{\hat{H}}(s_i^{k+N}) b_\tau^{k+N} d\tau$$

Finalmente, podemos diseñar una estrategia DTH para cada una de las funciones de utilidad a corto plazo propuestas. En cada caso la heurística se integra de

manera diferente. Por ejemplo, combinada con la probabilidad conjunta de no-detección (ecuación 3.19) tenemos:

$$J_{NDH}(s^k_{1:q}, v^k_{1:q}, b^k_\tau) = \int_{\tau^{k+N}} \prod_{i=1}^{q} \overline{\hat{H}}(s_i^{k+N}) \tilde{b}_\tau^{k+N|k+N} d\tau^{k+N} \tag{3.35}$$

Análogamente, usando LET (ecuación 3.28) como la función a corto plazo:

$$\begin{aligned} LETH(s^k_{1:q}, v^k_{1:q}, b^k_\tau) = \sum_{j=1}^{N} \int_{\tau^{k+j}} \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | s_i^{k+j}, \tau^{k+j}) \tilde{b}_\tau^{k+j|k+j-1} d\tau^{k+j} + \\ + \int_{\tau^{k+N}} \prod_{i=1}^{q} \overline{\hat{H}}(s_i^{k+N}) \tilde{b}_\tau^{k+N|k+N} d\tau^{k+N} \end{aligned} \tag{3.36}$$

Finalmente utilizando DTR (ecuación 3.32) como función a corto plazo, la estrategia se convierte en:

$$\begin{aligned} DTRH(s^k_{1:q}, v^k_{1:q}, b^k_\tau) = \sum_{j=1}^{N} \lambda^{j-1} \int_{\tau^{k+j}} \left[1 - \prod_{i=1}^{q} P(\overline{D}_i^{k+j} | \tau^{k+j}, s_i^{k+j}) \right] b_\tau^{k+j|k+j-1} d\tau^{k+j} + \\ + \lambda^N \int_{\tau^{k+N}} \left[1 - \prod_{i=1}^{q} \overline{\hat{H}}(s_i^{k+N}) \right] \tilde{b}_\tau^{k+N|k+N} d\tau^{k+N} \end{aligned} \tag{3.37}$$

Simplemente incluyendo la heurística propuesta, reduciremos la miopía de cualquier estrategia local sin añadir tiempo de computación apreciable.

3.4.1.4 Otras Estrategias Útiles

En esta sección se describen otras dos estrategias útiles para el MTS cuando la posición del objetivo está descrita por una distribución Gaussiana. La primera estrategia, Máxima Pendiente (Maximum Slope, MS), utiliza las propiedades geométricas de la Gaussiana asumiendo que los sensores no obtendrán medidas, que es el peor caso posible. La segunda estrategia, Mínima Entropía (Minimum Entropy, ME), se basa en asumir que nuestra estimación del objetivo es correcta y que los agentes tendrán medidas con un ruido con covarianza dependiente de la distancia.

Definimos la creencia de la posición del objetivo como la función de densidad de probabilidad Gaussiana $b^k_\tau \sim N(\hat{\tau}^k, \Sigma^k)$. b^k_τ representa la distribución de la posición actualizada del objetivo. Nos referimos a $\tilde{b}^k_\tau \sim N(\tilde{\tau}^k, \tilde{\Sigma}^k)$ como la posición del objetivo predicha por el controlador cuando ejecutamos las acciones $v^k_{1:q}$.

Máxima Pendiente: la función de utilidad J_{MS} para esta estrategia (ecuación 3.38) está definida como la suma de los valores de la distribución Gaussiana a lo largo de la trayectoria de los agentes. Esta definición nos permite conducir a los agentes a través de la máxima pendiente.

$$J_{MS}(s_i^k, v_i^k, b_\tau^k) = \sum_{j=1}^{N} e^{-\frac{1}{2}(s_i^{k+j} - \tilde{\tau}^{k+j})^T (\tilde{\Sigma}^{k+j})^{-1} (s_i^{k+j} - \tilde{\tau}^{k+j})} \tag{3.38}$$

Para calcular $\tilde{\tau}^{k+j}$ y $\tilde{\Sigma}^{k+j}$ utilizamos el filtro extendido de Kalman (Extended Kalman Filter, EKF).

Esta estrategia, que no necesita comunicación durante la fase de optimización, tiene una desventaja: todos los agentes persiguen el mismo pico de probabilidad. Esto hace que la búsqueda sea más subóptima cuando los agentes están en posiciones cercanas. En el caso de un único agente este comportamiento subóptimo no existe.

Entropía Mínima: en vez de conducir a los agentes hacia la media de la Gaussiana, esta estrategia minimiza la incertidumbre que el grupo de agentes tiene sobre la posición del objetivo. Esta incertidumbre se calcula como la entropía del sistema en el último estado estimado de los agentes:

$$J_{ME}(s_{1:q}^k, v_{1:q}^k, b_\tau^k) = \frac{1}{2} \ln(2\pi e)^2 |\tilde{\Sigma}^{k+N}| \tag{3.39}$$

El cálculo de $\tilde{\Sigma}^{k+N}$ se realiza utilizando una versión modificada del EKF.

3.4.1.5 Resumen de Estrategias

La Tabla 3.1 resume todas las estrategias presentadas en esta sección, asignando a cada una su ecuación y comentando sus características más representativas.

3.5 Resumen

En este capítulo hemos definido el problema de MTS y mostrando sus propiedades. También se han presentado las condiciones necesarias para optimizar el tiempo de detección esperado maximizando la probabilidad de detección. La existencia de esta condición muestra por qué el MTS es un problema complejo para el que no se han dado

TABLE 3.1: Resumen de las Estrategias.

Estrategia	Descripción	Ecuación	opt
LET	Tiempo esperado local promedio.	3.28	min
DTR	Probabilidad de detección descontada por el tiempo.	3.32	max
DTH	Heurística que aproxima la observación esperada. Se puede combinar con el resto.	3.35	min/max
LETH	Variante no-miope de LET al combinarlo con DTH.	3.36	min
DTRH	Variante no-miope de DTR al combinarlo con DTH.	3.37	max
MS	Seleccionar la máxima pendiente.	3.38	max
ME	Minimizar la entropía en el instante $k+N$.	3.39	min
D	Eventos de detección.	3.14	max
ND	Eventos de no-detección.	3.19	min
NDH	Variante heurística de ND al combinarlo con DTH.	3.35	min

soluciones satisfactorias hasta donde hemos podido comprobar. Además, se ha presentado cómo evaluar la trayectoria de los agentes de forma probabilística para buscar objetivos dinámicos.

Se han propuesto tres estrategias que abordan el problema del MTS: minimizar el tiempo local esperado (LET), maximizar la probabilidad de detección descontada por el tiempo (DTR) y optimizar una heurística de tiempo descontado (DTH). Además, combinando DTH con las demás funciones de utilidad probabilísticas, se han diseñado estrategias heurísticamente informadas como: LETH, DTRH y NDH.

Chapter 4

PLANTEAMIENTO DISCRETO

"I do engineering, not religion" Daniel J. Bernstein

4.1 Contenido

En este capítulo analizamos el MTS desde un planteamiento discreto, donde la región a explorar se divide en celdas, haciendo que la distribución que describe la posición del objetivo sea una función de masa de probabilidad (Probability Mass Function, PMF), que indica la probabilidad de que el objetivo se encuentre en cada celda. El objetivo de este capítulo es ofrecer soluciones para la capa de control cuando el MTS es discreto.

Una vez que la región de búsqueda está representada por celdas, podemos trasformar el espacio de búsqueda en un grafo de rejilla $G =< V, E >$, donde los vértices son las celdas y las aristas son las conexiones entre las celdas. Tanto las posiciones de los agentes como la localización objetivo estarán representados por los correspondientes vértices del grafo, pudiendo cambiar de estado por medio de las aristas.

En este capítulo proponemos dos soluciones. Primero, en la Sección 4.2, expresamos el MTS bajo el paradigma de la programación con restricciones con dominios finitos y, utilizando un resolutor de código libre [Kuchcinski, 2003], proporcionamos las acciones óptimas que los agentes deben realizar. Este método sólo sirve cuando el objetivo es estático, el sensor es ideal, y el número de variables de decisión es pequeño. Segundo, en la Sección 4.3, proponemos un algoritmo aproximado basado en el método de entropía cruzada, que obtiene soluciones subóptimas para buscar un objetivo dinámico en el mínimo tiempo posible con cualquier sensor.

4.2 Solución Global: Programación con Restricciones

- *Modelado del Problema:* Discreto
- *Representación de la Información del Objetivo:* PMF general
- *Optimización:* Global
- *Horizonte:* N
- *Sensor:* Ideal
- *Objetivo:* Estático
- *Capa de Fusión Sensorial:* RBE
- *Controlador Multiagente:* Optimización centralizada de programación con restricciones

Queremos planear N pasos adelante para obtener el mejor conjunto de acciones secuenciales v_i^{k*} para cada agente i: $v_i^{k*} = \{u_i^k, \cdots, u_i^{k+N-1}\}$. Por lo tanto, cada agente realiza N acciones comenzando en el estado s_i^k, siguiendo los estados $\{s_i^k, ..., s_i^{k+N}\}$ y realizando las observaciones $\{z_i^{k+1}, \cdots, z_i^{k+N}\}$. La solución multiagente para el MTS, formulada como una maximización, debe resolver la siguiente función de optimización:

$$\arg\max_{v_{1:q}^k} J(s_{1:q}^k, v_{1:q}^k, b_\tau^k) \tag{4.1}$$

Utilizando la estrategia DTR para un objetivo estático, la función de utilidad es (Sección 3.4.1.2):

$$J_{DTR}(s_{1:q}^k, v_{1:q}^k, b_\tau^k) = \sum_{j=1}^{N} \lambda^j \sum_{\tau \in V} \left[1 - P(D_{1:q}^{k+j} | \tau, s^{k+j})\right] \prod_{i=1}^{j-1} P(\overline{D}_{1:q}^{k+i} | \tau, s^{k+i}) b_\tau^k \tag{4.2}$$

Podemos simplificar la ecuación si asumimos que el sensor es ideal, ya que el valor acumulado al visitar una celda que ya ha sido observada es cero. Así la recompensa instantánea que obtiene cada agente r_i^k es:

$$r_i^{k+j} = \begin{cases} \lambda^{j-1} b_\tau^k(s_i^{k+j}) & \text{Si } \left(s_i^{k+j} \neq s_m^l, \forall l < k+j, \forall m \in (1,q)\right) \wedge \left(\nexists n | s_n^{k+j} = s_i^{k+j} \wedge n < i\right) \\ 0 & \text{en otro caso} \end{cases} \tag{4.3}$$

La expresión establece que el agente i obtiene 0 si s_i^{k+j} ha sido visitado o si ya ha sido observado por otro agente en el mismo instante y $\lambda^j b_\tau^k(s_i^{k+j})$ si visita la posición s_i^{k+j} por primera visita.

La función de utilidad simplificada utilizando la ecuación 4.3 es:

$$J_{DTR}(s^k_{1:q}, v^k_{1:q}, b^k_\tau) = \sum_{j=1}^{N} \sum_{i=1}^{q} r^k_i \tag{4.4}$$

La solución se puede definir en el espacio de los estados del agente:

$$S^k = \begin{bmatrix} s^{k+1}_1 & \dots & s^{k+N}_1 \\ \vdots & \dots & \vdots \\ s^{k+1}_q & \dots & s^{k+N}_q \end{bmatrix} \tag{4.5}$$

y en el de las recompensas que obtiene cada agente en cada instante:

$$R^k = \begin{bmatrix} r^{k+1}_1 & \dots & r^{k+N}_1 \\ \vdots & \dots & \vdots \\ r^{k+1}_q & \dots & r^{k+N}_q \end{bmatrix} \tag{4.6}$$

La solución óptima para el MTS planteado en esta sección consiste en el conjunto de estados secuenciales S^{k*} que maximiza la siguiente ecuación:

$$S^{k*} = \arg \max_{s^{k+1:k+N}_{1:q}} J_{DTR}(s^{k:k+N}_{1:q}, b^k_\tau) \tag{4.7}$$

Para modelar el problema de MTS en el paradigma de programación con restricciones con dominios finitos, determinamos las variables de decisión, la función a optimizar y diseñamos las restricciones.

