



AFRL-AFOSR-VA-TR-2022-0035

Multiscale Dynamic Data Driven Guidance and Control for Autonomous Vehicle Networks

Cowlagi, Raghvendra
WORCESTER POLYTECHNIC INST MA
100 INSTITUTE RD
WORCESTER, MA, 01609-2280
US

09/08/2021
Final Technical Report

DISTRIBUTION A: Distribution approved for public release.
--

Air Force Research Laboratory
Air Force Office of Scientific Research
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 08-09-2021		2. REPORT TYPE Final		3. DATES COVERED (From - To) 01 Jan 2017 - 31 Dec 2020	
4. TITLE AND SUBTITLE Multiscale Dynamic Data Driven Guidance and Control for Autonomous Vehicle Networks				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER FA9550-17-1-0028	
				5c. PROGRAM ELEMENT NUMBER 61102F	
6. AUTHOR(S) Raghvendra Cowlagi				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) WORCESTER POLYTECHNIC INST MA 100 INSTITUTE RD WORCESTER, MA 01609-2280 US				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AF Office of Scientific Research 875 N. Randolph St. Room 3112 Arlington, VA 22203				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/AFOSR RTA2	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-AFOSR-VA-TR-2022-0035	
12. DISTRIBUTION/AVAILABILITY STATEMENT A Distribution Unlimited: PB Public Release					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT This project investigates coupled sensor configuration and planning (CSCP) for autonomous systems, which is a mode of active control of information in context to a decision-making problem. We consider a scenario consisting of a network of mobile vehicles called sensors, and a separate network of mobile vehicles called actors. Sensors gather information about the environment whereas actors perform desired tasks. Specifically, the actors perform tasks encoded in terms of multi-vehicle route-planning problems in a threatening environment. The threat is an unknown spatiotemporally-varying scalar field that is estimated using observations made by sensors. The major accomplishments and successes of this project are as follows: 1. Multiple research contributions for the optimal placement of sensors to collect observations of the most relevance to the actors' route-planning problem.					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			ERIK BLASCH
U	U	U	UU	34	19b. TELEPHONE NUMBER (Include area code) 426-7311

AFOSR Final Report

Grant Title: Multiscale Data-Driven Guidance and Control for Autonomous Vehicle Networks

Grant # FA9550-17-0028

Reporting Period: Jan. 1, 2017 – Dec. 31, 2020.

Author: Raghvendra V. Cowlagi (Principal Investigator)

Abstract

This project investigates coupled sensor configuration and planning (CSCP) for autonomous systems, which is a mode of active control of information in context to a decision-making problem. We consider a scenario consisting of a network of mobile vehicles called *sensors*, and a separate network of mobile vehicles called *actors*. Sensors gather information about the environment whereas actors perform desired tasks. Specifically, the actors perform tasks encoded in terms of multi-vehicle route-planning problems in a threatening environment. The threat is an unknown spatiotemporally-varying scalar field that is estimated using observations made by sensors. The major accomplishments and successes of this project are as follows:

1. Multiple research contributions for the optimal placement of sensors to collect observations *of the most relevance* to the actors' route-planning problem. To the best of our knowledge, this project is the first to establish an explicit relationship between the sensor placement problem and the actor's route-planning problem. These publications were published in top-ranked journals and presented at prestigious conferences.
2. Recognition of project personnel through multiple awards and professional milestones, including tenure and promotion for the principal investigator and attainment of doctoral degrees by two graduate research assistants.
3. Transition of techniques to AFRL; specifically, the application of the CSCP approach to the ESCAPE dataset collected in experiments conducted by the Sensor Data Exploitation Branch at the AFRL Information Directorate.

Research Contributions

1. Methods for simultaneous sensor placement and route-planning for a single vehicle with a finite parametric model of a spatially-varying temporally-static threat are developed [3,4,13]. Near-optimal plans with significantly fewer sensor observations are demonstrated in comparison to state-of-the-art methods. Convergence guarantees and suboptimality bounds are provided.
2. Methods for simultaneous sensor placement and route-planning for a single vehicle with a finite parametric model of a spatiotemporally-varying threat are developed [9,10]. Centralized and decentralized methods are developed and benefits over state-of-the-art methods are demonstrated.
3. Methods for simultaneous sensor placement and route-planning for a single vehicle with a nonparametric model of a spatially-varying temporally-static threat are developed [6,7]. Near-optimal plans with significantly fewer sensor observations are demonstrated in

comparison to state-of-the-art methods. Convergence guarantees and suboptimality bounds are provided. Approximation methods are developed to maximize a context-relevant information gain metric

4. Methods for single- and multi-vehicle decentralized route-planning to satisfy linear temporal logic specifications under kinematic constraints are developed [1,2,5,11,12].
5. Methods for simultaneous sensor placement and multi-vehicle route-planning for a single vehicle with a nonparametric model of a spatially-varying temporally-static threat are developed [1,8]. Bayesian optimization is employed to maximize a context-relevant information gain metric.

Sample scholarly publications are attached.

Transitions

1. Students to employment: Two former doctoral students completed their doctoral dissertations while funded from this grant. One student (Benjamin Cooper) is now employed at BAE Systems, Woburn, MA, whereas the other (Jie Fang) is employed at The MathWorks Inc.
2. Techniques to AFRL: through the 2021 AFRL Summer Faculty Fellowship Program, the PI is actively working to apply the computational methods developed under this grant to real-world data. Specifically, the PI's AFRL Lab Advisor is Dr. Peter Zulch, and the SFFP effort is aimed at analyzing the ESCAPE 2018 (Experiments, Scenarios, Concept of Operations, and Prototype Engineering) dataset.

Collaborations

The PI received a 2021 AFRL Summer Faculty Fellowship at AFRL/RI. The Lab Advisor is Dr. Peter Zulch. The SFFP includes support for one graduate student (Chase St. Laurent) who is currently working towards his doctoral dissertation under the PI's advice.

Awards

1. *2021 Graduate Student Paper Competition Finalist* at AIAA SciTech Guidance, Navigation, and Controls Conference to Dr. Jie Fang (former Ph.D. student)
2. *2019 WPI Sigma Xi Outstanding Doctoral Dissertation Award* to Dr. Benjamin Cooper (former Ph.D. student)
3. *2019 Romeo L. Moruzzi Young Faculty Award* for Innovation in Undergraduate Education to the PI
4. *2019 Morgan-Worcester Award for Distinguished Instructorship* to the PI
5. *2018 James Nichols Heald Research Fund* to the PI
6. *2017 Sigma Xi Outstanding Junior Faculty award* to the PI
7. *2017 WPI Graduate Research Innovation Exchange (GRIE) First place* to Dr. Benjamin Cooper (former Ph.D. student)

Publications

Journal Publications

1. J. Fang, Z. Zhang, and R. V. Cowlagi, “Decentralized route-planning for multi-vehicle teams to satisfy a subclass of linear temporal logic specifications,” *Automatica*, in review.
2. Z. Zhang, R. Du, and R. V. Cowlagi, “Randomized sampling-based trajectory optimization for UAVs to satisfy linear temporal logic specifications,” *Aerospace Science and Technology*, vol. 96, p. 105591, January 2020.
3. R. Du and R. V. Cowlagi, “Interactive sensing and planning for a quadrotor vehicle in partially known environments,” *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 7, pp. 1601–1611, July 2019.
4. B. S. Cooper and R. V. Cowlagi, “Interactive planning and sensing in unknown static environments with task-driven sensor placement,” *Automatica*, vol. 105, no. 7, pp. 391–398, July 2019.
5. B. S. Cooper and R. V. Cowlagi, “Path-planning with waiting in spatiotemporally-varying threat fields,” *PLoS ONE*, vol. 13, no. 8, p. e0202145, August 2018.

Peer-reviewed Conference Papers

6. C. L. St. Laurent and R. V. Cowlagi, “Depth-first coupled sensor configuration and path-planning in unknown static environments,” in *Proceedings of the 2021 European Control Conference*, July 2021.
7. C. L. St. Laurent and R. V. Cowlagi, “Coupled sensor configuration and path-planning in unknown static environments,” in *Proceedings of the 2021 American Control Conference*, New Orleans, LA USA 26–28 May 2021.
8. J. Fang, H. Zhang, and R. V. Cowlagi, “Interactive route-planning and mobile sensing with a team of robotic vehicles in an unknown environment,” in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2021 AIAA SciTech Forum & Exposition, 2021.
9. B. S. Cooper and R. V. Cowlagi, “Decentralized interactive planning and sensing in an unknown spatiotemporal threat field,” in *2019 Sixth Indian Control Conference (ICC)*, Hyderabad, India, 18–20 December 2019, pp. 110–115.
10. B. S. Cooper and R. V. Cowlagi, “Interactive planning and sensing in spatiotemporally varying uncertain environments,” in *2019 18th European Control Conference (ECC)*, Naples, Italy, 25–28 June 2019, pp. 2058–2064.
11. Z. Zhang and R. V. Cowlagi, “A fast sampling-based optimal route-planning algorithm to satisfy linear temporal logic specifications,” in *Proceedings of the Guidance, Navigation, and Control Conference*, 2018 AIAA SciTech Forum & Exposition, no. AIAA 2018-1590, Kissimmee, FL, USA, January 8–12 2018.
12. J. Fang, Z. Zhang, and R. V. Cowlagi, “Decentralized route-planning to satisfy global linear temporal logic specifications on multiple aircraft,” in *Proceedings of the Guidance, Navigation, and Control Conference*, 2018 AIAA SciTech Forum & Exposition, no. AIAA 2018-1862, Kissimmee, FL, USA, January 8–12 2018.

13. B. S. Cooper and R. V. Cowlagi, “Interactive planning and sensing in uncertain environments with task-driven sensor placement,” in *Proceedings of the 2018 American Control Conference*, Milwaukee, WI, USA, 27–29 June 2018, pp. 1003–1008.
14. R. Du and R. V. Cowlagi, “Interactive sensing and path-planning with incremental 3D path repair for a quadrotor UAV in cluttered and partially known environments,” in *Proceedings of the 56th IEEE Conference on Decision & Control*, Melbourne, VIC, Australia, 12–15 December 2017, pp. 933–938.

Invited Talks

1. *U.S. Army Small Unmanned Aircraft Systems Technology Innovation Event*, U.S. Army Combat Capabilities Development Center-Soldier Center (CCDC); Virtual Conference; October 2020.
2. *Department of Mechanical Engineering, Michigan State University*, East Lansing, MI, October 2019.
3. *Department of Mechanical Engineering, Worcester Polytechnic Institute*, October 2019.
4. *Department of Aeronautics, Imperial College of London*, London, UK, June 2018.
5. AFOSR PI Meetings (DDIP Program): annually 2017 – 2019.
6. Conference Presentations: at each of the venues of the conferences listed under “Peer-reviewed Conference Papers” above.

Community Efforts

In Summer 2018, the PI conducted a two-day workshop for high-school students to introduce basic concepts of guidance, navigation, and control. The workshop was part of a summer camp at WPI under the Mass Space Grant Consortium.

References

Extensions Granted: One year no-cost extension granted for calendar year 2020.

Interactive Planning and Sensing in Unknown Static Environments with Task-Driven Sensor Placement

Benjamin S. Cooper^a, Raghvendra V. Cowlagi^a

^a*Aerospace Engineering Program, Worcester Polytechnic Institute, Worcester, MA 01609, USA.*

Abstract

We address planar path-planning for a mobile vehicle to traverse a planar workspace with minimum exposure to a spatially varying scalar field called the threat field. The threat field is unknown, and is estimated by a finite number of sensors that take pointwise noisy measurements. We propose an iterative sensor placement and path-planning algorithm. At each iteration, regions of interest in the workspace are identified near the path with minimum estimated cost, and the next set of sensor locations is determined to improve the confidence of threat field estimates in these regions of interest. This paper breaks new ground in dynamic data-driven autonomy, in that an explicit bidirectional interaction between a path-planning algorithm and a sensor network separate from the actor vehicle is established. We provide theoretical results and corroborative numerical simulation results on the convergence and optimality of the proposed iterative algorithm. We show that the proposed algorithm provides significant improvements over existing approaches, including information-maximizing sensor placement methods.

1 Introduction

Suppose we wish to enable an autonomous vehicle to plan its path between given initial and destination points, but the environment is unknown. Sensors are available to take measurements and estimate an environment map. Where should these sensors be placed? The collection and processing of sensor measurements is usually separated from path-planning. We propose that the removal of this separation can be highly beneficial.

Related Work: The literature on motion-planning and control provides several standard techniques including applications of optimal control theory for vehicle guidance (Bryson and Ho, 1975), geometric path-planning algorithms based on workspace grids, cell decompositions, or visibility roadmaps (LaValle, 2006), and randomized sampling-based methods (LaValle and Kuffner, 2001; Karaman and Frazzoli, 2011). Motion-planning *under uncertainty* is typically formulated as a Markov Decision Process (MDP) or a partially observable MDP (POMDP), and can be solved using dynamic programming (Bertsekas, 2000). For computational tractability, belief space roadmaps are discussed (Prentice and Roy, 2009). Typical sources of uncertainty in motion-planning include modeling errors, measurement noise, and uncertainty in the environment map (Kurniawati et al., 2012), the latter of which is addressed in this work. Simulta-

neous localization and mapping (SLAM) addresses the reduction in uncertainty in the environment map and in the vehicle's state (Chakravorty and Saha, 2008; Lerner et al., 2007). SLAM methods rely on onboard sensors and do not address spatially distributed sensors.

Standard estimation techniques include the Kalman filter (KF), extended KF, unscented KF, and maximum likelihood estimators (Xie et al., 2007; Kay, 1993); regularization and iterative methods (Aster et al., 2011); and model and subset selection techniques (Park and Klabjan, 2017; Neumaier, 1998). The literature on *distributed* sensing and estimation addresses sensor placement and scheduling are addressed for estimating distributed processes (Krause et al., 2008; Ucinski, 2010; Das and Moura, 2017; Wu et al., 2018; Wang et al., 2018). Optimal sensor placement is typically characterized by maximum information or minimum entropy (Cochran and Hero, 2013). Maximization of the determinant, the trace, or the largest eigenvalue of the Fisher Information Matrix is another frequently used performance metric. Other metrics include least squared error, mutual information, Battacharya coefficients, Hellinger distance, and Kullback-Lieber divergence (Cochran and Hero, 2013; Krause et al., 2008; Mu et al., 2015).

Motion strategies for *mobile* sensor networks to maximize an information metric are reported (Demetriou et al., 2013; Martinez and Bullo, 2006; Martinez, 2010). The principal applications include target tracking and localization (Martinez and Bullo, 2006; Adurthi and Singla, 2014; Farmani et al., 2014; Xu et al., 2013). A

Email addresses: bscooper@wpi.edu (Benjamin S. Cooper), rvcowlagi@wpi.edu (Raghvendra V. Cowlagi).

distinction between *task-driven* and information-driven performance metrics is discussed by Kreucher et al. (2005), who conclude that a *task-specific* performance metric (e.g. squared tracking error) is beneficial.

The proposed approach is an iterative estimate-and-optimize method used in the *optimal design of experiments (DoE)* (Pronzato, 2008). Cooper (2018) provides comparisons of the proposed work against DoE methods.

Proposed Work and Contributions: We address planar path-planning for a mobile vehicle, which we call the *actor* vehicle, to traverse a planar workspace $\mathcal{W}$ with minimum exposure to a spatially varying scalar field, called the *threat field*. The threat field is unknown, and is estimated using a finite number of pointwise noisy measurements. We study the problem of sensor placement to optimize the actor’s performance.

This paper breaks new ground in *dynamic data-driven autonomy*, in that an explicit bidirectional interaction between a path-planning algorithm and a sensor network is considered. Prior works in the literature have focused either on the problem of finding paths with minimum estimated threat exposure (Meuleau et al., 2009) or on the problem of distributed estimation of a spatio-temporally varying scalar field (Demetriou et al., 2013; Madankan et al., 2014). This paper is the first to consider an “orchestration” of a sensor network to acquire measurements *most relevant to a concurrent path-planning problem*. We demonstrate the benefits of this approach over the traditional *information-driven* sensor placement. Building on a preliminary version of this work, (Cooper and Cowlagi, 2018), this paper proves optimality of the proposed approach and provides thorough comparisons to existing sensor placement approaches.

The rest of this paper is organized as follows. In Sec. 2, we introduce the problem elements. In Sec. 3 we describe the proposed interactive planning and sensing approach and its properties. In Sec. 4, we present numerical simulation results, and we conclude the paper in Sec. 5.

2 Problem Formulation

We denote by $\mathbb{R}$ and $\mathbb{N}$ the sets of real and natural numbers, respectively; by $[N]$ the set $\{1, \dots, N\}$ for $N \in \mathbb{N}$; and by $\mathbb{I}_{(N)}$ the identity matrix of size N . Let $\mathcal{W} \subset \mathbb{R}^2$ be a closed square region, called the *workspace*, in which the actor and the sensors move. In this workspace, we formulate a grid consisting of N_G uniformly placed points. The coordinates in a prespecified Cartesian coordinate axis system of the i^{th} grid point are denoted $\mathbf{x}_i$, for $i \in [N_G]$. We consider a strictly positive scalar field $c : \mathcal{W} \rightarrow \mathbb{R}_{>0}$, called the *threat field*, which represents unfavorable regions with higher intensity. The actor is assumed to traverse grid points according to a “4-connectivity rule.” We neglect vehicle kinematic / dynamic constraints, and note that such constraints can be easily incorporated in this setup (Cowlagi and Tsiotras, 2012).

