INTERACTIVE ROUTE-PLANNING AND MOBILE SENSING WITH A TEAM OF MULTIPLE ROBOTIC VEHICLES TO SATISFY LINEAR TEMPORAL LOGIC SPECIFICATIONS

by

Jie Fang

A dissertation submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN MECHANICAL ENGINEERING.

June 2020

APPROVED:

Dr. Raghvendra V. Cowlagi, Advisor
Associate Professor, Aerospace Engineering Department.

Dr. Michael A. Demetriou, Committee Member
Professor, Aerospace Engineering Department.

Dr. Jie Fu, Committee Member
Assistant Professor, Robotics Engineering Program.

Dr. Xiangrui Zeng, Committee Member,
Assistant Professor, Mechanical Engineering Department.

Dr. Yihao Zheng, Graduate Committee Representative,
Assistant Professor, Mechanical Engineering Department.

Acknowledgements

Five years ago, I started my journey of Ph.D. in WPI. Five year might only be a small piece in one's life, but these five years definitely are very important to me and lead me to the next stage in my life. During these five years, I was lucky to meet many amazing people who gave me so much help and support, especially my advisor, my committee member, family and friends.

First of all, I would like to express my sincere gratitude to my advisor Raghvendra V. Cowlagi. Ragu continuously provided encouragement and was always willing and enthusiastic to assist in any way he could throughout the research project. On the way to pursuing the degree, Raghu guided me to achieve each small goal step by step and encouraged me to keep challenging myself. At the end of this journey, I am so excited to see these achievements which I never think about at the beginning and I have to say, without his supervision and encouragement, it would be impossible for me to finish my Ph.D. Meanwhile, Raghu also inspired me by his hardworking and passionate attitude.

I would like to thank all my committee members. Dr. Yihao Zheng, for serving as the committee representative. Dr. Xiangrui Zeng, for his inspiring comments during the defense. Dr. Jie Fu, for her insightful suggestions during both the proposal and the final defense. Finally, I want to thank Dr. Micheal A. Demetriou, from whom I took many control courses and realized the beauty of math.

I am grateful for the financial support of the Mechanical Engineering department through Teaching Assistant positions, and later through the United State Air Force Office of Scientific Research (FA9550-17-1-0028). This support make this research possible.

I would like to thank my labmates, Zetian Zhang, Benjamin Cooper, Yin Xiao, Chase St. Laurent, who gave me lots of help and suggestions related to the research, and Ruixiang Du who stimulated my interest in programming. Meanwhile, I also would like to thank my friends Mengqiao Yu, Livia M. Motz, Jighjigh Ivase, Mucheng Li, Zahra Noori, who brought lots of joy to my Ph.D. life.

I thank my parents and grandparents who always try to understand and respect my decisions. Their trust and support make me pursue my dream bravely. Now I am so happy to share this big achievement with them. Finally, I want to thank Benjamin who gave me lots of encouragements and technical help to my research. He also inspired me to keep moving forward by his hardworking and perseverance.

Abstract

Unmanned aerial and terrestrial vehicles (UXV) are envisioned to serve a broad variety of civilian and military applications including mountain search and rescue, cargo pickup/drop off, target tracking and source localization etc. To encode high level behavioral specifications on such vehicles, linear temporal logic (LTL) formula are widely used. To satisfy such specifications by a team of multiple robotic vehicles, two main technical problems are commonly considered. On the one hand, the *planning problem* addresses the collaboration among a team of vehicles and route planning to satisfy the given global mission. A practical example is the package delivery with least total traveled distance. The solution to this problem is based on the accurate knowledge of the environment. On the other hand, the sensing problem addresses the deployment of a set of sensors to measure subregions of unknown environment which is crucial for the planning problem. An example for this problem is search and rescue in an unknown region.

A heterogeneous multi-vehicle team is considered where one set of vehicles called *sensors* are deployed to explore/map the environment, while the other set called *actors* are deployed to carry out intelligent collaborative tasks based on the map generated by the sensors. A traditional solution to the problem is to decouple the planning and sensing and solve each problem separately. Here, the sensors are required to rebuild the whole map as much as possible. However, this decoupled approach is wasteful in sensory resources (numbers of sensors and measurements). Instead, we propose a bootstrapping, iterative, and interactive route-planning and sensor placement technique that finds near-optimal routes for actors which are required to work collaboratively to satisfy a global task. The goal of this dissertation is to investigate the benefit of this *coupled planning and sensing algorithm* to the problem of route-planning in an unknown/uncertain environment. The contributions of this work are as follows.

1. We consider the problem of optimal planning under the assumption that accurate knowledge of the environment is available. The objective of this problem is to decompose the global specification into local specifications called *tasks* and compute the optimal route for each

actor such that the total traveled distance is minimized. A decentralized network eliminates the possibility of a single centralized decision maker in the team. Instead, actors communicate and converge to a conflict-free task assignment. A novelty of this work is that, besides *independent tasks* which only require a single actor, the proposed planning algorithm also assigns *collaborative tasks*, i.e., tasks simultaneously assigned to multiple actors. This algorithm synchronizes the actors' routes with minimum total waiting durations. To address vehicle's kinematic constraints, the lifted graph technique is implemented to update the rewards in the proposed task assignment algorithm.

2. An interactive planning and sensing algorithm is developed for the aforesaid planning problem in an unknown environment. This problem now requires a set of sensors to explore the most "informative" regions iteratively and use the measurements to construct the unknown environment. The concept of information gain is used to quantify the potential benefits of taking new sensor measurements. At each iteration, optimal actors' routes are computed to accomplish the intelligent task based on latest knowledge of environment. The task-driven information gain is evaluated and used to guide the sensors to reduce the uncertainty of current actors' routes. The interactive between the planning and sensing will not stop until certain terminate condition is satisfied, i.e., enough confidence of obstacle-free routes for actors is achieve. This coupled planning and sensing strategy aims to doing the planning and sensing simultaneously and will converge with far fewer measurements than is required when planning and sensing are separate.

Future work considers different extension of the proposed interactive planning and sensing framework including: sensor reconfiguration cost, taking measurement along the trajectory and more complex LTL global specification.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Literature Review

1.1 Motivation and Problem Statement

Motion-planning and coordination for a team of networked mobile robots plays an important role in autonomy. It is highly relevant for applications such as formation flying, target tracking and pursuit, source localization, sensing and estimation of distributed processes and surveillance. We use linear temporal logic (LTL) to specify these high level behavioral specification for mobile robots. LTL is a formal language that includes temporal operators such as `always` and `eventually` in addition to the usual logical operators `and`,`or`, and `not`. For example, LTL can be used to specify UXV motion to ''`visit region A followed by region B and always avoid obstacles`''. The global LTL specification is given and required to be satisfied by all vehicles collaboratively. Here, we assume the global specification is a conjunction of multiple formulae, each of which is treated as a task to be assigned to one or more vehicles.

Motion-planning and coordination for a multi-vehicle system relies heavily on the knowledge of the environment. However, an accurate map of environment may not always be available. Therefore, this dissertation addresses two major component problems: (1) sensing and mapping of the unknown environment, and (2) task assignment and route planning to satisfy given global mission specifications. A traditional solution to the problem is to decouple these two components and two sets of vehicles: *actors* and *sensors* , which are deployed to accomplish each duty separately.

To understand the unknown environment, a set of sensors are deployed to take measurements in the unknown environment. A popular topic of unknown environment mapping is simultaneous localization and mapping (SLAM) which traverses the environment with onboard sensors. By contrast, we are interested in a scenario where mobile sensors, separate from the primary actors,

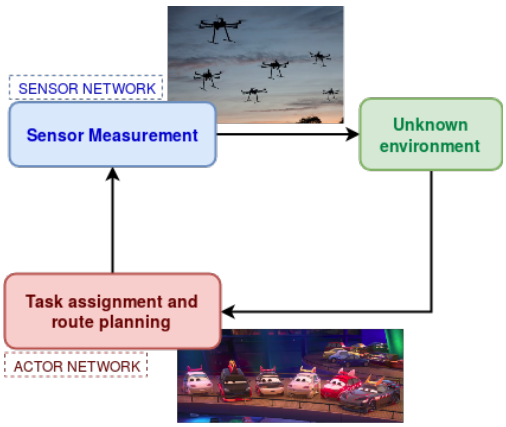


Figure 1.1: Diagram of interactive planning and sensing.

can be placed in the environment. Based on the map updated by the sensory measurements, a conflict free task assignment and optimal route for each actor to satisfy the global specification is computed.

However, in many cases, the actors' optimal routes lie within a relatively small subregion of the environment. In such cases, the actors can find near-optimal routes even if the environment map accuracy is limited only to this subregion, while the rest of the environment remains unknown or less accurately known. This limited map information can be gained efficiently using far fewer sensory resources (numbers of sensors and/or measurements) compared to what is required to gain the *entire* environment map information. When sensory resources are expensive, e.g., deployment of surveillance aircraft, the following research question becomes important: **can sensor placement be coupled with actors' route-planning to reduce the required amount of sensor data?** In this dissertation, we propose a technique called *interactive planning and sensing* where the separation between the two components is removed. Fig. 1.1 displays the diagram of the algorithm and emphasizes the interaction between the planning and sensing.

To drive sensor exploration, the concept of information gain is used to quantify the potential benefits of taking new measurements in certain regions. Differ from the traditional methods where the information gain is evaluated corresponding to the whole environment, we propose task-driven information gain which is computed based on the entropy reduction over a subregion near the actors current routes. A greedy exploration strategy is implemented to maximize this task-driven information gain and encourage sensory exploration of the most "informative" regions required by

the actors. The crucial innovation is that the proposed algorithm attempts to reduce the uncertainty of the actors' routes to satisfy the given global task rather than that of the entire map or pre-defined sensing region. The unknown environment will be updated using the latest measurements iteratively.

The planning problem addressed in this dissertation is computation of an optimal route for each actor to satisfy the given global task. The global specification is assumed to be a conjunction of independent or collaborative tasks. Here, each independent tasks will be assigned to a vehicle and each collaborative task will be simultaneously assigned to multiple vehicles. The objective of planning is to decompose the global task into local tasks for each actor or determine the task assignment among actors while minimizing total traveled distance. We consider a decentralized network among the team of vehicles and a static communication topology is assumed which allows vehicles to communicate and converge a conflict-free task assignment to satisfy the global task.

As in Fig. 1.1, at each iteration, based on the latest knowledge of map, the group of actors update the optimal task assignment to satisfy the global task. The set of "informative" subregions are selected by each actor based on its current optimal route. The actors reach decentralized consensus about informative subregions and sensors are assigned to map these subregions such that the total distance traveled by sensors is minimized. The iteration between the planning and sensing terminate when the uncertainty of actors' route reduces below a desired threshold.

For a concrete example, consider the problem of search and rescue. A type of discovery vehicle equipped with a drone is implemented to do the search and rescue operation as shown in Fig. 1.2(a). The drone gives the rescue team a bird's-eye view of potential survivors or possible hazards that could get in the way of rescuing stranded individuals. The drone can also broadcast live of footage of what it's seeing back to the rescue crew back at the SUV, giving teams a clear, immediate view of what lies ahead. This ability is critical as changes in terrain from avalanches, mudslides, floods, and earthquake can make maps ineffective. The trajectory of the drone can be controlled by the rescue crew in the SUV through a tablet app in Fig. 1.2(b). The drones use both live camera and thermal imaging to assist the rescue team to quickly search the scene from safe distance. For search and rescue, the collaboration of SUV and drones is important. For this problem, we study "informative" regions for drones to take measurements and update the map and

(a) Discovery rescue team (b) Trajectory planning for drones

Figure 1.2: Land Rover's search and rescue SUV with drones. [1]

then compute optimal routes for each SUV.

1.2 Literature Review

An increasing amount of attention has been paid to the problem of route-planning for a team of networked vehicles to complete collaborative tasks formulated using LTL specification in recent literature [2–5]. In general, the objective is to find an obstacle-free or risk-free trajectory/path for each vehicle in the team so that the global LTL specification is satisfied. However, different from the route-planning for the single vehicle system [6], an extra level of planning is required for the multi-vehicle system: task assignment which determines how the group of vehicles should collaborate to accomplish the given mission. Based on the communication structure of the multi-vehicle system, two types of networks are studied: centralized and decentralized network. For the centralized network [7–9], all information from each vehicle in the team will be passed to a single decision maker and used to determine the optimal task assignment. However, stable communication among vehicles in the team is crucial to ensure the latest and correct information is transmitted to the decision maker. Compared to the centralized network, the decentralized network has higher flexibility [10–12]. Instead of having a single decision maker , a conflict-free task assignment is achieved by the communication among neighboring vehicles.

1.2.1 Route-planning under Linear Temporal Logic Specification

Linear temporal logic (LTL) is a formal system, similar to the commonly used propositional logic system [13]. In addition to the standard operators `and`, `or`, and `not`, LTL includes temporal operators such as `always`, `eventually`, and `until`. LTL has been used in software design for the specification of "correct" behaviors of algorithms [14, 15]. More recently, LTL has been applied for specifying behaviors of dynamical systems, and in particular, behaviors of mobile vehicles [16–18]. For robotic mobile vehicles, high-level intelligent "tasks" as well as properties of safe behaviors, e.g. *"perform persistent surveillance in region A until a target is found, then report data to region B, never fly in region C, and finally return to base"* can be formulated with LTL specifications.

The general approach to vehicle route-planning to satisfy LTL specifications is based on formal control synthesis techniques for dynamical systems [19–21]. A Büchi automaton, whose accepting runs are exactly the strings that satisfy the given LTL specification, is constructed [22–25]. Next, a product of this Büchi automaton with a finite state model, called a *discrete abstraction,* of the dynamical system is constructed. A search algorithm, such as Dijkstra's algorithm is then employed on this product automaton for control synthesis. Applications of this general approach to motion-planning of mobile vehicles are well-studied in the literature [16, 26, 27].

The construction of a discrete abstraction is a crucial step in the aforesaid approach to route-planning to satisfy LTL specifications. For a vehicle modeled as a linear dynamical system, a discrete abstraction can be constructed using polytopic partitions of a compact domain of the state space, with control laws to enable transitions between adjacent polytopes or invariance within a polytope [21, 28]. Similar approaches based on state space partitioning have been studied for non-linear dynamical models [29–31]. However, these approaches often result in abstractions extremely large number of discrete states, even for low-dimensional dynamical systems.

1.2.2 Route-Planning with Vehicle Kinematic Constraint

Vehicle kinematic models are typically ignored in the literature on multi-vehicle LTL satisfaction. Instead, the literature either **(1)** considers extremely simple models such as single or double integrators, cf. [32] or **(2)** considers linear models, cf. [4], or **(3)** altogether ignores the vehicle model and presupposes a finite state discrete abstraction, cf. [5]. For vehicles with nonholonomic kinematic constraints, including the simple Dubins car model, typical methods of constructing discrete abstractions [33] may not apply, and the presumption of a discrete abstraction is not justified. The technique of CBTA is implemented in this dissertation to address vehicle's kinematic constraint. Refer [34] for further details.

1.2.3 Decentralized Task Assignment

To satisfy the given specification on the team's behavior - henceforth called *global* specification, the notion of distributability of LTL specifications is discussed in [35,36], which addresses the issue of whether a given LTL specification for a team of vehicles can be decomposed into "local" specifications for each vehicle, given the capabilities of each vehicle to satisfy such specifications.

On the one hand, several *centralized* algorithms are reported to decompose the global specifications into local specifications – i.e., *tasks* – for each vehicle [4,35–38]. Such algorithms typically rely on a discrete abstraction *of the team,* namely, a suitable "product" of the discrete abstractions of each vehicle's model, and on methods to decompose global LTL specifications [39, 40]. As the number of vehicles increases, the dimension of the product abstraction increases exponentially. Although sampling-based techniques are implemented in [41] to construct a reduced team abstraction, only asymptotic optimality can be guaranteed. Meanwhile, using randomized sampling-based algorithms for local route-planning is particularly ill-advised in this context because these algorithms can report different route costs for the same specification in different invocations by the task assignment algorithm. This behavior can disrupt the convergence of the task assignment algorithm.

On the other hand, several *decentralized* algorithms [5, 32, 42–45] provide motion coordina-

tion and control, including temporal synchronization, to satisfy local LTL specifications. These methods assume that the tasks for each vehicle are assigned a priori. A global LTL specification is considered in [46]. However, these algorithms address the satisfaction of *pre-assigned* local specifications for each vehicle, i.e., conflict-free task assignment is not addressed, rather it is presumed.

Compared to the existing decentralized algorithms, the techniques to decompose the global LTL specification into local specifications are not available. The lack of decentralization is a serious computational hurdle because the complexity of searching an optimal path to satisfy a given LTL specification depends on the dimension of the discrete abstraction. Discrete abstractions of single vehicle models are themselves high-dimensional transition systems [47]; the product of multiple such models for a team of vehicles becomes impractically large. A reduced-order product is reported [5], but its construction presupposes that the local LTL specifications are already known. Randomized sampling-based algorithms are reported to address extremely large dimensional transition systems [41, 48, 49], but these algorithms sacrifice optimality. Decentralized task assignment algorithms can invoke local route-planning algorithms over multiple iterations, and possibly with the same local specifications. Using randomized sampling-based algorithms for local route-planning is particularly ill-advised in this context because these algorithms can report different route costs for the same specification in different invocations by the task assignment algorithm. This behavior can disrupt the convergence of the task assignment algorithm.

1.2.4 Sensor Placement

The study of sensor placement is to find locations of sensors to optimize certain performance. In some cases, the goal is to maximize sensors' coverage [50–52]. In these cases, linear integer programming can be implemented to determine sensors' optimal locations [53, 54]. On the other hands, we define certain performance metric to quantify the amount of information can be collected by sensors and now the goal is to determine sensors' locations to maximize this metric. Some popular performance metric implemented in current literatures are: entropy, determinant, largest eigenvalue, Fisher Information matrix and mutual information [55, 56].

1.2.5 Task-driven Unknown Environment Exploration

As we have discussed above, both the task assignment and route planning rely heavily on accurate knowledge of the environment which may not be available all the time. To represent the unknown/uncertain environment, the technique of probabilistic occupancy map (POM) is used. Each uniformly sized square cell in this map is associated with a random variable indicating the probability of presence of an obstacle in the cell [57–60]. Typical POM-based approaches use sensor measurements to compute Bayesian updates to these random variables [61, 62].

To drive sensor exploration [61–65], the concept of *information gain* [66] is used to quantify the potential benefits of taking new sensor measurements. The exploration of the most "informative" regions has been recently studied, where the objective of maximizing information gain in general encourages sensory exploration of unknown regions [67]. Greedy exploration strategies, i.e., driving sensors to regions with the highest information gain, have been studied [68, 69]. are implemented in which the algorithms select the next best observation position with maximum information gain. A machine learning based technique: Bayesian optimization is studied [70] to increase the computational efficiency for information gain. Other related works include a curiosity-driven exploration strategy [71], information gain maximization on a hexagaonal decomposition [72], and applications to source detection [73], target localization [74] and search-and-capture [75]. The adaptive informative path planning algorithm [76, 77] addresses sensor path-planning to compute the most "informative" *path* inside a given sensing region.

1.3 Dissertation Overview and Statement of Contributions

The goal of this dissertation is to develop an interactive planning and sensing algorithm for multi-vehicle system to satisfy the given LTL specification in unknown environments.

1.3.1 Overview

Two complementary bodies are involved in the problem: one on motion coordination to perform intelligent tasks, and the other on exploration/mapping in unknown environment. We start with the study of each component separately and then investigate the potential benefit of the interactive planning and sensing algorithm which couples these two components. Two set of vehicles:*actors* and *sensors* are deployed to address each component individually.

For the problem of motion coordination to perform intelligent tasks, the assumption of having accurate knowledge of environment is given. We extend the single vehicle system to multi-vehicle system which brings an additional decision making problem: how the team of vehicles should collaborate with each other to satisfy the given global specification. We observe three gaps in the existing literatures: (1) the decentralized techniques mentioned in the literatures above avoid the problem of task assignment or assume the tasks for each vehicle are pre-assigned. (2) vehicle kinematic models are not addressed or typically ignored on multi-vehicle LTL satisfaction. (3) *collaborative* tasks: those simultaneously assigned to multiple vehicles, are not considered. We address all these three gaps in Chapter 2 and Chapter 3.

Next, we consider the problem of unknown environment mapping. An effective balance between exploration and exploitation is addressed. We consider a scenario where mobile sensors can be placed in the environment and take measurements. Different from the simultaneous localization and mapping which aims to recreating the map as much as possible, our goal is to rebuild part of the unknown environment which are important to perform intelligent tasks using limited sensory measurements. However, the obvious challenges is that the actors' route-planning to satisfy the given global task depends on the environment map, which leads to a "chicken-and-egg" paradox.

We proposed a bootstrapping and iterative technique that simultaneously finds sensor placements and near-optimal actor routes in Chapter 4. A task-driven information gain is evaluated based on entropy reduction over a subregion "near" the actors' current routes. A set of "informative" subregions are selected correspondingly using MINMISSstrategy [78]. The computational burden from evaluating information gain is also addressed, a machine-learning-based technique: Bayesian optimization is implemented to model the function of information gain and predict the

"informative" region without computing the information gain for each potential sensing locations.

1.3.2 Contributions

We make the following contributions toward motion coordination for multi-vehicle system to satisfy LTL specification in unknown/uncertain environments.

Route-Planning for Multi-Vehicle Teams - Independent Tasks. We propose a decentralized approach to motion-planning and coordination of a team of mobile vehicles to collaboratively satisfy a global LTL specification. The global LTL specficiation is assumed to be a conjunction of independent tasks where each task is assigned to a single vehicle. The proposed technique draws upon two methods previously reported in the literature: the consensus-based bundle algorithm (CBBA) and the method of lifted graphs. Here, CBBA is a method for decentralized task assignment, which we apply for decomposing the global LTL specification into specifications for each vehicle. The method of lifted graphs addresses the nonholonomic kinematic constraints of the vehicle, and enables the construction of discrete abstraction of the vehicle model based on partitioning of the vehicle's 2D workspace, rather than its state space. The proposed technique considers nonholonomic vehicle kinematic constraints, and does not require complete controllability or linearization, which distinguishes this technique from previously reported techniques based on workspace partitioning. The trajectory computed by the proposed technique can be feasibly tracked by vehicles subject to a nonholonomic constraint.

Route-Planning for Multi-Vehicle Teams - Collaborative Tasks. We propose a method for synchronizing vehicle routes to satisfy collaborative tasks which need to be assigned to multiple vehicles. The proposed work is suited to applications where teams of mobile robotic vehicles are to execute complex and intelligent tasks, the environment map is known, and robot localization is not a significant problem. Examples include warehouse inventory management and "last-mile" pickup and delivery of goods. The proposed route-planning method does not interfere with the low-level control. Therefore, a real-world implementation of the proposed method can be treated as a high-level waypoint generation method. Turn radius constraints on such waypoint tracking are

incorporated into the proposed method. This method can be implemented on any mobile robot with waypoint tracking capability. The proposed algorithm achieves task assignment with minimum total waiting. For the sake of numerical comparisons, we also propose a new extension of the consensus-based group algorithm [79] to satisfy LTL specifications with kinematically feasible routes.

Interactive Planning and Sensing. We propose a bootstrapping and iterative technique that simultaneously finds sensor placements and near-optimal actor routes. A task-driven information gain is evaluated based on entropy reduction over a subregion "near" the actors' current routes. Optimal actors' routes are updated by a decentralized technique to satisfy the given global LTL specification. The crucial innovation is that the proposed sensor placement attempts to reduce the entropy *of the actors' route cost* rather than that of the entire map or pre-define sensing regions as literatures discussed above. A set of "informative" subregions are selected by each actor using the MINMISS strategy [78]. The actors reach decentralized consensus about informative subregions and sensors are assigned to map these subregions such that the total distance traveled by sensors is minimized. The iterations of route-planning and sensor placement terminate when the entropy of actors' route cost reduces below a desired threshold. We also provide the analysis of the convergence rate versus the available sensory resources and a study of the pre-defined desired threshold.

Task-Driven Information Gain Maximization with Bayesian Optimization. To identify a set of "informative" subregion for actors' route-planning, the interactive planning and sensing algorithm requires evaluation of the information gain for each vertex in the environment, which is computationally expensive especially when the dimension of the environment is large. For computational efficiency, a machine-learning-based technique, Bayesian optimization, is implemented to predict the set of sensors locations without repeated computation of information gain. Bayesian optimization uses Gaussian process regression to model the function of information gain and then predict the sensors locations via expected improvement acquisition function. To build the model of information gain, we select the training data on a set of entropy values of a certain vertex and its neighboring vertices randomly. Compared to the original interactive planning and sensing al-

gorithm, a much smaller number of vertices are required to compute the information gain which reduces the computational burden substantially. A comparison of computational efficiency required by the interactive planning and sensing algorithm with/without Bayesian optimization is provided. Furthermore, the accuracy of the technique is also addressed.

The following is the list of the publications addressed the topics shown above:

- J.Fang, Z.Zhang, R.V.Cowlagi. *Decentralized Route-Planning to Satisfy Global Linear Temporal Logic Specifications on Multiple Aircraft*, Guidance, Navigation, and Control Conference, AIAA SciTech 2018, Orlando, FL, USA, January 2018.

- J.Fang, Z.Zhang, R.V.Cowlagi. *Decentralized Route-Planning for Multi-Vehicle Teams to Satisfy A Subclass of Linear Temporal Logic Specifications.* IEEE Transactions on Automation Science and Engineering [Submitted]

- J.Fang, H.Zhang, R.V.Cowlagi. *Interactive Route-Planning and Mobile Sensing with a Team of Robotic Vehicles in an Unknown Environment.* Guidance, Navigation, and Control Conference, AIAA SciTech 2020.[Submitted]

- J.Fang, R.V.Cowlagi. *Decentralized Route-Planning and Mobile Sensor Placement with Task-Driven Information Gain Optimization for a Team of Multiple Robotic Vehicles in Unknown Environments.* [Under preparation]

Chapter 2

Route-planning for Multi-Vehicle System: Independent Tasks

2.1 Problem Elements Overview

In this chapter, a global LTL specification is required to be satisfied by a multi-vehicle system. Firstly, we assume that the given global LTL specification is a conjunction of a set of independent tasks where each one is assigned to a vehicle. Perfect knowledge of the environment and a static communication network are given a prior. We are looking for a decentralize technique to compute optimal route for each vehicle in the team so that the global mission can be accomplished and the total traveled distance is minimized. In what follows, $\mathbb{N}, \mathbb{Z}$, and $\mathbb{R}$ denote the set of natural, integer, and real numbers, respectively. For any $N \in \mathbb{N}$, define $[N] := \{1, \ldots, N\}$. Furthermore, decision-making, communications, and control software onboard each vehicle are collectively referred to as an *agent*. The vehicles are assumed to be identical and interchangeable.

For the reader's convenience, a nomenclature table is provided in Table 2.1.

2.1.1 Vehicle Model

To consider the route-planning for each vehicle in the team, two vehicle models are addressed as follows.

Unconstrained particle model (UCPM) The vehicle state is $\mathbf{x} = (x, y) \in \mathcal{W}$, namely, the position of the vehicle center of mass in a prespecified Cartesian coordinate system. The UCPM vehicle can move in any direction at constant unit speed and it can instantaneously stop, every route in $\mathcal{L}_\mathcal{G}$ is traversable. The UCPM is appropriate for slow-moving differential drive robots

Symbol	Meaning
$\mathcal{W}$	2D environment.
$\mathcal{G}$	Grid map corresponding to the environment.
$\mathcal{G}_H$	Lifted graph.
ρ	Vehicle turning radius.
N^{C}	Number of region of interested.
N_{A}	Number of actors (agents).
$\mathcal{N}_i$	Neighbors of agent i.
N_{TI}	Number of independent tasks.
r_{ij}	Reward of task j for agent i.
$\mathbf{b}_i$	Task bundle for agent i.
$\mathbf{p}_i$	Task path for agent i.
$\bar{\mathcal{J}}[H](\mathbf{v})$	H-cost for path $\mathbf{v}$.
ϕ^{S}	Safety specification.
ϕ^{L}	Liveness specification.

Table 2.1: Parameters for decentralized route-planning - Independent task.

commonly used in warehousing [80]. The UCPM discrete abstraction is the graph $\mathcal{G}$.

Kinematically constrained particle model (KCPM) Here we consider two modes of motion: a *hover* mode, where the vehicle can remain stationary, *forward motion* mode, where it moves at a constant unit speed. The KCPM is appropriate for fast-moving vehicles that can hover, e.g. single- or multi-rotor aircraft, or fixed-wing/rotorcraft hybrid designs. The vehicle state is $\mathbf{x} = (x, y, \psi) \in \mathcal{D} := \mathbb{R}^2 \times \mathbb{S}^1$, where ψ is the direction of the velocity vector. The vehicle state evolves as:

$$\dot{x}(t) = a\cos\psi(t), \qquad \dot{y}(t) = a\sin\psi(t), \qquad \dot{\psi}(t) = u(t), \tag{2.1}$$

where the heading rate u is the control input, and $a = 1$ (resp. $a = 0$) in the forward motion mode (resp. hover mode). The set of admissible control input values is the interval $\left[-\frac{1}{\rho}, \frac{1}{\rho}\right]$, with $\rho > 0$. An admissible trajectory traces a curve with a minimum radius of turn of ρ in the forward motion mode. The parameter ρ can be different for each vehicle in the team, thereby modeling heterogeneous maneuvering capabilities. The method of *lifted graphs* is used to construct a discrete abstraction for the KCPM. Briefly, this method defines "lifts" the graph $\mathcal{G}$ to define a new graph whose vertices correspond to sequences of adjacent cells in the workspace that are

traversable within the kinematic constraints. Route-planning is then performed by searching in this new lifted graph. §2.3.1 provides a brief overview of this method, and the reader is referred to [33] for further details.

A limitation of this work is that we do not explicitly consider uncertainty in either the UCPM or the KCPM. The proposed work develops a high-level route planner, and presumes a low-level feedback controller that can track the planned route despite modeling uncertainty. Furthermore, the result of the proposed route planner are vehicles routes corresponding to sequences of cells in the workspace. That is, each route is a *region* in the workspace, rather than a specific trajectory, which provides an inherent robustness to the result of the route planner.

2.1.2 Workspace Cell Decomposition

Let $\mathcal{W} \in \mathbb{R}^2$ denote a planar region where the vehicles collaboratively operate. Consider a cell decomposition, i.e. a partition of $\mathcal{W}$ into convex subregions called *cells*. We denote by $N^{\mathrm{C}} \in \mathbb{Z}_+$ the number of cells, and by $R_i \subset \mathcal{W}$ the subregion associated with the i^{th} cell, for each $i = 1, \ldots, N^{\mathrm{C}}$. We associate with this partition a graph $\mathcal{G} := (V, E)$ such that each vertex of $\mathcal{G}$ is uniquely associated with a cell, and each edge of $\mathcal{G}$ is uniquely associated with a pair of geometrically adjacent cells. We denote by $\mathsf{cell}(v)$ the element of $\{R_i\}_{i=1}^{N^{\mathrm{C}}}$ associated with the vertex $v \in V$. A *path* $\mathbf{v}$ in $\mathcal{G}$ is a finite or infinite sequence $(v_0, v_1, \ldots)$ of vertices, such that $v_0, v_k \in V$, and $(v_{k-1}, v_k) \in E$, for each $k \in \mathbb{N}$. The number of vertices in a path is called its *length*. Note that a path in $\mathcal{G}$ can contain cycles. We denote by $\mathcal{L}_{\mathcal{G}}$ the collection of all paths in $\mathcal{G}$.

Special cells called *base locations* and *regions of interest (ROI)* are prespecified. The number of base locations is equal to the number N_{A} of robotic vehicles in the team, and each robot is assumed to start from one of these locations. The vertices in V associated with these cells are denoted by $v_{\mathrm{B},1}, \ldots, v_{\mathrm{B},N_{\mathrm{A}}}$. The number of ROIs is N_{R}. Each ROI is a connected union of cells, i.e. the k^{th} ROI is $\cup_{i \in \varsigma_k} R_i$, where $\varsigma_k \subseteq \{1, \ldots, N^{\mathrm{C}}\}$, $k = 1, \ldots, N_{\mathrm{R}}$ are prespecified. The associated vertices are denoted $v_{\mathrm{R},\ell k}$, $\ell \in \varsigma_k$.

For every $t_{\mathrm{f}} \in \mathbb{R}_+$ and $u \in \mathcal{U}_{t_{\mathrm{f}}}$, we define the *$\mathcal{G}$-trace* of a trajectory $\xi(\cdot; \xi_0, u)$ as the path

$tr(\xi, \mathcal{G}) = (v_0, v_1, \ldots) \in \mathcal{L}_{\mathcal{G}}$ with minimum length such that $\mathsf{x}\xi\,(0; \xi_0, u) \in \mathsf{cell}(v_0)$, and

$$\mathsf{x}\xi\,(t; \xi_0, u) \in \cup_{k=0}^{P} \mathsf{cell}(v_k), \qquad t \in [0, t_{\mathrm{f}}]\,, \qquad k \in \mathbb{N}. \tag{2.2}$$

We denote by $\mathcal{L}_\Gamma(\xi_0) \subseteq \mathcal{L}_{\mathcal{G}}$ the collection of $\mathcal{G}$-traces of all admissible trajectories for every $t_{\mathrm{f}} \in \mathbb{R}_+$. Informally, the path $tr(\xi, \mathcal{G})$ is associated with the sequence of cells that defines a "channel" in $\mathcal{W}$, such that the curve $\mathsf{x}\xi(t), t \in [0, t_{\mathrm{f}}]$, lies within this channel. The curve $\mathsf{x}\xi(t)$ and the trajectory $\xi(t)$ are said to *traverse* this channel of cells.

A simplifying assumption in this paper is that multiple vehicles are allowed to be within the same cell at the same time-step. The justification for this assumption is that the size of cells is assumed to be significantly larger than the dimensions of the vehicle (but comparable to the minimum radius of turn in forward motion), thereby allowing enough room for more than one vehicle within a cell.