Las variables de decisión son los estados de los agentes S^k (ecuación 4.5) y las recompensas R^k (ecuación 4.6). Aunque estas variables están relacionadas, las tratamos de forma separada para mejorar la búsqueda de la solución por el resolutor, y utilizamos restricciones para conectar sus valores. Por lo tanto, si q y N son el número de agentes y el horizonte de decisión respectivamente, el número de variables de decisión es $2qN$.

La función de optimización para los estados es:

$$\max J(s^{k:k+N}_{1:q}) = \max \sum_{i=1}^{q} \sum_{j=1}^{N} r^{k+j}_i \tag{4.8}$$

Las restricciones que se deben cumplir son las siguientes:

1. El estado inicial es fijo para todos los agentes (cada agente puede comenzar en cualquier vértice V del grafo):

$$s_i^k = c | c \in V$$

2. Los estados deben ser válidos, es decir deben estar conectados por las aristas del grafo. Esta restricción es codificada utilizando un diagrama multivaluado de decisión (Multivalued Decision Diagram, MDD) [Cheng and Yap, 2010], que permite definir las conexiones como una tabla de pares de valores.

3. Las recompensas tienen que estar ponderadas por la función de tiempo de descuento. Para esto, construimos una tabla que conecta cada vértice $c \in V$ con su valor real de acuerdo con el instante en el que se visita j:

$$g(c, j) = \lambda^j b_\tau^k(c) \quad \forall c \in V \quad \forall j \in (1, N)$$

Luego asignamos a cada estado del agente s_i^{k+j} su correspondiente recompensa r_i^{k+j} utilizando la siguiente restricción:

$$element(s_i^{k+j}, j, r_i^{k+j} := g(s_i^{k+j}, j))$$

donde dependiendo del valor que toma s_i^{k+j} y el parámetro temporal j, se selecciona el correspondiente término de tabla $g(s_i^{k+j}, j)$.

Además, incluimos las siguientes restricciones para mejorar el rendimiento del algoritmo:

1. *Límite superior e inferior.* Definimos el límite inferior como la suma de las recompensas asociadas con las variables de los estados que ya han sido fijadas (etiquetadas, $grounded(s_i^{k+j})$):

$$R_{min} = \sum_{i=1}^{q} \sum_{j \in grounded(s_i^{k+j})} r_i^{k+j} \tag{4.9}$$

Definimos el límite superior como la recompensa máxima que la solución puede obtener. Esto se calcula mediante una heurística voraz que tiene en cuenta los valores máximos que pueden obtener las variables no fijadas todavía:

$$\hat{R}_{max} = \sum_{i=1}^{q} \sum_{j \notin grounded(s_i^{k+j})} \max r_i^{k+j} \tag{4.10}$$

La función de utilidad se reescribe como sigue:

$$J(s_{1:q}^{k:k+N}) = R_{min} + \hat{R}_{max} \tag{4.11}$$

2. Para mejorar aún más la eficiencia del algoritmo podemos incluir la siguiente aproximación: no puede haber estados de los agentes repetidos excepto en el instante inicial:

$$\text{alldifferent}(s_{1:q}^{k+1:k+N})$$

Esta restricción permite al algoritmo devolver la solución óptima cuando la solución del problema planteado no incluye posiciones repetidas.

4.3 Solución Aproximada: Optimización de Entropía Cruzada

- *Modelado del Problema:* Discreto
- *Representación de la Información del Objetivo:* PMF general
- *Optimización:* Aproximación global
- *Horizonte:* N+Heurística
- *Sensor:* Cualquiera
- *Objetivo:* Dinámico
- *Capa de Fusión Sensorial:* RBE
- *Controlador Multiagente:* Optimización centralizada de entropía cruzada

El segundo algoritmo propuesto para resolver el problema de decisión en el MTS discreto está basado en una metodología de muestreo llamada optimización de entropía cruzada (Cross Entropy Optimization, CEO) [Rubinstein and Kroese, 2004; Boer et al., 2002]. El objetivo de este algoritmo es proporcionar una solución subóptima global que implemente la capa del controlador de los agentes y que mejore planteamientos previos en términos de tratabilidad computacional. Esta solución funciona con cualquier tipo de sensor, dinámica del objetivo y además maneja el acoplamiento de la información. Esto es posible debido a la naturaleza de muestreo del método, donde cada muestra es una posible solución que es evaluada por la función de utilidad. Para instancias grandes del MTS, el número de muestras que se necesitan para obtener la solución óptima es elevado y por lo tanto, tenemos que encontrar un término medio entre optimalidad y tiempo de cálculo.

El núcleo del algoritmo es un proceso iterativo de aprendizaje, donde en vez de calcular las mejores acciones directamente, aprendemos la mejor distribución de las acciones. Así, la solución no es la secuencia de acciones, sino la distribución discreta de las acciones

que mejor se ajustan a una función de optimización. El método, en la práctica, minimiza la discrepancia entre dos distribuciones utilizando la divergencia de Kullback Leibler. Estas dos distribuciones son la distribución óptima esperada de las acciones y la mejor distribución encontrada por el algoritmo en cada iteración.

Dados los estados iniciales de los agentes $s^k_{1:q}$ y la distribución inicial de la posición del objetivo $b^{k|k}_\tau$ el algoritmo devuelve las mejores N acciones secuenciales $v^{k*}_{1:q}$. Queremos aprender la distribución de probabilidad que nos permite muestrear el plan óptimo ($v^{k*}_{1:q} = u^{k:k+N-1}_{1:q}$). CEO aprende esta probabilidad iterando dos fases. En la primera fase, muestrea soluciones de la distribución que tiene en esa iteración y selecciona las muestras que son mejores. "Mejor" se define de acuerdo con el criterio de optimización, es decir, de acuerdo con la función de utilidad (DTR, LET, LETH, DTRH o D). En la segunda fase, obtiene los parámetros de la distribución de las muestras seleccionadas minimizando la entropía cruzada.

Para resolver el MTS utilizando CEO, tenemos que trasformar la solución en forma de acciones $v^k_{1:q}$ en la distribución de probabilidad $\hat{p}$ (la probabilidad de que el agente i realice u en el instante k). Así, si se cuenta con $|A|$ acciones, cada muestra X es una matriz binaria de tamaño $|A| \times N \times q$. En cada iteración j, CEO calcula la estimación de la función de utilidad empleada ($\hat{J}^j$) y, cuando converge, resulta la utilidad óptima.

El algoritmo 14 muestra el método CEO formulado para maximizar una función de utilidad. Las etapas son las siguientes:

- Línea 1: inicializar las probabilidades de realizar cada acción u en cada instante k de tiempo para cada agente i $\hat{p}^0_{uki}$.
- Línea 3: generar un conjunto de soluciones $\aleph^j = \{X^j_1, \ldots, X^j_M\}$ muestreando sus valores de acuerdo a la distribución $\hat{p}^j_{uki}$.
- Línea 6: actualizar la recompensa esperada $\hat{J}^j$ utilizando las muestras $\aleph^j$.
- Línea 7: considerar que la mejor solución es un evento raro con baja probabilidad (ρ) y seleccionar las mejores soluciones $\aleph^j_{elite}$.
- Línea 8: aprender $\hat{p}^{j+1}_{uki}$ como la función que mejor ajusta las muestras seleccionadas $\aleph^j_{elite}$.
- Línea 2: si se satisface la condición de parada, entonces $p^* = \hat{p}^{j+1}_{uki}$, sino $j = j + 1$ y el algoritmo vuelve a la línea 3.
- Línea 10: extraer la secuencia de acciones de la probabilidad optimizada: $v^{k*}_{1:q} = \arg\max p^*$.

Algorithm 14 Algoritmo CEO

$\hat{p}^0 \leftarrow \mathbf{1}_{|A|\times N\times q} \cdot \frac{1}{|A|}$
while $|\hat{J}^j - \hat{J}^{j-N_I}| < \epsilon$ **do**
  $\{X_1^j, \ldots, X_M^j\} \sim \hat{p}^{j-1}$
  $\{w_1^j, \ldots, w_M^j\} \leftarrow \{J(X_1^j), \ldots, J(X_M^j)\}$
  $W_{ord} \leftarrow sort(\{w_1^j, \ldots, w_M^j\}$
  $\hat{J}^j \leftarrow W_{ord}(\lceil \rho \cdot M \rceil)$
  $\aleph_{elite} \leftarrow \{X_i^j | J(X_i^j) \geq \hat{J}^j\}$
  $\hat{p}^j \leftarrow \alpha \left(\frac{\sum_{X_i \in \aleph_{elite}} x_i^j}{|\aleph_{elite}|} \right) + (1-\alpha)\hat{p}^{j-1}$
end while
$v^* \leftarrow \arg\max \hat{p}^j$

4.4 Resultados

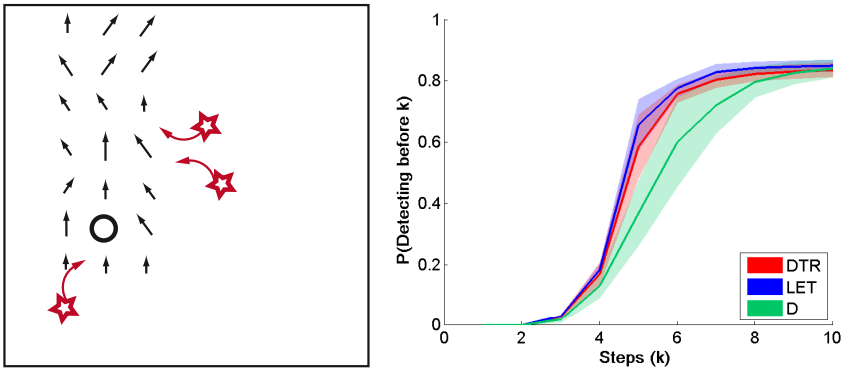
En esta sección mostramos que las estrategias propuestas reducen el tiempo de detección sobre un único escenario. Con este propósito, vamos a usar el algoritmo de optimización propuesto en la Sección 4.3 para calcular la trayectoria del equipo de agentes utilizando diferentes estrategias. Un análisis más detallado, sobre múltiples escenarios, de todos los algoritmos y funciones de utilidad, se encuentra recogido en la tesis en inglés.

Las estrategias se evalúan mediante la Ganancia de Información (Information Gain, IG), que no es más que la probabilidad acumulada de detectar al objetivo antes de un instante k: $P(T \leq k)$. Esta distribución es una medida que nos permite inducir el tiempo esperado teórico. Altos valores de IG en instantes previos implican que los agentes encuentran el objetivo antes.

El escenario elegido, es decir, la situación inicial de los agentes y la probabilidad inicial de encontrar el objetivo, está descrito de manera esquemática en la Figura 4.1(a). Contamos con tres agentes ($q = 3$) con localizaciones diferentes y con un horizonte de decisión de $N = 10$. La estrella roja indica donde comienzan los agentes, y la flecha roja, su posible trayectoria. El circulo negro representa la región inicial con probabilidad no nula para encontrar el objetivo y las flechas negras indican su desplazamiento a lo largo del tiempo. Para diseñar la dinámica del objetivo utilizamos un mapa de viento [MathWorks, 2010].

Para calcular el IG que se muestra en la Figura 4.1(b) optimizamos 50 veces las acciones de los agentes con cada estrategia y mostramos la media y la varianza. Se comparan las siguientes estrategias: DTR (Sección 3.4.1.2), LET (Sección 3.4.1.2) y D (Sección 3.3.1).