We define a graph $\mathcal{G} = (V, E)$, where each vertex in V is uniquely associated with a grid point, and labeled by integers $1, 2, \dots, N_G$. The edge set E is the set of pairs of vertices associated with adjacent grid points. Edge transition costs are assigned as $g((i, j)) := c(\mathbf{x}_j)$, for $i, j \in [N_G]$, $(i, j) \in E$. A *path* in the graph $\mathcal{G}$ between vertices i_s and i_g is a sequence $\mathbf{v} = (v_0, v_1, \dots, v_M)$ of successively adjacent vertices with $v_0 = i_s$ and $v_M = i_g$. The *cost* of this path is defined as $J(\mathbf{v}) := \sum_{k=1}^M g((v_{k-1}, v_k))$. The actor’s motion-planning problem is a graph search problem on $\mathcal{G}$: namely, to find a path with minimum cost between initial and goal grid points $i_s, i_g \in [N_G]$. We refer to this path as the *true optimal* path, denoted $\mathbf{v}^*$.

The unknown threat field is parametrized as $c(\mathbf{x}) = 1 + \Phi^T(\mathbf{x})\Theta$, where $\phi_n : \mathbb{R}^2 \rightarrow \mathbb{R}_{\geq 0}$ are basis functions, $\Phi(\mathbf{x}) := [\phi_1(\mathbf{x}) \dots \phi_{N_P}(\mathbf{x})]^T$, and $\Theta := [\theta_1 \dots \theta_{N_P}]^T$ is a vector of unknown parameters. The basis functions are prespecified: specifically, we assume $\phi_n(\mathbf{x}) := \exp((\mathbf{x} - \bar{\mathbf{x}}_n)^T(\mathbf{x} - \bar{\mathbf{x}}_n)/2\nu_n)$, for each $n \in [N_P]$, where $\nu_n \in \mathbb{R}_{>0}$ and $\bar{\mathbf{x}}_n \in \mathcal{W}$ are prespecified constants. In a practical application, large values of N_P may be chosen to achieve a desired accuracy in the path cost estimates. To this end, we study the performance of the proposed algorithm for different values of N_P (see Sec. 4).

The region $\mathcal{R}_n^{\text{sup}} := \{\mathbf{x} : \|\mathbf{x} - \bar{\mathbf{x}}_n\| \leq 3\sqrt{\nu_n}\}$ is defined as the *region of significant support* for ϕ_n . Whereas the functions ϕ_n do not have compact support, 99.74% of the volume enclosed under each ϕ_n is enclosed by the restriction of ϕ_n to $\mathcal{R}_n^{\text{sup}}$. The constants ν_n and $\bar{\mathbf{x}}_n$ are chosen such that (A) $\cup_{n \in [N_P]} \mathcal{R}_n^{\text{sup}}$ covers the entire workspace, and (B) the regions $\mathcal{R}_n^{\text{sup}}, \mathcal{R}_p^{\text{sup}}$ are disjoint for $n \neq p$. When N_P is a perfect square, we choose $\bar{\mathbf{x}}_n$ on a uniform $\sqrt{N_P} \times \sqrt{N_P}$ grid in $\mathcal{W}$, and define $\nu_n := (\frac{\sqrt{2}}{6}(\bar{\mathbf{x}}_n - \bar{\mathbf{x}}_p))^2$ for adjacent basis functions with indices $n, p \in [N_P]$.

This choice of Gaussian basis functions is justified by the fact that square integrable functions can be approximated with arbitrary precision by linear combinations of Gaussian functions (Calcaterra, 2008). The proposed algorithm does not depend on this choice, and other basis functions such as orthogonal wavelets can be used.

A finite number of sensors $N_S \ll N_G$ are available to take pointwise measurements in the workspace. A number $N_L \leq N_S$ of these sensors are placed at distinct grid points. The set of these grid points, called a *placement*, is denoted by $\mathbf{s} = \{s_1, \dots, s_{N_L}\}$. The measurement taken by each sensor is $z_k := c(\mathbf{x}_{s_k}) + \eta_k$, where $\eta_k \sim \mathcal{N}(0, \sigma_k^2)$, for each $k \in [N_L]$.

We define a *region of estimability* for each basis ϕ_n , which is a region in which a sensor must be located to confidently estimate the parameter θ_n . To this end, let $\bar{\sigma} := \max\{\sigma_k\}_{k=1}^{N_S}$. For Gaussian η_k , if the signal-to-noise ratio (SNR) is greater than three, i.e., $\frac{\theta_n \phi_n}{\bar{\sigma}} \geq 3$,

the probability of detecting a signal is at least 0.9974 (Miller et al., 2005). For the basis ϕ_n , $\frac{\theta_n \phi_n}{\sigma} \geq 3$ occurs in the ball $\left\{ \mathbf{x} \in \mathcal{W} : \|\mathbf{x} - \bar{\mathbf{x}}_n\| \leq \sqrt{2\nu_n \log(\theta_n/3\sigma)} \right\}$. This ball can be under-approximated a priori using a rough order of magnitude estimate of θ_n , which is typically available. We assume that upper and lower bounds $\bar{\theta}_n$ and $\underline{\theta}_n$ are known, and that these bounds are within two orders of magnitude of each other. The estimates of parameters made from measurements taken within this ball are independent from estimates of other parameters. That is, if z_n, z_p are measurements taken by sensors placed in $\mathcal{R}_n^{\text{est}}$ and $\mathcal{R}_p^{\text{est}}$, respectively, with $n \neq p$ and $n, p \in [N_P]$, then $\text{Covar}(\hat{\theta}_n, \hat{\theta}_p | z_n, z_p) \approx 0$. To this end, we define the *region of estimability* for ϕ_n is the set $\mathcal{R}_n^{\text{est}} := \left\{ \mathbf{x} : \|\mathbf{x} - \bar{\mathbf{x}}_n\| \leq (1 - 1/\sqrt{2})\Delta\bar{\mathbf{x}} \right\}$, $\Delta\bar{\mathbf{x}} := \min\{\|\bar{\mathbf{x}}_n - \bar{\mathbf{x}}_p\| : n, p \in [N_P]\}$.

The actor can avail of estimates of the threat field parameters generated using sensor measurements over a sequence of placements $\{\mathbf{s}_\ell\}_{\ell \in \mathbb{N}}$. Therefore, the actor's path-planning problem is reformulated by considering edge transition costs based on the *estimated* threat field. For a given set $\mathbf{s}$ of sensor locations, let $\hat{\Theta}$ denote a parameter estimate. Then the estimated cost of a path $\mathbf{v} = (v_0, \dots, v_M)$ in $\mathcal{G}$ is $\hat{J}(\mathbf{v}) := M + \sum_{k=1}^M \Phi^T(\mathbf{x}_k)\hat{\Theta}$.

Problem 1 Find a sequence of sensor placements $\{\mathbf{s}_\ell\}_{\ell \in \mathbb{N}}$, and a path $\hat{\mathbf{v}}^*$ in $\mathcal{G}$ with minimum estimated cost.

A desirable property is that the minimum *estimated* cost be close to the minimum *true* cost, i.e., $|J(\hat{\mathbf{v}}^*) - J(\mathbf{v}^*)|$ be small, which is achieved by the proposed approach.

3 Interactive Planning and Sensing (IPAS)

For a given parameter estimate $\hat{\Theta}$, the actor's problem of finding an optimal path in $\mathcal{G}$ can be solved using Dijkstra's algorithm (Bertsekas, 2000). The novelty of the proposed work is that Problem 1 is solved by the following iterative method. At each iteration $\ell \in \mathbb{N}$, this method finds a parameter estimate $\hat{\Theta}_\ell$, the estimation error covariance matrix P_ℓ , a sensor placement $\mathbf{s}_\ell$, and a path $\mathbf{v}_\ell^*$ in $\mathcal{G}$ with minimum estimated cost, such that with a high probability

$$\lim_{\ell \rightarrow \infty} \hat{J}(\mathbf{v}_\ell^*) = \lim_{\ell \rightarrow \infty} J(\mathbf{v}_\ell^*) \leq J(\mathbf{v}^*) + \varepsilon, \quad (1)$$

where ε is a suboptimality bound precisely identified in Theorem 5. The process of finding the next sensor placement $\mathbf{s}_{\ell+1}$ based on the actor's current optimal path $\mathbf{v}_\ell^*$ is called *sensor reconfiguration*. The algorithm terminates when the variance of the estimated path cost $\text{Var}[\hat{J}(\mathbf{v}^*)]$ reaches a value smaller than a prespecified threshold.

This interactive planning and sensing (IPAS) algorithm is described in Lines 1–9 of Fig. 1 and the sensor reconfiguration policy is described in Lines 10–12.

The IPAS algorithm is initialized in Line 1 with an arbitrary sensor placement denoted $\mathbf{s}_0$. The algorithm maintains a set $\mathcal{I} \subset [N_P]$ of parameter indices, called *identified* parameters. At each iteration $\ell \in \mathbb{N}$, in Line 4, the algorithm determines the set $\mathcal{I}$ from the current sensor placement $\mathbf{s}_\ell = \{s_{\ell,1}, \dots, s_{\ell,N_L}\}$ recursively as

$$\mathcal{I} := \mathcal{I} \cup \{n : \exists k \in [N_L] \text{ with } \mathbf{x}_{s_{\ell,k}} \in \mathcal{R}_n^{\text{sup}}\}. \quad (2)$$

Informally, for each $n \in \mathcal{I}$, at least one sensor is placed within the region of significant support of ϕ_n in at least one iteration of the algorithm including the ℓ^{th} iteration. The complement of $\mathcal{I}$ is $\mathcal{I}^c := [N_P] \setminus \mathcal{I}$.

The algorithm also maintains a set of indices $\mathcal{L} = \{m_1, \dots, m_{N_L}\} \subset [N_P]$. The sensor reconfiguration policy determines $\mathcal{L}$ as discussed in Sec. 3.1. The size of this set $\mathcal{L}$ is $N_L \leq N_S$. In Line 5, the parameter estimate $\hat{\Theta}_\ell$, and the estimation error covariance P_ℓ are computed from the sensor measurements $\mathbf{z}_\ell \in \mathbb{R}^{N_L}$. To ensure well-posedness we define $T \in \mathbb{R}^{N_L \times N_P}$ element-wise as $T_{ij} := 1$ if $j = m_i$, and $T_{ij} := 0$ otherwise, for $i \in [N_L], j \in [N_P]$. Next, $H \in \mathbb{R}^{N_L \times N_L}$ is defined as $H_{ij} := \phi_{m_i}(\mathbf{x}_{s_{\ell,j}})$ for each $i, j \in [N_L]$. Recall that $\mathbf{x}_{s_{\ell,k}}$ denotes the grid point at which the k^{th} sensor is placed and is the closest available grid point to the center of m_k^{th} basis. The construction of the set $\mathcal{L}$ ensures each row of H has a non-zero entry. Due to the disjointness of regions of significant support, the matrix H has full rank N_L . The parameter estimates and estimation error covariance can then be computed using a standard recursive least squares method as follows:

$$P := TP_\ell T^T, \quad G_\ell := PH^T(HPH^T + R)^{-1}, \quad (3)$$

$$\hat{\Theta}_{\ell+1} := \hat{\Theta}_\ell + T^T G_\ell (\mathbf{z}_\ell - HT\hat{\Theta}_\ell), \quad (4)$$

$$P_{\ell+1} := (\mathbb{I}_{(N_P)} - T^T G_\ell HT)P_\ell. \quad (5)$$

This recursive method is initialized with $\hat{\Theta}_0 := \mathbf{0}$, and $P_0 := \lambda_0 \mathbb{I}_{(N_P)}$, where λ_0 is an arbitrary large constant. The threat field is then estimated as $\hat{c}(\mathbf{x}) := 1 + \Phi^T(\mathbf{x})\hat{\Theta}_\ell$. We refer to this threat field estimate as an *optimistic estimate* because for all indices $m \in \mathcal{I}^c$, the parameter $\hat{\theta}_m$ retains its initialized value of zero. Note that least-squares estimation problem in Line 5 of Fig. 1 is well-posed, and the estimates $\hat{\theta}_n$ are unbiased for each identified parameter, i.e., for each $n \in \mathcal{I}$.

At each iteration, N_L parameter estimates are updated by placing a sensor in each of $\mathcal{R}_\ell^{\text{est}}$, $\ell \in \mathcal{L}$, while the other parameters are treated as “nuisance variables” for that iteration (Routtenberg and Tong, 2015). Equations (3)–(5) provide the computational convenience of maintaining the sizes of $\hat{\Theta}_\ell$ and P_ℓ constant at each iteration.

If the variance of estimated path cost (Line 6) is lower than a threshold $\varepsilon_2(\mathbf{v}_\ell^*)$, Dijkstra's algorithm is executed in Line 7 to determine an optimal path $\mathbf{v}_\ell^*$ in $\mathcal{G}$, i.e., $\mathbf{v}_\ell^*$

Interactive Planning and Sensing

- 1: Set initial sensor placement $\mathbf{s}_0 \subset \{1, \dots, N_G\}$.
- 2: Set $\ell := 0$, $StopCondition := false$, and $\mathcal{I} := \emptyset$.
- 3: **while** $\neg StopCondition$ **do**
- 4: Find the set $\mathcal{I}$ by (2).
- 5: Update $\hat{\Theta}_\ell$, P_ℓ , by (3)–(5) and construct $\hat{c}(\mathbf{x})$.
- 6: **if** $Var[\hat{J}(\mathbf{v}_\ell^*)] \leq \varepsilon_2(\mathbf{v}_\ell^*)$ **then**
- 7: Execute Dijkstra's algorithm to find $\mathbf{v}_\ell^*$.
- 8: SENSOR RECONFIGURATION
- 9: Set $\ell := \ell + 1$, and evaluate $StopCondition$.

SENSOR RECONFIGURATION

- 10: Find the set $\mathcal{K}$ by Eqn. (6).
- 11: Find the set $\mathcal{L}$ as discussed in Sec. 3.1.
- 12: For each $m \in \mathcal{L}$, place a sensor in $\mathcal{R}_m^{\text{est}}$ to get $\mathbf{s}_{\ell+1}$.

Fig. 1. Pseudocode for the proposed IPAS algorithm .

has minimum expected cost with the threat field estimate at the ℓ^{th} iteration. The sensor reconfiguration policy is executed to obtain a new sensor placement $\mathbf{s}_{\ell+1}$, and the iteration repeats. Expressions for the path cost variance and for $\varepsilon_2(\mathbf{v}_\ell^*)$ are provided next.

3.1 Task-Driven Sensor Reconfiguration Policy

The sensor reconfiguration policy in the IPAS algorithm collects measurements that are of most relevance to the “task” (i.e., the path planning problem) at hand. To this end, this policy identifies (Line 10) a minimal set of indices $\mathcal{K} \subseteq [N_P]$ such that the current path lies within the union of regions of significant support defined by the corresponding basis functions, i.e.,

$$\mathcal{K} := \{n \in [N_P] : \mathcal{V}_\ell \cap \mathcal{R}_n^{\text{sup}} \neq \emptyset\}, \quad (6)$$

where $\mathcal{V}_\ell \subset \mathcal{W}$ is the set of grid-points of vertices in the path $\mathbf{v}_\ell^*$. We call the set $\mathcal{K}$ the *minimal cover* of $\mathbf{v}_\ell^*$.

The rationale of the proposed IPAS algorithm is that, typically, $\mathcal{K}$ has fewer than N_P elements. The parameters θ_n for $n \in \mathcal{K}$ are considered to be of the “most relevance” to the path-planning problem. Therefore, sensors can be placed to improve the confidence in the estimation of these parameters, possibly at the expense of reduced confidence in the other remaining parameter estimates.

Next, in Line 11, we compute a set of indices $\mathcal{L}$ defined as $\mathcal{L} := \mathcal{K}$ if $\mathcal{K} \cap \mathcal{I}^C = \emptyset$ or otherwise $\mathcal{L} := \mathcal{K} \cap \mathcal{I}^C$. Informally, $\mathcal{K} \cap \mathcal{I}^C$ is the set of indices of parameters that are of relevance to the path-planning problem but not yet identified. The sensor placement $\mathbf{s}_{\ell+1}$ in Line 12 consists of locations within the region of estimability of basis function ϕ_m , for each $m \in \mathcal{L}$. Ideally, sensors should be placed at grid points closest to the locations $\bar{\mathbf{x}}_m$ for each $m \in \mathcal{L}$, i.e., the maxima of the basis functions ϕ_m .

It is possible that $\mathcal{L}$ has more than N_S elements. In this case, $\mathcal{L}$ is sorted $\{m_1, m_2, \dots\}$ such that the distances of $\bar{\mathbf{x}}_{m_i}$ to their closest grid points are in non-decreasing order with increasing i . Then $\mathcal{L}$ is redefined as $\mathcal{L} := \{m_1, m_2, \dots, m_{N_S}\}$. This ordering of $\mathcal{L}$ is equivalent to ordering by SNR for sensor measurements.