2.1.3 LTL$_{-X}$ specifications

The LTL$_{-X}$ syntax is a restricted version of LTL which excludes the *next* operator. The reader is referred to [81] for further details. An *atomic proposition* is a statement that is either true or false at a given instant of time. An LTL$_{-X}$ formula over a set of atomic propositions Λ is recursively defined as (1) Every atomic proposition $\lambda_n \in \Lambda$ is a LTL formula; and (2) If ϕ_1 and ϕ_2 are LTL formulae, then $\neg\phi_1$ (negation), $\phi_1 \vee \phi_2$ (conjunction), and $\phi_1 \mathcal{U} \phi_2$ are also LTL formulae. The formula $\phi_1 \mathcal{U} \phi_2$ means that ϕ_2 eventually becomes true and ϕ_1 remains true until ϕ_2 becomes true. The temporal operators $\Diamond$ (`eventually`), and $\Box$ (`always`) are defined as $\Diamond\phi_1 := (\phi_1 \vee \neg\phi_1)\mathcal{U}\phi_1$ and $\Box\phi_1 := \neg(\Diamond\neg\phi_1)$.

A *word* $\overline{\omega} = (\omega_0, \omega_1, \ldots)$ is a sequence such that each ω_n is a subset of $\{\lambda_n\}_{n=0}^{N_{\mathrm{R}}}$. For $m, n \in \mathbb{Z} \geqslant 0, n \geqslant m$, the word $(\omega_m, \omega_{m+1}, \ldots, \omega_n)$ is denote $\overline{\omega}_m^n$. The *satisfaction of an LTL formula* ϕ by the word $\overline{\omega}$ is recursively defined as: (1) $\overline{\omega}$ satisfies λ_n if $\lambda_n \in \omega_0$, (2) $\overline{\omega}$ satisfies $\neg\phi$ if $\overline{\omega}$ does not satisfy ϕ, (3) $\overline{\omega}$ satisfies $(\phi_1 \vee \phi_2)$ if $\overline{\omega}$ satisfies ϕ_1 and $\overline{\omega}$ satisfies ϕ_2, (4) $\overline{\omega}$ satisfies $(\phi_1 \mathcal{U} \phi_2)$ if there exists $n \geqslant 0$ such that $\overline{\omega}_n^\infty$ satisfies ϕ_2 and for every $m < n, \overline{\omega}_m^n$ satisfies ϕ_1.

Consider atomic propositions associated with each ROI λ_n, $n \in [N_{\mathrm{R}}]$, such that λ_n is `true` whenever a prespecified number of vehicles are simultaneously within the ROI.

Each route $\mathbf{v} = (v_0, v_1, \ldots) \in \mathcal{L}_{\mathcal{G}}$ defines a word $\overline{\omega}(\mathbf{v}) = (\omega_0, \omega_1, \ldots,)$ where $\omega_\ell := \{\lambda_n \mid v_\ell = c_n\}$. The route $\mathbf{v}$ is *satisfies* an LTL$_{-X}$ formula ϕ if the word $\overline{\omega}(\mathbf{v})$ satisfies ϕ.

Consider routes $\mathbf{v}_i \in \mathcal{L}_{\mathcal{G}}$, $i \in [N_{\mathrm{A}}]$, associated with the motion of each vehicle in the team. Each of these paths defines a word $\overline{\omega}_i(\mathbf{v}_i) = (\omega_{0,i}, \omega_{1,i}, \ldots)$, which we concatenate to define $\overline{\omega}(\mathbf{v}_1, \ldots, \mathbf{v}_{N_{\mathrm{A}}}) := (\omega_{0,1}, \omega_{0,2}, \ldots, \omega_{1,1}, \omega_{1,2}, \ldots)$. Here, the rule of "concatenation" is that $\omega_{\ell,i}$ appears before $\omega_{m,k}$ in $\overline{\omega}(\mathbf{v}_1, \ldots, \mathbf{v}_{N_{\mathrm{A}}})$ if $\ell < m$ or else if $\ell = m$ and $i < k$. The paths $(\mathbf{v}_1, \ldots, \mathbf{v}_{N_{\mathrm{A}}})$ collectively satisfy an LTL$_{-X}$ formula ϕ if $\overline{\omega}(\mathbf{v}_1, \ldots, \mathbf{v}_{N_{\mathrm{A}}})$ satisfies ϕ.

The desired LTL$_{-X}$ specification ϕ to be collaboratively satisfied by the team of vehicles is assumed to be a conjunction of a *safety* specification ϕ^{S} and a *liveness* specification ϕ^{L} in the standard form $\phi = \phi^{\mathrm{S}} \wedge \phi^{\mathrm{L}}$. A typical safety specification of interest to aerial vehicles is obstacle avoidance. Typical liveness specifications of interest to robotic vehicles are: (1) *Sequencing* specifications of the form $\diamond(\lambda_{n_1} \wedge \diamond(\lambda_{n_2} \wedge \diamond(\ldots \wedge \diamond\lambda_{n_\ell})))$, which involve visiting ROIs in a specific sequence, and (2) *Coverage* specifications of the form $\diamond\lambda_n$ or $\square\diamond\lambda_n$, which involve visiting ROIs once or infinitely often.

The safety specification ϕ^{S} is to be satisfied by *all* vehicles in the team. The liveness specification ϕ^{L} is to be satisfied collaboratively and is assumed to be a conjunction of N_{T} specifications, ϕ_j, $j \in [N_{\mathrm{T}}]$, i.e. $\phi^{\mathrm{L}} := \bigwedge_{j=1}^{N_{\mathrm{T}}} \phi_j$. Each ϕ_j is either a sequencing or coverage specification, i.e.,

$$\phi_j \in \{\diamond\lambda_{n_1}, \square\diamond\lambda_{n_1}, \diamond(\lambda_{n_1} \wedge \diamond(\lambda_{n_2} \wedge \diamond(\ldots \wedge \diamond\lambda_{n_\ell})))\},$$

with $n_1, n_2, \ldots \in [N_{\mathrm{R}}]$. The proposed approach is to treat each specification ϕ_j as a "task" to be assigned to a vehicle. The specifications ϕ_j are assumed to be either *independent*, i.e. it can be satisfied by one vehicle acting independently of all others, or *collaborative*, i.e. its satisfaction requires collaboration among multiple vehicles. Note that the preceding definition of atomic propositions λ_n allows for such collaborative specifications to be formulated. Without loss of generality, the collaborative tasks are labeled $1, \ldots, N_{\mathrm{TC}}$ and the independent tasks are labeled $N_{\mathrm{TC}} + 1, \ldots, N_{\mathrm{T}}$. The number of vehicles required to satisfy the j^{th} collaborative specification is

denoted M_j. In what follows, the terms 'task j' and 'specification ϕ_j' are synonymously used.

2.2 Problem Formation

The main problem of interest is formulated as follows.

Problem 1. *Given an LTL$_{-X}$ specification ϕ over Λ, find routes $(\mathbf{v}_1, \ldots, \mathbf{v}_{N_\mathrm{A}})$ with each $\mathbf{v}_i \in \mathcal{L}_\mathcal{G}$ such that the word $\overline{\omega}(\mathbf{v}_1, \ldots, \mathbf{v}_{N_\mathrm{A}})$ satisfies ϕ.* □

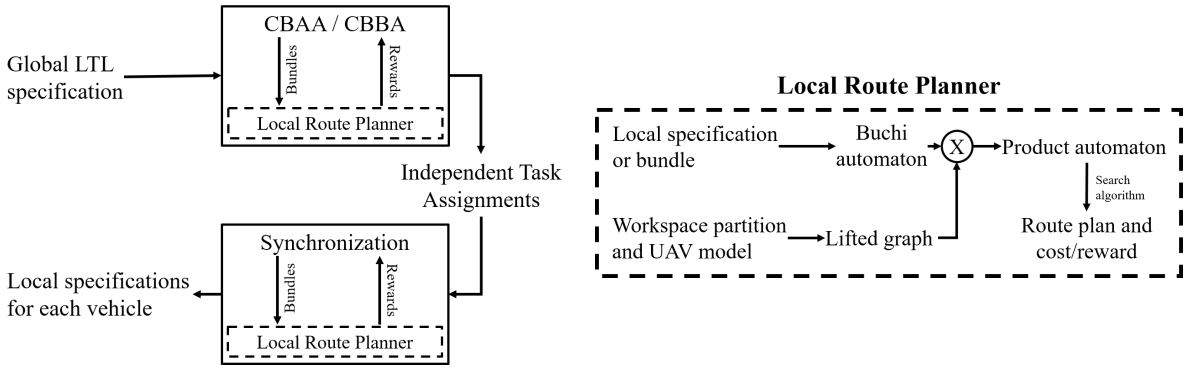
We seek a *decentralized* solution to Problem 1 where agent $i \in [N_\mathrm{A}]$ computes the path $\mathbf{v}_i$. Agents are assumed to communicate with each other over a static network, where the set of neighbors of agent $i \in [N_\mathrm{A}]$ is denoted $\mathcal{N}_i$. We require that the routes solving Problem 1 be traversable by the vehicles.

2.3 Decentralized Route Planning

The proposed method for solving Problem 1 is to first find local specifications ψ_i such that $\phi^\mathrm{L} = \bigwedge_{i=1}^{N_\mathrm{A}} \psi_i$. The specification to be satisfied by agent i is then given by $\phi^\mathrm{S} \wedge \psi_i$ for each $i \in [N_\mathrm{A}]$. Next, each agent i executes a route-planning algorithm to find a kinematically feasible route $\mathbf{v}_i$ satisfying $\phi^\mathrm{S} \wedge \psi_i$. To find routes traversable by the KCPM, we discuss the method of *lifted graphs* in §2.1.1. This method computes the *H-cost* of a route $\mathbf{v}$, denoted $\bar{\mathcal{J}}(\mathbf{v})$. Routes with a finite H-cost are guaranteed to be KCPM-traversable [33]. Fig. 2.1 conceptually illustrates the proposed approach.

2.3.1 Lifted Graph

To consider vehicle's kinematic constraint, we discuss the method of lifted graph. For nonholonomic vehicle models, edge transitions in $\mathcal{G}$ (i.e., transitions between successive cells) are vehicle-state-dependent, especially when the cell dimensions are comparable to vehicle maneuvering characteristics such as the minimum radius of turn. Furthermore, it is not possible to design



(a) Decomposition of global LTL specification into local specifications for each vehicle.

(b) Route-planning for each vehicle.

Figure 2.1: Conceptual illustration of proposed approach.

control laws that guarantee transitions between adjacent cells from *every* initial state; such transitions may not be possible owing to the controllability properties of the vehicle model discussed in §2.1.1.

To incorporate some information in $\mathcal{G}$ about the vehicles' physical motion, we discuss the *method of lifted graphs* [82]. Consider the workspace cell decomposition graph $\mathcal{G} = (V, E)$, and for every integer $H \geqslant 0$, define

$$V_H := \{(v_0, \ldots, v_H) : (v_{k-1}, v_k) \in E,\ k = 1, \ldots, H, v_k \neq v_m,\ \text{ for } k, m \in \{0, \ldots, H\},\ \text{with } k \neq m\}\,.$$

Every element $\mathbf{i} \in V_H$ is an ordered $(H+1)$-tuple of the elements of V, and this tuple corresponds to a sequence of successively adjacent cells. We denote by $[\mathbf{i}]_k$ the k^{th} element of $\mathbf{i}$, for $k < m \leqslant H+1$. Let E_H be a set of all pairs $(\mathbf{i}, \mathbf{j})$, with $\mathbf{i}, \mathbf{j} \in V_H$, such that $[\mathbf{i}]_k = [\mathbf{j}]_{k-1}$, for every $k = 2, \ldots, H+1$, and $[\mathbf{i}]_1 \neq [\mathbf{j}]_{H+1}$. The *lifted graph* $\mathcal{G}_H$ is defined as the directed graph (V_H, E_H). Every path $\mathbf{v} = (v_0, v_1, \ldots)$ in the graph $\mathcal{G}$ can be uniquely mapped to a path $\overline{\mathbf{h}} = (\mathbf{i}_0, \mathbf{i}_1, \ldots)$ in $\mathcal{G}_H$, where $\mathbf{i}_k = (v_k, v_{k+1}, \ldots, v_{k+H}) \in V_H$ for each $k \in \mathbb{N}$. We refer to the construction of $\mathcal{G}_H$ from $\mathcal{G}$ as *lifting* of $\mathcal{G}$.

The primary benefit of lifting $\mathcal{G}$ is that edge transitions costs in $\mathcal{G}_H$ can encode characteristics of vehicles' workspace traversal in accordance with its kinematic and dynamic constraints. One example of such edge transition costs is the following, which associates certain forward- and backward reachability properties of the vehicle model with edge transitions in $\mathcal{G}_H$.

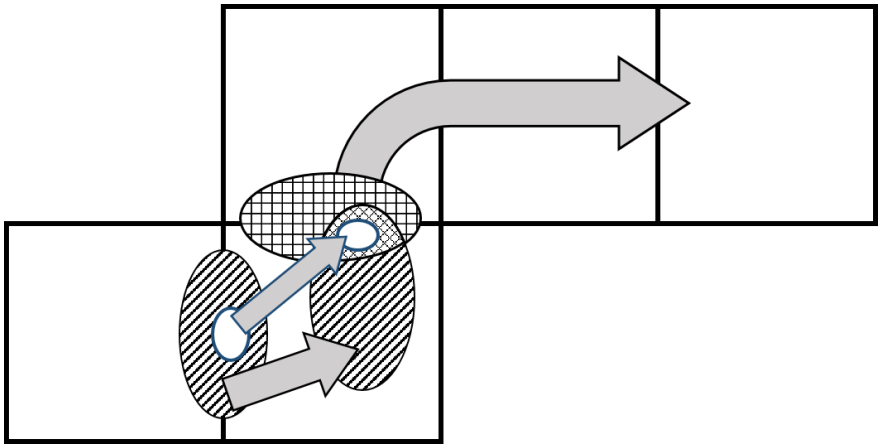


Figure 2.2: Illustration (with $H = 3$) of edge cost assignment in the lifted graph.

Consider an element $(\mathbf{i}, \mathbf{j}) \in E_H$, with $\mathbf{i}, \mathbf{j} \in V_H$. Let $\mathcal{S}(\mathbf{i}) \subset \mathcal{D}$ be a set of states associated with $\mathbf{i} \in V_H$ such that $\mathsf{x}\mathcal{S}(\mathbf{i}) \subseteq \mathsf{cell}([\mathbf{i}]_1) \cap \mathsf{cell}([\mathbf{i}]_2)$. Thus, the position components of the elements in $\mathcal{S}(\mathbf{i})$ lie on the boundary between the first and second cells corresponding to the vertices of V that constitute the ordered H-tuple $\mathbf{i}$. Next let $\mathcal{Q}(\mathbf{j}) \subset \mathcal{D}$ be a set of states such that $\mathsf{x}\mathcal{Q}(\mathbf{j}) \subseteq \mathsf{cell}([\mathbf{j}]_1) \cap \mathsf{cell}([\mathbf{j}]_2)$ and for every state $\xi_\mathrm{q} \in \mathcal{Q}(\mathbf{j})$ there exists a traversal time t_q and an admissible control input $u_\mathrm{q} \in \mathcal{U}_{t_\mathrm{q}}$ such that $\mathsf{x}\xi(t; \xi_\mathrm{q}, u_\mathrm{q}) \in \bigcup_{k=1}^{H+1} \mathsf{cell}([\mathbf{j}]_k)$, for all $t \in [0, t_\mathrm{q}]$. Informally, $\mathcal{Q}(\mathbf{j})$ is the set of all states whose position components lie on the boundary between the first and second cells of $\mathbf{j}$, and such that the traversal of the geometric region defined by the cells associated with the tuple $\mathbf{j}$ is possible from any initial state within $\mathcal{Q}(J)$.

Next, let $\mathcal{R}_\mathbf{i} : \mathcal{S}(\mathbf{i}) \to 2^\mathcal{D}$ be a reachability map associated with the sets $\mathcal{S}(\mathbf{i})$, defined by

$$\mathcal{R}_\mathbf{i}(\xi_\mathrm{s}) := \Big\{ \xi_\mathrm{t} \in \mathcal{D} \mid \xi_\mathrm{t} \in \cup_{t \in \mathbb{R}_+} \cup_{u \in \mathcal{U}_t} \xi(t; \xi_\mathrm{s}, u), \text{ and } (\cup_{\tau \in [0,t]} \mathsf{x}\xi(\tau; \xi_\mathrm{s}, u)) \cap (\mathcal{W} \backslash \mathsf{cell}([\mathbf{i}]_2)) = \varnothing \Big\}, \quad (2.3)$$

where $\xi_\mathrm{s} \in \mathcal{S}(\mathbf{i})$ and $2^\mathcal{D}$ is the collection of all subsets of $\mathcal{D}$. Informally, $\mathcal{R}_\mathbf{i}(\xi_\mathrm{s})$ is the set of all states that can be reached from ξ_s by trajectories whose position components always remain within the second cell of $\mathbf{i}$. Finally, for $(\mathbf{i}, \mathbf{j}) \in E_H$, define $\hat{\mathcal{S}}(\mathbf{i}, \mathbf{j}) := \{\xi_\mathrm{s} \in \mathcal{S}(\mathbf{i}) : \mathcal{R}_\mathbf{i}(\xi_\mathrm{s}) \cap \mathcal{Q}(\mathbf{j}) \neq \varnothing\}$. (see Fig. 2.2)

Now consider a path $\mathbf{v} = \{v_0, v_1, \ldots\} \in \mathcal{L}_\mathcal{G}$. For $k \in \mathbb{N}$, let $\mathbf{i}_k := (v_k, \ldots, v_{k+H})$. Clearly,

$(\mathbf{i}_k, \mathbf{i}_{k+1}) \in E_H$. We define $g_H : E_H \to \mathbb{R}_+$ and $\mathcal{S}(\cdot)$ as follows:

$$\mathcal{S}(\mathbf{i}_{k+1}) := \bigcup_{\xi_s \in \hat{\mathcal{S}}(\mathbf{i}_k, \mathbf{i}_{k+1})} \left(\mathcal{R}_{\mathbf{i}_k}(\xi_s) \cap \mathcal{Q}(\mathbf{i}_{k+1})\right), \tag{2.4}$$

$$g_H(\mathbf{i}_k, \mathbf{i}_{k+1}) := \begin{cases} \chi, & \text{if } \mathcal{S}(\mathbf{i}_{k+1}) = \varnothing, \\ 1, & \text{otherwise,} \end{cases} \tag{2.5}$$

where $\chi \gg 1$. The transition cost of $(\mathbf{i}_k, \mathbf{i}_{k+1}) \in E_H$ is $g_H(\mathbf{i}_k, \mathbf{i}_{k+1})$. Finally, the *$H$-cost* of a path $\mathbf{v} \in \mathcal{L}_\mathcal{G}$ is:

$$\bar{\mathcal{J}}[H](\mathbf{v}) := H + \sum_{k=0}^{P-H} g_H(\mathbf{i}_k, \mathbf{i}_{k+1}). \tag{2.6}$$

Algorithms for computing the sets $\mathcal{R}(\cdot), \mathcal{S}(\cdot)$, and $\mathcal{Q}(\cdot)$ are described, based on geometric arguments, in [34]. The following observation is crucial.

Proposition 1 ([83]). *Let $\mathbf{v} = (v_0, \ldots, v_P)$ be a path with length less than χ in $\mathcal{L}_\mathcal{G}$, and let $\xi_0 \in \mathcal{D}$ be prespecified such that $\mathsf{x}\xi_0 \in \mathsf{cell}(v_0) \cap \mathsf{cell}(v_1)$. Then $\mathbf{v} \in \mathcal{L}_\Gamma(\xi_0)$ if and only if $\bar{\mathcal{J}}[H](\mathbf{v}) < \chi$.*

The lifted graph $\mathcal{G}_H$ with the preceding edge transition costs is a finite state model for an individual vehicle's motion. Proposition 1 characterizes the paths in $\mathcal{G}$ that can be feasibly traversed while satisfying the vehicle's kinematic and dynamic constraints.

In the light of Proposition 1, route-planning for a single vehicle to satisfy a LTL$_{-X}$ specification ϕ is implemented using Dijkstra's search algorithm on a product transition system. This product transition system is constructed by computing the product of the lifted graph with the Büchi automaton associated with the specification ϕ (see Fig. 2.1(b)). Additional details on this route-planning algorithm are available in a recent work by the second and third authors of this paper [83].

The availability of this route-planning algorithm for each vehicle then reduces Problem 1 to finding an appropriate decomposition of the global liveness specification ϕ^{liveness} into local specifications ψ_i, $i = 1, \ldots, N_\text{A}$, such that $\phi^{\text{liveness}} = \bigwedge_{i=1}^{N_\text{A}} \psi_i$. The specification to be satisfied by the i^{th} vehicle is then given by $\phi^{\text{safety}} \wedge \psi_i$. In the following sections, we discuss such a decomposition

of the global liveness specification.

2.3.2 Consensus-Based Bundle Algorithm

To achieve an efficient collaboration among the team of vehicle to satisfy the given global LTL specification, the technique of consensus-based bundle algorithm is implemented to decompose the global task into local tasks for each vehicle in the team. A static communication network among the team of vehicles is modeled as an undirected graph with unit edges. We define the communication network diameter D as the maximum length of existing path. For example, if agent i and k are connected in the graph, then message can be exchanged between these two agents.

As previously stated, the global LTL specification is assumed to be a conjunction of N_{TI} independent specifications and N_{TC} dependent specifications. We address first the decomposition of the independent specifications into local specifications for each vehicle. To this end, we consider each of the N_{TI} independent specifications to be a "task" to be assigned to vehicles, and apply task assignment algorithms from the literature [84]. For the reader's convenience, a brief overview of the relevant algorithms, namely, the *consensus-based auction algorithm* (CBAA) and the *consensus-based bundle algorithm* (CBBA) is presented next.

These algorithms attempt to maximize a total *reward,* which is computed as the sum of reward gained by each vehicle. The reward, denoted by r_{ij}, quantifies the incentive for the i^{th} vehicle for performing the j^{th} task. In the proposed approach, we associate the reward with the distance traveled by vehicles (i.e., higher rewards for lower distances), as precisely defined later.

CBAA is applicable when $N_{\mathrm{TI}} = N_{\mathrm{A}}$, and each vehicle is to be assigned at most one task. CBAA achieves decentralized task assignment by an iterative process consisting of an auction phase and a consensus phase. In the auction phase, each vehicle bids for tasks asynchronously and generates a list of winning bids. A centralized auctioneer is not required [84]. In the first iteration, each vehicle bids for a task with maximum reward. Next, in the consensus phase, these lists of winning bids are communicated among neighboring agents. If the i^{th} vehicle is outbid for its desired task j_i then the vehicle tries to bid on other tasks by decreasing its bid for the desired task j_i by the amount $\eta_i^1 - \eta_i^2 + \varepsilon$ so that an agreement can be reached among vehicles, where η_i^1

and η_i^2 are the maximum and second-to-maximum rewards in the current iteration, and $0 < \varepsilon \leqslant \frac{1}{N_\mathrm{A}}$ is a small positive constant. The iterative convergence of CBAA is guaranteed [84], and the value of ε influences the rate of convergence.

CBBA is applicable in the general case when $N_\mathrm{TI} \neq N_\mathrm{A}$ and each vehicle can perform more than one task. Instead of individual tasks, CBBA addresses *task bundles* and *task paths*: a task bundle is a list of tasks, and a task path associated with a task bundle is an ordered sequence of tasks in that bundle. We denote by $\mathbf{b}_i$ and $\mathbf{p}_i$ a task bundle and task path, respectively, for the i^th vehicle. Note that a task path is different from a path in the workspace $\mathcal{W}$ or its associated cell decomposition graph $\mathcal{G}$, as defined in §2.1.2.

CBBA operates in two phases, similar to CBAA, an auction phase followed by a consensus phase. In the auction phase, each vehicle bids on a task bundle instead of individual tasks. Accordingly, rewards are associated with task bundles and task paths instead of individual tasks, as defined below. Each vehicle updates its task bundle and task path as follows:

$$\mathbf{b}_i = \mathbf{b}_i \oplus_\mathrm{end} j_i, \qquad\qquad \mathbf{p}_i = \mathbf{p}_i \oplus_{n(j_i)} j_i,$$

where j_i is the desired task, and $n(j_i)$ is the index position in the path $\mathbf{p}_i$ at which this task is inserted to provide the largest reward. The symbol $\oplus$ denotes insertion of an element into a list, and its subscript indicates the index position of insertion.

In the consensus phase of CBBA, three parameters are exchanged and updated among neighboring vehicles: the highest bid list, the winning agent list for each task, and the time stamp of exchanging information from neighbors. If the i^th vehicle is outbid on its desired task j_i, it releases not only this task, but all tasks in its current bundle that were added after the task j_i. The convergence to conflict-free task assignments of the iterative auction and consensus phases in CBBA is guaranteed if the reward r_{ij} satisfies a *diminishing marginal gain* (DMG) condition [84]: namely, that the reward of a task j does not increase as other tasks are added into the bundle before it:

$$r_{ij}[\mathbf{b}_i] \geqslant r_{ij}[\mathbf{b}_i \oplus_\mathrm{end} \mathbf{b}]. \tag{2.7}$$

for all $i \in \{1, \ldots, N_\mathrm{A}\}$ and $j \in \mathcal{N}^\mathrm{I} = \{j_1, j_2, \ldots, j_{N_\mathrm{TI}}\}$. In what follows, we define the rewards

for the route-planning problem of interest, and comment on the satisfaction of the DMG condition.

2.3.3 Application of CBAA/CBBA to Route-Planning

We discuss here the proposed decentralized route-planning technique to solve Problem 1 with independent task assignments, i.e., for the case only when $N_{\mathrm{T}} = N_{\mathrm{TI}}$. Assignments of dependent tasks is discussed in the next section.

Recall that the j^{th} task in the context of the proposed approach is uniquely associated with the LTL specification ϕ_j, for each $j \in \mathcal{N}^{\mathrm{I}} = \{j_1, j_2, \ldots, j_{N_{\mathrm{TI}}}\}$. A task path $\mathbf{p}_i$ is an ordered sequence of tasks, and is therefore uniquely associated with a new local LTL specification. To be precise, consider a task path $\mathbf{p}_i = (j_1, j_2, \ldots, j_N)$, for $N \leqslant N_{\mathrm{T}}$. We associate with $\mathbf{p}_i$ the local specification $\psi(\mathbf{p}_i)$ defined by

$$\psi(\mathbf{p}_i) := \Diamond(\phi_{j_1} \wedge \Diamond(\phi_{j_2} \wedge \Diamond(\ldots \wedge \Diamond\phi_{j_N}))), \tag{2.8}$$

which encodes the ordered sequence of tasks in $\mathbf{p}_i$. This specification $\psi(\mathbf{p}_i)$ provides a way to associate a task path with a vehicle route using the previously discussed local route-planning discussed in §2.3.1. The result of this route-planning algorithm is a path in $\mathcal{G}$ denoted $\mathbf{v}(\mathbf{p}_i)$.

A crucial innovation of the proposed technique is that the reward functions r_{ij} for a given task path $\mathbf{p}_i$, as computed by the i^{th} vehicle during the execution of CBBA, is determined based on the H-cost of the path $\mathbf{v}(\mathbf{p}_i)$ in $\mathcal{G}$. This step ensures that the execution of the task path is kinematically feasible for the vehicle. To be precise, the reward function is recursively defined as:

$$r_{ij} = \bar{r}_j - \min_{n \leqslant |\mathbf{p}_i|} \bar{\mathcal{J}}[H](\mathbf{v}(\mathbf{p}_i \oplus_n j)), \tag{2.9}$$

where $\bar{r}_j$ is a sufficiently large prespecified positive constant. Recall that $\mathbf{p}_i \oplus_n j$ denotes a task path obtained by inserting the j^{th} task at the n^{th} position in task path $\mathbf{p}_i$.

The definition of r_{ij} shown in (2.9) satisfies the DMG condition, because the marginal increase in H-cost $\bar{\mathcal{J}}[H](\mathbf{v}(\mathbf{p}_i \oplus_n j)) - \bar{\mathcal{J}}[H](\mathbf{v}(\mathbf{p}_i))$ is nonnegative for any n. Therefore, the convergent property of the CBBA algorithm is retained.

Chapter 3

Route-planning for Multi-Vehicle System: Collaborative Tasks

In Chapter 2, we proposed a decentralized route-planning algorithm to decompose the given LTL specification into local specifications among multi-vehicle system. We assumed that the global LTL specification is a conjunction of independent tasks which is assigned to a vehicle. However, in some practical applications, the collaboration among multiple vehicles to satisfy a complex task is required. To address these situations, another type of task is consider in this chaper: collaborative task. For the collaborative task, multiple vehicles are required to work together simultaneously or each collaborative task is assigned to multiple vehicles. The assignment of collaborative task to multiple vehicles are of practical relevance. For example, in a search-and-rescue mission, two or more vehicles, each equipped with different sensors, may be required to visit and search the same region at the same time. In a warehouse application, two or more vehicles may be required to jointly transport heavy cargo. Same workspace decomposition and vehicle model as discussed in Chapter 2 are considered for the route-planning for the collaborative tasks.

Similar assumption is also made here: we treat each specification ϕ_j as a "task" to ba assigned to a vehicle. Differ from the scenario considered in Chapter 2, here we assume the specifications ϕ_j is either *independent*, i.e. it can be satisfied by one vehicle acting independently of all others, or *collaborative*, i.e. its satisfaction requires collaboration among multiple vehicles. Note that the preceding definition of atomic propositions λ_n allows for such collaborative specifications to be formulated. Without loss of generality, the collaborative tasks are labeled $1, \ldots, N_{\mathrm{TC}}$ and the independent tasks are labeled $N_{\mathrm{TC}}+1, \ldots, N_{\mathrm{T}}$. The number of vehicles required to satisfy the j^{th} collaborative specification is denoted M_j. In what follows, the terms 'task j' and 'specification ϕ_j' are synonymously used.

For the reader's convenience, a nomenclature table is provided in Table 3.1.

Symbol	Meaning
N_{TC}	Number of collaborative tasks.
N_{T}	Total number of tasks.
M_j	Number of vehicles required to accomplish the j^{th} collaborative task.
$\mathcal{F}_i$	Reward matrix for vehicle i.
$\mathbf{f}_{ij}$	j^{th} row of reward matrix for vehicle i.
f_{ijk}	Vehicle i's reward knowledge of task j for vehicle k.
$\mathcal{G}_i$	Assignment matrix for vehicle i.
$\mathbf{g}_{ij}$	j^{th} row of assignment matrix for vehicle i.
g_{ijk}	Vehicle i's assignment knowledge of task j for vehicle k.
$\mathbf{u}_{ij}^*$	Optimal assignment for task j based on vehicle i's knowledge.
$\tau_{\mathbf{u}_{ij}^*,j}$	Waiting time required by the group $\mathbf{u}_{ij}^*$ for task j.
n_{last}	Index of last collaborative task in the task path for vehicle i.
n_j^*	Optimal insert position for task j.
h_{ij}	Task j's availability for vehicle i.
$\bar{r}_j$	Reward constant for task j in Line 9 in Fig. 3.1.
s_{ik}	Latest communication iteration between vehicle i and k.

Table 3.1: Parameters for decentralized route-planning - Collaborative task.

3.1 Problem Formation

Similar problem of interest discussed in Problem 1 in Chapter 2 is considered again. Now we assume the given LTL specification contains both independent and collaborative tasks. We seek a *decentralized* solution to the problem where agent $i \in [N_{\mathrm{A}}]$ computes the path $\mathbf{v}_i$. The number of vehicles required to work together to satisfy each collaborative task is given a prior and the assigned vehicles need to arrive at specified location simultaneously with minimum total waiting time. The group of agents are assumed to communicate with each other over a static network, where the set of neighbors of agent $i \in [N_{\mathrm{A}}]$ is denoted $\mathcal{N}_i$. We require that the routes solving Problem 1 be traversable by the vehicles.

3.2 Decentralized Synchronization Route-Planning

In this section, we consider a more general decentralized route-planning problem for multi-vehicle system where both independent and collaborative tasks are specified. We propose a de-

centralized algorithm which achieves a task assignment with minimum total waiting. For the sake of numerical comparison, we also propose a new extension of the consensus-based group algorithm [79] to satisfy LTL specifications with kinematically feasible routes.

3.2.1 Consensus-Based Grouping Algorithm

The method of CBBA is not capable of assigning the same (collaborative) task to multiple agents. In the general case where i.e., $N_{\mathrm{TC}} > 0$, not only are such multi-agent assignments required, but the temporal synchronization of agents to achieve collaborative tasks is also required. One possible approach to the solution of this problem is based on the so-called *consensus-based grouping algorithm* (CBGA, [79]). Note that [79] does not refer to LTL; the application of CBGA to satisfying LTL formulae is an extension that we propose for the sake of comparison, referred by the acronym E-CBGA.

The CBGA operates in phases of bundle construction, auction, and consensus. Each agent maintains matrices with information about the reward and task assignment as known by all the other agents. These matrices are exchanged and updated during the consensus phase.

The E-CBGA approach to the solution of Problem 1 is then to apply the CBGA to the set of specifications $[N_{\mathrm{T}}]$ with rewards defined by H-costs as in (2.9). Each agent i is assigned a task path $\mathbf{p}_i$, and the local liveness specification ψ_i is found per (2.8). Because the CBGA has no temporal considerations, multi-agent assignments of collaborative specifications must be synchronized. For example, if two task paths $\mathbf{p}_i$ and $\mathbf{p}_k$ include $j \in [N_{\mathrm{TC}}]$, then the agents i and k are to visit an ROI simultaneously. One of the agents, say i, will arrive "early"; to synchronize the two agents' visits, agent i must wait at this ROI until agent k arrives.

Because such synchronization is performed a posteriori in the E-CBGA approach, the sum of the waiting durations of all agents during the execution of the overall global specification can be large. To address this issue, we propose a novel approach where the synchronization of tasks is addressed during task assignment. Consequently, the proposed approach achieves assignments with minimum waiting durations.