Como se puede observar en la Figura 4.1(b), las estrategias propuestas para abordar el MTS (LET y DTR) obtienen valores más altos de IG en los instantes iniciales. Por lo tanto, reducen más el tiempo de búsqueda que la estrategia de probabilidad de detección conjunta (D), que es la que se utiliza en trabajos previos como [Eagle, 1984; Bourgault et al., 2004; Gan and Sukkarieh, 2010]. Por ejemplo, utilizando LET y DTR en el



(a) Escenario (b) IG

FIGURE 4.1: Comparación de las estrategias propuestas LET y DTR con la estrategia de máxima detección (D) a la hora de reducir el tiempo de búsqueda.

instante $k = 5$ se obtiene un 20% más de probabilidades de detectar el objetivo que usando D.

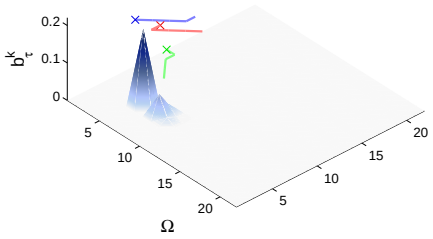
La Figura 4.2 muestra las trayectorias de los agentes utilizando las diferentes estrategias para los instantes $k = 5, 10$. Se muestra cómo la masa de probabilidad se va moviendo siguiendo la dinámica del objetivo y cómo los agentes, según van observando, la van modificando. Vemos que, tanto DTR (Figura 4.2(b)) como LET (Figura 4.2(d)), distribuyen mejor a los agentes que D (Figura 4.2(f)). Esto ocurre porque las estrategias propuestas (LET y DTR) tienen en cuenta el instante en el que se observa cada región de búsqueda. De hecho, LET obtiene el mejor resultado (Figura 4.2(d)), ya que distribuye mejor a los agentes para acumular la probabilidad cuanto antes.

4.5 Resumen

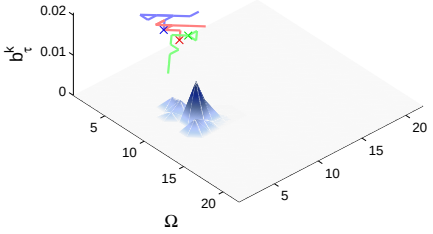
Este capítulo aborda el MTS como un problema de optimización discreta, donde las acciones de los agentes pueden tomar un número finito de valores (p.e. puntos cardinales) y la región de búsqueda está modelada como un grafo de rejilla.

Se han propuesto dos soluciones para calcular las mejores N acciones de los agentes para buscar un objetivo en el mínimo tiempo posible. La primera, CP-MTS, basada en la programación con restricciones se puede aplicar a problemas con pocas variables de decisión y objetivos estáticos. La segunda solución, basada en un método estocástico (CEO-MTS), resuelve el MTS para objetos dinámicos.

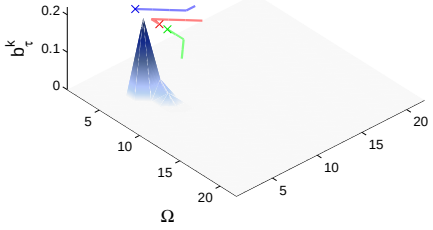
La solución CP-MTS utiliza la estrategia DTR para calcular la probabilidad acumulada de detección descontada por el tiempo para un horizonte finito. Este método es global y óptimo dentro del horizonte de decisión.



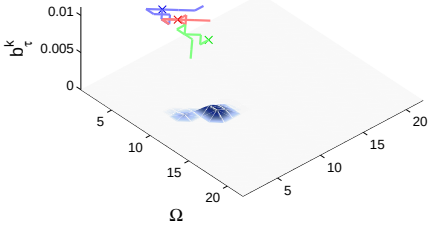
(a) DTR k=5



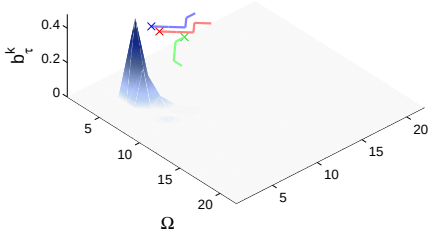
(b) DTR k=10



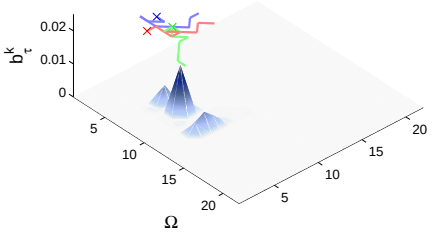
(c) LET $k = 5$



(d) LET $k = 10$



(e) D $k = 5$



(f) D $k = 10$

FIGURE 4.2: Simulaciones de las trayectorias con $q = 3$ y $N = 10$.

El segundo algoritmo, CEO-MTS, acepta cualquier tipo de sensor y optimiza cualquier estrategia de MTS propuesta. Este método es computacionalmente tratable y obtiene soluciones subóptimas para la búsqueda de un objetivo dinámico. Para problemas grandes de MTS la complejidad computacional depende del compromiso entre el número de muestras que se generan y la optimalidad que se requiere.

Se ha realizado un análisis de las estrategias MTS y de las soluciones presentadas. Los resultados muestran que CEO-MTS es una aproximación válida para implementar la capa de control aunque obtiene soluciones subóptimas. Los experimentos corroboran que las estrategias propuestas minimizan el tiempo para detectar el objetivo, mejorando soluciones previas. También se ha mostrado en esta tesis cómo las versiones heurísticas

de las estrategias reducen considerablemente la localidad de las decisiones. Además, se ha mostrado que la combinación del RBE como capa de fusión sensorial con el CEO-MTS como capa de control es una buena solución para diseñar el sistema autónomo que resuelve la tarea de MTS.

Chapter 5

PLANTEAMIENTO CONTINUO

"Information: the negative reciprocal value of probability" Claude Shannon

5.1 Contenido

Este capítulo analiza el MTS modelado de manera continua, donde las acciones toman valores reales y la distribución de la posición del agente es una función de densidad probabilística (Probability Density Function, PDF). Aproximamos la optimización global trasformando el problema en una optimización lineal a tramos, donde la misma acción de control se utiliza durante un intervalo constante de tiempo. Por tanto, tenemos N observaciones y N acciones de control para cada agente.

Proporcionamos dos soluciones: 1) una combinación del método del gradiente con la estrategia DTH para la capa del controlador que reduce la localidad y mejora soluciones previas; y 2) un algoritmo que mejora la capa de fusión sensorial teniendo en cuenta los retrasos en la comunicación de las medidas cuando la posición del objetivo sigue una distribución Gaussiana.

La primera solución, basada en los trabajos de [Mathews et al., 2007; Gan and Sukkarieh, 2010] es rápida, válida para su ejecución en línea, subóptima y no-miope. En ella, nos enfrentamos al MTS como un problema descentralizado donde la búsqueda es realizada por un equipo de agentes sensoriales independientes y autónomos, aunque la centralización del algoritmo es directa. El algoritmo de optimización se basa en el método del gradiente. Debido a la complejidad para calcular el Jacobiano de la función de utilidad cuando el objetivo se está moviendo, asumimos que el objeto es estático durante el proceso de decisión [Mathews, 2008].

La segunda solución aborda un problema real que ocurre en las redes de sensores: las medidas de los sensores llegan tarde y desordenadas a la capa de fusión sensorial, repercutiendo en el proceso de la estimación Bayesiana y causando problemas de eficiencia en el controlador. El problema, conocido como medidas fuera de secuencia (Out Of Sequence, OOS), puede resolverse modificando el filtro Bayesiano para que acepte las medidas retardadas. En la solución propuesta en este capítulo, la posición del objetivo está descrita por una función de distribución Gaussiana. Para aceptar las medidas desordenadas modificamos el filtro de Kalman extendido (Extended Kalman Filter, EKF) y así conseguimos que la información sobre la localización del objetivo sea más precisa.

5.2 Solución Aproximada: Optimización basada en el Gradiente

- *Modelado del Problema:* Continuo
- *Representación de la Información del Objetivo:* PDF general
- *Optimización:* Local
- *Horizonte:* N + Heurística
- *Sensor:* Diferenciable
- *Objetivo:* Estático
- *Capa de Fusión Sensorial:* RBE descentralizado/centralizado
- *Controlador Multiagente:* Optimización de gradiente descentralizada/centralizada

En esta sección proponemos una solución a la búsqueda coordinada de múltiples agentes que incorpora las ventajas de la optimización continua descentralizada [Gan and Sukkarieh, 2010; Mathews et al., 2007; Bourgault et al., 2004] y las estrategias no miopes presentadas en esta tesis. Así reducimos los problemas de localidad y reducimos el tiempo para detectar el objeto. El objetivo del proceso de optimización es obtener el plan de acciones que optimiza la función de utilidad.

Formulando el MTS como un problema de minimización la solución para un agente es:

$$v_i^{k*} = \arg\min_{v_i^k} J(s_i^k, v_i^k, b_\tau^k) \tag{5.1}$$

El problema de optimización descrito puede resolverse utilizando el método de descenso de gradiente, que sólo requiere las derivadas parciales de la función de utilidad con respecto a cada acción ($\frac{\partial J}{\partial v_i^k}$). Iterativamente actualizamos el vector de acciones utilizando el método de descenso de gradiente [Mathews et al., 2007; Gan and Sukkarieh, 2010]. Sea $v_i^{k,t}$ el vector de acciones para el agente i en el instante k y t el paso de iteración, la mejora de la solución viene dada por la siguiente expresión:

$$v_i^{k,t+1} = v_i^{k,t} - \delta \frac{\partial J}{\partial v_i^{k,t}} \tag{5.2}$$

donde δ es el tamaño de paso que ajusta la razón de convergencia [Mathews et al., 2007]. Ya que las funciones de utilidad para el MTS son no-convexas por naturaleza, este método sólo puede garantizar soluciones locales. Las derivadas parciales pueden calcularse directamente utilizando la formulación explícita recogida en [Gan and Sukkarieh, 2010].

5.2.1 Coordinación Multiagente

La coordinación de los agentes se realiza por negociación utilizando la función NDH (ecuación 3.35). Hemos elegido NDH (no-detección con observación esperada) por cuatro razones fundamentales: permite coordinación anónima ya que los agentes no tienen que conocer el modelo del sensor ni la dinámica del resto de los agentes; es fácil de implementar; existe una prueba de convergencia en el caso descentralizado [Mathews et al., 2007]; y los resultados experimentales han mostrado buenos resultados cuando el horizonte de decisión es pequeño.

Para extender la estrategia NDH para la coordinación multiagente consideramos q agentes que trabajan en equipo. Por lo tanto, el equipo tiene que optimizar de forma conjunta la función de no-detección con observación esperada.