3.2 Algorithm Termination Criterion

The termination criterion for this algorithm, denoted by the boolean variable *StopCondition*, is based on the variance of the estimated path cost variance, which is in turn computed from the parameter estimation error covariance. Because the path cost depends linearly on the parameters, it is a Gaussian r.v. Because parameter estimates are unbiased, the path cost estimate is also unbiased whenever all parameters with indices in the set $\mathcal{K}$ are identified. Note that the sensor reconfiguration policy *ensures* that parameters with indices in $\mathcal{K}$ become identified as the algorithm iterates.

At the ℓ^{th} iteration of the IPAS algorithm, the variance of the estimated cost of path $\mathbf{v}_\ell^* = (v_0, \dots, v_M)$ is $Var[\hat{J}(\mathbf{v}_\ell^*)] = \sum_{i=0}^M \Phi^T(\mathbf{x}_{v_i}) P_\ell \Phi(\mathbf{x}_{v_i})$. The boolean *StopCondition* is *true* if $Var[\hat{J}(\mathbf{v}_\ell^*)] \leq \varepsilon_1(\mathbf{v}_\ell^*)$, or is *false* otherwise. Here, $\varepsilon_1(\mathbf{v}_\ell^*)$ is a path-dependent threshold chosen based on the grid resolution. Consider the hypothetical ideal case where a sensor can be placed at every grid point, and the measurement error variance of each sensor is $\bar{\sigma}$. Then the estimation error covariance is $P_{\text{grid}} := (H_{\text{grid}}^T R^{-1} H_{\text{grid}})^{-1}$, where $H_{\text{grid}} := [\Phi(\mathbf{x}_1) \ \Phi(\mathbf{x}_2) \ \dots \ \Phi(\mathbf{x}_{N_G})]^T$ and $R := \bar{\sigma} \mathbb{I}_{(N_G)}$. Then we define

$$\varepsilon_1(\mathbf{v}_\ell^*) := \lambda_1^2 \sum_{i=0}^M \Phi^T(\mathbf{x}_{v_i}) P_{\text{grid}} \Phi(\mathbf{x}_{v_i}), \quad (7)$$

$$\varepsilon_2(\mathbf{v}_\ell^*) := \lambda_2^2 \sum_{i=0}^M \Phi^T(\mathbf{x}_{v_i}) P_{\text{grid}} \Phi(\mathbf{x}_{v_i}), \quad (8)$$

where $\lambda_2 > \lambda_1 \geq 1$ are prespecified constants. Note that P_{grid} can be computed before taking any measurements.

The computation of $Var[\hat{J}(\mathbf{v}_\ell^*)]$ is well-defined because the least-squares estimation problem in Line 5 is well-posed. Due to the unbiasedness of parameter estimates, the proposed algorithm can also reduce the variance and mean square error, *both*, of the estimated path cost to arbitrarily small values.

3.3 Convergence and Optimality

In this section, we provide a series of technical results to prove that the main result of this paper: the proposed IPAS algorithm terminates and upon termination the algorithm provides a near-optimal path.

Proposition 2 *The IPAS algorithm terminates in a finite number of iterations whenever the number of parameters N_P is finite, for any number of sensors $N_S > 0$.*

Proof. Omitted for brevity, see (Cooper, 2018).

In what follows, we denote by $\bar{\ell}$ the number of iterations required for the IPAS algorithm to terminate, and by $\mathcal{I}$ the set of identified indices after $\bar{\ell}$ iterations. The next result follows immediately due to Prop. 2, due to the definition of *StopCondition*, and due to the fact that the path cost estimate is a Gaussian r.v.

Corollary 3 $\mathbb{P} \left[J(\mathbf{v}_{\bar{\ell}}^*) - \hat{J}(\mathbf{v}_{\bar{\ell}}^*) \leq 3\sqrt{\varepsilon_1(\mathbf{v}_{\bar{\ell}}^*)} \right] \geq 0.9974$.

Lemma 4 Let $\mathbf{v}_{\mathcal{I}}^*$ be a path of minimum true cost within the region $\cup_{k \in \mathcal{I}} \mathcal{R}_k^{\text{sup}}$. Then

$$\mathbb{P} \left[\hat{J}(\mathbf{v}_{\bar{\ell}}^*) - J(\mathbf{v}_{\mathcal{I}}^*) \leq 3\sqrt{\varepsilon_2(\mathbf{v}_{\bar{\ell}}^*)} \right] \geq 0.9974. \quad (9)$$

Proof. Omitted for brevity, see (Cooper, 2018).

Next, we state the main result of the paper as follows.

Theorem 5 The path $\mathbf{v}_{\bar{\ell}}^*$ satisfies either

$$\mathbb{P} \left[J(\mathbf{v}_{\bar{\ell}}^*) - J(\mathbf{v}^*) \leq 3(\sqrt{\varepsilon_1(\mathbf{v}_{\bar{\ell}}^*)} + \sqrt{\varepsilon_2(\mathbf{v}^*)}) \right] \geq 0.9948,$$

$$\text{or } \mathbb{P} \left[J(\mathbf{v}_{\bar{\ell}}^*) - J(\mathbf{v}^*) \sim O(\sqrt{\frac{N_G}{10}}) \right] \geq 0.9974.$$

Proof. Because N_P is finite, either all parameters are identified, i.e., $\mathcal{I} = [N_P]$, or a proper subset of parameters is identified, i.e., $\mathcal{I} \subset [N_P]$. In either case, there is a finite $L \in \mathbb{N}$ such that $\mathcal{I}$ becomes invariant after the L^{th} iteration. Therefore, for each $\ell > L$, $\mathcal{K}_{\ell} \cap \mathcal{I} = \emptyset$, and $\mathcal{L} = \mathcal{K}$ becomes invariant: otherwise a sensor is placed in the region of estimability of at least one parameter with index in $\mathcal{I}^C$, by consequence this parameter is identified by (2), and $\mathcal{I}$ changes in the next iteration.

First suppose either $\mathcal{I} = [N_P]$, and/or $\mathbf{v}^*$ is covered by $\cup_{k \in \mathcal{I}} \mathcal{R}_k^{\text{sup}}$. By Lemma 4, $\mathbb{P}[\hat{J}(\mathbf{v}_{\bar{\ell}}^*) - J(\mathbf{v}^*) \leq 3\sqrt{\varepsilon_2(\mathbf{v}^*)}] \geq 0.9974$. By Corl. 3,

$$\mathbb{P} \left[\hat{J}(\mathbf{v}_{\bar{\ell}}^*) - J(\mathbf{v}^*) \leq 3\sqrt{\varepsilon_2(\mathbf{v}^*)} \right] \text{ and } \mathbb{P} \left[J(\mathbf{v}_{\bar{\ell}}^*) - \hat{J}(\mathbf{v}_{\bar{\ell}}^*) \leq 3\sqrt{\varepsilon_1(\mathbf{v}_{\bar{\ell}}^*)} \right] \geq 0.9974^2 = 0.9948,$$

and it follows that $\mathbb{P}[J(\mathbf{v}_{\bar{\ell}}^*) - J(\mathbf{v}^*) \leq 3(\sqrt{\varepsilon_1(\mathbf{v}_{\bar{\ell}}^*)} + \sqrt{\varepsilon_2(\mathbf{v}^*)})] \geq 0.9948$.

Now suppose $\mathcal{I} \neq [N_P]$, and $\mathbf{v}^*$ does not entirely lie within $\cup_{k \in \mathcal{I}} \mathcal{R}_k^{\text{sup}}$. Recall that the threat field estimate is optimistic, i.e., for grid points $\mathbf{x}_i$ such that $\mathbf{x}_i \notin \cup_{k \in \mathcal{I}} \mathcal{R}_k^{\text{sup}}$, $\hat{c}(\mathbf{x}_i) \leq 1 + 0.011 \sum_{k \in \mathcal{I}} \theta_k$ because the maximum value of ϕ_k outside $\mathcal{R}_k^{\text{sup}}$ is 0.011.

The true value of the threat field at this grid point is $c(\mathbf{x}_i) \geq 1 + 0.011 \sum_{k \in \mathcal{I}} \theta_k$. Therefore, the regret associated with not including i is at most $\hat{c}(\mathbf{x}_i) - c(\mathbf{x}_i) \leq 0.011 \sum_{k \in \mathcal{I}} (\theta_k - \underline{\theta}_k)$. The number of such vertices is at most $O(\sqrt{N_G})$. It follows that $\hat{J}(\mathbf{v}_{\bar{\ell}}^*) - J(\mathbf{v}^*) \sim 0.022 \sum_{k \in \mathcal{I}} (\theta_k - \underline{\theta}_k) O(\sqrt{N_G}) \sim O(10^{-2} \sqrt{N_G})$ and by consequence, $\mathbb{P}[J(\mathbf{v}_{\bar{\ell}}^*) - J(\mathbf{v}^*) \sim O(10^{-2} \sqrt{N_G})] \geq 0.9974$. $\square$

Remark 6 The preceding proof relies on the unbiasedness of estimates of identified parameters. Otherwise, the estimate of the regret of not including a grid point i such that $\mathbf{x}_i \notin \cup_{k \in \mathcal{I}} \mathcal{R}_k^{\text{sup}}$, will be biased. Equations (3)–(5) guarantee such unbiasedness.

3.4 Other Characteristics of the IPAS Algorithm

Comparison with traditional sensor placement: The traditional approach to the solution of Problem 1 is to first deploy sensors to optimize some metric of the quality of the threat field estimate, and to then plan a path using the estimated field. We refer to such traditional approaches as *information-driven*.

The solution of Problem 1 improves as the variance of the estimated path cost decreases, i.e., for a path $\hat{\mathbf{v}}^*$ with minimum estimated cost, if $\text{Var}[\hat{J}(\hat{\mathbf{v}}^*)]$ is small, then $|\hat{J}(\hat{\mathbf{v}}^*) - J(\hat{\mathbf{v}}^*)|$ is small with a high probability. It follows that sensor placement for Problem 1 must attempt to minimize $\text{tr}[P_{\bar{\ell}}|_{\mathcal{K}}]$. In any information-driven method $\text{tr}[P_{\bar{\ell}}|_{\mathcal{K}}]$ is at least as large as that obtained from the IPAS algorithm because an information-driven method is not concerned with $\mathcal{K}$ at all, whereas the IPAS algorithm reduces $\text{tr}[P_{\bar{\ell}}|_{\mathcal{K}}]$. We assume that the total number of measurements made is the same in either case. Therefore we note the following result.

Proposition 7 Let $\hat{\mathbf{v}}_{\text{IPAS}}^*$ be the solution to Problem 1 returned by the IPAS algorithm, and let $\hat{\mathbf{v}}_{\text{ID}}^*$ be the solution to Problem 1 returned by an information-driven approach. Then $J(\hat{\mathbf{v}}_{\text{IPAS}}^*) \leq J(\hat{\mathbf{v}}_{\text{ID}}^*)$.

Computational Complexity: The computational complexity of the IPAS algorithm depends on the number of calls to Dijkstra's algorithm in Line 7 in Fig. 1, the worst-case complexity of which is $O(N_G + N_G \log(N_G))$ (Bertsekas, 2000). The total number of iterations of the algorithm scales linearly with the number of parameters N_P . Therefore, the worst-case computational complexity of the IPAS algorithm is $O(N_P(N_G + N_G \log(N_G)))$.

Sensor reconfiguration cost: We acknowledge the need to include in future work the cost of physically relocating a team of mobile sensors to execute the iterative sensor placement provided by the IPAS algorithm. This cost may capture the time and/or fuel spent in physically moving the sensors, while satisfying the kinematical and dynamical constraints of the mobile sensors. To address

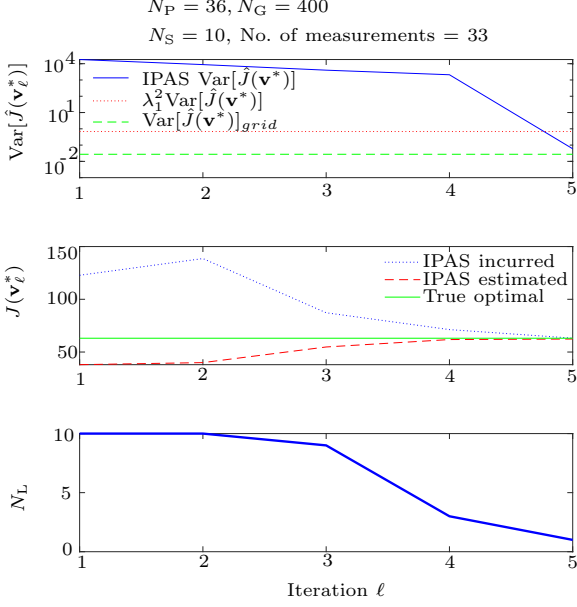


Fig. 2. Convergence behavior of the IPAS algorithm.

this cost, results from the multi-vehicle path planning literature may be leveraged, cf. (Bullo et al., 2009).

4 Results and Discussion

The numerical simulations reported in this section are all performed with a square workspace $\mathcal{W} = [-1, 1] \times [-1, 1]$ in nondimensional units. The threat field is constructed as in Sec. 2, with constants $\bar{x}_n$ chosen to provide uniform separation between the basis function peaks. The ground truth values of the parameters θ_n are randomly generated by sampling the uniform distribution $\mathcal{U}(0, 10)$. Therefore, $\theta_n = 0$ and $\theta_n = 10$ for each $n \in [N_P]$. MATLAB[®] code for reproducing these simulations is available: <https://github.com/bscooperplanning/static-ipas> See (Cooper, 2018) for further detailed numerical studies.

4.1 Illustrative Example

Figures 2 and 3 provide results of the implementation of the algorithm on an illustrative example. In this example, the algorithm termination criterion is set with the constant $\lambda_1 = 5$ in (7). The numbers of grid points, parameters, and sensors are $N_G = 400$, $N_P = 36$, and $N_S = 10$, respectively. The IPAS algorithm minimizes the estimated path cost. To study the algorithm's performance, we compare this estimated cost against the true cost of the true optimal path, and the *incurred cost*, which is true cost of the path found by the algorithm.

First, note in Fig. 2 (top) that the IPAS algorithm terminates in 5 iterations. For each of the iterations $\ell \in [2, 5]$, Fig. 3 shows the threat estimate $\hat{c}$ by a colormap, the path $\mathbf{v}_\ell^*$ of minimum estimated cost by white circles, and

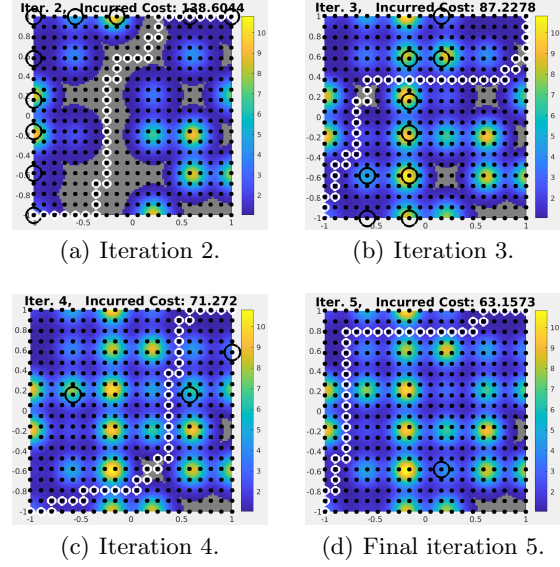


Fig. 3. Visualization of the iterative and interactive planning and sensing (IPAS) process for $N_P = 36$ and $N_S = 10$.

the sensor placement $\mathbf{s}_\ell$ by black circles. Whereas the termination criterion requires that the path cost variance be at most $\varepsilon_1(\mathbf{v}_\ell^*) = 0.683$, in this case, at termination the actual path cost variance is much lower, namely $\text{Var}[\hat{J}(\mathbf{v}_5^*)] = 0.059$. In Fig. 3, the gray regions are of significant support of bases with non-identified parameters, i.e., those ϕ_n with $n \in \mathcal{I}^C$. Because the IPAS algorithm is “optimistic” (as discussed in Sec. 3), the path $\mathbf{v}_2^*$ traverses through the gray regions.

Second, note in Fig. 2 (middle) that upon termination, the incurred and estimated costs are nearly identical: namely $J(\mathbf{v}_5^*) = 63.16$ and $\hat{J}(\mathbf{v}_5^*) = 62.33$. This observation agrees with the low value of $\text{Var}[\hat{J}(\mathbf{v}_5^*)]$. More importantly, the optimal *estimated* cost is close to the *true* optimal cost, which agrees with the theoretical result of Theorem 5. Before termination, the estimated cost is lower than the true optimal cost because of the optimistic threat estimates.

Third, note in Fig. 2 (bottom) that the number of sensors placed in each iteration is not always N_S . The sum of the numbers of sensors placed at each iteration, over all of the five iterations, is 33. Crucially, note that the total number of measurements taken by the IPAS algorithm is less than the number of parameters, and yet the path found by the algorithm is nearly identical in cost to the *true* optimal path. This example therefore illustrates the most powerful feature of the IPAS algorithm: not only can it find a near-optimal path with $N_S \ll N_P$, but it can find a near-optimal path with fewer *measurements* than parameters, i.e., even with $\sum_{\ell=1}^{\bar{\ell}} N_L(\ell) < N_P$, where $N_L(\ell)$ is the number of sensors placed at each iteration, and $\bar{\ell}$ is the total number of iterations.

