Model-based Techniques for Dependable Decision Making in Groups of Agents Operating Autonomously

A Dissertation Presented

by

Meng Wang

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

December 2011

Stony Brook University

The Graduate School

Meng Wang

We, the dissertation committee for the above candidate for the Doctor of Philosophy degree, hereby recommend acceptance of this dissertation.

Dr. Alex Doboli, Dissertation Advisor
Associate Professor, Department of Electrical & Computer Engineering

Dr. Xin Wang, Chairperson of Defense
Assistant Professor, Department of Electrical & Computer Engineering

Dr. Sangjin Hong,
Associate Professor, Department of Electrical & Computer Engineering

Dr. Monica Fernandez-Bugallo,
Associate Professor, Department of Electrical & Computer Engineering

Dr. George Kamberov,
Associate Professor, Department of Computer Science, Stevens Institute of Technology

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Model-based Techniques for Dependable Decision Making in Groups of Agents Operating Autonomously

by

Meng Wang

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2011

Massively distributed embedded systems are rapidly emerging as a key concept for many modern applications. Autonomous agents with mobile sensors are breakthrough concept in technology. However, providing efficient and scalable decision making capabilities to such systems is currently a significant challenge, especially to have flexible strategies with predictable performance in hard-to-predict conditions.

My thesis first proposes a goal-oriented model to allow automated synthesis of distributed controllers, which implement and interact through models of different semantics. Scalability of descriptions is realized through defining the nature of interactions that can occur among decision modules while leaving to task of optimally implementing these interactions by the execution environment. Applications with data acquisition for CPS system are offered.

The thesis also proposes an approach to performance predictive collaborative control of autonomous agents operating in environments with fixed targets. A trajectory generation algorithm which considers the physical characteristics of autonomous mobile agents, for example, dimensions, weight, velocity constraints and many others. is used in modeling. An Integer Linear Programming based model is used to optimize collaboration to achieve maximum task accomplishment and flexibility. It also offers detailed experimental insight on the quality, scalability and computational complexity of the proposed method.

Another important challenge for Cyber Physical Systems is data acquisition through groups of mobile agents to optimize monitoring. Each agent optimizes locally dose not necessarily result in overall optimization without global predictions. An asynchronous interaction scheme using gaming theory between agents to maximize the utility of the acquired data is purposed. Experiments study the effectiveness of the scheme in comprehensive data acquisition while minimizing redundant data collection.

Contents

List of Figures

List of Tables

Acknowledgments

The completion of my dissertation and subsequent Ph.D. has been a long journey. Looking back, I am very grateful for all I have received throughout these years. All the help and support from professors, colleagues, friends, family and other kind people led me where I am now. I would like to give special thanks to the following people here particularly.

In the first place I would like to record my gratitude to my adviser, Dr. Alex Doboli, for his supervision, advice, and guidance from the very early stage of my Ph.D. study as well as giving me extraordinary research experiences through out the work. He provided me encouragement and support in various ways. His truly scientist attitude and knowledge inspire and enrich my growth as a student and a researcher. Without him, This thesis would not have been completed or written. One simply could not wish for a better advisor.

I gratefully acknowledge my committee members: Dr. Alex Doboli, Dr. Xin Wang, Dr. Sangjin Hong, Dr. Monica Fernandez-Bugallo, and Dr. George Kamberov, for their time and effort in guiding and reviewing this work, and attending my dissertation defense. Thanks to Dr. Wendy Tang for being my preliminary defense committee members. All of their advice is greatly appreciated.

I'd also like to give a heartfelt, special acknowledge to all of my colleagues in VLSI Systems Design Laboratory for their encouragement and support over the last few years.I wish to thank Pengbo Sun, Varun Subramanian, Cristian Ferent, Anurag Umbarkar, Yang Zhao, Ying Wei, Michael Gilberti, Sankalp Kallakuri, Junling Zhou, Jing Gao, Hui Zhang and Yulei Weng for their advice and their willingness to share their bright thoughts with me, which were very fruitful for shaping up my ideas and research. My graduate studies would not have been the same without them.

My gratitude is also extended to Professor Thomas Robertazzi, Professor Daniel Curiac, Dan Pescaru and Codruta Istin, from Automatics and Computers at Politehnica University(Romania), Professor Simona Doboli, from Department of Computer Science at Hofstra University, inspired, assisted and encouraged me in various ways during my PhD research work. It's my honor to have them as co-authors of my publication.

I am deeply indebted to my friends at Stony Brook University, Yao Li, Pengbo Sun, Xiaolan Ba, Yi Jiang, Zhongkui Tan, Keyi Chen, Shiyuan Zhang, Lei Wang, Miao Zhao, Ping Hao, Yajing Zhao, Lin Sun and Jianping Schoolman and Rong Cai. It was a pleasure to share joy and sadness of life with wonderful people like them and I will never forget their thoughtful support at the most needed time in my life.

I thank the Kepler Asset Management Company for the wonderful experience of being a intern. My thanks go out to Bin Fang, Xin Chen, Yuxiang Gao, Yao Li, Shiyuan Zhang for enabling me to do this job successfully.

Many people on the faculty and staff of Department of Electrical Engineering at Stony Brook University assisted and encouraged me during my course of studies. I am especially grateful to Profs. Yuanyuan Yang, Rachel Ingrassia, Anthony Olivo, Deborah Kloppenburg, Lisa Chichura for all that they have taught me and all the help they provided. I was also greatly inspired pedagogically by Prof. Stephen Sussman-Fort,for whom I was a Teaching Assistant for more than two years, and I thank the students whom I was privileged to teach and from whom I also learned much.

Of course no acknowledgments would be complete without giving thanks to my parents, Guizhen Li and Qiansheng Wang. Both have instilled many admirable qualities in me and given me a good foundation with which to meet life. Theyve taught me about hard work and self-respect, about persistence and about how to be independent. I also want to thank uncle Dakuan Zhang, aunt Lisa Wang and nephew Victor Zhang for their generous support and help. This dissertation is also a gift to my grandparents Nanping Wang, Yuqing Chen, Jianhua Li and Shuzhen Zhang for their deep love since I was born.

Finally, I would like to address my deepest thank to my husband Yingwu Wang. Through his love, patience, support and unwavering belief in me, I've been able to complete this long dissertation journey. He has also brought me so many happy and beautiful memories throughout this journey. Thanks to my son, Terrence YiTeng Wang, you are the best encouragement when I ever think of giving up.

Of course, despite all the assistance provided by Prof. Alex Doboli and others, I alone remain responsible for the content of the following, including any errors or omissions which may unwittingly remain.

Chapter 1

Introduction

This chapter introduces the motivations, goals and contributions of this thesis. It also states the overall design flow of the proposed model-based techniques for dependable decision making in groups of autonomously agents. More precisely, this chapter (i) explains the advantages and the challenges of mobile distributed embedded systems; (ii) discusses the advantages of parameterized decision making procedures; (iii) gives an overview of the proposed model-based techniques for dependable decision making, including location-aware, flexible task management for collaborative and dependable distributed data acquisition strategies; and (iv) provides applications of data acquisition for CPS.

1.1 Motivation and methods overview

With sensing and electronic devices becoming extremely cheap and small in size, massively distributed embedded systems are rapidly emerging as a key concept in almost every field of modern technology. They are widely applied in infrastructure management, environmental monitoring, energy conservation, health care, homeland security, manufacturing, and many others. Research on distributed control and decision making of such embedded systems are are still going on ([1] to [6]). Distributed embedded systems enable significantly superior decision making qualities since related problems may be tackled together instead of separately. This helps improving the robustness of the systems as experience has shown that many disasters occur due to unaccounted correlations between sub-systems.

Existing approaches focus on either centralized control or on local control [7], sometimes using ideas inspired from social or biological systems. Centralized control is well understood and easy to realize, however, drawback of centralized control is its lack of scaling for large systems. In contrast, local control works well for large systems but, with the exception of simple situations, its overall performance is hard to predict and usually not optimal.

Providing reliable and efficient decision making capabilities for massively distributed embedded systems currently represents a main challenge [7]. We believe the main challenge comes from the following aspects:

- Providing scalable descriptions for such applications.
- Hard to predict, unknown environmental conditions like wind, rain clouds, bird flocks, and many more, can significantly influence the trajectories and speed of mobile agents and thus render any pre-computed plans infeasible.

We believe that an important challenge arises due to the wide variety of interactions that emerge among the composing sub-systems, some of which are hard to anticipate a-priori, or might change their importance dynamically during execution.

An ideal distributed decision making mechanism should possess three main characteristics: flexibility, scalability, and predictability. We argue that such systems need to be built with parameterized control procedures and different decision making strategies should be dynamically introduced or removed depending on the specific optimization goals and operation conditions of the application. A key component of the approach is a suitable model and specification notation for massively parallel applications.

This thesis presents a novel control model and the related specification constructions for developing massively distributed embedded applications. It defines the operation goals of each sub-system (e.g., the criteria to be maximized or minimized during execution) and the physical capabilities of a module to achieve a certain goal (such as its maximum processing speed, or highest bandwidth). Different interaction types are introduced depending on the way the sub-systems influence each other's goals and capabilities. These interactions are optimally implemented by the execution environment.

Cyber-Physical Systems (CPS) are expected to continuously provide optimized decisions based on accurate monitoring of environmental conditions [12]. Often, the execution platform for CPS is a network of mobile data sensing and processing systems, such as robots or other mobile agents.

Mobile agents, a modern CPS, perform a large set of activities, including computation of trajectories and identification of the control parameters for traveling along a trajectory, signal sampling and processing (including image processing, data integration), as well as communication with other mobile agents. In many real world scenarios, activities such as target detection and handling, and assessment of the results, are also performed online by mobile agents [32, 33, 34, 35]. In many instances, a group of mobile agents must operate collaboratively to tackle complex tasks [43, 44, 42, 45, 46]. The nature of collaboration is often decided dynamically at run time, depending on the context-specific situations. For example, a mobile agent might not be able to meet the deadlines set for its tasks due to unforeseen overheads, such as the time required to avoid moving obstacles. In this case, the agent might inquire collaborations from neighboring agents to jointly perform tasks. Agents with more flexible deadlines might decide to satisfy the request, and participate in collaboration.

Scalability is one of the main challenges in distributed control theory of mobile agents. Present control techniques are restricted to single or small numbers of agents. Although there are some very interesting ongoing work in swarm control [36, 37, 38, 39, 40, 41], the theory of scalable and robust distributed control is still largely unstudied.

Another important challenge is to offer predictable and reliable operation in hard-to-predict environments and situations. Traditionally, reactive control has been a commonly accepted solution for situations that cannot be characterized off-line. Depending on conditions identified online, the controller selects the most suitable response from a set of predefined strategies. Each response strategy is characterized by specific outcomes and performance, such as execution time, energy consumption, and so on. While certain "fixed-point" properties can be proven for reactive behavior (like stability and reachability) [45, 51, 52], other properties, which depend on dynamic attributes (e.g., the frequency of being in a state), are harder to prove unless restrictive functioning conditions are assumed. Thus, important performance attributes, i.e. execution time and resource (energy) consumption, are hard to correctly estimate and guarantee for reactive control procedures. The alternative to reactive procedures are off-line static control methods [44, 53]. These methods work very well if the operating conditions and the environment are fully known, and hence the agent's behavior is deterministic. The performance of the control methods can be precisely estimated in this case. However, static methods have little or no flexibility in adapting to unknown situations.

This thesis presents our modeling procedure expressing agent trajectory generation. The modeling procedure is part of distributed control algorithms for large-scale groups of mobile agents, as it captures the moving of agents through the physical space. The modeling procedure considers the physical characteristics of agents, e.g., dimensions, weight, and velocity constraints, etc. Multi-mode behavior of agents are adopted to adapt to the unknown or changing environmental conditions and goals. In addition, the agent mobility is modeled using the trajectory generation algorithm by Yakimenko [42].

This thesis proposes an approach to performance predictive collaborative control strategies for efficient operation in dynamic conditions with fixed targets. The thesis also offers detailed experimental insight on the quality, scalability and computational complexity of the proposed method.

Data acquisition is an essential task of CPS. Guaranteeing dependable and predictable operation is challenging considering that the mobile agents operate in environments with many unknown and changing attributes.

A main challenge is data acquisition through the mobile agents to construct precise models while minimizing the cost of creating the models. Model-based CPS uses dynamically-constructed models to make decisions during operation. For example, consider a CPS for environmental monitoring and protection, which needs to detect the position of zones polluted with toxic substances as well as the nature and level of the toxic substances. The decisions on how to dispatch the mobile agents to maximize pollution detection and the utility of the acquired data samples are performed based on a model that predicts the position of spills and their dynamics over time.

The distributed data acquisition problem also originates a mixture of collaborative and competing actions between the participating agents. Devising interaction schemes to maximize data acquisition utility while agents can collaborate or compete is still a challenging problem. Resource allocation, e.g., the frontend resources used for localizing the sound source by a node and the selected data communication paths, determine main factors defining the error of the models used in decision making. The quality of resource allocation schemes depends to a high degree on the characteristics of the environment, e.g., ambient noise, and monitored phenomena, like trajectory.

This thesis proposes a novel, asynchronous interaction scheme of data acquisition between agents that operate decoupled for most of the time. The scheme is based on a mathematical model of data acquisition utility, and how utility changes as more data is acquired. The model captures aspects, like gaining new insight into the process, increasing the confidence of the models used in decision making, and the collaborative-competitive aspect of agent interactions. The model is then utilized to infer the selection ratios that decide how alternative sampling steps are used by an agent to maximize the data acquisition utility, while the agent operates decoupled from the other agents. Experiments study the effectiveness of the proposed scheme in producing comprehensive data acquisition while minimizing the amount of redundant data collected by the decoupled autonomous mobile agents.

One thing to clarify is that collision avoidance is not part of the work because the modeling method considers only point agents (agents do not have dimensions). Collision avoidance can be added to the decision making scheme by incorporating the probability of having two agents sample points closer than a minimum distance and then devising schemes that minimize this probability. Collision avoidance methods are discussed in the literature by Tomlin et al. and Sastry et al. [84, 85, 86, 87], among others. The proposed decision making scheme consider macro decisions, such as the likelihood to stay on deviate from the predefined trajectory collision avoidance would be a micro level decision.

1.2 Goals and contributions

The goal of this thesis is to develop model-based techniques for dependable decision making in groups of autonomously agents. The proposed model should have three main characters (i) flexibility to adapt to partially unknown or changing environment, (ii) scalability to large number of agents, and (ii) predictability of the resulting performance.

The novel contribution of this thesis include:

- One semantic model to allow automated synthesis of controllers of massively distributed embedded systems. The model is implemented through models of different semantics. Scalability of descriptions is realized through defining the nature of interactions that can occur among decision modules while leaving the task of optimally implementing these interactions to the execution environment. The notation defines the operation goals of each sub-system (e.g., the criteria to be maximized or minimized during operation) and the physical capabilities of a module to achieve a certain goal. Different interaction types are introduced depending on the way sub-systems influence each other's goals and capabilities. Compared to similar work, the proposed model and specification notation differs in that they focus on optimal goal satisfaction in massively distributed systems and not on algorithmic descriptions. Therefore, the constructs are orthogonal to existing notations as they concentrate on the interactions between groups of nodes, or nodes and environment and less on the behavior of the individual nodes. This simplifies specification and helps scalable decision making.

- An approach is to perform predictive collaborative control of mobile agents operating in hard-to-predict environments with fixed targets. An Integer Linear Programming model is proposed to describe the dynamic collaboration.

- An adaptive interaction scheme is used for solving the optimization problem of acquiring data distributed in time and space through a set of mobile decoupled agents. The adaptive interaction scheme is based on game theory to maximize the utility of the acquired data while minimizing redundant data collection.

- The proposed decision making scheme is a mixture of both static and dynamic decision making. Static decision making aims at preplanned or known activities, such as travel along predefined trajectories, and is computed off line using a centralized method. Dynamic decision making performs aolaptations adapt to deviations from original plans or each agent's own interest. It is computed online using a decentralized method. The two parts are discussed in Chapters 3 and 4 respectively.

1.3 Examples of Data Acquisition in CPS

Data acquisition plays an significant role in Cyber Physical Systems. It is not overstated that most of the CPS applications require data acquisition to operate successfully from environment, human users, or various databases. Autonomous mobile agents have a lot of advantages over fixed data collection units. First of all, the mobility makes it possible to do large-scale data acquisition with relatively few agents. They are more adaptive to uncertain environments, changeable topology and more robust as a whole system. There are also many examples in the field of homeland security, climate research, contaminant and pollution control, and health care, where data acquisition is one of the main tasks of the systems composed of groups of autonomous mobile agents. The following are some examples of data acquisition for such systems.

Radiation Level Detection

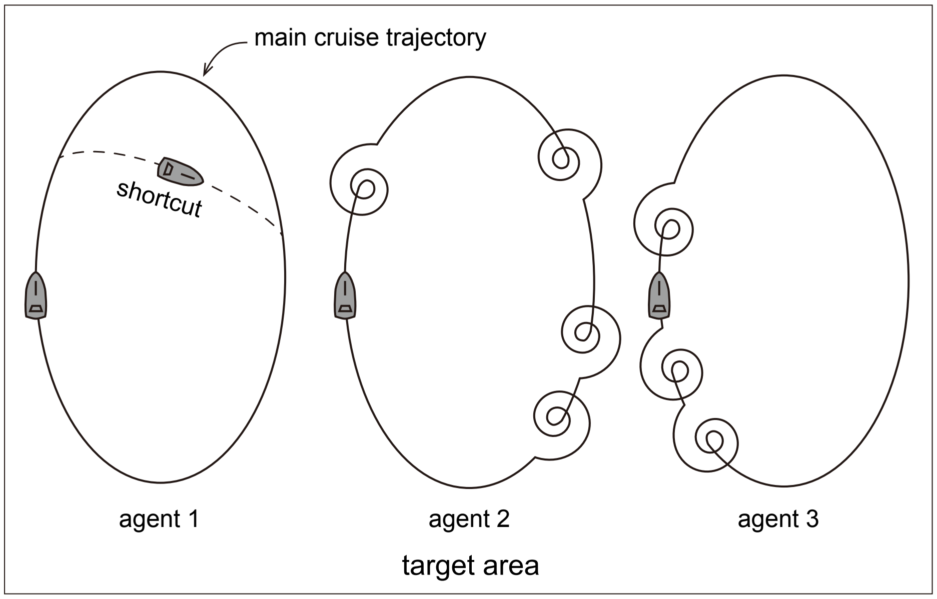


Figure 1.1: Main trajectories and exploration paths

One example is radiation level detection using groups of aerial or marine autonomous agents. Using autonomous agents such as Unmanned Aerial Vehicle (UAVs) and Unmanned Marine Vehicle (UMVs) is necessary in dangerous environments, such as recently areas affected severely by Japan Nuclear Crisis. In this example, groups of UAVs and UMVs cruised along their main trajectories (Figure 1.1) and performed data collection tasks. The goal was to collect as much useful information as possible during a certain time range. To fulfill this purpose, each individual autonomous agent may decide online to follow an exploration path that deviates from the main trajectory to collect data according to the collection strategies.

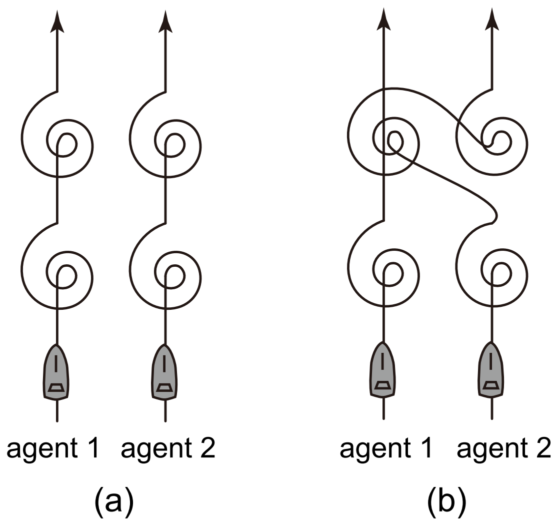


Figure 1.2: Exploration paths and collaboration:(a) Each agent do decision making to generate exploration paths to collect more data; (b) Collaboration between agent 1 and agent 2: agent 2 took part of the tasks from agent 1

In general, each autonomous agent makes online decisions to generate exploration paths (Figure 1.2(a)). Data integration is performed during collection. The result can be sent to another agent for collaboration purposes, or to its own decision making. The decision making points may be preset to certain time interval or distance intervals, and a temporary decision making point may be inserted when abnormal data appears or the results of data acquisition triggers such an event.

The main trajectories are preset off-line in the target areas, but may also be changed online when the following changes occur:

1. *Deadline:* The main trajectories may need to be altered to make shortcuts or extensions when the deadline of the whole task changes to achieve a goal.

2. *Working Environment:* Some conditions in the working environment may make the data collected in certain areas invalid temporarily, or may even result in the inability

to perform the tasks. Such conditions include wind direction or water level change, ocean streams, extreme weather (e.g. high wave, storm), and many more.

3. *Pop-up targets:* There are situations where pop-up targets appear to some agents and they may need to generate additional exploration paths, or change main trajectories temporarily within allowance of the deadline if the position of the pop-up targets is far away. Such situations include the need more data in certain areas due to missing agents.

Collaboration happens when some agents are not able to finish their task within deadline and others have enough flexibility to help. Flexibility here can refer to spare time, or energy source. Figure 1.2(b) illustrates such a case. Agent 1 was unable to finish its tasks within the deadline so it sent out an inquiry to the other agents to ask for help. Agent 2 who is nearby received the inquiry and performed decision making by calculating its cost function (flexibility) according to its own data collecting strategies and decided to help. Agent 2 then responds to agent 1's inquiry and goes along the newly generated trajectory. Agent 1 skips one exploration path and goes directly to the next task.

Decision making and data collection procedures are performed in a distributed manner and only loosely coupled information that is necessary is exchanged occasionally. The information exchange is for collaboration to maximize overall information gain within permitted time range. Instead of exchanging all the collected data with other agents for decision making, only high level interpreted data can be sent out. This saves bandwidth, energy, and exchange time, thus improves the efficiency of the on-line decision making system. The communication between agents contains the following two aspects:

- *Data exchange:* data exchange may occur in two situations: fixed time interval or event trigger. The trigger event can be abnormal data detected as stated above, discovered pop-up targets, reconstruction of sub-groups, change of topology, or addition/loss of agents.

- *Status exchange:* status exchange includes the following two aspects: emergency tasks such as calling for help; or changing main trajectories. Agents should update their status with their colleagues if the two situations occur.

The above schemes provide a complete way of acquiring maximum data from the targeted area without violating the deadline. There are also some other applications in this category, for example, toxic gas release detection, or liquid/solid contamination examination.

Factory/Plant Cleaning Robots

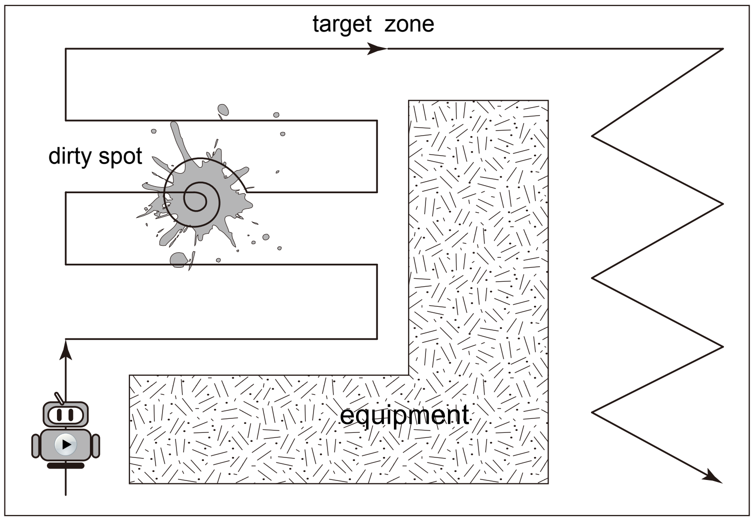


Figure 1.3: Example trajectories for factory cleaning robots

The following example is a team of cleaning robots who operate autonomously to perform factory cleaning tasks. The goal is to finish a set of cleaning tasks in target zones within certain time range, and as fast as possible. Each robot has a preset trajectory to go along in its target zone, and is able to do four actions: watering, sweeping, brushing, and drying. As shown in Figure 1.3, when a dirty spot is detected, the set of four actions will be performed sequentially. Dirty data is collected while moving and data integration is executed before information is sent out for collaboration purposes.

In this example, we assume that the original contamination level data contains four categories at each data point, which include grease level, dust level, harmful level, and damp level. The data point in data pool will be 3D to 5D depending on the dimension of the position (2D or 3D) and whether it contains time information. If all the data are transferred to the other agent for collaboration purposes, the data is abundant and some may not be necessary for the other agent for decision making. Instead of sending the four contamination level data at each point, sending the results of data integration would save communication time and energy. One example of data integration method in this case is to calculate weighted sum of the four contamination level data according to their importance, and sending one data instead:

$$C(x, y) = \sum_{i=1}^{4} \alpha_i * D_i \tag{1.1}$$

α_i is the weight of each factor, which may be affected by the harmfulness of the contamination or time/energy to clean this kind of contamination. D_i is the dirty level of the four kinds of contaminations. The level usually is represented by the total amount or density of contaminations.

The robot that received the collaboration requests containing $C(x, y)$ will calculate its own time and energy consumption to go to help, and will make decisions accordingly. All the robots who received requests also send out their decision to avoid performing the same task repeatedly.

Collaboration may occur in many different forms, and the following are three typical cases:

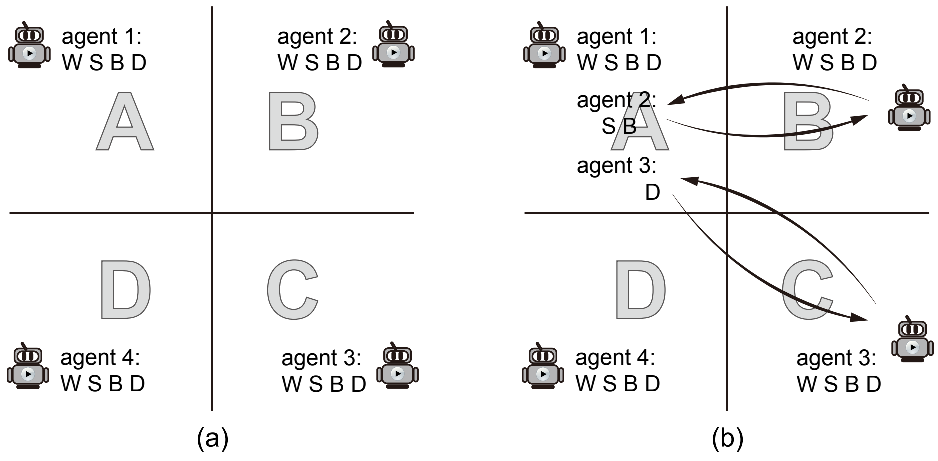


Figure 1.4: Collaboration of a team of cleaning robots (W:wartering, S:sweeping, B:brushing, D:drying)

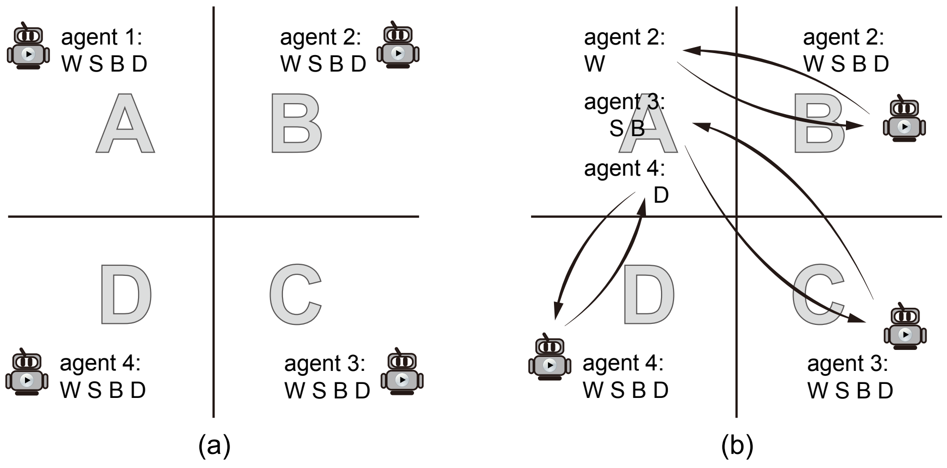


Figure 1.5: Collaboration of a team of cleaning robots (W:wartering, S:sweeping, B:brushing, D:drying)

1. *Big dirty areas detected:* Four robots form a team to do cleaning tasks in target zones. Each robot is assigned to one zone originally, as shown in Figure 1.4(a). During contamination level data collection and cleaning tasks, robot 1 detected a large dirty area. After collecting data from the dirty area, robot 1 decided that it cannot finish the cleaning task before the deadline, so it sent out an inquiry to ask for help. Robots

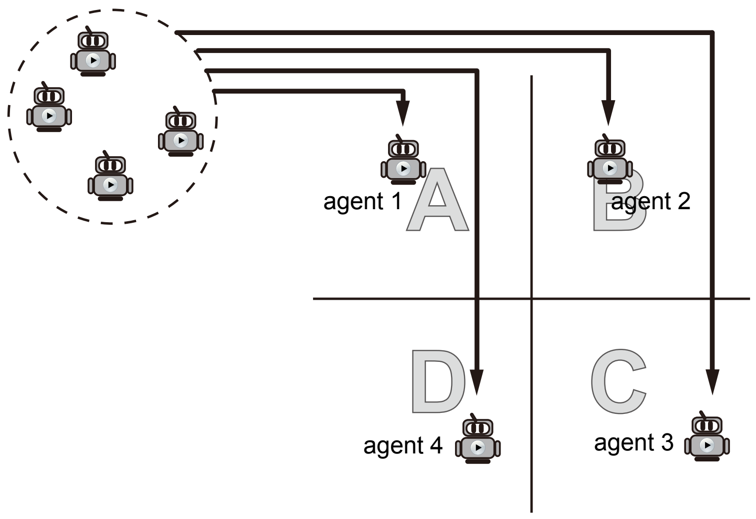


Figure 1.6: Data exchange of a team of cleaning robots

2, 3 and 4 received the inquiry, calculated their own time consumption and robot 1 responded by informing other robots. Figure 1.4(b) is one example of optimized solutions.

2. *One agent is broke or is out of power:* Robot 1 retires from the tasks in this case. In addition to sending out an inquiry for help, robot 1 needs to send out the data it collected already. The results of data integration can be sent instead of the whole data pool. Figure 1.5(b) shows an option that robots 2, 3 and 4 performed robot 1's task to clean zone A.

3. *Sharing collected data:* In some cases, robots may collect data useful for other robots. As shown in Figure 1.6, robots 4 and 3 passed zones A and B respectively on their way to their own target zones D and C. The data collected on the passage was useful to robots 1 and 2. Integrated data was sent to robots 1 and 2 respectively after robots 3 and 4 entered zone D and C.

1.4 Thesis outline

This thesis is organized as follows:

- Chapter 2 proposes a goal-oriented description for CPS and dynamically-constructed models to make decisions during operation. It defines the operation goals of each sub-system and allows automated synthesis of distributed controllers depending on the way the sub-systems influence each other. These interactions are optimally implemented instead of preset.

- Chapter 3 presents distributed control methods for large-scale groups of autonomous vehicles. It introduces the ideas of performance predictive collaborative control of autonomous mobile agents operating in environments with fixed targets, and summarizes the trajectory generation algorithm that is used to model and simulate the moving vehicles. Experimental results for the algorithm are also offered.

- Chapter 4 proposes a novel, asynchronous interaction scheme between agents to maximize the utility of the acquired data. An adaptive interaction scheme is described and an algorithm for dependable decision making strategy originating from game theory is provided. Experiments study the effectiveness of the scheme in comprehensive data acquisition while minimizing redundant data collection with comparison of data acquisition strategies in literature.

- The final chapter presents the conclusion.

Chapter 2

Model and Specification of Massively Distributed Embedded Systems

2.1 Introduction

Massively distributed embedded systems are rapidly emerging as a key enabling concept for many modern applications in infrastructure management, environmental monitoring, energy conservation, health care, homeland security, manufacturing, and many other fields [1, 2, 3, 4, 5, 6]. This is not only due to sensing and electronic devices becoming extremely cheap and small in size (thus, deployable in large numbers) but also because of the potential of having significantly superior decision making quality if related problems are tackled together instead of separately. Moreover, this helps in improving the robustness of the systems as experience has shown that many disasters occur due to unaccounted correlations between the sub-systems. Providing reliable and efficient decision making capabilities to massively distributed embedded systems currently represents a main challenge [7].

The envisioned decision making paradigm is different from existing approaches, which either focus on centralized control or on local control [7], many times using ideas inspired from social or biological systems. Centralized control is well understood and reliable but does not scale well for large systems. In contrast, local control works well for large systems but with the exception of simple situations, its overall performance is hard to be captured. We argue for a distributed decision making mechanism in which parameterized control procedures implementing different strategies are dynamically introduced or removed depending on the specific optimization goals and operation conditions of the application. The controllers operate according to different models depending on the nature of their decision making. For example, reactive controllers use physical inputs for local control but operate under the constraints set by more abstract decision making procedures, which analyze broader situations based on aggregated (abstracted) data and procedures. The proposed decision making mechanism is envisioned to be flexible, scalable, and predictable. A key component of the approach is a suitable model and specification notation for massively parallel applications.