Asumiendo que existe una capa de fusión sensorial descentralizada (Decentralized Data Fusion, DDF), que sincroniza la creencia sobre la posición del objetivo en cada agente, de tal manera que ${}^i b_\tau^k = {}^j b_\tau^k = b_\tau^k \; \forall \; \{i,j\} \in (1,q)$, y que las observaciones de los sensores de cada agente individual son independientes, la función de utilidad para un agente desde la perspectiva del grupo es:

$$J_{NDH}(s_{1:q}^k, v_{1:q}^k, b_\tau^k) = \int_\tau \prod_{i=1}^{q} \prod_{j=1}^{N} P(\overline{D}_i^{k+j} | \tau, s_i^{k+j}) \prod_{i=1}^{q} \overline{\hat{H}}(s_i^{k+N}) \; b_\tau^k \; d\tau \tag{5.3}$$

Para expresar la ecuación 5.3 de forma descentralizada, cada agente tiene que comunicar la probabilidad de no detección a la red de agentes. Estableciendo esta comunicación, la función de utilidad para cada agente i se reescribe como:

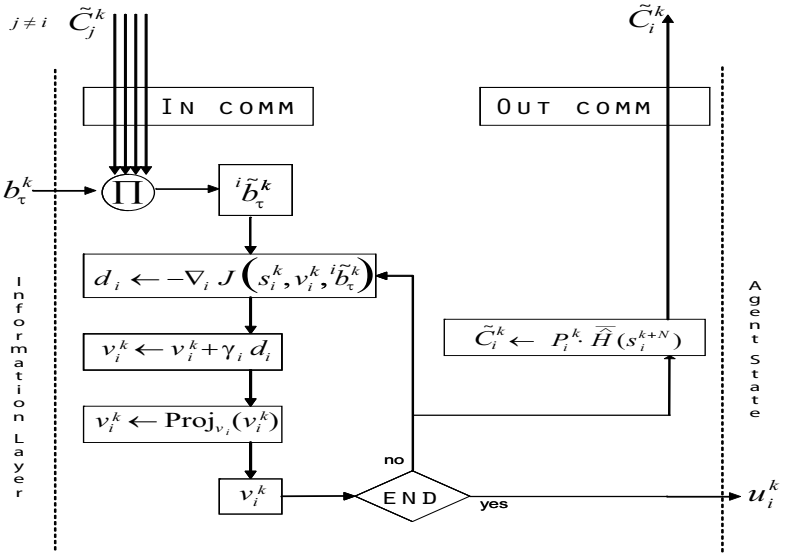


FIGURE 5.1: Algoritmo de decisión con observación esperada.

$$J_{NDH}(s_i^k, v_i^k, b_\tau^k) = \int_\tau P(\overline{D}_i^{k+1:k+N}|\tau, s_i^{k+1:k+N})\overline{\hat{H}}(s_i^{k+N}){}^i\tilde{b}_\tau^k \, d\tau \tag{5.4}$$

$$= \int_\tau P_i^k \overline{\hat{H}}(s_i^{k+N}){}^i\tilde{b}_\tau^k \, d\tau \tag{5.5}$$

donde

$${}^i\tilde{b}_\tau^k = \prod_{j=1,j\neq i}^{q} P(\overline{D}_j^{k+1:k+N}|\tau, s_j^{k+1:k+N}) \prod_{j=1,,j\neq i}^{q} \overline{\hat{H}}(s_j^{k+N}) \, b_\tau^k \tag{5.6}$$

$$= \prod_{j=1,j\neq i}^{q} P_j^k \prod_{j=1,j\neq i}^{q} \overline{\hat{H}}(s_j^{k+N}) \, b_\tau^k \tag{5.7}$$

es la creencia del agente i modificada por la información $P_j^k \triangleq \prod_{l=1}^{q} P(\overline{D}_j^{k+l}|\tau, s_j^{k+l})$ y $\overline{\hat{H}}(s_j^{k+N})$ que comunican los demás agentes $\{j \in (1,q), j \neq i\}$.

Utilizando la DDF como capa de fusion sensorial el equipo sincroniza la creencia de la posición del objetivo b_τ^k. Así que la información que cada agente i tiene que comunicar, en cada iteración de la optimización, es la siguiente:

$$\tilde{C}_i^k = P_i^k \overline{\hat{H}}(s_i^{k+N}) \tag{5.8}$$

El algoritmo que implementa el controlador con observación esperada se muestra en la Figura 5.1.

Esta solución propuesta para resolver el MTS de forma descentralizada integra la función de probabilidad conjunta de no-detección con la observación esperada. Así, tiene en

cuenta la recompensa futura, reduce los problemas de localidad y reduce el tiempo para encontrar el objetivo.

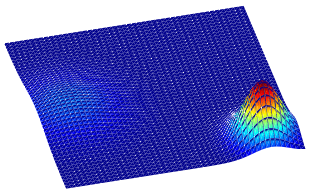
5.2.2 Resultados

En esta sección mostramos el sistema funcionando de forma descentralizada en dos escenarios sintéticos diferentes. Comparamos la función no-detección (ND) utilizada en trabajos previos como [Mathews et al., 2007; Gan and Sukkarieh, 2010] con su versión heurística propuesta en esta tesis (NDH). Es decir, hemos analizado la mejora del rendimiento de búsqueda al utilizar la observación esperada (estrategia DTH) dentro de una configuración multiagente descentralizada. En la versión en inglés de la tesis, se recogen múltiples resultados sobre diferentes escenarios, para analizar también el comportamiento de todas las funciones de utilidad sobre el método del gradiente.

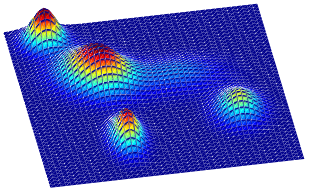
La Figura 5.2 compara el rendimiento de equipo de agentes coordinados utilizando las dos estrategias. Se han elegido dos escenarios: un escenario con dos regiones disjuntas (Figura 5.2(a)) y un escenario complejo con múltiples máximos locales (Figura 5.2(b)). Los resultados muestran que utilizando la observación esperada, es decir, la estrategia NDH, la búsqueda es más eficiente comparada con la estrategia local en ambos escenarios.

Las Figuras 5.2(c), 5.2(d), 5.2(e) y 5.2(f), obtenidas mediante un proceso de simulación, muestran el comportamiento del equipo en un instante determinado. Cada agente está representado por una línea de color, donde: la línea delgada muestra el camino ya recorrido, la línea punteada representa el resto de la trayectoria planeada y la línea gruesa corresponde a la nueva planificación. El mapa de escala de grises indica la probabilidad de encontrar el objetivo actualizada por las observaciones de los agentes (p.e. el pico de probabilidad en la parte de abajo de la Figura 5.2(b) ha sido ya aplanado por el equipo en la Figura 5.2(d) y 5.2(f)). Las Figuras 5.2(g) y 5.2(h) muestran la ganancia de información (IG) obtenida por el equipo en cada instante. La medida IG corresponde a la probabilidad que tiene el equipo de agentes de haber encontrado el objetivo en cada instante, es decir, la probabilidad acumulada. Cuanto más elevada sea esta medida en los instantes iniciales mayor será la reducción del tiempo de detección.

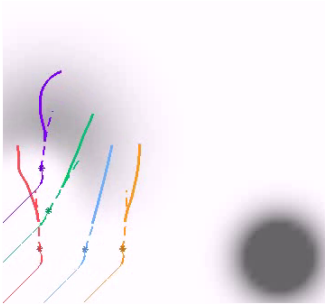
En el primer escenario, el equipo que utiliza la observación esperada (Figura 5.2(e)) evita quedarse atrapado en el óptimo local en el que se visita la región más cercana pero poco probable, y explora áreas con más probabilidad que están fuera del horizonte de decisión. Mientras tanto, la estrategia sin heurística (observación esperada) hace que los agentes se queden en el óptimo local (Figura 5.2(c)). Este hecho se pone de manifiesto en un aumento más rápido de la acumulación de la probabilidad de detección (Fig 5.2(g)). De hecho, utilizando la observación esperada, en $400s$, el objetivo se detecta con un 100% de certeza, mientras que sin utilizar la heurística mencionada la probabilidad de detectar el objetivo es menor del 50% (Figura 5.2(g)). En el segundo escenario, los agentes guiados



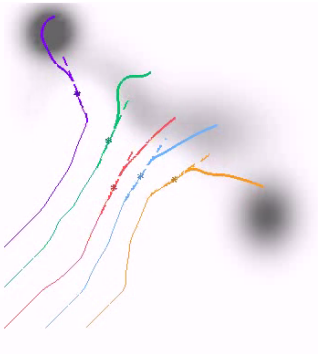
(a) Escenario con dos Gaussianas b_τ^0



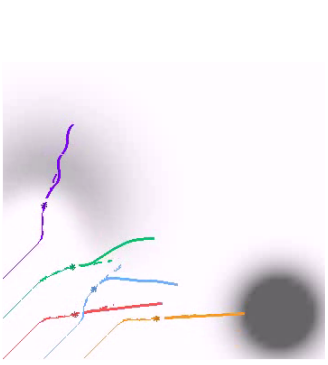
(b) Escenario complejo b_τ^0



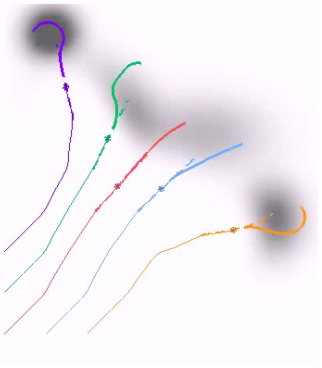
(c) No observación esperada



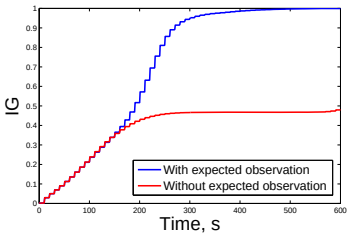
(d) No observación esperada



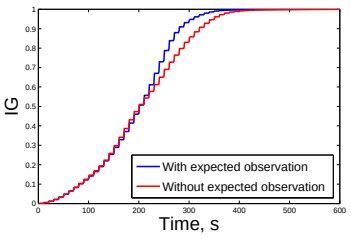
(e) Observación esperada



(f) Observación esperada



(g) Escenario con dos Gaussianas IG



(h) Scenario complejo IG

FIGURE 5.2: Coordinación multiagente en dos escenarios diferentes. El rendimiento de búsqueda se compara entre la estrategia ND (línea roja) y NDH (línea azul). Se han utilizado 5 agentes para esta simulación.

por la estrategia heurística propuesta (Figura 5.2(f)) se dispersan más hacia regiones donde obtienen alta probabilidad de detectar el objetivo en el futuro, mejorando la búsqueda a largo plazo. Por ejemplo, en el instante de simulación presentado en las

Figuras 5.2(d) y 5.2(f), los agentes ya han observado los máximos locales cercanos y actualizado el mapa de probabilidades, pero dependiendo de la estrategia utilizada las trayectorias de los agentes comienzan a ser diferentes. Mientras que en la Figura 5.2(f) el agente amarillo ya está observando la región que se encuentra más a la derecha, en la Figura 5.2(d), el amarillo todavía está lejos de ese área. Esto implica que la ganancia de información se incremente más rápidamente con la estrategia propuesta, como se muestra en la Figura 5.2(h).

5.3 Solución para el Problema de las Medidas Desordenadas para el Caso Especial Gaussiano

- *Modelado del Problema:* Continuo
- *Representación de la Información del Objetivo:* Distribución Gaussiana
- *Optimización:* Local
- *Horizonte:* N
- *Sensor:* Continuo
- *Objetivo:* Dinámico
- *Capa de Fusión Sensorial:* OOS EKF
- *Controlador Multiagente:* Optimización de descenso de gradiente

El grupo de agentes, considerado como una red sensorial, está sujeto a problemas de retrasos en la comunicación. Cuando un agente observa la región de búsqueda y envía la medida a la red, el mensaje puede llegar desordenado y con retraso a la capa de fusión sensorial (en el caso descentralizado, cada agente funciona como un centro de fusión sensorial). Esto afecta a la calidad de la estimación de la posición del objeto. En la literatura este problema se conoce como el problema de medidas desordenadas (Out Of Sequence, OOS) [Bar-Shalom and Chen, 2005]. Como se menciona en [Bourgault et al., 2004], mejorando la capa de fusión sensorial, también se mejora el comportamiento de los agentes y su cooperación. Por lo tanto, abordando el problema del OOS nuestra estimación del objetivo será más precisa y las tareas de búsqueda y seguimiento del objetivo se realizarán mejor.

La solución propuesta considera que la posición del objetivo está modelada como una función de densidad de probabilidad Gaussiana. Además, se conoce la posición esperada del objetivo, su varianza, y su modelo probabilístico de movimiento. Se han tenido en cuenta las fuentes de ruido siguientes: los errores de las medidas de los sensores y la

incertidumbre del movimiento del objetivo. El sensor embarcado en el agente es capaz de medir la distancia del agente al objetivo. Se estima la posición del objetivo usando un filtro de Kalman extendido (EKF) [Mutambara, 1998]. Para abordar el problema del OOS adaptamos los métodos propuestos por [Besada-Portas et al., 2011] para una configuración multisensorial distribuida.