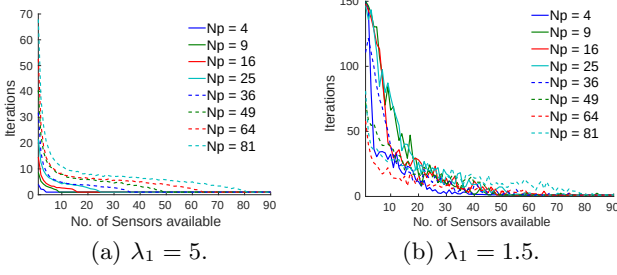


Fig. 4. Number of iterations until convergence.

Table 1
Average # measurements for IPAS convergence, and for FP to match path suboptimality of IPAS for $N_S = 1$.

N_P	4	9	16	25	36	49	64	81
FP	4.24	9.37	16.29	25.39	36.10	49.19	63.72	80.88
IPAS	4.00	8.99	15.03	21.97	31.55	40.80	51.41	67.84

4.2 Convergence and Optimality

To demonstrate the performance of the IPAS algorithm for increasing number of parameters N_P , we performed numerical simulations for every combination of $N_P \in \{4, 9, 16, 25, 36, 49, 64, 81\}$ with $N_S \in \{1, 2, \dots, 90\}$, and with fixed $N_G = 400$ and $\lambda_1 = 5$. The results gathered for every combination (N_S, N_P) were averaged over 100 simulations, for different threat fields with randomly generated parameter values.

Figure 4(a) shows results on the number of iterations $\bar{\ell}$ required to terminate. As intuitively expected, the number of iterations decreases with increasing number of sensors, irrespective of the number of parameters. Importantly, the algorithm converges in a small number of iterations even in cases where $N_S \ll N_P$. For example, note that for $N_P = 81$, the algorithm converges in less than 10 iterations with as few as 12 sensors.

A striking result is that the incurred costs of paths found by the IPAS algorithm are close to the *true* optimal cost. Table 1 shows the average number of measurements required by the IPAS algorithm for the actor's incurred cost to lie within 0.1% of the true optimal cost.

As discussed in Sec. 3.2, smaller values of the constant λ_1 in (7) imply more stringent termination criteria. In Figure 4(b), the convergence results from a set of simulations run with $\lambda_1 = 1.5$ show an increase in the iterations required, however the exponential behavior between iterations and N_S remains. Note that for higher N_S the number of required iterations is similar for $\lambda_1 = 5$ and $\lambda_1 = 1.5$.

4.3 Comparisons with Information-Driven Approaches

We compare the IPAS algorithm to an information-driven approach that attempts to maximize the so-called

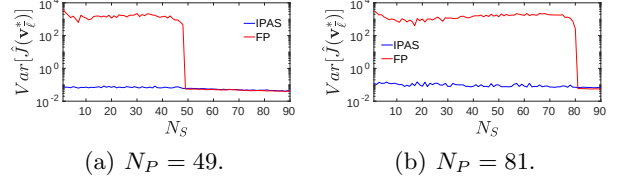


Fig. 5. Path cost variance comparison.

frame potential (FP), which is a scalar metric that measures the orthogonality of the rows of the measurement matrix. A near-optimal greedy placement algorithm and its software implementation are readily available in the literature (Ranieri et al., 2014). We also considered comparisons with mutual information maximization (Krause et al., 2008). However, we found that mutual information maximization provided actor path costs nearly identical to those achieved by FP maximization (FPM), but required up to two orders of magnitude more computation time. Numerical simulations were performed for IPAS and FPM for the same set of combinations of N_P and N_S as discussed in Sec. 4.1. For each (N_P, N_S) combination, we performed 100 simulations with different randomly generated threat fields.

To ensure that these comparisons were fair, FPM was allowed the same number of *measurements* as IPAS, i.e., the quantity $\sum_{\ell=1}^{\bar{\ell}} N_L(\ell)$, which is greater than N_S .

Table 1 compares the average number of measurements required by IPAS to converge, against the number of measurements that FPM required to achieve the same path cost suboptimality as the IPAS approach. The IPAS algorithm outperforms FPM in all cases. Importantly, note that FPM always requires nearly equal measurements to N_P , whereas IPAS requires fewer than N_P measurements in all but one case. Finally, note that as N_P increases, the difference $N_P - \sum_{\ell=1}^{\bar{\ell}} N_L(\ell)$ increases. In other words, the number of measurements required using IPAS is sublinear with respect to the environment parameter N_P .

Figure 5 shows the variance of estimated costs of the paths resulting from FPM compared to that resulting from the IPAS algorithm. In each of the cases, the IPAS algorithm achieves a path cost variance at least as low, and often lower than that achieved by FPM.

To summarize, not only does the IPAS algorithm consistently find paths whose true costs are near-optimal, but it also ensures that the confidence in the estimated path cost is high (i.e., path cost variance is low).

4.4 Performance with Extremely Small N_S

We present here an example with $N_P = 100$ and $N_S = 10$, which is an *order of magnitude* smaller than N_P . Figure 6 shows the path resulting from the IPAS algorithm on this problem. Note the significant regions of the map

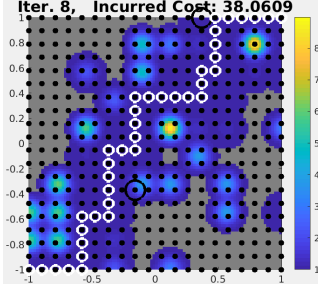


Fig. 6. Example with $N_P = 100$ and $N_S = 10$, the required number of measurements for convergence is only 55.

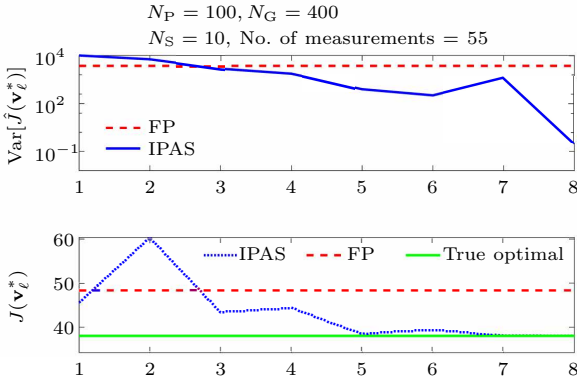


Fig. 7. Comparison of IPAS approach and an information-driven approach (frame potential).

which remain gray, corresponding to many parameters that were not identified. The estimated, incurred, and true path costs, the variance of the estimated path cost, and the numbers of sensor placements at each iteration are shown in Fig. 7. Note that the path resulting from the IPAS algorithm is in fact a true optimal path.

By comparison, an information-driven approach like FPM places sensors in the gray regions, whereas those sensor measurements will provide no benefit for this particular path planning problem. In this example, the total number of measurements taken by the IPAS algorithm over all iterations was 55, which is roughly half of $N_P = 100$. In contrast, when FPM with 55 measurements was implemented, the resultant path with minimum estimated cost was significantly suboptimal compared to the true optimal path as shown in Fig. 7 (top). Furthermore, the variance of the estimated path cost resulting from FPM remained four orders of magnitude higher than that resulting from the IPAS algorithm.

5 Conclusions

The proposed IPAS algorithm provides sensor placement and a path with minimum estimated cost. The termination criterion of this algorithm ensures that the estimated cost has low variance, and is therefore close to the true cost. We provided a probabilistic bound on the suboptimality of the resultant path. Numerical

simulation results corroborated this theoretical result, i.e., the IPAS algorithm resulted in near-optimal paths. Compared to a state-of-the-art information-driven sensor placement method, the IPAS algorithm consistently resulted in paths with lower costs, and required fewer measurements to do so.

Acknowledgement: This research is funded by US AFOSR grant #FA9550-17-1-0028.

References

- N. Adurthi and P. Singla. Information Driven Optimal Sensor Control for Efficient Target Localization and Tracking. In *Proc. 2014 American Control Conference*, pages 610–615, Portland, OR, USA, Jun 2014.
- R. C. Aster, B. Borchers, and C. H. Thurber. *Parameter estimation and inverse problems*, volume 90. Academic Press, 2011.
- D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 2000.
- A. E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Taylor & Francis, New York, NY, USA, 1975.
- F. Bullo, J. Cortés, and S. Martinez. *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Princeton University Press, 2009.
- C. Calcaterra. Linear combinations of gaussians with a single variance are dense in l_2 . In *Proceedings of the World Congress on Engineering*, volume 2, 2008.
- S. Chakravorty and R. Saha. Simultaneous planning localization and mapping: A hybrid Bayesian/frequentist approach. *Proceedings of the American Control Conference*, pages 1226–1231, 2008. ISSN 07431619. doi: 10.1109/ACC.2008.4586660.
- D. Cochran and A. O. Hero. Information-driven sensor planning: Navigating a statistical manifold. *2013 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2013 - Proceedings*, (0):1049–1052, 2013. doi: 10.1109/GlobalSIP.2013.6737074.
- B. S. Cooper. *Interactive Planning and Sensing for Aircraft in Uncertain Environments with Spatiotemporally Evolving Threats*. PhD thesis, Worcester Polytechnic Institute, 2019. Preprint available at <http://labs.wpi.edu/ace-lab/publications/>.
- B. S. Cooper and R. V. Cowlagi. Interactive planning and sensing in uncertain environments with task-driven sensor placement. In *Proceedings of the 2018 American Control Conference*, Milwaukee, WI, USA., June 2018.
- R. V. Cowlagi and P. Tsiotras. Hierarchical motion planning with dynamical feasibility guarantees for mobile robotic vehicles. *IEEE Transactions on Robotics*, 28(2):379 – 395, 2012.
- S. Das and J. M. F. Moura. Consensus+innovations distributed kalman filter with optimized gains. *IEEE Transactions on Signal Processing*, 65(2):467–481, Jan 2017. ISSN 1053-587X. doi: 10.1109/TSP.2016.2617827.
- M. Demetriou, N. Gatsonis, and J. Court. Coupled controls-computational fluids approach for the estimation of the concentration from a moving gaseous source in a 2-d domain with a Lyapunov-guided sensing aerial vehicle. *IEEE Transactions on Control Systems Technology*, 22(3):853–867, 2013. doi: 10.1109/TCST.2013.2267623.

- N. Farmani, L. Sun, and D. Pack. Optimal Uav Sensor Management and Path Planning for Tracking. *The ASME 2014 Dynamic System and Control Conferences*, pages 1–8, 2014. doi: 10.1115/DSCC2014-6232.
- S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30:846–894, 2011.
- S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. ISBN 0-13-345711-7.
- 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*, 9:235–284, 2008. ISSN 15324435. doi: 10.1145/1102351.1102385.
- C. Kreucher, A. O. Hero, and K. Kastella. A comparison of task driven and information driven sensor management for target tracking. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, CDC-ECC '05*, 2005:4004–4009, 2005. doi: 10.1109/CDC.2005.1582788.
- H. Kurniawati, T. Bandyopadhyay, and N. M. Patrikalakis. Global motion planning under uncertain motion, sensing, and environment map. *Autonomous Robots*, 33(3):255–272, 2012. ISSN 09295593. doi: 10.1007/s10514-012-9307-y.
- S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
- S. M. LaValle and J. J. Kuffner, Jr. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.
- R. Lerner, E. Rivlin, and I. Shimshoni. Landmark selection for task-oriented navigation. *IEEE Transactions on Robotics*, 23(3):494–505, 2007. ISSN 15523098. doi: 10.1109/TRO.2007.895070.
- R. Madankan, S. Pouget, P. Singla, M. Bursik, J. Dehn, M. Jones, A. Patra, M. Pavolonis, E. B. Pitman, T. Singh, and P. Webley. Computation of probabilistic hazard maps and source parameter estimation for volcanic ash transport and dispersion. *Journal of Computational Physics*, 271:39–59, 2014.
- S. Martinez. Distributed interpolation schemes for field estimation by mobile sensor networks. *IEEE Trans. Control Systems Technology*, 18(2):491–500, 2010.
- S. Martinez and F. Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(4):661–668, 2006.
- N. Meuleau, E. Benazera, R. I. Brafman, E. A. Hansen, et al. A heuristic search approach to planning with continuous resources in stochastic domains. *Journal of Artificial Intelligence Research*, 34:27–59, 2009.
- J. N. Miller, J. C. Miller, et al. Statistics and chemometrics for analytical chemistry. 2005.
- B. Mu, L. Paull, M. Graham, J. How, and J. Leonard. Two-Stage Focused Inference for Resource-Constrained Collision-Free Navigation. *Robotics: Science and Systems*, 2015.
- A. Neumaier. A Comparison of Subset Selection and Adaptive Basis Function Construction for Polynomial Regression Model Building. *SIAM Review*, 40(3):636–666, 1998.
- Y. W. Park and D. Klabjan. Subset Selection for Multiple Linear Regression via Optimization. (i), 2017. URL <http://arxiv.org/abs/1701.07920>.
- R. Prentice and N. Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *International Journal of Robotics Research*, 28(11-12):1448–1465, 2009.
- L. Pronzato. Optimal experimental design and some related control problems. *Automatica*, 44(2):303–325, 2008. ISSN 00051098. doi: 10.1016/j.automatica.2007.05.016.
- J. Ranieri, A. Chebira, and M. Vetterli. Near-Optimal Sensor Placement for Linear. 62(5):1135–1146, 2014.
- T. Routtenberg and L. Tong. Estimation after Parameter Selection: Performance Analysis and Estimation Methods. *IEEE Trans. Signal Processing*, 64(20):1–13, 2015. ISSN 1053587X. doi: 10.1109/TSP.2016.2580533.
- D. Ucinski. Sensor network scheduling for identification of spatially distributed processes. *Conference on Control and Fault-Tolerant Systems, SysTol'10 - Final Program and Book of Abstracts*, 20(3):493–504, 2010. ISSN 1641-876X.
- L. Wang, J. Markdahl, Z. Liu, and X. Hu. Decentralized cooperative tracking subject to motion constraints. *Automatica*, 96:121 – 126, 2018. ISSN 0005-1098.
- Z. Wu, M. Fu, Y. Xu, and R. Lu. A distributed kalman filtering algorithm with fast finite-time convergence for sensor networks. *Automatica*, 95:63 – 72, 2018. ISSN 0005-1098.
- L. Xie, D. Popa, and F. L. Lewis. *Optimal and robust estimation: with an introduction to stochastic control theory*. CRC press, 2007.
- Y. Xu, J. Choi, S. Dass, and T. Maiti. Efficient bayesian spatial prediction with mobile sensor networks using gaussian markov random fields. *Automatica*, 49(12):3520 – 3530, 2013. ISSN 0005-1098.

Decentralized Route-Planning for Multi-Vehicle Teams to Satisfy A Subclass of Linear Temporal Logic Specifications

Jie Fang^a Zetian Zhang^b Raghvendra V. Cowlagi^c

^a*Mathworks, Inc., Natick, MA, USA.*

^b*Amazon.com, Inc., Seattle, WA USA.*

^c*Aerospace Engineering Dept., Worcester Polytechnic Institute, Worcester, MA, USA.*

Abstract

Linear temporal logic (LTL) formulae are widely used to provide high level behavioral specifications on mobile robots. We propose a decentralized route-planning method for a networked team of mobile robots to satisfy a common (global) LTL specification. The global specification is assumed to be a conjunction of multiple formulae, each of which is treated as a task to be assigned to one or more vehicles. The proposed method leverages the consensus-based bundle algorithm for decentralized task assignment, thereby decomposing the global specification into local specifications for each vehicle. We develop a new algorithm to assign *collaborative* tasks: those simultaneously assigned to multiple vehicles. Rewards in the proposed task assignment algorithm are computed using the so-called lifted graph, which ensures the satisfaction of minimum turn radius constraints on vehicular motion. This algorithm synchronizes the vehicles' routes with minimum waiting durations. The proposed method is compared against a multi-vehicle task assignment algorithm from the literature, which we extend to apply for satisfying LTL specifications. The proposed route-planning method is demonstrated via numerical simulation examples and a hardware implementation on networked single-board computers.

We address decentralized route-planning for a team of networked mobile robots to complete collaborative tasks given by linear temporal logic (LTL) specifications. Vehicle kinematic constraints are also considered.

In addition to the usual logical operators, the LTL formalism includes temporal operators such as **always**, and **eventually**. For mobile vehicles, high-level intelligent missions and safety properties, e.g. “*perform 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. Compared to the traditional vehicle route-planning problem (VRP) (Toth and Vigo, 2002), more complex robotic missions, e.g., those involving persistent surveillance and/or conditional behavior can be specified by LTL (Ulusoy et al., 2013).

The general approach to route-planning for satisfying LTL specifications is based on formal control synthesis techniques for dynamical systems (Belta et al., 2017). A Büchi automaton, whose accepting runs are exactly

the strings that satisfy the given LTL specification, is constructed (Duret-Lutz, 2013). 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 executed on this product automaton for control synthesis. Applications of this general approach to route-planning of mobile vehicles are well-studied (Kloetzer and Belta, 2007; Cowlagi and Zhang, 2017). However, the literature on coordinated route-planning for a team of multiple mobile robots to satisfy LTL specifications on the team’s behavior – henceforth called *global* specifications – has gaps that this paper seeks to bridge.