This chapter presents a novel control model and the related specification for developing massively distributed embedded applications. We believe that a main challenge in providing scalable descriptions for such applications is due to the wide variety of interactions that emerge among the composing subsystems, some of which are hard to anticipate a-priori, or might change their importance dynamically during execution. The proposed solution is to describe the nature of interactions that might occur among modules while leaving the task of optimally implementing these interactions to the compiler and execution environment.

More specifically, the proposed model defines the operation goals of each sub-system (e.g., the criteria to be maximized or minimized during execution) and the physical capabilities of a module to achieve a certain goal (such as its maximum processing speed, highest bandwidth, etc). Different interaction types are introduced depending on the way the sub-systems influence each others goals and capabilities. The specification is compiled into a network of Decision Modules (DMs), which use reactive models to control decisions at the physical level, and more abstract formalisms, such as Task Graphs and Markov Decision Processes, to perform broader, more strategic decisions. The multi-semantic DMs interact through (i) constraints by which the strategic modules restrict the reactive DMs to guarantee satisfaction of the global goals and (ii) feedback offered by the reactive DMs about the feasibility of the constraints. Note that the methods presented in Chapter 3 and 4 are methods at the physical level as well as Task Graph level.

Section 2.2 summarizes the related work. Section 2.3 defines the distributed control model and the main language concepts. Finally, conclusions are offered.

2.2 Related Work

Developing concurrent processing systems is known to be a difficult and error-prone activity [8, 9, 10]. Existing programming models for concurrent processing differ depending on the granularity of the supported parallelism and communication [11]. Traditionally, concurrent threads synchronize and communicate through low level mechanisms, like semaphores, critical regions, send-receive primitives, etc. Interrupts are the main hardware support for communication [12]. More recent work suggests coordination models, in which interfaces define the proper cooperation and communication between threads [11]-[13]. Coordination models describe a common tuple space for placing and retrieving data (e.g., Linda [14]), adaptive parallelism by need-based activation and deactivation of threads (like Piranha [15]), manipulation of the tuple space including aggregation (i.e. Bonita [16]), constrained based communication (i.e., Law-Governed Linda [17]), moving objects from one space to another (e.g., Objective Linda [18]), and dynamic transformation of the tuple space objects (like in Gamma [19]). Compositional models, a special class of coordination models, include constructs for integrating components into bigger programs, such that the properties of the components are preserved. The integration constructs include logic clauses, like parallel AND operations in Strand [20], and higher-order functions [21, 22]. Interface-based design [23]-[26] is based on black-box models defining the system properties of the component interfaces, like arrival rate, latency, and capacity of shared resources.

The concept of Visual Programming (VP) was arguably proposed in the 80s [27], however, it is only recently that its advantages for embedded applications became apparent. VP languages have been proposed for applications like managing smart oilfields [4, 28], vehicle tracking [29, 30], contour finding [29], and environmental monitoring [2]. Region Streams [29] is a functional macro programming language for sensor networks. Specification is based on successive filtering and functional processing of data pools. The data model is based on continuous data streams sampled from the environment and groups of nodes defined by their specific interests in space and over time. Language constructs enable aggregation of the data streams from a region and application of a function to the streams in a region. Abstract Task Graphs [2] is also a functional specification in which tasks sample from and place data into pools. There is no other interaction type between tasks. Channels are filters for associating only specific data from a pool to a task. Tasks are executed periodically or when input data is available. Semantic Streams [5] implements a query-based programming paradigm that fits well applications in which sensor networks operate as large distributed databases. Queries formulated as logic programs are converted by the compiler into a service graph for the network. Data sensing is modeled as streams. Other constructs include filtering by specifying properties of the streams, defining regions and sub-regions of the physical space, and performance requirements (e.g., quality of service). Kairos [30] proposes a set of language-independent extensions for describing global behavior of sensor networks controlled centrally. The extensions assume shared memory to allow any node to iterate through its neighbors and address arbitrary nodes.

Similar to [5, 29], the data model of the proposed specification description is based on data pools and continuous data streams to the modules. However, it differs in that it focuses on optimal decision making in massively distributed environment and not on algorithmic descriptions. Therefore, we argue that the notation is orthogonal to the existing languages as it concentrates on the interactions between groups of nodes, or nodes and environment, and less on the behavior of the individual nodes.

2.3 Goal-Oriented Model

Cyber-physical Systems (CPS) are defined by a large set of interacting sub-systems. Each sub-system has well defined functionality and performance requirements, however, the interactions are dynamic, and their nature and intensity changes during execution. Figure 2.1 illustrates a simplified CPS for intelligent traffic management. The goal of the system is to optimize the traffic flow of a region by adjusting the traffic-light characteristics to the specifics of the traffic flow. The example comprises of four sub-systems, the sub-system represented by the moving cars, the two data collection sub-systems (based on video cameras and sound-based tracking), and the sub-system formed by the traffic-lights of the region. Each sub-system has a well defined functionality.

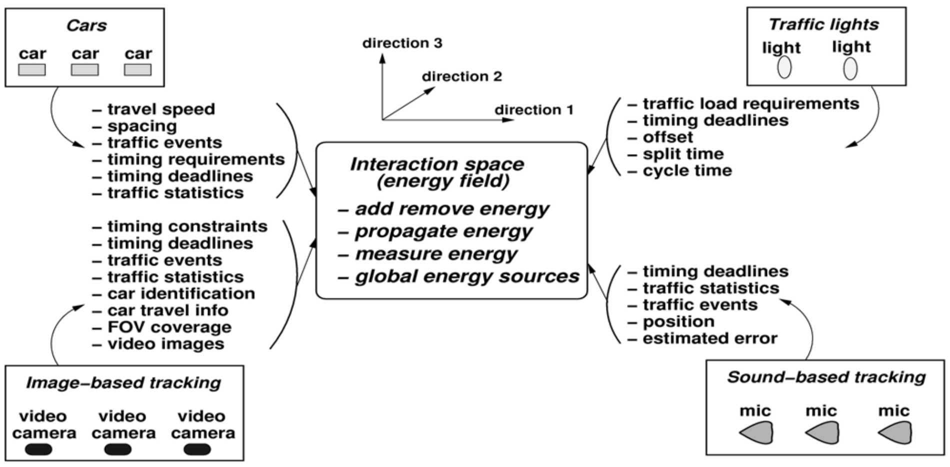


Figure 2.1: Interactions in Cyber-Physical Systems

- The image-based tracking system collects video images of traffic to get information like car position and speed, distance between cars, average number of cars passing through a region, size of car clusters, and so on. Video images have a certain precision in time and coverage: they are acquired at certain time intervals, and they might offer only a partial field-of-view covering. The sub-system can also detect events, like accidents or certain special cars, like police cars.

- The sound-based tracking system follows the moving of specific cars, such as police cars, fire trucks, and ambulances. Having a tracking alternative can be very useful if the field of view to a vehicle is lost, or if the video-based tracking is too slow to meet the necessary timing constraints of the application. The system computes the position of a tracked car with a given precision and at certain intervals of time. The system also computes traffic statistics, like average speed, and detects traffic events, such as stopped cars.

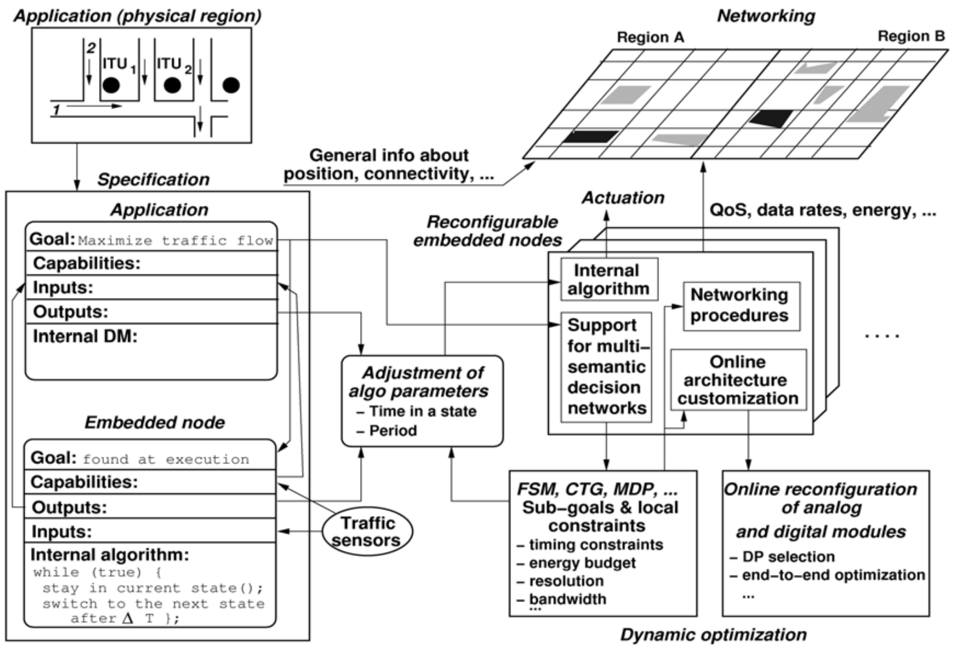
- The moving cars form a group of mobile embedded nodes. The mobile nodes can interact with each other both directly through a wireless, ad-hoc network, and indirectly by sharing the same physical space at a certain time. The node dynamics is described by attributes like speed and distance to neighboring cars.
- The traffic-lights subsystem includes all traffic-lights of a region. The traffic-light controllers are connected in a network to better coordinate their parameters offset, cycle and split times.

2.3.1 Sub-system Modeling: Functionality and Performance

Each CPS sub-system is defined as shown in Figure 2.2.

The figure illustrates the main characteristics of the proposed distributed control concept and the related specification language. The wide geographical areas from which data is sampled define pools of heterogeneous data (e.g., images, temperature, humidity, number of moving vehicles, etc.). The association of data acquisition to the physical area is defined graphically, as shown in the left figure by the larger and smaller rectangles. For each defined area, the user specifies the goals that must be achieved by the distributed application, such as maximizing the traffic flow through the area, or keeping the pollution level below a preset limit.

The execution platform is a massively distributed network of embedded controllers. Each controller node comprises hardware for sensing and actuation as well as processing,



Application (physical region)
ITU 1
ITU 2
Networking
Region A
Region B
General info about position, connectivity, ...
Specification
Application
Goal: Maximize traffic flow
Capabilities:
Inputs:
Outputs:
Internal DM:
Embedded node
Goal: found at execution
Capabilities:
Outputs:
Inputs:
Internal algorithm:
while (true) { stay in current state(); switch to the next state after Δ T };
Traffic sensors
Adjustment of algo parameters
– Time in a state
– Period
Actuation
QoS, data rates, energy, ...
Reconfigurable embedded nodes
Internal algorithm
Networking procedures
Support for multi-semantic decision networks
Online architecture customization
FSM, CTG, MDP, ...
Sub-goals & local constraints
– timing constraints
– energy budget
– resolution
– bandwidth
Online reconfiguration of analog and digital modules
– DP selection
– end-to-end optimization
Dynamic optimization

Figure 2.2: Sub-system description for distributed decision making and control

storage, and communication. In addition, hardware is reconfigurable, so that the available pool of resources at a node can be configured to meet different performance trade-offs, like variable processing speed, energy consumption, resolution, storage space, and so on. Hence, the model assumes that the functionality (algorithm) of each node is well defined and fixed but the nodes performance is parametric and dynamically changing at execution.

The following three aspects are the core of the proposed model:

- *Separation of algorithmic aspects from goals*: Algorithms describe the ontology of the application, and, over time, remain the same for all nodes in the network. In contrast, goals define optimization criteria, performance, safety, etc. They depend on the specific execution platform and conditions, and change dynamically in time. While building algorithms is arguably more efficiently done by humans, finding the parameters for optimal execution is cumbersome but can be automated.

- *Description of interdependent, heterogeneous sub-systems*: Global goals in large applications are likely to transcend different sub-system types. Also, the significance of the various related components can change over time. This invalidates static interaction schemes between sub-systems.

- *Avoiding explicit descriptions of synchronization and data transfers*: Explicit specification of internode communication reduces scalability and reusability. It is hard and unreliable to attempt capturing all possible interactions between the components of massively large scale applications. Instead, the design environment ought to identify the best interaction schemes between components, so that the overall goals as well as the goals of the modules are met.

The proposed goal-oriented model comprises of separate Decision Modules (DMs) that operate to optimize a well-defined set of goals while the overall goals of the application are also being optimized. Each module executes a set of parameterized behaviors (algorithms) for which the parameters are automatically computed based on the information provided through the goal-oriented descriptions. DMs interact with each other only in small numbers but can perform global decisions by using data aggregated from large regions and by strategic decision methods (e.g., based on Markov Decision Processes). Figure 2.3 illustrates a network, which comprises different decision making models (Finite State Machines, and Markov Decision Processes).

Similar to other specification languages for sensor networks, the proposed data model is based on data pools associated to regions and groups. Modules sample inputs from and generate outputs to a data pool for region. Regions represent continuous collections of tokens, such as a geographical area. Groups are discrete collections of tokens. Regions and groups can be associated to a specific physical area of the environment, or can be described by their defining properties.

Efficient and robust decision making must be guaranteed for hard-to-predict conditions. The overall behavior should be capable of autonomously meeting performance requirements, e.g., real-time constraints, bandwidth limitations, energy constraints, speed requirements, and precision. The underlying decision making model (DMM) is based on a multi-semantic decision making networks that represents the overall application performance at different levels of abstraction and using different evaluation formalisms. The low levels employ reactive models, e.g., Finite State Machines, which are capable of tackling unexpected situations. The upper levels use less flexible models but with more predictable performance, such as Data Flow Graphs and Task Graphs. Thus, the semantic hierarchy offers a smooth transition from the fully reactive behavior at the embedded node level and the deterministic operation at the application level. The number of abstraction levels and the decision making models for each level depend on the application.

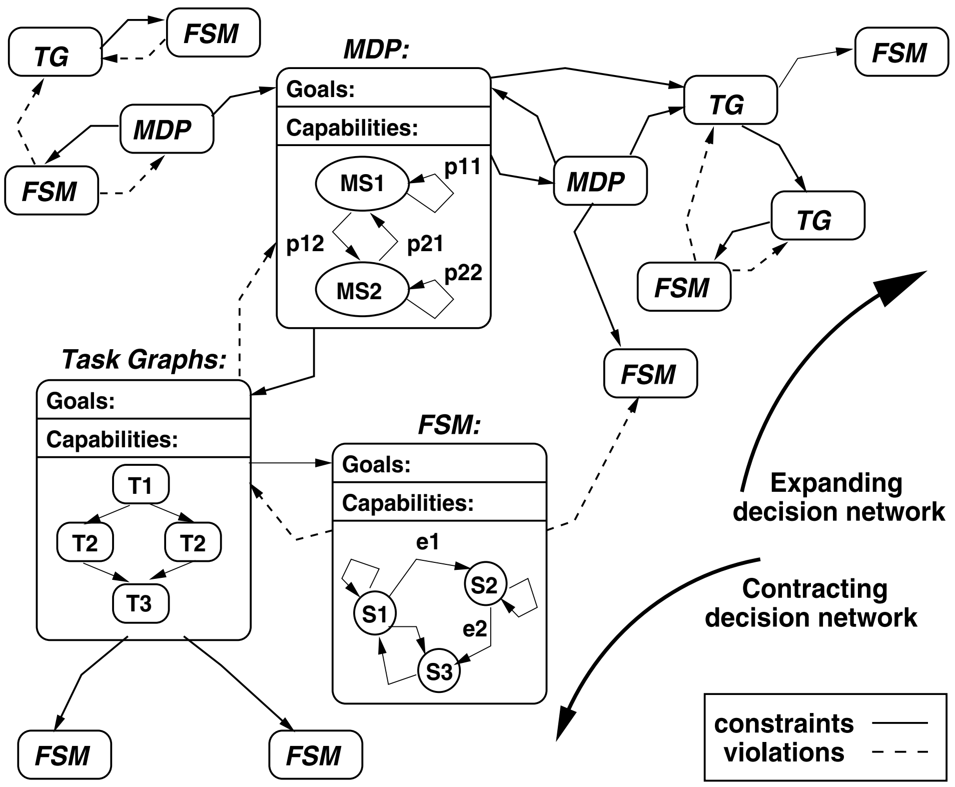


Figure 2.3: Specification and distributed control concept

The interaction between the different semantic models is achieved through top-down and bottom-up constraint transformation along the entire multi-semantic network: the top-down mechanism constrains the lower decision making levels by bounds introduced for their goals. As long as the low level operation stays within the bound it can be guaranteed that the overall performance is satisfied. The bottom-up mechanism signals when constraints (imposed by the upper levels) cannot be satisfied by the lower levels. Figure 2.3 shows a semantic hierarchy with three models, the bottom layers represent reactive behavior, and the top layers offer a performance predictive description of the system as Task Graphs (TGs) with data dependencies and Markov Decision Processes (MDPs).

The scheduling results of TGs are used to compute timing constraints for the reactive behavior of the bottom level DMs. As long as their reactive operation stays within these constraints, the overall timing requirement can be guaranteed. Bottom-up constraints express the amount of performance violation occurred at the physical level, and ought to be considered when re-computing the decision at the upper levels. The propagation of top-down and bottom-up constraints is performed continuously at runtime.

The networked decision making in Figure 2.3 operates as follows. The reactive DMs employ input sampling and event driven decision making mechanisms, such as Finite State Machines (FSMs). The reactive behavior switches from one task to another depending on the occurrence of events. For example, the system switches from S_1 to S_2, if event e_1 occurs. Each embedded unit executes its own controller. Hardware reconfiguration offers the capability of selecting online task implementations with different parameters, like speed, energy, memory, and communication bandwidth. Several individual controllers might decide to collaborate by building collectively a shared description that defines how the individual controllers use jointly any shared resources to achieve common objectives, such as the access over time (schedule) of vehicles accessing intersections. For example, the shared description can be a Conditional Task Graph (CTG) [31], which includes decisions specific to different operation conditions, such as various traffic loads. Note that CTG decision making is at the level of small local areas rather than individual controllers. Next, the CTG descriptions of the correlated areas are used to build description covering broader areas, such as those at the Markov Decision Process level. This step distributes the overall goals into goals for the individual sub-systems using information from the Markov Decision Process level.

The execution environment determines the structure of the distributed multi-semantic decision network as well as the nodes performance parameters. The network expands and shrinks depending on the nature of the application.

2.3.2 Emergent Interactions

The sub-systems interact in complex ways. Interactions include not only exchanging data between the sub-systems, like in traditional concurrent tasks, but also modifying the goals, constraints, and other parameters of the participating sub-systems. For example, the travel speed of cars sets the timing constraint of the image-based tracking sub-system, so that certain field of view coverage is guaranteed. The traffic characteristics (e.g., average speed and average number of passing cars) determine the timing parameters of the traffic-lights, like cycle time, offset and split time. The timing deadlines for "special" cars, like police cars and fire trucks, act as hard timing constraints for both the image-based and sound-based tracking sub-systems.

Many interactions emerge for specific conditions. For example a high traffic load triggers the need to optimize the timing parameters of the traffic-lights, however, if the traffic load is very low then the default, periodic timing is sufficient, and hence there is no need of establishing interactions between the traffic-light and image-based tracking sub-systems. Also, the sound-based tracking sub-system needs to interact with the other sub-systems only in special conditions, such as if there is no direct sight of view or if the timing requirements are tight. In general, two sub-systems interact with each other whenever the activities of one have a significant impact on the behavior of the other sub-system. The significance of the impact is dynamic as it changes in time and space. Moreover, the identity of the interacting sub-systems is hard to predict a-priori due to the large number of potential interactions and the likelihood of adding new sub-systems to an existing CPS application. Incremental changes to the CPS functionality should be performed without recompiling the entire application, and without stopping the execution of the existing sub-systems. Adopting a conservative approach in implementing emergent interactions can lead to excessive communication overhead due to excessive synchronization and data exchange between sub-systems.

We suggest the concept of *interaction space* as the main mechanism for defining and implementing emergent interactions between sub-systems. Interaction spaces are inspired from energy fields: interaction spaces define procedures for (i) adding and removing energy from a field, (ii) propagating energy, and (iii) measuring the energy without changing the status of the space. In addition, global energy sources define overall interaction needs for the sub-systems. The routines of each CPS subsystem have defined a footprint in terms of their impact on the interaction space. The footprint can change dynamically. Emergent interactions between sub-systems are identified by following the footprint of the sub-systems, such as a sub-system that generates significant amounts of energy interacts with another sub-system that is sensitive to the energy. Once an interaction is identified, dedicated communication channels are set-up between the sub-systems to provide the needed synchronization and data exchange with a reduced communication overhead. Figure 2.1 illustrates the concepts. The space has multiple orthogonal directions, which may or may not correspond to the Cartesian directions. The directionality of interactions results from the way energy is added (input) and removed (output) from the space. The influence of each action ripples through the space along each direction depending on a propagation resistance along that direction. The influences propagate more if the resistance is lower. The propagation resistance can change dynamically depending on the impact of the sub-system that adds or removes energy. Also, the influence can be eliminated in case a sub-system must have only a local influence.

A possible representation of an interaction space can be based on electrical fields. For our example, the cars entering the region can be interpreted as being analogous to electrical charge, thus adding energy to the electrical field corresponding to the interaction space. The cars leaving the region represent removing of energy from the field. The two tracking sub-systems measure the energy of the interaction space. Each route of the region defines a direction of the space. The traffic characteristics along a route, e.g., the average speed of cars, define the propagation resistance along that direction. The propagation resistance changes dynamically for each car or group of cars depending on the moving characteristics of the cars. Each traffic-light behaves as a switch that controls the moving of the charge particles along the existing routes, hence control the propagation resistance along directions. This representation can be used to identify emergent interactions between separate sub-systems and also between the components of the same sub-system. For example, if the traffic flow increases significantly along a certain route then the related energy increase affects a certain area of the region. The traffic-lights in that area must be coordinated with each other as they control the propagation resistance. Another example refers to the emerging interactions between video-based and sound-based tracking system. The coverage dropping of the video-based sub-system is described as a steep increase in the propagation resistance along the route. The resistance increase can be compensated through a similar decrease if the sound-based tracking sub-system can provide the necessary tracking. Hence, an emerging interaction is established between the two sub-systems. Note that the interaction space representation defines only a qualitative view of a CPS application, hence it is not used for quantitative reasoning, such as for detailed decision making and control at the level of the individual sub-systems.

2.3.3 Main Specification Constructs

The main constructs of the proposed goal-oriented specification notation are illustrated in Figure 2.4.

The basic specification entity is a Decision Module (DM), which corresponds to local points, physical areas, zones, and larger regions. As explained above, each DM executes certain application specific algorithms for sensing, processing, and actuation. This part is internal to each DM and is specified using an existing programming language, like C++ or Java. The focus of the proposed specification notation is on describing the local and

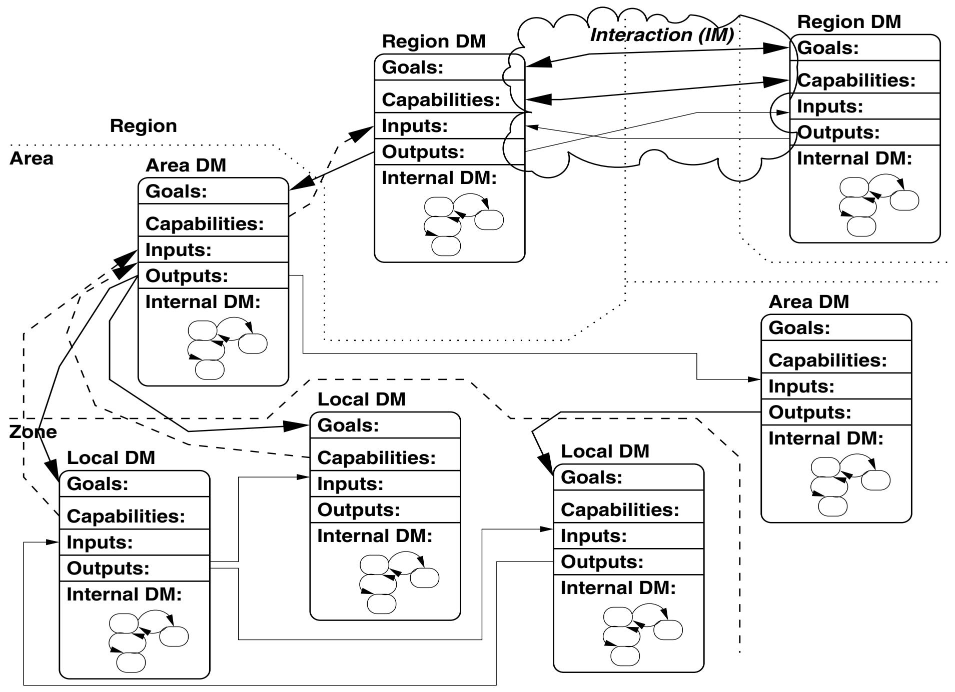


Figure 2.4: Specification for scalable decision making

global goals and the nature of interactions between modules. The compiler automatically produces the actual interaction mechanism, communication mechanism and parameters. For simplicity, the presentation is descriptive instead of a formal one. Figure 2.4 illustrates that a DM has four main parts: inputs, outputs, capabilities, and goals:

- *Inputs*: DM inputs either sample the pool of data associated to a physical region or are interaction data coming from other DMs. The acquisition semantics of an input can be continuous time, discrete time, or event driven. Moreover, inputs can refer only to certain facets of physical signals, e.g., voltage, current, phase, and frequency of a signal. The user can also specify aggregation of signals over time and space (integrals for continuous inputs and sums for discrete signals) and rate of change (sensitivities) with respect to time, space, or other signals (derivates for continuous inputs and differences for discrete inputs).

- *Outputs*: DM outputs relate to the outputs produced by the algorithm of a module. Outputs include physical actuation and control signals.

- *Capabilities*: Each module has a limited set of physical capabilities, such as the highest amount of service requests it can service, local memory, processing speed, energy resources, communication distance and bandwidth, and so on. These capabilities affect the ability of a node to maximize its local goals, and also percolate in influencing the quality of the optimal value that is reached for the overall application. For example, the low processing speed of a node or lack of memory might restrict the amount of alternatives that are locally analyzed, and affect the quality of the optimization process conducted by the node. The user can define attributes, like the upper and lower bounds and average value in time and space of a capability.

- *Goals*: The part indicates the goals to be optimized by each DM. Goals can be expressed either as maximizing or minimizing a cost function, or as a constraint satisfaction requirement, in which the goal expression must either exceed or fall below a threshold value. The cost function is defined over outputs and capabilities.

Components interact through interactions. The interactions are identified dynamically based on the interaction space concept discussed in Subsection 2.3.2. The following kinds of interactions can be established between DMs in the proposed notation:

- *Collaborative interactions*: Collaborative interactions are set up between DMs, which have nonconflicting (non-competing) goals. Modules interact with each other through inputs and outputs, e.g., one module produces outputs to the pools that serve as inputs to another module. The goals and capabilities of the modules are not affected by these kinds of interactions.

- *Competing interactions*: Such interactions are between DMs with competing goals, like optimizing the goal of one DM affects adversely the optimization of the other DM. Similar to collaborative interactions, modules interact through their inputs and outputs but cannot change their goals or capabilities.

- *Guiding interactions*: Guided interactions are between DMs at consecutively higher levels in the semantic hierarchy. DMs at upper levels generate outputs that are used to set the goals of the modules at lower levels. This way the first modules are steering the goals and hence the behavior of the latter modules.

- *Enabling interactions*: Enabling interactions are information transfers from lower (reactive) semantic levels to the upper levels. A lower DM transmits information about its capabilities to an upper DM, so that the latter can use this knowledge during a decision making that might affect the goals set for the lower DM through guided interactions. Enabling interactions are the information links through which upper DMs acquire knowledge about the actions that are ongoing in the real world.

In the proposed specification notation, the nature of interactions is dynamic and does not explicitly state the DMs participating to it. Instead, the user describes the conditions (thresholds) under which DMs start to interact with each other, like the change of an input, exceeding a threshold value, exceeding certain capabilities, etc. Interactions are described using Interaction Modules (IMs). IMs define any data transformation (such as data aggregation, and filtering) that is needed if DMs of different formalisms are interacting. Similar to DMs, IMs have goals and capabilities. Moreover, the user must specify for each of the four interaction types, the formal mechanism (TGs, MDPs, etc.) used to resolve the interaction.

2.4 Conclusions

Massively distributed embedded systems are rapidly emerging as a breakthrough concept for many modern applications. However, providing efficient and scalable decision making capabilities to such systems is currently a significant challenge. The thesis proposes a model and specification language to allow automated synthesis of distributed controllers, which implement and interact through models of different semantics. Scalability of descriptions is realized through defining the nature of interactions that can occur among decision modules while leaving the task of optimally implementing these interactions to the execution environment. The notation defines the operation goals of each sub-system (e.g., the criteria to be maximized or minimized during operation) and the physical capabilities of a module to achieve a certain goal. Different interaction types are introduced depending on the way subsystems influence each others goals and capabilities.

Compared to similar work, the proposed model and specification notation differs in that they focus on optimal goal satisfaction in massively distributed systems and not on algorithmic descriptions. Therefore, the language concepts are orthogonal to existing notations as it concentrates on the interactions between groups of nodes, or nodes and environment and less on the behavior of the individual nodes. This is expected to simplify specification and help scalable decision making.

Chapter 3

Trajectory Computation and Planning for Distributed Collaborating Mobile Agents

3.1 Introduction

Autonomous mobile agents perform a large set of activities, including the computation of trajectories and identification of the control parameters for traveling along a trajectory, signal sampling and processing (including image processing, data integration), as well as communication with other autonomous mobile agents. In many practical scenarios activities such as target detection and handling, and assessment of the results are also performed online by autonomous mobile agents. [32, 33, 34, 35]

Scalability is one of the main challenges in distributed control theory of autonomous mobile agents, which will be addressed future in this chapter. Present control techniques are restricted to single or small numbers of autonomous mobile agents. All through there are some very interesting ongoing work in swarm control [36, 37, 38, 39, 40, 41], the theory of scalable and robust distributed control is still largely unstudied.

This chapter presents our modeling procedure expressing mobile agent trajectory generation. The modeling procedure is part of developing distributed control algorithms for large-scale groups of autonomous mobile agents, as it captures the moving of autonomous mobile agents through the physical space. The modeling procedure considers the physical characteristics of mobile agents, e.g., dimensions, weight, and velocity constraints. Multi-mode behaviors of autonomous mobile agents are adopted to adapt to the unknown or changing environmental conditions and goals. In addition, the mobile agent mobility is modeled using the trajectory generation algorithm by Yakimenko [42].

Another main challenge related to mobile agent control is devising flexible strategies with predictable performance in hard-to-predict conditions. This thesis proposes an approach to performance predictive collaborative control of autonomous mobile agents operating in environments with fixed targets. The chapter also offers detailed experimental insight on the quality, scalability and computational complexity of the proposed method.

In many instances, a group of autonomous mobile agents must operate collaboratively to tackle complex tasks [43, 44, 42, 45, 46]. The nature of collaboration is often decided dynamically at run time, depending on the context-specific situations. For example, a mobile agent might not be unable to meet the deadlines set for its tasks due to unforeseen overheads, such as the time required to avoid moving obstacles. In this case, the mobile agent might inquire collaboration from neighboring mobile agents to perform jointly the tasks. Autonomous mobile agents with more flexible deadlines might decide to satisfy the request, and participate to collaboration.

An important challenge is to offer predictable and reliable operation in hard-to-predict environments and situations. Traditionally, reactive control has been the commonly accepted solution for situations that cannot be characterized off-line. Depending on conditions identified online, the controller selects the most suitable response from a set of predefined strategies. Each response strategy is characterized by specific outcomes and performance, such as execution time, and energy consumption. While certain "fixed-point" properties can be proven for reactive behavior (like stability and reachability) [45, 51, 52], other properties, which depend on dynamic attributes (e.g., the frequency of being in a state), are harder to prove unless restrictive functioning conditions are assumed. Thus, important performance attributes, i.e. execution time and resource (energy) consumption, are hard to correctly estimate and guarantee for reactive control procedures. The alternative to reactive procedures are off-line static control methods [44, 53]. These methods work very well if the operation conditions and the environment are fully known, and hence the autonomous mobile agent behavior is deterministic. The performance of the control methods can be precisely estimated in this case. However, static methods have little or no flexibility in adapting to unknown situations. In conclusion, it is challenging to devise general, performance-predictable control strategies for efficient operation in dynamic conditions.

The chapter presents the distributed control model and an approach to devising performance predictive methods for collaborative control of autonomous mobile agents operating in environments with fixed targets. Section two summarizes the related work. Section 3.3.1 discusses the trajectory generation algorithm and experimental results. Section 3.4.1 defines the addressed collaborative problem, presents the modeling solution, ILP modeling of the problem and experimental results. Finally, conclusions are put forth.