Note that the E-CBGA approach is itself a contribution of this work to the literature, in that it

ensures that kinematic traversability of routes found using the CBGA.

3.2.2 Decentralized Synchronization Algorithm

In the general case of Problem 1 when $N_{\mathrm{TC}} > 0$, we first execute the CBBA to assign independent tasks among the agents, and then execute the proposed algorithm to assign collaborative tasks.

The proposed algorithm iterates over a bundle construction phase and a consensus phase as described in Figs. 3.1–3.3. The iteration counter is denoted t. Each agent i maintains and updates at each iteration a task bundle and task path, a reward matrix, an assignment matrix, a set of agents assigned to collaborate on each task $j \in [N_{\mathrm{TC}}]$, and the waiting duration required for synchronization. The reward and assignment matrices are denoted by $\mathcal{F}_i(t)$ and $\mathcal{G}_i(t)$, the j^{th} row of these matrices by $\mathbf{f}_{ij}(t)$ and $\mathbf{g}_{ij}(t)$, the element in the j^{th} row and k^{th} column by $f_{ijk}(t)$ and $g_{ijk}(t)$, respectively. An availability flag for task j is denoted $h_{ij}(t)$. The optimal group of agents for task j based on agent i's current knowledge is denoted by $\mathbf{u}^*_{ij}(t)$ and the corresponding waiting durations for the group is defined as $\tau_{\mathbf{u}^*_{ij},j}(t)$. Note that $\mathcal{F}_i(t), \mathcal{G}_i(t) \in \mathbb{R}^{N_{\mathrm{T}} \times N_{\mathrm{A}}}$ at each iteration $t \in \mathbb{N}$.

The element $f_{ijk}(t)$ is the i^{th} agent's current knowledge of the reward for agent k for task j at iteration t. The element $g_{ijk}(t) = 1$ if agent i agrees to assign the task j to agent k. The flag $h_{ij}(t) = 1$ (resp., $h_{ij}(t) = 0$) if task j is available to agent i (resp., not available).

The proposed algorithm is initialized with $\mathbf{b}_i(0)$ and $\mathbf{p}_i(0)$ with task indices resulting from the execution of CBBA. The matrices $\mathcal{F}_i(0)$ and $\mathcal{G}_i(0)$ are set to zero.

At each iteration $t \in \mathbb{N}$, let $n_{\mathrm{last}}(t)$ denote the sequential position in $\mathbf{p}_i(t)$ of the last collaborative task. At $t = 1$, when no collaborative tasks are yet assigned, $n_{\mathrm{last}}(1) = 0$. In the bundle construction phase, each agent i updates $\mathbf{b}_i(t)$, $\mathbf{p}_i(t)$, $\mathcal{F}_i(t)$, and $\mathcal{G}_i(t)$ by optimally inserting available tasks into $\mathbf{b}_i(t)$ and $\mathbf{p}_i(t)$. To this end, for each $j \in [N_{\mathrm{TC}}]$ that is not present in $\mathbf{b}_i(t)$, an optimal position for insertion n^*_j is computed (Line 8 in Fig. 3.1). Note that sequential positions only *after* $n_{\mathrm{last}}(t)$ are considered, due to which rewards for tasks already assigned do not change due to insertion of task j.

Bundle Construction Phase at Iteration $t \geqslant 1$

1: $\mathbf{b}_i(t) = \mathbf{b}_i(t-1), \quad \mathbf{p}_i(t) = \mathbf{p}_i(t-1),$
2: $\mathcal{F}_i(t) = \mathcal{F}_i(t-1), \quad \mathcal{G}_i(t) = \mathcal{G}_i(t-1)$
3: $\mathbf{u}_i^*(t) = \mathbf{u}_i^*(t-1), \quad \tau_{\mathbf{u}_i^*}(t) = \tau_{\mathbf{u}_i^*}(t-1)$
4: $h_{ij}(t) = 0$, for each $j \in [N_{\mathrm{TC}}]$
5: $n_{\mathrm{last}}(t) := \max\{n_j | j \in [N_{\mathrm{TC}}] \cap \mathbf{p}_i(t)\}$
6: **while** $|\mathbf{b}_i(t)| < N_{\mathrm{T}}$ **do**
7: **for** $j \in [N_{\mathrm{TC}}] \setminus \mathbf{b}_i(t)$ **do**
8: $n_j^* := \arg\max_{n_{\mathrm{last}}(t) \leq n \leq |\mathbf{p}_i|} r_{ij}$
9: $f_{iji}(t) := \bar{r}_j - \bar{\mathcal{J}}(\mathbf{v}(\mathbf{p}_i \oplus_{n_j^*} j|_j))$
10: $\{\mathbf{u}_j, \tau_j\} = \Delta_{ij}$
11: **if** $\|\mathbf{g}_{ij}(t)\|_1 < M_j$ **then**
12: $h_{ij}(t) = 1$
13: **else**
14: **if** $\tau_j < \tau_{\mathbf{u}_{ij}^*, j}(t)$ and $i \in \mathbf{u}_j, \quad h_{ij}(t) = 1$
15: $f_{iji}(t) = f_{iji}(t) h_{ij}(t)$
16: $j^* := \arg\max_{j \in [N_{\mathrm{TC}}] \setminus \mathbf{b}_i(t)} r_{ij} h_{ij}(t),$
17: $\mathbf{b}_i(t) := \mathbf{b}_i(t) \oplus_{\mathrm{end}} j^*, \quad \mathbf{p}_i(t) := \mathbf{p}_i(t) \oplus_{n_{j^*}^*} j^*$
18: $\mathbf{u}_{ij^*}^*(t) = \mathbf{u}_{j^*}, \quad \tau_{\mathbf{u}_{ij^*}^*, j^*}(t) = \tau_{j^*}$
19: $g_{ij^*i}(t) = 1$

Figure 3.1: Bundle construction phase of the proposed algorithm.

The reward $f_{iji}(t)$ is updated based on the traversal cost $\bar{\mathcal{J}}(\mathbf{v}(\mathbf{p}_i \oplus_{n_j^*} j|_j))$ of partial completion of $\mathbf{p}_i \oplus_{n_j^*} j$ (Line 9 in Fig. 3.1). By partial completion we mean the sequential execution of tasks in $\mathbf{p}_i$ until the $n_j^{*\text{th}}$ task. The set of agents $\mathbf{u}_j$ assigned to collaborative task j and the agents' waiting duration τ_j are updated per the calculations Δ_{ij} described in Fig. 3.2. For each collaborative task j, if an insufficient number of agents are assigned, i.e., if $\|\mathbf{g}_{ij}(t)\|_1 < M_j$, then the availability flag is set to $h_{ij}(t) = 1$ (Line 12 in Fig. 3.1). If $\|\mathbf{g}_{ij}(t)\|_1 \geqslant M_j$, a sufficient number of agents are currently assigned to task j, but this assignment may not be optimal in terms of the agents' total waiting duration $\tau_{\mathbf{u}_{ij}^*, j}(t)$. Therefore, agent i can collaborate on this task if the waiting duration τ_j is lower than $\tau_{\mathbf{u}_{ij}^*, j}(t)$ (Line 14 in Fig. 3.1).

After updating the task availability flags $h_{ij}(t)$, the element $f_{iji}(t)$ of the reward matrix is updated (Line 15 in Fig. 3.1) such that the rewards for non-available tasks are reset to zero. Task j^* with maximum reward is found (Line 16 in Fig. 3.1) and inserted into $\mathbf{p}_i(t)$ at the optimal sequential position $n_{j^*}^*$. Next, the set of agents $\mathbf{u}_{ij^*}^*(t)$ and the minimal waiting duration $\tau_{\mathbf{u}_{ij^*}^*, j^*}(t)$ is

Waiting Duration Calculation Δ_{ij}

Input: $\mathcal{F}_i,\ \mathcal{G}_i$ **Output:** $\mathbf{u}^*_{ij}, \tau_{\mathbf{u}^*_{ij},j}$

1: $\mathbf{u}^*_{ij}(t) := \mathbf{u}^*_{ij}(t-1), \quad \tau_{\mathbf{u}^*_{ij},j}(t) = \tau_{\mathbf{u}^*_{ij},j}(t-1)$
2: **for** $k \in N_{\mathrm{A}}$ such that $f_{ijk} > 0$ **do**
3: Find $\mathbf{u} \subset [N_{\mathrm{TC}}]$ such that $|\mathbf{u}| = M_j - 1$ and $\sum_{m\in\mathbf{u}}(f_{ijm}(t) - f_{ijk}(t))^2$ is minimal.
4: $\tau := \max_{m\in\mathbf{u}}\{f_{ijm}(t)\} - \min_{m\in\mathbf{u}}\{f_{ijm}(t)\}$
5: **if** $\tau < \tau_{\mathbf{u}^*_{ij},j}(t)$ **then**
6: $\tau_{\mathbf{u}^*_{ij},j}(t) := \tau, \quad \mathbf{u}^*_{ij}(t) := \mathbf{u}$
7: Return $\mathbf{u}^*_{ij}(t), \tau_{\mathbf{u}^*_{ij},j}(t)$

Figure 3.2: Waiting duration calculation for task j by agent i.

updated (Line 18 in Fig. 3.1).

In the waiting duration calculations Δ_{ij}, the agent i attempts to form a team of M_j agents to collaborate on task j such that the team's total waiting duration is minimal. To this end, each agent k with a nonzero reward for task j is identified (Line 2 in Fig. 3.2) and $M_j - 1$ "teammates" for this agent are found (Line 3). These teammates for agent k own the $|M_j - 1|$ closest rewards to agent $k's$ reward f_{ijk}.

For the set $\mathbf{u}$ of agents, the maximum waiting duration is the time difference between the arrivals of the first and last agents at the ROI. In this paper, we assume that 1 time unit is taken to move between two adjacent cells, and therefore the arrival time for each agent in $\mathbf{u}$ can be computed based on length of its path in the workspace. The time for agent i to start task j is $\bar{r}_j - f_{iji}$. The larger the reward f_{iji}, earlier the agent i arrives at the ROI associated with task j. Thus, the maximum waiting duration in the set $\mathbf{u}$ of agents is the difference between the maximum and minimum reward (Line 4 in Fig. 3.2). The set $\mathbf{u}^*_j(t)$ with the least maximum waiting duration is then considered the optimal assignment of M_j agents to task j (Line 6).

In the consensus phase, agents exchange and update their knowledge. Each agent i maintains a timestamp $s_{ik}(t)$ of the last communication with each neighboring agent $m \in \mathcal{N}_i$. When agent i communicates with a neighboring agent k, it updates the elements of the reward matrix $\mathcal{F}_i(t)$ if agent k has more recent information. Specifically, if for any $m \in [N_{\mathrm{A}}]$, $s_{km}(t) > s_{im}(t)$, then agent k has communicated with agent m more recently than agent i (Line 3 in Fig. 3.3). The set

Consensus Phase At Iteration $t \geqslant 1$

1: **for** $j \in [N_{\mathrm{TC}}]$ **do**
2: **for** $k \in \mathcal{N}_i$ **do**
3: **if** $s_{km}(t) > s_{im}(t), \forall m \in N_{\mathrm{A}}$ **then**
4: $f_{ijm}(t) := f_{kjm}(t), \quad s_{im}(t) := s_{km}(t)$
5: $\{\mathbf{u}^*_{ij}(t), \tau_{\mathbf{u}^*_{ij},j}(t)\} = \Delta_{ij}$
6: **if** $i \notin \mathbf{u}^*_{ij}(t)$ **then**
7: Remove task j from $\mathbf{b}_i(t)$ and $\mathbf{p}_i$
8: $f_{iji}(t) = 0, \quad g_{iji}(t) = 0$

Figure 3.3: Consensus phase of the proposed algorithm.

of agents $\mathbf{u}^*_{ij}(t)$ and the waiting duration $\tau_{\mathbf{u}^*_{ij},j}(t)$ is recalculated (Line 5 in Fig. 3.3). If agent i is outbid by other agents for task j, then the task j is removed from $\mathbf{b}_i$ and $\mathbf{p}_i$ and $f_{iji}(t)$ and $g_{iji}(t)$ are reset to zero (Line 8 in Fig. 3.3).

3.2.3 Convergence and Computational Complexity

The finite termination properties of the proposed algorithm are summarized in the following two results.

Lemma 1. *At any iteration* $t \geqslant 1$, *suppose there exists a task index* $j^\dagger \in [N_{\mathrm{TC}}]$ *such that* $j^\dagger$ *maximizes* $f_{iji}(t)$ *for each* $i \in \mathbf{u}^*_{j^\dagger}(t)$. *Then* $\mathbf{u}^*_{j^\dagger}(s) = \mathbf{u}^*_{j^\dagger}(t)$ *for all* $s \geqslant t$.

Proof. If an agent i is assigned to collaborative task $j^\dagger$, let $\tau^i_{j^\dagger}$ denote the duration for which agent i is required to wait to synchronize with other agents assigned to this task. By definition in Fig. 3.2,

$$\tau^i_{j^\dagger}(t) = \begin{cases} \max\{\mathbf{f}_{ij^\dagger}(t)\} - f_{ij^\dagger i}(t), & \text{if } f_{ij^\dagger i} < \max\{\mathbf{f}_{ij^\dagger}(t)\}, \\ 0, & \text{otherwise.} \end{cases}$$

For any set of agents $\mathbf{u} \subset [N_{\mathrm{A}}]$ with $|\mathbf{u}| = M_{j^\dagger}$, including the set $\mathbf{u}^*_{j^\dagger}$, note that the maximum waiting duration (defined in Lines 4–6 of Fig. 3.2) satisfies

$$\tau_{\mathbf{u},j^\dagger}(t) = \max\left\{\max_{i\in\mathbf{u}}\{\tau^i_{j^\dagger}(t)\}, \ \ \tau_{\mathbf{u},j^\dagger}(t-1)\right\}. \tag{3.1}$$

It follows that $\tau_{\mathbf{u},j^\dagger}(t) \leqslant \tau_{\mathbf{u},j^\dagger}(s)$ for every $s \geqslant t$, i.e., the maximum waiting duration for any set of agents assigned to the collaborative task $j^\dagger$ does not increase over the algorithm's iterations. As shown in [85], the property (3.1) also implies that for any two sets $\mathbf{u}_1, \mathbf{u}_2 \subset [N_\mathrm{A}]$ with $|\mathbf{u}_1| = |\mathbf{u}_2| = M_{j^\dagger}$,

$$\tau_{\mathbf{u}_1,j^\dagger}(t) \leqslant \tau_{\mathbf{u}_2,j^\dagger}(t) \Leftrightarrow \tau_{\mathbf{u}_1,j^\dagger}(s) \leqslant \tau_{\mathbf{u}_2,j^\dagger}(s) \text{ for every } s \geqslant t.$$

If there exists $j^\dagger \in [N_\mathrm{TC}]$ as in the statement of the Lemma, then $\tau_{\mathbf{u}^*_{j^\dagger},j^\dagger}(t) \leqslant \tau_{\mathbf{u},j^\dagger}(t)$, for every $\mathbf{u} \subset [N_\mathrm{A}]$ with $|\mathbf{u}| = M_j$ and $t \in \mathbb{N}$, which means that the maximum waiting duration for $\mathbf{u}^*_{j^\dagger}$ always remains the least among all possible sets that can be assigned to task $j^\dagger$, and the Lemma follows. □

Proposition 2. *Let δ be the diameter of the agents' communication network graph. Then the proposed algorithm terminates in at most $\frac{1}{2}N_\mathrm{TC}(N_\mathrm{TC}+1)\delta$ iterations.*

Proof. According to the consensus protocol in Line 4 of Fig. 3.3, at most $N_\mathrm{TC}\delta$ iterations after bundle construction are required for all agents to reach consensus on the reward matrix, i.e., for some iteration $t \leqslant N_\mathrm{TC}\delta$,

$$f_{ij^\dagger m}(t) = f_{\ell j^\dagger m}(t), \quad i,\ell,m \in [N_\mathrm{A}] \text{ and } j^\dagger \in [N_\mathrm{TC}]. \tag{3.2}$$

After consensus, $\mathbf{u}^*_{j^\dagger}$ is found in Line 5 in Fig. 3.3. By (3.2), the condition in the statement of Lemma 1 is satisfied for some $j^\dagger_1 \in [N_\mathrm{TC}]$, and $\mathbf{u}^*_{j^\dagger_1}$ does not change after iteration t. Similarly, after at most $(N_\mathrm{TC}-1)\delta$ further iterations, another task $j^\dagger_2 \in [N_\mathrm{TC}]\backslash\{j^\dagger_1\}$ meets the condition in the statement of Lemma 1, and consequently $\mathbf{u}^*_{j^\dagger_2}$ is fixed. Continuing further so on, it follows that in at most $\frac{1}{2}N_\mathrm{TC}(N_\mathrm{TC}+1)\delta$ iterations, $\mathbf{u}^*_{j^\dagger}$ is fixed for all $j^\dagger \in [N_\mathrm{TC}]$, and the algorithm terminates. □

At each iteration of the proposed algorithm, the only nontrivial computation is of the reward in Line 9 in Fig. 3.1, which involves route-planning to satisfy an LTL formula. Per [33], the worst-case complexity of this computation is $\mathcal{O}\left[|\mathcal{G}|N_\mathrm{T}4^H\left(4+\log(|\mathcal{G}|N_\mathrm{T}4^H)\right)\right]$. Therefore, by Prop. 1, the worst-case computational complexity of each agent is $\mathcal{O}\left[\delta N^2_\mathrm{TC}|\mathcal{G}|N_\mathrm{T}4^H\left(4+\log(|\mathcal{G}|N_\mathrm{T}4^H)\right)\right]$.

This result shows a significant innovation over the state-of-the-art because the computational complexity does not depend on the number of agents, simply due to decentralization.

In conclusion, we proposed a decentralized route-planning algorithm to enable a networked mobile vehicles to satisfy certain given LTL specifications. Two different types of tasks: Independent task and collaborative task are addressed in Chapter 2 and Chapter 3 separately. Compared to the existing literature provides either centralized algorithms or otherwise assumes individual vehicle specifications a priori, the proposed algorithm includes the decomposition of the global LTL specification into local specifications for each vehicle. Minimum turn radius constraints on the vehicles' motion, which are ignored in the existing literature, are addressed in the proposed approach by the lifted graph algorithm. Specifically, H-costs defined on the lifted graphs are used to define rewards for agents during decentralized task assignment. The proposed algorithm achieves task assignments that result in significantly reduced waiting durations for the team, as compared to assignments made by an easy extension of the existing literature (E-CBGA).

Chapter 4

Interactive Planning and Sensing

In Chapter 2 and Chapter 3, we discuss decentralized route-planning algorithms for the multi-vehicle system and enable the team of vehicles to work collaboratively to satisfy the given global mission. However, as we mentioned above, the route-planning relies on perfect knowledge of the environment which may not be available in some practical applications. To consider the problem of route-planning in an unknown/uncertain environment, we remove the assumption of accurate knowledge of environment in this chapter. To simplify the problem, we decompose it into two sub-problems: unknown environment exploration and route-planning to satisfy the given LTL specification by multi-vehicle system. We consider a heterogeneous multi-vehicle team where one set of vehicles called *sensors* are deployed to explore/map the environment, while the other set called *actors* are deployed to carry out intelligent collaborative tasks based on the map generated by the sensors. However, we are facing a "chicken-and-egg" paradox: we can't compute the optimal routes for actors to satisfy the global LTL specification because we don't have accurate information of the environment. We don't know where to put sensors because we don't have optimal routes for actors. To solve this "chicken-and-egg" paradox, we propose a new technique for task-driven sensor placement technique that enables interactions between the actors and sensors networks. At each iteration, the technique decides a set of "informative" subregions to take measurements and used them to update the environment. Then the decentralized route-planning approach discussed in Chapter 2 and Chapter 3 can be implemented to compute the optimal routes for *actors* to satisfy the given mission based on latest environment information. The conceptual diagram of the interactive planning and sensing approach is shown in Fig. 4.1. The loop between the actors and sensors network will not terminate until certain convergence condition is satisfied. To analyze the benefit of proposed algorithm, we compare it to a typical information driven sensing algorithm on convergence rate, optimality etc.

For the reader's convenience, a nomenclature table is provided in Table 4.1.

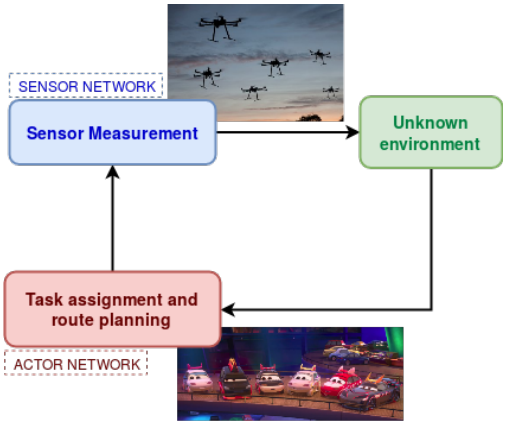


Figure 4.1: Diagram of interactive planning and sensing.

Symbol	Meaning
$\mathcal{W}$	2D environment.
N^{C}	Number of cells.
N^{A}	Number of actors.
N^{S}	Number of sensors.
$\mathcal{G}, V, E$	POM with vertices V and edges E.
$\Omega(v_i)$	Occupancy probability of vertex v_i.
$\bar{w}$	Constant penalty in Eqn. (4.1).
$H(\mathbf{v})$	Entropy of route $\mathbf{v}$.
$\mathcal{W}_i'$	Subregion of interest computed in Eqn. (4.5).
$\mathcal{W}_i''$	Subregion with non-zero information gain computed in Eqn. (4.12).
$\mathbf{S}_i$	Sensors positions converged by MINMISS.
ϵ	Threshold for terminate condition in Eqn. (4.4).
$I(v_i)$	Information gain of vertex v_i.

Table 4.1: Parameters for interactive planning and sensing.

4.1 Problem Overview

4.1.1 Probabilistic Occupancy Map

All vehicles collaboratively operate in an environment $\mathcal{W} \in \mathbb{R}^2$, which is partitioned into N^{C} uniform *cells*. Associated with this partition is a grid map graph $\mathcal{G} := (V, E)$ such that each vertex of $\mathcal{G}$ is uniquely associated with a cell, and each edge of $\mathcal{G}$ is uniquely associated with a pair of adjacent cells. The obstacle-occupancy probability of cell v is $\Omega(v) := p(v \text{ is occupied})$. We assume cell occupancy is mutually independent, and the probability of occupancy of the entire

map is $\Omega(\mathcal{W}) = \prod_{v \in V} \Omega(v)$. The graph $\mathcal{G}$ and the occupancy probabilities $\Omega(v)$ together define the POM; in a minor abuse of notation we denote the POM and the graph, both, by $\mathcal{G}$ and cells and vertices, both, by v. We assume that the environment is static, i.e., the true cell occupancies do not change.

A *route* $\mathbf{v}$ in $\mathcal{G}$ is a sequence $(v_0, v_1, \ldots)$ of vertices, such that $v_\ell \in V$, and $(v_{\ell-1}, v_\ell) \in E$, for each $\ell \in \mathbb{N}$. For actors, the cost of a route is the sum of transition costs w of all edges in the route, defined by:

$$w(v_\ell, v_m) := 1 - \Omega(v_m) + \bar{w}\Omega(v_m). \tag{4.1}$$

Here $\bar{w}$ is a large prespecified constant that penalizes the route cost when the vertex v_m is occupied. Note that the route cost is a discrete random variable with entropy

$$H(\mathbf{v}) = \sum_{v_\ell \in \mathbf{v}} - (\Omega(v_\ell) \log \Omega(v_\ell) + \Omega'(v_\ell) \log \Omega'(v_\ell)), \tag{4.2}$$

where $\Omega'(v_\ell) := 1 - \Omega(v_\ell)$. If the map is perfectly known, i.e., each $\Omega(v_\ell)$ is either 0 or 1, then $H(\mathbf{v}) = 0$. For sensor routes, alluding to the search-and-rescue motivating example, we assume that the sensors do not face obstacles (e.g., by flying overhead), and the transition cost is unity for all edges.

4.1.2 Binary Sensing Model

A network of mobile vehicles with N^{A} actors and N^{S} sensors is considered. The vehicles are labeled by integers $1, \ldots, N^{\mathrm{A}}, \ldots N^{\mathrm{A}} + N^{\mathrm{S}}$, and we denote the index sets $[N^{\mathrm{A}}] := \{1, \ldots, N^{\mathrm{A}}\}$ and $[N^{\mathrm{S}}] := \{N^{\mathrm{A}} + 1, \ldots, N^{\mathrm{A}} + N^{\mathrm{S}}\}$. Sensors can sense the occupancy of cell v_ℓ, and measure either $z_\ell = 1$ for "occupied" and $z_\ell = 0$ for "unoccupied. For simplicity, we assume that measurements are taken at selected cells, but *not along the routes* traveled by the sensors. In practice this assumption reflects limited onboard data storage space and situations when each measurement consumes time and energy, e.g., on-board image processing computations required to detect obstacles from camera images. A binary sensing model shown in Fig. 4.2 is considered for the sensors.

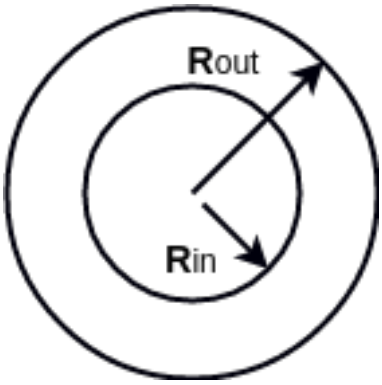


Figure 4.2: Binary Sensing Model.

For some practical sensing models, the detection range varies in different environment conditions, i.e., relative distance between the target and the sensor. As shown in Fig. 4.2, the target will be detected correctly within an inner disk. Certain probability of accuracy is guaranteed by sensors if the target is in an annulus between the inner and outer disk. No detection is returned from sensors if the targets are outside of the outer disk. In our work, we set the detection model for a sensor placed at cell v_m as:

$$p(z_\ell = 1|v_m) = \begin{cases} 1, & \text{if } v_\ell = v_m,\ v_\ell \text{ occupied} \\ 0.8, & \text{if } (v_\ell, v_m) \in E,\ v_\ell \text{ occupied} \\ 0.2, & \text{if } (v_\ell, v_m) \in E,\ v_\ell \text{ free} \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

This detection model returns an accurate measurement if the target and sensor are overlapped and returns the measurement with 80% accuracy if the target is inside of sensor's sensing range.

For the group of sensors, we are looking for optimal routes $(\mathbf{v}_1, \ldots, \mathbf{v}_{N^{\mathrm{S}}})$ to take measurements at specified locations which are computed by the group of actors to accomplish the given intelligent task. Here, we assume each sensor in the group are exactly same and the optimal routes for the group of sensors to take all requested measurements can be computed via the approach described in Chapter 2.

4.2 Interactive Route-Planning And Sensing

In this section, we propose a technique to find actor and sensor routes iteratively and is summarized in Fig. 4.3. Each actor $i \in [N^{\mathrm{A}}]$ maintains a copy of the POM $\mathcal{G}$, which is initialized with $\Omega(v_m) = 0.5$ for all $v_m \in V$. The techniques introduced in Chapter 2 and Chapter 3 are implemented to decompose the global LTL specification ϕ and achieve the route-planning for the group of actors in the team, which we leverage here to first compute an optimal route $\mathbf{v}_i$ given the current POM (Line 1 in Fig. 4.3). Next, two subregions $\mathcal{W}_i'$ and $\mathcal{W}_i''$ related to this route are computed (Lines 3-4). The definitions of these subregions are provided in §4.2.1 and 4.2.2. A *task-driven information gain* is then computed and a set of sensor placements $\mathbf{S}_i \in \mathcal{W}$ is selected by each actor (Lines 3-4). The actors reach consensus $\mathbf{S}^* \in \mathcal{W}$ on the sensor placements, i.e., N^{S} cells with the highest task-driven information gain. Optimal routes $\mathbf{v}_j$ are then computed for each sensor $j \in [N^{\mathrm{S}}]$ to minimize the total distance traveled (Line 7 in Fig. 4.3). Using the new measurements from sensors placed at $\mathbf{S}^*$, each actor updates its POM, recomputes the optimal route $\mathbf{v}_i$, and the iteration repeats (Lines 8-9).

The algorithm termination condition *StopCondition* is evaluated with current routes $\mathbf{v}_i,\ i \in [N^{\mathrm{A}}]$. This condition is true when the entropy of each route $\mathbf{v}_i$ is small enough. Specifically, *StopCondition* is true whenever

$$H(\mathbf{v}_i) \leqslant \varepsilon |\mathbf{v}_i|, \quad \text{for each } i \in [N^{\mathrm{A}}]. \tag{4.4}$$

The constant ε in (4.4) is set to $\varepsilon = 0.0243$, which ensures that the average probability of occupancy of all vertices along $\mathbf{v}_i$ is 0.01. The termination criterion (4.4) encodes a confidence threshold in the actors' route costs, i.e., it ensures that the uncertainty in these route costs is sufficiently small.

The following subsections provide details of the steps involved in the iterative technique in Fig. 4.3.

Iterative route-planning and sensor placement

Find $\mathbf{v}_i$ based on $\mathcal{G}_i$ using technique discussed in Chapter 2 and Chapter 3
while $\neg$ *StopCondition* **do**
  Find $\mathcal{W}_i'$ (see Eqn. (4.5))
  Find $\mathcal{W}_i''$ by (4.12) and the task-driven information gain $I(v_m)$ for each $v_m \in \mathcal{W}_i''$ (see Fig. 4.4)
  Select $\mathbf{S}_i$ as described in §4.2.3
  Reach an agreement $\mathbf{S}^*$ over team of vehicles
  Compute optimal route for sensor j using technique discussed in Chapter 2 and Chapter 3
  Update $\mathcal{G}_i$ based on sensors measurements
  Recompute $\mathbf{v}_i$ based on $\mathcal{G}_i$ using technique discussed in Chapter 2 and Chapter 3
return $[\mathbf{v}_i]$

Figure 4.3: Pseudo code for the proposed interactive route-planning and sensor placement, for each actor $i \in [N^{\mathrm{A}}]$, and each sensor $j \in [N^{\mathrm{S}}]$.

4.2.1 Task-Driven Subregion of Interest

For each actor's route $\mathbf{v}_i$, a local subregion of interest $\mathcal{W}_i' \subset \mathcal{W}$ is defined as the set of cells with uncertain occupancy and lying "near" the route. Specifically, "nearness" is defined as follows. Let $\mathcal{N}_i := \{v_\ell \in \mathbf{v}_i \mid 0 < \Omega(v_\ell) < 1\}$ be the set of cells with uncertain occupancy in the route $\mathbf{v}_i$. Consider the hypothetical "worst-case" scenario that sensors placed at each cell in $v_m \in \mathcal{N}_i$ measure $z_m = 1$. Then consider a hypothetically update to the POM (see §4.2.2), which may be used to recompute a route $\mathbf{v}_i'$. We define

$$\mathcal{W}_i' := \mathcal{N}_i \cup \{v_\ell \in \mathbf{v}_i' \mid 0 < \Omega(v_\ell) < 1\} \tag{4.5}$$

as the subregion of interest to actor i.

4.2.2 Task-Driven Information Gain

In this section we define a task-driven information gain to be computed for every cell $v_\ell \in \mathcal{W}_i'$. Restricting the computation of this information gain to the subregion $\mathcal{W}_i'$ is a crucial innovation of this paper in enabling interactions between sensor placement and route-planning. We also briefly

describe Bayesian updates of the POM using sensors measurements; the reader interested is referred to [61] for a thorough discussion on Bayesian updates of POMs.

We define the entropy of subregion $\mathcal{W}_i'$ by

$$H(\mathcal{W}_i') := \sum_{v_\ell \in \mathcal{W}_i'} -(p(v_\ell)\log p(v_\ell) + p'(v_\ell)\log p'(v_\ell)). \tag{4.6}$$

The *task-driven information gain* $I(v_\ell)$ about the subregion $\mathcal{W}_i'$ at iteration t is the expected reduction of map uncertainty in $\mathcal{W}_i'$ by a sensor measurement at cell $v_\ell \in \mathcal{W}$. To be precise:

$$\begin{aligned} I(v_\ell) :=& H(\mathcal{W}_i'|\mathbf{X}_{1:t-1}, \mathbf{Z}_{1:t-1}) \\ & -E[H(\mathcal{W}_i'|\mathbf{X}_{1:t-1}, \mathbf{Z}_{1:t-1}, x(t), z(t))]. \end{aligned} \tag{4.7}$$

Here, $H(\mathcal{W}_i'|\mathbf{X}_{1:t-1}, \mathbf{Z}_{1:t-1})$ is the entropy of $\mathcal{W}_i'$ conditioned on sensor locations $\mathbf{X}_{1:t-1}$ and measurements $\mathbf{Z}_{1:t-1}$ over the previous iterations $1, \ldots, t-1$. The term $E[H(\mathcal{W}_i'|\mathbf{X}_{1:t-1}, \mathbf{Z}_{1:t-1}, x(t), z(t))] = E[H(\mathcal{W}_i'|\mathbf{X}_{1:t}, \mathbf{Z}_{1:t})]$ is the expected entropy conditioned on $\mathbf{X}_{1:t-1}$, $\mathbf{Z}_{1:t-1}$, and the new sensor placement $x(t)$ and measurement $z(t)$. Per [63]:

$$\begin{aligned} & E[H(\mathcal{W}_i'|\mathbf{X}_{1:t}, \mathbf{Z}_{1:t})] \\ & = \textstyle\sum_{z(t)} p(z(t)|\mathbf{X}_{1:t}, \mathbf{Z}_{1:t-1}) H(\mathcal{W}_i'|\mathbf{X}_{1:t}, \mathbf{Z}_{1:t}), \end{aligned} \tag{4.8}$$

which is an average uncertainty of $\mathcal{W}_i'$ over all possible measurements. For each measurement $z(t)$, the *belief* of any $v_\ell \in \mathcal{W}_i'$ is defined by:

$$bel(v_\ell) := p(v_\ell \text{ is occ.} \mid \mathbf{X}_{1:t-1}, \mathbf{Z}_{1:t-1}, x(t), z(t)). \tag{4.9}$$

The posterior distribution before the new measurement is

$$\overline{bel}(v_\ell) := p(v_\ell \text{ is occ.} \mid \mathbf{X}_{1:t}, \mathbf{Z}_{1:t-1}). \tag{4.10}$$

Therefore, a Bayesian update of the belief is:

$$\begin{aligned} bel(v_\ell) &= \eta\, p\left(z(t)|x(t)\right)\ p(v_\ell \text{ is occ.} \mid \mathbf{X}_{1:t}, \mathbf{Z}_{1:t-1}) \\ &= \eta\, p\left(z(t)|x(t)\right)\ \overline{bel}(v_\ell). \end{aligned} \tag{4.11}$$

where $\eta^{-1} := p(z(t) \mid \mathbf{X}_{1:t}, \mathbf{Z}_{1:t-1})$ is a normalizing constant and $p(z(t)|x(t))$ is the sensor model (4.3). As is common in the literature, the new measurement $z(t)$ is assumed independent of $\mathbf{X}_{1:t-1}$ and $\mathbf{Z}_{1:t-1}$.