On the one hand, several *centralized* algorithms are reported to decompose the global specifications into local specifications – i.e., *tasks* – for each vehicle (Chen et al., 2011; Karimadini and Lin, 2010; Ulusoy et al., 2013; Kloetzer and Mahulea, 2016; Schillinger et al., 2018a; Shoukry et al., 2017). Such algorithms often 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 (Loizou and Kyriakopoulos, 2004; Schillinger et al., 2018b). As the number of vehicles increases, the dimension of the product abstraction increases exponentially.

Email addresses: jfang@wpi.edu (Jie Fang),
zzhang@wpi.edu (Zetian Zhang), rvcowlagi@wpi.edu
(Raghvendra V. Cowlagi).

Sampling-based asymptotically optimal techniques are reported to construct team product abstractions with fewer states (Kantaros and Zavlanos, 2018b,c).

On the other hand, several *decentralized* algorithms (Filippidis et al., 2012; Guo and Dimarogonas, 2015; Guo et al., 2016; Tumova and Dimarogonas, 2016; Moarref and Kress-Gazit, 2017; Guo and Dimarogonas, 2017) provide motion coordination 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 (Kantaros and Zavlanos, 2018a). 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.

There are three gaps in the literature, which the proposed work addresses. First, *decentralized* methods 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 (Rungger and Zamani, 2016); the product of multiple such models for a team of vehicles becomes impractically large. A reduced-order product is reported (Tumova and Dimarogonas, 2016), 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 (Karaman and Frazzoli, 2012; Vasile and Belta, 2013; Kantaros and Zavlanos, 2018b,c), but these algorithms are ill-suited in the present context. 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 ill-advised here because these algorithms can report different route costs for the same specification in different invocations. This behavior can disrupt the convergence of the task assignment algorithm.

Second, 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. (Guo and Dimarogonas, 2015) or **(2)** considers linear models, cf. (Shoukry et al., 2017), or **(3)** altogether ignores the vehicle model and presupposes a finite state discrete abstraction, cf. (Tumova and Dimarogonas, 2016). For vehicles with nonholonomic kinematic constraints, including the simple Dubins car model, the presumption of a trivial discrete abstraction is not justified (Cowlagi and Zhang, 2017). The reason is that reachability analyses in the literature for such vehicles neglect state constraints (obstacles).

Third, the assignment of the *same* collaborative task(s) to multiple vehicles is not considered. Such assignments 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. In these applications, synchronization of multiple agents is required. Such synchronization entails situations where an agent has to wait while other collaborating agents arrive. Therefore, it is desirable to minimize the waiting time of agents.

The contributions of this paper are in bridging these three gaps in the literature with a decentralized route-planning method to satisfy a global LTL specification, including decentralized decomposition into local specifications. The proposed method leverages the consensus-based bundle algorithm (Choi et al., 2009). The lifted graph algorithm (Cowlagi and Zhang, 2017) is used to address kinematic turn radius constraints. In conjunction with these algorithms, we propose a new algorithm to assign *collaborative* tasks, namely, those assigned to multiple vehicles. The proposed algorithm achieves task assignment that minimizes the maximum waiting duration among all agents. For the sake of numerical comparisons, we also propose a new extension of the consensus-based group algorithm (Hunt et al., 2014) to satisfy LTL specifications with kinematically feasible routes. The proposed route planner is demonstrated via numerical simulation examples and a decentralized implementation on single-board computers. Preliminary results of this paper appear in (Fang et al., 2018), which did not address collaborative tasks.

The rest of the paper is organized as follows. We introduce the main problem elements in §1. In §2, we discuss the proposed algorithms for solving this problem. In §3, we provide illustrative numerical simulation examples, and we conclude the paper in §4.

1 Problem Formulation

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 $[N] := \{1, \dots, N\}$. The number of vehicles in the collaborative team is denoted N_A . 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.

The *workspace* $\mathcal{W} \in \mathbb{R}^2$ is a compact planar region where the vehicles operate. Consider a finite partition of $\mathcal{W}$ into convex subregions called *cells*, and the associated topological graph $\mathcal{G} := (V, E)$. Vertices in V are labeled by natural numbers. The cell associated with vertex $n \in V$ is denoted $\text{cell}(n) \subset \mathcal{W}$. A *route* $\mathbf{v}$ in $\mathcal{G}$ is a sequence $(v_0, v_1, \dots)$ of vertices, such that $v_\ell \in V$, and $(v_{\ell-1}, v_\ell) \in E$, for each $\ell \in \mathbb{N}$. We assume that 1 time unit is taken

for a vehicle to move between two adjacent cells, and that the time taken to complete each task is a constant. The collection of all routes in $\mathcal{G}$ is denoted $\mathcal{L}_{\mathcal{G}}$. Special cells called *regions of interest (ROIs)* are prespecified. The number of ROIs is N_R .

Atomic propositions are associated with each ROI λ_n , $n \in [N_R]$, such that λ_n is **true** whenever a prespecified number of vehicles are simultaneously within the ROI. Any cell in the workspace that is not an ROI is associated with a common atomic proposition. The LTL subclass considered in this work is called LTL_X and it excludes the **next** operator. The temporal operators $\Diamond$ (**eventually**) and $\Box$ (**always**) are considered. An LTL formula is constructed using one or more atomic propositions and logical and temporal operators. A *word* $\bar{\omega} = (\omega_0, \omega_1, \dots)$ is a sequence such that each ω_n is a subset of $\{\lambda_n\}_{n=0}^{N_R}$. For $m, n \in \mathbb{Z} \geq 0, n \geq m$, the word $(\omega_m, \omega_{m+1}, \dots, \omega_n)$ is denoted $\bar{\omega}_m^n$. The *satisfaction of formula* ϕ by the word $\bar{\omega}$ is recursively defined as: (1) $\bar{\omega}$ satisfies λ_n if $\lambda_n \in \omega_0$, (2) $\bar{\omega}$ satisfies $\neg\phi$ if $\bar{\omega}$ does not satisfy ϕ , (3) $\bar{\omega}$ satisfies $(\phi_1 \wedge \phi_2)$ if $\bar{\omega}$ satisfies ϕ_1 and $\bar{\omega}$ satisfies ϕ_2 , (4) $\bar{\omega}$ satisfies $(\phi_1 \mathcal{U} \phi_2)$ if there exists $n \geq 0$ such that $\bar{\omega}_n^\infty$ satisfies ϕ_2 and for every $m < n, \bar{\omega}_m^n$ satisfies ϕ_1 . Each route $\mathbf{v} = (v_0, v_1, \dots) \in \mathcal{L}_{\mathcal{G}}$ defines a word $\bar{\omega}(\mathbf{v}) = (\omega_0, \omega_1, \dots)$ where $\omega_\ell := \{\lambda_n \mid v_\ell = v_n\}$. The route $\mathbf{v}$ is *satisfies* formula ϕ if the word $\bar{\omega}(\mathbf{v})$ satisfies ϕ .

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

The desired specification ϕ is assumed to belong to a subclass of LTL such that ϕ is a conjunction of a *safety* specification ϕ^S and a *liveness* specification ϕ^L in the standard form $\phi = \phi^S \wedge \phi^L$. A safety specification is an invariance specification of the type “*undesirable events will not happen*,” whereas a liveness specification is an “eventuality” specification of the type “*desirable events will happen*” (Emerson, 1990).

A typical safety specification of interest to robotic vehicles is obstacle avoidance. Typical liveness specifications of interest to robotic vehicles are: (1) *sequencing* specifications of the form $\Diamond(\lambda_{n_1} \wedge \Diamond(\lambda_{n_2} \wedge \Diamond(\dots \wedge \Diamond\lambda_{n_\ell})))$, which involve visiting ROIs in a specific sequence, and (2) *coverage* specifications such as $\Diamond\lambda_n$ that involve visiting ROIs once.

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. The scope of this work is limited to liveness specifications in the normal form of conjunction of N_T specifications, ϕ_j , $j \in [N_T]$, i.e. $\phi^L := \bigwedge_{j=1}^{N_T} \phi_j$.

The specifications ϕ^S and each ϕ_j belong to the LTL_X class. Each ϕ_j is either a sequencing or coverage specification and may include boolean operators. Examples include:

$$\begin{aligned} \phi_j &= \Diamond\lambda_{n_1}, & \phi_j &= \Diamond(\lambda_{n_1} \wedge \Diamond(\lambda_{n_2} \wedge \Diamond(\dots \wedge \Diamond\lambda_{n_\ell}))), \\ \phi_j &= \mathbf{Env_Cond} \rightarrow \Diamond\lambda_n \end{aligned}$$

where $n_1, n_2, \dots \in [N_R]$ and **Env_Cond** is a proposition dependent on the state of the environment (e.g. target present or not present). 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 $1, \dots, N_{TC}$ and the independent tasks are labeled $N_{TC} + 1, \dots, N_T$. 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.

Problem 1 *Given an LTL_X specification ϕ over $\{\lambda_n\}_{n=0}^{N_R}$, find routes $(\mathbf{v}_1, \dots, \mathbf{v}_{N_A})$ with each $\mathbf{v}_i \in \mathcal{L}_{\mathcal{G}}$ such that the word $\bar{\omega}(\mathbf{v}_1, \dots, \mathbf{v}_{N_A})$ satisfies ϕ . $\square$*

We seek a *decentralized* solution to Problem 1 where agent $i \in [N_A]$ 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 $i \in [N_A]$ is denoted $\mathcal{N}_i$. We assume that the network topological graph is connected. We require that the routes solving Problem 1 be traversable by the vehicles. To this end, we consider the following vehicle models.

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 commonly used in warehousing (Stavrou et al., 2018). 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:

$$\dot{x}(t) = a \cos \psi(t), \quad \dot{y}(t) = a \sin \psi(t), \quad \dot{\psi}(t) = u(t), \quad (1)$$

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 $[-\frac{1}{\rho}, \frac{1}{\rho}]$, 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 (see Ex. 3 in §3).

The constant speed assumption in the KCPM is justified from a practical perspective. Most aircraft, whether fixed-wing or rotorcraft, spend the majority of their flight time in cruise, which is flight at constant speed and constant altitude. Continuous variations in speed are avoided for the sake of engine efficiency. The distinction between the UCPM and KCPM is motivated primarily by the recent advances of vertical take-off and landing (VTOL) aircraft, which are capable of hovering. In such aircraft, the transitions between cruise flight and hover are short transient maneuvers that can be ignored for route-planning purposes.

The method of *lifted graphs* is used to construct a discrete abstraction for the KCPM. Briefly, this method considers *sequences* of length $(H + 1)$ of edge transitions in the cell decomposition graph $\mathcal{G}$, where $H \geq 1$ is a user-specified integer, and associates these edge transition sequences with kinematically admissible trajectories. In other words, this method “lifts” the graph $\mathcal{G}$ to define a new graph whose vertices correspond to sequences of adjacent cells that are traversable within the kinematic constraints. Route-planning is then performed by searching this new lifted graph. To this end, this method computes the H -cost of a route $\mathbf{v}$, denoted $\tilde{J}(\mathbf{v})$. Routes with a finite H -cost are guaranteed to be KCPM-traversable. This method was previously developed by the second and third authors of this paper (Cowlagi and Zhang, 2017). §2.3 provides a brief overview of this method; however its details are too lengthy and tedious to be included in this manuscript.

A limitation of this work is that Problem 1 does not consider modeling uncertainty. 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. This assumption is justified by the fact that 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 robustness to the route plan.

Example 1. Consider an instance of Problem 1 with global specification $\phi := \phi^S \wedge \phi^L$, $\phi^S := \square(\lambda_1 \wedge \neg \lambda_2)$, and

$$\phi^L := \diamond \lambda_3 \wedge \diamond \lambda_4 \wedge \diamond \lambda_5 \wedge \diamond \lambda_6 \wedge \diamond (\lambda_7 \wedge \diamond \lambda_8) \wedge \diamond \lambda_9. \quad (2)$$

Here, $N_A = 4$, ROIs $\lambda_3, \dots, \lambda_9$ are indicated by yellow cells in Fig. 1(b), and the vehicles’ initial locations are indicated by red cells. ROIs λ_2 (obstacles) are indicated in

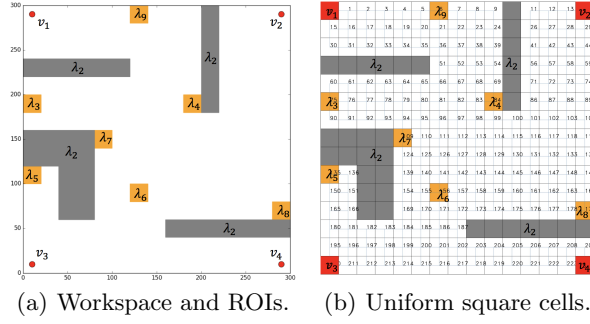


Fig. 1. An instance of Problem 1 (details in Ex. 1).

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 conjunctive form $\phi^L = \bigwedge_{j=1}^{N_T} \phi_j$ with $N_T = 6$, $N_{TC} = 2$, $\phi_1 := \diamond \lambda_4$, $\phi_2 := \diamond \lambda_6$, $\phi_3 := \diamond \lambda_3$, $\phi_4 := \diamond \lambda_5$, $\phi_5 := \diamond (\lambda_7 \wedge \diamond \lambda_8)$, and $\phi_6 := \diamond \lambda_9$. $\square$

2 Decentralized Route-planning

The proposed method for solving Problem 1 is to first find local specifications ψ_i such that $\phi^L = \bigwedge_{i=1}^{N_A} \psi_i$. The specification to be satisfied by agent i is then given by $\phi^S \wedge \psi_i$ for each $i \in [N_A]$. Next, each agent i executes a route-planning algorithm to find a kinematically feasible route $\mathbf{v}_i$ satisfying $\phi^S \wedge \psi_i$. To find routes traversable by the KCPM, we apply the method of lifted graphs (§2.3).

After the local specifications ψ_i are determined, each agent constructs a Büchi automaton, whose accepting runs are exactly the strings satisfying $\phi^S \wedge \psi_i$. Next, a product of this Büchi automaton with a finite state model called a *discrete abstraction* of the dynamical system is constructed. For the UCPM, this discrete abstraction is the same as the cell decomposition graph $\mathcal{G}$. For the KCPM, this discrete abstraction is provided by the method of lifted graphs (§2.3). Next, a search algorithm, such as Dijkstra’s algorithm, is executed on this product automaton to find an optimal run. Dijkstra’s algorithm is known to be sound and complete, i.e., it returns a solution whenever it exists, or terminates within a finite number of iterations otherwise and returns failure. The run of the product automaton is then projected onto the discrete abstraction to recover the route plan. This process is widely used in LTL route-planning, and we refer the reader interested to (Cowlagi and Zhang, 2017; Kloetzer and Belta, 2007) for technical details.

2.1 Finding Local Liveness Specifications

First, consider the special case of Problem 1 with zero collaborative specifications, i.e., $N_{TC} = 0$. For this case,

we apply the *consensus-based bundle algorithm* (CBBA, (Choi et al., 2009)) to assign to each agent i the local liveness specification ψ_i . A brief overview of this algorithm is as follows.

2.1.1 Consensus-Based Bundle Algorithm

The CBBA is based on decentralized auction algorithms (Zavlanos et al., 2008) for near-optimal task assignment. Each agent i maintains a *task bundle* $\mathbf{b}_i$, which is a set of task indices, and a *task path* $\mathbf{p}_i$, which is an ordered sequence of task indices in $\mathbf{b}_i$. After the termination of the CBBA, each agent computes a unique task path and task bundle. Suppose $\mathbf{p}_i = (j_1, j_2, \dots, j_N)$, with each $j_\ell \in \{N_{TC} + 1, \dots, N_T\}$. Then we define

$$\psi_i := \Diamond(\phi_{j_1} \wedge \Diamond(\phi_{j_2} \wedge \Diamond(\dots \wedge \phi_{j_N}))), \quad (3)$$

which is a local liveness specification that encodes the ordered sequence in $\mathbf{p}_i$. The specification ψ_i associates with $\mathbf{p}_i$ a vehicle route $\mathbf{v}(\mathbf{p}_i)$ using the route-planning method discussed in §2.3. The “incentive” for any agent to perform task j is quantified by a sufficiently large positive constant $\bar{r}_j$. For the numerical simulation results in §3, we set $\bar{r}_j = 100$. Each bundle is then associated with a *reward* for which we propose the following definition of the *marginal reward* r_{ij} for adding a new task j :

$$r_{ij}[\mathbf{b}_i] := \bar{r}_j - \min_{n \leq |\mathbf{p}_i|} \bar{J}(\mathbf{v}(\mathbf{p}_i \oplus_n j)), \quad (4)$$

where $\oplus$ denotes insertion of an element into a list, and its subscript indicates the index of insertion into $\mathbf{p}_i$.

The CBBA consists of two phases: auction and consensus. In the auction phase, each agent i keeps inserting tasks j into $\mathbf{b}_i$ and $\mathbf{p}_i$ by $\mathbf{b}_i := \mathbf{b}_i \oplus_{\text{end}} j$, and $\mathbf{p}_i := \mathbf{p}_i \oplus_{n_j^*} j$, where $n_j^* := \arg \min_{n \leq |\mathbf{p}_i|} \bar{J}(\mathbf{v}(\mathbf{p}_i \oplus_n j))$. Then, the agent i bids on a task bundle with maximum reward. A centralized auctioneer is not required; instead decentralized auction is achieved through consensus.