Los agentes envían las medidas de los sensores que son recibidas por la capa de fusión con un retraso aleatorio que está limitado por el retraso mínimo δ_{min} y el retraso máximo de la red δ_{max}. Así, cualquier medida enviada en el instante k llega a la capa de información en el intervalo $\delta_{min} \leq \delta \leq \delta_{max}$.

El filtro propuesto tiene dos pasos iterativos: la predicción (Algoritmo 15) y la actualización/asimilación (Algoritmo 16).

Los agentes comunican sus medidas z_i^k a la vez que el tiempo en que se recogieron. En cada instante, cada agente comprueba si existen nuevas medidas para asimilarlas y predice el nuevo estado del objetivo. Para actualizar la información $b_\tau^{k|k}$ con las medidas desordenadas, efectuamos la asimilación en el espacio de información siguiendo las ecuaciones del filtro de información extendido (Extended Information Filter, EIF). Para predecir la nueva posición y su incertidumbre asociada seguimos las ecuaciones en el espacio de estados del EKF.

El algoritmo de predicción (Algoritmo 15) estima la media $\hat{\tau}$ y la matriz de covarianza σ de la distribución de la posición del objetivo para el siguiente instante $k+1$, que, en notación de Kalman está definida como: $\hat{\tau}^{k+1|k}$, $\Sigma^{k+1|k}$. Utiliza el Jacobiano F de la función de movimiento del objetivo $f(\hat{\tau}^{k|k})$ y la matriz de covarianza del ruido asociado al movimiento del objetivo Q^k.

Algorithm 15 Predicción OOS EKF

Require: $\hat{\tau}^{k|k}$ ▷ Posición estimada
Require: $\Sigma^{k|k}$ ▷ Error de la posición
Require: Q^k ▷ Covarianza del ruido de la dinámica

1: $\hat{\tau}^{k+1|k} \leftarrow [p_x + \Delta k V_\tau + w_x^k, p_y + \Delta k V_\tau + w_y^k]^T$
2: $F \leftarrow \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
3: $\Sigma^{k+1|k} \leftarrow F\Sigma^{k|k}F^T + Q^k$

Para el paso de asimilación, Algoritmo 16, necesitamos definir una ventana de tiempo l que describe durante cuantos instantes necesitamos guardar los estados y las varianzas para ser capaces de asimilar las medidas desordenadas. Idealmente, este parámetro está fijado al valor del retraso máximo de la red ($l = \delta_{max}$), es decir, el tiempo máximo que un agente necesita para recibir una medida procedente del resto de los agentes. Una vez que hemos definido la ventana de tiempo, cada agente tiene que guardar la media y la varianza desde el instante $k-l$ hasta el instante actual k ($\{\hat{\tau}^{t|t-1}, \Sigma^{t|t-1} | \forall t \in (k-l, k)\}$). También tiene que guardar las medidas asimiladas proyectadas en el espacio

de información ($\{i^t, I^t | \forall t \in (k-l, k)\}$). El paso de asimilación actualiza la distribución de la posición del objetivo con las nuevas medidas devolviendo: $\hat{\tau}^{k+1|k+1}$, $\Sigma^{k+1|k+1}$.

Algorithm 16 Asimilación OOS

Require: l ▷ Ventana de memoria δ_{max}
Require: $\{\hat{\tau}^{t|t-1}, \Sigma^{t|t-1} | \forall t \in (k-l, k)\}$ ▷ Memoria de la posición predicha
Require: $\{i^t, I^t | \forall t \in (k-l, k)\}$ ▷ Memoria de las medidas proyectadas
Require: R^k ▷ Covarianza del ruido del sensor

```
if Nuevas medidas (Z)? then
    t = MinTimeStamp(Z)
    for j = t, k do
        Y^{j|j-1} ← (Σ^{j|j-1})^{-1}
        y^{j|j-1} ← Y τ̂^{j|j-1}
        Z^j ← GetMeasurementsAtK(Z, j)
                ▷ Z^j es el conjunto z_a^j donde a es el agente y j es el tiempo
        for z_a^j ∈ (Z^j) do
            H = (τ̂^{j|j-1} − s_a^j)/‖s_a^j − τ̂^{j|j-1}‖
            zc_a^j ← z_a^j − ‖s_a^j − τ̂^{j|j-1}‖ + H τ̂^{j|j-1}        ▷ Medida corregida
            i^j ← i^j + H^T R^{-1} zc_a^j
            I^j ← I^j + H^T R^{-1} H
            Guardar los nuevos valores i^j, I^j
        end for
        Y^{j|j} ← Y^{j|j-1} + I^j
        y^{j|j} ← y^{j|j-1} + i^j
        Σ^{j|j} ← (Y^{j|j})^{-1}
        τ̂^{j|j} ← Σ^{j|j} y^{j|j}
        (τ̂^{j+1|j}, Σ^{j+1|j}) ← Prediction(τ̂^{j|j}, Σ^{j|j})        ▷ Alg. 15
        Guardar los nuevos valores τ̂^{j+1|j}, Σ^{j+1|j}
    end for
end if
```

El conjunto de medidas Z que llega en el instante k al agente son estructuras que proporcionan información sobre el instante j en que la medida fue tomada y el agente a que la tomó, es decir, elementos de la forma z_a^j. En el algoritmo 16, $MinTimeStamp(Z)$ devuelve el tiempo de la medida más antigua y $GetMeasurementsAtK(Z, j)$ devuelve las medidas tomadas en el instante j del conjunto Z (Z^j). Así, cada vez que el agente recibe medidas, primero mira la medida más antigua y aproxima la estimación desde ahí. Como el algoritmo admite mediadas no lineales, necesitamos corregir la medida z_a^j mediante la siguiente ecuación:

$$zc_a^j = z_a^j - \|s_a^j - \hat{\tau}^{j|j-1}\| + H\hat{\tau}^{j|j-1}$$

donde H es el Jacobiano del sensor, es decir, la derivada parcial de la distancia euclídea con respecto a la posición del agente. Una vez que tenemos la medida corregida, calculamos la información i_a^j y la matriz de información I_a^j asociada con la observación. Esto se realiza para cada medida desordenada tomada en el instante j. Después sumamos todas las informaciones para obtener la información actualizada de la posición del objetivo

$y^{j|j}$ en el instante j y predecimos la nueva posición del objetivo utilizando el Algoritmo 15. Realizamos este proceso hasta que no quedan medidas por asimilar.

Ejecutando los pasos de predicción y de asimilación obtenemos una solución al problema del OOS para la estimación de la posición del objetivo dentro del grupo de agentes. Este proceso se ejecuta en paralelo al controlador, que utilizará esta información para planear la mejor secuencia de acciones en las tareas de búsqueda y seguimiento del objetivo.

5.3.1 Resultados

Se ha analizado el rendimiento del filtro OOS propuesto para la capa de fusión sensorial en la tarea de búsqueda de objetos móviles. Se ha estudiado sí la estimación de la localización del objetivo mejora y sí su incertidumbre se reduce.

Para ello, por un lado, se ha realizado un estudio comparativo entre tres filtros diferentes: 1) el EKF OOS propuesto (OOS); 2) un filtro que descarta las medidas de los sensores que llegan con retraso (Descartar); y 3) el EKF en una red de comunicación sin retrasos (NoDesordenadas). Por otro lado, se ha estudiado el efecto de utilizar tres estrategias diferentes en el controlador en términos de estimación de la localización del objeto: máxima pendiente (Maximum Slope, MS), mínima entropía (Minimum Entropy, ME) y probabilidad de no-detección con observación esperada (Non-Detection Heuristic, NDH).

La Figura 5.3 muestra un ejemplo de simulación de las trayectorias del equipo de agentes utilizando los diferentes filtros y estrategias. Las etiquetas en la izquierda de la figura describen la estrategia usada y las de la parte de arriba especifican el filtro utilizado. Las trayectorias de los agentes están representadas por las líneas coloreadas de rojo, verde y azul, y la trayectoria del objetivo está descrita por la línea amarilla. Las cruces coloreadas representan las posiciones de los agentes y el círculo amarillo la posición del objeto. Finalmente, el mapa de altura representa la creencia del equipo de la localización del objetivo, es decir, la función de probabilidad de densidad Gaussiana. Desde el punto de vista de la estimación de la posición del objetivo, la mejor trayectoria de los agentes es la que hace que la Gaussiana esté más concentrada, como por ejemplo OOSDTH o ME sin medidas desordenadas).

Los resultados recogidos a lo largo de la tesis (versión completa en inglés) han mostrado que el filtro propuesto aborda con éxito el problema del OOS ya que tiende a aproximar el rendimiento de un filtro EKF en una red sin retrasos (situación ideal). Además se ha comprobado que tener en cuenta las medidas desordenadas en el filtro reduce el error de estimación de la posición del objetivo y su incertidumbre.

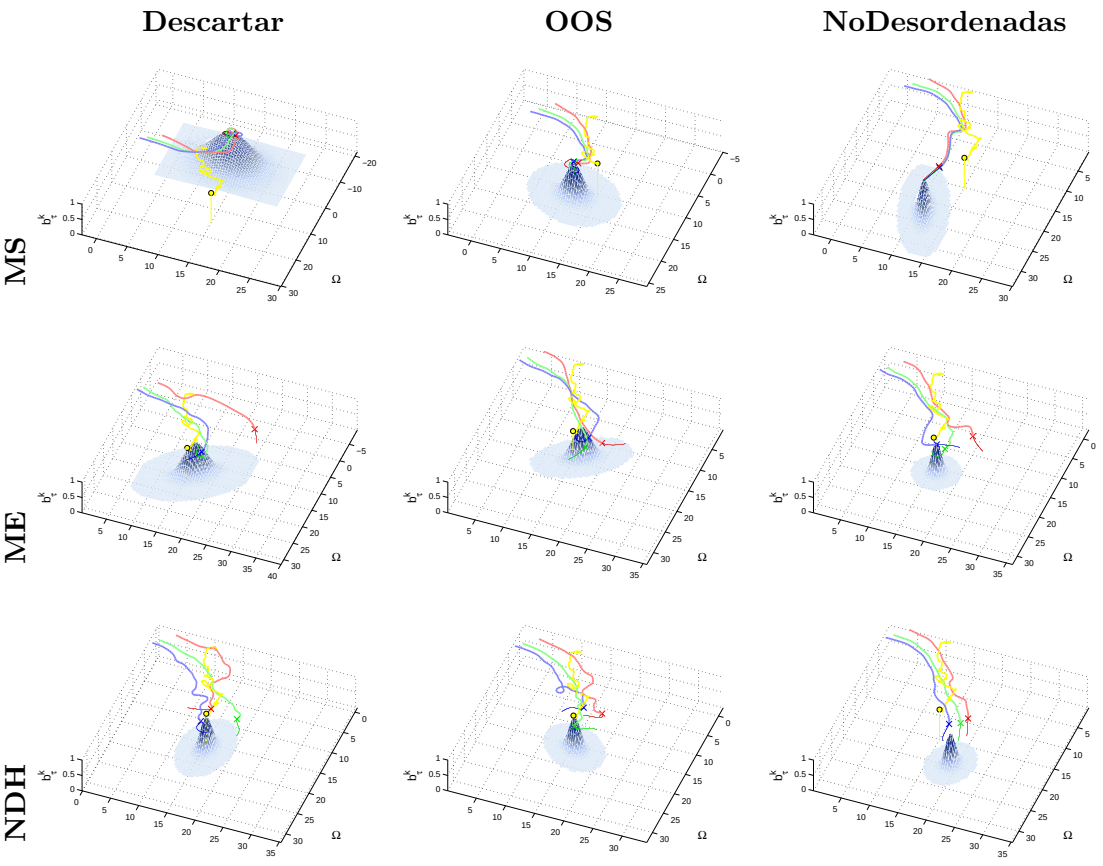


FIGURE 5.3: Simulación de las trayectorias de los agentes utilizando los diferentes filtros en la capa de fusión sensorial.