3.2 Related Work

This section summarizes existing work on swarm control, collaborative decision making and control for astomous systems.

Reynolds [40] proposes an approach based on bird-flock simulation. The model called ‘boid model’, controls the behavior of individuals in a group that is part of three-dimensional (3D) animation. The flock is modeled as particle systems [65] with a large number of particles having their own behaviors and geometrical states(e.g. orientation). The particles perform low complexity behaviors to move towards a non-collision aggregation. Distributed control is applied to control over each actor's speed and orientation to avoid collision avoidance, match velocity and center a flock. Distance and direction offset the vectors used in obstacle avoidance strategies, such as force field and steer-to-avoid. Force field model assuming repulsion force are emitted from obstacles and magnitudes of the repulsion force is inverse proportional to the distance between obstacles and individuals. Steer-to-avoid model observes obstacles only in front of the flying agent and computes a radial vector to direct the agent flying beyond the edge to avoid collision. Local center (center of neighboring boids) is used in flock aggregation action instead of some traditional approaches that use global center (center of the whole flock). Dynamic velocity matching is based on relative speeds between neighboring flockmates and not the absolute positions. In addition, group behavior, such as direct control the flock over time, is conducted using a central technique called scripting. Simulation results of this model shows 'flock-like' motion, but it is still hard to measure the validity of the motions.

Brogan and Hodgins [36] describe a method to control group behaviors for systems with significant dynamics. Significant dynamics means the motion of individual member is affected by the dynamic properties which can not be described by mass-point. Restrictions on gradients of velocity, acceleration and direction are applied since individuals are not treated as mass-point, as in other methods. Individuals in moving groups must preserve their relative positions and speed, and cluster into groups. Global information (such as obstacle positions, group velocity) is adopted in collaboration while local communication with nearby neighbors decides the reactive behaviors. This method has the limitation in collision avoidance caused by members moving towards averaging destination locations.

Gazi and Passino [63, 64] give a mathematical proof of cohesion stability of swarm aggregation with artificial potentials. Members of swarms are treated as points regardless of their dimensions. The author develop continuous time models for swarm formation and use the models to find the resulting swarm sizes. The artificial social potential function sums all the attractions and repulsions from all the other members to perform long-range-attraction and short-range-repulsion. The results apply to problems in 1D as well as in high-dimensions. Distributed aspect of swarms is also shown in the proving procedure as each member in the swarm is performing distributed optimization (computing its part of the gradient of the global function at its position). The experimental results show an increasing density of the swarm along with a growing number of members since the space occupation of the swarm is independent to the number of members, which is inconsistent with biology studied examples. Other limitations include: the position of all members in the crew needs to be known to every units, which causes a high volume of communication compared to strategies using nearest neighbor rules. Also, collision avoidance and sense range are not discussed yet in the work.

Brock and Khatib [37] suggest a similar two phase, two-dimensional planning method for real-time obstacle avoidance and motion control of multiple robots. The planning phase generates the basic trajectories, which are modeled as elastic strips. The execution phase incrementally adjusts velocity and tunes trajectory to react to the changing environment and unexpected obstacles. The novel elastic strips framework is contrived to describe the trajectories of robots, which is similar to elastic bands [56] but with more multi-dimensional freedom. Collision avoidance is implemented by means of changing the time parameters associated with the elastic strips. Elastic strips are modified continuously as assuming obstacles have a repulsive force to the strips. The elastic strips deform when the robot is trying to avoid obstacles, and regain after that to continue with their predefined tasks. The proposed method assumes that only a small number of collaborating robots operate in the shared workspace, and hence adopt centralized computation. So, the method is not applicable when the number of group members grows largely.

Yamaguchi [46] presents a distributed motion coordination strategy (called distributed smooth time-varying feedback control) for multiple mobile robots. The inter-robots strategy includes attractions and repulsions between robots and target to urge troop formation in hunting operation. Repulsions between robots, obstacles and invaders provide a collision avoidance mechanism. 2D control inputs to robots are vectors, namely ‘formation vector’, which pull the robots to the direction of the vector with a strength proportional to the norm of the vectors. Formation vectors are built heuristically by mapping surrounding environment of the robot during collaboration. Relative position of other robots and target has linear effect on input control vector for a certain robot. The advantage of this method is a predictable stability and formation controllability. Although theoretically this method can be applied to groups with large number of robots, the computation effort still limits the application of the method.

Leonard and Fiorelli [38] propose a distributed control approach to support cooperation of multiple autonomous vehicles. This method is based on artificial potentials and virtual leaders. The artificial potential implements logarithmic laws. The vehicles are point-mass. Biological models show that self-ordered animals in nature tempt to (i) get closer to the far-away neighbors in a limited range, (ii) be apart from too-closed neighbors, and (iii) keep up with neighbors in the course of aggregation. The proposed control laws simulate this process by setting the control force as minus the gradient of the sum of all relative distances (artificial potentials), hence affecting an individual vehicle and then minimizing the potential of the vehicle. Virtual leaders are virtual moving points, which make the neighboring vehicles have a larger potential. The movement of virtual leaders bring the nearby groups along the desired directions. Since there is no real leader, the behavior of each individual are affected by all neighboring vehicles. This makes a group more robust. An asymptotic dissipation term is added to the control law to achieve better stability by matching the velocities of vehicles with the desired velocity. A Lyapunov function is defined to prove the improved stabling. There is no geometrical information of autonomous mobile agents considered in the model. This model only considers singles mode formation.

Walter, Sannier and others [41] perform Virtual Battlespace simulation for an mobile agent swarm using a Graphics Processing Unite (GPU) platform called ADAPTIV. The methodology is based on how humans are involved in swarm operation, and adopt centralized control. Only simple behavior is performed by each unit in the swarm. An insect-inspired digital pheromone field, called markers, produces autonomous mobile agent attraction to targets, and repulsion from obstacle and other dangerous areas. The control operator is a hex grid data structure with numbers representing magnitudes of different effects. For example, the pheromone flavor of a certain grid can be threats or targets. The magnitude shows the strength. Target-flavor pheromone is attractive to the autonomous mobile agent at that location while threat-flavor pheromone makes a repulsion. A clear view of the entire battlefield is necessary for the control center to gather and update information of the digital pheromone field. The units in swarms operate after receiving updated digital pheromone data. The performance of GPU exceeds that of CPU in vector based simulation. However, centralized control causes a high computational complexity and having human-in-loop strategy limits the flexibility of the method.

Another potential-field-based coordination methods for mobile agents is proposed by Mamei, Zambonelli and Leonardi [39]. The method is named "Co-Field", which is a coordination field similar to force field but expresses context-related information. This field is generated by agents themselves, and conveys through wireless communication infrastructure. Mobile agents take movements following the gradient downhill, uphill, or equipotential lines of the computational field without centralized control. Similar to Van Dyke Parunak [61]'s pheromone field, "co-field" is updated by agents to keep the dynamic environment similar to reality. The method was used to simulate urban Traffic Management scenario with street/corner fields, traffic fields to coordinate agents, and conduct traffic load balancing control. The communication infrastructure implemented by middleware may be centralized (through an external server) or decentralized (embedded in mobile agents). A methodology for coordinated operation is still under research and it is a common challenge in most field orientation control strategies. In addition, updating "Co-field" requires a full awareness of all the information in the whole field, which limits the scalability of this method.

Schouwenaars, Valenti and Feron [53] propose a real-time, mixed-integer linear programming (MILP) optimizer to control manned and unmanned aerial vehicles for team missions. A team of human-operated vehicles and autonomous mobile agents operate in a partially known environment. A task scheduler generates way points as the input for the MILP module. The continuous-time states of an autonomous mobile agent are expressed as linear differential equations with position and velocity. A cost function with both stage cost and terminal cost (e.g. time consumption, travel distance) is optimized under bounded velocity and acceleration constraints. Obstacle avoidance and loiter state flying mode are also introduced by constraints of the linear program problem. The whole decision loop is composed of preprocessing, optimization, and postprocessing. The first step selects a proper cost function, detects the neighboring obstacles, and determines intermediate way points. The MILP optimizes the routing result by solving the problem formulated by preprocessing step. Because of disturbances spaces (e.g. wind), the initial velocity might be out of the safety range. Thus the postprocessing checks the feasibility of the optimized results and updates the current trajectories. This mechanism also uses back-up trajectories to enhance the robustness of the trajectory planning method when postprocessing finds infeasible trajectories. A natural language interface allows the manned and unmanned vehicle to communicate with each other. Real-time simulation-in-the-loop space (SIL) test in loiter pattern, avoiding pop-up targets, and conducting two consecutive mission are performed successfully. This method is capable of executing multiple mode mission and has human control interface. The limitation is that it is only applied to one autonomous mobile agent and one manned air craft cooperation, hence is not scalable to large teams.

Subramanian and Cruz [62] use an adaptive Markov chain to target randomly pop-up threats, and express collaboration to reach the targets. Pop-up threads are up at random locations with random observable durations. The model uses Markov chain to predict possible locations for pop-up target in the future, and is only interested in the probability of appearance of next targets. The Markov chain is expressed using event index with each node in the chain represent an event. Event index needs to be converted into time to achieve autonomous mobile agent control. A collaborating movement strategy assigns a target to the autonomous mobile agent, so that it minimizes the number of the steps for reaching the target thus reducing the time cost dramatically. Integer Programming (IP) is utilized to optimize the assignment. In this Markov chain prediction method, tables are updated either when an autonomous mobile agent reaches the target, or the target disappears. So, it is not limited in the assumption that autonomous mobile agents always reach the target location before it disappears as assumed in [50]. Only locations of a pre-generated set are possible pop-up locations. In reality, the locations are hard to predict, and are not likely to be covered by a discrete set. The Markov table dose not have enough information for prediction before the first several pop-up threads appear/disappear.

Rathbun and Capozzi [58, 57] suggest an evolutionary approach to path planning in uncertain environments. The cost function in this method is defined as the probability of autonomous mobile agents colliding with an obstacle. The Evolutionary Algorithm (EA) has the ability to perform nonlinear optimization with complex constrains, which is desirable since autonomous mobile agent states and environmental conditions are not necessarily linear functions. EA also has the feature of minimizing the changes from the original trajectories when an autonomous mobile agent is replanned dynamically. The uncertainty of the expected obstacle motions gives this problem stochastic properties. The mutation step of the EA strategy is as follows: (i) mutate and propagate, (ii) crossover, (iii) go to goal, and (iv) mutate and match. The selection process uses round-robin tournament to avoid local minima. The cost function minimizes collision possibilities with both mobile and stationary obstacles, and minimizes fuel consumption. The practicability of a certain segment of the planned path is inverse proportioned with the distance between the starting point and that path. Re-planned paths must not be dramatically different from the current trajectory, especially in nearby segments, due to the limited response time of autonomous mobile agents. The accuracy of the predicted probabilities is increasing as more information is gathered during operation because of the property of EA. There is no collaborating behavior discussed in this approach.

Stentz [59] presents a graph-search-based algorithm to calculate the optimal path in unknown environments. The technique is called 'D*' method. The robot is operating with an on-board sensor, which can detect obstacles, and map the data on to a global map of the work area. The map is not necessarily accurate and comprehensive, and is updated by robots during traveling. D* adopts states and directional arcs with associated cost to represent the work space. It uses backpointers (point form one state to the next state) to express the planned paths. The method utilizes two functions: PROCESS-STATE (generate path), and MODIFY-COST (update cost of arcs and list of states). Two functions are iteratively called until robots reach the destination. Although more efficient than the traditional brute-force optimal re-planning method [60] and overcoming some limitation by assuming fully known environments, the method is too computational intensive for large work spaces and multi-autonomous-mobile-agent teams. On the other hand, dramatic changes of trajectory after path replanning is hard to be carried out for high speed mobile devices.

Next table summarizes the taxonomy of related work in terms of properties of targets, environments, number of autonomous mobile agents and the goal of the purposed models.

The following is a table summarizes the above method as well as some other work in four different aspects: decision model, state information, time invariance and inspiration. Decision model shows the proposed method is decentralized or centralized, reactive, hierarchical or not, and so on. State information is the location of physical storage of the states of autonomous mobile agents, and data representation. Time invariance examines the variability of method over time and number of models contains in the method. At last, inspiration is the origination of the method.

Table 3.1: Problem modeling

Method	Targets	Environments	Autonomous Mobile Agents	Goal
Reynolds [40]	fixed	mobile obstacles	• group • geometrical model • aware of other member's speed and position	group formation
Brogan et al. [36]	fixed	mobile obstacles	• group • geometrical and mass-point model • aware of other member's speed and position	group formation
Tanner et al. [51]	fixed	mobile obstacles	• group • mass-point model • aware of of other member's speed and position	group formation
Gazi and Passino [63, 64]	fixed	mobile obstacles	• group • mass-point model • aware of other member's speed and position	group formation
Yeung and Bekey [47]	fixed	fixed and mobile obstacles	• group • mass-point model • sense obstacle during travelings	minimize distance
Yamaguchi [46]	fixed	fixed and mobile obstacles	• group • geometrical model • sense obstacle during travelings	group formation
Leonard and Fiorelli [38]	fixed	mobile obstacles	• group • geometrical model • aware of other member's speed and position	group formation

Table 3.2: Problem modeling–continued I

Method	Targets	Environments	Autonomous Mobile Agents	Goal
Van Dyke Parunak [61]	fixed	mobile obstacles pop-up thread	• group • geometrical model • sense pheromone in the field	group formation
Mamei et al. [39]	fixed	mobile obstacles	• group • geometrical model • follow ”force” in the field	traffic balance
Shim et al. [49]	fixed	fixed obstacles	• single • geometrical model • sense obstacles using radar	exploration
Schouwenaars et al. [53]	fixed	fixed obstacles pop-up thread	• single • geometrical model • communication through user interface	optimization(e.g. time cost, travel distance)
Subramanian and Cruz [62]	fixed	pop-up thread	• group • mass-point model • aware of historical pop-up threat	target tackling
Rathbun and Capozzi [58, 57]	fixed	mobile obstacles pop-up thread	• single • mass-point model • sense the environment	target tackling; optimization(e.g. time cost, travel distance)
Stentz [59]	fixed	fixed and mobile obstacles	• single • mass-point model • sense the environment	target tackling
Soulignac and Taillibert [48]	fixed	fixed and mobile obstacles	• single • mass-point model • sense the environment	optimization(e.g. time cost, travel distance)

Table 3.3: Problem modeling–continued II

Method	Targets	Environments	Autonomous Mobile Agents	Goal
Brock and Khatib [37]	fixed	mobile obstacles	• group • geometrical model • aware of neighbor's motion	coordination and cooperation
Walter et al. [41]	fixed	mobile obstacles	• group • geometrical model • sense pheromone in the field	group formation
Gil et al. [43]	tasks with different priorities; and pop-up targets	no obstacles	• group • mass-point model • aware of task priority information	target tackling

Table 3.4: Related work

Method	Decision Model	State Information	Time Invariance	Inspiration

Reynolds [40]	• decentralized
	• reactive
	• repulsion only
	• discrete time
	• not hierarchical
		centralized;
		graph representation;
			fixed over time;
			single model;
				biology: flocks,
				herds, schools

Brogan et al. [36]	• decentralized
	• not reactive
	• attraction and repulsion
	• continuous time
	• hierarchical
		centralized;
		graph representation;
			fixed over time;
			single model;
				biology: flocks,
				herds, schools

Tanner et al. [51]	• decentralized
	• not reactive
	• attraction and repulsion
	• continuous time
	• not hierarchical
		decentralized;
		graph representation;
			fixed over time;
			single model;
				biology;
				graph theory

Gazi and Passino [63, 64]	• decentralized
	• not reactive
	• attraction and repulsion
	• continuous time
	• not hierarchical
		decentralized;
		graph representation;
			fixed over time;
			single model;
				biology

Yeung and Bekey [47]	• decentralized
	• reactive
	• continuous time
	• hierarchical
		decentralized;
		graph representation;
			fixed over time;
			single model;
				biology

Brock and Khatib [37]	• decentralized
	• reactive
	• discrete time
	• hierarchical
		decentralized;
		field representation;
			fixed over time;
			single model;
				nature

Table 3.5: Related work–continued I

Method	Decision Model	State Information	Time Invariance	Inspiration

Yamaguchi [46]	• decentralized
	• reactive
	• discrete time
	• not hierarchical
		decentralized;
		graph representation;
			time-varying
			feedback control law;
			single model;
				other control
				methodology

Leonard and Fiorelli [38]	• decentralized
	• reactive
	• attraction and repulsion
	• continuous time
	• not hierarchical
		decentralized;
		field representation;
			fixed over time;
			single model;
				biology

Walter et al. [41]	• decentralized
	• reactive
	• attraction and repulsion
	• discrete time
	• not hierarchical
		decentralized;
		field and grid
		representation;
			fixed over time;
			single model;
				biology:
				insect, e.g. ants

Van Dyke Parunak [61]	• decentralized
	• reactive
	• attraction and repulsion
	• discrete time
	• hierarchical
		decentralized;
		field representation;
			fixed over time;
			multiple model;
				biology;
				physics

Mamei et al. [39]	• decentralized
	• reactive
	• attraction and repulsion
	• continuous time
	• not hierarchical
		decentralized;
		field representation;
			fixed over time;
			single model;
				physics

Shim et al. [49]	• centralized
	• discrete time
	• not hierarchical
	• mapped based trajectory planning
		centralized;
		graph representation;
			fixed over time;
			single model;
				physics

Table 3.6: Related work–continued II

Method	Decision Model	State Information	Time Invariance	Inspiration

Schouwenaars et al. [53]	• centralized
	• discrete time
	• not hierarchical
	• communication based trajectory planning
		centralized;
		graph representation;
			fixed over time;
			multiple model;
				physics

Subramanian and Cruz [62]	• centralized
	• discrete time
	• not hierarchical
	• prediction based trajectory planning
		centralized;
		grid representation;
			vary over time;
			single model;
				Marcov Chain

Gil et al. [43]	• centralized
	• continuous time
	• not hierarchical
	• priority based scheduling
		centralized;
		graph representation;
			vary over time;
			single model;
				nature
				physics

Rathbun and Capozzi [58, 57]	• centralized
	• continuous time
	• not hierarchical
		centralized
		graph representation;
			fixed over time;
			single model;
				nature
				biology

Stentz [59]	• centralized
	• continuous time
	• hierarchical
		centralized
		grid representation;
			fixed over time;
			single model;
				physics

Soulignac and Taillibert [48]	• centralized
	• continuous time
	• hierarchical
		centralized
		grid representation;
			fixed over time;
			single model;
				physics
				graph theory

3.3 Proposed Distributed Control Model and Trajectory Generation for Autonomous Mobile Agents

This section introduces the distributed control model that we intend to pursue. But, before defining the model, we systematized the main characteristics that must be tackled by any distributed control approach. This is important not only for placing our model in context, but also shows the differences compared to other work.

3.3.1 Proposed Distributed Control Model

Distributed control algorithms address the following main aspects:

- *Global goals*: This refers to the overall goals of distributed control. Goals can be defined globally, such as finding extreme values, cumulative attributes, global moving pattern, maximizing information acquisition, maximizing safety, etc, or locally, such as meeting an imposed distance constraint between neighbors.
- *Environment*: The environment defines the context in which vehicles operate. Environments might include static and/or moving obstacles.
- *Controlled variables*: This relates to the number and type of the controlled variables. For example, variables can include those for vehicle movement, like velocity magnitude and direction, and gradients of change, the characteristics of the data acquisition process, e.g., the rate of data sampling, and the nature of actuation. Approaches also differ from those that control single variables to those that decide a large number of variables.
- *Constraints*: Constraints constrict the state and controlled variables to express the goals of distributed control, obstacles, other moving vehicles that should not collide

with each other, communication ranges, processing capabilities of the vehicles, and inertias.

- *Local control laws*: The laws define the local behavior of the vehicles. The control laws capture the state of the neighbors, obstacles, and goals. The laws can follow two philosophies, based on local interactions and based on an artificial potential fields.

With respect to the variability over time, systems can be static (no variation over time) and dynamic (gradient over time):

- *Global goals*: Majority of the existing methods assume static objectives and goals, such as the formation of a certain pattern, collision avoidance, and navigation to a given target. Dynamic goals are useful not only for defining changing missions for the system, but also for increasing its security and better adjustment to changing environments. For example, if there are no external threats then the main goal would be that of maximizing data acquisition, which changes to maximizing safety in the presence of hazards.

- *Environment*: Most of the existing techniques consider environments with dynamic characteristics, such as variable temperature, air conditions but also moving obstacles.

- *Controlled variables*: The sets of state and control variables are fixed in arguably all existing approaches. This is explained by the fact that the mobile individuals have a simple behavior, which does not adapt significantly to the environment. However, in a multi-mode situation, the sets of state and control variables ought to be changed depending on the actual operation mode. For example, in data acquisition mode, the state and control variables relate to maximizing the collected information amount. In contrast, a different variable set is used in hunting mode.

  State and control variables might also change if individuals are differentiated depending on their priorities, like virtual leaders, and communication-oriented entities.

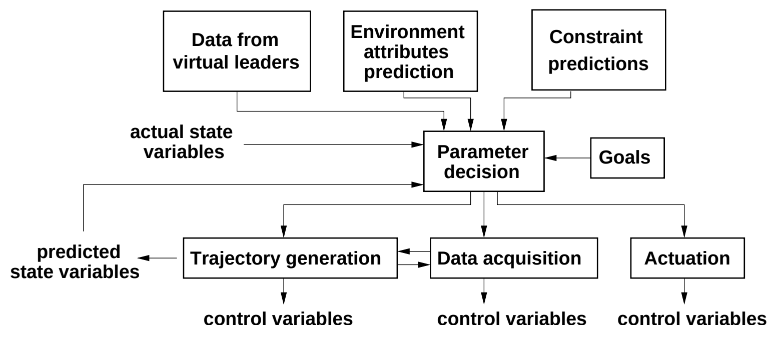


Figure 3.1: Vehicle actions

- *Local control laws*: Existing methods assume a fixed functionality for their individuals. Some approaches achieve a limited degree of adaptation to dynamic situations by adjusting functional parameters through negotiations between neighbors and interactions with the environment.

Our goal is to study distributed control methods for large-scale groups of autonomous mobile agents. The goals set for the autonomous mobile agent groups refer to both global and local attributes, and might change over time. The environmental properties might also modify in time, including the position of mobile obstacles. This determines that the constraints that ought to be satisfied by the obtained control laws have to contemplate variable constraints. Finally, each autonomous mobile agent has multi-mode behavior, including multiple data acquisition scenarios, communication procedures, actuation mechanisms, and so on. During operation, the distributed control methods must decide not only the coordinated moving of the vehicles (e.g., to avoid collision) but also the synchronization between the moving characteristics and the global goals (e.g., the amount of acquired information). As a consequence, the local control laws, and the sets of state and control variables must have a dynamic character too.

3.3.2 Trajectory Generation Algorithm

The trajectory generation algorithm and its implementation is a part of mission planning. However, with the development of embedded computing system, it's not necessary

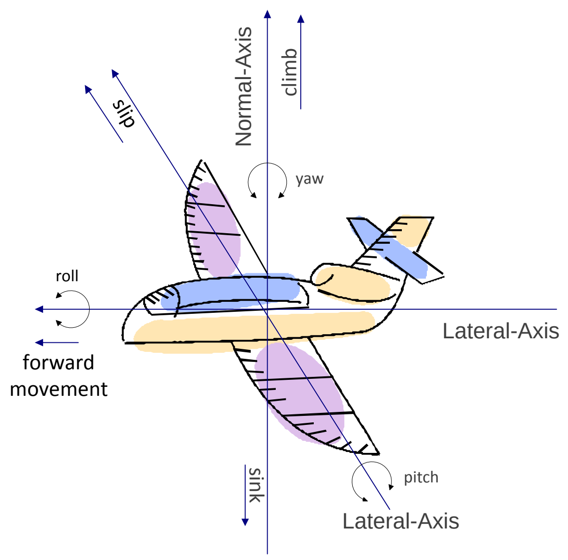


Figure 3.2: Aircraft dynamic actions

anymore to build the trajectory generating system in the ground control station. The system can be implemented now as an on-board system, as real-time trajectory generation is more advantageous has than remote control, and also aids multi-autonomous-mobile-agent cooperation. This section summarizes the trajectory generation algorithm proposed by Yakimenko [42]. The trajectory generation problem is that of producing an optimized trajectory of points $T \in \{T_1, T_2, T_3, \ldots\}$, such that each point satisfies all constraints on dynamics and control, and the predefined initial and final states (position and velocity) of the trajectory are met. Possible optimization criteria include flying time, flying distance, fuel consumption, autonomous mobile agent security, and maximizing information acquisition.

The autonomous mobile agent movements are obviously described in a 3-D coordinate system, and are controlled by rotation around its center of gravity, as shown in Figure 3.2. The autonomous mobile agent trajectory generation problem involves not only the air vehicles, but also one or more mission planning and control stations, payloads, necessary data links, and other ground support equipment, like launch and recovery equipment.

The aircraft model is as follows. Let (x_1, x_2, x_3) denote the local level coordinates over a flat region with x_1 axis pointing EAST, x_2 axis pointing NORTH, and x_3 pointing DOWN.

A autonomous mobile agent's center of mass defines its position. The variables V (airspeed), γ (flight path angle), and χ (azimuth angle) are needed to fully define the autonomous mobile agent's dynamics [42].

As explained in [42], in the above coordinates system, the aircraft is modeled as a set of three-dimensional point-mass differential equations:

$$\dot{x}_1 = V \cdot \cos\gamma \cdot \cos\chi \qquad \dot{x}_2 = V \cdot \cos\gamma \cdot \sin\chi$$

$$\dot{x}_3 = -V \cdot \sin\gamma \qquad \dot{V} = g(n_x - \sin\gamma) \tag{3.1}$$

$$\dot{\gamma} = \frac{g(n_z \cdot \cos\phi - \cos\gamma)}{V} \qquad \dot{\chi} = \frac{g \cdot n_z \cdot \sin\phi}{V \cdot \cos\gamma} \qquad \dot{m} = -C_s$$

where

$$n_x = \frac{[\overline{T}(\delta_T, \overline{n}) \cdot T_{max}(M, x_3, c) \cdot \cos(\alpha + \epsilon_T) - D(\alpha, M, x_3, c)]}{m \cdot g}$$

$$n_z = \frac{[\overline{T}(\delta_T, \overline{n}) \cdot T_{max}(M, x_3, c) \cdot \sin(\alpha + \epsilon_T) + L(\alpha, M, x_3, c)]}{m \cdot g}$$

$$\dot{\overline{n}} = \frac{k_T \cdot \delta_T - \overline{n}}{t_\delta}$$

The state vector of the autonomous mobile agent is $z = \{x_1, x_2, x_3, V, \gamma, \chi\}^T$, and the three-variable control vector is $u = \{\delta_T, n_z(\alpha), \phi\}^{T}$[1]. The states and controls of the autonomous mobile agent are fully defined by the above two vectors in its state and control space.

The trajectory generation algorithm by Yakimenko [42] uses polynomials for approximating an autonomous mobile agent's trajectory. Consider polynomials of degree n as the reference functions for the aircraft trajectory x_i, ($i = 1, 2, 3$) in the 3-D coordinate

[1]Nomenclature: δ_T is the throttle position, n_z is the normal projection of load factor, and ϕ is the bank angle

system:

$$x_i(\tau) = \sum_{k=0}^{n} a_{ik} \frac{(max(1, k-2))! \cdot \tau^k}{k!} \tag{3.2}$$

$$x_i'(\tau) = \sum_{k=1}^{n} a_{ik} \frac{(max(1, k-2))! \cdot \tau^{k-1)}}{(k-1)!} \tag{3.3}$$

$$x_i''(\tau) = \sum_{k=2}^{n} a_{ik} \cdot \tau^{k-2} \tag{3.4}$$

where τ (virtual arc) is an argument. The degree n of the polynomial is decided by the maximum time derivative of the aircraft coordinates, and the constrains at the initial and final points of the trajectory. The degree n is greater than the maximum orders of the time derivatives at the terminal points [42]. The values of the polynomial coefficients a_{ik} are determined by the boundary conditions of the trajectory. For example, the coefficients of polynomials of degree 3 are as follows [42]:

$$\begin{aligned} a_{i0} &= x_{i0} \qquad a_{i1} = x'_{i0} \\ a_{i2} &= -\frac{2x'_{if} + 4x'_{i0}}{\tau_f} + \frac{6(x_{if} - x_{i0})}{\tau_f^2} \\ a_{i3} &= \frac{6(x'_{if} + x'_{i0})}{\tau_f^2} - \frac{12(x_{if} - x_{i0})}{\tau_f^3} \end{aligned} \tag{3.5}$$

τ_f (length of the virtual arc) is the main optimization parameter. More optimization goals can be added to the problem description, such as the minimum flight time and the minimum gas consumption. This trajectory formulation has the advantage that the velocity history is independent of the trajectory. Thus, different speed histories can be achieved along a trajectory, so that the predefined final airspeed can be met.

By using virtual arc τ as an optimization parameter, the trajectory and the velocity history are made independent to each other. Thrust history is defined as the points where the thrust value changes. The thrust values belong to a known set. For example, the thrust history might include the points at which the autonomous mobile agent starts to accelerate and decelerate, respectively. τ_{T1} and τ_{T2} $(0 \leq \tau_{T1} < \tau_{T2} \leq \tau_f)$, such that the thrust switches from δ_{Tmax} to δ_{Tmin} at τ_{T1} to let the autonomous mobile agent accelerate as fast as possible, and then switches from δ_{Tmin} to δ_{Tmax} at τ_{T2} to let the autonomous mobile agent decelerate.

The algorithm for finding the thrust history positions the switching points using a binary search algorithm. The resulting thrust history is verified with respect to goals like minimum-length trajectory, minimum flying time, and fuel consumption, and meets the imposed constraints (e.g., velocity gradients, and obstacles). As shown in Eqs. (3.6), the derivative of the speed in terms of τ (virtual arc) is not affected by the value of the trajectory coordinates (x_1, x_2, x_3) [42]:

$$V'(\tau) = g(n_x - \sin\gamma)\frac{dt}{d\tau} = \frac{g(n_x - \sin\gamma)}{\lambda(\tau)} \tag{3.6}$$

where

$$\lambda(\tau) = \frac{d\tau}{dt}{}^{2} \tag{3.7}$$

As a result, τ can be optimized based on a preset thrust history, or the thrust history can be the second optimization parameter. For example, if trajectory generation is a time-optimization problem, the optimum control law is the rule for thrust on/off switchings [42]. So, after solving the shortest time optimization problem, we can set two switch points τ_{T1} and τ_{T2} $(0 \leq \tau_{T1} < \tau_{T2} \leq \tau_f)$, such that the thrust switches from δ_{Tmax} to δ_{Tmin} at τ_{T1} to let the mobile agent accelerate as fast as possible, and then switches from δ_{Tmin} to δ_{Tmax} at τ_{T2} to let the autonomous mobile agent decelerate. It is obvious that to achieve the shortest time, $\tau_{T1} = 0$ and $0 < \tau_{T2} \leq \tau_f$. If we are looking for the minimum fuel consumption solution, the second optimization parameter will be a constant δ_T^*.

The boundary constraints on the trajectory are imposed as follows. The derivative of the initial and terminal state vectors and controls are given in terms of time. However, in

[2] λ in Eqs. (3.7) is the virtual speed.

order to calculate the coefficients a_{ik} of Eqs. (3.2), it is necessary to know the initial and final coordinates $(x_{i0},\, x_{if})$ $(i = 1,\, 2,\, 3)$ of the aircraft, and their derivatives x'_{i0}, x'_{if}, x''_{i0}, x''_{if} with respect to the argument τ. Hence, we have to relate the state vector derivatives to the expressions for $\dot{x}_i$ and $\ddot{x}_i$.

The time derivatives of the mobile agent coordinates are expressed as the following equations with respect to the definition of λ:

$$\dot{x}_i(\tau) = \frac{dx_i}{d\tau} \cdot \frac{d\tau}{dt} = x'_i \cdot \lambda(\tau)$$

$$\ddot{x}_i(\tau) = \frac{d(x'_i(\tau) \cdot \lambda(\tau))}{d\tau} \cdot \frac{d\tau}{dt} = x''_i \cdot \lambda^2 + \dot{x}_i \cdot \lambda'$$

$$i = 1, 2, 3$$

Then, for x' and x'' one can get:

$$x'_i = \frac{\dot{x}_i}{\lambda} \qquad x''_i = \frac{\ddot{x}_i - \dot{x}_i \cdot \lambda'}{\lambda^2} \qquad i = 1, 2, 3$$

The corresponding value of λ_i and λ'_i are as follows:

$$\lambda_0 = V_0 \qquad \lambda'_0 = \frac{\dot{V}_0}{V_0} \qquad \lambda_f = V_f \qquad \lambda'_f = \frac{\dot{V}_f}{V_f}$$

The predicted state and control variables are computed based on the thrust history and imposed boundary constraints as follows. The numerical solutions of the predicted state and control variables along the trajectory are calculated iteratively over N consecutive points that are equally placed along the virtual arc (N is selected by the user):

$$\Delta\tau = \frac{\tau_f}{N - 1}$$

The airspeed at the jth point is determined by the airspeed at $(j-1)$-th point (V_{j-1}) together with $n_{x\,j-1}, \gamma_{j-1}$, and λ_{j-1} [42]:

$$V_j = V_{j-1} + \frac{g(n_{x\,j-1} - \sin\gamma_{j-1})\Delta\tau}{\lambda_{j-1}} \tag{3.8}$$

$\Delta\tau$ results from this equation.