To compute the expected entropy of $\mathcal{W}_i'$ in (4.8), we follow the procedure outlined in Fig. 4.4. We identify a set of cells $\mathcal{M}$ with uncertain occupancy and within the sensor range when a sensor is placed at v_ℓ. The set of potential measurements that sensor can possibly return is Γ, e.g., for $\mathcal{M} = \{v_k, v_m\}$,

$$\begin{aligned} \Gamma = \{&\{z_k = \text{Occupied}, z_m = \text{Occupied}\}, \\ &\{z_k = \text{Occupied}, z_m = \text{Free}\}, \{z_k = \text{Free}, z_m = \text{Occupied}\}, \\ &\{z_k = \text{Free}, z_m = \text{Free}\}\}. \end{aligned}$$

For each $\gamma \in \Gamma$, the entropy of $\mathcal{W}_i'$ is computed as the sum of entropy of each vertex $v_i \in \mathcal{W}_i'$, by Eqn. (4.6). These beliefs of cells $v_m \in \mathcal{M}$ are updated in Lines 4–7 of Fig. 4.4. Because the environment is static, the predicted posterior of cell v_m is equal to its belief $bel(v_m)$ computed at the previous $t-1$ (Line 5). The normalizing constant corresponding to the measurement γ is computed by considering the two possibilities, namely, v_m is free or occupied. Finally, the belief of cell v_m is updated in Line 7. The entropy of $\mathcal{W}_i'$ conditioned on sensor placement at $x(t) = v_\ell$ and measurement $z(t) = \gamma$ at iteration t is then computed in Line 8. Summing up the expected entropy (Line 10) over all possible measurements γ returns $E[H(\mathcal{W}_i'|\mathbf{X}_{1:t}, \mathbf{Z}_{1:t})]$.

The Bayesian update described above may be familiar from the simultaneous localization and mapping literature [61]. The proposed innovation is in the definition of task-driven information gain. Specifically, the most "informative" cell is where the information gain is the highest. However, finding the information gain for *every cell* in the environment is computationally expensive. By contrast, the task-driven information gain in (4.7) defines information in the context of the cur-

Information Gain Computation

Compute $H(\mathcal{W}_i'|\mathbf{X}_{1:t-1}, \mathbf{Z}_{1:t-1})$ by Eqn. (4.6)
$\mathcal{M} := v_\ell \cup \{v_m \in \mathcal{W}_i' \mid (v_\ell, v_m) \in E,\ 0 < \Omega(v_m) < 1\}$
for $\gamma \in \Gamma$ **do**
  for $v_m \in \mathcal{M}$ **do**
    $\overline{bel}(v_m) = bel(v_m)$
    $\eta_m^{-1} := \sum_{v_m \text{ free, occ.}} p(z_m|v_\ell)\, \overline{bel}(v_m)$
    $bel(v_m) := \eta_m p(z_m|v_\ell)\, \overline{bel}(v_m)$
  Compute $H(\mathcal{W}_i'|\mathbf{X}_{1:t-1}, \mathbf{Z}_{1:t-1}, v_\ell, \gamma)$ by Eqn. (4.6)
  $\eta^{-1} = \prod_{v_m \in \mathcal{W}_i'} \eta_m^{-1}$
  $E[H(\mathcal{W}_i'|\mathbf{X}_{1:t}, \mathbf{Z}_{1:t})]$ += $\eta^{-1} H(\mathcal{W}_i'|\mathbf{X}_{1:t-1}, \mathbf{Z}_{1:t-1}, v_i, \omega)$

Figure 4.4: Pseudocode for calculating $I(v_i)$ at iteration t by vehicle i

rent actor routes. Because this task-driven information gain is concerned only with the subregion $\mathcal{W}_i'$, we restrict its computation to $\mathcal{W}_i''$ defined by

$$\mathcal{W}_i'' := \mathcal{W}_i' \cup \{v_m | (v_m, v_\ell) \in E \text{ for each } v_\ell \in \mathcal{W}_i'\}. \tag{4.12}$$

The subregion $\mathcal{W}_i''$ is such that at least one cell of $\mathcal{W}_i'$ is within range of a sensor placed anywhere in $\mathcal{W}_i''$. The size of $\mathcal{W}_i''$ depends on region covered by the current optimal route $\mathbf{v}_i$, and therefore $|\mathcal{W}_i''| \ll |\mathcal{W}|$. Furthermore, we provide a discussion of computational efficiency of proposed technique due to decentralization of all computations (including actor and sensor route-planning) and its scalability in Chapter 6.

4.2.3 Sensor Placement and Route-planning

After finding $I(v_\ell)$ for each vertex $v_\ell \in \mathcal{W}_i''$, each actor $i \in [N^{\mathrm{A}}]$ finds a set $\mathbf{S}_i$ of N^{S} sensor locations to maximize the total gained information. Given a sensing region with $\mathcal{K}$ cells, the cell with minimum overall miss probability is selected repeatedly by the MINMISS strategy [78] until N^{S} measurement locations are determined.

Initially, let $\mathbf{S}_i = \varnothing$ and let $\mathcal{K}$ be the set of cells with maximum information gain. If $|\mathcal{K}| > N^{\mathrm{S}}$,

then the sensor locations are found using the MINMISS strategy. Briefly, MINMISS is a greedy strategy that achieves maximum coverage of cells within range of each of the N^{S} sensors.

If $|\mathcal{K}| \leqslant N^{\mathrm{S}}$, all cells in $\mathcal{K}$ are selected as sensor locations. In this case the remaining $N^{\mathrm{S}} - |\mathcal{K}|$ measurements are taken at cells with the next largest information gains.

As previously stated, we assume that the sensors do not face obstacles (e.g., aerial vehicles flying above any terrestrial obstacles). Therefore, sensor route-planning is relatively easy. We implement decentralized sensor route-planning to move the sensors to new locations selected at each iteration of the proposed technique. This route-planning method has been described in Chapter 2 and Chapter 3, and minimizes the total distance traveled by sensors at each iteration.

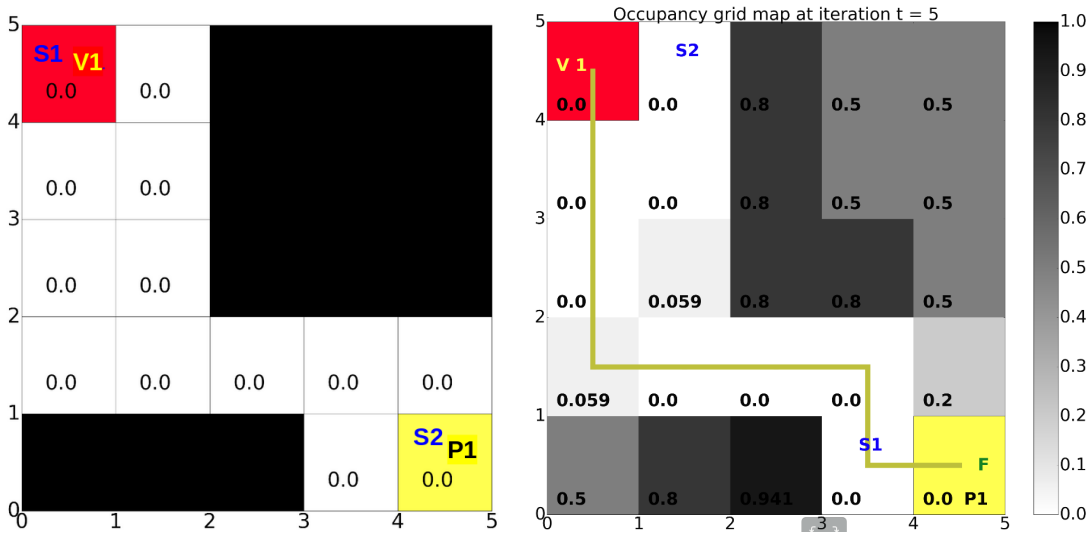
4.2.4 Convergence

The convergence of the proposed iterative technique is trivially proven as follows. At each iteration, the occupancies of cells at which sensors are placed are perfectly measured, i.e., the entropy of these cells becomes zero. Therefore these cells provide zero information gain in all future iterations. Therefore, the algorithm runs for at most $|\mathcal{W}|/N^{\mathrm{S}}$ iterations before a sensor has been placed at every cell in some prior iteration. Then the map is perfectly known, and *StopCondition* is satisfied, i.e., the iterations must stop.

4.2.5 Simple Example

To illustrate the proposed interactive planning and sensing algorithm in Fig. 4.3, we illustrate a simple case with $N^{\mathrm{A}} = 1$ actor and $N^{\mathrm{S}} = 2$ sensors in a 5×5 POM as shown in Fig. 4.5. The ground truth map is shown in Fig. 4.5(a). The initial probability of occupancy for each cell $\Omega(v_i)$, for $i = 0, \ldots, 24$, is indicated in Fig. 4.5(a). Initially, except for the base cell v_1 in red and ROI P_1 in yellow, the occupancy of all other cells is unknown, i.e. $\Omega(v_i) = 0.5$. Two sensors are initially located in the base cell and ROI P_1 respectively. The penalty $\bar{w}$ in the edge transition cost in (4.1) is set to 200.

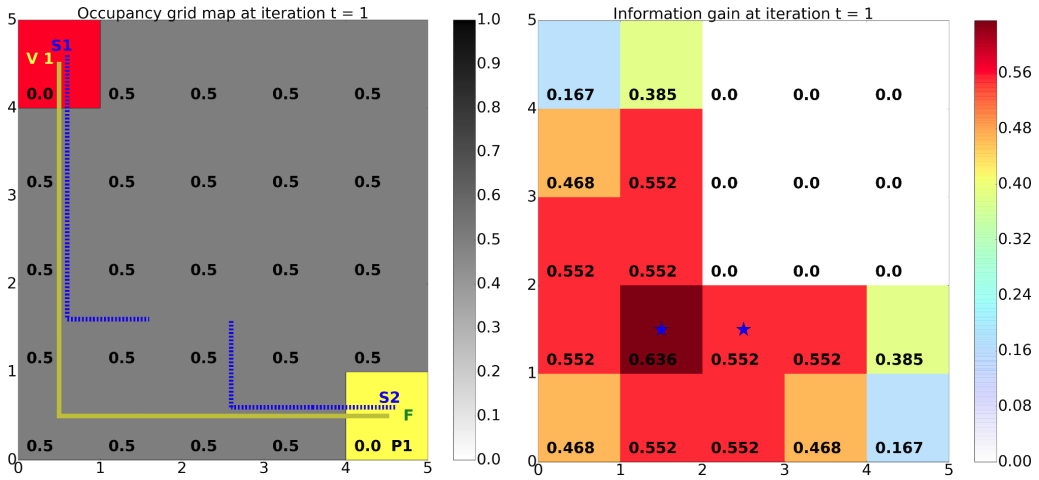
For this example, the proposed algorithm converges in four iterations shown in Fig. 4.6. At



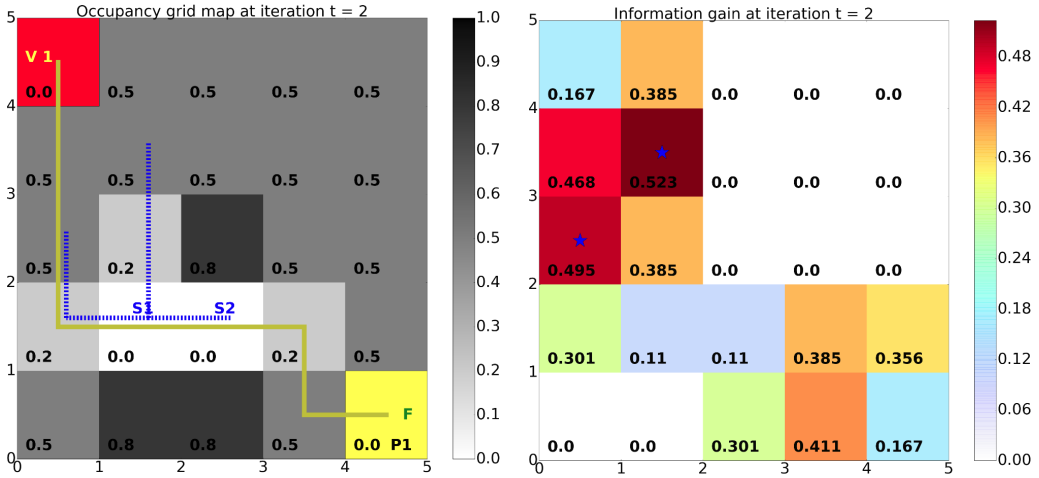
(a) True occupancy grid map. (b) Final optimal route and the POM.

Figure 4.5: Illustrative example with $N^{\mathrm{A}} = 1$ vehicle and $N^{\mathrm{S}} = 2$ sensors.

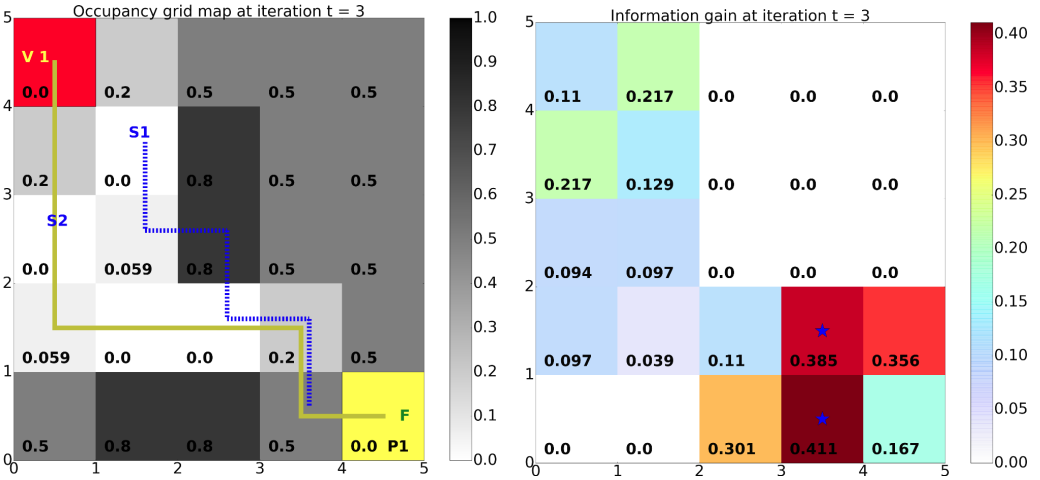
iteration $t = 1$, the optimal route for the actor vehicle to visit P_1 is the yellow line in Fig. 4.6(a). The task-driven information gain is indicated in Fig. 4.6(b), where the blue stars indicate desired sensor locations (cells with highest task-driven information gain). The routes planned for the two sensors to reach these locations are shown in dashed blue lines in Fig. 4.6(a). Using these measurements, the occupancy probability is updated as described in §4.2.2, and the iterations repeat. Shades of gray in the left column of Fig. 4.6 indicate occupancy probabilities (black = 1, white = 0). The optimal route and the knowledge of the occupancy grid map after convergence are shown in Fig. 4.5(b). Note that the occupancy probabilities in areas near the optimal route are near 0 or 1, i.e., it is known with near certainty whether these cells are occupied or free. By contrast, regions farther away (e.g., top right corner) have high entropy, i.e., there is greater uncertainty in the knowledge of occupancies of these cells.



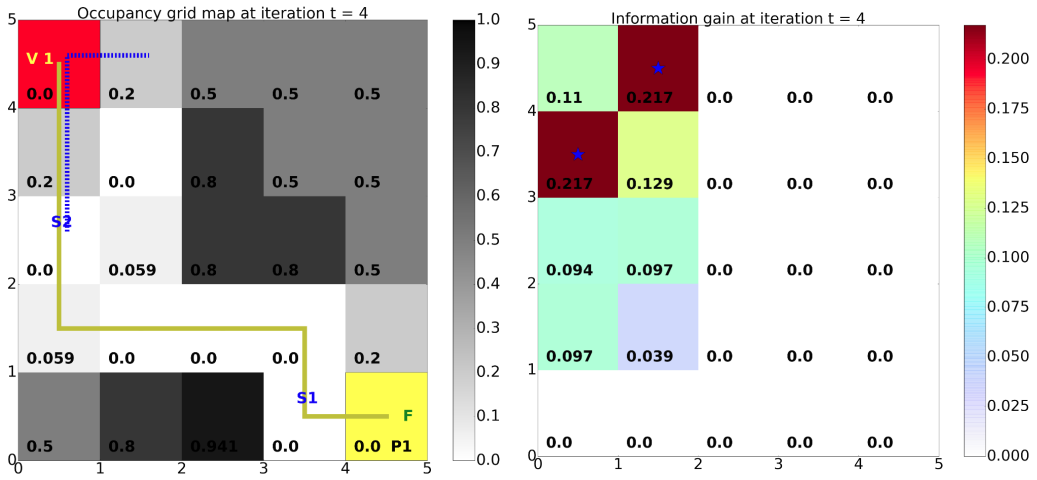
(a) Knowledge of $\mathcal{G}(t = 1)$. (b) Information gain at $t = 1$.



(c) Knowledge of $\mathcal{G}(t = 2)$. (d) Information gain at $t = 2$.



(e) Knowledge of $\mathcal{G}(t = 3)$. (f) Information gain at $t = 3$.



(g) Knowledge of $\mathcal{G}(t = 4)$. (h) Information gain at $t = 4$.

Figure 4.6: Grid map and information gain at iteration t = 1, 2, 3, 4

Chapter 5

Bayesian Optimization for Information Gain Computation

In this chapter, we address the computational efficiency of the interactive planning and sensing approach discussed in Chapter 4. To identify a set of "informative" subregions for measurement at each iteration, the proposed approach has to compute the information gain for each vertex in the environment. As shown in Fig. 4.4, it is difficult to derive a smooth function of information gain in terms of vertices. This prevents using first- and second-order derivative methods like gradient descent, and Newton's method to evaluate the maximum/minimum value of the information gain. To reduce the computational burden of calculating the information gain, we introduce a machine-learning-based technique: Bayesian Optimization and use it to model and find the maximum of the objective function which is expensive to evaluate. Bayesian optimization implements Gaussian process regression to model the objective function and uses an acquisition function to decide where to sample the points with optimal value of objective function.

5.1 Bayesian Optimization

5.1.1 Gaussian Process Regression

Gaussian process regression is a Bayesian statistical approach which can be implemented to model a "black-box" function. More specifically, we can use Gaussian processes to describe a distribution over functions. A Gaussian process can be specified by its mean and covariance function as

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{5.1}$$

The covariance function or kernel function $k(x, x')$ describes the similarity between the points x and x'. The closer the point x' is to x in the input space, the more confidence or less uncertainty there is for the prediction at point x'. In addition, the kernel functions are required to be positive semi-definite. A common kernel function, which is used in this work, is *power exponential* or *Gaussian* kernel which is defined as,

$$k(x, x') = \exp(-\frac{||x - x'||^2}{l}) \tag{5.2}$$

The parameter l refers the length-scale which informally can be thought of as the distance we have to move in input space before the function value can change significantly or how fast the objective function $f(x)$ changes with x.

The unknown parameter from the kernel function, e.g., l, is also called a *hyper-parameter*. For generality, we set the hyper-parameter as θ. The log marginal likelihood is defined as follows,

$$\log p(\mathbf{y}|X, \theta) = -\frac{1}{2}\mathbf{y}^T k^{-1}\mathbf{y} - -\frac{1}{2}\log|k| - \frac{n}{2}\log 2\pi \tag{5.3}$$

In our case, the random variables $\mathbf{y}$ represent the value of the function $f(x)$. The first component in Eqn. (5.3) indicates the data-fitting performance. The second component is the complexity penalty and the last component is the normalization constant.

The value of hyper-parameter θ can be computed by maximizing the log marginal likelihood. The partial derivative of the log marginal likelihood w.r.t. the hyper-parameter θ is derived as:

$$\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|X, \theta) = \frac{1}{2}\mathbf{y}k^{-1}\frac{\partial k}{\partial\theta_j}k^{-1}\mathbf{y} - \frac{1}{2}\text{tr}(k^{-1}\frac{\partial k}{\partial\theta_j}) \tag{5.4}$$

By setting the Eqn. (5.4) as 0, we can determine the value of hyper-parameter θ correspondingly. Based on the Eqn. (5.4), the computational complexity of hyper-parameter training comes from inverting the kernel function k. Standard matrix inversion of positive semi-definite symmetric requires time $\mathcal{O}(n^3)$ for inversion of an $n \times n$ matrix. In our case, the size of the training data dominants the computational complexity.

Once the hyperparameter θ is computed, the mean and covariance functions for the Gaussian

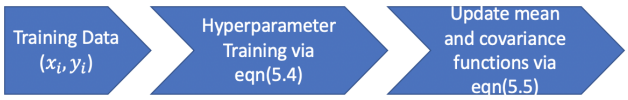


Figure 5.1: Diagram of Gaussian Process Regression.

distribution over function $f(x)$ can be derived as:

$$y_*|x_*, X, \mathbf{y} \sim \mathcal{N}(k(x_*, X)k(X, X)^{-1}\mathbf{y}, k(x_*, x_*) - k(x_*, X)k(X, X)^{-1}k(X, x_*)) \tag{5.5}$$

or the distribution of value $y_* = f(x_*)$ conditioned on the given training data $(X, \mathbf{y})$ can then be evaluated via Eqn. (5.5).

The diagram of how Gaussian process regression models function $f(x)$ with a set of training data $(X, \mathbf{y})$ and predicts the distribution for test samples x_* is shown in Fig. 5.1. In this work, we used the Gaussian process regression to model the function of information gain which has been defined in Eqn. (4.7) in Chapter 4 and shown as follows,

$$\begin{aligned} I(v_\ell) :=& H(\mathcal{W}_i'|\mathbf{X}_{1:t-1}, \mathbf{Z}_{1:t-1}) \\ & - E[H(\mathcal{W}_i'|\mathbf{X}_{1:t-1}, \mathbf{Z}_{1:t-1}, x(t), z(t))]. \end{aligned} \tag{5.6}$$

Refer to [86] for more details about modeling functions via Gaussian process regression. The objective here is to predict the vertices with maximum information gain or,

$$v^* = \text{argmax}_{v \in \mathcal{W}_i'} I(v) \tag{5.7}$$

5.1.2 Expected Improvement Acquisition Function

With the model computed by the Gaussian process regression discussed in §5.1.1, we address the second important component of Bayesian optimization. The acquisition function is used to decide the sampling points in the search space which should return the maximum value. Typically, the acquisition function captures the trade off in the problem of exploration and exploitation. For exploration, we sample the points with higher uncertainty or larger covariance value. For exploitation, the points with maximum prediction value or higher mean value are selected. The objective

of implementing an acquisition function is to determine a set of optimal sampling points to maximize the regression function. Some common acquisition functions for Bayesian optimization are *maximum probability of improvement*, *expected improvement*, and *upper confidence bound*.

The general steps of the Bayesian optimization procedure are shown as follows:

- Find the sampling point x_t by maximizing the acquisition function $u(x|D_{1:t-1})$ over the Gaussian process regression: $x_t = \arg\max_x u(x|D_{1:t-1})$.
- Evaluate y_t from the objective function $y_t = f(x_t)$.
- Add the sample point to previous samples set $D_{1:t} = \{D_{1:t-1}, (x_t, y_t)\}$ and update the Gaussian process model.

At each iteration t, with current sample points or training data $D_{1:t-1}$, we construct the model using Gaussian process regression. Then evaluate the acquisition function and select next sample points x_t which return maximum value of function. Next, we add the new sample points into the training data and update the corresponding Gaussian process model. Refer to [70] for more details about the Bayesian optimization.

In our case, the objective function we want to model is the function of information gain and the expected improvement is used as the acquisition function. At each iteration, we select the vertices for evaluating the information gain as the training data randomly. Gaussian process regression is implemented to model the information gain function based on current training data. Then a set of vertices with maximum expected improvement are identified as the sensor positions for the next iteration. The steps described above will be repeated to select each new set of "informative" subregions for environment exploration until the interactive planning and sensing algorithm reaches the convergence condition.

5.1.3 Proposed Algorithm with Bayesian Optimization

As discussed in Fig. 4.4, updating the information gain for each vertex $v_m \in \mathcal{W}_i''$ is computational expensive, especially when the environment is large. Recall the set $\mathcal{W}_i''$ computed in

Bayesian Optimization for Information Gain

1: Select random training data $[(x(v_i), I(v_i))]$.
2: Generate posterior probability distribution over function of information gain.
3: **while** $|\mathbf{S}_i| < N^{\mathrm{S}}$ **do**
4: $\quad v_i^* = arg\max_{v_i \in \mathcal{W}_i'' \setminus \mathbf{S}_i} EI(v_i)$.
5: $\quad$ Add v_i^* into $\mathbf{S}_i$

Figure 5.2: Pseudo code for computing information gain and sensor placement via Bayesian optimization

Eqn. (4.12) which refers to the set of vertices with non zero information gain. To select the sensors locations more efficiently at each iteration, we implement Bayesian optimization. As discussed in §5.1.1 and §5.1.2, without evaluating the information gain for all potential sensors locations, Bayesian optimization is able to identify a set of "informative" subregions with maximum information gain more efficiently. The pseudo code of combining the interactive planning and sensing algorithm with Bayesian optimization is shown in Fig. 5.2.

As shown in the algorithm, two phases are involved in the optimization: constructing the Bayesian model for information gain and using an acquisition function to determine the optimal sensors locations. In the model estimation phase, Gaussian process regression takes inputs $[(x(v_i), I(v_i))]$ as the training data to model the function of information gain. Here $x(v_i) = (H(v_i), \sum_{(v_i,v_j)\in E} H(v_j))$ is a set of entropy values related to the vertex v_i, and $I(v_i)$ is the actual information gain. Refer to [70] for more details on Gaussian process regression. At each iteration, a certain number of vertices are selected randomly as the training data to get the posterior distribution over the information gain function.

The expected improvement acquisition function is then implemented in the second phase and used to select the optimal sensor locations with maximum expected improvement of information gain. With the Bayesian model developed in the first phase, the expected improvement of vertex v_i is defined as:

$$EI(v_i) = E[\max\{I(v_i) - I^*, 0\}] \tag{5.8}$$

where $I(v_i)$ is the information gain of vertex v_i evaluated by the Bayesian model and I^* is the current maximum information gain. As shown in [70], the expected improvement of vertex v_i can

be computed as:

$$EI(v_i) = \begin{cases} Z(v_i)\Phi(\frac{Z(v_i)}{\sigma(v_i)}) + \sigma(v_i)\phi(\frac{Z(v_i)}{\sigma(v_i)}), & \sigma(v_i) > 0 \\ 0, & \sigma(v_i) = 0 \end{cases} \tag{5.9}$$

where $Z(v_i) = \mu(v_i) - I^* - \zeta$ and $\mu(v_i), \sigma(v_i)$ are the mean and standard deviation of the information gain for vertex v_i. The parameter ζ is the trade-off between the exploration and exploitation during the optimization. With larger ζ, the importance of the estimation from the Bayesian model for the expected improvement gets smaller or the uncertainty of the information gain for the vertex v_i will become the dominant component for the expected improvement. Φ and ϕ are the CDF and PDF of the standard normal distribution, respectively. The vertex v_i^* with maximum expected improvement will be selected repeatedly until enough number of sensor locations for next iteration are identified.

By replacing Lines 4–5 in Fig. 4.3 by the Bayesian optimization shown in Fig. 5.2, a set of "informative" regions can be identified without computing the information gain for each potential sensor location which increases the computational efficiency substantially. A comparison of the computational burden between the proposed technique with and without Bayesian optimization is provided in Chapter 6.

Chapter 6

Numerical Simulations and Results

6.1 Route-planning In Known Environments: Results and Discussion

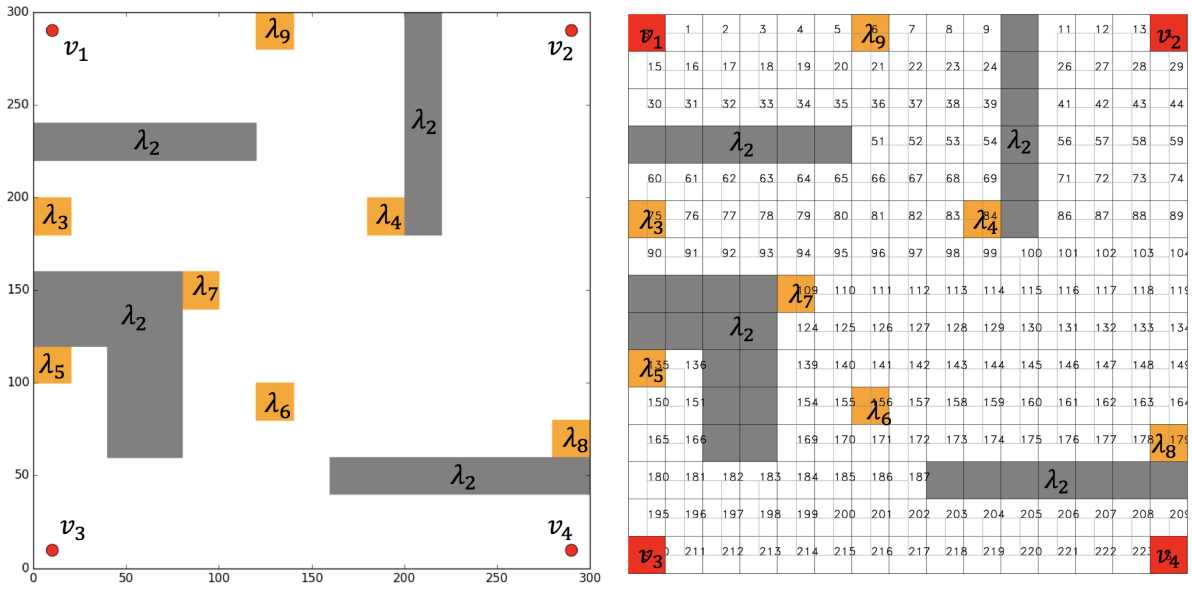
In this section, we present results of numerical simulations to illustrate the proposed decentralized route-planning algorithm discussed in Chapter 2 and Chapter 3 in known environments. Without loss of generality, both independent and collaborative tasks are included in the given global LTL specification. An analysis of the algorithm's performance characteristics, and a comparison to a consensus-based grouping approach is provided.

6.1.1 Illustrative Example

Example 1. Consider an instance of Problem 1 as shown in Fig. 6.1 with global specification $\phi := \phi^{\mathrm{S}} \wedge \phi^{\mathrm{L}}$, $\phi^{\mathrm{S}} := \Box(\lambda_1 \wedge \neg\lambda_2)$, and

$$\phi^{\mathrm{L}} := \Diamond\lambda_3 \wedge \Diamond\lambda_4 \wedge \Diamond\lambda_5 \wedge \Diamond\lambda_6 \wedge \Diamond(\lambda_7 \wedge \Diamond\lambda_8) \wedge \Diamond\lambda_9. \tag{6.1}$$

Here $N_{\mathrm{A}} = 4$, and the global specification (6.1) refers to a global task involving coverage, sequencing, and obstacle avoidance. ROIs $\lambda_3, \ldots, \lambda_9$ are indicated by yellow cells in Fig. 6.1(b), and the vehicles' initial locations are indicated by red cells. ROIs λ_2 (obstacles) are indicated in gray. Finally, λ_1 is associated with the entire workspace, i.e., all cells. Informally, the vehicles are to visit all of the yellow-colored ROIs are to be visited at least once while avoiding obstacles. The ROIs λ_4 and λ_6 are to be collaboratively (simultaneously) visited by 2 and 3 vehicles, respectively. The ROI λ_8 is to be visited after visiting ROI λ_7. The liveness specification is in a conjuctive



(a) Workspace and ROIs.

(b) Uniform cell decomposition.

Figure 6.1: An instance of Problem 1 (details in Example 1).

form $\phi^{\mathrm{L}} = \bigwedge_{j=1}^{N_{\mathrm{T}}}$ with $N_{\mathrm{T}} = 6$, $N_{\mathrm{TC}} = 2$, $\phi_1 := \Diamond\lambda_4$, $\phi_2 := \Diamond\lambda_6$, $\phi_3 := \Diamond\lambda_3$, $\phi_4 := \Diamond\lambda_5$, $\phi_5 := \Diamond(\lambda_7 \wedge \Diamond\lambda_8)$, and $\phi_6 := \Diamond\lambda_9$. The workspace is decomposed into uniform square cells in a 15×15 grid as shown in Fig. 6.1(b). First, we ignore vehicles' kinematic constraints, i.e., set $H = 0$. The positive constants for computing the reward of independent and dependent tasks were defined as $\bar{r}_j = 100$ for each $j \in [N_{\mathrm{T}}]$. The agent communication topology is shown in Table 6.1, and the UCPM is considered.