In the consensus phase, a time-stamped lists of the highest bid and the winning agent for each task is exchanged and updated among neighboring agents based on pre-defined decision rules for conflict resolution (see (Choi et al., 2009) for details). If the i^{th} agent is outbid on task j , it releases not only this task, but all tasks in its current bundle $\mathbf{b}_i$ that were added after the task j .

The convergence of the CBBA is guaranteed if the marginal reward r_{ij} satisfies a *diminishing marginal gain* (DMG) condition: namely, that the marginal reward of a task j does not increase as other tasks are added into the bundle before it. The definition (4) of r_{ij} satisfies the DMG condition because the marginal increase $\bar{J}(\mathbf{v}(\mathbf{p}_i \oplus_n j)) - \bar{J}(\mathbf{v}(\mathbf{p}_i))$ is nonnegative for any n . Therefore, the convergent property of the CBBA is retained under the proposed definition of reward.

2.1.2 Consensus-Based Grouping Algorithm

The CBBA is not capable of assigning the same (collaborative) task to multiple agents. In the general case where i.e., $N_{TC} > 0$, 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, (Hunt et al., 2014)). Note that (Hunt et al., 2014) does not refer to LTL. The application of CBGA to satisfying LTL formulae, denoted E-CBGA, is an extension that we develop for the sake of comparison against the proposed approach.

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 $[N_T]$ with rewards defined by H -costs as in (4). Each agent i is assigned a task path $\mathbf{p}_i$, and the local liveness specification ψ_i is found per (3). 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 $j \in [N_{TC}]$, 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.

2.1.3 Proposed Algorithm

In the general case of Problem 1 when $N_{TC} > 0$, 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. 2–4. 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 $j \in [N_{TC}]$, 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

Bundle Construction Phase at Iteration $t \geq 1$

```

1:  $\mathbf{b}_i(t) = \mathbf{b}_i(t-1)$ ,  $\mathbf{p}_i(t) = \mathbf{p}_i(t-1)$ ,
2:  $\mathcal{F}_i(t) = \mathcal{F}_i(t-1)$ ,  $\mathcal{G}_i(t) = \mathcal{G}_i(t-1)$ 
3:  $\mathbf{u}_i^*(t) = \mathbf{u}_i^*(t-1)$ ,  $\tau_{\mathbf{u}_i^*,j}(t) = \tau_{\mathbf{u}_i^*,j}(t-1)$ 
4:  $h_{ij}(t) = 0$ , for each  $j \in [N_{\text{TC}}]$ 
5:  $n_{\text{last}}(t) := \max\{n_j | j \in [N_{\text{TC}}] \cap \mathbf{p}_i(t)\}$ 
6: while  $|\mathbf{b}_i(t)| < N_T$  do
7:   for  $j \in [N_{\text{TC}}] \setminus \mathbf{b}_i(t)$  do
8:      $n_j^* := \arg \max_{n_{\text{last}}(t) \leq n \leq |\mathbf{p}_i|} r_{ij}$ 
9:      $f_{iji}(t) := \bar{r}_j - \tilde{\mathcal{J}}(\mathbf{v}(\mathbf{p}_i \oplus_{n_j^*} j | j))$ 
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  $i \in \mathbf{u}_j$ ,  $h_{ij}(t) = 1$ 
15:       $f_{iji}(t) = f_{iji}(t)h_{ij}(t)$ 
16:       $j^* := \arg \max_{j \in [N_{\text{TC}}] \setminus \mathbf{b}_i(t)} r_{ij}h_{ij}(t)$ ,
17:       $\mathbf{b}_i(t) := \mathbf{b}_i(t) \oplus_{\text{end}} j^*$ ,  $\mathbf{p}_i(t) := \mathbf{p}_i(t) \oplus_{n_{j^*}} j^*$ 
18:       $\mathbf{u}_{ij^*}^*(t) = \mathbf{u}_{j^*}$ ,  $\tau_{\mathbf{u}_{ij^*}^*,j^*}(t) = \tau_{j^*}$ 
19:       $g_{ij^*i}(t) = 1$ 

```

Fig. 2. Bundle construction phase of the proposed algorithm.

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 $\mathcal{F}_i(t), \mathcal{G}_i(t) \in \mathbb{R}^{N_T \times N_A}$ 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 $n_{\text{last}}(t)$ denote the sequential position in $\mathbf{p}_i(t)$ of the last collaborative task. At $t = 1$, when no collaborative tasks are yet assigned, $n_{\text{last}}(1) = 0$. 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 $j \in [N_{\text{TC}}]$ that is not present in $\mathbf{b}_i(t)$, an optimal position for insertion n_j^* is computed (Line 8 in Fig. 2). Note that sequential positions only *after* $n_{\text{last}}(t)$ are considered, due to which rewards for tasks already assigned do not change due to insertion of task j .

The reward $f_{iji}(t)$ is updated based on the traversal cost $\tilde{\mathcal{J}}(\mathbf{v}(\mathbf{p}_i \oplus_{n_j^*} j | j))$ of partial completion of $\mathbf{p}_i \oplus_{n_j^*} j$ (Line 9 in Fig. 2). By partial completion we mean the sequential execution of tasks in $\mathbf{p}_i$ until the n_j^{th} task.

Waiting Duration Calculation Δ_{ij}

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

```

1:  $\mathbf{u}_{ij}^*(t) := \mathbf{u}_{ij}^*(t-1)$ ,  $\tau_{\mathbf{u}_{ij}^*,j}(t) = \tau_{\mathbf{u}_{ij}^*,j}(t-1)$ 
2: for  $k \in N_A$  such that  $f_{ijk} > 0$  do
3:   Find  $\mathbf{u} \subset [N_{\text{TC}}]$  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:      $\tau_{\mathbf{u}_{ij}^*,j}(t) := \tau$ ,  $\mathbf{u}_{ij}^*(t) := \mathbf{u}$ 
7: Return  $\mathbf{u}_{ij}^*(t), \tau_{\mathbf{u}_{ij}^*,j}(t)$ 

```

Fig. 3. Waiting duration calculation for task j by agent i .

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. 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. 2). If $\|\mathbf{g}_{ij}(t)\|_1 \geq M_j$, a sufficient number of agents are currently assigned to task j , but this assignment may not be optimal in terms of the agents' 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. 2).

After updating the task availability flags $h_{ij}(t)$, the element $f_{iji}(t)$ of the reward matrix is updated (Line 15) such that the rewards for non-available tasks are reset to zero. Task j^* with maximum reward is found (Line 16) 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 updated (Line 18).

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 maximum waiting duration is minimal. To this end, each agent k with a nonzero reward for task j is identified (Line 2 in Fig. 3) 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 $\mathbf{u}$, the waiting duration is the time difference between the arrivals of the first and last agents at the ROI. Because one time unit is taken to move between two adjacent cells, the arrival time for each agent in $\mathbf{u}$ can be computed based on its route length. 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). The set $\mathbf{u}_{ij}^*(t)$ 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

Consensus Phase At Iteration $t \geq 1$

```

1: for  $j \in [N_{TC}]$  do
2:   for  $k \in \mathcal{N}_i$  do
3:     if  $s_{km}(t) > s_{im}(t), \forall m \in N_A$  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:      $f_{iji}(t) = 0, \quad g_{iji}(t) = 0$ 

```

Fig. 4. Consensus phase of the proposed algorithm.

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 $m \in [N_A]$, $s_{km}(t) > s_{im}(t)$, then agent k has communicated with agent m more recently than agent i (Line 3 in Fig. 4). The set of agents $\mathbf{u}_{ij}^*(t)$ and the waiting duration $\tau_{\mathbf{u}_{ij}^*, j}(t)$ is recalculated (Line 5 in Fig. 4). 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. 4).

2.2 Convergence, Complexity, and Optimality

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

Lemma 2 *At any iteration $t \geq 1$, suppose there exists a task index $j^\dagger \in [N_{TC}]$ 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 $s \geq t$.*

Proof. If an agent i is assigned to collaborative task $j^\dagger$, let $\tau_{j^\dagger}^i$ 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,

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

For any set of agents $\mathbf{u} \subset [N_A]$ 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) satisfies

$$\tau_{\mathbf{u}, j^\dagger}(t) = \max \left\{ \max_{i \in \mathbf{u}} \{\tau_{j^\dagger}^i(t)\}, \quad \tau_{\mathbf{u}, j^\dagger}(t-1) \right\}. \quad (5)$$

It follows that $\tau_{\mathbf{u}, j^\dagger}(t) \leq \tau_{\mathbf{u}, j^\dagger}(s)$ for every $s \geq t$, 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 (Choi et al., 2009), the property (5) also implies that for any two sets $\mathbf{u}_1, \mathbf{u}_2 \subset [N_A]$ with $|\mathbf{u}_1| = |\mathbf{u}_2| = M_{j^\dagger}$,

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

If there exists $j^\dagger \in [N_{TC}]$ as in the statement of the Lemma, then $\tau_{\mathbf{u}_{j^\dagger}^*, j^\dagger}(t) \leq \tau_{\mathbf{u}, j^\dagger}(t)$, for every $\mathbf{u} \subset [N_A]$ with $|\mathbf{u}| = M_{j^\dagger}$ 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. $\square$

Proposition 3 *Let δ be the diameter¹ of the agents' communication network graph. Then the proposed algorithm terminates in at most $\frac{1}{2}N_{TC}(N_{TC}+1)\delta$ iterations.*

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

$$f_{ij^\dagger m}(t) = f_{\ell j^\dagger m}(t), \quad i, \ell, m \in [N_A] \text{ and } j^\dagger \in [N_{TC}]. \quad (6)$$

After consensus, $\mathbf{u}_{j^\dagger}^*$ is found in Line 5 in Fig. 4. By (6), the condition in the statement of Lemma 2 is satisfied for some $j_1^\dagger \in [N_{TC}]$, and $\mathbf{u}_{j_1^\dagger}^*$ does not change after iteration t . Similarly, after at most $(N_{TC} - 1)\delta$ further iterations, another task $j_2^\dagger \in [N_{TC}] \setminus \{j_1^\dagger\}$ meets the condition in the statement of Lemma 2, and consequently $\mathbf{u}_{j_2^\dagger}^*$ is fixed. Continuing further so on, it follows that in at most $\frac{1}{2}N_{TC}(N_{TC} + 1)\delta$ iterations, $\mathbf{u}_{j^\dagger}^*$ is fixed for all $j^\dagger \in [N_{TC}]$, and the algorithm terminates. $\square$

Note that the task assignment resulting from the proposed algorithm is deadlock-free. To see this, consider for the sake of contradiction a deadlocked task assignment: agent i_1 is assigned to complete task j_1 first, followed by j_2 whereas agent i_2 is assigned to complete task j_2 first, followed by j_1 . The deadlock is that agents i_1 and i_2 are assigned same tasks in opposite sequence. The proposed algorithm will never result in such a task assignment because after the assignment for one task, say j_1 , is complete, other tasks (e.g., j_2) can only be inserted *after* j_1 due to Line 8 in Fig. 2.

Proposition 4 *For each task $j \in [N_{TC}]$, the proposed algorithm minimizes the maximum waiting duration $\tau_{\mathbf{u}_j^*, j}$ over all agents assigned to the task.*

Proof. Per the proof of Prop. 1, each $j \in [N_{TC}]$ satisfies the condition in the statement of Lemma 2 in some iteration. After this iteration, per the proof of Lemma 2, $\tau_{\mathbf{u}_j^*, j}(t) \leq \tau_{\mathbf{u}, j}(t)$ for every $\mathbf{u} \subset [N_A]$ with $|\mathbf{u}| = M_j$ and $t \in \mathbb{N}$, which means that the maximum waiting duration for $\mathbf{u}_j^*$ always remains the least among all possible sets that can be assigned to task j . Furthermore, due to the insertion policy in Line 8 in Fig. 2, the assignment of

¹ The diameter of a graph is the largest distance between any pair of vertices, where distance is measured as the number of edges in a path between vertices.

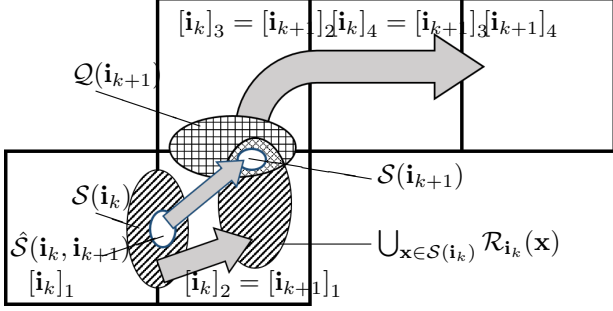


Fig. 5. Illustration (with $H = 3$) of edge cost assignment in the lifted graph.

other tasks does not affect the waiting duration of task j in further iterations of the algorithm, and the optimality of $\tau_{\mathbf{u}_j^*, j}$ is preserved. $\square$

Note on computational complexity: At each iteration of the proposed algorithm, the only nontrivial computation is of the reward in Line 9 in Fig. 2, which involves route-planning to satisfy an LTL formula. Per (Cowlagi and Zhang, 2017), the worst-case complexity of this computation is $\mathcal{O}(|\mathcal{G}|N_T 4^H (4 + \log(|\mathcal{G}|N_T 4^H)))$. Therefore, by Prop. 1, the worst-case computational complexity of each agent is $\mathcal{O}[\delta N_{TC}^2 |\mathcal{G}|N_T 4^H (4 + \log(|\mathcal{G}|N_T 4^H))]$. This result shows a significant innovation over the state-of-the-art because the computational complexity does not depend on the number of agents.

2.3 Lifted Graph

To ensure the traversability of routes by the KCPM, the *lifted graph* algorithm is applied. A brief and informal description of the lifted graph is provided here; for technical details the reader interested is referred to (Cowlagi and Zhang, 2017).

Recall the cell decomposition graph $\mathcal{G} = (V, E)$, and for every integer $H \geq 0$, define

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

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 $k < m \leq H + 1$. 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 $k = 2, \dots, H + 1$, 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) .

Edge transitions costs in $\mathcal{G}_H$ encode kinematic traversability characteristics of a vehicle by associating forward- and backward reachability properties of the vehicle model. To this end, consider an edge $(\mathbf{i}, \mathbf{j}) \in E_H$, with $\mathbf{i}, \mathbf{j} \in V_H$. This edge is associated with a sequence of $H + 2$ successively adjacent cells, as shown in Fig. 5. Let $\mathcal{S}(\mathbf{i})$ be a set of states of the vehicle model such

that the position components of the elements in $\mathcal{S}(\mathbf{i})$ lie on the boundary between the first and second cells, as shown in in Fig. 5. Next let $\mathcal{Q}(\mathbf{j})$ be a set of states of the vehicle model 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 $\mathcal{Q}(\mathbf{j})$. Next, let $\mathcal{R}_i(\mathbf{x})$ be the set of all states that can be reached from $\mathbf{x}$ by trajectories whose position components always remain within the second cell of $\mathbf{i}$. Finally, we define $\hat{\mathcal{S}}(\mathbf{i}, \mathbf{j}) := \{\mathbf{x} \in \mathcal{S}(\mathbf{i}) : \mathcal{R}_i(\mathbf{x}) \cap \mathcal{Q}(\mathbf{j}) \neq \emptyset\}$ (see Fig. 5.)

Now consider a path $\mathbf{v} = \{v_0, v_1, \dots, v_P\} \in \mathcal{L}_{\mathcal{G}}$. For $k \in \mathbb{N}$, let $\mathbf{i}_k := (v_k, \dots, v_{k+H})$. Clearly, $(\mathbf{i}_k, \mathbf{i}_{k+1}) \in E_H$. We define $g_H : E_H \rightarrow \mathbb{R}_+$ and $\mathcal{S}(\cdot)$ as follows:

$$\mathcal{S}(\mathbf{i}_{k+1}) := \bigcup_{\mathbf{x} \in \hat{\mathcal{S}}(\mathbf{i}_k, \mathbf{i}_{k+1})} (\mathcal{R}_{\mathbf{i}_k}(\mathbf{x}) \cap \mathcal{Q}(\mathbf{i}_{k+1})), \quad (7)$$

$$g_H(\mathbf{i}_k, \mathbf{i}_{k+1}) := \begin{cases} \infty, & \text{if } \mathcal{S}(\mathbf{i}_{k+1}) = \emptyset, \\ 1, & \text{otherwise,} \end{cases} \quad (8)$$

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:

$$\bar{J}(\mathbf{v}) := H + \sum_{k=0}^{P-H} g_H(\mathbf{i}_k, \mathbf{i}_{k+1}). \quad (9)$$

Efficient algorithms for computing the sets $\mathcal{R}(\cdot)$, $\mathcal{S}(\cdot)$, and $\mathcal{Q}(\cdot)$ for the KCPM are available (Cowlagi and Zhang, 2017). Any route in $\mathcal{G}$ with finite H -cost is guaranteed to be traversable (Cowlagi and Zhang, 2017).

Route-planning is performed by each agent i to satisfy the local specification ψ_i using the lifted graph as a discrete abstraction for the KCPM, or the original graph $\mathcal{G}$ for the UCPM. The reader is referred to (Cowlagi and Zhang, 2017) for technical details, including a detailed discussion on computational complexity.