5.4 Resumen

En este capítulo nos enfrentamos al MTS como una optimización continua, aproximándolo como una optimización lineal a tramos. Las acciones toman valores reales, los sensores son continuos y derivables y la posición del objetivo está modelada como una función de densidad de probabilidad (PDF). Primero hemos resuelto el problema de decisión del MTS utilizando un algoritmo de descenso de gradiente, cuyo núcleo es la estrategia DTH, y segundo, hemos abordado el problema del OOS (las medidas llegan desordenadas a la capa de fusión sensorial) cuando se usa una función de densidad de probabilidad Gaussiana, modificando el filtro de información.

En el caso continuo, incorporamos la novedosa solución para reducir el problema de la localidad cuando tenemos un horizonte limitado de decisión. Hemos aproximado la recompensa esperada en el futuro como una heurística que estima la observación esperada y hemos diseñado una nueva función de utilidad para la coordinación de los agentes. Las acciones que optimizan la función de utilidad son calculadas mediante negociación y el método del gradiente. Esta solución funciona como un controlador de tiempo real de horizonte finito válido para la coordinación multiagente en tareas

de búsqueda en entornos probabilísticos. Los resultados de simulación y los estudios estadísticos muestran que este método mejora trabajos previos en escenarios complejos, donde la probabilidad de detectar el objetivo se acumula más rápidamente.

Los resultados muestran que las estrategias de MTS combinadas con el algoritmo de Gradiente son más efectivas en términos de reducir el tiempo de detección. Además, la estrategia DTH, que incorpora la observación esperada, reduce la miopía de la solución. Los resultados recogidos en la tesis en inglés también muestran que las estrategias MTS locales (DTR y LET) con un horizonte de decisión pequeño se comportan de forma similar a la estrategia de detección y que las estrategias que usan la heurística obtienen mejor rendimiento. Por lo tanto, el horizonte de la estrategia debe ser elegido tan amplio como el tiempo de computación lo permita y la heurística de observación esperada debe ser empleada.

Además, hemos desarrollado un sistema autónomo multiagente que aborda el problema del OOS (las medidas de los sensores llegan desordenadas a las capas de fusión sensorial) para la tarea de búsqueda y seguimiento de un objeto cuando su posición está descrita por una función de densidad de probabilidad Gaussiana. El sistema está compuesto por una capa de manejo de la información que utiliza un filtro de Kalman extendido para el OOS, y un controlador que guía al equipo en la consecución de la tarea. Los resultados proporcionados muestran los beneficios de utilizar el filtro de información OOS propuesto para la búsqueda y el seguimiento, ya que se mejora la estimación de la posición del objetivo.

Chapter 6

CONCLUSIONES

"Is it probable that probability brings certainty?" Blaise Pascal

Esta tesis aborda la Búsqueda de Tiempo Mínimo (Minimum Time Search, MTS) de un objeto perdido con múltiples agentes con dinámica restringida. En la tesis se ha puesto de manifiesto que existen muchas formas de aproximarse al problema, pero que la formulación Bayesiana ha permitido explotar totalmente la incertidumbre intrínseca del MTS, tanto para poder utilizar información previa sobre la localización del objetivo como para calcular las mejores trayectorias de los agentes. En este sentido, se ha estudiado la estrategia de búsqueda más utilizada en la literatura, maximizar la probabilidad de detección del objetivo a lo largo del camino de los agentes, y se ha demostrado que ésta no minimiza necesariamente el tiempo. Por medio de una condición necesaria, se ha establecido que existen multitud de escenarios en los que maximizar la probabilidad no minimiza el tiempo de encontrar el objetivo. Esto nos ha permitido definir el MTS como un problema especial dentro de la teoría de búsqueda óptima que necesita, para resolverlo, nuevas estrategias que tengan en cuenta el tiempo.

Con este propósito, se han desarrollado dos estrategias locales que minimizan el tiempo de detección. La primera, LET, se basa en la definición clásica del tiempo esperado utilizando una ventana de decisión de horizonte N, sin perder la viabilidad del computo. La segunda, DTR, utilizando una función de tiempo descontado, parametrizamos la importancia de las decisiones dependiendo del instante en el que se ejecutan. Ambas estrategias han mostrado, por medio de estudios estadísticos, mejores respuestas para reducir el tiempo de búsqueda del objetivo que soluciones previas.

También se ha mostrado que las estrategias con horizonte limitado no tienen en cuenta el efecto de las acciones en el futuro a largo plazo y que por lo tanto, presentan problemas de miopía. Por esta razón, se ha desarrollado una tercera estrategia, DTH, que incorpora una heurística que aproxima la observación esperada de los agentes en el futuro cuando

una serie de decisiones ya han sido tomadas. Esta estrategia reduce los problemas de localidad siendo mas anticipatoria con respecto al futuro. Se ha diseñado de tal forma que la complejidad del problema no aumenta significativamente y que es fácilmente combinable con cualquier estrategia probabilística local. Esto se consigue modelando la heurística como un sensor y por lo tanto su inclusión en LET o DTR es directa. Se ha mostrado mediante análisis estadísticos que DTH reduce la miopía de los agentes considerablemente.

Para optimizar las estrategias propuestas, se han desarrollado tres soluciones, que implementan la capa de control del sistema autónomo. Para el MTS discreto se propone, por un lado, una solución global y óptima basado en un modelo de programación con restricciones para dominios finitos que únicamente es viable para buscar objetos estáticos, cuando el número de variables de decisión es pequeño, y por otro lado, un método estocástico basado en la optimización de entropía cruzada que a pesar de ser una aproximación resulta ser muy competitivo para la búsqueda de objetivos dinámicos. Para el MTS continuo se ha propuesto el uso de un método de gradiente que permite optimizar las aciones de los agentes de forma descentralizada.

Además, se ha mejorado la capa de fusión sensorial mediante un filtro de información que resuelve el problema de los retrasos en las comunicaciones cuando el objetivo está definido por una función de densidad de probabilidad Gaussiana.

Las soluciones de la capa de control conectadas con la capa de fusión sensorial proporcionan un sistema inteligente y autónomo capaz de encontrar un objeto en movimiento en el menor tiempo posible.

6.1 Resumen de Contribuciones

En esta sección describimos una por una las contribuciones desde el puto de vista de las conclusiones. Además, se proporciona una discusión de los resultados para cada contribución. Las contribuciones principales se condensan en el Capítulo 3, donde se han descrito la formulación del MTS y sus propiedades, y las estrategias Bayesianas propuestas. Además, se ha contribuido con una serie de soluciones en los Capítulos 4 y 5, donde podemos encontrar tres algoritmos diferentes para la capa de decision (CP, CEO y Gradiente) y un algoritmo para la capa de fusión sensorial (OOS).

6.1.1 Formulación General del MTS

El MTS se ha presentado como un problema muy complejo motivado por múltiples aplicaciones donde el tiempo importa. Analizando el MTS como un problema de recolección

de información, la teoría Bayesiana se ha mostrado como una forma intuitiva y potente para aproximarse al problema.

Se ha verificado matemáticamente, mediante una condición necesaria, que maximizar la probabilidad cuando los agentes poseen dinámica restringida, no minimiza necesariamente el tiempo. De hecho, no podemos garantizar que la condición necesaria se cumpla y por tanto el MTS necesita nuevas estrategias que optimicen el tiempo en todos los casos. Aunque los investigadores habían considerado que minimizar el tiempo esperado era demasiado complejo, en este trabajo de tesis se ha mostrado cómo abordar el problema de la importancia del tiempo y cómo minimizarlo en la tarea de búsqueda. Estos resultados teóricos han sido posteriormente corroborados mediante experimentación en los capítulos 4 y 5.

6.1.2 Búsqueda Bayesiana Multiagente de Objetivos Dinámicos

La evaluación de las decisiones de sistema multiagente para buscar objetos dinámicos se ha formalizado siguiendo la teoría Bayesiana. Se ha propuesto una notación probabilística unificada que generaliza la construcción de funciones de utilidad probabilísticas. Estas funciones, que actúan como un filtro de información, han permitido tanto desarrollar el núcleo de la capa de fusión sensorial, como la construcción de estrategias para calcular las acciones de los agentes.

6.1.3 Estrategias Tratables para Minimizar el Tiempo de Búsqueda

Basándonos en trabajos previos como el libro de la búsqueda óptima [Stone, 1975] y la tesis [Mathews, 2008] se ha modelado la capa de decisión de los agentes como un controlador de alto nivel en lazo abierto que, utilizando la información que proporciona la capa de fusión sensorial, optimiza las acciones que los agentes deben realizar. Para que el cálculo de las acciones sea computacionalmente tratable se ha aproximado la optimización, limitando el horizonte de decisión y asumiendo que conocemos el modelo de movimiento del objetivo para estimar su posición. Una vez simplificado el controlador, hemos diseñado tres estrategias para resolver el MTS:

- Tiempo Esperado Local (Expected Local Time, LET): utilizando una ecuación para calcular el tiempo esperado que conserva su validez cuando la probabilidad total es menor que uno, hemos mostrado que podemos diseñar una función de utilidad que minimiza el tiempo para un horizonte limitado. Es decir, los caminos de longitud N realizados por los agentes tienen el menor tiempo esperado comparado con los demás caminos de igual longitud. Por lo tanto, esta estrategia minimiza el tiempo esperado en esa ventana de decisión.

- Recompensa de Tiempo Descontando (Discounted Time Reward ,DTR): utilizando una función de tiempo descontado (una función decreciente que depende del tiempo), que asigna pesos específicos a la probabilidad de detección acumulada, se ha construido una función de utilidad que debe ser maximizada para lograr minimizar el tiempo de detección. Esto ocurre porque las decisiones que se toman con anterioridad son mejor valoradas, modelando así la importancia del tiempo. Por lo tanto, los caminos de los agentes con mayor DTR encuentran el objetivo antes.
- Heurística de Tiempo Descontado (Discounted Time Heuristic, DTH): modelando un sensor que aproxima el valor esperado de observar el objetivo (recompensa a largo plazo), se han reducido los problemas de miopía de estrategias locales o a corto plazo como LET y DTR. Eliminando la localidad se ha mejorado la minimización del tiempo de búsqueda. Además, gracias a la forma en que se ha diseñado la heurística no se incrementa la complejidad del problema ni se aumenta significativamente el tiempo de cómputo en la optimización.

Los resultados han mostrado que LET y DTR reducen el tiempo para encontrar el objetivo y que mejoran estrategias previas existentes en la literatura. LET ha presentado mejor comportamiento general porque minimiza la media, pero en algunos escenarios, la posibilidad de ajustar la importancia del tiempo que ofrece DTR, ha conseguido mejores resultados.

Desde el punto de vista de la miopía, es importante añadir que LET y DTR se calculan para un horizonte fijo y no incluyen ninguna información del futuro, y que por lo tanto han presentado problemas de localidad. De hecho, acumular demasiado rápido las probabilidades cuando el horizonte es pequeño puede hacer que en el futuro se pierdan buenas soluciones. Por ejemplo, en escenarios con objetos dinámicos, los agentes que presentan un comportamiento muy voraz pueden ser llevados a un óptimo local. En este sentido, DTR ha presentado mayor sensibilidad a la miopía.

Una manera de reducir este problema, y además mejorar decisiones futuras, es utilizar la estrategia DTH. La heurística, combinada con LET o DTR, o incluso con la estrategia de máxima detección (D), ha dado buenos resultados a la hora de reducir la localidad de las soluciones y contribuir en la minimización del tiempo. Los estudios realizados, han mostrado que DTR y D son mas sensibles a la heurística, ya que ésta rompe la homogeneidad de las contribuciones a la función de utilidad, mientras que LET únicamente usa la heurística como una “sugerencia”. Por lo tanto, la heurística debe ser ajustada para cada función de utilidad. De todas maneras, se ha observado que las versiones heurísticas de LET, DTR y D son buenas aproximaciones para desarrollar un controlador que funcione en línea para guiar a los agentes en la búsqueda.