	Agent 1	Agent 2	Agent 3	Agent 4
Neighboring agents	$\{1\}$	$\{1, 3\}$	$\{2, 4\}$	$\{3\}$

Table 6.1: Agent communication topology in Example 1.

First, the CBBA is executed, resulting in the following task paths,

$$\mathbf{p}_1 = (6, 3), \qquad \mathbf{p}_2 = (5), \qquad \mathbf{p}_3 = (4), \qquad \mathbf{p}_4 = \text{null},$$

which is equivalent to the following independent local specifications, per (2.8):

$$\psi_1 := \Diamond(\lambda_9 \wedge \Diamond\lambda_3), \qquad \psi_2 := \Diamond(\lambda_7 \wedge \Diamond\lambda_8), \qquad \psi_3 := \Diamond\lambda_5, \qquad \psi_4 := \text{null}.$$

This assignment is achieved over three bundle construction and consensus iterations. The rewards maintained by each agent for each task, the assignment of tasks, and the task paths are indicated in Tables 6.2–6.4.

In the first bundle construction phase at $t = 1$, each agent i places bids for all independent tasks and calculates an optimal task path (i.e., sequence of visits to ROIs). As shown in Table 6.2, agent 2 is outbid by agent 1 for tasks $3, 5$, and 6 and outbid by agent 3 for task 4. Agent 4 is outbid by agent 3 for all tasks. In the consensus phase at $t = 1$, the reward information is exchanged among neighboring agents, and each agent accordingly updates its knowledge of task assignments. After two further iterations the CBBA converges to the aforesaid local specifications $\psi_1, \ldots, \psi_4$.

Table 6.2: CBBA phases in Example 1: iteration $t = 1$

		Agent 1				Agent 2				Agent 3				Agent 4			
		ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_3	ϕ_4	ϕ_5	ϕ_6
Bundle const. phase	Reward	82	64	44	93	68	50	30	79	76	94	42	65	62	80	28	51
	Assignment	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
	Path $\mathbf{p}_i$	$(6, 3, 5, 4)$				$(6, 3, 5, 4)$				$(4, 3, 6, 5)$				$(4, 3, 6, 5)$			
Consensus phase	Reward a	82	64	44	93	82	94	44	93	76	94	42	79	76	94	42	65
	Assignment	1	1	1	1	1	3	1	1	3	3	3	2	3	3	3	3
	Path $\mathbf{p}_i$	$(6, 3, 5, 4)$				null				$(4, 3)$				null			

Table 6.3: CBBA phases in Example 1: iteration $t = 2$

		Agent 1				Agent 2				Agent 3				Agent 4			
		ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_3	ϕ_4	ϕ_5	ϕ_6
Bundle const. phase	Reward	82	64	44	93	82	94	68	93	76	94	56	79	76	94	54	77
	Assignment	1	1	1	1	1	3	2	1	3	3	3	2	3	3	4	4
	Path $\mathbf{p}_i$	$(6, 3, 5, 4)$				(5)				$(4, 3, 5)$				$(6, 5)$			
Consensus phase	Reward	82	94	68	93	82	94	68	93	82	94	68	93	76	94	56	79
	Assignment	1	3	2	1	1	3	2	1	1	3	2	1	3	3	3	2
	Path $\mathbf{p}_i$	$(6, 3)$				(5)				(4)				null			

Table 6.4: CBBA phases in Example 1: iteration $t = 3$

		Agent 1				Agent 2				Agent 3				Agent 4			
		ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_3	ϕ_4	ϕ_5	ϕ_6
Bundle const. phase	Reward	82	94	68	93	82	94	68	93	82	94	68	93	76	94	56	79
	Assignment	1	3	2	1	1	3	2	1	1	3	2	1	3	3	3	2
	Path $\mathbf{p}_i$	$(6, 3)$				(5)				(4)				null			
Consensus phase	Reward	82	94	68	93	82	94	68	93	82	94	68	93	82	94	68	93
	Assignment	1	3	2	1	1	3	2	1	1	3	2	1	1	3	2	1
	Path $\mathbf{p}_i$	$(6, 3)$				(5)				(4)				null			

Table 6.5: Synchronization phases in Example 1: iteration $t = 1$

		Agent 1	Agent 2	Agent 3	Agent 4
Bundle const. phase	$\mathcal{F}_i$	$\begin{bmatrix}75 & 0 & 0 & 0\\ 83 & 0 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 87 & 0 & 0\\ 0 & 79 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 75 & 0\\ 0 & 0 & 83 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 0 & 79\\ 0 & 0 & 0 & 87\end{bmatrix}$
	$\mathcal{G}_i$	$\begin{bmatrix}1 & 0 & 0 & 0\\ 1 & 0 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 0 & 1\\ 0 & 0 & 0 & 1\end{bmatrix}$
	Path $\mathbf{p}_i$	$(6, 2, 1, 3)$	$(1, 2, 5)$	$(4, 2, 1)$	$(2, 1)$
Consensus phase	$\mathcal{F}_i$	$\begin{bmatrix}75 & 87 & 0 & 0\\ 83 & 79 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}75 & 87 & 75 & 0\\ 83 & 79 & 83 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 87 & 75 & 79\\ 0 & 79 & 83 & 87\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 75 & 79\\ 0 & 0 & 83 & 87\end{bmatrix}$
	$\mathcal{G}_i$	$\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & 1 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 1 & 1\\ 0 & 1 & 1 & 1\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 1 & 1\\ 0 & 0 & 1 & 1\end{bmatrix}$
	Path $\mathbf{p}_i$	$(6, 2, 1, 3)$	$(2, 5)$	$(4, 2, 1)$	$(2, 1)$

Next, the proposed algorithm for assigning collaborative tasks 1 and 2 (§3.2.2) is executed. Tasks 1 and 2 require $M_1 = 2$ and $M_2 = 3$ vehicles, respectively. The iterative updates of the reward and assignment matrices $\mathcal{F}_i$ and $\mathcal{G}_i$ maintained by each agent are indicated in Tables 6.5–6.7.

In the first bundle construction iteration (Table 6.5), each agent i assigns both collaborative tasks to itself. Note that the paths $\mathbf{p}_i$ maintained by each agent include the previously assigned independent tasks. Each agent selects for each of the collaborative tasks M_j agents to minimize the waiting time (Line 4 of Figure 3.2) based on its current reward matrix. For example, agent 2 removes itself from task 1 because agents 1 and 3 can complete this task without any waiting. After two further iterations of bundle construction and consensus, the proposed algorithm converges and results in the following task paths

$$\mathbf{p}_1 = (6, 2, 1, 3), \qquad \mathbf{p}_2 = (2, 5), \qquad \mathbf{p}_3 = (4, 2, 1), \qquad \mathbf{p}_4 = \text{null},$$

which is equivalent to the following local specifications per (2.8)

$$\psi_1 := \Diamond(\lambda_9 \wedge \Diamond(\lambda_6 \wedge \Diamond(\lambda_4 \wedge \Diamond\lambda_3))), \qquad \psi_2 := \Diamond(\lambda_6 \wedge \Diamond(\lambda_7 \wedge \Diamond\lambda_8)),$$
$$\psi_3 := \Diamond(\lambda_5 \wedge \Diamond(\lambda_6 \wedge \Diamond\lambda_4)), \qquad \psi_4 := \text{null}.$$

Table 6.6: Synchronization phases in Example 1: iteration $t = 2$

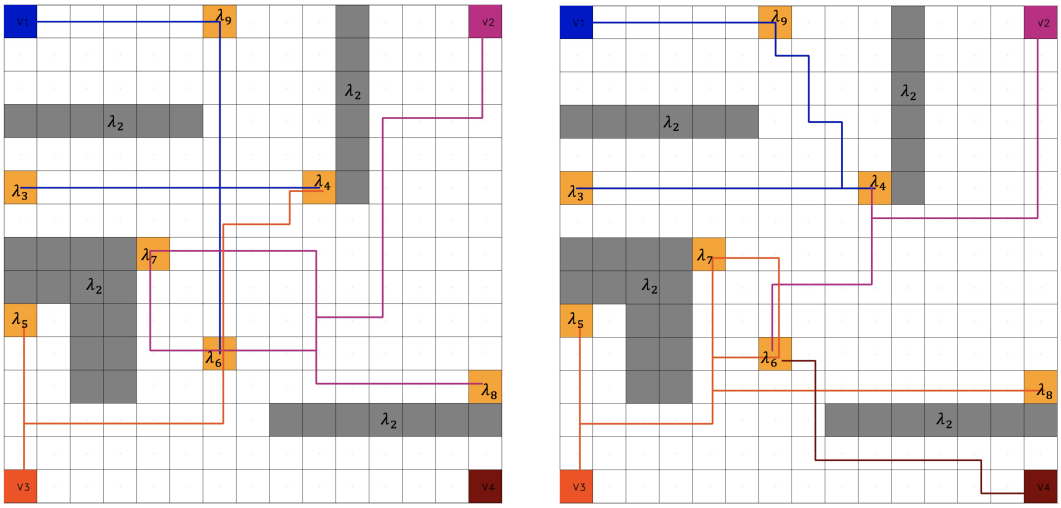
		Agent 1	Agent 2	Agent 3	Agent 4
Bundle const. phase	$\mathcal{F}_i$	$\begin{bmatrix}75 & 87 & 0 & 0\\ 83 & 79 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}75 & 87 & 75 & 0\\ 83 & 81 & 83 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 87 & 75 & 79\\ 0 & 79 & 83 & 87\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 75 & 79\\ 0 & 0 & 83 & 87\end{bmatrix}$
	$\mathcal{G}_i$	$\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & 1 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 1 & 1\\ 0 & 1 & 1 & 1\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 1 & 1\\ 0 & 0 & 1 & 1\end{bmatrix}$
	Path $\mathbf{p}_i$	$(6, 2, 1, 3)$	$(1, 2, 5)$	$(4, 2, 1)$	$(2, 1)$
Consensus phase	$\mathcal{F}_i$	$\begin{bmatrix}75 & 87 & 75 & 0\\ 83 & 81 & 83 & 0\end{bmatrix}$	$\begin{bmatrix}75 & 87 & 75 & 79\\ 83 & 81 & 83 & 87\end{bmatrix}$	$\begin{bmatrix}75 & 87 & 75 & 79\\ 83 & 81 & 83 & 87\end{bmatrix}$	$\begin{bmatrix}0 & 87 & 75 & 79\\ 0 & 79 & 83 & 87\end{bmatrix}$
	$\mathcal{G}_i$	$\begin{bmatrix}1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 1 & 1\\ 0 & 1 & 1 & 1\end{bmatrix}$
	Path $\mathbf{p}_i$	$(6, 2, 1, 3)$	$(2, 5)$	$(4, 2, 1)$	$(2, 1)$

Table 6.7: Synchronization phases in Example 1: iteration $t = 3$

		Agent 1	Agent 2	Agent 3	Agent 4
Bundle const. phase	$\mathcal{F}_i$	$\begin{bmatrix}75 & 87 & 75 & 0\\ 83 & 81 & 83 & 0\end{bmatrix}$	$\begin{bmatrix}75 & 87 & 75 & 79\\ 83 & 81 & 83 & 87\end{bmatrix}$	$\begin{bmatrix}75 & 87 & 75 & 79\\ 83 & 81 & 83 & 87\end{bmatrix}$	$\begin{bmatrix}0 & 87 & 75 & 79\\ 0 & 79 & 83 & 87\end{bmatrix}$
	$\mathcal{G}_i$	$\begin{bmatrix}1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 1 & 1\\ 0 & 1 & 1 & 1\end{bmatrix}$
	Path $\mathbf{p}_i$	$(6, 2, 1, 3)$	$(1, 2, 5)$	$(4, 2, 1)$	$(2, 1)$
Consensus phase	$\mathcal{F}_i$	$\begin{bmatrix}75 & 87 & 75 & 79\\ 83 & 81 & 83 & 87\end{bmatrix}$	$\begin{bmatrix}75 & 87 & 75 & 79\\ 83 & 81 & 83 & 87\end{bmatrix}$	$\begin{bmatrix}75 & 87 & 75 & 79\\ 83 & 81 & 83 & 87\end{bmatrix}$	$\begin{bmatrix}75 & 87 & 75 & 79\\ 83 & 81 & 83 & 87\end{bmatrix}$
	$\mathcal{G}_i$	$\begin{bmatrix}1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$
	Path $\mathbf{p}_i$	$(6, 2, 1, 3)$	$(2, 5)$	$(4, 2, 1)$	null

Fig. 6.2(a) illustrates the task assignment and the corresponding route for each vehicle. Vehicle 1 starts at the blue-colored ROI and then moves to ROIs λ_9, λ_6, λ_4 and λ_3 sequentially. Vehicle 2 starts at the magenta-colored ROI and then moves to ROIs λ_6, λ_7 and λ_8 sequentially. Vehicle 3 starts at the red-colored ROI and then moves to ROIs λ_5, λ_6 and λ_4 sequentially. Finally, vehicle 4 remains at the maroon-colored ROI because it has no task assigned. The sum of the lengths of the routes taken by the vehicles is 97 units, whereas the total waiting time is 4 units. To complete task 1 (with $M_1 = 2$) vehicles 1 and 3 arrive at ROI λ_4 simultaneously and for task 2 (with $M_2 = 3$) vehicles 1 and 3 arrive at ROI λ_6 and wait for 2 time units until vehicle 2 arrives.

By way of comparison, task assignment and route-planning by implementing the E-CBGA



(a) Task assignment by proposed technique.

(b) Task assignment by E-CBGA.

Figure 6.2: Routes of all vehicles in Example 1: routes of vehicles 1–4 indicated in blue, magenta, red, and maroon, respectively.

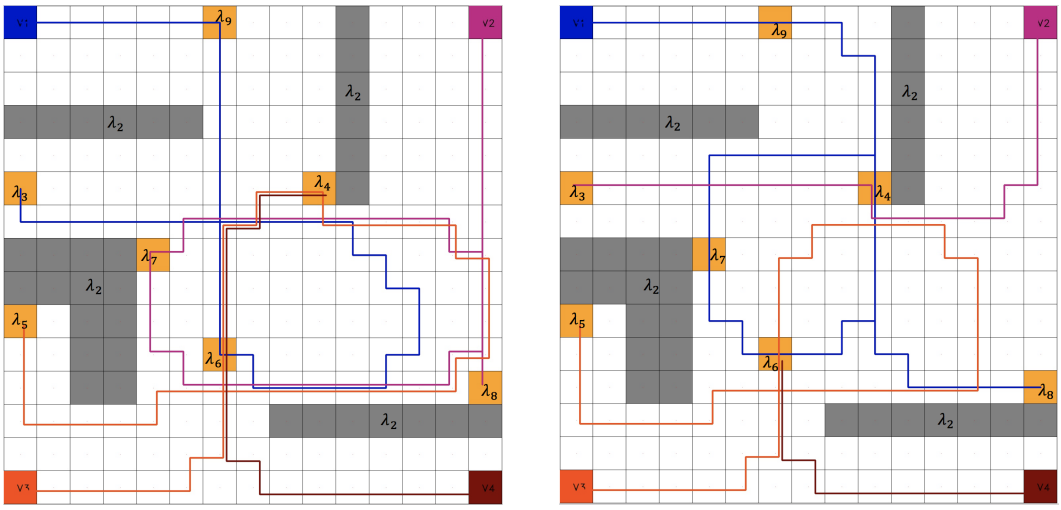
approach (§3.2.1) results in the following local specifications

$$\psi_1 = \Diamond(\lambda_9 \wedge \Diamond(\lambda_4 \wedge \Diamond\lambda_3)) \qquad \psi_2 = \Diamond(\lambda_4 \wedge \Diamond\lambda_6),$$
$$\psi_3 = \Diamond(\lambda_5 \wedge \Diamond(\lambda_6 \wedge \Diamond(\lambda_7 \wedge \Diamond\lambda_8)) \qquad \psi_4 = \Diamond\lambda_6$$

The route traveled by each vehicle is shown in Fig. 6.2(b). In this case, the sum of lengths of all the vehicles' routes is 94 units. However, the waiting time for vehicles 2, 3, and 4 is 2, 6, and 10 time units respectively, i.e. a total of 18 time units compared to 4 time units due to the proposed algorithm. □

Example 2. We consider Example 1 again, but each vehicle is now modeled by the KCPM. The minimum radius of turn is set to 3 units, where one unit is the side of a square cell. The lifted graph algorithm is applied with $H = 4$ to calculate the rewards for each agent, and to compute routes satisfying each agents' local specifications. The resultant local specifications for each agent are:

$$\psi_1 := \Diamond(\lambda_9 \wedge \Diamond(\lambda_6 \wedge \Diamond\lambda_3)) \qquad \psi_2 := \Diamond(\lambda_7 \wedge \Diamond\lambda_8),$$
$$\psi_3 := \Diamond(\lambda_6 \wedge \Diamond(\lambda_4 \wedge \Diamond\lambda_5)) \qquad \psi_4 := \Diamond(\lambda_6 \wedge \Diamond\lambda_4).$$



(a) Task assignment by proposed technique.

(b) Task assignment by E-CBGA.

Figure 6.3: Routes of all vehicles in Example 2: routes of vehicles 1–4 indicated in blue, magenta, red, and maroon, respectively.

The routes of the vehicles to satisfy these local specifications are shown in Fig. 6.3(a). First, in comparison with Example 1, notice in Fig. 6.2(a) that the routes of all vehicles involve several sharp turns, which cannot be executed by a vehicle with minimum turn radius of 3 units. By contrast, in Fig. 6.3 such sharp turns are absent. Each route indicated in Fig. 6.3 is guaranteed to contain a continuously differentiable curve with minimum turn radius of 3 units.

In comparison, the vehicles routes associated with the task assignments by the E-CBGA approach are shown in Fig. 6.3(b). In this example, the sum of lengths of vehicle routes resulting from the proposed algorithm is 147 units, whereas that resulting from E-CBGA is 125 units. However, the total waiting duration resulting from the proposed algorithm is 10 time units, whereas that resulting from E-CBGA is 24 time units. As in Example 1, whereas the proposed algorithm results in longer paths, the total waiting duration is significantly lower. □

Example 3. Consider Example 1 yet again for the KCPM, but with different minimum turn radii for each vehicle. Namely, these turn radii are 4, 3, 3, and 2 units, respectively. By the proposed method, the local specifications for each agent are:

$$\psi_1 := \Diamond(\lambda_9 \wedge \Diamond(\lambda_4 \wedge \Diamond\lambda_6)) \qquad \psi_2 := \Diamond(\lambda_4 \wedge \Diamond(\lambda_6 \wedge \Diamond\lambda_3)),$$

$$\psi_3 := \Diamond\lambda_5 \qquad \psi_4 := \Diamond(\lambda_6 \wedge \Diamond(\lambda_7 \wedge \Diamond\lambda_8)).$$

The routes of the vehicles to satisfy these local specifications are shown in Fig. 6.4(a). First, note that these local specifications and routes are different compared to those in Example 2, although the global specification is the same. For example, the route planned for vehicle 1 (blue) is significantly different because of the larger minimum turn radius in this example. In comparison, the vehicles routes associated with the task assignments by the E-CBGA approach are shown in Fig. 6.4(b). The sum of lengths of vehicle routes resulting from the proposed algorithm is 107 units, whereas that resulting from E-CBGA is 117 units. The total waiting duration resulting from the proposed algorithm is 12 time units, whereas that resulting from E-CBGA is 22 time units.

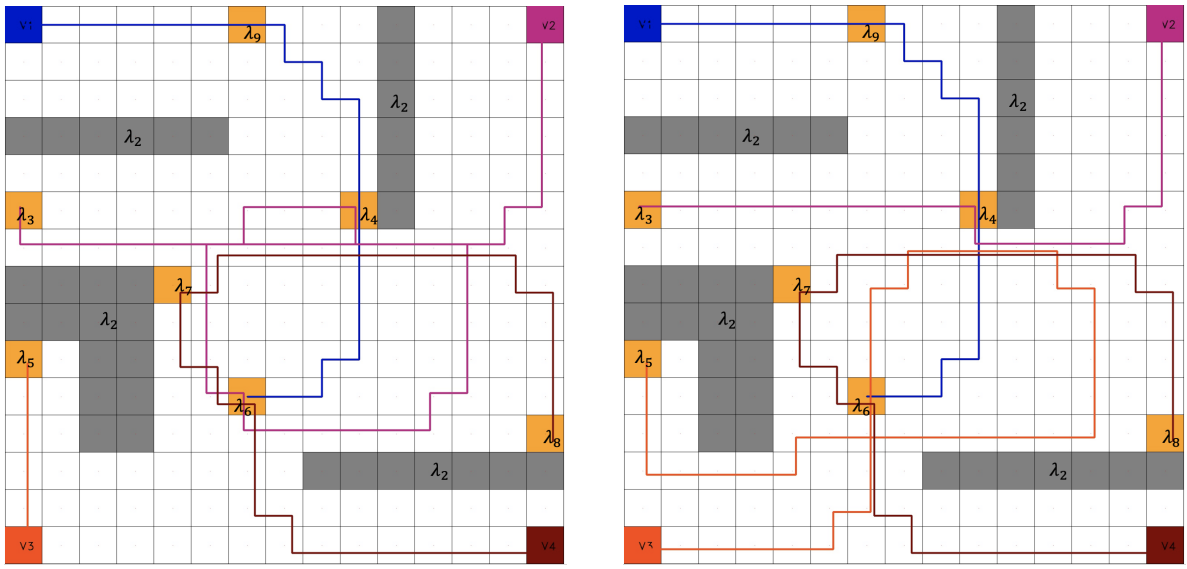
Example 4. The specification $\phi = \phi^{\mathrm{S}} \wedge \phi^{\mathrm{L}}$ in this example involves the operator `implies` denoted $\rightarrow$, which enables vehicular motion conditioned on the environment. The safety and liveness specifications are $\phi^{\mathrm{S}} = \Box(\lambda_1 \wedge \neg\lambda_2)$, and $\phi^{\mathrm{L}} = (\wedge_{m=3}^{9} \Diamond\lambda_m \wedge (\texttt{(Found\_T)}_m \rightarrow \Diamond\lambda_{13})) \wedge (\wedge_{n=10}^{12}\lambda_n \wedge (\texttt{(Found\_T)}_m \rightarrow \Diamond\lambda_{13}))$, where the proposition $\texttt{(Found\_T)}_m$ is `true` whenever a target of interest is present in the ROI λ_m. These ROIs are indicated in Fig. 6.5. Informally, this specification requires the vehicles to visit ROIs $\lambda_3, \ldots, \lambda_{12}$ to search for a target. If this target is found, then the vehicles are to report the finding at ROI λ_{13}. The ROIs $\lambda_{10}, \lambda_{11}$, and λ_{12} are to be searched collaboratively by two vehicles simultaneously, i.e., $M_{10} = M_{11} = M_{12} = 2$. Here $N_{\mathrm{A}} = 6$ for the UCPM, and the agent communication network is shown below. This example mirrors various real-world applications such as search-and-rescue, target detection, and warehouse inventory management.

	$i=1$	2	3	4	5	6
$\mathcal{N}_i$	$\{3,4\}$	$\{3,6\}$	$\{1,2,6\}$	$\{1,5\}$	$\{4,6\}$	$\{2,3,5\}$

The sensory mechanism of detecting the target is not of interest in this work. We assume that this target is present in ROIs $\lambda_6, \ldots, \lambda_9, \lambda_{11}$, and λ_{12}. Vehicles detect the target whenever they visit these ROIs.

The proposed algorithm results in the local specifications

$$\psi_1 := \Diamond(\lambda_5 \wedge \mu_5 \wedge \Diamond(\lambda_8 \wedge \mu_8 \wedge \Diamond(\lambda_{11} \wedge \mu_{11}))),$$
$$\psi_2 := \Diamond(\lambda_{10} \wedge \mu_{10} \wedge \Diamond(\lambda_3 \wedge \mu_3)), \qquad \psi_4 := \Diamond(\lambda_{12} \wedge \mu_{12}),$$



(a) Proposed method. (b) E-CBGA.

Figure 6.4: Routes of all vehicles in Example 3: routes of vehicles 1–4 indicated in blue, magenta, red, and maroon, respectively.

$$\psi_3 := \Diamond(\lambda_4 \wedge \mu_4 \wedge \Diamond(\lambda_6 \wedge \mu_6 \wedge \Diamond(\lambda_7 \wedge \mu_7 \wedge \Diamond(\lambda_9 \wedge \mu_9)))),$$
$$\psi_5 := \Diamond(\lambda_{12} \wedge \mu_{12} \wedge \Diamond(\lambda_{11} \wedge \mu_{11})), \qquad \psi_6 := \Diamond(\lambda_{10} \wedge \mu_{10}),$$

where $\mu_m := (\texttt{Found\_T})_m \rightarrow \Diamond\lambda_{13}$. The vehicle routes resulting from the proposed algorithm are indicated in Fig. 6.5.

In this example, the sum of lengths of vehicle routes resulting from the proposed algorithm is 241 units, whereas that resulting from the E-CBGA approach is 181 units. However, the total waiting duration resulting from the proposed algorithm is 14 time units, whereas that resulting from the E-CBGA approach is 42 time units.

Example 5. In this example we consider $N_{\mathrm{A}} = 8$ and $N_{\mathrm{R}} = 15$ for the UCPM. The fixed communication topology among vehicles is described in Table 6.8. The global specification is $\phi = \phi^{\mathrm{S}} \wedge \phi^{\mathrm{L}}$ with ϕ^{S} as in Example 4 and $\phi^{\mathrm{L}} := (\wedge_{m=3}^{7}\lambda_m) \wedge (\Diamond\lambda_8 \wedge ((\texttt{Found\_T})_8 \rightarrow \Diamond\lambda_{15})) \wedge (\wedge_{n=9}^{11}\lambda_n) \wedge (\wedge_{k=12}^{14}(\Diamond\lambda_k \wedge ((\texttt{Found\_T})_k \rightarrow \Diamond\lambda_{15})))$ with $M_9 = 2$ and $M_i = 3$ for $i = 10, \ldots, 14$.

Informally, ROIs $\lambda_3, \ldots, \lambda_7$ are to be visited by one vehicle, λ_9 by two vehicles simultaneously, and ROIs $\lambda_{10}, \lambda_{11}$ by three vehicles simultaneously. ROI λ_8 (resp. ROIs $\lambda_{12}, \lambda_{13}, \lambda_{14}$) are

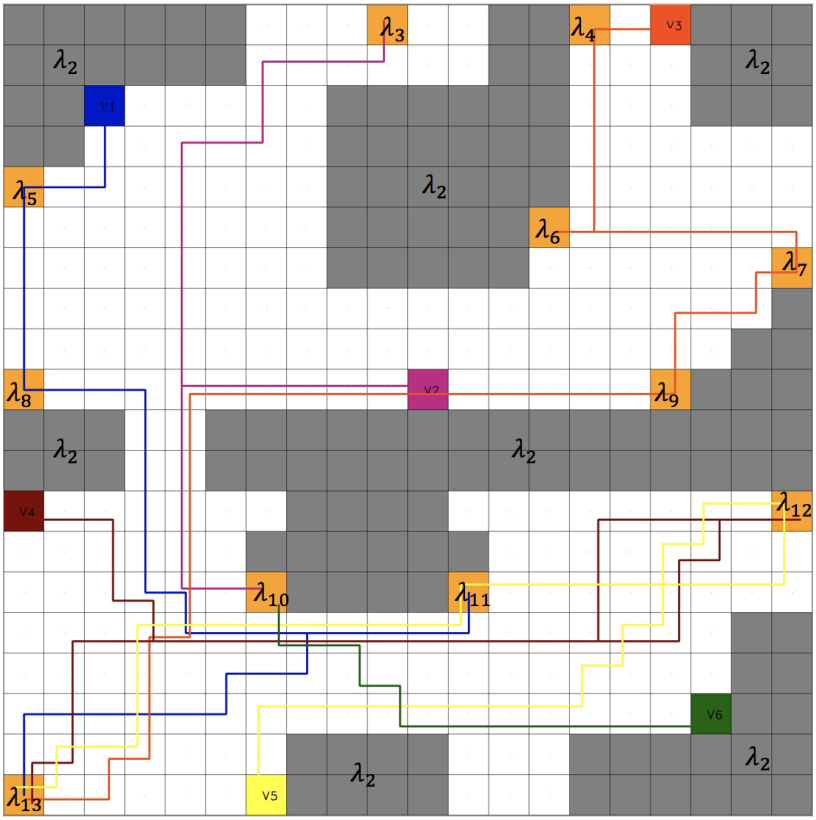


Figure 6.5: Routes of all vehicles in Example 4: routes of vehicles 1–6 indicated in blue, magenta, red, maroon, yellow, and green respectively.

	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8
Neighboring agents	$\{3,4,7\}$	$\{4,5,8\}$	$\{1,5,6\}$	$\{1,2,8\}$	$\{2,3,7\}$	$\{3,7,8\}$	$\{1,5,6\}$	$\{2,4,6\}$

Table 6.8: Agent communication topology in Example 5.

to be visited by one vehicle (resp. three vehicles simultaneously), and if an object of interest is present in any of these ROIs, then λ_{15} is to be visited for reporting its detection. As before, the sensory mechanism of detecting the object is not of interest in this work. We assume that this object is present in ROIs λ_8 and λ_{12}.

The local liveness specifications resulting from the proposed algorithm are as follows:

$$\psi_1 := \Diamond(\mu_{14} \wedge \Diamond\mu_{12}),$$

$$\psi_2 := \Diamond(\lambda_7 \wedge \Diamond(\lambda_{10} \wedge \Diamond\mu_{13})),$$

$$\psi_3 := \Diamond(\lambda_4 \wedge \Diamond(\lambda_{11} \wedge \Diamond\lambda_9))$$

$$\psi_4 := \diamond(\lambda_{11} \wedge \diamond\lambda_9),$$
$$\psi_5 := \diamond(\lambda_3 \wedge \diamond(\mu_{14} \wedge \diamond(\mu_{12} \wedge \diamond\mu_{13}))),$$
$$\psi_6 := \diamond(\lambda_6 \wedge \diamond\lambda_{10}),$$
$$\psi_7 := \diamond(\mu_8 \wedge \diamond(\mu_{14} \wedge \diamond\mu_{13})),$$
$$\psi_8 := \diamond(\lambda_5 \wedge \diamond(\lambda_{10} \wedge \diamond(\lambda_{11} \wedge \diamond\mu_{12}))).$$

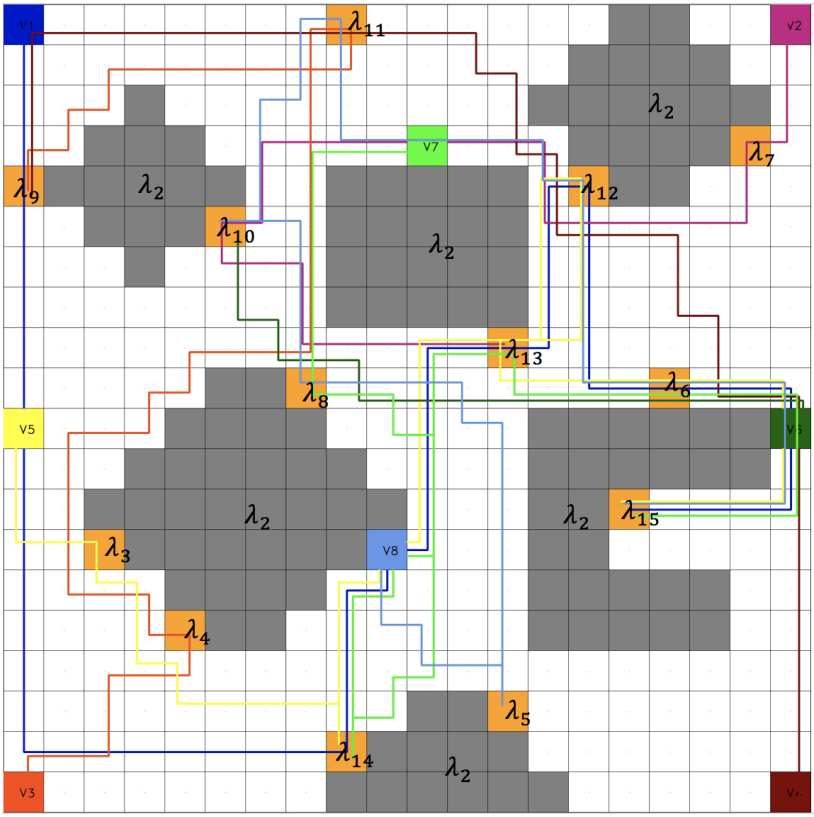
where $\mu_m := \texttt{(Found\_T)}_m \rightarrow \diamond\lambda_{15}$. The routes followed by each of the vehicles are shown in Fig. 6.6. The initial base locations for vehicles 1–8 are indicated by cells colored dark-blue, magenta, red, maroon, yellow, dark-green, light-green, and light-blue, respectively.

In this example, the sum of lengths of vehicle routes resulting from the proposed algorithm is 384 units and the total waiting time is 58 time units. However, the E-CBGA approach failed to reach a feasible solution. Specifically, by the E-CBGA result, ROI λ_{10} and λ_{11} are assigned to vehicle 1, 6 and 7 collaboratively. Vehicle 1 and 6 are assigned to ROI λ_{10} first, then λ_{11}. However, vehicle 7 is assigned to visit these two ROIs in the opposite order (i.e., λ_{11} first, then λ_{10}) which leads to infinite waiting time for these three vehicles. This situation occurs because λ_{10} and λ_{11} are to be visited by three vehicles simultaneously, but the E-CBGA approach has no temporal considerations in its task assignment.

Example 6. The specification $\phi = \phi^{\mathrm{S}} \wedge \phi^{\mathrm{L}}$ in this example involves visits to several ROIs in specified ordered sequences. Here ϕ^{S} is same as in Example 4 and $\phi^{\mathrm{L}} := \wedge_{j=1}^{13}\phi_j$, where ϕ_j are detailed in Table 6.10. Informally, the specification ϕ_2 for example requires two vehicles to simultaneously visit ROI λ_4 first and ROI λ_{21} next, in that order. This example mirrors a mobility-on-demand application, where multiple transport vehicles visit pickup and dropoff locations in sequence (i.e., pickup first, dropoff next). Such applications are widely important for, say, school buses or autonomous taxi/ride-share vehicles. We consider $N_{\mathrm{A}} = 10$ with a fixed communication topology shown in Table 6.9.