The corresponding time interval can be approximated using average airspeed [42]:

$$\Delta t_j = \frac{2\sqrt{\sum_{i=1}^{3}(x_{i\,j} - x_{i\,j-1})^2}}{V_j + V_{j-1}} \tag{3.9}$$

Thus, $\lambda_j = \frac{\Delta\tau}{\Delta t_j}$ can be calculated in order to get other dynamic (state) and control parameters at each step ($j = 1, 2, \cdots, N$).

Other parameters can be computed at each point [42]:

$$\gamma = -\arcsin\Big(\frac{x_3'}{\sqrt{\sum_{i=1}^{3} x_i^2}}\Big)$$

$$\chi = \begin{cases} \arctan\big(\frac{x_2'}{x_1'}\big) & \text{if} \quad x_1' \geq 0 \\ -x_2'\Big(\pi - \mid \arctan\big(\frac{x_2'}{x_1'}\big) \mid\Big) & \text{if} \quad x_1' < 0 \end{cases}$$

χ (bank angle) and n_z are calculated iteratively along with the coordinates and states variables at each point [42]:

$$\phi = \begin{cases} \arctan\big(\frac{V\cdot\lambda\cdot\chi'\cdot\cos\gamma}{V\cdot\lambda\cdot\gamma' + g\cdot\cos\gamma}\big) & \text{if } V\cdot\lambda\cdot\gamma' \geq -g\cdot\cos\gamma \\ -(\chi'\cdot\cos\gamma)\Big(\pi - \mid \arctan\big(\frac{V\cdot\lambda\cdot\chi'\cdot\cos\gamma}{V\cdot\lambda\cdot\gamma' + g\cdot\cos\gamma}\big) \mid\Big) & \text{if } V\cdot\lambda\cdot\gamma' < -g\cdot\cos\gamma \end{cases}$$

$$n_z = \frac{1}{g}\sqrt{(V\cdot\lambda\cdot\chi'\cdot\cos\gamma)^2 + (V\cdot\lambda\cdot\gamma' + g\cdot\cos\gamma)^2}$$

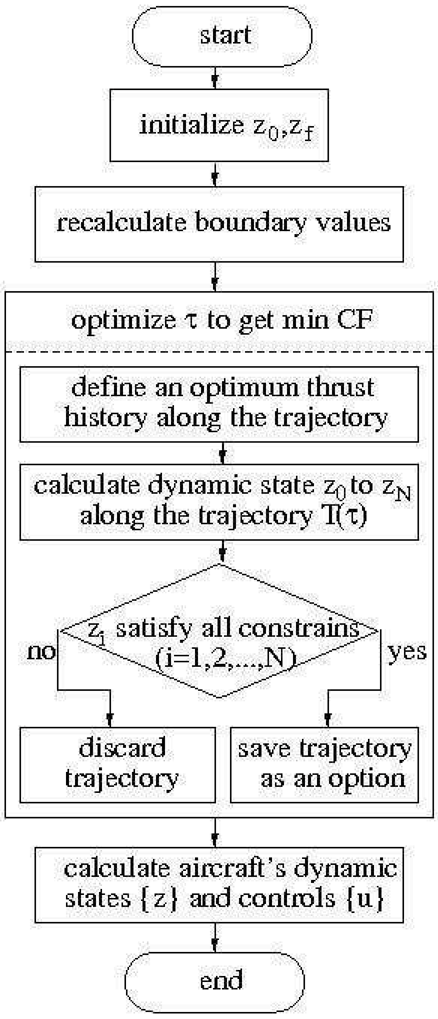

start
initialize z_0, z_f
recalculate boundary values
optimize τ to get min CF
define an optimum thrust history along the trajectory
calculate dynamic state z_0 to z_N along the trajectory T(τ)
z_i satisfy all constrains (i=1,2,...,N)
no
yes
discard trajectory
save trajectory as an option
calculate aircraft's dynamic states {z} and controls {u}
end

Figure 3.3: Flowchart of the optimum trajectory generation algorithm

where γ' and χ' can be computed from the equations [42]:

$$\gamma' = -\frac{x_3''(x_1'^2 + x_2'^2) - x_3'(x_1'x_1'' + x_2'x_2'')}{(\sum_{i=1}^{3} x_i'^2)^{\frac{3}{2}} \cos\gamma}$$

$$\chi' = \frac{(x_2''x_1' - x_2'x_1'') \cdot \cos^2\chi}{x_1'^2}$$

The trajectory generation problem can be defined as an constrained optimization problem as follows: minimize the cost function while meeting the constraints of the air dynamic characteristics of the autonomous mobile agent, and satisfying the preset initial and final states:

$$min\ F_{cf}(z_i) \quad z_i \in \{z\} \quad \text{under constraints}\, E(F_{cf}) \leq \varepsilon$$

where z is the set of admissible state vectors, and $E(F_{cf})$ is a penalty function.

There is no special restriction on the optimization method, and any zero order algorithm can be adopted. However, using proper search strategy during optimization can make the trajectory generation much more efficient. To choose a proper optimization method is very critical to improve the response time of the real time onboard trajectory generation system. The flowchart of the trajectory generation algorithm was given in Figure 3.3.

3.3.3 Experiments

The set of experiments was defined to study the following aspects of the trajectory generation algorithm: the characteristics of the algorithm, the accuracy of the solution, and the execution time of the algorithm. The algorithm was coded in Matlab, and run on a Pentium laptop. A set of several hundreds experiments were run. The distance between the initial and final points covered ranges from 500 m to 100,000 m, and the speed changed in the range 20 m/s to 200 m/s.

Figures 3.4 and 3.5 show an example of a trajectory that was computed using the

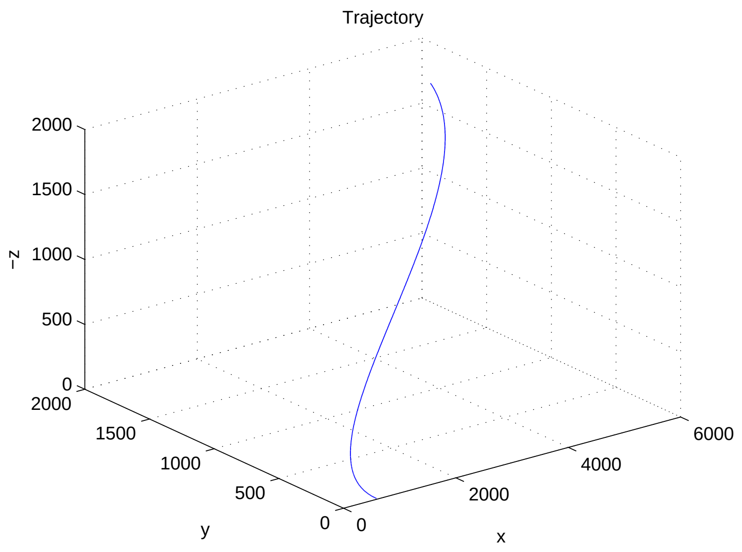


Figure 3.4: Optimum trajectory in 3-D coordinates system

algorithm with the goal of finding the shortest trajectory between initial and terminal points. The trajectory also satisfies the next constrains: initial point: $x_0 = [600\ 2\ -3]$ and $V_0 = 20$, and final point: $x_f = [5000\ 1500\ -2000]$ and $V_f = 80$, respectively.

Preset initial and final points

Since the parameters a_{ik} of the reference polynomials are determined by the boundary states of the aircraft (Eqs. (3.5)), they guarantee that the initial and final points of the trajectory are exact as we imposed. In the example, the first time derivative of the autonomous mobile agent coordinates at the initial and terminal points are set as follows: $\frac{dx_0}{dt} = [1\ 20\ 1]$, and $\frac{dx_f}{dt} = [1\ 56.56\ -56.56]$. The plot in Figure 3.7 shows the result projected on the horizontal plane. The first time derivative at the final point is -56.56, and a notch appears in the plot $\frac{d\gamma}{dt}$ to achieve the value.

Airspeed

The thrust history generated during trajectory optimization had the goal of getting the final point as soon as possible. It is obvious that the autonomous mobile agent should

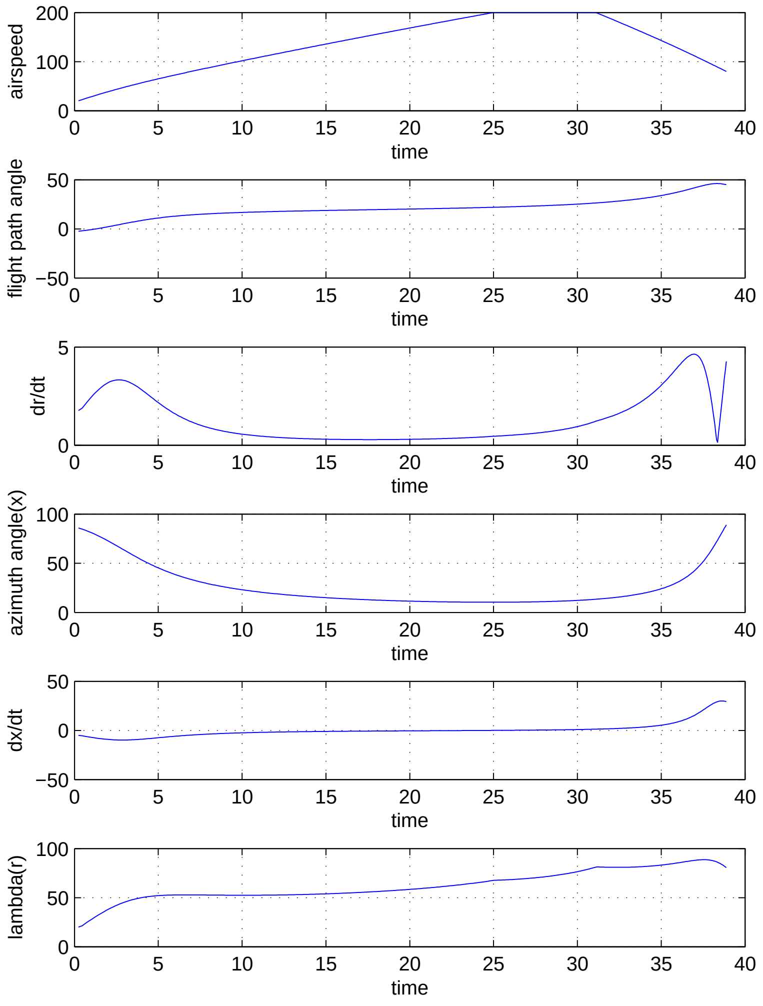


Figure 3.5: Dynamic states

Table 3.7: Properties of the trajectory (N = 500)

x_0 (m)	$\frac{dx_0}{dt}$ (m)	x_f (m)	$\frac{dx_f}{dt}$ (m)	V_0 (m/s)	V_f (m/s)	$V*_f$ (m/s)	Trajectory length (m)
(600 2 -3)*	(1 20 1)	(5000 1500 -2000)	(1 56.56 -56.56)	20	80	80.13	5235.80
(600 2 -3)**	(14.14 10 -10)	(5000 1500 -2000)	(-30 -47.95 -56.56)	20	80	80.11	5187.10
(600 2 -3)***	(3.5 68 3.5)	(5000 1500 -2000)	(2 113.12 -113.12)	70	160	159.89	5261.87

accelerate at its maximum accelerate rate, then go with it maximum air speed, and decelerate at a certain to achieve desired final speed, as shown in the first plot of Figure 3.5. However, if the goal is to save the fuel consumption, or to keep the autonomous mobile agent flying at a certain speed in order to let data acquisition work properly, then the thrust history is generated in a different way.

Flight path and azimuth angles

In the above example, the time derivative of the flight path angle and azimuth angle were set as constrains. For example, the time derivative of the azimuth angle reached the upper bound at the end of the trajectory to achieve the desired final speed, but because of the constraint, the plot of azimuth angle doesn't show a sharp rise at the end, while it satisfies the direction of airspeed ($\frac{dx_f}{dt} = [1\ 56.56\ \ -56.56]$) at the final point in the $x_1 - x_2$ plane.

Length of the trajectory

If the flight path and azimuth angles at the two terminal points are coherent with the direction pointing from the initial to final point, then the length of the optimum trajectories are very close to the straight distance between the two points. The dashed-dot plot in Figure 3.6 shows this case. Otherwise, if the initial airspeed is heading towards a very different direction from that of the destination, the desired final flight path and the azimuth angles cannot be achieved by flying straightly from the starting point, hence the trajectory goes extra distance to reach the desired states, as shown in the dashed-line plot in Figure 3.6.

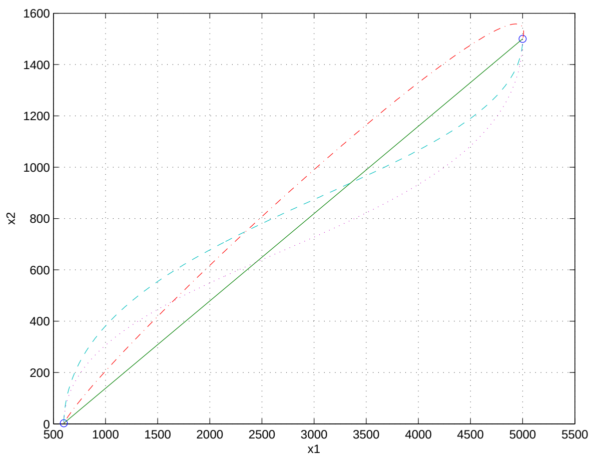


Figure 3.6: Trajectories for different initial and final states: solid line is for straight connected path; dashed line is for case * in Table 3.7; dashed-dot line is for case ** in Table 3.7; and dot line is for case *** in Table 3.7

Shape of the trajectory

The change of the azimuth angle during flight can be reflected by the the projection of the trajectory on the horizontal plane (as shown in the 4-th, and 5-th plots from the top in Figure 3.5). The change of the flight path angle is reflected by the projection in the vertical plane.

Impact of the initial and final airspeed direction

Figures 3.7 and 3.8 illustrate an example of the impact of the airspeed direction on trajectories and thrust histories. All the conditions to compute the trajectories in the example are the same excepting the direction of the initial and final airspeed direction. The solid curve is convex and then concave because the directions of the speed at the initial and final points are very close to the x_2 coordinate. Hence, the autonomous mobile agent has to go back slightly along the x_2 direction to satisfy this condition. The dash-dot curve is close to a straight line since the final condition is easy to be satisfied with a single convex curve.

Regarding the thrust histories, since the initial speed direction of the dash-dot line is very different from the trajectory heading direction at the initial point, it takes longer time to the autonomous mobile agent that flies following the dash-dot trajectory to accelerate to the maximum speed. As a result, it arrived at the terminal earlier than if it had flown along the solid trajectory. This example shows that it is not necessary that the shorter trajectory also leads to a shorter flying duration.

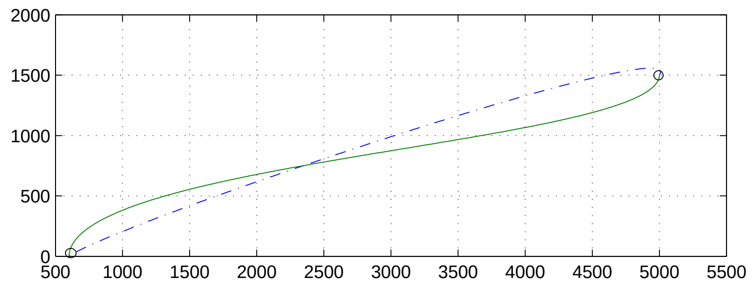


Figure 3.7: Impact of initial and final airspeed directions on the trajectory

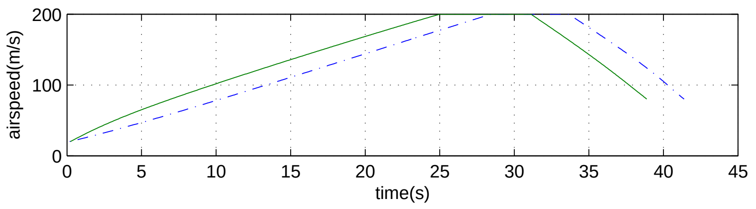


Figure 3.8: Impact of the initial and final airspeed directions on the thrust history: solid line is for $\frac{dx_0}{dt} = [1\ 20\ 1]$, $\frac{dx_f}{dt} = [1\ 56.56\ -56.56]$; and dash-dot line is for $\frac{dx_0}{dt} = [14.14\ 10\ -10]$, $\frac{dx_f}{dt} = [-30\ -47.95\ -56.56]$

Accuracy of the final airspeed

For $N \geq 1000$, in 94% of the cases, the relative error of the final speed was below 1.50%. Obviously, the longer the trajectory, the larger interval there is between two adjacent points, which decreases the accuracy of simulation results for long distances. However, the error can be reduced by increasing N, as shown in Figure 3.9. The number of final points with accuracy above 2.5% increases almost twice when N increases from 500 to 5,000.

There is an exception to this rule when the total distance between terminal points is not long enough for the aircraft to accelerate or decelerate to the desired final air

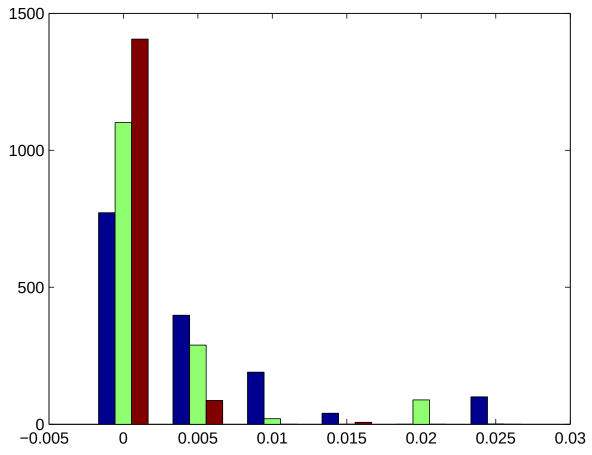


Figure 3.9: Relative Error Vs. N: Blue-N=500; green-N=1000, and red-N=5000

speed. For example, if the initial and final state are $z_0 = [-33\ 799\ \ -3012\ 20\ 0° \ 15°]$, $zf = [-467\ 888\ \ --2987\ 120\ \ -20°\ \ -160°]$ and $V_0 = 20$, $V_f = 120$, $N = 1000$ then the maximum air speed (accelerate with δ_{Tmin}) can be achieved over the generated shortest trajectory with time optimum thrust history is 97.99m/s, and the relative error is 18.342%, which is much above the average accuracy.

Execution time

As shown in Figure 3.10, the execution time of the algorithm increases almost linearly with N.

3.4 Flexible Collaborative Task Management for Autonomous Mobile Agents

This section discusses static decision making of the proposed scheme. Static decision making aims at pre-planned or known activities, such as traveling along predefined trajectories, and is computed off-line using centralized method.

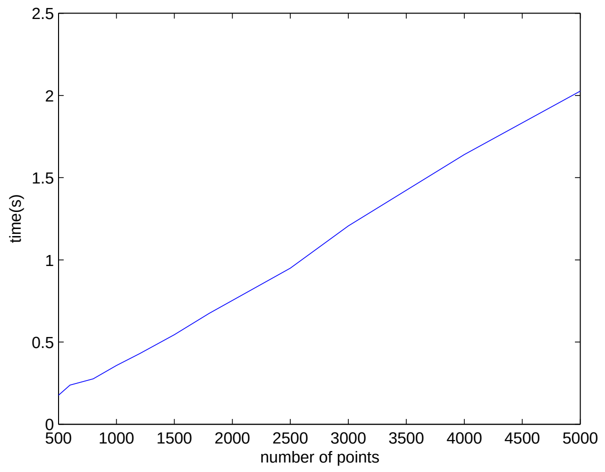


Figure 3.10: Execution time Vs. N

3.4.1 Problem Description and Modeling

The proposed method uses ILP(Integer Linear Programming) to find the task scheduling for autonomous mobile agents. It considers a group of autonomous mobile agents which can cooperate to meet the requirements of the application. The described algorithm finds the optimal path for handling the geographically distributed tasks in addition to resource allocation and task scheduling. Producing flexible solutions is also a novelty of the proposed method as compared to similar work.

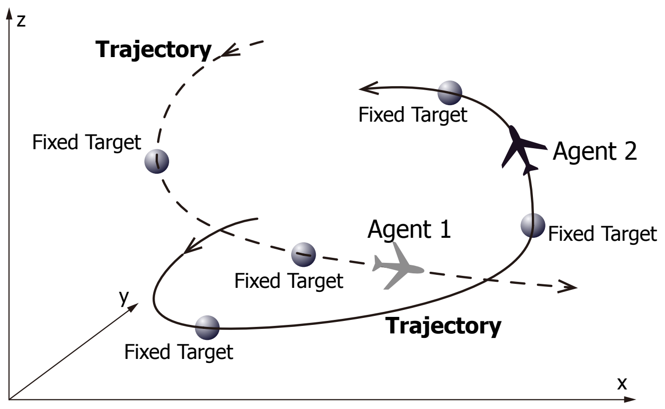


Figure 3.11: Cooperative operation of multiple autonomous mobile agents

This section defines the studied problem and presents the proposed modeling method.

Figure 3.11 summarizes the autonomous mobile agent behavior scenario. Multiple autonomous mobile agents must cooperatively tackle targets located in a 3D environment. Each autonomous mobile agent moves along a non-linear trajectory at a variable speed using a trajectory computing algorithm similar to [42, 54]. Autonomous mobile agents are heterogeneous, and they might have different dynamic characteristics (e.g., speed and acceleration). The speed magnitude and speed gradients are bounded.

Fixed targets are positioned at known locations, and must be tackled before a predefined time limit. Otherwise, the entire mission is considered to be compromised. The group of autonomous mobile agents must tackle all fixed targets before their time limit expires. The flexibility requirement states that the solution should maximize the chances of completing the mission in case autonomous mobile agents experience unexpected conditions that delay their activities. Fixed obstacles are present in the 3D environment, and must be avoided by the moving autonomous mobile agents. The tackling of targets comprises of the following sequence of activities: (i) flying to the target, (ii) detecting the target (e.g., through different sensors), (iii) handling the target (such as taking the picture of the target), and (iv) assessing the results of the activity [44, 53]. The three latter activities have known execution times, but the first step might take variable durations, depending on the position and flight characteristics of the autonomous mobile agent. Each autonomous mobile agent can handle multiple targets.

Autonomous mobile agents can collaboratively handle the same fixed target. For example, one autonomous mobile agent might detect and handle the target, and then transmit all the information to another autonomous mobile agent, which will do the assessment of the results. This scenario is useful considering that autonomous mobile agents have different capabilities (e.g., achievable speed), and resources (such as amounts of fuel). A slower autonomous mobile agent positioned nearby can do the assessment, while the more powerful autonomous mobile agent moves towards tackling the next target. As a trade-off, the collaborative scenario involves communication overhead for information transmission between the autonomous mobile agents, and also additional distances to be traveled (thus, higher fuel consumption) by the autonomous mobile agents involved in collaboration.

In summary, the addressed problem is as follows. An algorithm must be developed for tackling N fixed targets by M autonomous mobile agents with known but different characteristics. Each fixed target must be tackled before its predefined time limit expires. In addition, the flexibility of tackling new targets (known only at execution time) must be maximized.

Collaborative Approach

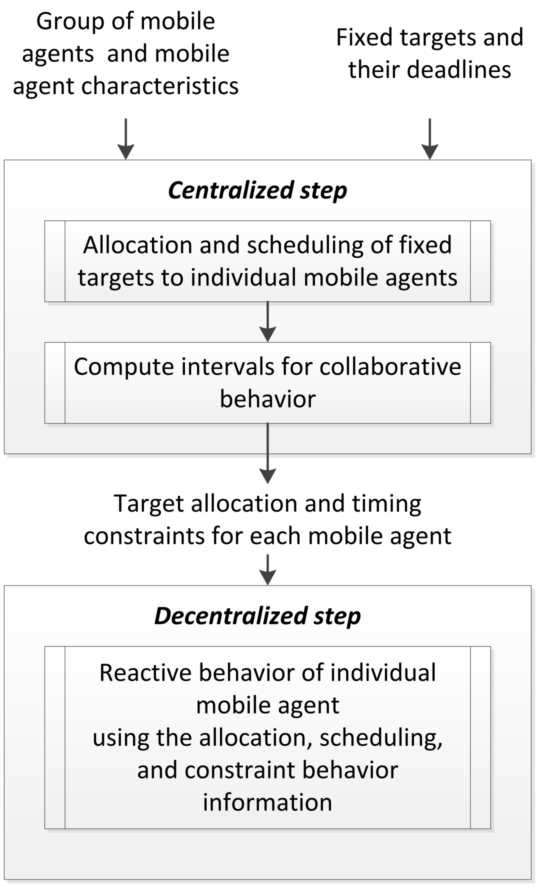

Group of mobile agents and mobile agent characteristics
Fixed targets and their deadlines
Centralized step
Allocation and scheduling of fixed targets to individual mobile agents
Compute intervals for collaborative behavior
Target allocation and timing constraints for each mobile agent
Decentralized step
Reactive behavior of individual mobile agent using the allocation, scheduling, and constraint behavior information

Figure 3.12: Decision making in collaborative autonomous mobile agent operation

Figure 3.12 presents the decision making approach that we are proposing in this paper for controlling the behavior of autonomous mobile agent groups. The approach represents a trade-off between centralized decision making, which is efficient and offers predictable performance (e.g., satisfaction of the predefined deadlines), and decentralized control, which is more scalable and flexible in tackling new situations. Centralized decision making is at the level of the entire autonomous mobile agent group, and decentralized control is at the level of each individual autonomous mobile agent. The approach uses an off-line, centralized decision making step to compute the allocation of fixed targets to each autonomous mobile agent, and the scheduling in time of the activities related to the handling of a target. In addition, the centralized step also calculates the constraints that encompass the collaborative behavior of each autonomous mobile agent. During operation, each agent decides dynamically (after a collaboration request has been formulated) whether it will participate to the collaboration, or not. The decision is made depending on its current status and geographical position so that the deadlines of its allocated targets are not violated. As collaboration requests are formulated dynamically and cannot be predicted off-line, the optimization goal is to maximize the chances of an autonomous mobile agent to participate to collaborations by computing the allocation and scheduling solution that maximizes the flexibility of an autonomous mobile agent to participate to collaborations.

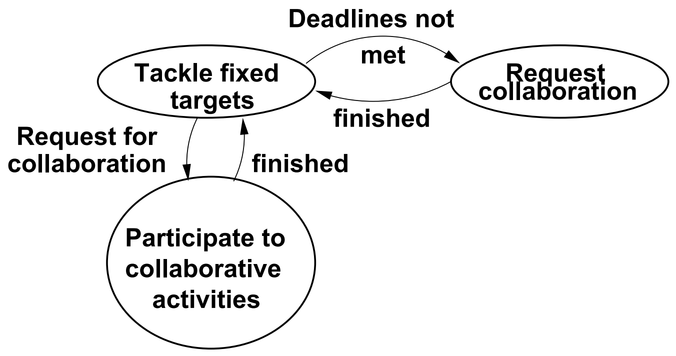


Figure 3.13: Decentralized controller for autonomous mobile agent operation

Each autonomous mobile agent executes its own decentralized controller, which implements a reactive behavior expressed through a Finite State Machine (FSM). The controller decides the specific actions of an autonomous mobile agent, e.g., pursuing a fixed targets, or satisfying a request for collaboration. Figure 3.13 shows the structure of a simplified reactive controller. In normal operation mode, the autonomous mobile agent is tackling a fixed target following the allocation and scheduling decisions of the centralized step. If the controller detects that the deadlines fixed for the target currently being handled cannot be met, it formulates a request for collaboration. If the request is granted by another agent then the autonomous mobile agent moves on to handling the next assigned target. If the request is not granted then the autonomous mobile agent might decide to continue with the current activity even though this results in violating the targets deadline, or leaving the task unfinished in order to move on to the next assigned target. The focus of the section is on the centralized decision making algorithm, including target allocation and scheduling, and computing of constraints related to the collaborative actions of the decentralized controller. The next subsection presents autonomous mobile agent behavior modeling for centralized decision making.

Problem Modeling

Figure 3.14 illustrates the task graph for tackling N fixed targets. Each target tackling activity is an independent thread of tasks consisting of separate tasks for flying, detection, handling, and assignment. Task dependencies express the required order of performing the tasks. As the N targets are known, the times for detection, handling, and assignment are fixed for a given autonomous mobile agent type. Please note that these times are different for autonomous mobile agents with different characteristics. In contrast, the flight time is not known in advance because the time required for reaching a target depends on the computed autonomous mobile agent trajectory. Moreover, the autonomous mobile agent trajectory depends on the position of the fixed target previously tackled by the autonomous mobile agent, and therefore on the previous decisions on target allocation and scheduling.

Fixed targets. The task graph includes tasks that can represent a collaborative behavior between multiple autonomous mobile agents in tackling the same fixed target. For example, after the target was detected by an autonomous mobile agent, the method allows that other

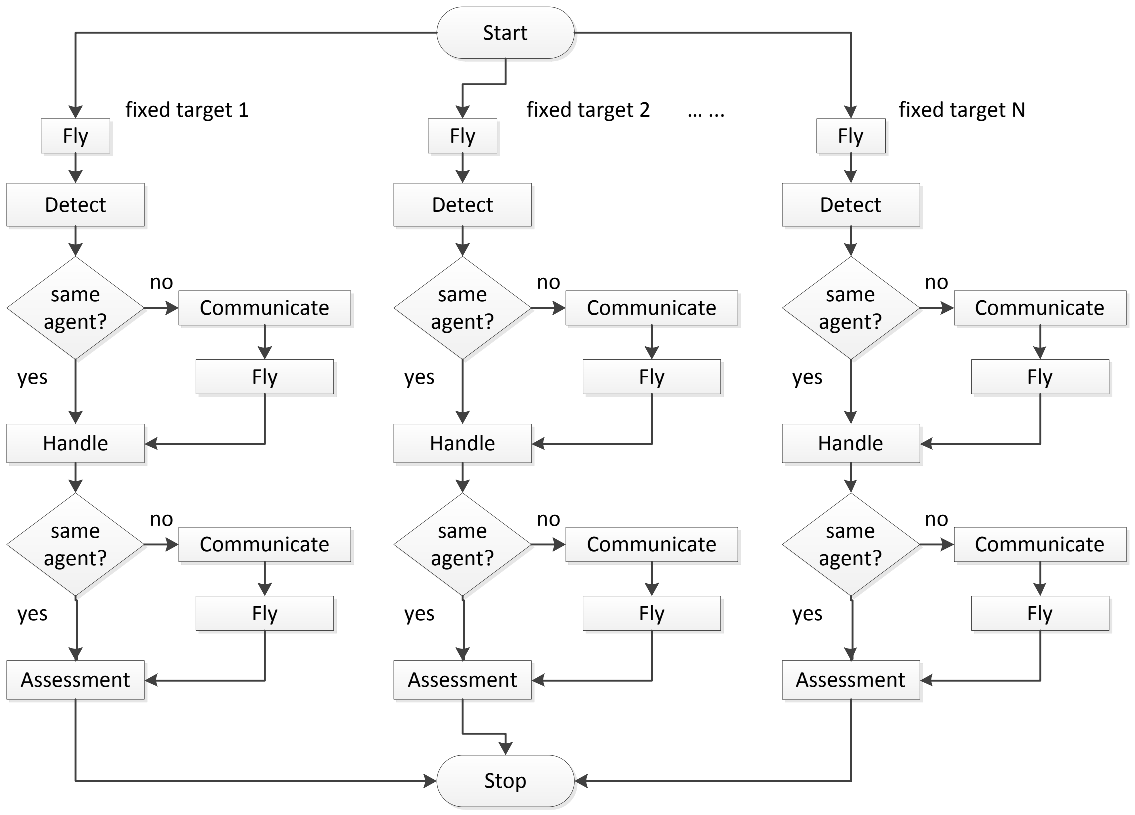


Figure 3.14: Task graphs for target tackling

autonomous mobile agents handle and/or assess the target. These actions are represented by conditional blocks (the blocks labeled as 'same agent?' in the figure), which continue with the tasks for communication and flight, if a different autonomous mobile agent is involved. The communication time is fixed (as the amount of data to be transferred is given), but the flight time is variable as it varies with the position of the mobile agent entering the collaboration. The flight time includes the total time spent by an autonomous mobile agent for moving for a new activity as well as the time for accomplishing that activity. Since the nature of the collaborative activity is decided on-line, the actual flight time is not known during the step of off-line centralized decision making, and instead the methodology should maximize the overall capability of an autonomous mobile agent group for collaborative activity.