Además, se ha visto que, en ciertos escenarios, la condición necesaria para que maximizar la detección también minimice el tiempo, se cumple. En el caso de que la posición del

objeto esté descrita por una distribución de densidad de probabilidad Gaussiana o por distribuciones muy homogéneas, la estrategia de maximizar la probabilidad de detección (o minimizar la probabilidad de no-detección) constituye una buena aproximación para resolver el MTS. Por ejemplo en el MTS continuo, cuando sólo podemos usar un horizonte de decisión pequeño, la utilización de la estrategia de no-detección combinada con la heurística ha dado buenos resultados.

6.1.4 Solución para el MTS Discreto Usando un Modelo de Programación con Restricciones

Modelando el MTS bajo el paradigma de programación con restricciones (Constraint Programming, CP) para dominios finitos, hemos diseñado un algoritmo óptimo para buscar objetos estáticos cuando el número de variables de decisión es pequeño. Utilizando un resolutor de código abierto [Kuchcinski, 2003] se ha implementado la optimización de la estrategia DTR. Aún así, este método sólo resuelve una subinstancia del problema de MTS, ya que asume que el objeto es estático durante el horizonte de decisión y que el sensor es ideal. Para poder solucionar instancias de mayor número de variables de decisión se ha introducido una aproximación que asume que cada posible localización solo se visita una vez.

Los resultados muestran que para pequeñas instancias es una buena solución para su ejecución en línea. Además, la aproximación que no repite localizaciones ha dado muy buenos resultados, en términos de tiempo de optimización y calidad de la solución. Por lo tanto, se ha mostrado que es un método eficaz para instancias de tamaño medio, aunque en algunos escenarios, donde la solución óptima necesita visitar posiciones repetidamente, se comporta de manera subóptima.

6.1.5 Solución Aproximada y Estocástica para el MTS Discreto con Objetivos Dinámicos

Se ha diseñado e implementado un algoritmo aproximado, basado en el método de entropía cruzada [Rubinstein and Kroese, 2004], que soluciona los escenarios con objetos dinámicos y que admite cualquier tipo de sensor, es decir, resuelve el MTS general. Este método, que proporciona soluciones subóptimas, se ha mostrado como una aproximación viable para ser implementada como controlador de los agentes. El algoritmo se ha probado para optimizar todas las estrategias de MTS propuestas.

El análisis de optimalidad para objetos estáticos ha mostrado que el algoritmo aproxima bastante bien el óptimo global (calculado mediante la contribución de la sección 6.1.4), aunque presenta un pequeño offset y problemas de localidad debido a la no convexidad del problema.

Desde el punto de vista de las estrategias se ha observado que este método, utilizando LET y DTR, reduce el tiempo necesario para detectar el objetivo. Además, combinando estas estrategias con la de DTH, los problemas de localidad han sido reducidos.

6.1.6 Solución No Miope para el MTS Continuo

Gracias al trabajo realizado por [Mathews et al., 2007] y [Gan and Sukkarieh, 2010] y a la estrategia DTH propuesta en esta tesis, se ha diseñado una solución descentralizada no miope basada en el método del gradiente que resuelve el MTS. Las soluciones proporcionadas por este método son subóptimas y locales. Una innovadora forma de diseñar la heurística, modelándola como un sensor, permite el diseño de algoritmos informados para resolver el MTS en los casos en los que la observación esperada es demasiado compleja para ser calculada en un tiempo factible. Los resultados han mostrado que la mejora del comportamiento del equipo de agentes es grande en comparación con el coste computacional de utilizar la heurística. Hay que resaltar que aunque esta solución proporciona respuestas muy rápidas, asume que el objeto es estático.

Asumir que el objeto es estático simplifica el problema de estimación y del cálculo del Jacobiano para el método del gradiente, pero en muchas aplicaciones reales, los objetos se mueven. Esta solución se puede aplicar para solucionar MTS con objetos de dinámica lenta, es decir, problemas en los que el objetivo no cambia su posición durante el proceso de negociación de los agentes para calcular las acciones. También, idealmente, se podría recalcular la mejor trayectoria en cada intervalo incluyendo así de forma implícita la dinámica del objetivo. En cualquier otro caso este algoritmo descentralizado es aproximado.

6.1.7 Solución para Resolver el Problema de Medidas Desordenadas en el MTS Continuo

En el caso de que la posición del objetivo pueda ser modelada como una distribución Gaussiana, se ha propuesto una capa de fusión sensorial para resolver el problema de los retrasos en la comunicación de las observaciones entre los agentes. Se ha observado que cuando existen retrasos en la transmisión de la información sobre la posición del objetivo, el rendimiento de la búsqueda y el seguimiento del objeto es deficiente. Abordando el problema del OOS hemos conseguido que el equipo de agentes obtenga una estimación más precisa de la posición del objeto y por tanto sean capaces de calcular decisiones más acertadas. Los resultados han mostrado que utilizando el filtro propuesto como capa de fusión sensorial, el tiempo de búsqueda se reduce y el seguimiento del objeto se mejora.

6.2 Trabajo Futuro y Problemas Sin Resolver

Qué se espera en los próximos años? Profundizando en las posibles líneas futuras de investigación, está claro que la búsqueda óptima se enfocará en encontrar un controlador descentralizado que haga uso de la teoría de la información y de la probabilidad, y en su implementación en plataformas reales. Además, mientras no exista un algoritmo polinómico que resuelva el problema, demostrando que $NP = P$, los investigadores desarrollarán mejores algoritmos aproximados. La dirección de mejora de los métodos es un vector de varias componentes: tipo de sensor, dinámica de los objetivos, miopía, optimalidad, descentralización y eficiencia computacional. Cuando todas esas características sea consideradas dentro del sistema autónomo, el problema será finalmente solucionado.

En términos de la capa de control tenemos que diferenciar las aproximaciones discretas de las continuas.

En la optimización discreta hemos visto que los métodos de muestreo proporcionan soluciones subóptimas debido a la naturaleza estadística de los algoritmos y a la no convexidad del problema. Por lo tanto, una de las mejoras sería el diseño de métodos más eficientes de muestreo donde se puedan utilizar podas informadas para eliminar soluciones peores. También se ha observado que, a veces, cuando se encuentra una solución buena durante la optimización, ésta se pierde debido al suavizado que realizamos en el aprendizaje de la distribución y de la dominancia de otras soluciones subóptimas. La idea de utilizar un archivo o hacer "backtraking" a una solución más prometedora de la región del espacio de búsqueda son posibles maneras de reducir este efecto. La descentralización en la aproximación discreta con una prueba de convergencia puede contribuir a diseñar algoritmos de cooperación en línea mucho más rápidos y eficientes. En este sentido, el modelo de CP se podría convertir en un CP descentralizado y resolverlo con algoritmos distribuidos. En el caso de CEO podríamos, tanto explotar su paralelización (qué es directa), desarrollar el CEO descentralizado, o utilizar la misma idea que se ha aplicado para descentralizar el método del gradiente.

En el plano de la optimización continua, las mejoras se tienen que centrar en tratar el problema de los objetivos dinámicos. Las estrategias probabilísticas con objetivos dinámicos utilizan la estimación de los estados futuros e introducen una gran complejidad dentro de la optimización del gradiente. Así que, una derivación explícita del gradiente, aunque sea aproximada, sería una manera de acercarse a este problema desde el punto de vista de la ejecución en línea. Además, también sería necesario demostrar que se puede descentralizar cuando los objetivos se encuentran en movimiento.

Algoritmos de estimación de la distribución más sofisticados que CEO, como la optimización Bayesiana, también puede mejorar la optimización de la solución. Por ejemplo, teniendo en cuenta las dependencias entre las variables de decisión. En otros campos, nuevos algoritmos, como los métodos aproximados para resolver POMDPs, abren

también una línea muy interesante de investigación. Por ejemplo, servirían para comparar el rendimiento de la optimización, así como para diseñar métodos de programación dinámica específicos para la búsqueda óptima de tiempo mínimo y desarrollar técnicas aproximadas para mejorar su escalabilidad. Recientemente, algunas técnicas que mezclan programación con restricciones con las formalizaciones de POMDP parecen viables para enfrentarse al MTS.

Eliminar la miopía del algoritmo es otra de las complicaciones a tratar en el MTS ya que calcular la recompensa esperada exacta no es computacionalmente tratable dentro de la optimización. Además, el acoplamiento de la información también tiene que ser tenido en cuenta. Una buena línea de investigación sería el desarrollo de heurísticas más precisas y rápidas que aproximaran la observación esperada cuando el objetivo se encuentra en movimiento.

En términos de la capa de fusión, se podría generalizar el algoritmo OOS para una función de densidad de probabilidad cualquiera, no necesariamente Gaussiana. Desde el punto de vista de la implementación, la comunicación entre los agentes debe ser optimizada para proporcionar robustez al sistema.

De todas maneras, pensamos que la mejora más importante, que cambiará el paradigma de la búsqueda óptima, está relacionada con la fusión sensorial dentro de las estrategias de solución. Por ejemplo, se podría plantear una estructura información-tiempo que condense el conocimiento temporal de una forma optimizada. Esto mejoraría la cooperación y la coordinación de los agentes, la comunicación entre ellos, los métodos de optimización y además acelerará los controladores drásticamente. En el futuro, tendremos redes de plataformas sensoriales, auto-organizadas, coordinadas o en cooperación, activas y móviles, heterogéneas, y flexibles, que comunicarán su percepción del mundo y actuarán siguiendo un objetivo común. Este objetivo incluirá aprender del entorno, minimizar el tiempo de ejecución de la tarea y maximizar la recompensa del equipo.

Finalmente, la implementación en plataformas reales y consecuentemente la realización de tareas de búsqueda utilizando vehículos autónomos es el objetivo final de esta investigación. Además, la generalización de estos modelos teóricos para nuevas aplicaciones pueden ser un campo de investigación interesante.

List of Figures

List of Tables

List of Algorithms

Bibliography

Amato, C., Bernstein, D. S., and Zilberstein, S. (2010). Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems*, 21(3):293–320.

Anisi, D. A., Ogren, P., and Hu, X. (2010). Cooperative minimum time surveillance with multiple ground vehicles. *IEEE J AC*, 55(12):2679–2691.

Bar-Shalom, Y. and Chen, H. (2005). IMM estimator with out-of-sequence measurements. *IEEE Trans. Aeros. Electron. Syst.*, 41(1):90–98.

Bellingham, J., Richards, A., and How, J. P. (2002). Receding horizon control of autonomous aerial vehicles. In *Proc. American Control Conf the 2002*, volume 5, pages 3741–3746.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, 2003 edition.

Benkoski, S. J., Monticino, M. G., and Weisinger, J. R. (1991). A survey of the search theory literature. *Naval Research Logistics (NRL)*, 38(4):469–494.

Berger, J., Happe, J., Gagne, C., and Lau, M. (2009). Co-evolutionary information gathering for a cooperative unmanned aerial vehicle team. In *Proc. 12th Int. Conf. Information Fusion FUSION '09*, pages 347–354.

Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2002). The complexity of decentralized control of markov decision processes. *Math. Oper. Res.*, 27:819–840.

Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*. Athena Scientific.

Bertuccelli, L. and How, J. (2006). Search for dynamic targets with uncertain probability maps. In *Proc. American Control Conference*, page 6.

Besada-Portas, E., Lopez-Orozco, J. A., Besada, J., and de la Cruz, J. M. (2009). Multisensor out of sequence data fusion for estimating the state of discrete control systems. *IEEE Trans. Autom. Control*, 54(7):1728–1732.

Besada-Portas, E., Lopez-Orozco, J. A., Besada, J., and de la Cruz, J. M. (2011). Multisensor fusion for linear control systems with asynchronous, out-of-sequence and erroneous data. *Automatica*.