The local liveness specifications resulting from the proposed algorithm are as follows:

$$\psi_1 := \diamond(\lambda_{10} \wedge (\lambda_{10} \rightarrow \diamond\lambda_{16})),$$
$$\psi_2 := \diamond(\lambda_6 \wedge (\lambda_6 \rightarrow \diamond\lambda_{19}) \wedge \diamond(\lambda_4 \wedge (\lambda_4 \rightarrow \diamond\lambda_{21}))),$$



(a) Task assignment by proposed technique.

Figure 6.6: Routes of all vehicles in Example 5: routes of vehicles 1–8 indicated in dark-blue, magenta, red, maroon, yellow, dark-green, light-green, and light-blue respectively.

$$\psi_3 := \Diamond(\lambda_3 \wedge (\lambda_3 \rightarrow \Diamond\lambda_{20})),$$

$$\psi_4 := \Diamond(\lambda_8 \wedge (\lambda_8 \rightarrow \Diamond\lambda_{20}) \wedge \Diamond(\lambda_6 \wedge (\lambda_6 \rightarrow \Diamond\lambda_{19}) \wedge \Diamond(\lambda_{12} \wedge (\lambda_{12} \rightarrow \Diamond\lambda_{18})))),$$

$$\psi_5 := \Diamond(\lambda_9 \wedge (\lambda_9 \rightarrow \Diamond\lambda_{18}) \wedge \Diamond(\lambda_{13} \wedge (\lambda_{13} \rightarrow \Diamond\lambda_{16}))),$$

$$\psi_6 := \Diamond(\lambda_{14} \wedge (\lambda_{14} \rightarrow \Diamond\lambda_{17}) \wedge \Diamond(\lambda_{10} \wedge (\lambda_{10} \rightarrow \Diamond\lambda_{16}) \wedge \Diamond(\lambda_{12} \wedge (\lambda_{12} \rightarrow \Diamond\lambda_{18})))),$$

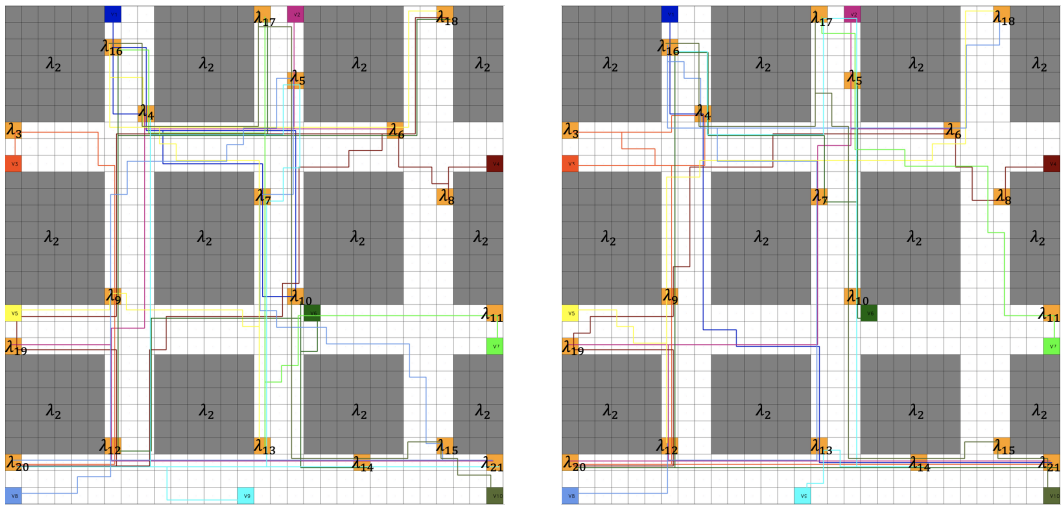
$$\psi_7 := \Diamond(\lambda_{11} \wedge (\lambda_{11} \rightarrow \Diamond\lambda_{17}) \wedge \Diamond(\lambda_{13} \wedge (\lambda_{13} \rightarrow \Diamond\lambda_{16}))),$$

$$\psi_8 := \Diamond(\lambda_5 \wedge (\lambda_5 \rightarrow \Diamond\lambda_{20}) \wedge \Diamond(\lambda_7 \wedge (\lambda_7 \rightarrow \Diamond\lambda_{21}))),$$

$$\psi_9 := \Diamond(\lambda_4 \wedge (\lambda_4 \rightarrow \Diamond\lambda_{21}) \wedge \Diamond(\lambda_5 \wedge (\lambda_5 \rightarrow \Diamond\lambda_{20}) \wedge \Diamond(\lambda_7 \wedge (\lambda_7 \rightarrow \Diamond\lambda_{21})))),$$

$$\psi_{10} := \Diamond(\lambda_{15} \wedge (\lambda_{15} \rightarrow \Diamond\lambda_{16}) \wedge \Diamond(\lambda_{14} \wedge (\lambda_{14} \rightarrow \Diamond\lambda_{17}))).$$

The vehicle routes resulting from the proposed algorithm are indicated in Fig. 6.7(a), whereas those resulting from the state-of-the-art approach are indicated in Fig. 6.7(b). In this example, the



(a) Task assignment by proposed technique.

(b) Task assignment by E-CBGA.

Figure 6.7: Routes of all vehicles in Example 3: routes of vehicles 1–10 indicated in dark-blue, magenta, red, maroon, yellow, dark-green, light-green, light-blue, cyan, and olive respectively.

Table 6.9: Agent communication topology in Example 6.

	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5
Neighboring agents	$\{2,3,5,6,10\}$	$\{1,4,6,7,8\}$	$\{1,5,7,9,10\}$	$\{2,5,7,8\}$	$\{1,3,4,8,9\}$
	Agent 6	Agent 7	Agent 8	Agent 9	Agent 10
Neighboring agents	$\{1,2,8,9,10\}$	$\{2,3,4,9,10\}$	$\{2,4,5,6,10\}$	$\{3,5,6,7,10\}$	$\{1,3,6,7,8,9\}$

sum of lengths of vehicle routes resulting from the proposed algorithm is 870 units, whereas that resulting from the E-CBGA approach is 773 units. However, the total waiting time resulting from the proposed algorithm is 5 time units, whereas that resulting from the E-CBGA approach is 216 time units. As in Examples 1–4, whereas the proposed algorithm results in longer paths, the total waiting time is significantly lower compared to the E-CBGA approach.

6.1.2 Discussion

The primary advantage of the proposed algorithm over the E-CBGA/EH-CBGA approach is that the total waiting duration of the team of agents is significantly reduced. More importantly, in certain cases such as Example 5, the E-CBGA may not even produce a feasible solution as

ϕ_j	Informal description	Number of vehicles M_j
$\phi_1 := \Diamond(\lambda_3 \wedge (\lambda_3 \rightarrow \Diamond\lambda_{20}))$	Pick-up at λ_3 and drop-off at λ_{20}	1
$\phi_2 := \Diamond(\lambda_4 \wedge (\lambda_4 \rightarrow \Diamond\lambda_{21}))$	Pick-up at λ_4 and drop-off at λ_{21}	2
$\phi_3 := \Diamond(\lambda_5 \wedge (\lambda_5 \rightarrow \Diamond\lambda_{20}))$	Pick-up at λ_5 and drop-off at λ_{20}	2
$\phi_4 := \Diamond(\lambda_6 \wedge (\lambda_6 \rightarrow \Diamond\lambda_{19}))$	Pick-up at λ_6 and drop-off at λ_{19}	2
$\phi_5 := \Diamond(\lambda_7 \wedge (\lambda_7 \rightarrow \Diamond\lambda_{21}))$	Pick-up at λ_7 and drop-off at λ_{21}	2
$\phi_6 := \Diamond(\lambda_8 \wedge (\lambda_8 \rightarrow \Diamond\lambda_{20}))$	Pick-up at λ_8 and drop-off at λ_{20}	1
$\phi_7 := \Diamond(\lambda_9 \wedge (\lambda_9 \rightarrow \Diamond\lambda_{18}))$	Pick-up at λ_9 and drop-off at λ_{18}	1
$\phi_8 := \Diamond(\lambda_{10} \wedge (\lambda_{10} \rightarrow \Diamond\lambda_{16}))$	Pick-up at λ_{10} and drop-off at λ_{16}	2
$\phi_9 := \Diamond(\lambda_{11} \wedge (\lambda_{11} \rightarrow \Diamond\lambda_{17}))$	Pick-up at λ_{11} and drop-off at λ_{17}	1
$\phi_{10} := \Diamond(\lambda_{12} \wedge (\lambda_{12} \rightarrow \Diamond\lambda_{18}))$	Pick-up at λ_{12} and drop-off at λ_{18}	2
$\phi_{11} := \Diamond(\lambda_{13} \wedge (\lambda_{13} \rightarrow \Diamond\lambda_{16}))$	Pick-up at λ_{13} and drop-off at λ_{16}	2
$\phi_{12} := \Diamond(\lambda_{14} \wedge (\lambda_{14} \rightarrow \Diamond\lambda_{17}))$	Pick-up at λ_{14} and drop-off at λ_{17}	2
$\phi_{13} := \Diamond(\lambda_{15} \wedge (\lambda_{15} \rightarrow \Diamond\lambda_{16}))$	Pick-up at λ_{15} and drop-off at λ_{16}	1

Table 6.10: Liveness specifications in Example 6.

	Ex. 1	Ex. 2	Ex. 3	Ex. 4	Ex. 5	Ex. 6
Reduction in total waiting duration	77.78%	58.33%	45.45%	66.67%	N/A	97.68%
Increase in total route length	3.191%	17.60%	-5.98%	33.15%	N/A	12.55%
Number of iterations required	6	13	13	11	14	16

Table 6.11: Path lengths and waiting durations due to the proposed algorithm, as compared to E/EH-CBGA.

it does not consider temporal synchronization. Table 6.11 summarizes the reductions in waiting durations for each of the examples. Note that large reductions in waiting durations are achieved at the expense of increases in total route lengths.

As stated in Prop. 2, the proposed algorithm terminates in at most $\frac{1}{2}(N_{\mathrm{TC}}+1)N_{\mathrm{TC}}\delta$ iterations. It is known [85] that the CBBA, which we use to assign independent tasks, terminates in at most $N_{\mathrm{TI}}\delta$ iterations. Therefore, at most $(N_{\mathrm{TI}}+\frac{1}{2}(N_{\mathrm{TC}}+1)N_{\mathrm{TC}})\delta$ iterations are required overall to find local LTL specifications for each vehicle in the team. Table 6.11 shows the numbers of iterations required for termination in each of Examples 1–6.

	Agent 1	Agent 2	Agent 3
Neighboring agents	$\{2\}$	$\{1, 3\}$	$\{2\}$

Table 6.12: Agent communication topology for decentralized implementation on SBCs.

6.1.3 Hardware Implementation

A decentralized implementation of the proposed algorithm was carried out on a network of three Raspberry Pi B+ single-board computers (SBC). Each SBC implemented one agent. Communication among agents was enabled using a local area network (LAN). Each agent implemented the Lightweight and Communications and Marshalling (LCM) library [87] to pass messages using the unidirectional protocol (UDP). Whereas all agents communicated over the same *physical* LAN, inter-agent communications were restricted to emulate the fixed communication topology shown in Table 6.12.

In this implementation, a uniform 12×12 workspace cell decomposition was assumed, as shown in Fig. 6.8. The specified initial locations of vehicles are indicated by red-colored base locations. The specification is $\phi = \phi^{\mathrm{S}} \wedge \phi^{\mathrm{L}}$, where

$$\phi^{\mathrm{S}} := \Box(\lambda_0 \wedge \neg\lambda_1) \wedge (\Diamond\Box\lambda_2), \qquad \phi^{\mathrm{L}} := \Diamond\lambda_3 \wedge \Diamond\lambda_4 \wedge \Diamond\lambda_5 \wedge \Diamond\lambda_6. \tag{6.2}$$

Informally, this specification requires the vehicles to visit ROIs $\lambda_3, \ldots, \lambda_6$ and then return to the base λ_2. ROI λ_6 is required to be visited by two vehicles simultaneously, i.e., $N_{\mathrm{T}} = 4, N_{\mathrm{TI}} = 3$ and $N_{\mathrm{TC}} = 1$.

The proposed algorithm was successfully executed in this setup, and the following local liveness specifications were computed:

$$\psi_1 = \Diamond\lambda_3, \qquad \psi_2 = \Diamond(\lambda_5 \wedge \Diamond\lambda_6), \qquad \psi_3 = \Diamond(\lambda_4 \wedge \Diamond\lambda_6).$$

These specifications were computed after four iterations of bundle construction and consensus of the proposed algorithm. The sum of the lengths of routes to be followed by the vehicles was 71

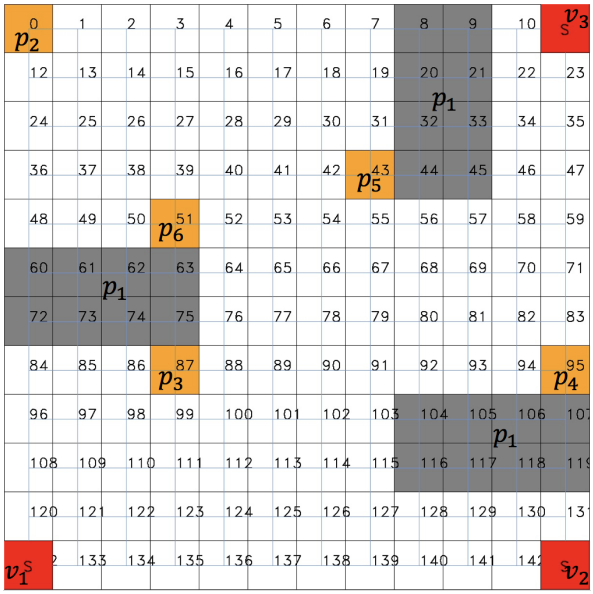


Figure 6.8: Workspace and cell decomposition considered for the implementation in §6.1.3.

units and the total waiting time required was 1 time unit (for agent 3). A video demonstrating the progress of this decentralized implementation is available at https://youtu.be/LKOVzu36PwI.

Whereas mobile robot hardware was not included in this implementation, the Raspberry Pi B+ SBC is readily compatible with several mobile robot platforms [88].

A decentralized route-planning algorithm was proposed to enable a networked team of mobile vehicles satisfy LTL specifications. Whereas the existing literature provides either centralized algorithms or otherwise assumes individual vehicle specifications a priori, the proposed algorithm includes the decomposition of the global LTL specification into local specifications for each vehicle. Minimum turn radius constraints on the vehicles' motion, which are ignored in the existing literature, are addressed in the proposed approach by the lifted graph algorithm. Specifically, H-costs defined on the lifted graphs are used to define rewards for agents during decentralized task assignment. The proposed algorithm achieves task assignments that result in significantly reduced waiting durations for the team, as compared to assignments made by a comparable algorithm (E-CBGA) based on an extension of the existing literature.

6.2 Interactive Planning and Sensing: Results and Discussion

In this section, we present results of numerical simulations to illustrate the interactive planning and sensing algorithm to explore the unknown environment and determine the optimal routes for multi-vehicle system to satisfy the given global LTL specification. A study of convergence rate, required sensory resources and other performance characteristics are provided. Meanwhile, a comparison of the performance against a typical information-driven approach from the literature is also given. A video demonstrating the unknown environment exploration by sensors which is discussed in Example 8 is available at: https://youtu.be/iId4MHs7Qso.

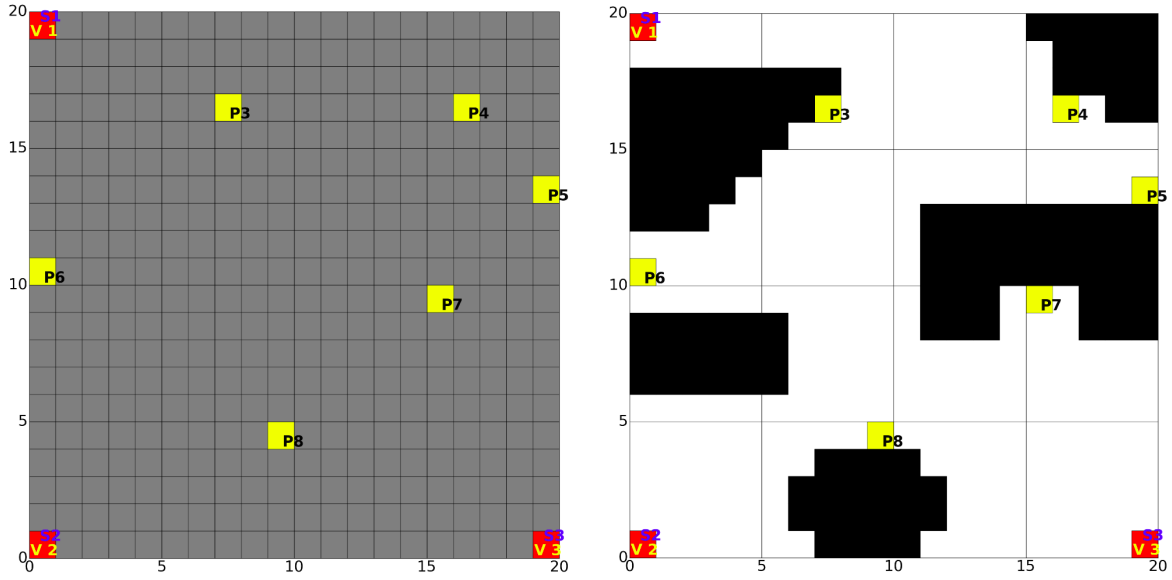
6.2.1 Illustrative Example

Example 7. Consider a team of six robotic vehicles: three terrestrial vehicles as actors (V1$\sim$V3) and three aerial vehicles as sensors (S1$\sim$S3) as illustrated in Fig. 6.9, with their initial locations indicated in red. The LTL specification is $\phi = \phi^{\mathrm{S}} \wedge \phi^{\mathrm{L}}$, where $\phi^{\mathrm{S}} = \Box(P_1 \wedge \neg P_2)$, and $\phi^{\mathrm{L}} = \wedge_{i=3}^{i=8} \Diamond P_i$.

Here P_1 refers to the entire environment, while P_2 refers to obstacles. The regions of interest (ROI) $P_3, \ldots, P_8$ are indicated in yellow. Informally, the specification ϕ requires actor vehicles to visit each of the ROIs $P_3, \ldots, P_8$ at least once each, and to avoid obstacles P_2. The order of visits these ROIs is not specified, and it is not specified which actor should visit which ROI. The location of obstacles is not known a priori to the actors. Fig. 6.9(b) shows the ground truth environment map with obstacles P_2 indicated in black. □

In this example, $N^{\mathrm{A}} = 3$ actor vehicles and $N^{\mathrm{S}} = 3$ sensors are employed in a 20×20 POM (Fig. 6.9). 6 ROIs are required to be visited at least once by the actors. Initially, the occupancy of all cells is unknown with $\Omega(v_i) = 0.5$, except for the ROIs and initial cells of the actors and sensors.

In this example, the proposed interactive planning and sensing algorithm converges in 25 iterations. The POM and information gain of each vertex $v_i \in V$ at iterations $t = 1, 10, 20, 25$ are shown in Fig. 6.10. Images in the left column indicate the POM at these iterations, along



(a) Initial POM with $p(v_i) = 0.5$ for all cells v_i.

(b) True grid map.

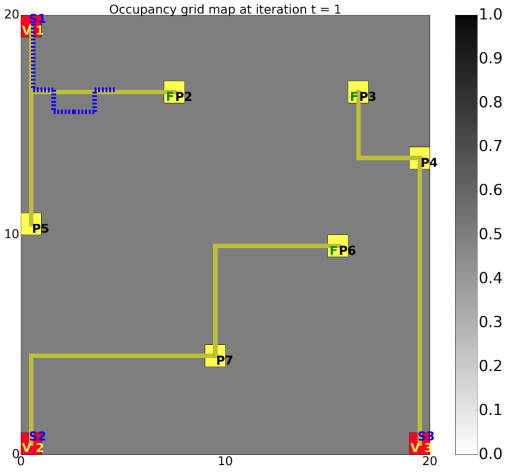
Figure 6.9: Occupancy grid map in Example 7.

with optimal actor and sensor routes (replanned for that particular iteration) in yellow and blue, respectively. Images in the right column indicate the task-driven information gain, with sensor placements (blue asterisks) at cells with the highest information gain. The final task assignment and the optimal route for each actor vehicle is shown in Fig. 6.13(a). The iterative change in entropy of the optimal route for each actor $i \in [N^{\mathrm{A}}]$ is shown in Fig. 6.11.

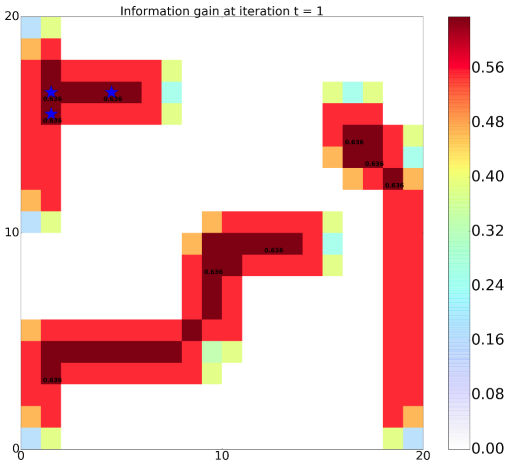
To appreciate the benefits provided by the proposed technique, consider a traditional alternative: first, information gain is computed without reference to the actors' routes, i.e., Eqn. (4.7) is evaluated for each cell in the entire map (recall that the proposed algorithm evaluates Eqn. (4.7) only for the subregion $\mathcal{W}_i''$). Next, sensors are placed at the cells with the highest information gain, and finally, a Bayesian update is applied. For a fair comparison, we iterate this alternative technique until the actors' routes satisfy the same entropy condition, namely Eqn. (4.4), as in the proposed technique. This condition ensures that the actor routes resulting from either the proposed or the alternative techniques satisfy the same confidence threshold. By the alternative approach, 50 iterations are required before this condition is satisfied.

The POM and information gain by this approach for some of these iterations in the alternative approach are shown in Fig. 6.12. In the absence of a specific subregion $\mathcal{W}_i''$ (§4.2.1) about which

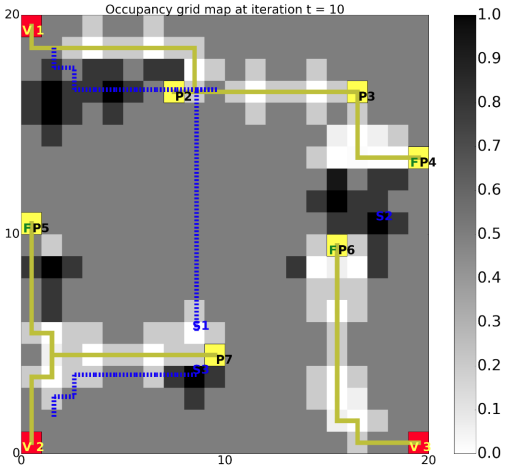
information is desired, the information gains (right column of images) are large for most of the cells in the map. Compare these images against those in Fig. 6.10.



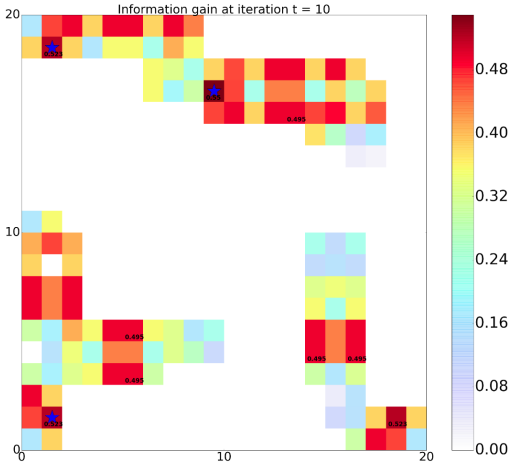
(a) Knowledge of $\mathcal{G}(t = 1)$.



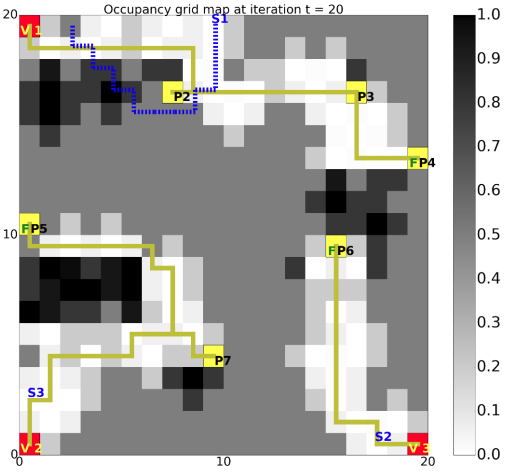
(b) Information gain at $t = 1$.



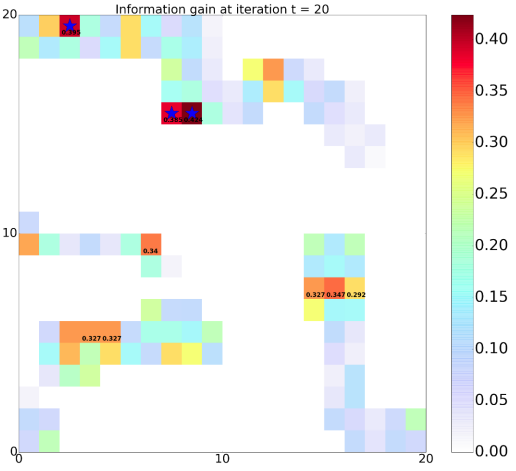
(c) Knowledge of $\mathcal{G}(t = 10)$.



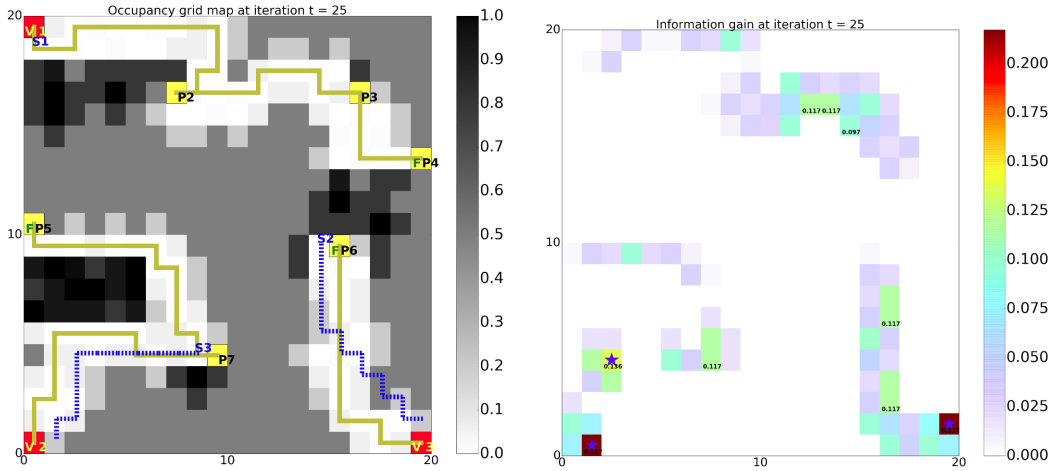
(d) Information gain at $t = 10$.



(e) Knowledge of $\mathcal{G}(t = 20)$.



(f) Information gain at $t = 20$.



(g) Knowledge of $\mathcal{G}(t = 25)$. (h) Information gain at $t = 25$.

Figure 6.10: POM and task-driven information gain at iterations t = 1, 10, 20, 25 of the proposed algorithm.

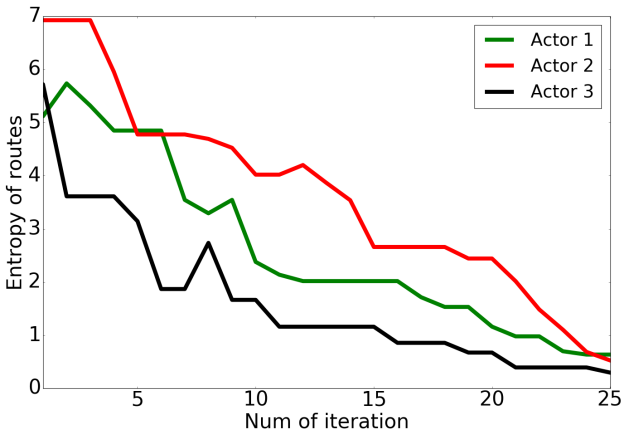
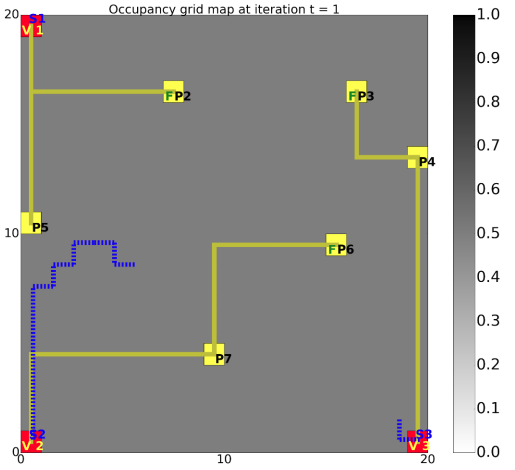


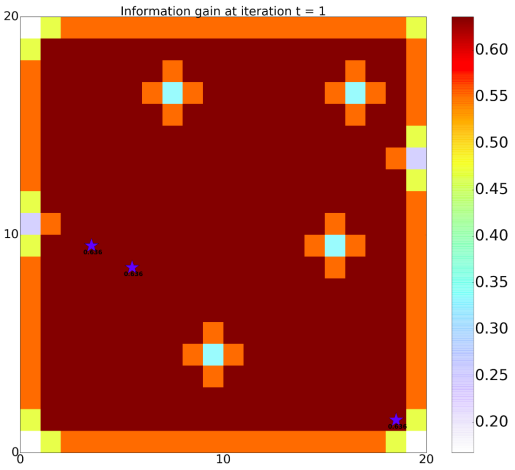
Figure 6.11: Iterative entropy changes in optimal actor routes.

Compare also the POM shown in Fig. 6.13(a) to that in Fig. 6.13(b), and the different resulting optimal routes. To achieve the same confidence threshold in actor route costs for this example, the traditional alternative needs approximately *twice* as many measurements, and the resultant POM has low entropy (i.e., low uncertainty about cell occupancy) in all regions in the environment. The proposed technique, by contrast, achieves the same confidence threshold in actor route costs with large high-entropy regions in the environment. To be precise, a comparison of the iterative entropy reduction for the entire POM is shown in Fig. 6.14. The proposed technique finds near-optimal routes with only a 35.6% reduction in entropy, whereas the traditional alternative needs a 78.2% reduction in entropy to achieve a similar result.

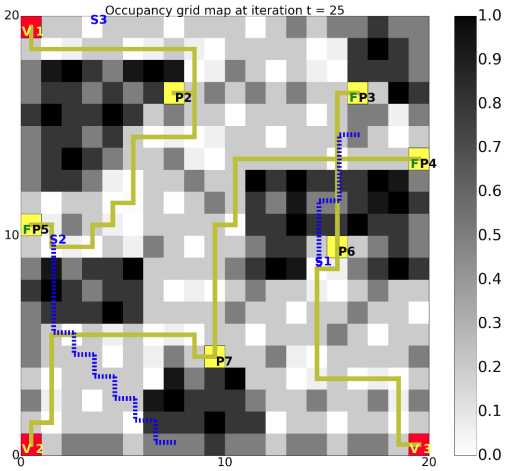
Stated differently, the gray-colored regions in Fig. 6.13(a) do not matter to the actors, and the



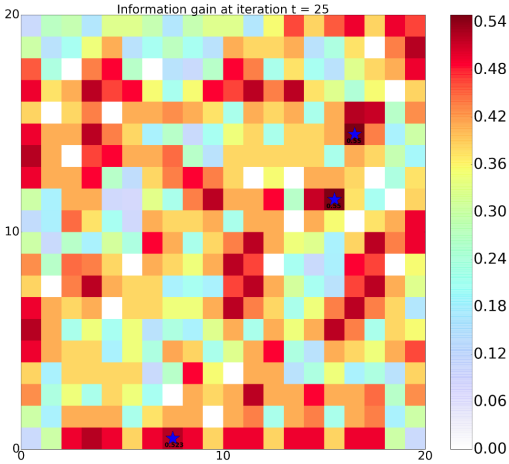
(a) Knowledge of $\mathcal{G}(t = 1)$.



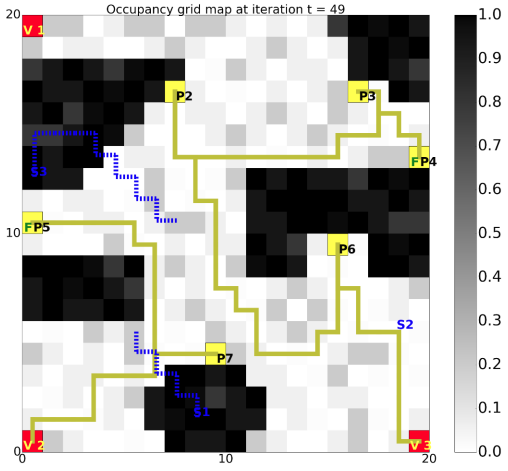
(b) Information gain at $t = 1$.



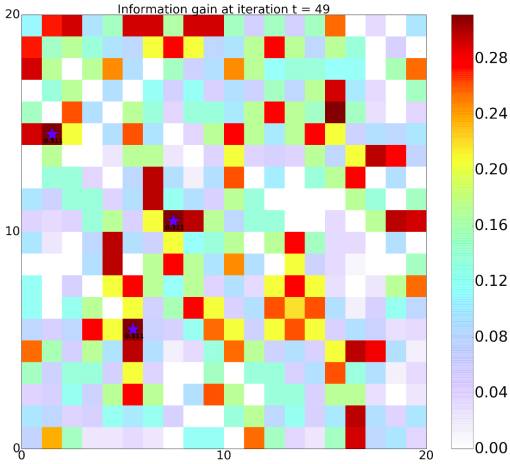
(c) Knowledge of $\mathcal{G}(t = 25)$.