3 Results and Discussion

3.1 Numerical Simulation Examples

The following simulations were performed on a desktop computer with a 3.4 GHz Intel Core i7 processor, 12GB RAM, and the Ubuntu 16.04 operating system. The software written to implement these simulations is freely available to the reader: https://github.com/jfangwpi/decentralized_route_planning.git. Some details from these examples are omitted for brevity and are available in the first author's doctoral thesis (Fang, 2020).

Example 1 (continued). The specification (2) refers to a global task involving coverage, sequencing, and obstacle avoidance. The workspace is decomposed into uniform square cells in a 15×15 grid as shown in Fig. 1(b). The agent communication network is shown below, and the UCPM is considered.

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

First, the CBBA is executed to assign independent tasks, which results to the task paths:

$$\mathbf{p}_1(0) = (6, 3), \quad \mathbf{p}_2(0) = (5), \quad \mathbf{p}_3(0) = (4), \quad \mathbf{p}_4(0) = \text{null},$$

which leads to the following local specifications, per (3): $\psi_1 := \diamond(\lambda_9 \wedge \diamond\lambda_3)$, $\psi_2 := \diamond(\lambda_7 \wedge \diamond\lambda_8)$, $\psi_3 := \diamond\lambda_5$, and $\psi_4 := \text{null}$. This assignment is achieved over three bundle construction and consensus iterations.

Next, the proposed algorithm assigns collaborative tasks 1 and 2. Tasks 1 and 2 require $M_1 = 2$ and $M_2 = 3$ vehicles, respectively. The iterative updates of the matrices $\mathcal{F}_i$ and $\mathcal{G}_i$ are indicated in Table 1.

In the first bundle construction iteration, each agent i assigns both collaborative tasks to itself. Each agent selects for each of the collaborative tasks M_j agents to minimize the waiting duration (Line 4 of Figure 3) 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, the proposed algorithm converges and results in the following task paths:

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

which leads to the following local specifications per (3) $\psi_1 := \diamond(\lambda_9 \wedge \diamond(\lambda_6 \wedge \diamond(\lambda_4 \wedge \diamond\lambda_3)))$, $\psi_2 := \diamond(\lambda_6 \wedge \diamond(\lambda_7 \wedge \diamond\lambda_8))$, $\psi_3 := \diamond(\lambda_5 \wedge \diamond(\lambda_6 \wedge \diamond\lambda_4))$, and $\psi_4 := \text{null}$.

Fig. 6(a) illustrates 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 vehicles 1 and 3 arrive at ROI λ_4 simultaneously and for task 2 vehicles 1 and 3 arrive at ROI λ_6 and wait for 2 time units until vehicle 2 arrives.

By comparison, task assignment by E-CBGA results in the following local specifications: $\psi_1 = \diamond(\lambda_9 \wedge \diamond(\lambda_4 \wedge \diamond\lambda_3))$, $\psi_2 = \diamond(\lambda_4 \wedge \diamond\lambda_6)$, $\psi_3 = \diamond(\lambda_5 \wedge \diamond(\lambda_6 \wedge \diamond(\lambda_7 \wedge \diamond\lambda_8)))$, $\psi_4 = \diamond\lambda_6$.

The route traveled by each vehicle is shown in Fig. 6(b). In this case, the sum of lengths of all the vehicles' routes is 94 units. However, the waiting duration 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. $\square$

Table 1
Proposed algorithm execution in Ex. 1.

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

Example 2. We consider Ex. 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: $\psi_1 := \diamond(\lambda_9 \wedge \diamond(\lambda_6 \wedge \diamond\lambda_3))$, $\psi_2 := \diamond(\lambda_7 \wedge \diamond\lambda_8)$, $\psi_3 := \diamond(\lambda_6 \wedge \diamond(\lambda_4 \wedge \diamond\lambda_5))$, $\psi_4 := \diamond(\lambda_6 \wedge \diamond\lambda_4)$. The routes of the vehicles to satisfy these local specifications are shown in Fig. 7(a). First, in comparison with Ex. 1, notice in Fig. 6(a) that the routes of all vehicles involve several sharp turns, which cannot be executed by a vehicle with

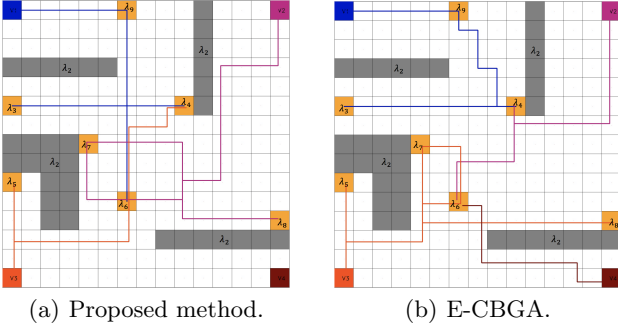


Fig. 6. Routes of all vehicles in Ex. 1: vehicles 1–4 indicated in blue, magenta, red, and maroon, respectively.

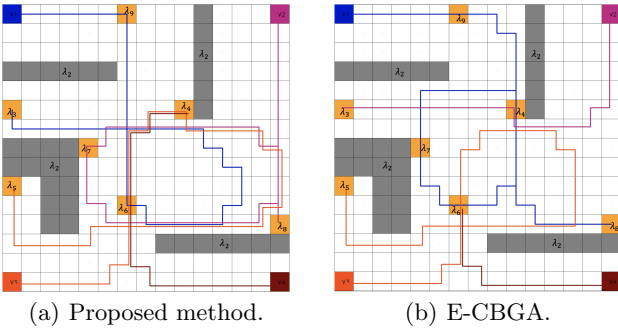


Fig. 7. Routes of all vehicles in Ex. 2: vehicles 1–4 indicated in blue, magenta, red, and maroon, respectively.

minimum turn radius of 3 units. By contrast, in Fig. 7 such sharp turns are absent. Each route indicated in Fig. 7 is guaranteed to contain a differentiable curve with minimum turn radius of 3 units.

By comparison, the vehicles routes associated with the task assignments by E-CBGA are shown in Fig. 7(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 Ex. 1, whereas the proposed algorithm results in longer paths, the total waiting duration is significantly lower. $\square$

Example 3. Consider Example 1 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: $\psi_1 := \Diamond(\lambda_9 \wedge \Diamond(\lambda_4 \wedge \Diamond(\lambda_6)))$, $\psi_2 := \Diamond(\lambda_4 \wedge \Diamond(\lambda_6 \wedge \Diamond(\lambda_3)))$, $\psi_3 := \Diamond(\lambda_5 \wedge \Diamond(\lambda_6 \wedge \Diamond(\lambda_7 \wedge \Diamond(\lambda_8))))$. The routes of the vehicles to satisfy these local specifications are shown in Fig. 8(a). First, note that these local specifications and routes are different compared to those in Ex. 2, although the global specification is the same. For example, the route planned for vehicle 1 (blue) is significantly different because of the larger min-

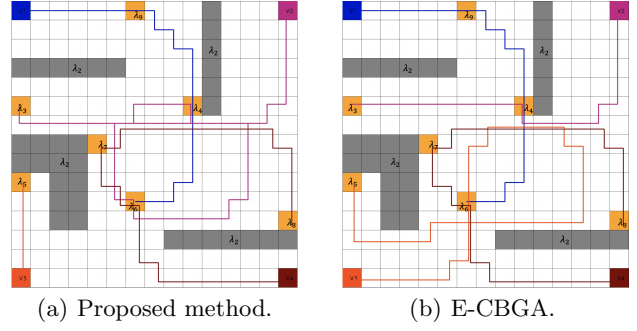


Fig. 8. Routes of all vehicles in Ex. 3: vehicles 1–4 indicated in blue, magenta, red, and maroon, respectively.

imum turn radius in this example. In comparison, the vehicles routes associated with the task assignments by E-CBGA are shown in Fig. 8(b). The sum of lengths of vehicle routes 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. $\square$

Example 4. The specification $\phi = \phi^S \wedge \phi^L$ in this example involves the operator **implies** denoted $\rightarrow$, which enables vehicular motion conditioned on the environment. The safety and liveness specifications are $\phi^S = \Box(\lambda_1 \wedge \neg \lambda_2)$, and $\phi^L = (\bigwedge_{m=3}^9 \Diamond \lambda_m \wedge ((\text{Found_T})_m \rightarrow \Diamond \lambda_{13})) \wedge (\bigwedge_{n=10}^{12} \lambda_n \wedge ((\text{Found_T})_m \rightarrow \Diamond \lambda_{13}))$, where the proposition $(\text{Found_T})_m$ is **true** whenever a target of interest is present in the ROI λ_m . These ROIs are indicated in Fig. 9. Informally, this specification requires the vehicles to visit ROIs $\lambda_3, \dots, \lambda_{12}$ 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 $N_A = 6$ 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 $\lambda_6, \dots, \lambda_9, \lambda_{11}$, and λ_{12} . Vehicles detect the target whenever they visit these ROIs.

The proposed algorithm results in the local specifications $\psi_1 := \Diamond(\lambda_5 \wedge \mu_5 \wedge \Diamond(\lambda_8 \wedge \mu_8 \wedge \Diamond(\lambda_{11} \wedge \mu_{11})))$, $\psi_2 := \Diamond(\lambda_{10} \wedge \mu_{10} \wedge \Diamond(\lambda_3 \wedge \mu_3))$, $\psi_4 := \Diamond(\lambda_{12} \wedge \mu_{12})$, $\psi_3 := \Diamond(\lambda_4 \wedge \mu_4 \wedge \Diamond(\lambda_6 \wedge \mu_6 \wedge \Diamond(\lambda_7 \wedge \mu_7 \wedge \Diamond(\lambda_9 \wedge \mu_9))))$, $\psi_5 := \Diamond(\lambda_{12} \wedge \mu_{12} \wedge \Diamond(\lambda_{11} \wedge \mu_{11}))$, $\psi_6 := \Diamond(\lambda_{10} \wedge \mu_{10})$, where $\mu_m := (\text{Found_T})_m \rightarrow \Diamond \lambda_{13}$. The resultant vehicle routes are indicated in Fig. 9.

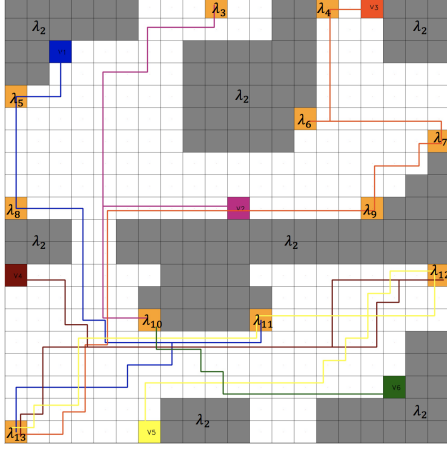


Fig. 9. Routes of all vehicles in Ex. 4: vehicles 1–6 indicated in blue, magenta, red, maroon, yellow, and green respectively.

The sum of lengths of vehicle routes from the proposed algorithm is 241 units, whereas that resulting from E-CBGA is 181 units. However, the total waiting duration resulting from the proposed algorithm is 14 time units, whereas that resulting from E-CBGA is 42 units. $\square$

Example 5. Here $N_A = 8$ and $N_R = 15$ for the UCPM. The fixed communication network among vehicles is shown below. The global specification is $\phi = \phi^S \wedge \phi^L$ with ϕ^S as in Ex. 4 and $\phi^L := (\wedge_{m=3}^7 \lambda_m) \wedge (\diamond \lambda_8 \wedge ((\text{Found_T})_8 \rightarrow \diamond \lambda_{15})) \wedge (\wedge_{n=9}^{11} \lambda_n) \wedge (\wedge_{k=12}^{14} (\diamond \lambda_k \wedge ((\text{Found_T})_k \rightarrow \diamond \lambda_{15})))$ with $M_9 = 2$ and $M_i = 3$ for $i = 10, \dots, 14$.

	$i = 1$	2	3	4
$\mathcal{N}_i$	{3, 4, 7}	{4, 5, 8}	{1, 5, 6}	{1, 2, 8}
	$i = 5$	6	7	8
$\mathcal{N}_i$	{2, 3, 7}	{3, 7, 8}	{1, 5, 6}	{2, 4, 6}

Similar to Ex. 4, this specification² also encodes vehicle behavior conditional on the presence of some target object (in ROIs λ_8 and λ_{12}). The vehicle routes are shown in Fig. 10. 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 reverse order (i.e., λ_{11} first, then λ_{10}) which leads to infinite waiting time for these three vehicles. This situation occurs because E-CBGA has no temporal considerations. $\square$

Another simulation example, which we denote **Example 6**, with $N_A = 10$ and $N_T = 13$ is provided in (Fang,

² The detailed local liveness specifications are available in (Fang, 2020).

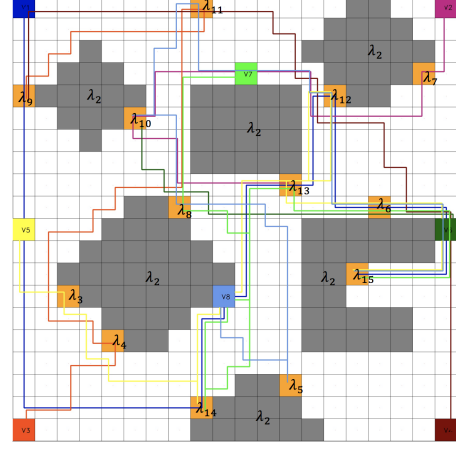


Fig. 10. Ex. 5: routes of vehicles 1–8 indicated in dark-blue, magenta, red, maroon, yellow, dark-green, light-green, and light-blue respectively.

Table 2

Route lengths and waiting durations due to the proposed algorithm as compared to E-CBGA.

	Redn. in waiting	Inc. in length	Num. iter.
Ex. 1	77.78%	3.191%	6
Ex. 2	58.33%	17.60%	13
Ex. 3	45.45%	-5.98%	13
Ex. 4	66.67%	33.15%	11
Ex. 5	N/A	N/A	14
Ex. 6	97.68%	12.55%	16

2020) but is omitted here for brevity.

The primary advantage of the proposed algorithm over E-CBGA is that the total waiting duration of the team of agents is significantly reduced. More importantly, in certain cases such as Ex. 5, the E-CBGA may not even produce a feasible solution as it does not consider temporal synchronization. Table 2 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.

By Prop. 1, the proposed algorithm terminates in at most $\frac{1}{2}(N_{TC} + 1)N_{TC}\delta$ iterations. It is known that the CBBA, which we use to assign independent tasks, terminates in at most $N_{TT}\delta$ iterations (Choi et al., 2009). Table 2 shows the numbers of iterations required for termination in each of the preceding examples.

3.2 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 LCM library (Huang et al., 2010) to pass messages using the unidirectional protocol. Whereas all agents communicated over the same *physical* LAN, inter-agent communications were restricted to emulate a fixed communication network with $\mathcal{N}_1 = \{2\}$, $\mathcal{N}_2 = \{1, 3\}$, and $\mathcal{N}_3 = \{2\}$. A uniform 12×12 workspace cell decomposition was considered. The specification is $\phi = \phi^S \wedge \phi^L$, where $\phi^S := \Box(\lambda_0 \wedge \neg \lambda_1) \wedge (\Diamond \Box \lambda_2)$, and $\phi^L := \Diamond \lambda_3 \wedge \Diamond \lambda_4 \wedge \Diamond \lambda_5 \wedge \Diamond \lambda_6$.

The proposed algorithm was successfully executed, and the following local liveness specifications were computed: $\psi_1 = \Diamond \lambda_3$, $\psi_2 = \Diamond(\lambda_5 \wedge \Diamond \lambda_6)$, and $\psi_3 = \Diamond(\lambda_4 \wedge \Diamond \lambda_6)$. These specifications were computed after four iterations of bundle construction and consensus of the proposed algorithm. The sum of the lengths of vehicle routes was 71 units and the total waiting duration was 1 time unit (for agent 3). A video demonstrating the progress of this decentralized implementation is available at <https://youtu.be/LK0Vzu36PwI>. The Raspberry Pi B+ SBC is readily compatible with several mobile robot platforms.

4 Conclusions

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 is shown to be superior to alternative approaches in the existing literature, and achieves task assignments with minimal waiting durations.

Acknowledgment

This work was supported by award #FA9550-17-1-0028 from the US Air Force Office of Scientific Research.