Besada-Portas, E., Lopez-Orozco, J. A., Lanillos, P., and de la Cruz, J. M. (2012). Localization of non-linearly modeled autonomous mobile robots using out-of-sequence measurements. *Sensors*, 12(3):2487–2518.

Blum, A., Chawla, S., Karger, D. R., Lane, T., Meyerson, A., and Minkoff, M. (2003). Approximation algorithms for orienteering and discounted-reward TSP. In *Proc. 44th Annual IEEE Symp. Foundations of Computer Science*, pages 46–55.

Boer, P. T. D., Kroese, D., Mannor, S., and Rubinstein, R. (2002). A tutorial on the cross-entropy method. *Annals of Operations Research*, 134.

Bourgault, F., Furukawa, T., and Durrant-Whyte, H. F. (2003). Optimal search for a lost target in a bayesian world. In *FSR*, pages 209–222.

Bourgault, F., Furukawa, T., and Whyte, H. F. D. (2004). Decentralized bayesian negotiation for cooperative search. In *IEEE Conf. Inteligent robots and Systems*.

Brocas, I. and Carrillo, J. D. (2008). The brain as a hierarchical organization. *American Economic Review*, 98(4):1312–46.

Candy, J. V. (2009). *Bayesian Signal Processing*. John Wiley & Sons.

Cassandra, A. R., Kaelbling, L. P., and Kurien, J. A. (1996). Acting under uncertainty: discrete bayesian models for mobile-robot navigation. In *Proc. IEEE/RSJ Int Intelligent Robots and Systems '96, IROS 96 Conf*, volume 2, pages 963–972.

Cheng, K. C. K. and Yap, R. H. C. (2010). An MDD-based generalized arc consistency algorithm for positive and negative table constraints and some global constraints. *Constraints*, 15(2):265–304.

Clark, J. and Fierro, R. (2005). Cooperative hybrid control of robotic sensors for perimeter detection and tracking. In *Proc. American Control Conference the 2005*, pages 3500–3505 vol. 5.

Dell, R. F., Eagle, J. N., Martins, G. H. A., and Santos, A. G. (1996). Using multiple searchers in constrained-path, moving-target search problems. *Naval Research Logistics*, 43(4):463–480.

Dobbie, J. M. (1974). A two-cell model of search for a moving target. *Operations Research*, 22(1):79–92.

DPI (2006). PLACOMAR-Plataforma de planificación, simulación y control para colaboración de múltiples vehculos autónomos marinos y aéreos (DPI2006-15661-C02-01), Ministerio de Ciencia e Innovación de España.

DPI (2009). System for surveillance, search and rescue in the sea by means of collaboration of autonomous marine and air vehicles (DPI2009-14552-C02-01), Ministerio de Ciencia e Innovación de España.

Eagle, J. N. (1984). The optimal search for a moving target when the search path is constrained. *Operations Research*, 32(5):1107–1115.

Eagle, J. N. and Yee, J. R. (1990). An optimal branch-and-bound procedure for the constrained path, moving target search problem. *Oper. Res.*, 38:110–114.

El-Mane Wongy, Frederic Bourgault, T. F. (2005). Multi-vehicle bayesian search for multiple lost targets. In *Proc. IEEE International Conference on Robotics and Automation*.

Facebook (2011). Facebook engenieering puzzles.

Feillet, D., Dejax, P., and Gendreau, M. (2005). Traveling salesman problems with profits. *Transportation Science*, 39:188–205.

Feller, W. (1966). *An introduction to probability theory and its applications*. Number 2 in Wiley series in probability and mathematical statistics. Wiley.

Flint, M., Polycarpou, M., and Fernndez-Gaucherand, E. (2002). Cooperative path-planning for autonomous vehicles using dynamic programming. In *15th IFAC world conference*. International Federation in Automatic Control.

Furukawa, T. (2002). Time-submimal trajectory planning for discrete non-linear systems. *Engineering Optimization*, 34(3):219–243.

Furukawa, T., Bourgault, F., Lavis, B., and Durrant-Whyte, H. F. (2006). Recursive bayesian search-and-tracking using coordinated uavs for lost targets. In *Proc. IEEE Int. Conf. Robotics and Automation ICRA 2006*, pages 2521–2526.

Gan, S. K. and Sukkarieh, S. (2010). Multi-UAV target search using explicit decentralized gradient-based negotiation. In *Proc. IEEE International Conference on Robotics and Automation*.

Gorji, A. A., Tharmarasa, R., and Kirubarajan, T. (2011). Performance measures for multiple target tracking problems. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1–8.

Grocholsky, B. (2002). *Information-Theoretic Control of Multiple Sensor Platforms*. PhD thesis, The University of Sydney.

Hollinger, G., Ferguson, D., Srinivasa, S., and Singh, S. (2009). Combining search and action for mobile robots. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ICRA'09, pages 800–805, Piscataway, NJ, USA. IEEE Press.

Hsu, D., Lee, W. S., and Rong, N. (2008). A point-based POMDP planner for target tracking. In *Proc. IEEE Int. Conf. Robotics and Automation ICRA 2008*, pages 2644–2650.

ISBA (2009). Statistics in the real world: The search for the scorpion. International Society for Bayesian Analysis.

Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*, volume 1. Cambridge University Press.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101:99–134.

Kagan, E. and Ben-Gal, I. (2006). An informational search for a moving target. In *Proc. IEEE 24th Convention of Electrical and Electronics Engineers in Israel*, pages 153–155.

Koopman, B. (1946). Search and screening. *OEG Report The Summary Reports Group of the Columbia University Division of War Research*, No. 56.

Koopman, B. (1980). *Search and screening: general principles with historical applications*. Pergamon Press.

Kosaka, A., Meng, M., and Kak, A. C. (1993). Vision-guided mobile robot navigation using retroactive updating of position uncertainty. In *Proc. of the 1993 IEEE International Conference on Robotics and Automation*, volume 2, pages 1–7.

Kratzke, T., Stone, L., and Frost, J. (2010). Search and rescue optimal planning system. In *Information Fusion (FUSION), 2010 13th Conference on*, pages 1–8.

Kuchcinski, K. (2003). Constraints-driven scheduling and resource assignment. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 8(3):355–383.

Lanillos, P., Besada-Portas, E., Pajares, G., and Ruz, J. J. (2012). Minimum time search for lost targets using cross entropy optimization. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 602 –609.

Lanillos, P., Javi, Y.-Z., Ruz, J. J., and Besada-Portas, E. (2013). A Bayesian approach for constrained multi-agent minimum time search in uncertain dynamic domains. In *Genetic and Evolutionary Computation (GECCO) Conference*.

Lanillos, P., Ruz, J. J., Pajares, G., and de la Cruz, J. M. (2009). Environmental surface boundary tracking and description using a uav with vision. In *Proc. IEEE Conf. Emerging Technologies & Factory Automation ETFA 2009*, pages 1–4.

Larrañaga, P. (2002). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, chapter An introduction to probabilistic graphical models, pages 25–54. Kluwer Academic Publishers, Boston/Dordrecht/London.

Lavalle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical report.

Lavis, B., Furukawa, T., and Durrant Whyte, H. F. (2008). Dynamic space reconfiguration for bayesian search and tracking with moving targets. *Auton. Robots*, 24(4):387–399.

Lopez-Orozco, J. A., de la Cruz, J. M., Besada, E., and Ruiperez, P. (2000). An asynchronous robust and distributed multisensor fusion system for mobile robots. *The International Journal of Robotics Research*, 19(10):914–932.

Mathews, G. (2008). *Asynchronous decision making for decentralised autonomous systems*. PhD thesis, The University of Sydney.

Mathews, G., Durrant-Whyte, H., and Prokopenko, M. (2007). Asynchronous gradient-based optimisation for team decision making. In *Decision and Control, 2007 46th IEEE Conference on*, pages 3145 –3150.

MathWorks (2010). Mathworks.

Mutambara, G. O. (1998). *Decentralized Estimation and Control for Multisensor Fusion*. CRC Press LLC.

Nettleton, E. W. and Durrant-Whyte, H. (2001). Delayed and asequent data in decentralized sensing networks. In *Proc. of SPIE Conf n 4571*, pages 1–9.

Pajares, G., Ortiz, J. R., Lanillos, P., Guijarro, M., la Cruz, J. D., and Santos, M. (2008). Generacion de trayectorias y toma de decisiones para uav's. *RIAI*, 5:83–92.

Papoulis, A. and Pillai, S. (2002). *Probability, random variables, and stochastic processes*. McGraw-Hill electrical and electronic engineering series. McGraw-Hill.

Pineau, J., Gordon, G., and Thrun, S. (2006). Anytime point-based approximations for large POMDPs. *J. Artif. Int. Res.*, 27(1):335–380.

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons.

Ross, P. E. (2011). Top 11 technologies of the decade. *IEEE Spectrum Magazine*, 48(1).

Rubinstein, R. Y. and Kroese, D. P. (2004). *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-carlo Simulation (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Russell, S. J., Norvig, P., Candy, J. F., Malik, J. M., and Edwards, D. D. (1996). *Artificial intelligence: a modern approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Sarmiento, A., Murrieta-Cid, R., and Hutchinson, S. (2009). An efficient motion strategy to compute expected-time locally optimal continuous search paths in known environments. *Advanced Robotics*, 23(12-13):1533–1560.

Savelsbergh, M. W. P. and Sol, M. (1995). The general pickup and delivery problem. *Transportation Science*, 29:17–29.

Simon, H., Egidi, M., Viale, R., and Marris, R. (2008). *Economics, bounded rationality and the cognitive revolution*. Edward Elgar.

Singh, S. and Krishnamurthy, V. (2003). The optimal search for a markovian target when the search path is constrained: the infinite-horizon case. *Automatic Control, IEEE Transactions on*, 48(3):493 – 497.

Smallwood, R. D. and Sondik, E. J. (1973). The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088.

Smith, S. C. and Seiler, P. (2003). Estimation with lossy measurements: jump estimators for jump systems. *IEEE Trans. Autom. Control*, 48(12):2163–2171.

Stone, L. D. (1975). *Theory of optimal search*. Academic Press, New York.

Stone, L. D. (1989). What's happened in search theory since the 1975 lanchester prize? *Operations Research*, 37(3):501–506.

Stone, L. D. (1992). Search for SS "Central America": Mathematical treasure hunting. *Interfaces*, 22(1):32–54.

Stone, L. D. (2011). In search of air france flight 447.

Tisdale, J., Kim, Z., and Hedrick, J. (2009). Autonomous path planning and estimation using UAVs. *Robotics and Automation*.

Trummel, K. E. and Weisinger, J. R. (1986). The complexity of the optimal searcher path problem. *Operations Research*, 34(2):324–327.

Washburn, A. R. (1998). Branch and bound methods for a search problem. *Naval Research Logistics (NRL)*, 45(3):243–257.

Yang, K., Gan, S. K., and Sukkarieh, S. (2010). An efficient path planning and control algorithm for RUAV's in unknown and cluttered environments. *Journal of Intelligent and Robotic Systems*, 57(1-4):101–122.

Yang, Y. (2005). *Cooperative Search by Uninhabited Air Vehicles in Dynamic Environment*. PhD thesis, University of Cincinnati.

Yang, Y., Minai, A., and Polycarpou, M. (2002). Decentralized cooperative search in uav's using opportunistic learning. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*.

Yang, Y., Minai, A., and Polycarpou, M. (2004). Decentralized cooperative search by networked uavs in an uncertain environment. In *Proc. American Control Conference the 2004*, volume 6, pages 5558–5563.

Zhang, K., Li, X., and Chen, Y. (2003). Multi-sensor multi-tracking with out-of-sequence measurements. In *Sixth International Conference of Information Fusion*, pages 672–679.

Zimmerman, S. (1959). An optimal search procedure. *The American Mathematical Monthly*, 66(8):690–693.