3.4.2 Proposed Algorithm

This section describes the centralized task assignment and scheduling problem as an Integer Linear Programming (ILP) problem. The model can then be solved using an existing ILP solver or an heuristic algorithm to obtain the centralized controller of an autonomous mobile agent group. Figure 4 is used to explain the ILP expressions. The following equations are used to build the ILP model: The following set of equations are used to build the ILP model:

1. **Task start time**

   The start time of a task i is larger than the end time of its preceding task j:

$$t_{i,start} \quad \geq \quad t_{j,end} \tag{3.10}$$

2. **Task end time**

$$t_{i,end} = t_{i,start} + x_{1,i}\, T_1 + x_{2,i}\, T_2 + ... + x_{M,i}\, T_M \tag{3.11}$$

The end time for executing a task (e.g., detection, handling, and assessment) is equal to the start time of the task plus the time T_i required for $Agent_i$ ($i = 1, ..., M$) to perform the task. Values T_i are constants for a set of autonomous mobile agents. Variable x_i is one, if the task is performed by $Agent_i$, otherwise it is zero.

3. **Task allocation to autonomous mobile agents**

   Each task pertaining to a fixed target must be allocated to exactly one autonomous mobile agent, which performs the task. For task k, this requirement is expressed as follows:

$$\sum_{i \in Agents} x_{i,k} = 1 \tag{3.12}$$

4. **Task scheduling to autonomous mobile agents**

   Each autonomous mobile agent can handle multiple fixed targets, one target at a time. The set of ILP equations must include relationships that constraint the mobile agent to execute a single task at a time. For the tasks pertaining to the same fixed target, these constraints are implicitly introduced by the equations 3.10, which represent the sequencing constraints of the tasks.

   For the tasks related to different fixed targets allocated to the same autonomous mobile agent, the constraint is that the agent tackles a new target only after it finished tackling the current target. Allowing the autonomous mobile agent to intertwine the tackling of the two targets would result in unnecessary time overhead due to the extra distance the autonomous mobile agent must travel between the two fixed target. The overhead obviously affects the optimality of the scheduling result.

   A 0/1 variable $z_{i,j}$ is defined for each pair of fixed targets i and j. If both targets are tackled by the same autonomous mobile agent, then the variable being one indicates that task i is tackled before task j, and after task j, if the variable is zero. This constraint is captured as follows:

$$T_{i,end} \leq T_{j,start} z_{i,j} \sum_{k \in Agents} x_{k,i} x_{k,j} + T_\infty (2 - z_{i,j} - \sum_{k \in Agents} x_{k,i} x_{k,j}) \tag{3.13}$$

$$T_{j,end} \leq T_{i,start} (1 - z_{i,j}) \sum_{k \in Agents} x_{k,i} x_{k,j} + T_\infty (1 + z_{i,j} - \sum_{k \in Agents} x_{k,i} x_{k,j}) \tag{3.14}$$

T_∞ is a very large value.

5. **Autonomous mobile agent flight time to fixed targets**

   The flight time T_{fly} to a fixed target depends on the previous position of an autonomous mobile agent, which results from the fixed target allocation and scheduling. Target allocation and scheduling is computed through ILP equation solving, and is obviously unknown at the time of setting up the ILP equations. The proposed solution is to introduce a 0/1 variable $w_{i,j}$ for each pair i and j of fixed targets to describe that the same autonomous mobile agent successively tackles the two target (one immediately after the other). If the variable is one then target i is handled right before j. Otherwise, the variable is zero. In addition, the same autonomous mobile agent must tackle both tasks.

   The flight time T_{fly} to a fixed target j is defined as follows:

$$T_{j,fly} \propto \sum_{\forall target_i} w_{i,j} Dist(target_i, target_j) \times (\sum_{k \in Agents} x_{i,k} x_{j,k}) \tag{3.15}$$

   The next constraint expresses that each task is tackled by one autonomous mobile agent after the agent handles exactly one task (with the exception of the "dummy" start node), so for each task i:

$$\sum_{\forall target_j} w_{i,j} = 1 \tag{3.16}$$

6. **Collaboration of autonomous mobile agents**

   In collaboration, the identity of the collaborating autonomous mobile agents and the nature of the activities involved in collaboration is not known for centralized decision making, but instead is decided during autonomous mobile agent operation. The centralized decision making assigns and schedules tasks so that the flexibility of collaboration (if needed) is maximized.

As shown in Figure 3.14, an autonomous mobile agent might decide to collaborate after each of the activities related to a task, such as the fly, detect, and handle activities. The flexibility for collaboration depends on the time slack between the end of the current activity and the beginning of the next activity, and the deadline of the target handling for which the collaborative action is requested. The more overlapping exists between the slack time and the deadline the more flexibility exists in collaborating to meet the deadline. If there is no slack time or no overlapping with the deadline(e.g., the deadline is before the starting of the slack time, or after the end of the slack time) then there is no possibility of the autonomous mobile agent to participate in handling the target.

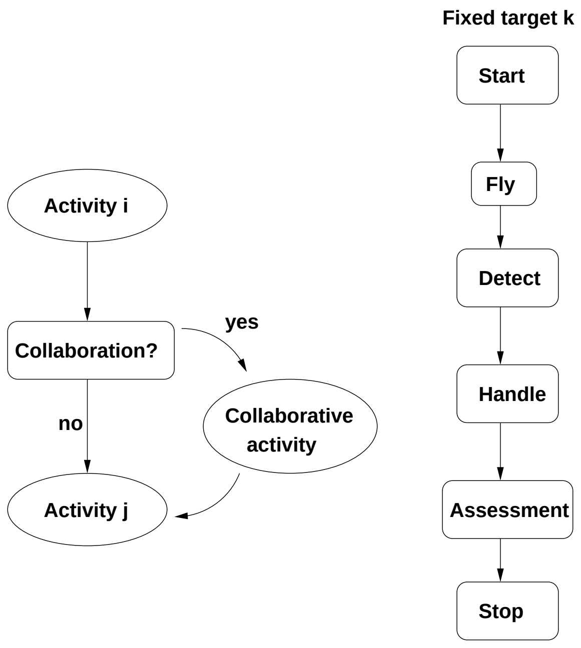


Figure 3.15: Modeling for dynamic collaboration

Figure 3.15 is used to explain the ILP equations for collaboration. For each target k, we define $SetC_k$ as the set of targets for which the assigned autonomous mobile agents are candidates for collaboration. $SetC_k$ can be set statically based on the geographical proximity of the targets (this information is known), or can be decided dynamically at run time depending on the current slack time of an autonomous mobile agent, and hence its flexibility to fly to more distant targets without violating the deadlines of its assigned targets. In this discussion, we assumed that $SetC_k$ is static.

The flexibility in participating to a collaborative handling of target k between activities i and j (scheduled in this order) is proportional to the following value:

$$Fl_{i,j,k} \propto [Activity_{i,end}, Activity_{j,start}] \cap [(Deadline_k - \sum T_k), deadline_k] \qquad (3.17)$$

Variables $Activity_{i,end}$ and $Activity_{i,start}$ are the end time of Activity i and the start time of Activity j. $[Activity_{i,end}, Activity_{j,start}]$ represents the time interval defined by the two moments. $Deadline_k$ is the deadline set for target k, and $\sum T_k$ is the time required to perform all activities related to target k, e.g., detect, handle, and assess.

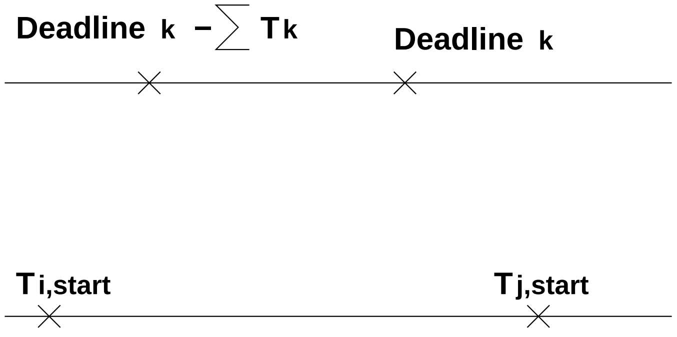


Figure 3.16: Flexibility in collaboration

Figure 3.16 illustrates the definition of the flexibility constraint for autonomous mobile agents collaborating on the handling of target k. Targets i and j are allocated to the autonomous mobile agent. The condition for collaboration is defined by the following equations:

$$T_{i,end} \leq Deadline_k - \sum T_k \qquad (3.18)$$

$$Deadline_k \leq T_{j,start} \tag{3.19}$$

The equation for the flexibility for autonomous mobile agent m is shown in equation 3.20. Where $x_{m,i}$ and $x_{m,j}$ are both equal to one, it means the same autonomous mobile agent m are handling $Activity_i$ and $Activity_j$. $x_{m,k}$ is required to equal to zero, which means autonomous mobile agent m is not initially assigned to $Activity_k$, but autonomous mobile agent m plans to handle $Activity_k$ on account of flexibility $Fl_{i,j,k}$'s calculation.

$$Fl_{i,j,k} \propto x_{m,i}x_{m,j}(1 - x_{m,k})(Deadline_k - T_{i,end})(T_{j,start} - Deadline_k) \tag{3.20}$$

The total flexibility of an autonomous mobile agent to participate to collaborative activities is:

$$Fl_{i,j} = \sum_{k \in SetC_k} Fl_{i,j,k} \tag{3.21}$$

The overall cost function includes a term to maximize the flexibility of autonomous mobile agents participating to collaborative activities.

7. **Cost function**

   The cost function is a weighted sum that express the goals of (i) minimizing the cumulated penalties for exceeding the predefined deadlines for the static targets, and (ii) maximizing the probability for tackling pop-up targets:

$$Cost = \alpha \times \sum_{\forall targets_i} (T_{i,end} - T_{i,deadline})^2 + \beta \times \sum pPP_i + \gamma \times \sum_{\forall targets_{i,j}} Fl_{i,j} \tag{3.22}$$

3.4.3 Experimental Results

This section presents the experimental results for the proposed algorithms. An heuristic algorithm was developed for solving the ILP model presented in Section IV. The algorithm is based on Simulated Annealing. It minimizes the cost function while satisfying all the constraints of the ILP model. Using an heuristic algorithm instead of an ILP solver offers two important advantages: it scales better than solvers for large ILP problems, and it does not have convergence problems, which is important for the reliability of the method. The algorithm was implemented in C language and run on a SUN Sparc workstation.

The experimental set-up varied the number of the targets to be tackled, the number and characteristics of the autonomous mobile agents, and the geographical position of the targets. Three different cost functions were optimized: (i) minimize the total execution time needed for tackling all targets, (ii) minimize the total distance traveled by the autonomous mobile agents, and (iii) maximize the flexibility of the solution for collaboration between the autonomous mobile agents. In addition to the quality of the solutions, experiments observed the computational characteristics of the heuristic algorithms, such as the execution time, the iteration at which the best solution was found, and the total number of iterations performed by Simulated Annealing. The scalability of the algorithm with the number of targets was also observed.

Table 3.8: Optimization for minimum total time

Ex.	# nodes	total time	best at (#)	total #	exec. time (sec.)
SN 1	11	64	23,244	43,244	49
SN 1	11	64	23,244	43,244	49
SN 2	14	89	13,664	33,664	37
SN 3	20	164	26,874	46,874	84
SN 4	38	337	9,954	29,954	205
SN 5	50 6	49	100	20,100	151

Table 3.8 presents the characteristics of the different experiments as well as the results obtained for minimizing the total time required for tackling all targets. This experiment was used as a reference for comparing the optimization results for minimizing the total

distance traveled by autonomous mobile agents and maximizing the flexibility, respectively. The second column indicates the number of nodes in the task graphs for target tackling (similar to Figure 3.14). The third column presents the minimum execution time for tackling all targets as found by the algorithm. Column four shows the iteration number at which the best solution was found. Column five indicates the total number of iterations performed by Simulated Annealing, and Column six presents the execution time of Simulated Annealing. As expected for an heuristic algorithm, the convergence does not increase with the problem size as the total number of iterations depends mainly on the stochastic dynamics of Simulated Annealing. The execution time increases with the problem size, however it remains reasonably large even for the larger examples. This indicates that the algorithm scales fairly well with the size of the problem. For the two smaller examples, we manually verified that the found results are optimal.

Table 3.9: Optimization for minimum total distance

Ex.	Min.dist.		Min.time		Improv.(%)	
	total distance	total time	total distance	total time	total distance	total time time
SN 1	56	81	64	70	12.5	13.5
SN 2	89	104	73	91	19.2	12.5
SN 3	200	256	280	164	28.5	35.9
SN 4	586	395	619	337	5.3	14.6
SN 5	601	356	650	349	7.5	1.9

Table 3.9 presents the optimization results for minimizing the total length traveled by autonomous mobile agents. Columns two and three indicate the total distance and the total time resulting for this cost function. For comparison purposes, Columns four and five show the total distance and total time found for the cost function minimizing the total time (also shown in Table 3.8). Finally, Columns six and seven indicate the relative improvements in terms of total distance and total time between the two optimization requirements. Column six shows that the optimized paths can be 5.3% to 28.5% shorter than the total paths for solutions optimized for time. However, the paid penalty is in longer total time, which can

be larger by values between 1.9% to 35.9%.

Table 3.10: Optimization for maximum flexibility

Ex.	Max. flex.	Init. flex.	Improv.(%)	Total time	Total dist.
SN 1	78	66	15	71	101
SN 2	112	16	85	99	155
SN 3	148	30	79	197	368
SN 4	272	76	72	408	873
SN 5	436	274	37	540	943

Table 3.10 offers results for resource allocation and scheduling optimized for flexibility. Column two presents the maximum flexibility. Column three gives the flexibility produced by a simple list scheduling algorithm. Column four shows the relative improvement. For comparison purposes, Columns five and six indicate the total time and total distance of the solutions optimized for flexibility. The experiments show significant improvement in the flexibility, between 15% and 85%. However, increased flexibility results at the penalty of longer times and traveled distances as compared to the previous two cost functions.

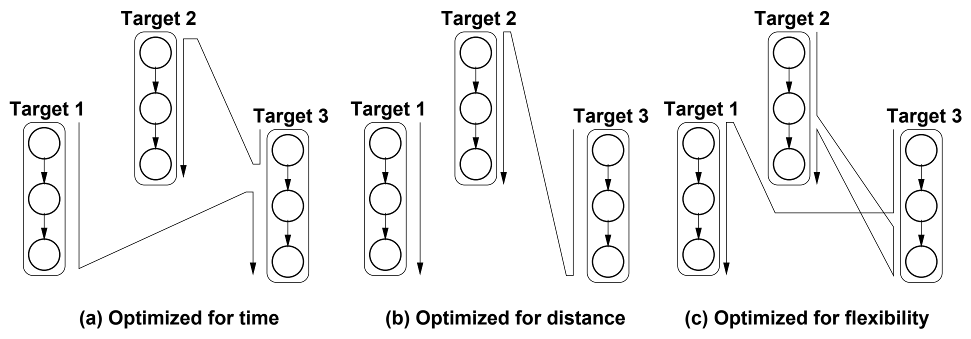


Figure 3.17: Flexibility in collaboration

Figure 3.17 illustrates the nature of the solutions found for example SN 1 and two autonomous mobile agents. Similar results were obtained also for the larger examples. Time minimizations always distribute targets to autonomous mobile agents such that there is an equal loading of the two autonomous mobile agents. The loading includes the execution times of the tasks as well as the distance traveled by the autonomous mobile agents. Distance optimizations tends to assign clusters of neighboring targets to autonomous mobile agents even though this might result in unequal loading of the autonomous mobile agents. Finally, flexibility optimization encourages a scheduling such that the two autonomous mobile agents perform as much as possible their assigned task in parallel.

3.5 Conclusions

This chapter presents the work on developing distributed control methods for large-scale groups of vehicles as well as an approach to performance predictive collaborative control of autonomous mobile agents tackling fixed targets. We introduced the goals, and summarized the trajectory generation algorithm used to model autonomous mobile agents. Experimental results of the algorithms are also offered.

We proposed different forms of expression, both linear and non-linear for trajectory generation model. Experimental results show that the proposed collaborative algorithm scales fairly well for large problems, has a reasonably long execution time, and can significantly improve the quality of the produced solutions, such as up to 28.5% reductions of the total path traveled by autonomous mobile agents and up to 85% improvement in the flexibility of the solution.

The related literature discusses several optimization algorithms used to build the model from experimental data, such as Simulated Annealing, and Neural Networks. So far, Neural Networks seem to give the best convergency. However, since this kind of model is hard to study, any impacts of physical conditions on the optimization are difficult to understand. As the relationship between flying conditions and characteristics of trajectory is not easy to express, one of the possible future directions is to classify the simulated results of flights which share certain common characteristics into groups, and then study inner-group and inter-group accordingly.

Chapter 4

Dependable Distributed Data Acquisition Through Groups of Autonomous Agents

This chapter discusses dynamic decision making at the physical level using game-theory-inspired ideas. The purpose of dynamic decision making is to decide an agent's deviations from the original (off-line) plans depending on the agent's goals. Any adaptation decisions are computed online using decentralized, game-theoretic method.

4.1 Introduction

Guaranteeing dependable and predictable operation of autonomous agents is challenging considering that the mobile agents operate in environments with many unknown and changing attributes. For example, conditions like wind, rain clouds, bird flocks, etc. can significantly influence the trajectories and speed of the agents and thus make any pre-computed plans infeasible.

Model-based CPS decision making uses dynamically-constructed models to make decisions during operation. For example, a CPS for environmental monitoring and protection, needs to detect the position of zones polluted with toxic substances as well as the nature and level of the toxic substances. The decisions on how to dispatch the mobile agents to maximize pollution detection and the utility of the acquired data samples are performed based on a model that predicts the position of spills and their dynamics over time. A main challenge is data acquisition through the mobile agents to construct precise models while minimizing the cost of creating the models.

Figure 4.1 illustrates the main characteristics of the problem of a group of mobile agents that optimize their data acquisition. The figure presents two agents (robots) that each have a scheduled path: agent 1 travels through points A, B and C, and agent 2 goes through points D and B. Along their paths, an agent identifies previously unknown regions of interest, e.g., the presence of a certain toxic substance. The unknown regions of interest are shaded in the figure, and the zones sampled by the agents are darkened. Note that multiple agents might sample the same zone. The goal is to maximize the utility (for decision making) of the data sampled by the agents. Depending on the nature of the application, utility can have different formulation, such as maximizing the physical size (e.g., surface, volume) of the found regions of interest, or maximizing the total quantity of identified substance.

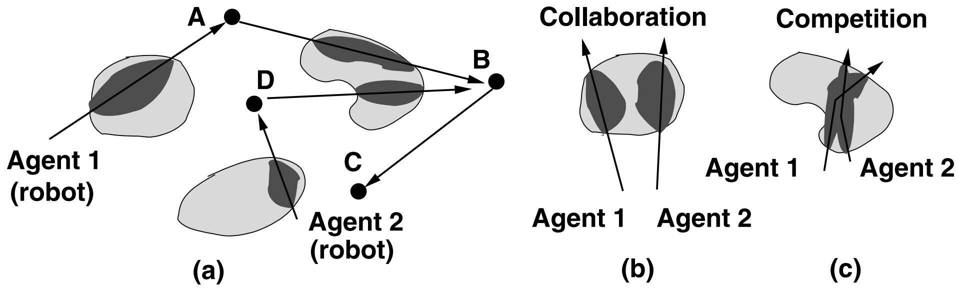


Figure 4.1: Problem description for distributed data acquisition

The distributed data acquisition problem originates a mixture of collaborative and competing actions between the participating agents. However, it is needed to clarify that the proposed decision making scheme is not based on voting, therefore the agents do not do any bidding and there is no auctioning mechanism. The scheme is inspired by interactive prisoners' dilemma problem [83]. For example, in Figure 4.1(b), the two agents sample different regions and thus increase the utility of each individual sampling as a more comprehensive, more accurate model is produced. This represents collaboration between the two agents. In contrast, in Figure 4.1(c), the two agents sample overlapping regions and hence generate redundant data of little use. As each agent operates to maximize its individual utility, the two agents compete on sampling the common areas. Moreover, the nature of the interactions between agents (e.g., collaborative or competing) changes dynamically depending on the evolution of the process and the utility of the actions. Devising interaction schemes to maximize data acquisition utility while agents can collaborate or compete is still a challenging problem.

The discussed problem can be formulated as follows:

$$\max Objective_{overall}, s.t. \tag{4.1}$$

$$\max_{k} Utility_k, \forall k \in S \tag{4.2}$$

$$Usedresources_k \leq Resources_k, \forall k \in S \tag{4.3}$$

$Objective_{overall}$ is the overall criterion (cost function) to be optimized for the data collection problem. For example, the objective could be to maximize the overall quantity of identified substance of interest, e.g., $Objective_{overall} = \int_V Q(v)\mathrm{d}v$, with $Q(v)$ being the substance density at any point v in volume V. Another case is if $Objective_{overall} = \int_V \delta(v)\mathrm{d}v$, where function $\delta(v)$ is one if the substance was detected at point v, and zero, otherwise. This objective function expresses the intention to maximize the complete clean-up of zones.

Equation (4.2) states that every agent of set S optimizes its own utility. The utility represents the agent's unique contribution to maximizing the application's objective. The equation also defines that the agents operate decoupled from each other. For example, if an agent is the only one that covers a certain physical zone then its utility coincides with the corresponding improvement of the objective function. If several agents cover the same zone then their utility is less as the improvement offered by an agent is achievable by another agent too.

Equation (4.3) states that the resources used by agent k to maximize its utility must be less than its available resources. The modeled resources include the available hardware, energy, and time budgets (time flexibility) in meeting the deadlines of the application [66].

The distributed data acquisition problem defined by equations (4.1)-(4.3) is solved by a procedure in which each agent implements asynchronous decision making based on a local model that predicts the characteristics of the tackled problem (e.g., the attributes of the monitored space) and the expected decisions of the other agents. This implicitly coordinates the agents to collaborate on maximizing the overall objective in equation (4.1) and minimizing the competition between agents as defined by formula (4.2).

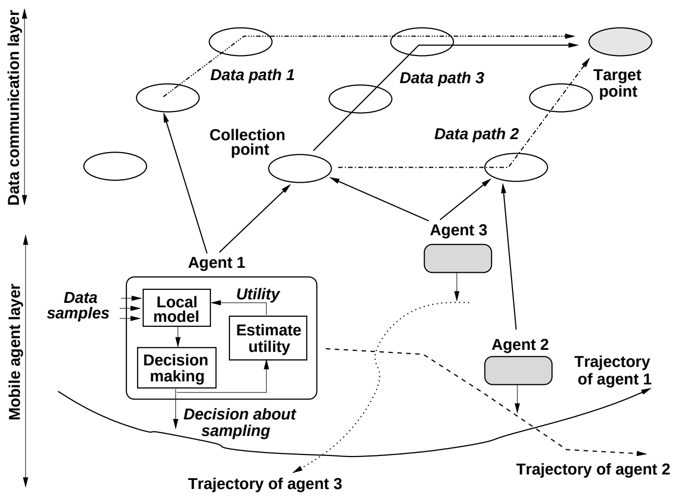


Figure 4.2: Operation principle of decoupled agents

4.1.1 Problem Description

The proposed approach to solve the distributed data acquisition problem based on mobile, decoupled agents includes two layers. As presented in Figure 4.2, the mobile agent layer includes several mobile agents, which travel along their own trajectory to acquire signals and data distributed in space and time. Then, the collected data is transmitted through a wireless connection from the agents to the collection points of the data communication layer. This layer includes embedded collection points which are linked through wireless links in a grid of cells. Data is sent along several data paths from the collection points to the target points, where the data acquired by the mobile agents is used in decision making.

At the mobile agent layer, every agent makes individual decisions based on a local model that is constructed using (i) data samples collected by the agent and (ii) data received occasionally from the collection points based on a loosely-coupled interaction scheme. The interaction scheme defines that agents interact with the data communication layer only after long time intervals while they operate totally decoupled between interactions. The effectiveness of the decisions is evaluated through a utility estimation module that uses the difference between the estimated and real utility to request synchronization to adjust the local models.

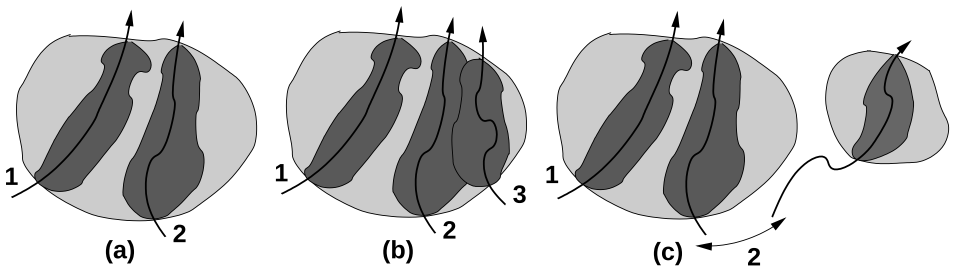


Figure 4.3: Utility function

The utility of the model changes as newly acquired data is integrated into the model to update the expression of *Function*. The utility change $\Delta Utility$ includes three components:

- *New insight*: New insight is added to the prediction capability of *Function*, if the data is sampled from new regions and/or conditions. For example, in Figure 4.3(a), the second agent samples a different zone than the zone from which agent 1 acquires its data. The two areas are shown as dark zones.

- *Increased confidence*: The new data improves the prediction capability of *Function*, while the variable set and the variable domains stay unchanged.

- *Enabling future insight*: The acquired data can have potentially a high usefulness in providing insight for future needs or can be utilized by other agents. For example, in Figure 4.3(c), an agent can decide to acquire data from the left or right zone. The gained insight is similar as both zones are not visited by other agents too. However, the utility of sampling the left zone is higher because there is one more agent analyzing a different part of the same zone. This creates the potential that the overall utility of the model is higher due to the more comprehensive insight about the zone.

At the level of the data transportation layer, the correlations between the trajectory of the mobile agents and the selected data communication paths influences both the experienced data loss and delays while forwarding data to the target point. The two issues are essential in deciding the quality of the data used in decision making. In general, data loss occurs when data stored in buffers is overwritten before it is forwarded either because of an ongoing data sampling or data reception. This situation also increases the delay of transmitting data to the target node. This chapter focuses only on the data acquisition through the mobile agents.

In conclusion, resource allocation, e.g., the frontend resources used for localizing the sound source by a node and the selected data communication paths, determine the three main factors defining the error of the models used in decision making. The quality of resource allocation schemes depends to a high degree on the characteristics of the environment, e.g., ambient noise, and monitored phenomena, i.e. trajectory.

This chapter proposes a novel, asynchronous interaction scheme between agents that operate decoupled for most of the time. The scheme is based on a mathematical model of data acquisition utility, and how utility changes as more data is acquired. The model captures aspects, like gaining new insight into the process, increasing the confidence of the models used in decision making, and the collaborative-competitive aspect of agent interactions. The model is then utilized to infer the selection ratios that decide how alternative sampling steps are used by an agent to maximize the data acquisition utility, while the agent operates decoupled from the other agents. Experiments study the effectiveness of the proposed scheme in producing comprehensive data acquisition while minimizing the amount of redundant (unnecessary) data collected by the decoupled agents.

The chapter has the following structure. Section 4.2 describes related work and Section 4.3 discusses proposed solution and Section 4.4 offers experimental results. Finally, our conclusions are put forth.

4.2 Related Work

Data acquisition strategies for autonomous robots are basically classified into three main types: predefined trajectories, *NBV(Next Beat View)* problem, and hybrid systems. In predefined trajectory systems, auto robots move along fixed trajectories and complete a series of tasks [71, 72]. Predefined trajectories work in static environment which do not have dynamic attributes, and require information about the environment to compute trajectories. These two features make strategies which computing trajectories in advance, hard to implement in reality. Predefined trajectories also have poor adaptability when facing different environments and potential environmental changes.

NBV problem [68, 73, 74, 76, 81], mainly calculates the next observation point at each step with information from neighboring points. *NBV* is an online path planning strategy and is more adaptive to environments but the primary drawback is that it is computationally intensive. In literature, simple *NBV* strategies include greedy strategies which are always seeking the highest benefit or lowest cost regardless of any other factors such as random points sweep, and wall to wall sweep [73]. These strategies require relatively less neighborhood information but do not work well in complicated scenarios. A *Utility function (UF)* is typically used to select next best exploration point among a set of candidates. Utility function is a measure of the benefit attained, when deciding among multiple states. In an unknown environment, which is usually the case in reality for autonomous robots, utility is expressed as expected value for probability distributed states [70]. The simplest form of *UF* contains only one factor which is usually the cost to travel to the new data collection point, and this cost could be time consumption, travel distance or information gain. In [77], relative entropy is used as information metric for the direction control. The exploration strategy in [78] navigates robots to the nearest accessible, unvisited frontier. [79] proposed three forms of criteria: closest unvisited, unscanned, or informative location without taking sensor range into account. There is another strategy category which employs linear combination of two or more terms such as distance traveled and the information gain. As proposed in [73], entropy gain from gathered information is an indicator of the benefit in *UF* and Euclidean distance to travel between points is cost. It is also applied to dynamic environment with moving obstacles. [74] builds a theoretically well defined *UF*; the cost term in selection criterion is formulated as the logarithm of the traveled distance to reach the candidate position and scan range, while the expected entropy increase is affected by scanned point, new point seen from the candidate position as well as robot position error and sensor accuracy. Another form of linear *UF* is proposed in [76] which uses the separation of the robots and the time needed for the robots to reach their destinations. The above linear form can be formulated into the following format [74]:

$$f(x) = g(x) - \beta \cdot c(x) \tag{4.4}$$

$g(x)$ is expected gain to select x as new observation position, $c(x)$ is the cost to reach x, and β is a coefficient indicating the weight of gain over cost.

Exponential function was proposed in [81] with sufficient experimental data to justify its effectiveness:

$$f(x) = A(x) \exp(-\lambda \cdot L(x)) \tag{4.5}$$

$A(x)$ measures unvisited area visible from x, $L(x)$ is the length of path to x, and λ is the weight. The best candidate is selected as it maximizes $f(x)$.

Another innovative non-linear form in [75] used a *UF* with both grid and topology information to consider not only travel costs scaled by time consumption, and information gain, but also the hot points on the map with different levels of attraction to mobile robots:

$$f(x) = \frac{g(x)}{Tr(x) + Tob(x)} \cdot Topology(x) \tag{4.6}$$

$g(x)$ is information gain at point x, $Tr(x)$ and $Tob(x)$ are time used to travel to point x and to observe at point x respectively, $Topology(x)$ is topology features extracted from a grid map.

Although utility functions are the dominative criterion in *NBV* problem, Amigoni and others proposed a multi-objective strategy [68], which looks for Pareto-optimal position candidates who are not dominated by other position candidates, and then pick the one that is nearest to the ideal solution, instead of using traditional utility functions which are defined in ad-hoc manner. This strategy overcomes the false 'best candidate' brought by weight factors in *UF* when trading off between benefit and cost, and has a relatively good adaptability from simple to complicated environments. However, it takes more steps in selecting best candidate. Table 4.1 summarizes some common *UF* forms in literature.

4.3 Algorithms for Mobile Agents Layer

This section presents the modeling of the mobile agents layer and then the decision making algorithms of the decoupled agents, so that the application objective is optimized.

4.3.1 Mobile Agents Layer Modeling

The model used for the mobile agents layer is defined as the following quadruple:

$$Model = < Function, Error, Utility, Cost > \tag{4.7}$$

Function is the mathematical expression of the model used in decision making to

Table 4.1: Utility functions for data acquisition strategies

UF form	Annotation	Pro.	Con.
$c(x)$	Cost to next best point	Low computational complexity	Not accurate
$f(x) = g(x) - \beta \cdot c(x)$	Linear composition of benefit and cost	Moderate computational complexity; adaptive to different environments; relatively accurate; may be suitable for multi-robot systems; may get efficient results;	weak mathematical foundation
$f(x) = A(x) \exp(-\lambda \cdot L(x))$	Non-linear composition of benefit and cost	relatively accurate	moderate to high computational complexity; weak mathematical foundation; unknown adaptability to different environments;
$f(x) = \frac{g(x)}{Tr(x)+Tob(x)} \cdot Topology(x)$	Non-linear composition of benefit and cost; contains both grid and topological information	Combine grid and topology information; online analysis of topological map; considerable efficiency; suitable for multi-robot systems	more steps in selecting best candidate; high computational complexity;

optimize the application objective 4.1. *Error* is the prediction error of the model. *Utility* is the utility of using *Function* in decision making by an individual agent, e.g., improving the effectiveness of the selected decisions. The local utility of an agent is also considered in equation (4.2). *Cost* is the cost of building the model, including time, hardware resources, and energy.

The four components of the model are correlated with each other. For example, the error of the model function is correlated to utility, such as a smaller error implies a higher utility of the model. Similarly, producing a model of smaller error requires a higher cost. The specific nature of the two correlations depends on the application. The utility of the models in achieving the application goals must be optimized while the cost stays within the available resource limits.