(d) Information gain at $t = 25$.

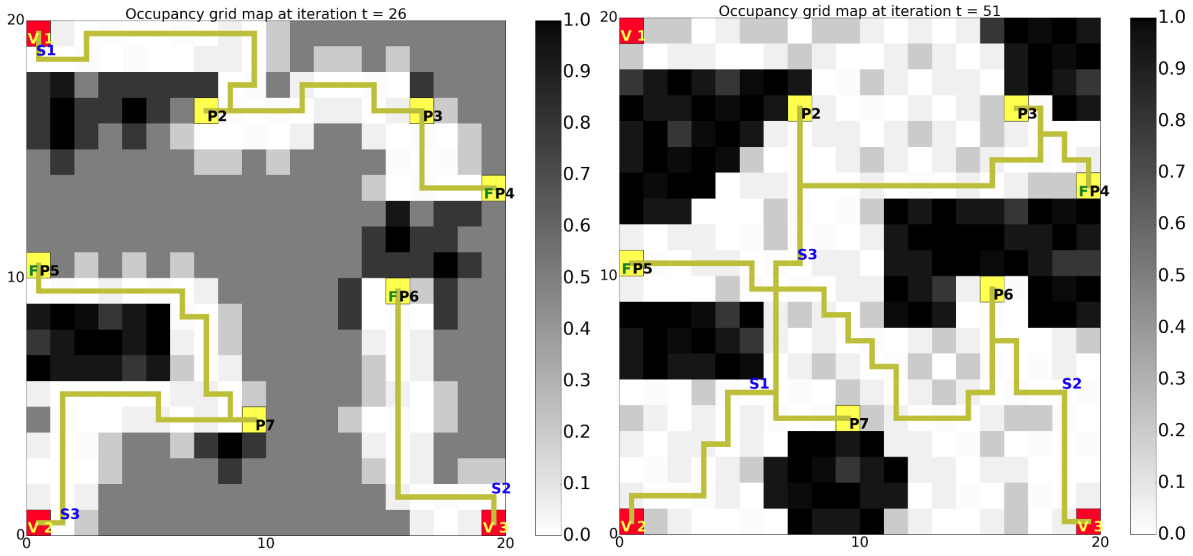


(e) Knowledge of $\mathcal{G}(t = 50)$.



(f) Information gain at $t = 50$.

Figure 6.12: POM and information gain at iterations t = 1, 25, 50 through a traditional information-driven sensor placement approach.



(a) Proposed method.

(b) Information-driven method.

Figure 6.13: Optimal routes converged from proposed technique and information-driven techniquee.

proposed technique does not place sensors in these regions. The crucial novelty of the proposed technique is that these regions are not known a priori; rather they are computed simultaneously with the actor routes. Furthermore, the total length of routes found by the proposed technique is 71 units, compared to 95 units for the routes planned by the alternative technique. The route-planning algorithm itself the same in either case, therefore the difference in cost is only due to differences in the POM. This means that not only does the proposed technique need fewer measurements, but also it finds lower cost actor routes.

Computational efficiency: For Example 7, the size of the subregions $\mathcal{W}_i''$ for each actor $i \in [N^{\mathrm{A}}]$ in each iteration of the proposed technique are shown in Fig. 6.15. The traditional alternative of finding information gain at each cell in the map is computationally expensive: Fig. 6.15 indicates that $|\mathcal{W}_i''|$ is an *order of magnitude* smaller than $|\mathcal{W}|$ in Example 7. The small size of $\mathcal{W}_i''$ significantly reduces the burden of computing information gain.

To further reduce the computational burden, Bayesian optimization is used to select the sensing locations as discussed in §5.1.3. At each iteration, $2\%|\mathcal{W}_i''|$ vertices are selected randomly as the training data to model the function of information gain. The expected improvement acquisition function with parameter $\zeta = 0.01$ is used to determine the sensors locations. The proposed interactive planning and sensing algorithm using Bayesian optimization converges in 26 iterations.

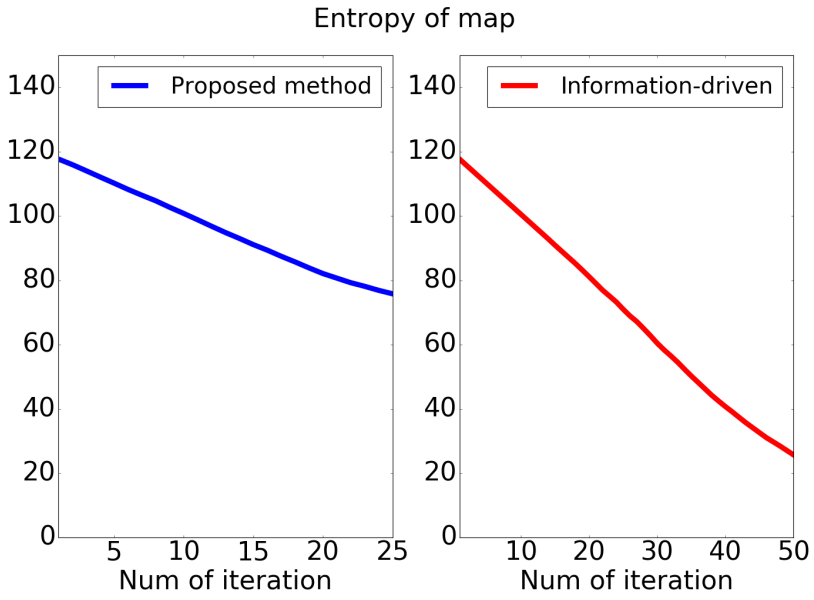


Figure 6.14: Entropy of environment $\mathcal{W}$

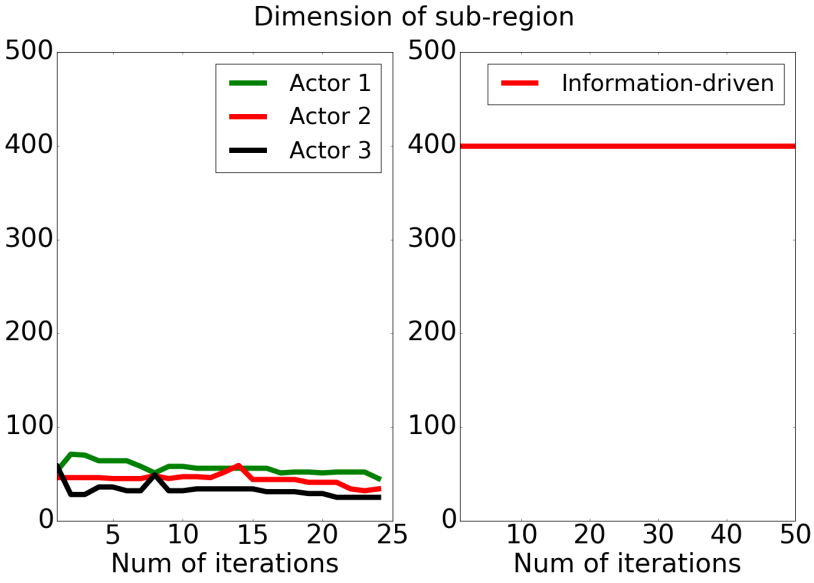
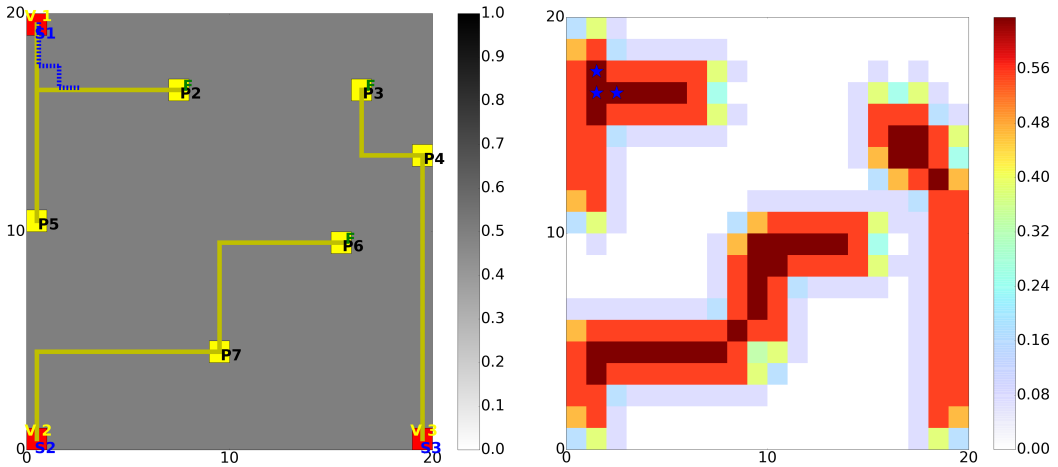


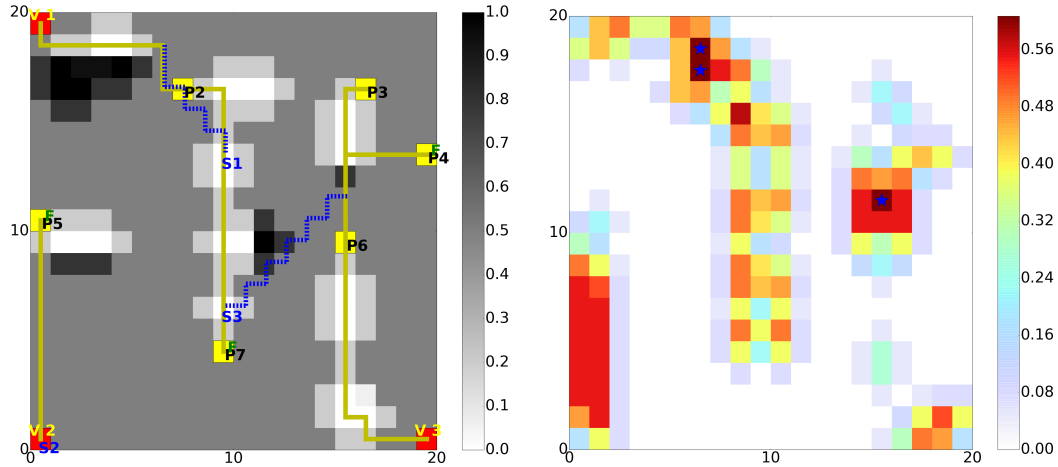
Figure 6.15: Dimension of the subregion $\mathcal{W}'(t)$

The POM and information gain of each vertex $v_i \in V$ at iterations $t = 1, 10, 20, 26$ are shown in Fig. 6.16. The computational complexity required by the proposed technique with and without Bayesian optimization is compared and shown in Fig. 6.17. Here, the computation burden is measured by the number of vertices which are required to compute the information gain. As shown in Fig. 6.17, much less number of vertices are required to evaluate the information gain to converge the sensors locations at each iteration via Bayesian optimization. Meanwhile, approximate same

convergence rate and length of total routes are computed by the proposed technique with Bayesian optimization. The accuracy to select the sensors locations with maximum information gain via Bayesian optimization is 99.8%.

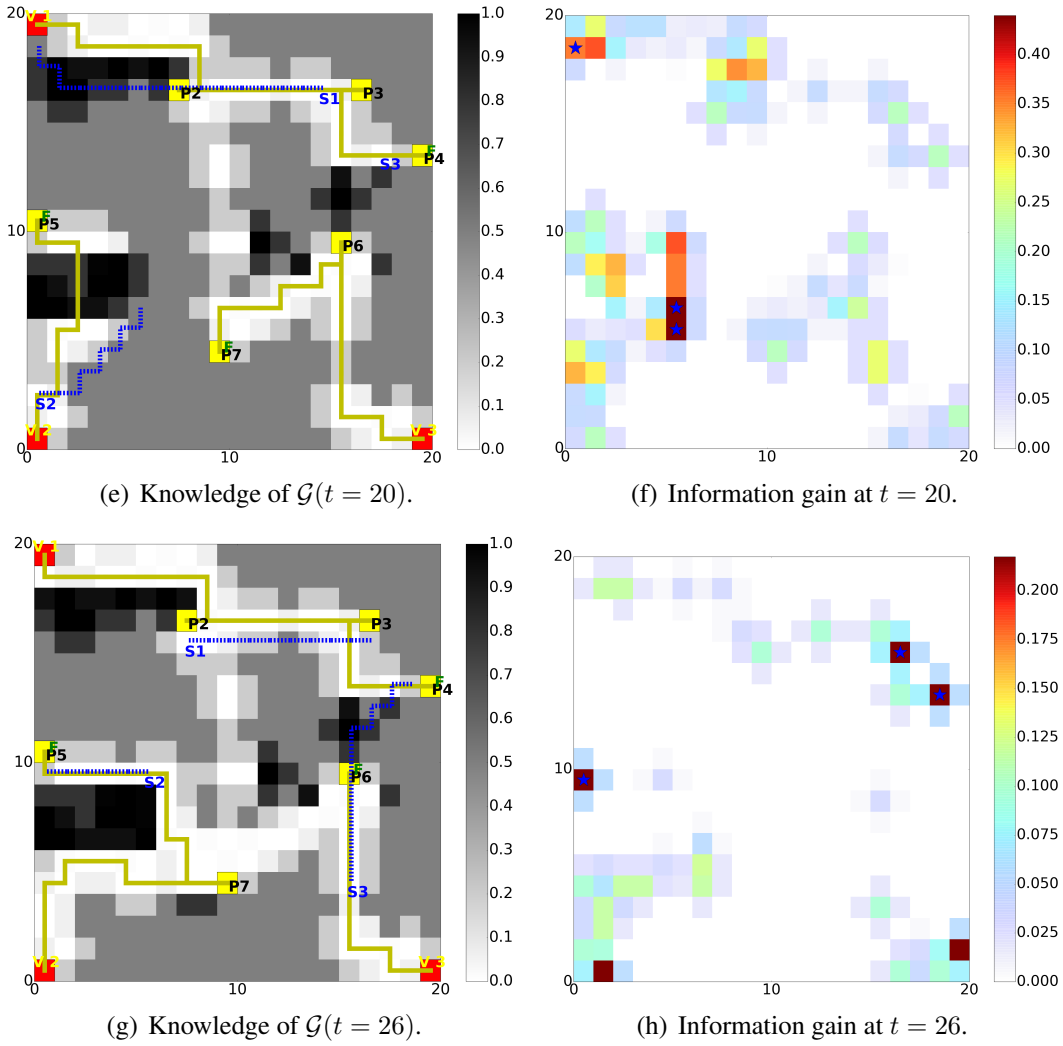


(a) Knowledge of $\mathcal{G}(t=1)$. (b) Information gain at $t=1$.



(c) Knowledge of $\mathcal{G}(t=10)$. (d) Information gain at $t=10$.

Example 8. Here is an example of mountain search and rescue. In the example, a group of hikers are trapped somewhere in the mountain. Based on their scheduled routine, they might be trapped in one of the 10 regions marked from P3–P12. $N^{\mathrm{A}} = 4$ actor vehicles initially locate in the corners are required to search all these regions at least once. Meanwhile, each actor vehicle is equipped with a sensor vehicle. The topographic map and the corresponding 30×30 grid map are shown in Fig. 6.18. Initially, we assume that the occupancy of all cells is unknown with $\Omega(v_i) = 0.5$, except for the ROIs and initial position of actors and sensors.



(e) Knowledge of $\mathcal{G}(t = 20)$.

(f) Information gain at $t = 20$.

(g) Knowledge of $\mathcal{G}(t = 26)$.

(h) Information gain at $t = 26$.

Figure 6.16: POM and task-driven information gain with Bayesian optimization at iterations t = 1, 10, 20, 26 of the proposed algorithm.

In this example, the proposed technique converges in 29 iterations. The POM and information gain of each vertex $v_i \in V$ at iterations t = 1,10,20,29 are shown in Fig. 6.19. The standard information driven approach converges in 90 iterations. The corresponding POM and information gain of each vertex $v_i \in V$ at iterations t = 1,30,60,90 are shown in Fig. 6.20. Based on the result above, much less measurement is required by the proposed technique. The total length of routes computed by the proposed technique is 102 units, compared to 131 units for the routes planned by the standard information driven approach. To further improve the computational efficiency, the proposed algorithm with Bayesian optimization is implemented which converges in 30 iterations. The POM and information gain of each vertex $v_i \in V$ at iterations t = 1,10,20,30 are shown in

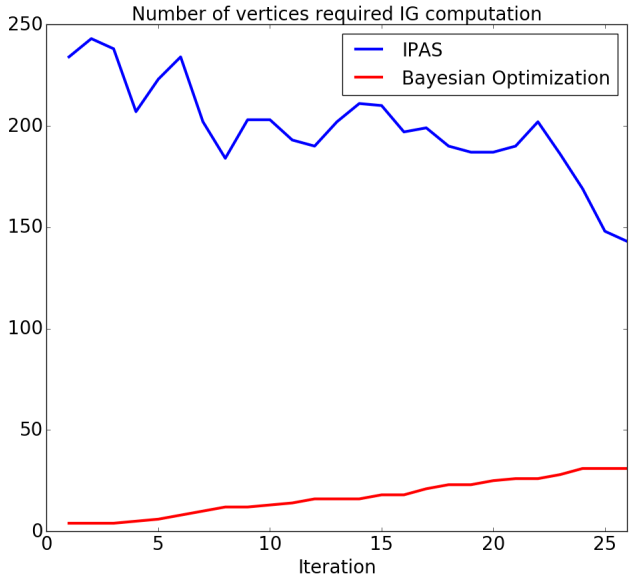
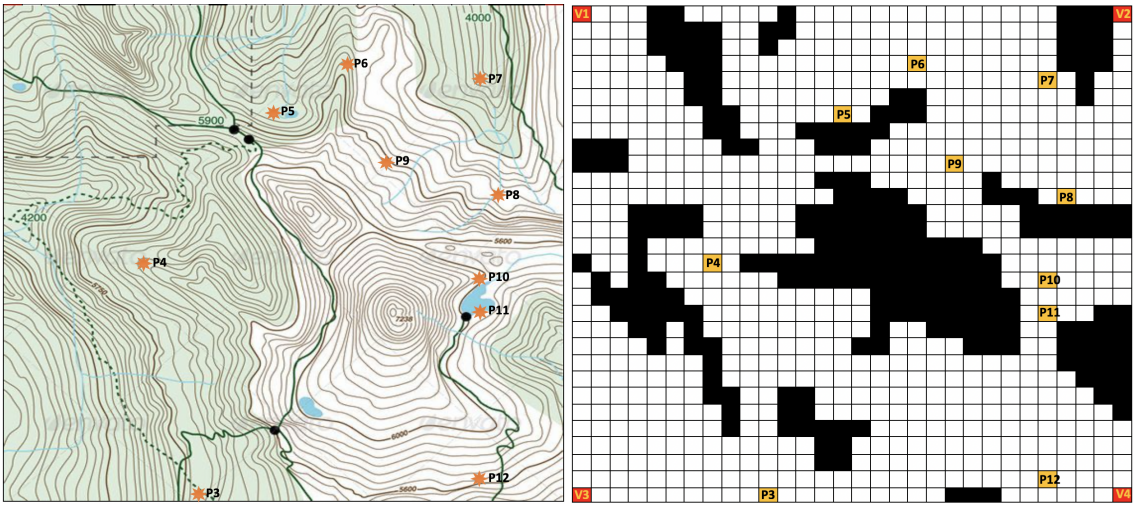


Figure 6.17: Comparison of computational efficiency

Fig. 6.20. The computation burden required by the proposed technique with and without Bayesian optimization is shown in Fig. 6.22. Bayesian optimization reduce the number of vertices for the computation of information gain. Meanwhile, the convergence rate and the total length of routes planned to satisfy the global mission are not affected by the Bayesian optimization.



(a) Topographic map of mountain. (b) Ground truth grid map.

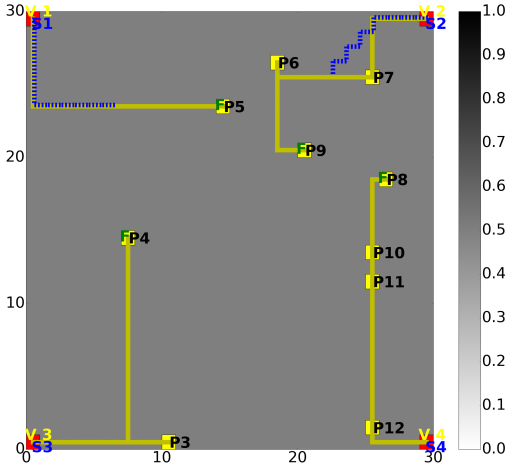
Figure 6.18: Search and rescue environment in Example 2

6.2.2 Scalability Analysis

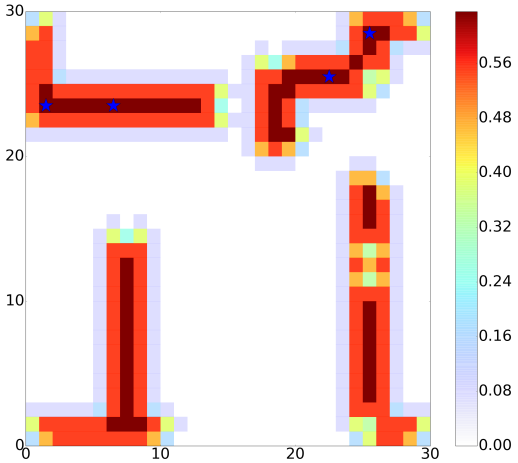
To study the scalability of the proposed technique, we performed a large number of simulation experiments with different numbers of actors, sensors, and LTL specification length (number of "tasks"), as shown in Fig. 6.23. For each case, we performed simulations over 100 randomly generated obstacle-ridden environments. The average number of iterations required for convergence of the proposed technique are indicated in Fig. 6.23. On the one hand note that, for a given number of sensors N^{S}, the numbers of actors or tasks do not affect the rate of convergence. On the other hand, note that the number of iterations for convergence sharply reduces with a moderate increase in the number of sensors, which brings us to a principal conclusion of this work: **a large number of sensors does not necessarily mean improved performance in uncertain environments; rather the judicious (task-driven) placement of even a small number of sensors can provide a desired level of performance.** By performance in this context we mean the actors' route costs and the entropy therein.

6.2.3 Threshold Analysis

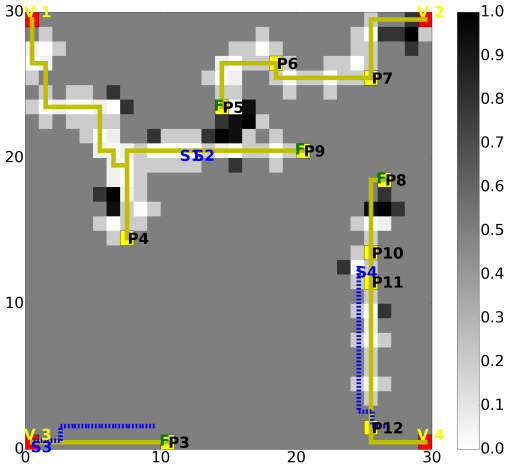
Finally, the effects of changing the threshold parameter ε in the termination condition (4.4) are shown in Fig. 6.24. As ε decreases, a higher confidence threshold is required of the actors' route costs. As expected, the total entropy reduction and the number of iterations for convergence increase moderately with decreasing ε. However, note that the relative cost error between the actual and estimated route costs does not change significantly. The average relative error for $\varepsilon \leqslant 0.06$ is approximately 6%. This observation provides another way of reducing the number of computations if needed, namely, by increasing the value of ε without significantly affecting the actors' route optimality.



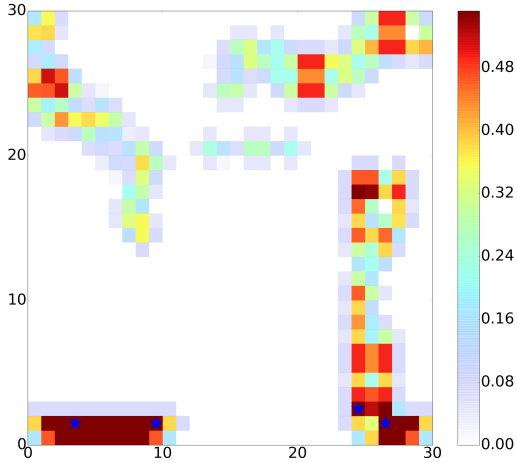
(a) Knowledge of $\mathcal{G}(t = 1)$.



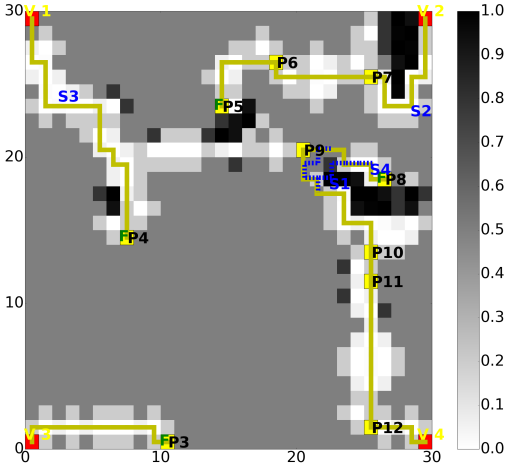
(b) Information gain at $t = 1$.



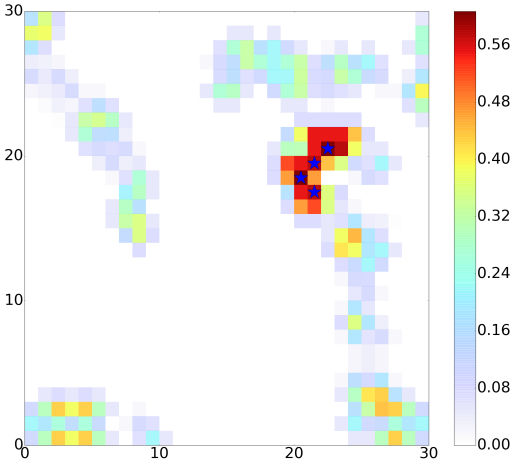
(c) Knowledge of $\mathcal{G}(t = 10)$.



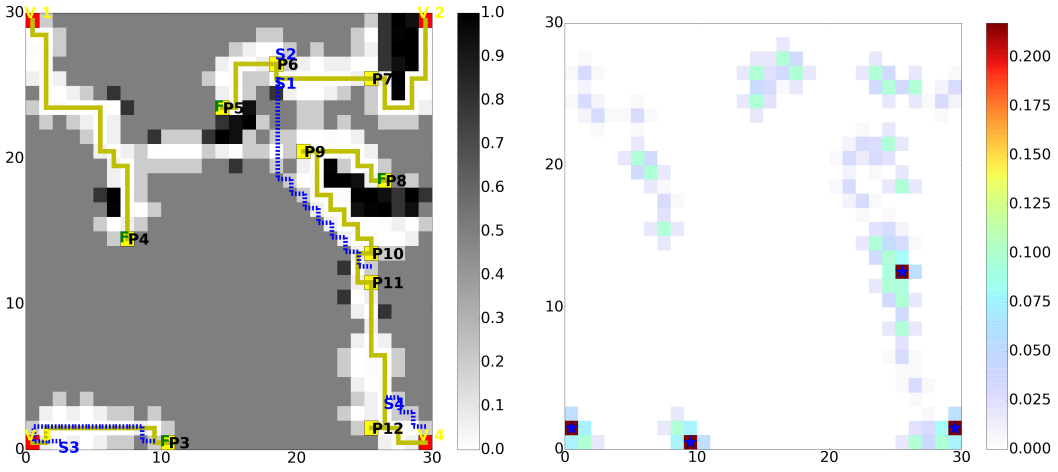
(d) Information gain at $t = 10$.



(e) Knowledge of $\mathcal{G}(t = 20)$.

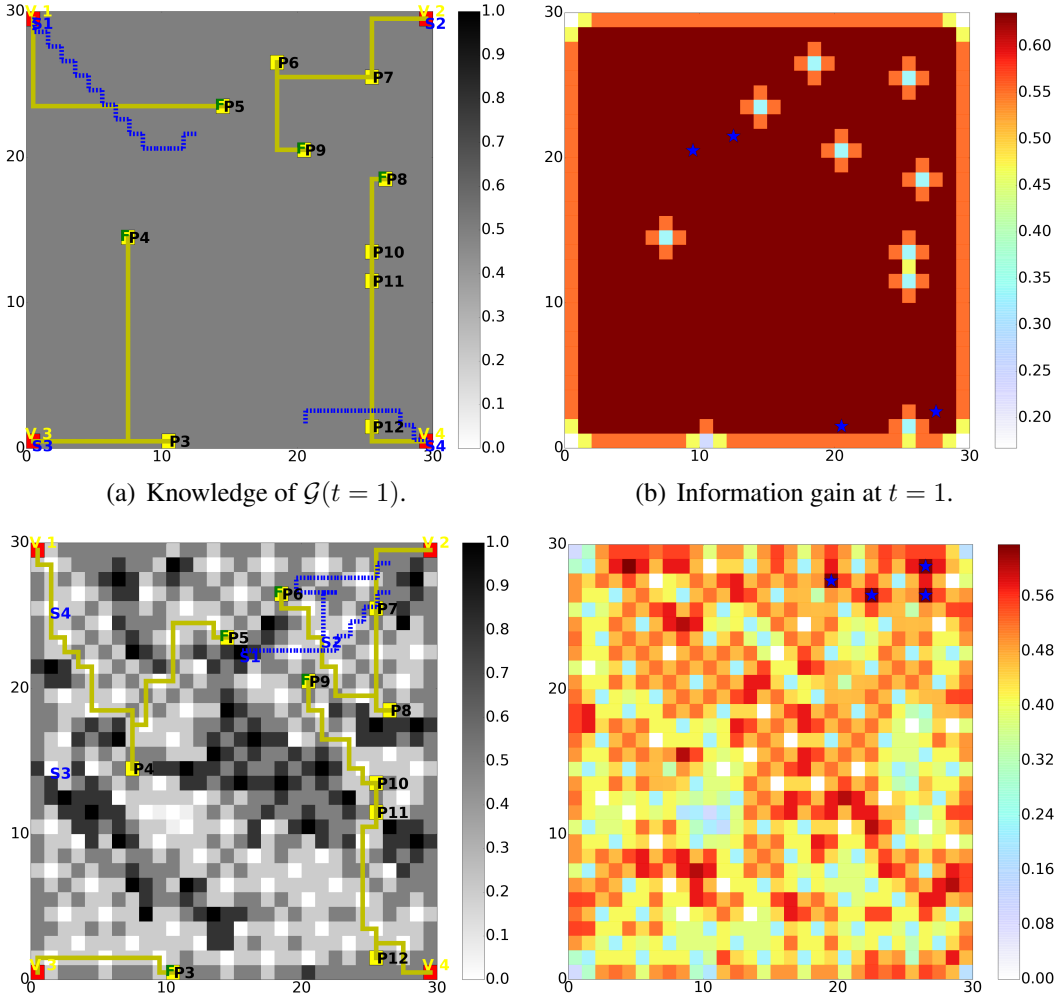


(f) Information gain at $t = 20$.



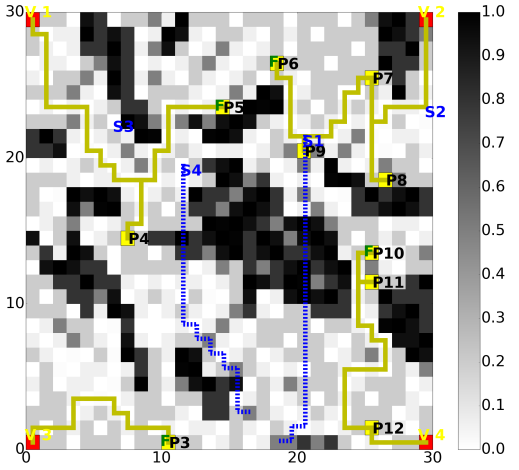
(g) Knowledge of $\mathcal{G}(t = 29)$. (h) Information gain at $t = 29$.

Figure 6.19: POM and task-driven information gain at iterations t = 1, 10, 20, 29 of the proposed algorithm for Example 2.

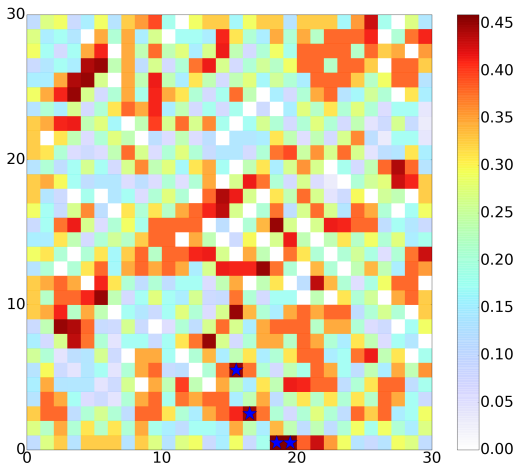


(a) Knowledge of $\mathcal{G}(t = 1)$. (b) Information gain at $t = 1$.

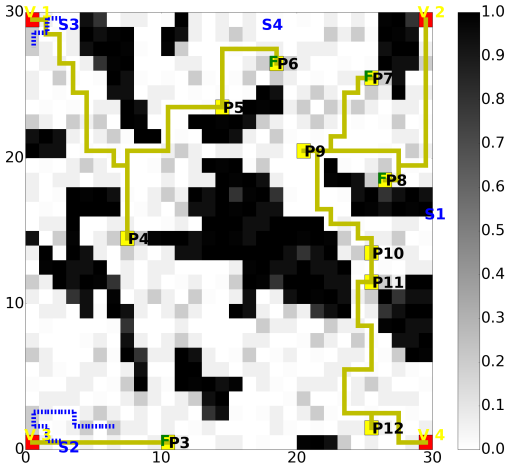
(c) Knowledge of $\mathcal{G}(t = 30)$. (d) Information gain at $t = 30$.