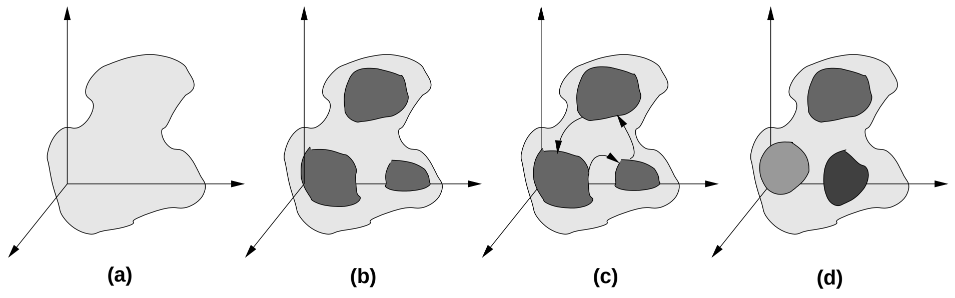


Figure 4.4: Model function types

Function expressions are continuous and differentiable functions over their domain. This represents situations in which the models are built for homogeneous media, such as if the same kind of sampled substance fills the considered space. For example, the amount n_s of solvent flowing into the air from a tank can be estimated using the formula [82]:

$$n_s = \frac{V}{R}[(a + \frac{ab}{p_T - b}) \ln \frac{p_T - b - aT_1}{p_T - b - aT_2} \frac{T_2}{T_1} + b(\frac{1}{T_2} - \frac{1}{T_1})] \tag{4.8}$$

where V is the volume of the head space, p_T the total pressure, R the gas constant, T_1 and T_2 the start and end temperature, and a and b are constants. Figure 4.4(b) describes homogeneous media but with discontinuous behavior, such as having three clusters in which different functions express the sampled data. For example, this situation can occur if the same solvent is flowing out through three different openings, so that the local pressure, temperatures and model parameters of the above equation are different for each cluster. The description in Figure 4.4(c) extends the previous case to situations in which the order of visiting the clusters also impacts the selected clusters. This is important if the model also predicts the causality of a phenomena, in which case the order is also important. The last figure represents a heterogeneous medium, in which three kind of substances fill the modeled physical area (shown with different patterns in the figure).

The new insight added through the acquired data includes two situations:

1. The data covers new situations, which is reflected through new variables being added to the expressions *Function* of the model. The change in utility, $\triangle_{(1,a)}Utility$, is described by the following expression:

$$\triangle_{(1,a)}Utility = \sum_{c \in newconds} (Utility(c) - Utility_0) \cdot probability(c) \tag{4.9}$$

   The change is equal to the sum of the difference of the utility value for each newly added condition c and the original utility $Utility_0$ times the probability of the condition being used in decision making. *newconds* is the set of all new conditions covered by the added data.

2. The data corresponds to previously uncovered physical zones. This extends the domains of the variables already present in the domain. Figure 4.3(a) illustrates this case. The data acquired by agent 2 extends the domains of the variables used in expressing Function based on the data sampled by agent 1. The change in utility is equal to:

$$\triangle_{(1,b)}Utility = \sum_{s \in \triangle Space} (Utility(s) - Utility_0) \cdot probability(s) \tag{4.10}$$

$\triangle Space$ represents the zone (e.g., volume or surface) from which the new data was acquired. $Utility_0$ is the original utility value, before the new data was acquired. $probability(c)$ is the probability of decision making utilizing information for $\triangle Space$.

The second component of $\triangle_{(2)}Utility$ refers to improving the utility value by increasing the confidence level of predictions through *Function*. More accurate predictions lead to superior decision making. Lets assume that the utility of *Function* in decision making depends on its mathematical form and error according to the relationship: $Utility = \Psi(Function, Error)$. Hence, the initial utility is $Utility_0 = \Psi(Function_0, Error_0)$. After considering the newly acquired data too, the function and its error change to *Function* and *Error*, respectively, thus $Utility = \Psi(Function, Error)$. The change in the utility value is as follows:

$$\begin{aligned}\triangle_{(2)}Utility &= \Psi(Function, Error) - \Psi(Function_0, Error_0)\\ &\approx \int_{Dom} probability(v)\frac{\partial\Psi}{\partial Function}(Function_0(v))\\ &\quad (Function(v) - Function_0(v))\mathrm{d}x \end{aligned} \tag{4.11}$$

where the change is computed over the entire domain *Dom* of *Function*. The third component $\triangle_{(3)}Utility$ characterizes the importance of the data acquired by an agent in the context of the data sampled by other agents for the same zone. The third component includes the following value computed for every pair of agents:

$$\triangle_{(2)}Utility = \frac{\int_{Dom} \triangle_{1,2}Utility(v)(Unique(v) - Overlap(v))\mathrm{d}v}{Dom} \tag{4.12}$$

$\triangle_{1,2}Utility = \triangle_{(1)}Utility + \triangle_{(2)}Utility$ is the sum of the first two components. Unique and Overlap defines if the agent does not or does overlap with the other agent in sampling data from *Dom*.

4.3.2 Adaptive Interaction Scheme

The considered distributed decision making scheme for solving the optimization problem defined by equations (4.1)-(4.3) is a mixture of planned and reactive actions. The planned activities define coarse-level objectives and set flexibility ranges that the low-level, reactive actions must meet. For example, the flexibility range of an agent indicates the extra time interval available to the agent to complete its scheduled activities without violating the constraint [66]. The reactive actions tackle any unknown or unpredicted aspects that occur during operation, such as uncertainties in the problem description and environment.

Every agent maximizes the following cost function in an attempt to implement equations (4.1) and (4.2):

$$\max Utility_k + \sum_{i \neq k} E[Utility_i] \tag{4.13}$$

where $Utility_k$ is the utility of agent k (the utility is computed using the local model of the agent) and $E[Utility_i]$ is the expected utility of the other agents. The first term reflects the requirement of equation (4.2) and the second term expresses the overall optimization objective in equation (4.1).

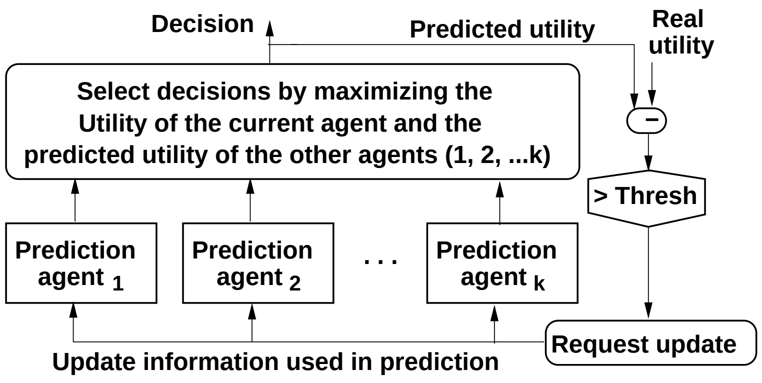

Decision
Predicted utility
Real utility
Select decisions by maximizing the Utility of the current agent and the predicted utility of the other agents (1, 2, ...k)
−
> Thresh
Prediction agent 1
Prediction agent 2
. . .
Prediction agent k
Request update
Update information used in prediction

Figure 4.5: Local decision scheme

The information needed to set up the formula for the expected utilities is collected through an asynchronous communication scheme between agents. The agents exchange data about their state (e.g., utilities) at intervals established based on need. The synchronization points re-adjust the local models to eliminate prediction errors about the other agents's utility, as the utility changes during asynchronous operation. Between two synchronization, the agents operate totally decoupled making predictions based on the state information received from the other agents. The goal of this communication scheme is to avoid excessive resource overhead, e.g., communication bandwidth. The scheme can also improve robustness of decision making as noise, errors, or outlier data is prevented from impacting the model representation and thus perturbing the functioning of the entire group of agents.

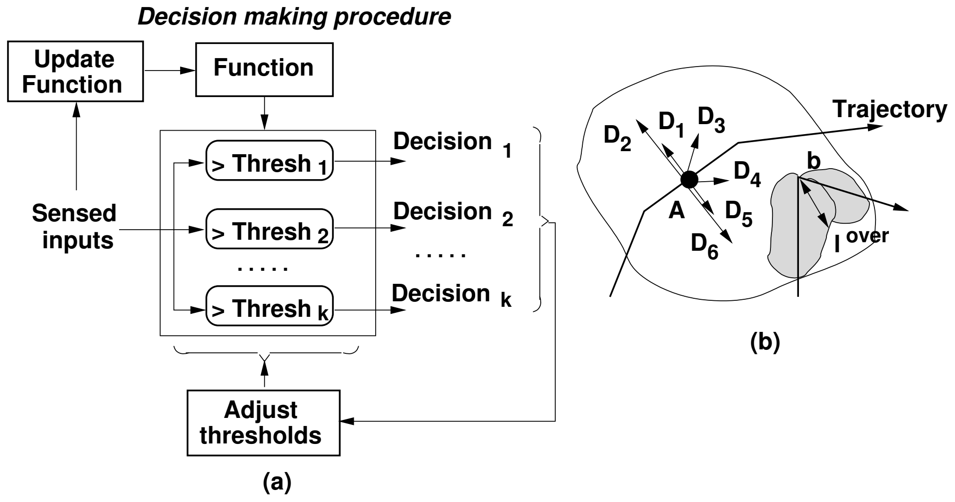


Figure 4.6: Sampling scheme

Figure 4.5 summarizes the local decision scheme of every agent. Each agent makes decisions that maximize the utility of the agent and the predicted utilities of the other agents (e.g., agents 1, 2, ... k). The total, predicted utility of a decision is compared to the real utility, acquired by analyzing the effects of the decision. If the difference is larger than a threshold value *Thresh* then the agent signals to the other agents its request to update the information used in predicting the utility of the other agents. The expected utility of the other agents is calculated by the prediction modules.

Figure 4.6 illustrates the data acquisition scheme of an agent moving along its trajectory. In every point of the trajectory, such as point A, the agent can select among p alternatives to acquire samples in addition to those along the trajectory. The figure shows six different alternatives (D_i), each having a different angle (with respect to the trajectory) and distance. The alternatives are selected such that two do not produce overlapping samples.

The selection ratio $p_{i,k}$ of alternative $D_{i,k}$ of agent k is formulated by the following set of equations:

$$\max \sum_i p_{i,k} Utility_{i,k} \tag{4.14}$$

$$\sum_i p_{i,k} = 1 \tag{4.15}$$

$$\min \sum_k \sum_i \sum_{l \neq k} \sum_j p_{i,k} p_{j,l} Overlap_{i,j} \tag{4.16}$$

$Utility_{i,k}$ is the utility of using decision $D_{i,k}$ of agent k and $Overlap_{i,j}$ is the overlapped data acquisition of agents k and l performing decisions $D_{i,k}$ and $D_{j,l}$, respectively. Equation (4.16) defines that the total overlapped sampling of all agents should be minimized, hence the total amount of unnecessary (redundant) samples should be kept to a minimum. Note that equation is nonlinear. The unknown selection ratios $p_{i,k}$ define the scheme used by agent k to sample the zone.

A simple alternative to solving equations (4.14)-(4.16) is that in which the selection ratio $p_{i,k}$ is computed using the following set of alternative heuristic rules:

1. *Heuristic 1*: The selection ratio $p_{i,k} \sim L_{trajectory}$, where $L_{trajectory}$ is the length of the trajectory of agent k. According to formula (4.9) and (4.10), this rule is expected to add more insight to the models as sampling along longer trajectories is likely to cover new areas.

2. *Heuristic 2*: The selection ratio $p_{i,k} \sim \frac{1}{Over}$, where $Over = \frac{\sum_{b \in Breaks} l_b^{over}}{L_{trajectory}}$. Term $Over$ estimates the amount of repeated samplings along the same trajectory. The amount increases with the length of the segments l_{over} which are broken through angles less than ninety degrees. Figure 4.6 illustrates a point b of the set Breaks of such angles. This rule reflects the uniqueness requirement of
formula 4.10.

3. *Heuristic 3*: The selection ratio $p_{i,k} \sim \frac{1}{Over_{k,l}}$, $Over_{k,l} = \sum_{k,l} \frac{l_{k,l}^{over}}{\min_{k,l} L_{trajectory}}$. The rule states that the selection ratio is inversely proportional to the total overlapping of the trajectory of agent k with the trajectory of agents l, which depends on the length $l_{k,l}^{over}$ of the trajectories' overlapping and the total lengths of two trajectories.

The three heuristic methods are simple and are computed only based on static information, e.g., the pre-assigned points that are visited by each agent. The trade-off between the gained utility (e.g., newly sampled points), the cost of reaching the sampled points, and the multiple sampling of the same point through decoupled agents is expressed only in a simple form. The second procedure to solve equations (4.14)-(4.16) expressing distributed data acquisition through decoupled agents is to consider a game theoretic approach [83]. This approach finds the probabilities $p_{i,k}$ of agent i performing task k knowing that each agent intends to maximize its own utility while minimizing their overlapping. The utility expression can be selected based on three forms described in the literature:

- *Linear form* [74]: The utility function is defined as $Utility(< x, y >) = Gain(< x, y >) - \beta cost(< x, y >)$, where $Gain$ is the gain obtained by sampling the physical point $< x, y >$ and cost is the cost of reaching the point from the current position, e.g., the length of the trajectory to the point. β is a weight.
- *Exponential form* [81]: The utility function is expressed as $Utility(< x, y >) = Gain(< x, y >) \exp(-\lambda cost(< x, y >))$. λ is a weight and all other variables have the same meaning as in the previous utility expression.
- *Non-linear composition* [75]: Utility is computed as $Utility(< x, y >) = \frac{Gain(<x,y>)}{cost(<x,y>) Sem(<x,y>)}$, where $Sem(< x, y >)$ defines the higher level information (semantics) extracted from point $< x, y >$.

The proposed procedure to solve equations (4.14)-(4.16) is based on insight from game theory. According to equation (4.14), each agent intends to maximize its own utility, which is the area of the zone from where data is collected. Without losing generality, we assume that the utility of sampling a previously un-sampled point is one, and that of re-sampling the same point is zero. Maximizing the covered area includes maximizing sampling from uncovered zones, and minimizing sampling from already covered zones (such zones cannot be avoided due to the autonomous operation of agents and the asynchronous communication scheme). Hence, sampling from overlapping zones by multiple agents implies solving for every agent k the following game theoretic equation [70]:

$$\min \max \sum_i p_{i,k} Utility_{i,k} \tag{4.17}$$

knowing that every agent optimizes a similar equation, hence plays the same game. In addition, the overlapping minimization condition defined in equation (4.16) can be restated as follows:

$$\min \sum_{\forall <x,y>} \sum_{k_{<x,y>}} \sum_i \sum_{l_{<x,y>}} \sum_j (p_{i,k})^{n_k(<x,y>)} (p_{j,l})^{n_l(<x,y>)} \tag{4.18}$$

$\forall <x,y>$ indicates all physical points that are sampled multiple times. $k_{<x,y>}$ and $l_{<x,y>}$ are two agent that sample point $<x,y>$. $n_k(<x,y>)$ is the number of decisions that agent k has to make to reach one point $<x,y>$, and $n_l(<x,y>)$ is the same number but for agent l. Assuming that $d_k(<x,y>)$ is the physical distance of agent k to point $<x,y>$ then value $n_k(<x,y>)$ can be estimated as $\frac{d_k(<x,y>)}{r_k}$, i and j indicate the available decisions (action), where r_k is the constant distance after which the agent k decides its next action. Similarly, $d_k(<x;y>) = \frac{d_k(<x;y>)}{r_l}$. The equation assumes that re-sampling the same point has utility zero, and otherwise the utility is one. Figure 4.7(a) illustrates the overlapped sampling of point $<x,y>$ by three agents.

Probabilities $p_{i,k}$ in equation (4.18) acts as correlators between the local decision making procedures of each decoupled agent. This is similar to correlated equilibria in game theoretic decisions [70]. The probabilities can be estimated in the following way. Using the method in [66], the minimum time t^{min} and maximum time t^{max} can be computed for the points

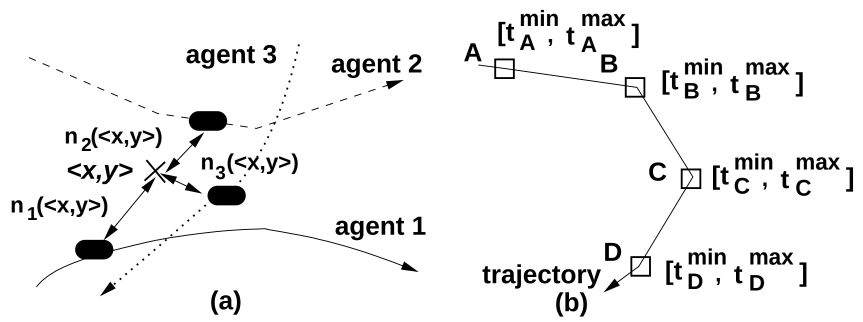


Figure 4.7: Game-theoretic decision making

defining a trajectory, as shown in Figure 4.7(b). Then, the probability for a given segment, such as segment A-B in the figure, can be computed as $p_{i,k} = \frac{t_B^{min} - t_A^{max}}{t_B^{max} - t_A^{min}}$. The probability characterizes the flexibility available to the agent to perform other actions in addition to moving to the next point of the trajectory. The value of p_j is adjusted depending on the current segment of the trajectory.

```
procedure decision_scheme (agent k)
  for every agent l != k do
    for every action j do
      compute prob$_{j,l}$ = (# of useful points)/
                             (total # of unexplored points);
    end for;
  end for;
  for every action i do
    Npoints$_i$ = 0;
    for every point <x,y> sampled through action i do
      Npoints$_i$++;
      S = 0;
      for every agent l != k do
        for every action j do
          if point <x,y> can be sampled by agent l then
            S = S + prob$_{i,k}$ x prob$_{j,l}$ ;
        end for;
      end for;
      compute prob$_{i,k}$ = (Npoints$_i$ - S)/Npoints$_i$ ;
    end for;
  end for;
  execute action i with highest prob$_{i,k}$ ;
end procedure
```

Figure 4.8: Game-theoretic heuristic

To summarize, the proposed approach to design decentralized decision making schemes finds the probabilities $p_{i,k}$ of agent k performing decision i knowing that each other agent intends to maximize its own utility too while minimizing their overlapping with other agents. The probabilities of re-sampling physical points also sampled by other agents act as correlators between the local decision making procedures of each decoupled agent. This is similar to correlated equilibriums in game theory. However, finding correlated equilibriums is not easy in the general case.

Instead, inspired by equations (4.14)-(4.16), we suggest a decision making heuristic for any agent k. Figure 4.8 presents the heuristic. At every decision making point, agent k calculates probabilities $prob_{j,l}$ that another agent l uses its action j. The probability is estimated as the number of useful points for action j (e.g., points for which the utility gain is positive because they were not sampled yet) over the total number of useful points for all available actions. Then, for all actions i of agent k, the method considers every physical point $< x, y >$ that can be sampled through the action, and computes the value S that indicates the overall probability that another agent l re-samples point S (equation (4.18)). The likelihood that an agent l re-samples the same point $< x, y >$ depends not only on the probability $prob_{i,k}$ that it reaches the point through action i but also on the probability that the agent has sufficient flexibility to do so without violating its timing constraints. The smaller is flexibility $f_l^{(t)}$, the lesser is the likelihood of selecting any actions that would deviate from the minimum path to the agent's destination. Flexibility $f_l^{(t)}$ is estimated as $\frac{\sum_{A,B \in minPath} \frac{t_B^{min} - t_A^{max}}{t_B^{max} - t_A^{min}}}{N-1}$. N is the number of points on the minimum path $minPath$. Points A and B are successive points on the minimum path. Finally, the number of useful points that are uniquely sampled by action i of agent k is approximated as the difference between the total number of useful points for the action, $Npoints_k$, minus those also re-sampled by other agents, as predicted by sum S. The highest probability action is executed (equation (4.14)).

4.4 Experimental Analysis

The experiments compared distributed data acquisition using the proposed decision making scheme in Section 4.3 and the methods using the reference utilities in Section 4.2. The goal was to maximize the area coverage of the sampled data while visiting a number of required points placed over the area of interest. All newly sampled points have utility one, and all re-sampled points have utility zero.

The experimental set-up run considered examples in which the following parameters were changed: (i) the number of mobile agents was 5, 10, 20, 50 and 100 autonomous agents, (ii) the number and position of required points, some of which are shown in Figure 4.9, and (iii) the length of the sampling path between consecutive decision making point, which was set as a random number of Gaussian distribution.

The resulting sampling results show that the proposed decentralized method produces a repulsion effect so that different agents avoid overlapping of their sampled points. This lowers the redundancy of sampled data. In certain instances, the overall sampling coverage of the area is improved as compared to the five traditional methods. Tables 4.2-4.10 present more detailed experimental results to compare the five methods based on utility maximization to the proposed model-based, game-theoretic approach. For each case, the experiments recorded 10 different runs of the acquisition scheme. This is required given the stochastic nature of the method. The five methods were presented in Section 4.2 and are summarized as following:

Method 1 Utility function is expressed as $U = g(x)$.

$g(x)$ is information gain, which is expressed as coverage gain in the experiments. The best candidate is selected as it maximizes U.

Method 2 Utility function is $U = g(x) - \beta \times c(x)$.

$g(x)$ is the data gain at point x, $c(x)$ is the cost to travel and β is the coefficient indicating the weight of gain over cost. The difference from method 1 is that cost is put into utility function calculation. β can be adjusted to change weight of cost. The best candidate is selected as it maximizes U.

Method 3 Utility function is $U = h(x) - \beta \times c(x)$.

It is proposed in [73], which also uses a linear form utility function but adopts entropy $h(x)$ at point x as measurement of information gain. The best candidate is selected as it maximizes U.

Method 4 Utility function is expressed as $U = A(x) \times \exp(-\lambda \times L(x))$.

$A(x)$ measures unvisited area visible from point x, $L(x)$ is the length of path to point x, and λ is the weight. The best candidate is selected as it maximizes U.

Method 5 is multi-objective exploration strategy proposed in [68].

Step 1: Select the following Pareto-optimal from all the candidates: maximum information gain, minimum travel cost, and minimum overlapping. Step 2: Select the one from Pareto-optimal as the best candidate which is nearest to the ideal position according to the distance calculation:

$D(p) = \sqrt[2]{(c(p) - c_M)^2 + (i(p) - i_M)^2 + (o(p) - o_M)^2}$

Method 6 is the game-theoretic heuristic approach proposed in Section 4.3.

Experiments with 5, 10, 20, 50 and 100 autonomous agents were studied. Each agent does its own decision making and does not rely on synchronized information. Figure 4.9 shows the examples of the pre-calculated trajectories we used in our experiments. Note that the nature of the preset trajectories affects the data acquisition results since the exploration paths are based on them. If the trajectories of agents are too far apart, no overlapping will happen no matter what exploration strategies are used. On the other hand, if too many agents are sampling a certain area, overlapping is unavoidable. The proposed game-theoretic approach is efficient in the situations that overlapping happens and can be avoided in some cases which require the preset trajectories of agents neither too close nor too far away from each other.

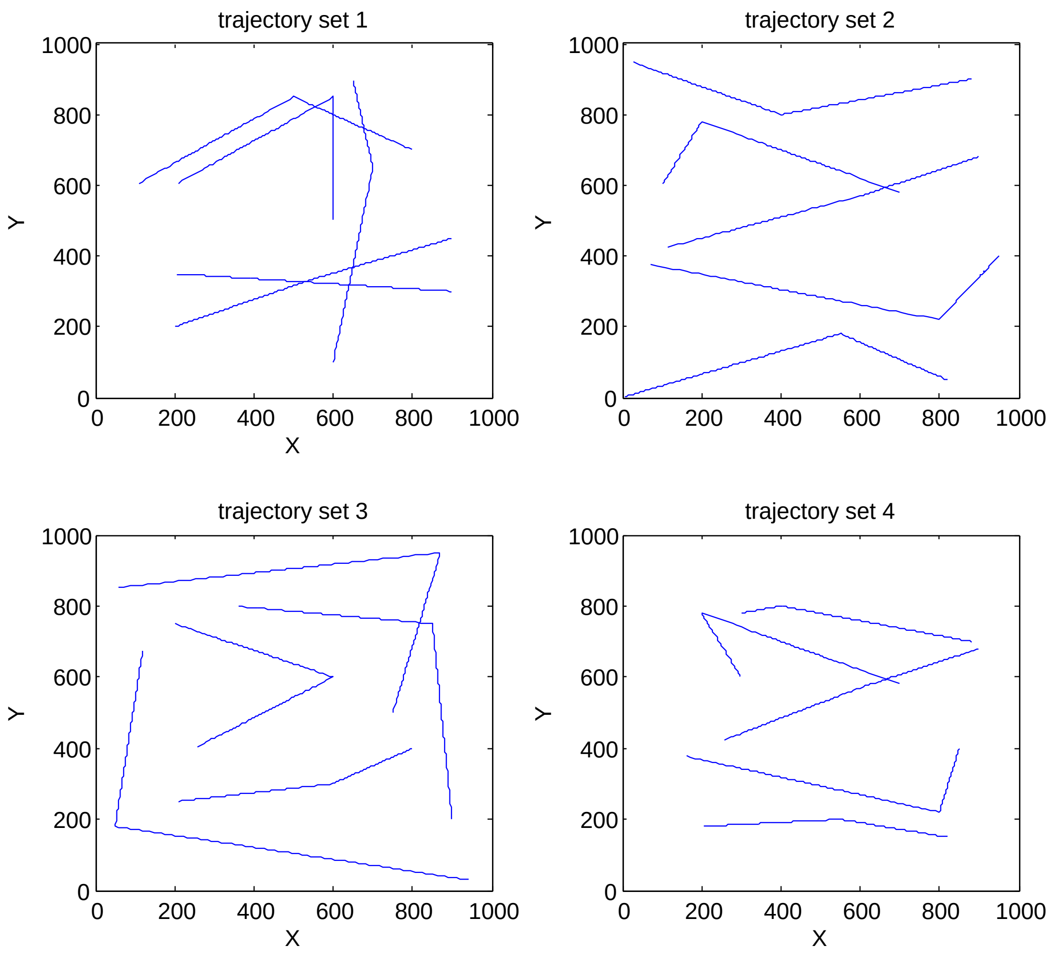


Figure 4.9: Example of the experimented trajectories

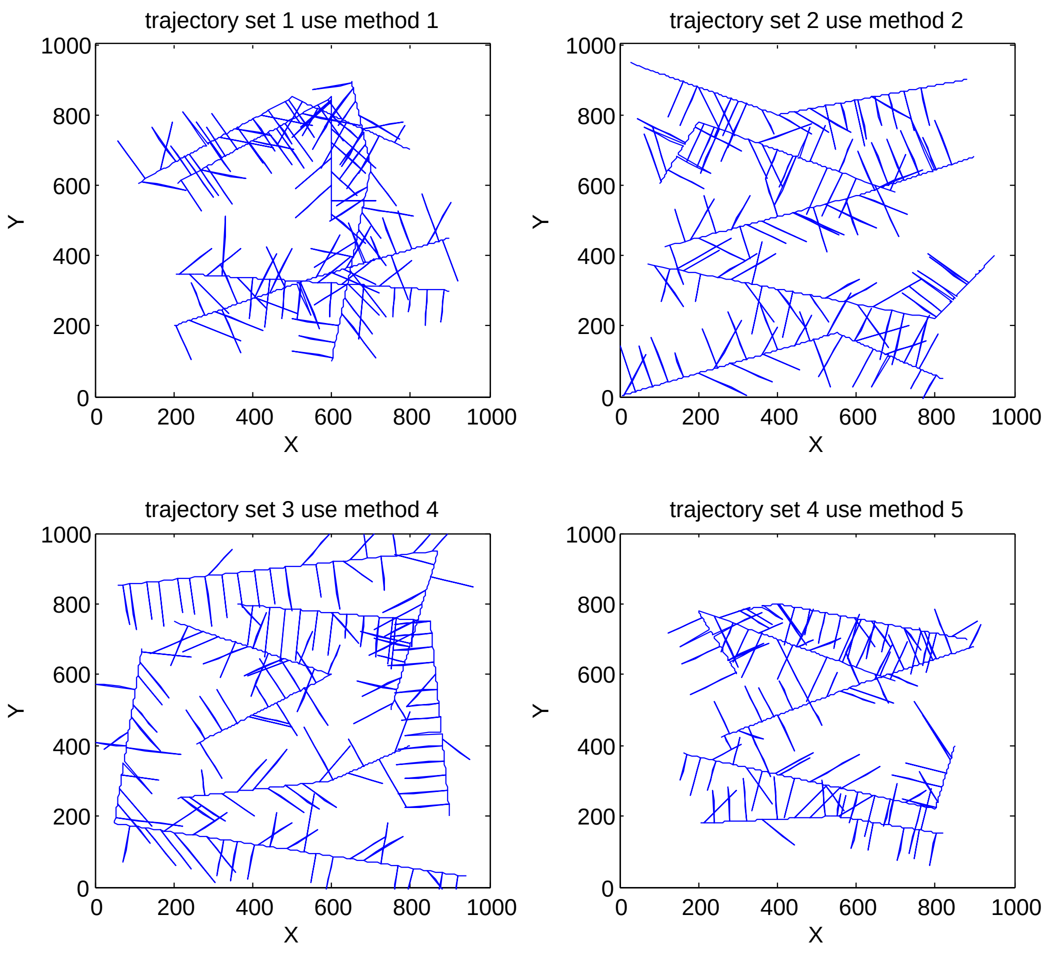


Figure 4.10: Example trajectories produced using method 1, 2, 4 and 5. The shorter lines surrounding the main trajectories indicate exploration conducted through decision making methods.

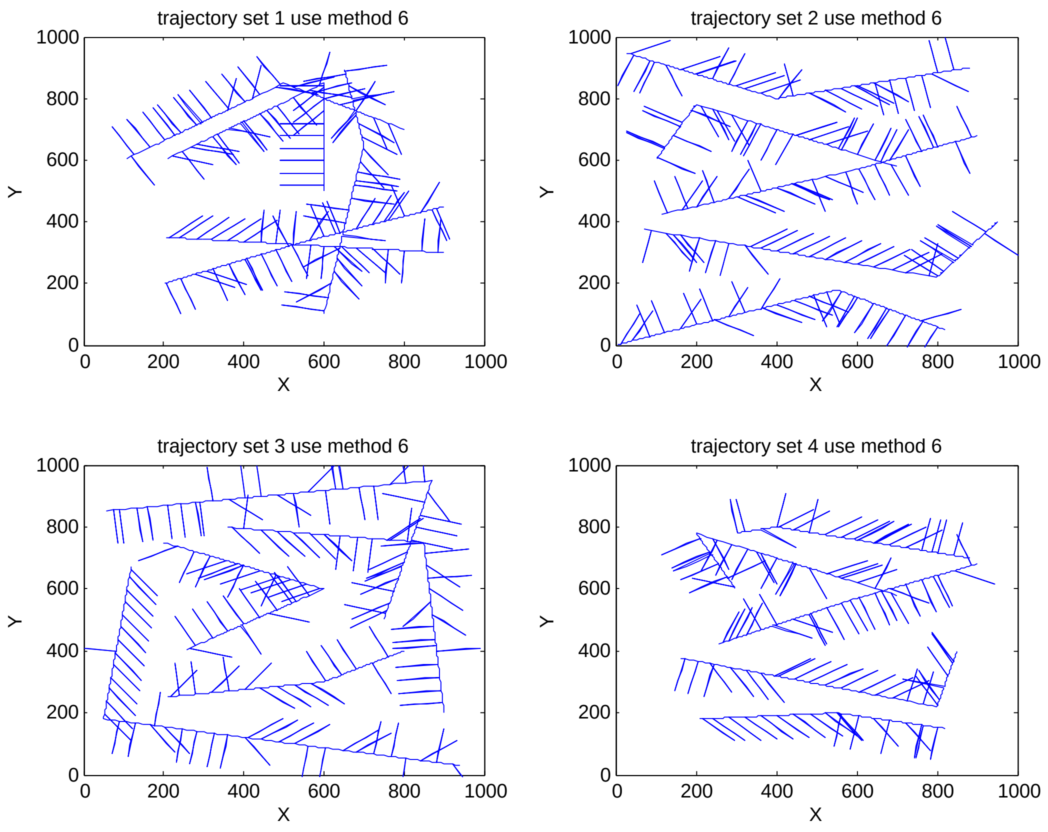


Figure 4.11: Example trajectories produced using method 6. The shorter lines surrounding the main trajectories indicate exploration conducted through the proposed game-theoretic decision making methods.

Four different sets of trajectories were used in the experiments with 5 agents. Since the results of the experiments focused on data coverage and redundancy, the 4 sets of main trajectories for the 5 agents considered different distribution in space, while still deviating to sample around the trajectories. These trajectories are the main trajectories that each agent follows during their tasks. The agents do optimal exploration at each decision making point and return to their own main trajectories and continue the tasks. Figures 4.10 and 4.11 show the results of decision making from related work and proposed method respectively. Figure 4.10 shows some examples of the exploration paths computed with method 1, 2, 4, 5 of related work. The figure for method 3 is not show since the nature of method 3 is similar to method 2 but it uses a different expression of information gain in calculation. Although each agent optimizes its own coverage at each decision making point using different utility functions, overlapping among different agents is obviously high in the final result. Figure 4.11 shows the same set of examples as Figure 4.10, but the exploration paths are computed using game-theoretic heuristic method discussed in Section 4.3. Comparing each set of final trajectories with those in Figure 4.10, a repulsion effect can be seen between the trajectories of different agents so that the overlapping among agents is reduced, and so is the data redundancy. At the same time, the coverage of the whole area is improved compared to the plots in Figure 4.10.

Tables 4.2 to 4.10 represent the statistical results of the experiments. The rows of Tables 4.2-4.5 show the following experimental results: (1) The first row gives the average number of multiple samples of a physical point due to an agent's own decisions, not the required points. (2) The second row shows the average number of samples only for multiple-sampled points, and due to both decisions and required points. (3) Row three gives the average number of samples for all points, including those sampled only once. It is the ratio of the total number of samples and the total number of distinct physical points that were sampled. (4) Row four presents the percentage of the sampled area defined as the ratio of the total number of visited points and the total number of points of target area. (5) Row five shows the number of physical points which were sampled multiple times but due to repeated rounds of exploration. This metric considers only the points that were re-sampled during a new round of exploration through the same agents. (6) Row six gives the total travel time of all agents. (7) Row seven describes the percentage of points sampled multiple times. (8) Row eight offers the percentage of unsampled points. (9) Row nine indicates the number of multiple-sampled points of a region. (10) Row ten shows the maximum number of sampling times of a point (maximum amount of redundancy).

Exp. Mlt. Vst.: Average number of multiple samples of a physical point situated on the exploration branches (only the visits on the branch count) due to agents decisions, $\frac{Total\ multiple\ visited\ times\ on\ the\ branches}{No.\ of\ points\ been\ multiple\ visited\ onthe\ branches}$, in which $multiple\ visited$ means visited more than once by different agents.

Mlt. Vst.: Average number of samples counted only for multiple-sampled points (the visits on the branch and main trajectories count), $\frac{Total\ multiple\ visited\ times}{No.\ of\ points\ been\ multiple\ visited}$.

Average Vst.: Average number of samples for all points, including those sampled only once, $\frac{Total\ visited\ times}{No.\ of\ points\ been\ visited}$.

Cover: The percentage of the sampled area defined as the ratio of the total number of visited points and the total number of points of target area, $\frac{Total\ No.\ of\ visited\ ponits}{No.\ of\ points\ intarget\ area}$.

Exp. Mlt. No.: Number of physical points which were sampled multiple times due to exploration by different agents.

Flight Time: Total travel time of all agents.

Mlt. Vst.: Percentage of points sampled multiple times.

Uncover: percentage of unsampled points, $(1 - cover)$.

Mlt. Vst. No.: Indicates the number of multiple-sampled points of a region.

Max Sample: Maximum number of sampling times of a point (maximum amount of redundancy).

Table 4.2 represents the data acquisition results of trajectory set 1 as in the top-left small figure in Figure 4.9. Attribute 'Exp. Mlt. Vst.', multiple visited times per point, is greater than 2. Since metric 'Max Sample' is 3, 'Exp. Mlt. Vst.' must be between 2 and 3. Method 6 samples on the average by about 10% less redundant data. On the overall, considering also the points that are sampled only one time, Method 6 samples on the average by 20% less times points (row three). The overall multiple-visited times, attribute 'Mlt. Vst.', is affected by the distribution pattern of the main trajectories more than the exploration strategy itself. Metric 'Average Vst.' shows the repeated visit times. This is between 1 and 'Max Sample', which is 3. Attribute 'Average Vst. No.' of methods 1 to 5 is between 1.41 and 1.45, and is less than 1.11 for method 6, which is more than 20% less. The coverage result of Method 6 is also good compared to the other methods (about 15% more coverage). The absolute number of coverage percentile is not very high because the main trajectories are not very dispersed in this set of tests and the exploration length is limited. Attribute 'Exp. Mlt. No.' counts the repeated visited times on the branches as generated by methods 1 to 6. It also represents the data redundancy due to different decision making methods. This indicates that the proposed method (method 6) decreases by 54% to 62% the number of multiply-visited points on the decision branches. This indicates again that less redundant information is acquired from the points. If the points on the main trajectories are also considered in the redundancy-related metrics (rows 7 and 9) then the improvement is between 66% to 70%. Attribute 'Uncover' is the opposite to attribute 'Cover'. Metric 'Max Sample' is the maximum visited times on the whole map.

Tables 4.3 to 4.5 show the same characteristics as the previous case (Table 4.2) except that the results are for trajectory sets 2 to 4. Trajectory set 2 is the top-right small figure in Figures 4.9 to 4.11. Compared to trajectory set 1, the distribution pattern of main trajectories is very dispersed. Attribute 'Max Sample' is 2 in all the six methods. Attribute

Table 4.2: Data acquisition results for 5 agents for trajectory set 1

	meth. 1	meth. 2	meth. 3	meth. 4	meth. 5	meth. 6	min imprv.	max imprv.
Exp. Mlt. Vst.	2.1799	2.2469	2.2334	2.1990	2.1814	2.0115	7.%	10.48%
Mlt. Vst.	2.1163	2.1567	2.1504	2.1338	2.1186	2.0072	5.16%	6.93%
Average Vst.	1.4288	1.4541	1.4407	1.4468	1.4113	1.1095	21.39%	23.70%
Cover	40.29	39.30	38.86	38.93	36.99	42.26	4.88%	14.23%
Exp. Mlt. No.	47.10	49.40	45.30	47.90	40.20	18.40	54.23%	62.75%
Flight Time	792.68	773.94	764.79	753.72	689.96	734.97	N.A.	N.A.
Mlt. Vst.	15.46	15.39	14.86	15.33	13.60	4.59	66.24%	70.29%
Uncover	59.71	60.70	61.14	61.07	63.01	57.74	3.30%	8.35%
Mlt. Vst. No.	96.60	96.20	92.90	95.80	85.00	28.70	66.24%	70.29%
Max Sample	3.00	3.00	3.10	3.10	3.00	2.20	26.67%	29.03%

Table 4.3: Data acquisition results for 5 agents for trajectory set 2

	meth. 1	meth. 2	meth. 3	meth. 4	meth. 5	meth. 6	min imprv.	max imprv.
Exp. Mlt. Vst.	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	0.00%	0.00%
Mlt. Vst.	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	0.00%	0.00%
Average Vst.	1.1393	1.1244	1.1350	1.1287	1.1078	1.0099	8.83%	11.36%
Cover	56.88	57.46	56.98	57.22	53.57	54.62	-4.93%	1.97%
Exp. Mlt. No.	28.20	26.10	28.20	27.80	18.70	2.40	87.17%	91.49%
Flight Time	924.80	894.24	892.53	888.38	807.09	881.52	N.A.	N.A.
Mlt. Vst.	7.90	7.14	7.68	7.36	5.76	0.54	90.56%	93.12%
Uncover	43.12	42.54	43.02	42.78	46.43	45.38	-6.66%	2.27%
Mlt. Vst. No.	49.40	44.60	48.00	46.00	36.00	3.40	90.56%	93.12%
Max Sample	2.00	2.00	2.00	2.00	2.00	2.00	0.00%	0.00%

‘Exp. Mlt. Vst.’ and ‘Mlt. Vst.’ are also 2. In this set of trajectories, the overlapping is less likely compared to trajectory set 1 since agents are relatively far away from each other and the limited length of exploration path eliminated some of the multiple visits. As a result, in this set of tests, redundancy among different agent is not a issue. Metric ‘Cover’ for method 6 does not show much improvement. The slight difference among ‘Cover’ is caused by the difference due to random exploration length generation. On the other hand, attribute ‘Exp. Mlt. No.’, ‘Mlt. Vst.’ and ‘Mlt. Vst. No.’ for method 6 are small in this case, which indicates that repeated visits are almost eliminated for very dispersed trajectories sets.

Trajectory set 3 and 4 show the advantages of method 6 in reducing redundancy. Improvement is between 76% and 93%. For example, as shown in Table 4.5, method 6 reduces redundant sampling by about 90%. If the preset trajectories are more dispersed, like for trajectory two (top-right sub-figure in Figure 4.11), the amount of potential sampling overlapping is less since the mobile agents are well separated from each other. The coverage percentages are similar to the values for trajectory one. The travel time (row six) is shorter in most cases by up to about 10%.

Figure 4.12 presents a graphical representation of the following performance attributes of the six methods: cover (top-left) (row 4), number of redundant samplings (bottom-left) (row 5), percentage of points visited multiple times (row 7), and maximum number of samples per point (bottom-right) (row 10). The second and third plots show that method 6 produces significant savings in redundant sampling.

Tables 4.7-4.10 indicate that the above conclusions are valid for groups of 10, 20, 50 and 100 agents. The target areas (the region in which agents move) in the experiments are expanded in proportion to the number of agents. The purpose is to show the results of increased number of agents and eliminate the effect of other factors. Table 4.6 describes the number of agents and corresponding size of target area used in the experiments.

Figures 4.13-4.16 present a graphical representation of the following performance attributes for the six methods for 5, 10, 20, 50 and 100 agents: cover (row 4), number

Table 4.4: Data acquisition results for 5 agents for trajectory set 3

	meth. 1	meth. 2	meth. 3	meth. 4	meth. 5	meth. 6	min imprv.	max imprv.
Exp. Mlt. Vst.	2.0031	2.0291	2.0309	2.0105	2.0056	2.0000	0.16%	1.52%
Mlt. Vst.	2.0020	2.0174	2.0203	2.0057	2.0036	2.0000	0.10%	1.00%
Average Vst.	1.1167	1.1028	1.1172	1.1044	1.0838	1.0188	5.99%	8.81%
Cover	62.13	61.36	60.75	61.26	57.36	59.46	-4.30%	3.65%
Exp. Mlt. No.	28.00	23.10	28.30	24.50	15.90	4.60	71.07%	83.75%
Flight Time	997.56	956.95	968.25	952.83	857.27	951.60	N.A.	N.A.
Mlt. Vst.	7.23	6.21	6.98	6.37	4.78	1.12	76.59%	84.51%
Uncover	37.87	38.64	39.25	38.74	42.64	40.54	-7.06%	4.92%
Mlt. Vst. No.	45.20	38.80	43.60	39.80	29.90	7.00	76.59%	84.51%
Max Sample	2.10	2.40	2.60	2.10	2.10	2.00	4.76%	23.08%

Table 4.5: Data acquisition results for 5 agents for trajectory set 4

	meth. 1	meth. 2	meth. 3	meth. 4	meth. 5	meth. 6	min imprv.	max imprv.
Exp. Mlt. Vst.	2.0976	2.0541	2.0696	2.0647	2.0729	2.0000	2.63%	4.65%
Mlt. Vst.	2.0544	2.0301	2.0383	2.0325	2.0373	2.0000	1.48%	2.65%
Average Vst.	1.2592	1.2425	1.2511	1.2370	1.2323	1.0173	17.45%	19.21%
Cover	41.65	42.54	41.92	42.66	39.62	42.62	-0.08%	7.59%
Exp. Mlt. No.	36.00	34.30	34.00	31.10	27.70	3.60	87.00%	90.00%
Flight Time	753.62	732.74	729.72	718.01	654.65	715.99	N.A.	N.A.
Mlt. Vst.	10.22	9.97	10.11	9.78	8.83	0.74	91.67%	92.80%
Uncover	58.35	57.46	58.08	57.34	60.38	57.38	-0.06%	4.98%
Mlt. Vst. No.	63.90	62.30	63.20	61.10	55.20	4.60	91.67%	92.80%
Max Sample	2.90	2.80	2.90	2.90	2.80	2.00	28.57%	31.03%

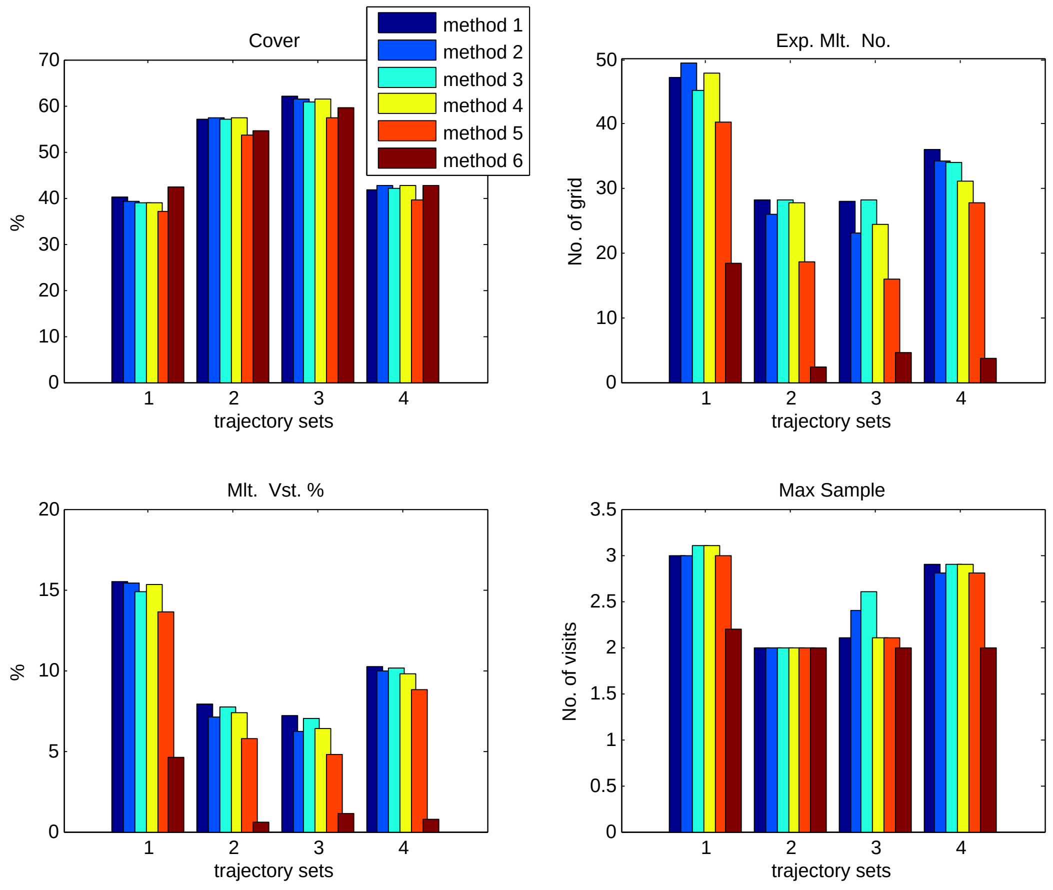


Figure 4.12: Comparison of methods 1 to 6 for 5 agents

Table 4.6: Size of target area for different number of agents

No. of agents	Size of target area
5	1000x1000
10	1400x1400
20	2000x2000
50	3000x3000
100	4480x4480

Table 4.7: Data acquisition results for 10 agents

	meth. 1	meth. 2	meth. 3	meth. 4	meth. 5	meth. 6	min imprv.	max imprv.
Exp. Mlt. Vst.	2.1790	2.1693	2.1781	2.1639	2.1250	2.0141	5.22%	7.57%
Mlt. Vst.	2.1157	2.1171	2.1174	2.1089	2.0804	2.0093	3.42%	5.11%
Average Vst.	1.2720	1.2836	1.2712	1.2678	1.2449	1.0544	15.31%	17.86%
Cover	47.31	46.23	46.17	46.59	44.20	47.61	0.64%	7.72%
Exp. Mlt. No.	74.20	75.70	70.60	73.00	60.00	20.10	66.50%	73.45%
Flight Time	1647.02	1592.22	1590.96	1577.46	1443.54	1563.48	N.A.	N.A.
Mlt. Vst.	11.53	11.74	11.19	11.25	10.02	2.56	74.43%	78.16%
Uncover	52.69	53.77	53.83	53.41	55.80	52.39	0.57%	6.11%
Mlt. Vst. No.	141.30	143.80	137.10	137.80	122.80	31.40	74.43%	78.16%
Max Sample	3.00	3.10	3.00	3.00	3.00	2.20	26.67%	29.03%

Table 4.8: Data acquisition results for 20 agents

	meth. 1	meth. 2	meth. 3	meth. 4	meth. 5	meth. 6	min imprv.	max imprv.
Exp. Mlt. Vst.	2.1660	2.1544	2.1756	2.1633	2.1587	2.0354	5.52%	6.44%
Mlt. Vst.	2.1165	2.1085	2.1191	2.1138	2.1068	2.0350	3.41%	3.97%
Average Vst.	1.2973	1.3027	1.3017	1.3002	1.2623	1.1111	11.98%	14.71%
Cover	56.60	56.24	56.33	56.17	53.02	55.47	-2.00%	4.62%
Exp. Mlt. No.	203.60	204.80	201.90	206.90	159.30	90.70	43.06%	56.16%
Flight Time	4151.32	4011.78	4020.32	3970.05	3624.73	3870.40	N.A.	N.A.
Mlt. Vst.	15.07	15.36	15.19	15.14	12.57	5.96	52.61%	61.22%
Uncover	43.40	43.76	43.67	43.83	46.98	44.53	-2.61%	5.21%
Mlt. Vst. No.	376.70	384.00	379.70	378.40	314.20	148.90	52.61%	61.22%
Max Sample	3.10	3.30	3.20	3.20	3.20	3.00	3.23%	9.09%

Table 4.9: Data acquisition results for 50 agents

	meth. 1	meth. 2	meth. 3	meth. 4	meth. 5	meth. 6	min imprv.	max imprv.
Exp. Mlt. Vst.	2.2328	2.2237	2.2056	2.2169	2.1991	2.0380	7.33%	8.73%
Mlt. Vst.	2.1559	2.1556	2.1468	2.1519	2.1370	2.0370	4.68%	5.52%
Average Vst.	1.3428	1.3383	1.3400	1.3353	1.3060	1.1190	14.32%	16.67%
Cover	55.15	55.19	54.82	54.69	51.71	55.25	0.10%	6.84%
Exp. Mlt. No.	494.80	489.00	496.30	480.40	386.30	207.30	46.34%	58.23%
Flight Time	9489.44	9201.79	9218.34	9052.13	8304.64	8850.93	N.A.	N.A.
Mlt. Vst.	16.36	16.16	16.25	15.92	13.91	6.34	54.45%	61.25%
Uncover	44.85	44.81	45.18	45.31	48.29	44.75	0.13%	7.33%
Mlt. Vst. No.	920.00	908.90	914.30	895.30	782.60	356.50	54.45%	61.25%
Max Sample	4.00	4.00	4.00	4.00	4.00	3.00	25.00%	25.00%

Table 4.10: Data acquisition results for 100 agents

	meth. 1	meth. 2	meth. 3	meth. 4	meth. 5	meth. 6	min imprv.	max imprv.
Exp. Mlt. Vst.	2.5115	2.4961	2.4811	2.4853	2.4524	2.1828	11.00%	13.09%
Mlt. Vst.	2.3469	2.3393	2.3302	2.3348	2.3058	2.1464	6.92%	8.54%
Average Vst.	1.4307	1.4247	1.4233	1.4203	1.3868	1.1775	15.09%	17.70%
Cover	45.67	45.50	45.74	45.47	43.14	46.05	0.67%	6.74%
Exp. Mlt. No.	1065.30	1050.30	1060.10	1043.00	873.50	501.70	42.56%	52.91%
Flight Time	18653.45	18052.39	18091.79	17835.89	16308.40	17215.78	N.A.	N.A.
Mlt. Vst.	14.60	14.43	14.55	14.31	12.78	7.13	44.20%	51.18%
Uncover	54.33	54.50	54.26	54.53	56.86	53.95	0.56%	5.11%
Mlt. Vst. No.	1832.00	1809.50	1825.70	1795.60	1602.80	894.30	44.20%	51.18%
Max Sample	6.40	6.40	6.60	6.40	6.10	4.50	26.23%	31.82%

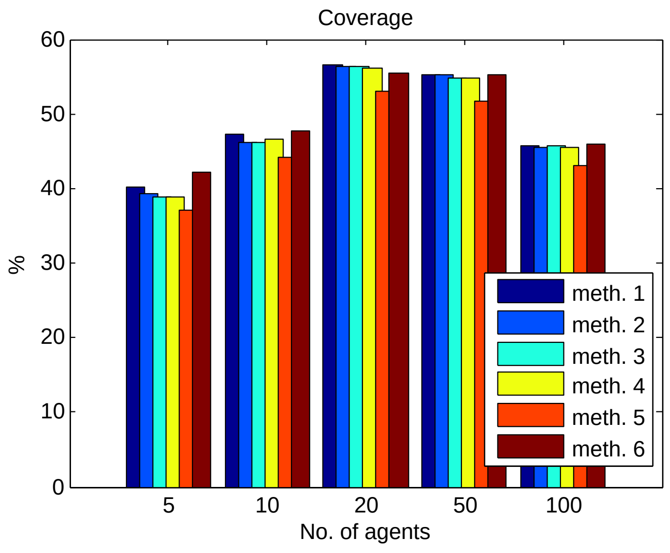


Figure 4.13: Coverage comparison for methods 1 to 6 for 5, 10, 20, 50 and 100 agents

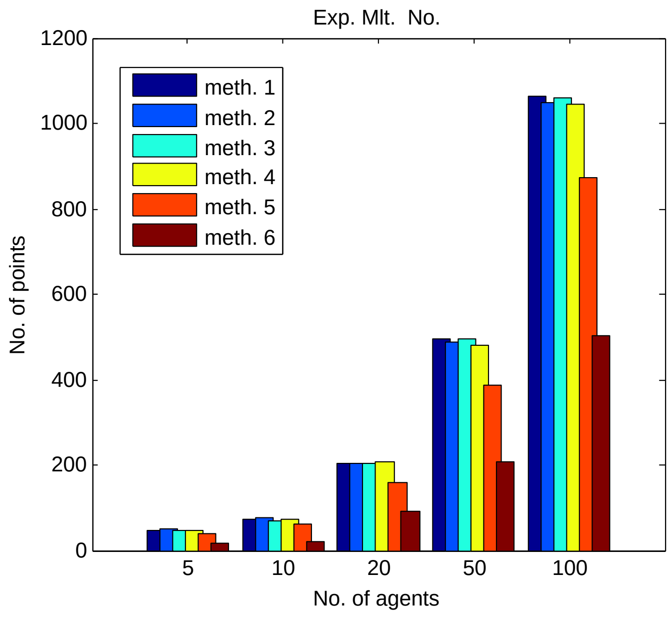


Figure 4.14: Attribute 'Exp. Mlt. No.' comparison for methods 1 to 6 and 5, 10, 20, 50 and 100 agents

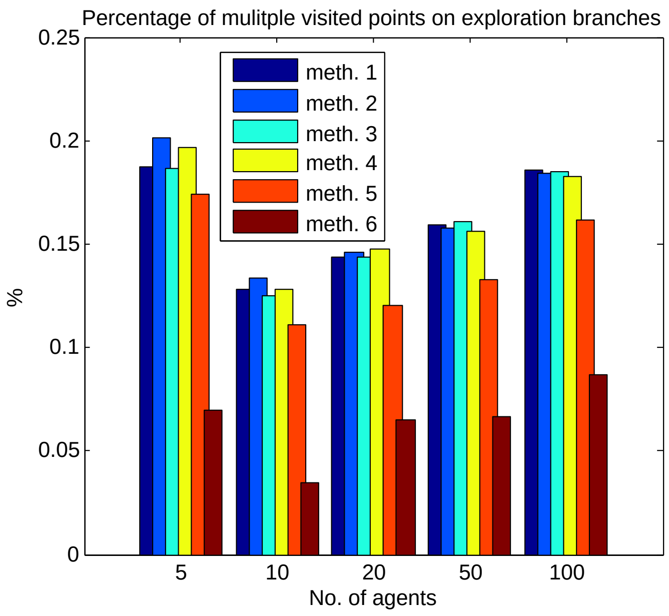


Figure 4.15: Multiple sample comparison for methods 1 to 6 and 5, 10, 20, 50 and 100 agents

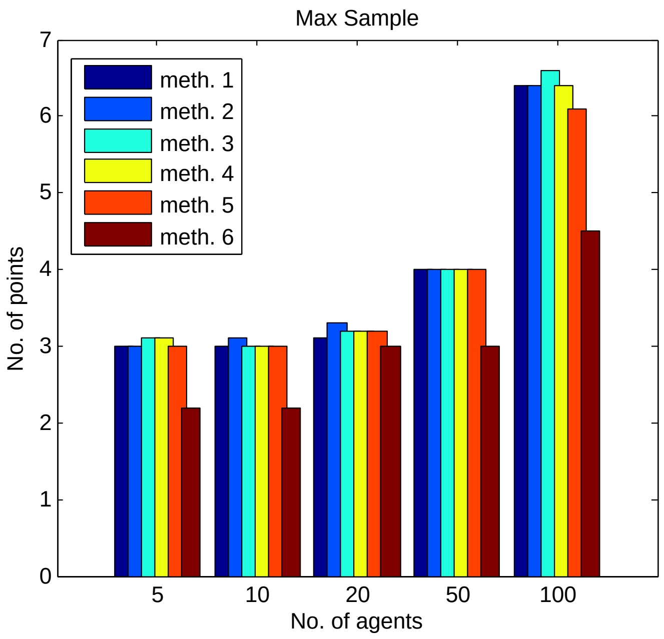


Figure 4.16: Attribute 'Max Sample' comparison of methods 1 to 6 and 5, 10, 20, 50 and 100 agents

of redundant samplings (row 5), percentage of points visited multiple times (row 7), and maximum number of samples per point(row 10). The proposed method shows competitive coverage in each case and dramatically reduced sample redundancy (Figures 4.14 and 4.15). Figure 4.16 indicates that the proposed method has less maximum sample time than the rest.

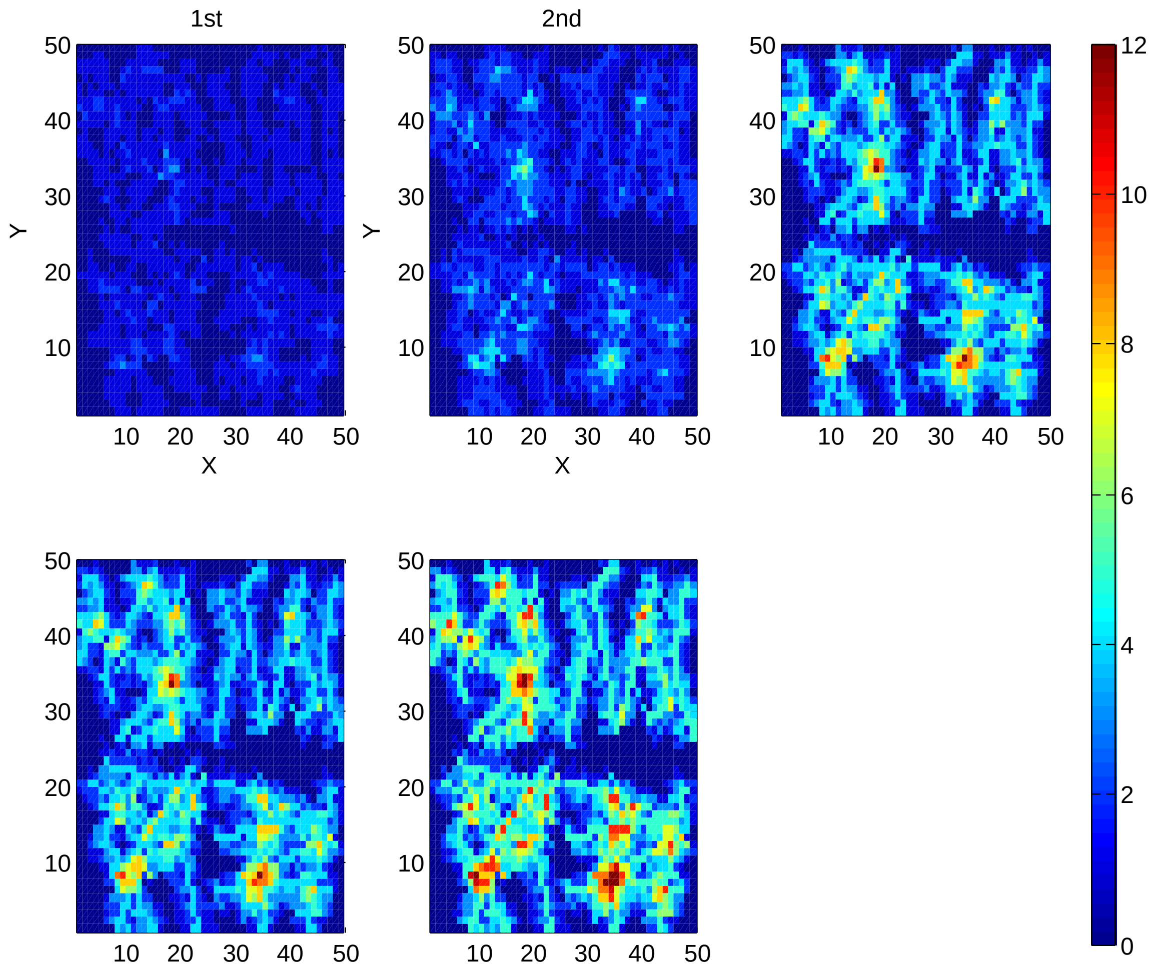


Figure 4.17: Data coverage and redundancy after each iteration for method 6 with 20 agents (carrying information of previous iterations). Blue areas are sampled less. Red areas are sampled more often.

Figures 4.17-4.28 present the data coverage and redundancy of another group of experiments. In this group of experiments, the proposed algorithm (method 6) and methods 1 to 5 run five iterations. Results of data coverage and redundancy are collected after each iteration. Two different conditions are analyzed in the experiments: data from previous

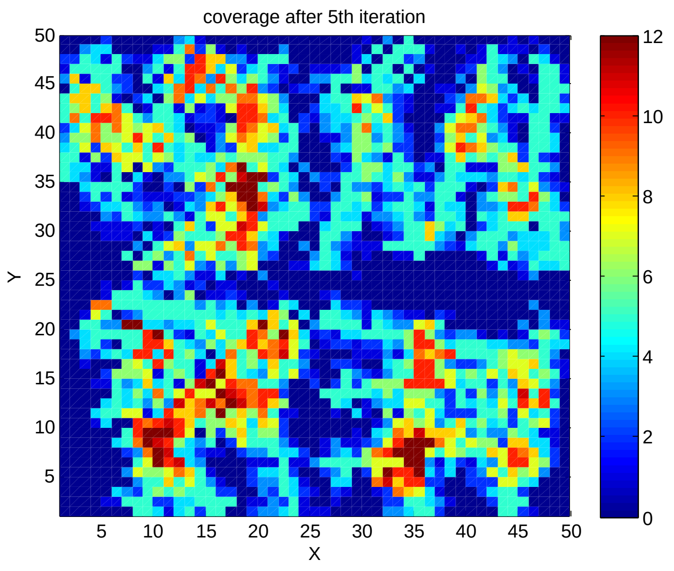


Figure 4.18: Data coverage and redundancy after 5th iteration for method 5 with 20 agents (carrying information of previous iterations). Blue areas are sampled less. Red areas are sampled more often.

iterations are carried over, or data from previous iterations are not carried over from one iteration to the next. In the first situation, the agents try to avoid previously sampled points when they do decision making. This group of experiments used 20, 50 and 100 agents. Figure 4.17 represents the data coverage and redundancy results after each iteration with up to 5 iterations for method 6 with 20 agents, who carried over information from previous iterations. Different color shows different times of data sampled after each iteration. The color changes from blue to red with the number of data sampled increasing from 0 to 12 and numbers larger than 12 is also displayed as red. Figure 4.17 shows the coverage change after repeatedly running the algorithm.

Figure 4.18 shows the results of similar experiments using method 5 mentioned previously, which is the best one among methods 1-5 according to Table 4.8. Both Figures 4.17 and 4.18 represent accumulated effect of repeatedly running respective strategies. The figures with results after 5th iteration show that method 6 generated less data redundancy than method 5. For example, the number of points sampled 12 times (maximum sampled time in this case) after 5 iteration using method 6 is 8 ,and the number of points sampled

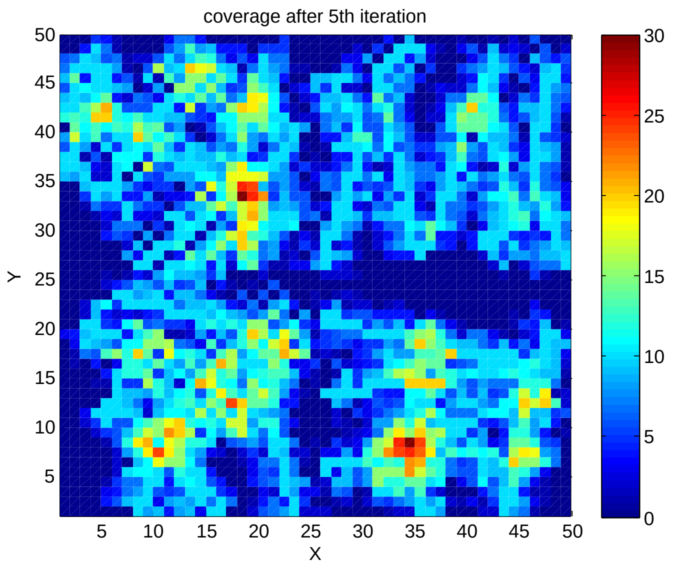


Figure 4.19: Data coverage and redundancy after 5th iteration for method 6 with 20 agents (without information of previous iterations). Blue areas are sampled less. Red areas are sampled more often.