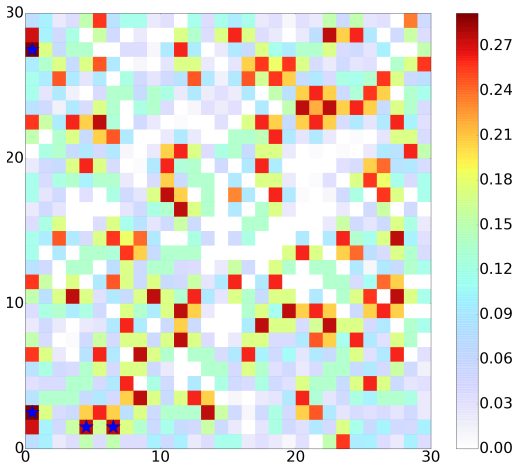
(e) Knowledge of $\mathcal{G}(t = 60)$.



(f) Information gain at $t = 60$.

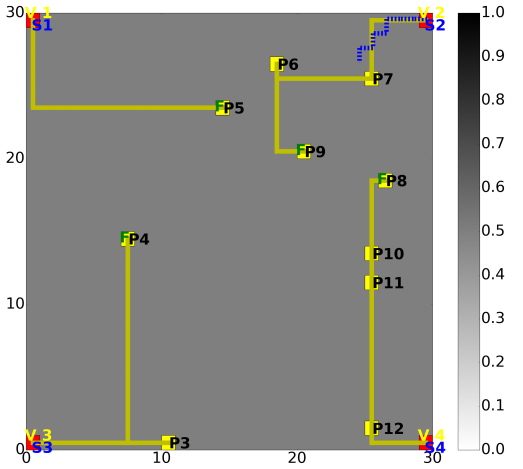


(g) Knowledge of $\mathcal{G}(t = 90)$.

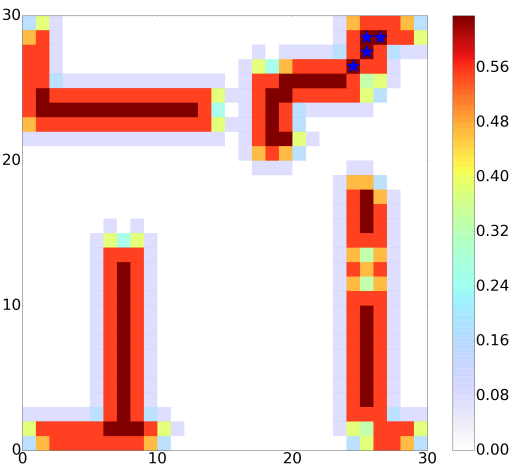


(h) Information gain at $t = 90$.

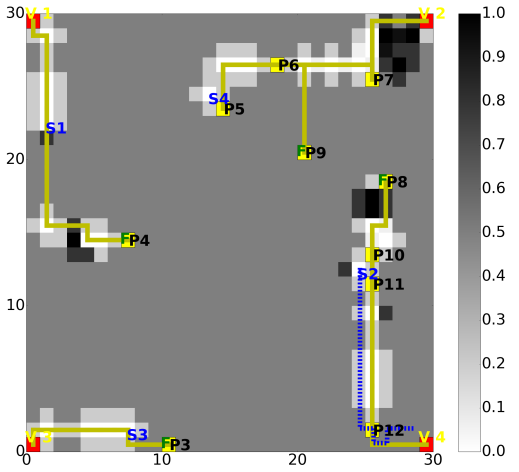
Figure 6.20: POM and task-driven information gain at iterations t = 1, 30, 60, 90 of standard information driven approach for Example 2.



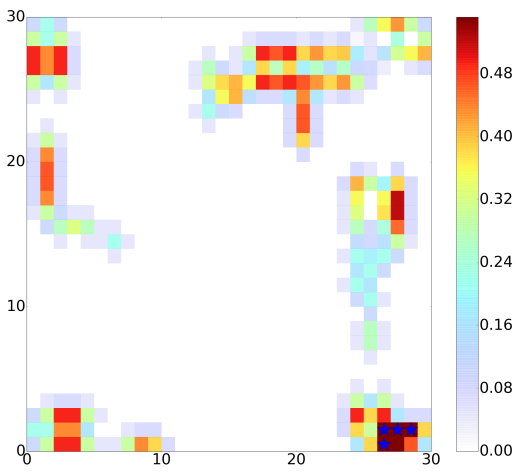
(a) Knowledge of $\mathcal{G}(t = 1)$.



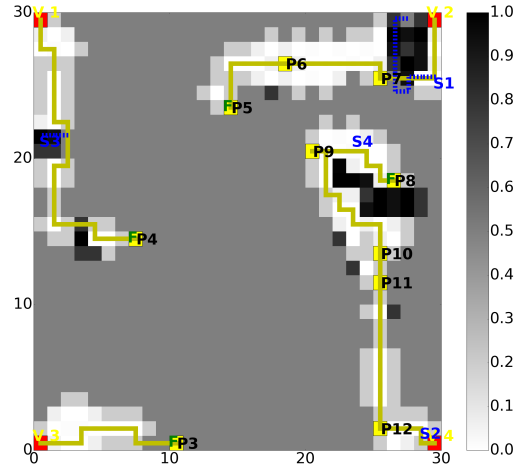
(b) Information gain at $t = 1$.



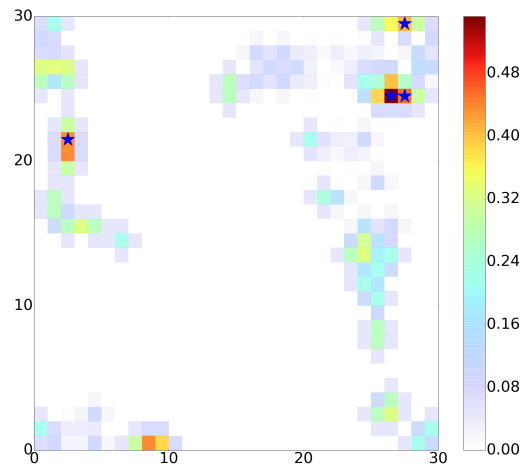
(c) Knowledge of $\mathcal{G}(t = 10)$.



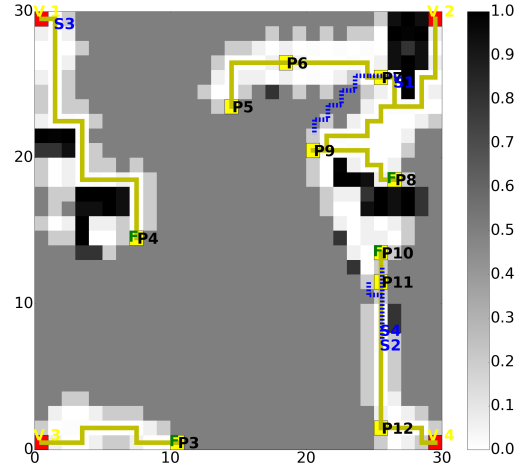
(d) Information gain at $t = 10$.



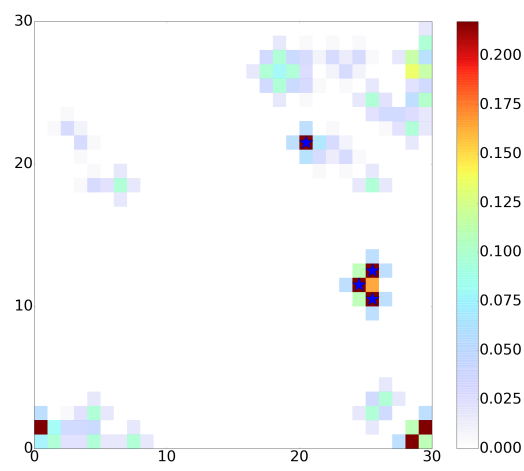
(e) Knowledge of $\mathcal{G}(t = 20)$.



(f) Information gain at $t = 20$.



(g) Knowledge of $\mathcal{G}(t = 30)$.



(h) Information gain at $t = 30$.

Figure 6.21: POM and task-driven information gain at iterations t = 1, 10, 20, 30 of the proposed algorithm for Example 2.

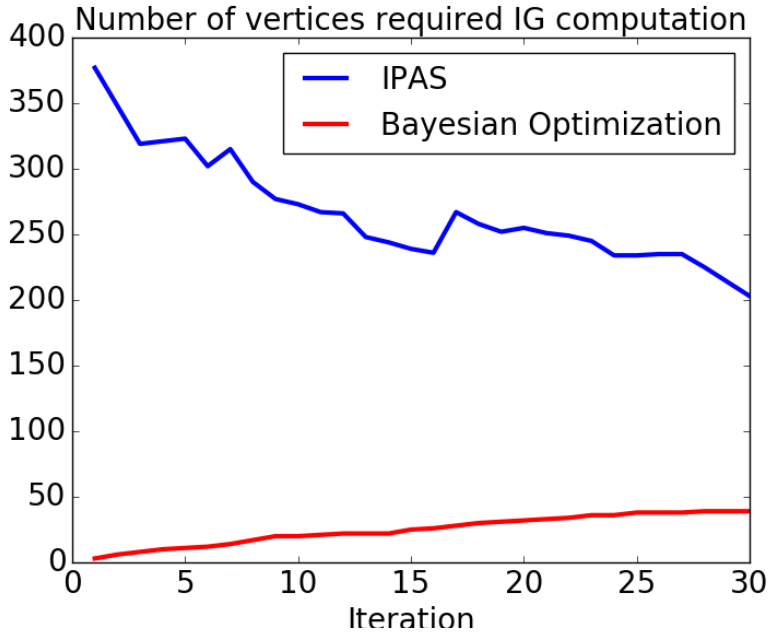


Figure 6.22: Comparison of computational efficiency for Example 2

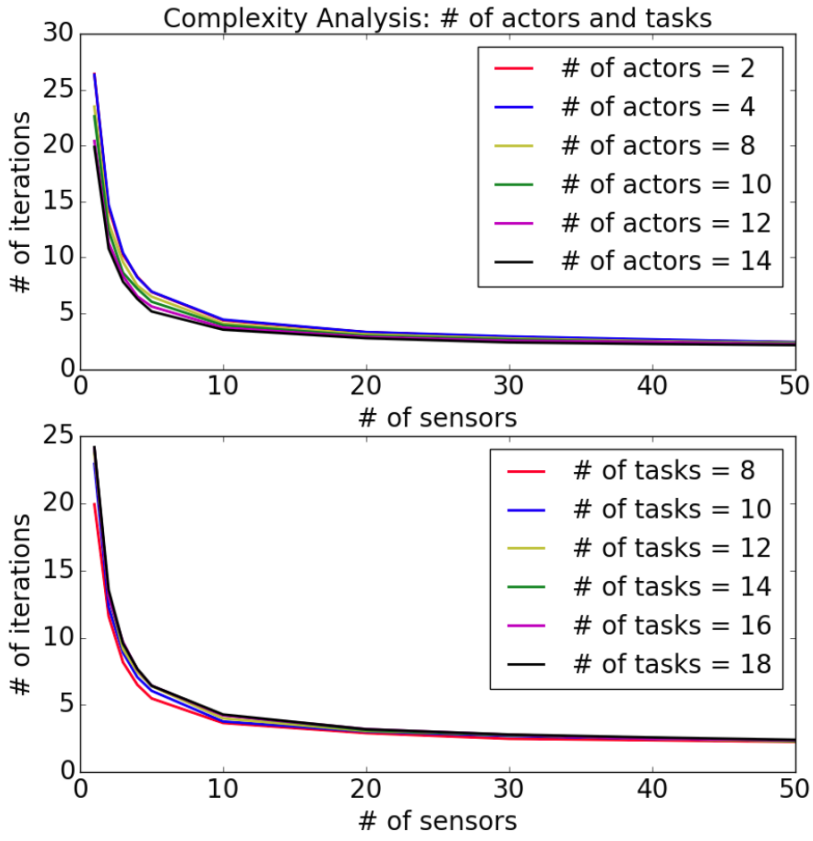


Figure 6.23: Complexity analysis for the proposed method.

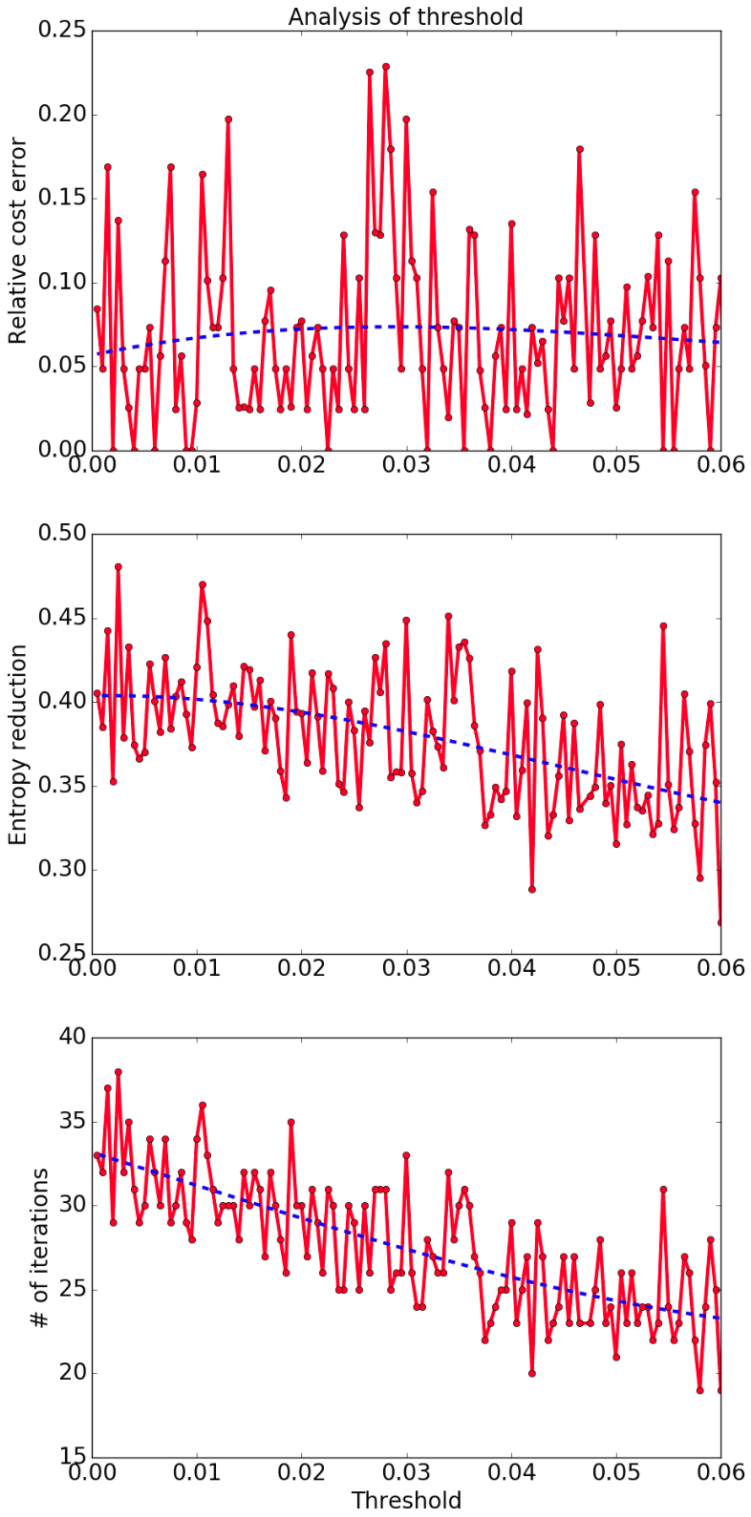


Figure 6.24: Analysis of threshold.

Chapter 7

Conclusions and Directions of Future Work

In this dissertation, we formulated and discussed the problem of decentralized route-planning for a multi-vehicle system to satisfy a given global LTL specification in an unknown environment. We decomposed the general problem into two topics: decentralized route-planning in known environments and unknown environment exploration. For route-planning of the multi-vehicle system in known environments, a decentralized route-planning approach is developed to determine the collaboration among team of vehicles to satisfy a given global LTL specification while minimizing the total traveled distance. For the unknown environment exploration, the concept of information gain is implemented to quantify the potential benefits of taking new measurements. Different from the standard information driven approach which aims to recover the whole environment as much as possible, task-driven information gain is defined to guide sensors to explore the unknown environment. By integrating these two topics, we proposed an interactive planning and sensing algorithm to guide the group of sensors to collect the most useful environment information and update the environment map for the decentralized route-planning to satisfy the requested mission. The code related to all the work discussed in this dissertation can be found in https://github.com/jfangwpi/task_allocation.git.

We addressed the decentralized route-planning problem by assuming perfect knowledge of environment is given a priori. We are looking for optimal routes for each vehicle to satisfy the global LTL specification in a 2D grid map. In this part, both independent and collaborative tasks are considered. A conflict-free task assignment can be determined by the proposed algorithms discussed in Chapter 2 and Chapter 3 over a static communication topology. The proposed algorithm integrates the consensus-based auction algorithm and synchronization algorithm to minimize the total traveled distance and waiting time. Once the local specification for each vehicle to work collaboratively has converged, Dijkastra's algorithm is used to compute the optimal routes. A comparison to the existing technique and the analysis of the convergence rate and optimality for the proposed

technique is given.

After gaining better understanding of the route-planning problem for multi-vehicle system, we removed the assumption of accurate knowledge of environment. The unknown environment is represented by a probability occupancy map. To solve the route-planning problem in unknown environments, two sets of vehicles are considered is this work: *actors* and *sensors*. The group of sensors, which are equipped with cameras or thermal detectors are guided to explore certain regions. Meanwhile, actors are required to work collaboratively to satisfy the global task. The concept of task-driven information gain is developed to identified a set of "informative" subregions to take measurements "near" the actors' current optimal routes and reduce the uncertainty.

To identify a set of locations with maximum information gain for sensors to take measurements, the evaluation of information gain for each potential sensors locations is required. This will be very computationally expensive, especially when the environment is large. To reduce the computational burden from computing the information gain, we introduced a machine-learning based technique: Bayesian optimization, to identify a set of "informative" regions without computing information gain for each potential vertex. The function for information gain is modeled via Gaussian process regression and an expected improvement acquisition function is then used to select sensors locations.

The following are possible future extensions of the proposed interactive planning and sensing framework.

Consider more complicated LTL specification. In this dissertation, certain temporal and logic operators can be used to specify the LTL specification. As a result, some types of missions are not covered. In the future work, more general LTL library should be implemented to take more complex LTL specifications into consideration.

Route-planning for Multiple Vehicles System in Time-varying Environments. In this dissertation, we assume the unknown environment is static. However, in practical cases such as the disaster scenario like floods or hurricanes, the environment will change along with time. Certain time-varying function can be selected to model such environment. Some modifications are re-

quired to the current algorithm to address the time-varying environment. For example, how sensor locations should be determined to ensure the latest and most valuable information is collected and how the convergence is guaranteed?

Task Assignment by Machine Learning Technique. In this dissertation, we implemented a decentralized route-planning approach to decompose the global LTL specification into local specifications and enable the collaboration among the team of vehicles. The convergence of the task assignment can be achieved by exchanging the reward of each task among team of vehicles. Certain reinforcement learning technique could be implemented to determine the task assignment with the inputs: environment, vehicles' initial positions and targets positions. The difficulty to implement such machine learning technique is to select the features to train the model. Furthermore, the optimality required by the machine learning technique should be considered and a comparison between the machine learning technique and current decision making algorithm can be addressed.

Improvement of Interactive Planning and Sensing by Considering Sensors' Transit. In this dissertation, we assume the sensors take measurement at selected cells, but not along the routes traveled by the sensors. In the future work, we can remove this assumption and consider the route-planning problem for sensors. Now the objective of sensors's route-planning is to computes routes for the group of sensors to maximize the total information collected along the routes. A trade-off between minimizing the total travel distance and maximize the total collected information should be considered.

Improvement of Coding The code shown in https://github.com/jfangwpi/task_allocation.git contains three main parts: creating a graph to represent either known or unknown grid map, route-planning, and task assignment. The main computational burden comes from updating reward or route-planning for certain scenarios, like considering vehicle's kinematic constraints. For this particular case, a search table constructed by CBTA can be built priori and the corresponding reward required by the task assignment can be updated by searching in this table. Meanwhile, certain efforts can also be placed on the task assignment algorithm itself and avoid computing some unnecessary rewards repeatedly which will definitely increase the computation efficiency.

Bibliography

[1] Hillary K. Grigonis, "Land rover's search and rescue suv launch infrared-equipped drones," 2017, [Online; accessed March 7, 2017]. [Online]. Available: https://www.digitaltrends.com/cool-tech/land-and-air-rover-red-cross/

[2] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI Commun.*, vol. 29, no. 1, pp. 151–162, 2016.

[3] S. L. Smith, J. Tumova, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *Int. J. Rob. Res.*, vol. 30, no. 14, pp. 1695–1708, Dec. 2011.

[4] Y. Shoukry, P. Nuzzo, A. Balkan, I. Saha, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "Linear temporal logic motion planning for teams of underactuated robots using satisfiability modulo convex programming," in *2017 IEEE 56th Conf. Decision and Control (CDC)*, Dec 2017, pp. 1132–1137.

[5] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local LTL specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, 2016.

[6] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[7] C. Phan and H. H. Liu, "A cooperative uav/ugv platform for wildfire detection and fighting," in *2008 Asia Simulation Conference-7th International Conference on System Simulation and Scientific Computing*. IEEE, 2008, pp. 494–498.

[8] M. Kloetzer and C. Belta, "Temporal logic planning and control of robotic swarms by hierarchical abstractions," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 320–330, 2007.

[9] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.

[10] M. Guo and D. V. Dimarogonas, "Reconfiguration in motion planning of single-and multi-agent systems under infeasible local ltl specifications," in *52nd IEEE Conference on Decision and Control*. IEEE, 2013, pp. 2758–2763.

[11] J. J. Tumova and D. V. Dimarogonas, "A receding horizon approach to multi-agent planning from local ltl specifications," in *2014 American Control Conference*. IEEE, 2014, pp. 1775–1780.

[12] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[13] E. A. Emerson, "Temporal and modal logic," *Handbook of Theoretical Computer Science, Volume B:*, 1990.

[14] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science*, Providence, RI, USA, October 31 - November 2 1977, pp. 46–57.

[15] V. S. Alagar and K. Periasamy, *Specification of Software Systems*, 2nd ed. London, UK: Springer-Verlag, 2011.

[16] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot motion planning and control in polygonal environments," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 864 – 874, October 2005.

[17] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Hybrid controllers for path planning: A temporal logic approach," in *Proceedings of the 44th IEEE Conference on Decision and Control*, Seville, Spain, 12 – 15 Dec. 2005, pp. 4885 – 4890.

[18] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion," *IEEE Robotics and Automation Magazine*, pp. 61 – 70, March 2007.

[19] P. Tabuada and G. J. Pappas, "Model checking LTL over controllable linear systems is decidable," in *Hybrid Systems: Computation & Control*, ser. LNCS 2623, O. Maler and A. Pnueli, Eds. Berlin: Springer-Verlag, 2003, pp. 498–513.

[20] P. Tabuada, "An approximate simulation approach to symbolic control," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1406–1418, 2008.

[21] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, February 2008.

[22] P. Wolper, "Constructing automata from temporal logic formulas: A tutorial," in *School organized by the European Educational Forum*. Springer, 2000, pp. 261–277.

[23] G. Holzmann, "The model checker SPIN," *IEEE Transactions on Software Engineering*, vol. 23, no. 5, pp. 279–295, 1997.

[24] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *Proceedings of the 13th Conference on Computer Aided Verification (CAV' 01)*, ser. Lecture Notes in Computer Science, H. C. G. Berry and A. Finkel, Eds., vol. 2102. New York: Springer-Verlag, 2001, pp. 53–65.

[25] A. Duret-Lutz, "Manipulating ltl formulas using spot 1.0," in *Automated technology for verification and analysis*. Springer, 2013, pp. 442–445.

[26] M. Kloetzer and C. Belta, "Temporal logic planning and control of robotic swarms by hierarchical abstractions," *IEEE Transaction on Robotics*, vol. 23, no. 2, pp. 320–330, 2007.

[27] T. Wongpiromsarn, "Formal methods for design and verification of embedded control systems: Application to an autonomous vehicle," Ph.D. dissertation, California Institute of Technology, 2010.

[28] X.-C. Ding, M. Lazar, and B. C, "Formal abstraction of linear systems via polyhedral Lyapunov functions," in *Proceedings of the 4th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS)*, Eindhoven, The Netherlands, Jun 6–8 2012.

[29] G. Reissig, "Computing abstractions of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 11, pp. 2583–2598, 2011.

[30] M. Zamani, G. Pola, M. Mazo, and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pp. 1804–1809, 2012.

[31] M. Rungger and M. Zamani, "Scots: A tool for the synthesis of symbolic controllers," in *Proceedings of the 19th international conference on hybrid systems: Computation and control*, 2016, pp. 99–104.

[32] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *Int. J. Rob. Res.*, vol. 34, no. 2, pp. 218–235, Feb. 2015.

[33] R. V. Cowlagi and Z. Zhang, "Route guidance for satisfying temporal logic specifications on aircraft motion," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 390–401, 2017.

[34] R. V. Cowlagi and P. Tsiotras, "Curvature-bounded traversability analysis in motion planning for mobile robots," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 1011–1019, 2014.

[35] Y. Chen, X. C. Ding, and C. Belta, "Synthesis of distributed control and communication schemes from global LTL specifications," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, Dec. 2011, pp. 2718–2723.

[36] M. Karimadini and H. Lin, "Synchronized task decomposition for two cooperative agents," in *2010 IEEE Conference on Robotics, Automation and Mechatronics*, Jun. 2010, pp. 368–373.

[37] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *International Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.

[38] P. Schillinger, M. Brger, and D. V. Dimarogonas, "Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 818–838, 2018.

[39] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multi-agent motion tasks based on LTL specifications," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 1, Dec. 2004, pp. 153–158 Vol.1.

[40] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Decomposition of finite ltl specifications for efficient multi-agent planning," in *Distributed Autonomous Robotic Systems*. Springer, 2018, pp. 253–267.

[41] Y. Kantaros and M. M. Zavlanos, "Temporal logic optimal control for Large-Scale Multi-Robot systems: 10400 states and beyond," in *2018 IEEE Conference on Decision and Control (CDC)*, Dec. 2018, pp. 2519–2524.

[42] I. Filippidis, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Decentralized multi-agent control from local LTL specifications," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, Dec. 2012, pp. 6235–6240.

[43] M. Guo, J. Tumova, and D. V. Dimarogonas, "Communication-free multi-agent control under local temporal tasks and relative-distance constraints," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 3948–3962, 2016.

[44] S. Moarref and H. Kress-Gazit, "Decentralized control of robotic swarms from high-level temporal logic specifications," in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, Dec 2017, pp. 17–23.

[45] M. Guo and D. V. Dimarogonas, "Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 797–808, April 2017.

[46] Y. Kantaros and M. M. Zavlanos, "Distributed optimal control synthesis for multi-robot systems under global temporal tasks," *Proceedings of the 9th ACM/IEEE*, 2018.

[47] M. Rungger and M. Zamani, "Scots: A tool for the synthesis of symbolic controllers," in *Hybrid Systems Computation and Control*, 2016.

[48] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning with deterministic μ-calculus specifications," in *2012 American Control Conference (ACC)*, Jun. 2012, pp. 735–742.

[49] C. I. Vasile and C. Belta, "Sampling-based temporal logic path planning," 2013.

[50] S. S. Dhillon and K. Chakrabarty, "Sensor placement for effective coverage and surveillance in distributed sensor networks," in *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, vol. 3. IEEE, 2003, pp. 1609–1614.

[51] S. S. Dhillon, K. Chakrabarty, and S. S. Iyengar, "Sensor placement for grid coverage under imprecise detections," in *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002.(IEEE Cat. No. 02EX5997)*, vol. 2. IEEE, 2002, pp. 1581–1587.

[52] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Near-optimal sensor placements: Maximizing information while minimizing communication cost," in *Proceedings of the 5th international conference on Information processing in sensor networks*, 2006, pp. 2–10.

[53] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE transactions on computers*, vol. 51, no. 12, pp. 1448–1453, 2002.

[54] V. Loscri, E. Natalizio, F. Guerriero, and N. Mitton, "Efficient coverage for grid-based mobile wireless sensor networks," in *Proceedings of the 11th ACM symposium on Performance evaluation of wireless ad hoc, sensor, & ubiquitous networks*, 2014, pp. 53–60.

[55] D. Cochran and A. O. Hero, "Information-driven sensor planning: Navigating a statistical manifold," in *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 2013, pp. 1049–1052.

[56] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 235–284, 2008.

[57] D. F. Wolf and G. S. Sukhatme, "Mobile robot simultaneous localization and mapping in dynamic environments," *Auton. Robots*, vol. 19, no. 1, pp. 53–65, Jul. 2005.

[58] D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Occupancy grid models for robot mapping in changing environments," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[59] G. Tanzmeister, J. Thomas, D. Wollherr, and M. Buss, "Grid-based mapping and tracking in dynamic environments using a uniform evidential environment representation," in *2014 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2014, pp. 6090–6095.

[60] T. Colleens, J. J. Colleens, and D. Ryan, "Occupancy grid mapping: An empirical evaluation," in *2007 Mediterranean Conference on Control Automation*, Jun. 2007, pp. 1–6.

[61] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, Aug. 2005.

[62] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Auton. Robots*, vol. 15, no. 2, pp. 111–127, Sep. 2003.

[63] E. Kaufman, T. Lee, Z. Ai, and I. Moskowitz, "Bayesian occupancy grid mapping via an exact inverse sensor model," *2016 American Control Conference (ACC)*, 2016.

[64] E. Kaufman, K. Takami, Z. Ai, and T. Lee, "Autonomous quadrotor 3D mapping and exploration using exact occupancy probabilities," *2018 IEEE Intl. Conf. Robotic Computing (IRC)*, 2018.

[65] E. Kaufman, T. Lee, and Z. Ai, "Autonomous exploration by expected information gain from probabilistic occupancy grid mapping," in *2016 IEEE Intl. Conf. Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, Dec. 2016, pp. 246–251.

[66] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, Nov. 2012.

[67] B. J. Julian, S. Karaman, and D. Rus, "On mutual information-based control of range sensing robots for mapping applications," *Int. J. Rob. Res.*, vol. 33, no. 10, pp. 1375–1392, Sep. 2014.

[68] F. Amigoni and V. Caglioti, "An information-based exploration strategy for environment mapping with mobile robots," *Rob. Auton. Syst.*, vol. 58, no. 5, pp. 684–699, May 2010.

[69] A. Krause and C. Guestrin, "Near-optimal observation selection using submodular functions," in *AAAI*, vol. 7, 2007, pp. 1650–1654.

[70] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[71] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "Variational information maximizing exploration," *Advances in Neural Information Processing Systems 29 (NIPS)*, p. 1109, May 2016.

[72] L. Paull, S. Saeedi, H. Li, and V. Myers, "An information gain based adaptive path planning method for an autonomous underwater vehicle using sidescan sonar," *2010 IEEE International Conference on Automation Science and Engineering*, 2010.

[73] G. Ferri, M. V. Jakuba, A. Mondini, V. Mattoli, B. Mazzolai, D. R. Yoerger, and P. Dario, "Mapping multiple gas/odor sources in an uncontrolled indoor environment using a bayesian occupancy grid mapping based method," *Rob. Auton. Syst.*, vol. 59, no. 11, pp. 988–1000, Nov. 2011.

[74] G. M. Hoffmann and C. J. Tomlin, "Mobile sensor network control using mutual information methods and particle filters," *IEEE Trans. Automat. Contr.*, vol. 55, no. 1, pp. 32–47, Jan. 2010.

[75] M. Sinay, N. Agmon, O. Maksimov, G. Levy, M. Bitan, and S. Kraus, "UAV/UGV search and capture of Goal-Oriented uncertain Targets*," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[76] A. Singh, A. Krause, C. Guestrin, W. J. Kaiser, and M. A. Batalin, "Efficient planning of informative paths for multiple robots," in *IJCAI*, vol. 7, 2007, pp. 2204–2211.

[77] A. Krause and C. E. Guestrin, "Near-optimal nonmyopic value of information in graphical models," Jul. 2012.

[78] Y. Zou and K. Chakrabarty, "Uncertainty-aware and coverage-oriented deployment for sensor networks," *Journal of Parallel and Distributed Computing*, vol. 64, no. 7, pp. 788–798, 2004.

[79] S. Hunt, Q. Meng, C. Hinde, and T. Huang, "A Consensus-Based grouping algorithm for multi-agent cooperative task allocation with complex requirements," *Cognit. Comput.*, vol. 6, no. 3, pp. 338–350, Apr. 2014.

[80] D. Stavrou, S. Timotheou, C. G. Panayiotou, and M. M. Polycarpou, "Optimizing container loading with autonomous robots," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 717–731, April 2018.

[81] C. Belta, B. Yordanov, and E. A. Gol, *Formal methods for discrete-time dynamical systems*. Springer, 2017, vol. 89.

[82] R. V. Cowlagi and P. Tsiotras, "Hierarchical motion planning with dynamical feasibility guarantees for mobile robotic vehicles," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 379 – 395, 2012.

[83] R. V. Cowlagi and Z. Zhang, "Route guidance for satisfying temporal logic specifications on aircraft motion," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 390–401, 2017.

[84] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Transaction on Robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[85] H. L. Choi, L. Brunet, and J. P. How, "Consensus-Based decentralized auctions for robust task allocation," *IEEE Trans. Rob.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.

[86] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.

[87] A. S. Huang, E. Olson, and D. C. Moore, "Lcm: Lightweight communications and marshalling," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 4057–4062.

[88] R. Ikhankar, V. Kuthe, S. Ulabhaje, S. Balpande, and M. Dhadwe, "Pibot: The Raspberry Pi controlled multi-environment robot for surveillance amp;amp; live streaming," in *2015*

International Conference on Industrial Instrumentation and Control, May 2015, pp. 1402–1405.