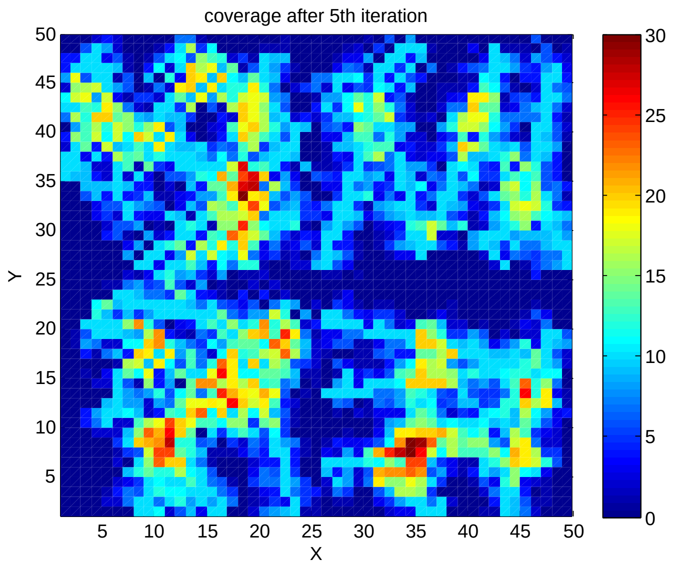


Figure 4.20: Data coverage and redundancy after 5th iteration for method 5 with 20 agents (without information of previous iterations). Blue areas are sampled less. Red areas are sampled more often.

more than 12 (including 12) times is 35 using method 5, which is a 77% improvement. The coverage after 5 iteration using method 6 is 76% versus 71% using method 5. The experiment did not go beyond 5 iterations because the number of choices at each decision making point is 4, and more than 5 iterations will generate data redundancy no matter which strategy is used.

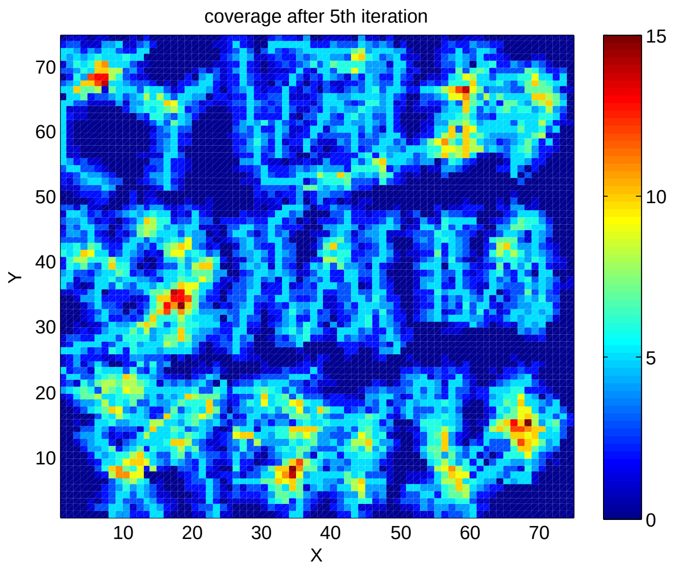


Figure 4.21: Data coverage and redundancy after 5th iteration for method 6 with 50 agents (carrying information of previous iterations). Blue areas are sampled less. Red areas are sampled more often.

Figures 4.19 and 4.20 shows the results of similar experiments with different conditions: data from previous iterations are not carried over. This is the repeated running of strategies in methods 5 and 6. The tone of Figure 4.20 is warmer than that in Figure 4.19, which indicates that data redundancy accrued less often when using method 6. Also, the coverage after 5 iterations using method 6 is 78% versus 76% using method 5.

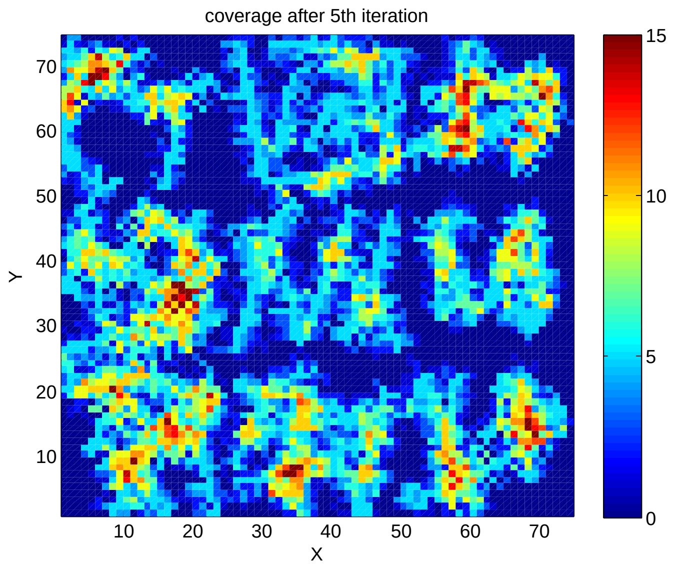


Figure 4.22: Data coverage and redundancy after 5th iteration for method 5 with 50 agents (carrying information of previous iterations). Blue areas are sampled less. Red areas are sampled more often.

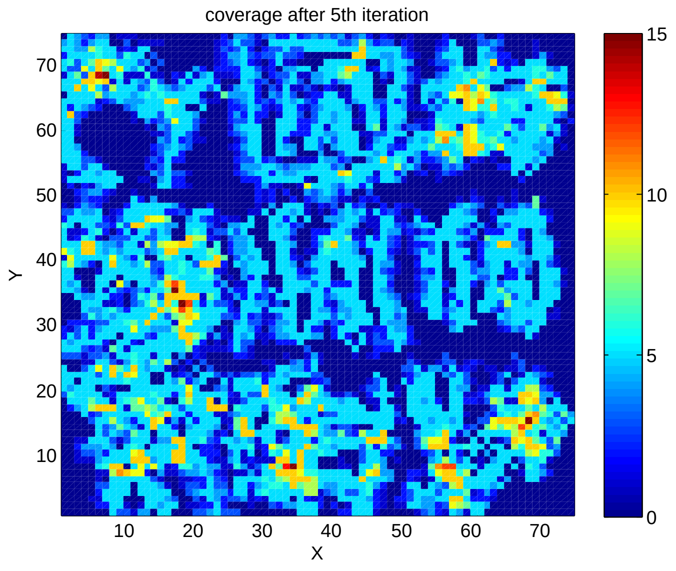


Figure 4.23: Data coverage and redundancy after 5th iteration for method 6 with 50 agents (without information of previous iterations)

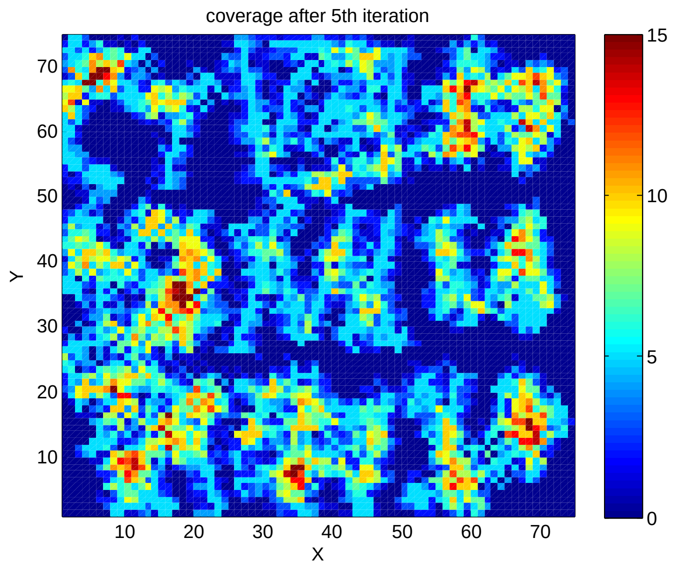


Figure 4.24: Data coverage and redundancy after 5th iteration for method 5 with 50 agents(without information of previous iterations). Blue areas are sampled less. Red areas are sampled more often.

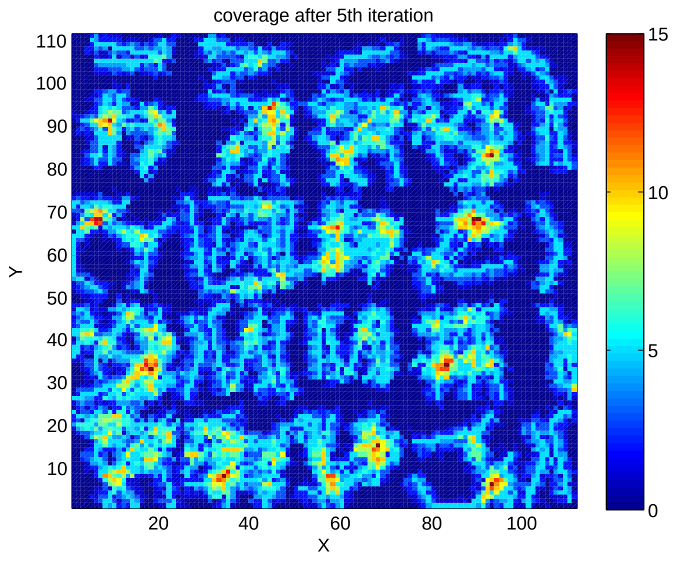


Figure 4.25: Data coverage and redundancy after 5th iteration for method 6 with 100 agents (carrying information of previous iterations). Blue areas are sampled less. Red areas are sampled more often.

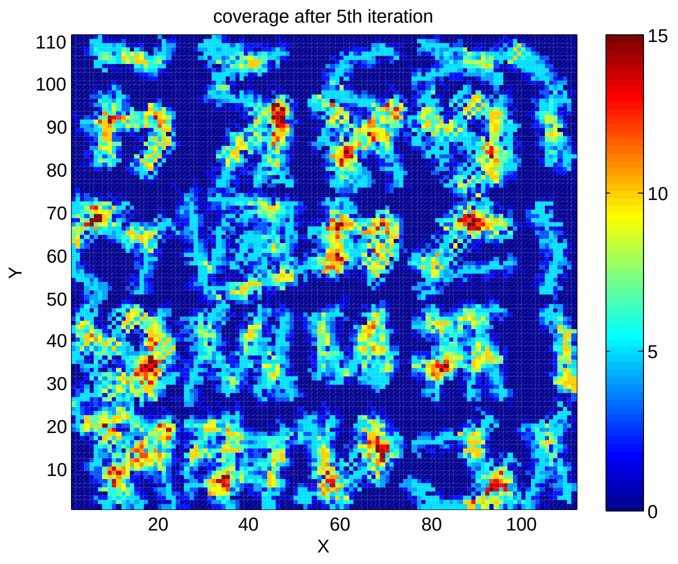


Figure 4.26: Data coverage and redundancy after 5th iteration for method 5 with 100 agents (carrying information of previous iterations). Blue areas are sampled less. Red areas are sampled more often.

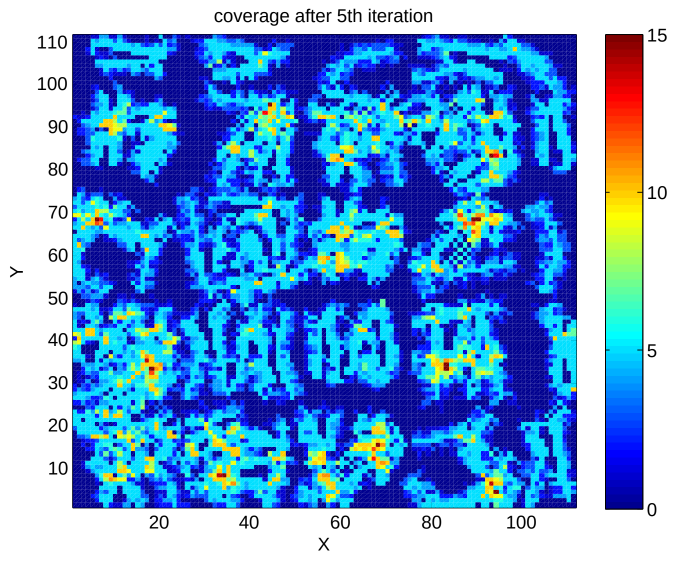


Figure 4.27: Data coverage and redundancy after 5th iteration for method 6 with 100 agents (without information of previous iterations). Blue areas are sampled less. Red areas are sampled more often.

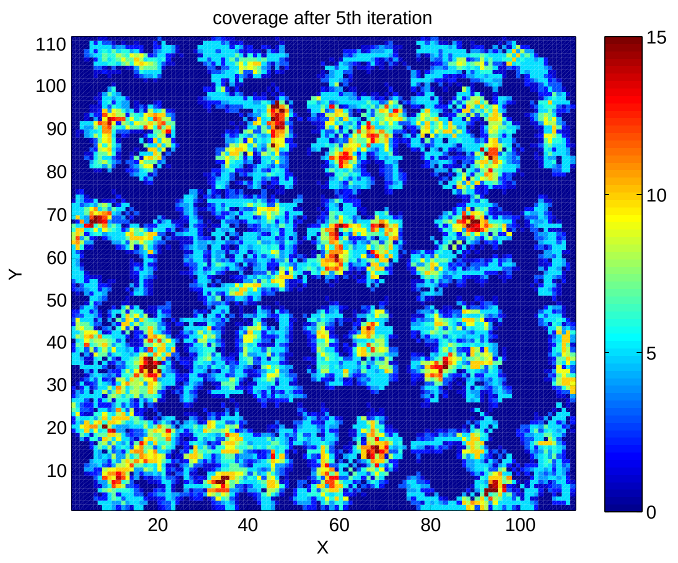


Figure 4.28: Data coverage and redundancy after 5th iteration for method 5 with 100 agents (without information of previous iterations). Blue areas are sampled less. Red areas are sampled more often.

Figures 4.21 - 4.24 show the results of experiments done in the above way, but with 50 agents. Figures 4.25 - 4.28 are with 100 agents. They all indicate similar conclusion as the cases with 20 agents. For example, comparing Figure 4.21 and Figure 4.22, the first one (method 6) has less samples on a single point and the average samples is 3.9 versus 4.9 using method 5. The coverage after 5 iterations using method 6 is 74% versus 68% using method 5.

4.5 Conclusion

CPS are expected to continuously provide optimized decisions based on accurate monitoring of environmental conditions. An important challenge is data acquisition through mobile agents to construct accurate models for decision making. This chapter presents a novel decentralized method for allocating mobile agents to sample physical areas while sporadically communicating with each other. It adopts a novel, asynchronous interaction scheme between agents to optimize the utility of the acquired data. The proposed game-theoretic heuristic suggests a method that decides the best candidate in each sampling step, while agents operate decoupled from each other. Experiments show that the proposed scheme can acquire data from interested areas and is able to get competitive data coverage but dramatically less data sampling redundancy by more than 50% as compared to other techniques.

Chapter 5

Conclusions

This thesis presents model-based techniques for dependable decision making in groups of autonomous mobile agents. The goal is to provide flexible, scalable, and predictable decision making for massively distributed mobile embedded systems.

The thesis first proposes a goal-oriented description for CPS and dynamically-constructed models to make decisions during operation. Scalability of descriptions is realized by defining the nature of interactions that can occur among decision modules while leaving to the execution environment the task of optimally implementing these interactions. The notation defines the operation goals of each sub-system (e.g., the criteria to be maximized or minimized during operation) and the physical capabilities of a module to achieve a certain goal. Different interaction types are introduced depending on the way the subsystems influence each other's goals and capabilities. It also refers to two applications to illustrate the model and the related decision making steps.

The thesis also presents distributed control methods for large-scale groups of autonomous agents. It introduces the ideas of performance predictive collaborative control of mobile agents operating in environments with fixed targets,and summarizes the trajectory generation algorithm that is used to model and simulate the moving vehicles. An Integer Linear Programming based model is used to optimize collaboration to achieve maximum task accomplishment and flexibility. It also offers detailed experimental insight on the quality, scalability and computational complexity of the proposed method.

A novel asynchronous interaction scheme between agents is described to maximize the utility of the acquired data. An adaptive interaction scheme is presented and the algorithm for dependable decision making strategy, originating from game theory, is provided. Experiments study the effectiveness of the scheme in comprehensive data acquisition while minimizing redundant data collection. The experimental results show that the proposed scheme reduced data redundancy significantly and achieved competitive or better data coverage than five other methods from the literature.

Bibliography

[1] C. Batini, E. Bertini, M. Comerio, A. Maurino, G. Santucci, "Visual languages and quality evolution in multichannel adaptive information systems", *Journal of Visual Languages & Computing*, Vol. 18, Issue 5, 2007, pp. 513-522.

[2] A. Bakshi, V. Prasanna, J. Reich, D. Larner, "The Abstract Task Graph: A methodology for architecture independent programming of networked sensor systems", *Proceedings of End-to-End, Sense-and-respond Systems, Applications and Services*, Seattle, Washington, 2005, pp. 19-24.

[3] L. Camara, E. Jungert, "A visual query language for dynamic process applied to a scenario driven environment", *Journal of Visual Languages & Computing*, Vol. 18, Issue 3, 2007, pp. 315-338.

[4] R. Soma, A. Bakshi, V. Prasanna, "A semantic framework for integrated asset management in smart oilfields", *Proceedings of the IEEE Symposium on Cluster Computing and Grid*, Rio de Janeiro, Brazil, May 2007, pp. 119-126.

[5] K. Whitehouse, F. Zhao, J. Liu, "Semantic streams: a framework for declarative queries and automatic data interpretation", *Technical Report*, Microsoft Research, MSR-TR-2005-45, 2005.

[6] A. Bakshi, V. Prasanna, A. Ledeczi, "Milan: A model based integrated simulation framework for design of embedded systems", *ACM Sigplan Notices*, Snowbird, Utah, 2001, pp.82-93.

[7] K. Passino, ”Biomimicry for Optimization, Control, and Automation”, *Springer*, 2005.

[8] E. Lee, ”The problem with threads”, *IEEE Computer*, Vol. 39, Issue 5, 2006, pp. 33-42.

[9] H. Sutter, J. Larus, ”Software and the concurrency revolution”, *ACM Queue*, Vol. 3, Issue 7, 2005, pp. 54-62.

[10] N. Zeldovich, A. Yip, F. Dabek, R. Morris, D. Mazieres, F. Kaashoek, ”Multiprocessor support for event driven programs”, *Proceedings of the USENIX Annual Technical Conference (USENIX 03)*, San Antonio Texas, June 2003.

[11] G. Papadopoulos, F. Arbab, ”Coordination models and languages, Advances in Computers - The Engineering of Large Systems”, *Academic Press*, Vol. 46, 1998, pp. 329-400.

[12] E. Lee, ”Cyber physical systems: Design challenges”, *Technical Report No. UCB/EECS-2008-8, University of California at Berkeley*, 2008.

[13] G. Papadopoulos, A. Stavrou, O. Papapetrou, ”An implementation framework for software architectures based on the coordination paradigm”, *Science of Computer Programming*, Vol. 60, Issue 1, 2006, pp. 27-67.

[14] S. Ahuja, N. Carrieri, D. Gelernter, ”Linda and friends”, *IEEE Computer*, Vol. 19, Issue 8, 1986, pp. 26-34.

[15] D. Gelernter, D. Kaminsky, ”Supercomputing out of the recycled garbage: Preliminary experience with Piranha”, *Proceedings of the ACM International Conference on Supercomputing*, Washington DC, 1992, pp. 417-427.

[16] A. Rowstron, A. Wood, ”Bonita: A set of tuple space primitives for distributed coordination”, *Proceedings of the International Conference on Systems Sciences*, Waikoloa, Hawaii, Vol. 1, 1997, pp. 379-388.

[17] N. Minsky, J. Leichter, "Law-governed Linda as a coordination model, in Object-Oriented Models and Languages for Concurrent Systems", *LNCS 924, Springer Verlag*, 1994, pp.125-145.

[18] T. Kielmann, "Designing a coordination model for open systems", *Proceedings of International Conference on Coordination Models, Languages and Applications, LNCS 1061, Springer Verlag*, 1996, pp. 267-284.

[19] J.P. Banatre, D. Le Metayer, "Gamma and the chemical reaction model: Ten years after, in Coordination Programming: Mechanisms, Models and Semantics", *World Scientific*, 1996, pp.1-39.

[20] I. Foster, "Compositional parallel programming languages", *ACM Transactions on Programming Languages and Systems*, Vol. 18, Issue 4, 1996, pp. 454-476.

[21] M. Cole, "Algorithmic Skeletons: Structured Management of Parallel Computation", *MIT Press*, 1989.

[22] D. B. Skillicorn, "Towards a higher level of abstraction in parallel programming", *Proceedings of Programming Models for Massively Parallel Computers*, Berlin, Germany, 1995, pp. 78-85.

[23] L. de Alfaro, T. Henzinger, "Interface-based design", *Engineering Theories of Software-Intensive Systems, NATO Science Series: Mathematics, Physics, and Chemistry*, Vol. 195, Springer, 2005, pp. 83-104.

[24] T. Henzinger, B. Horowitz, B. C. Kirsch, "Giotto: A time-triggered language for embedded programming", *Proceedings of IEEE*, Vol. 91, Issue 1, 2003, pp. 84-99.

[25] T. Henzinger, S. Matic, "An interface algebra for real-time components", *Proceedings of the Annual Real-Time and Embedded Technology and Applications Symposium*, San Jose, California, 2006, pp. 253-266.

[26] E. Wandeler, L. Thiele, "Real-time interfaces for interface-based design of real-time systems with fixed priority scheduling", *Proceedings of International Conference on Embedded Software (EMSOFT)*, Jersey City, New Jersey, 2005, pp. 80-89.

[27] W. Johnston, "Advances in dataflow programming languages", *ACM Computing Survey*, Vol. 36, Issue 1, 2004, pp. 1-34.

[28] C. Zhang, A. Bakshi, V. Prasanna, "ModelML: a markup language for automatic model synthesis", *Proceedings of the IEEE Conference o Information Reuse and Integration*, Las Vegas, Nevada, 2007, pp. 317-322.

[29] R. Newton, M. Welsh, "Region Streams: Functional macroprogramming for sensor networks", *Proceedings of the International Workshop on Data Management for Sensor Networks*, Toronto, Canada, 2004, pp. 78-87.

[30] K. Gummadi, O. Gnawali, R. Govindan, "Macro-programming wireless sensor networks using Kairos", *Proceedings of the International Conference on Distributed Computing in Sensor Systems*, 2005, pp. 126-140.

[31] P. Eles, A. Doboli, Z. Peng, P. Pop, "Scheduling with buss access optimization for distributed embedded Systems", *IEEE Transactions on VLSI Systems*, Vol. 8, No. 5, 2000, pp.472-491.

[32] J. Borenstein, Y. Koren, "Real-time obstacle avoidance for fast mobile robots", *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5), pp. 1179-1187, Sept/Oct. 1989.

[33] B. Capozzi, J. Vagners, "Evolving (Semi) Autonomous Vehicles", *Proc. of AIAA Guidance, Navigation and Control Conference*, 2001.

[34] D. Fogel, L. Fogel, "Optimal Routing of Multiple Autonomous Underwater Vehicles through Evolutionary Programming", *Proc. of Symposium on Autonomous Underwater Vehicle Technology*, pp. 44-47, 1990.

[35] I. Kaminer, O. Yakimenko, "Cooperative Control of small UAVs for Naval Applications", *43rd IEEE Conference on Decision and Control*, 2004.

[36] D. Brogan, J. Hodgins, "Group Behaviors for Systems with Significant Dynamics", *Autonomous Robots*, 4, pp. 135-153, Kluwer, 1997.

[37] O. Brock, O. Khatib, "Real-Time Obstacle Avoidance and Motion Coordination in a Multi-Robot Workcell", *Proceedings of the IEEE International Symposium on Assembly and Task Planning*, pp. 274-279, 1999.

[38] N. Leonard, E. Fiorelli, "Virtual Leaders, Artificial Potentials and Coordinated Control of Groups", *Proceedings of the IEEE Conference on Decision and Control*, pp. 2968-2973, 2001.

[39] M. Mamei, F. Zambonelli, L. Leonardi, "Distributed Motion Coordination with Co-Fields: a Case Study in Urban Traffic Management", *Proc. of the International Symposium on Autonomous Decentralized Systems*, 2003.

[40] C. W. Reynolds, "Flocks, Herds and Schools: A Distributed Behavioral Model", *Computer Graphics*, Vol. 21, No. 4, pp. 25-43, 1987.

[41] B. Walter, A. Sannier, D. Reiners, J. Oliver, "UAV Swarm Control: Calculating Digital Pheromone Fields with the GPU", *JDMS*, Vol. 3, No. 3, pp. 167-176, 2006.

[42] O. Yakimenko, "Direct method for rapid prototyping of near-optimal aircraft trajectories", *AIAA Journal of Guidance, Control, and Dynamics*, Vol.23, No.5, pp. 865-875,2000.

[43] A. Gil, K. Passino, S. Gananpathy, "Cooperative Task Scheduling for Networked Uninhabited Air Vehicles", *IEEE Transactions on Aerospace and Electronic Sys- tems*, 2007.

[44] J. How, E. King, Y. Kuwata, "Flight Demonstrations of Cooperative Control for UAV Teams", *AIAA "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, 2004.

[45] B. J. Moore, K. Passino, "Decentralized Redistribution for Cooperative Patrol", *International Journal on Nonlinear and Robust Control*, 2007.

[46] H. Yamaguchi, "A distributed motion coordination strategy for multiple nonholonomic mobile robots in cooperative hunting operations", *Robotics and Autonomous Systems*, Vol. 43, No. 4, pp. 257-282, 2003.

[47] D. Y. Yeung, G. Bekey, "A decentralized approach to the motion planning problem for multiple mobile robots", *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, Vol. 4, pp. 1779- 1784, 1987.

[48] M. Soulignac, P. Taillibert, "Fast Trajectory Planning for Multiple Site Surveillance through Moving Obstacles and Wind", *PlanSig*, 2006.

[49] D. Shim, H. Chung, H. Kim, S. Sastry, "Autonomous Exploration in Unknown Urban Environments for Unmanned Aerial Vehicles", *Proceedings of the AIAA GN & C Conference*, 2005.

[50] S. Subramanian, J. Cruz, et al., "Predicting Pop-Up Threats from a Adaptive Markov Model", *Cooperative Control and Optimization, In Conference on.*, 2003.

[51] H. Tanner, A. Jadbabaie, and G. Pappas, "Stable Flocking of Mobile Agents, Part I: Fixed Topology", *Proc. of IEEE Conference on Decision and Control*, pp. 2010-2015, 2003.

[52] H. Tanner, A. Jadbabaie, and G. Pappas, "Stable Flocking of Mobile Agents, Part I: Dynamic Topology", *Proc. of IEEE Conference on Decision and Control*, pp. 2016-2021, 2003.

[53] T. Schouwenaars, M. Valenti, E. Feron, J. How, "Linear Programming and Language Processing for Human/Unmanned-Aerial-Vehicle Team Missions", *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 2, pp. 303-313, 2006.

[54] O. Yakimenko, V. Dobrokhodov, "Airplane trajectory control at the stage of rendezvous with maveuvering boject algorithms syntheses", *Proceedings of the IEEE National Aerospace and Electronics Conference* , 1998.

[55] M. Wang, A. Doboli, T. Robertazzi, "Towards Scalable Distributed Control of Unmanned Autonomous Vehicles ", *Applied Computational Intelligence and Informatics, 2007. SACI '07. 4th International Symposium on*, 2007.

[56] S. Quinlan, O. Khatib, "Elastic Bands: Connecting Path Planning and Control", *Robotics and Automation Proceedings. 1993 IEEE International Conference on*, Vol. 2, pp. 802-807, 1993.

[57] D. Rathbun, B. Capozzi, "Evolutionary Approaches to Path Planning Through Uncertain Environments", *Proc. of the American Institute of Aeronautics and Astronautics (AIAA)*, 2002.

[58] D. Rathbun, S. Kragelund, A. Pongpunwattana, B. Capozzi, "An Evolution based Path Planning Algorithm for Autonomous Motion of a UAV Through Uncertain Environments", *Proc. of the Digital Avionics Systems Conference*, Vol. 2, pp. 8D2-1 - 8D2-12, 2002.

[59] A. Stentz, "Optimal and Efficient Path Planning for Partially-Known Environemnts", *Proc. of the International Conference on Robotics and Automation*, Vol. 4, pp. 3310-3317, 1994.

[60] A. Zelinsky, "A Mobile Robot Exploration Algorithm", *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 6, 1992.

[61] H. Van Dyke Parunak, S. Brueckner, J. Sauter, "Digital Pheromone Mechanisms for Coordination of Unmanned Vehicles", *AAMAS*, 2002.

[62] S. Subramanian, J. Cruz, "Adaptive Models of Pop-Up Threats for Multi- Agent Persistent Area Denial", *Proc. of the IEEE Conference on Decision and Control*, pp. 510-515, 2003.

[63] V. Gazi, K. Passino, "Stability Analysis of Swarms", *IEEE Transactions on Automatic Control*, Vol. 48, No. 4, pp. 692-697, 2003.

[64] V. Gazi, K. Passino, "A Class of Attractions/Repulsion Functions for Stable Swarm Aggregations", *International Journal of Control*, Vol. 77, No. 18, pp. 1567-1579, 2004.

[65] W. T. Reeves, "Particle Systemsa Technique for Modeling a Class of Fuzzy Objects", *ACM Transactions on Graphics*, Vol. 2, No. 2, pp. 91-108, 1983.

[66] M. Wang and A. Doboli, "Location-Aware, Flexible Task Management for Collaborating Unmanned Autonomous Vehicles", *Adaptive Hardware and Systems Conference*, 2009.

[67] Cypress Semiconductor Corporation, *PSoC Mixed Signal Array, Document No. PSoC TRM 1.21*, 2005.

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

[69] C. A. Coello, "An updated survey of GA-based multiobjective optimization techniques", *ACM Computing Surveys (CSUR)*, vol. 32, no. 2, pp. 109-143, 2000.

[70] Y. Shoham, K. Leyton-Brown, "Chapter 3: Introduction to Noncooperative Game Theory: Games in Normal Form", *MULTIAGENT SYSTEMS Algorithmic, Game-Theoretic, and Logical Foundations*, pp. 47-87.

[71] S.C.Zaharakis, A. Guez, "Time optimal robot navigation via the slack set method", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no.6, Nov/Dec 1990.

[72] T.-H.S. Li, Sheng-Sung Jian, Ming-Che Tsai, "Design and implementation of fuzzy trajectory following and planning control for mobile robots", *Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology*, TENCON, 2001.

[73] S. Hirashita, T. Yairi, "Information-based Exploration Strategy for Mobile Robot in Dynamic Environment", *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 2009.

[74] F. Amigoni, V. Caglioti, U. Galtarossa, "A mobile robot mapping system with an information-based exploration strategy", *Proceedings of the International Conference on Informatics in Control, Automation and Robotics*, pp. 71-78., 2004.

[75] M. Jia, G. Zhou, Z. Chen, "An Efficient Strategy Integrating Grid and Topological Information for Robot Exploration", *Proceedings of the 2004 IEEE Conference on Robotics, Automation and Mechatronics*, Singapore, 2004.

[76] A.K. Poernomo, H. S. Ying, "New Cost Function for Multi-Robot Exploration", *9th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Dec. 2006.

[77] G. Lidoris, K. Kuhnlenz, D. Wollherr, M. Buss, "Combined Trajectory Planning and Gaze Direction Control for Robotic Exploration", *IEEE International Conference on Robotics and Automation*, Apr. 2007.

[78] B. Yamauchi, A. Schultz, W. Adams, K. Graves, "Integrating map learning, localization and planning in a mobile robot", *Proceedings of Intelligent Control (ISIC)*, 1998. Held jointly with *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Intelligent Systems and Semiotics (ISAS)*, 1998, pp. 331-336.

[79] C. Tovey, S. Koenig, "Improved analysis of greedy mapping", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, pp. 3251-3257., 2003.

[80] W. Burgard, D. Fox, M. Moors, R. Simmons, S. Thrun, "Collaborative multi-robot exploration", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 476-481., 2000.

[81] H. H. Gonzalez-Banos, J. C. Latombe, "Navigation strategies for exploring indoor environments", *International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829-848., 2002.

[82] D. Basmadjian, "The Art of Modeling in Science and Engineering", *Chapman & Hall*, 1999

[83] Y. Shoham, K. Leyton-Brown, "Multiagent Systems: Algorithmic, Game-theoretic, and Logical Foundations", *Cambridge University Press*, 2009.

[84] C. Tomlin, G.J. Pappas and S. Sastry, " Conflict resolution for air traffic management: a study in multiagent hybrid systems", *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 509-521, 1998.

[85] C. Tomlin, J. Lygeros and S. Sastry, "A game theoretic approach to controller design for hybrid systems ", *Proceedings of the IEEE*, vol. 88, no. 7, pp. 949-970, 2000.

[86] C. Tomlin, I. Mitchell and R. Ghosh, "Safety verification of conflict resolution manoeuvres", *IEEE Transactions on Intelligent Transportation Systems*, vol. 2, no. 2, pp. 110-120, 2001.

[87] Y. Yoon, J.M. Park, H.J. Kim and S. Sastry, "Utilizing parallax information for collision avoidance in dynamic environments", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.