References

- C. Belta, B. Yordanov, and E. A. Gol. *Formal methods for discrete-time dynamical systems*. Springer International Publishing, 2017.
- 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*, pages 2718–2723, Dec. 2011.
- H. L. Choi, L. Brunet, and J. P. How. Consensus-Based decentralized auctions for robust task allocation. *IEEE Trans. Rob.*, 25(4):912–926, Aug. 2009.
- R. V. Cowlagi and Z. Zhang. Route guidance for satisfying temporal logic specifications on aircraft motion. *Journal of Guidance, Control, and Dynamics*, 40(2):390–401, 2017. doi: <https://doi.org/10.2514/1.G001829>.
- A. Duret-Lutz. Manipulating LTL formulas using Spot 1.0. In *Proceedings of the 11th International Symposium on Automated Technology for Verification and Analysis (ATVA'13)*, volume 8172 of *Lecture Notes in Computer Science*, pages 442–445, Hanoi, Vietnam, Oct. 2013. Springer. doi: 10.1007/978-3-319-02444-8_31.
- E. A. Emerson. Temporal and modal logic. In *Formal Models and Semantics*, pages 995–1072. Elsevier, 1990.
- J. Fang. *Interactive Route-planning and Mobile Sensing with a Team of Multiple Robotic Vehicles to Satisfy Linear Temporal Logic Specifications*. PhD thesis, Worcester Polytechnic Institute, Worcester, MA, USA, August 2020. URL <https://digital.wpi.edu/show/8336h474t>.
- J. Fang, Z. Zhang, and R. V. Cowlagi. Decentralized route-planning to satisfy global linear temporal logic specifications on multiple aircraft. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Kissimmee, FL, USA, January 8–12 2018.
- 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)*, pages 6235–6240, Dec. 2012.
- M. Guo and D. V. Dimarogonas. Multi-agent plan reconfiguration under local LTL specifications. *Int. J. Rob. Res.*, 34(2):218–235, Feb. 2015.
- M. Guo and D. V. Dimarogonas. Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks. *IEEE Trans. Automation Science and Engineering*, 14(2):797–808, 2017.
- M. Guo, J. Tumova, and D. V. Dimarogonas. Communication-free multi-agent control under local temporal tasks and relative-distance constraints. *IEEE Trans. Automatic Control*, 61(12):3948–3962, 2016.
- 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*, pages 4057–4062. IEEE, 2010.
- 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.*, 6(3):338–350, Apr. 2014.
- Y. Kantaros and M. M. Zavlanos. Distributed optimal control synthesis for multi-robot systems under global temporal tasks. *Proceedings of the 9th ACM/IEEE*, 2018a.
- Y. Kantaros and M. M. Zavlanos. STyLuS*: A temporal logic optimal control synthesis algorithm for Large-Scale Multi-Robot systems. *CoRR*, 2018b.
- 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)*, pages 2519–2524, Dec. 2018c.
- S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning with deterministic μ -calculus specifications. In *2012 American Control Conference (ACC)*, pages 735–742, June 2012.
- M. Karimadini and H. Lin. Synchronized task decomposition for two cooperative agents. In *2010 IEEE Conference on Robotics, Automation and Mechatronics*, pages 368–373, June 2010.
- M. Kloetzer and C. Belta. Temporal logic planning and con-

- trol of robotic swarms by hierarchical abstractions. *IEEE Transactions on Robotics*, 23(2):320–330, 2007.
- M. Kloetzer and C. Mahulea. Multi-robot path planning for syntactically co-safe ltl specifications. In *2016 13th International Workshop on Discrete Event Systems (WODES)*, pages 452–458, 2016.
- 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*, volume 1, pages 153–158 Vol.1, Dec. 2004.
- 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)*, pages 17–23, Dec 2017. doi: 10.1109/MRS.2017.8250926.
- M. Rungger and M. Zamani. Scots: A tool for the synthesis of symbolic controllers. In *Hybrid Systems Computation and Control*, 2016.
- P. Schillinger, M. Bürger, and D. V. Dimarogonas. Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. *The Int. J. Rob. Res.*, 37(7):818–838, 2018a.
- P. Schillinger, M. Bürger, and D. V. Dimarogonas. Decomposition of finite ltl specifications for efficient multi-agent planning. In *Distributed Autonomous Robotic Systems*, pages 253–267. Springer, 2018b.
- 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 Annual Conference on Decision and Control (CDC)*, pages 1132–1137, Dec 2017. doi: 10.1109/CDC.2017.8263808.
- D. Stavrou, S. Timotheou, C. G. Panayiotou, and M. M. Polycarpou. Optimizing container loading with autonomous robots. *IEEE Transactions on Automation Science and Engineering*, 15(2):717–731, April 2018. ISSN 1545-5955. doi: 10.1109/TASE.2017.2679485.
- P. Toth and D. Vigo. *The vehicle routing problem*. SIAM, Philadelphia, USA, 2002.
- J. Tumova and D. V. Dimarogonas. Multi-agent planning under local LTL specifications and event-based synchronization. *Automatica*, 70:239–248, 2016.
- 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*, 32(8):889–911, 2013.
- C. I. Vasile and C. Belta. Sampling-based temporal logic path planning. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.
- M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas. A distributed auction algorithm for the assignment problem. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 1212–1217, Dec. 2008.

Coupled Sensor Configuration and Path-Planning in Unknown Static Environments

Chase St. Laurent[†] and Raghvendra V. Cowlagi,^{*} *Senior Member, IEEE*

Abstract—We consider path-planning for a mobile agent in an unknown environment to be mapped by a sensor network, where the location and field of view of each sensor can be configured. To solve this problem we propose a coupled sensor configuration and path-planning (CSCP) iterative method, which finds an optimal sensor configuration (location and FoV) at each iteration, applies Gaussian process regression to construct a threat field estimate, and then finds a candidate optimal path with minimum expected threat exposure. We define a so-called task-driven information gain (TDIG) metric, the maximization of which provides sensor configurations. The TDIG quantifies the notion of acquiring sensor data of “most relevance” to path-planning. The CSCP iterations terminate when the path cost variance reduces below a prespecified threshold. Through numerical simulations we demonstrate that the CSCP algorithm finds near-optimal paths with significantly fewer sensor measurements compared to traditional methods.

I. INTRODUCTION

We consider path-planning for a mobile agent in an unknown environment comprised of threats. The threat is modeled as a scalar field and may be estimated using noisy observations made by a sensor network. Traditionally, the approach to solving such problems relies on the separation of path-planning and sensing/estimation. This separation, which arises from the separation principle in linear feedback control, may result in significantly suboptimal planning. We propose a *coupled* planning and sensing method, and demonstrate its superiority over the separated method.

This problem is inspired by the growing applications of heterogeneous networks of robotic vehicles in military and civilian operations. One such application is finding a safe path for a terrestrial vehicle in the aftermath of a natural disaster, while a team of aerial vehicles visually surveys the environment for damage, as illustrated in Fig. 1.

We are interested in the problem of finding optimal sensor *configurations* that can provide information of most relevance to the path-planning problem. Configurations are adjustable parameters of the sensors. In the aforementioned application context, each aerial vehicle carries an electro-optical or infrared (EO/IR) camera sensor, and its configurations include location in the environment and the altitude from which images are recorded. Increasing altitude provides the camera a larger field of view (FoV), but degrades the image resolution, which in turn makes the sensor noisier.

Related Work: Placement of sensors that take pointwise measurements of a scalar field is a well-studied area of direct relevance to this work. Optimal sensor placement is studied

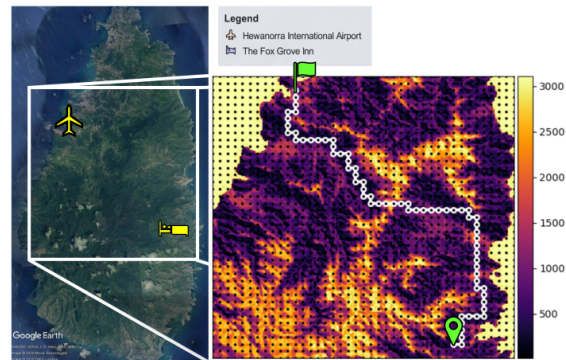


Fig. 1. Example workspace and true optimal path in a disaster relief application example (see §II for details).

for estimation of gaseous plumes [1], volcanic ash [2], using distributed sensor networks [3]–[5]. Guidance strategies for cooperative mobile sensors are reported for estimating processes governed by partial differential equations [6]–[8] or for target tracking [9]–[13]. The swarm robotics literature addresses coordination among multiple intelligent sensors with a bio-inspired hive-mind approach [14], [15]. The performance of mobile sensor placement is quantified using metrics such as the Fisher information matrix, entropy, Kullback-Leibler divergence, maximal information coefficient, mutual information, and least squared error [16], [17].

Nonparametric modeling of spatiotemporal processes using Gaussian process regression (GPR) is well-known [18]–[20]. GPR is a supervised machine learning technique, which is used in many applications such as modeling gas distributions in an environment [21] and cooperative perception and detection of forest fires [22]. GPR-based techniques for spatial process estimation with mobile sensor networks and near-optimal sensor placement are reported [17], [23].

A distinction between *task-driven* and information-driven performance metrics is reported [24], which notes that a *task-driven* metric (e.g. squared tracking error) benefits sensor placement. Such task-driven sensor placement is recently reported [25] for coupled sensor placement and path-planning in an unknown environment, where near-optimal path-planning is achieved with significantly fewer sensor measurements compared to information-driven methods.

The primary limitation of all of these works is that pointwise measurements of the scalar field are assumed. In applications such as thermal imaging of a forest fire, sensor measurements are area-based, and within that area many observations are made using computer vision techniques such as feature recognition. A pointwise measurement assumption

[†]Mechanical Engineering Department.

^{*}Aerospace Engineering Department, Worcester Polytechnic Institute, Worcester, MA, USA. {cstlaurent, rvcowlagi}@wpi.edu.

forces the aggregation of these observations, e.g., maximum or average value over the measured area, which causes data loss even before an estimation algorithm is employed.

Another well-researched area of some relevance to this paper is path-planning in unknown or uncertain environments based on exploration of the environment using onboard (proprioceptive) sensors, cf. [26], [27]. The proposed work addresses exteroceptive sensors.

Statement of Contributions: In this paper, we propose an iterative method that solves a coupled sensor configuration and path-planning problem using GPR nonparametric regression. This method finds an optimal sensor configuration, i.e., sensor location and FoV at each iteration, applies GPR to construct a threat field estimate, and then finds a candidate optimal path with minimum threat exposure. Crucially, the sensor configuration is coupled with path-planning in that it seeks to minimize the path cost variance. The iterations terminate when the path cost variance reduces below a prespecified threshold.

A crucial GPR step is the selection of an appropriate kernel for the domain. Methods to automatically discover ideal kernels are discussed in the literature [28], [29]. Similarly, GP hyperparameters define characteristic length scales, which affect regression performance [20]. Techniques for improving optimization speed of these hyperparameters are available, such as Rprop [30] and Bayesian regularization [31].

There are two main novel contributions of this work. First, we provide a method to configure sensors in the environment for the general case where observations are taken in a finite area rather than pointwise, without lossy aggregation of the observations. To the best of our knowledge, no comparable method is available in the literature. Furthermore, we consider two configurations for each sensor, namely, its location and FoV (which may be altitude-dependent). We explicitly model sensor noise dependence on FoV to emulate commonly used EO/IR sensors. By doing so, we quantify the “exploration v/s exploitation” trade-off, where “exploration” is achieved through large FoV (noisy measurements over large areas) and “exploitation” is achieved through small FoV (precise measurements over small areas).

Second, we demonstrate that the proposed iterative and coupled exteroceptive sensor configuration and path-planning method achieves near-optimal paths using fewer observations by *orders of magnitude* compared to traditional methods. The proposed method allows for a user-specified environment map resolution and a user-specified path cost variance threshold. Decreasing the termination threshold ensures lower path cost error at the expense of additional observations.

This paper is organized as follows. We provide a problem formulation in §II, the details of the proposed method in §III, numerical simulation results in §IV, and conclusions in §V.

II. PROBLEM FORMULATION

We denote by $\mathbb{R}$ and $\mathbb{N}$ the sets of real and natural numbers, respectively, and by $[N]$ the set $\{1, 2, \dots, N\}$ for any $N \in \mathbb{N}$. For any $\mathbf{a} \in \mathbb{R}^N$, $\mathbf{a}[i]$ is the i^{th} element of $\mathbf{a}$ and $\text{diag}(\mathbf{a})$ denotes the $N \times N$ diagonal matrix with

the elements of $\mathbf{a}$ on the principal diagonal. For any matrix $A \in \mathbb{R}^{M \times N}$, $A[i, j]$ is the element in the i^{th} row and j^{th} column. $\mathbf{I}_{(N)}$ denotes the identity matrix of size N .

The agent operates in a prespecified closed square region called the *workspace* $\mathcal{W}$, which belongs to an *environment* $\mathcal{E}$ such that $\mathcal{W} \subset \mathcal{E} \subset \mathbb{R}^2$. Consider a uniformly-spaced grid of points labeled by integers $i = 1, 2, \dots, N_g$. Consider a graph $\mathcal{G} = (V, E)$ whose vertices $V = [N_g]$ are uniquely associated with these grid points, and whose edges E consist of pairs of geometrically adjacent grid points. In a minor abuse of notation, we label the vertices the same as grid points. We denote by $\mathbf{p}_i = (p_{ix}, p_{iy})$ the coordinates of the i^{th} grid point and by Δp the distance between adjacent grid points.

A *threat field* $c : \mathcal{E} \rightarrow \mathbb{R}_{>0}$ is a strictly positive temporally static scalar field. We are interested in a path-planning problem of minimizing the agent’s threat exposure. A *path* $\pi = (\pi[0], \pi[1], \dots, \pi[A])$ between prespecified initial and goal vertices $i_{\text{start}}, i_{\text{goal}} \in V$ is a finite sequence, without repetition, of successively adjacent vertices such that $\pi[0] = i_{\text{start}}$ and $\pi[A] = i_{\text{goal}}$ for some $A \in \mathbb{N}$. We define the *path incidence vector* $\mathbf{v}_\pi \in \mathbb{R}^{N_g}$ such that $\mathbf{v}_\pi[i] = 1$ if $i = \pi[j]$ for $j \in [A] \setminus 0$ and $\mathbf{v}_\pi[i] = 0$ otherwise.

The *cost* of this path π is the total threat exposure along the path: $\mathcal{J}(\pi) := \Delta p \sum_{j=1}^A c(\mathbf{p}_{v_j})$. The main problem of interest is to find a path π^* of minimum cost.

We cannot solve this problem as stated because the threat field is unknown. The threat field can be observed by a network of $N_s \in \mathbb{N}$ sensors. Each of these sensors measures the threat in a circular field of view (FoV), which is a subregion of $\mathcal{E}$ as shown in Fig. 2(a). The center and radius of this circular FoV $\mathcal{S}_k \subset \mathcal{E}$, denoted $\mathbf{s}_k \in \mathcal{W}$ and $\varrho_k \in \mathbb{R}_{>0}$ for the k^{th} sensor, are parameters that we may choose for each $k \in [N_s]$. Maximum and minimum FoV radius constraints are specified as $\varrho^{\max}$ and $\varrho^{\min}$, respectively. The set of all sensor parameters is called a *configuration*, which we denote by $\mathbf{C} = \{\mathbf{s}_1, \varrho_1, \mathbf{s}_2, \dots, \varrho_{N_s}\}$.

Consider the set $V_S := \{\cup_{k \in [N_s]} \mathcal{S}_k \cap V\}$ of vertices with grid points within the union of FoVs of all sensors. We define the *sensor cover incidence vector* $\boldsymbol{\nu} \in \mathbb{R}^{N_g}$ such that $\boldsymbol{\nu}[i] = 1$ if $i \in V_S$ and $\boldsymbol{\nu}[i] = 0$ otherwise. Next, we define the *reduction incidence vector* $\mathbf{r}_\pi := (\boldsymbol{\nu} \& \mathbf{v}_\pi)$, i.e., $\mathbf{r}_\pi[i] = 1$ if and only if $\boldsymbol{\nu}[i] = 1$ and $\mathbf{v}_\pi[i] = 1$ for each $i \in [N_g]$.

As illustrated in Fig. 2(a), within $\mathcal{S}_k$ the k^{th} sensor takes $M_k \in \mathbb{N}$ pointwise and noisy measurements of the threat field. In this paper, we prespecify M_k , future work may consider M_k is proportional to ϱ_k^2 . The spatial points $\mathbf{x}_{km} \in \mathcal{S}_k$, for $m = 1, 2, \dots, M_k$, where these measurements are taken are uniformly distributed within the subregion $\mathcal{S}_k$, illustrated by red dots in Fig. 2(a). Each of these measurements is:

$$z_{km} = c(\mathbf{x}_{km}) + \eta_{km}, \quad k \in [N_s], \quad m \in [M_k].$$

The i.i.d measurement error $\eta_{km} \sim \mathcal{N}(0, \sigma_k^2)$ is normally distributed with $\sigma_k^2 := \exp((\pi \varrho_k^2)/|\mathcal{E}|) - 1$, where $|\mathcal{E}|$ denotes area. We denote by $\mathbf{z} = (z_{11}, \dots, z_{1M_1}, \dots, z_{N_s M_{N_s}})$ the measurements made by all sensors.

We may use the sensor observations to construct a stochastic estimate of the threat field, and in turn, use this estimate

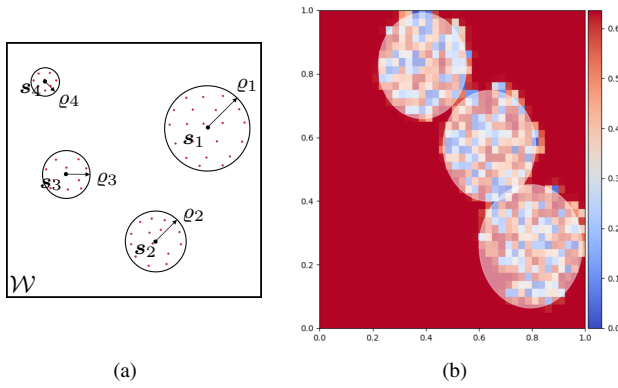


Fig. 2. (a) Illustration of sensor configuration. (b) Sensor placement, coverage (translucent circles), and error covariance matrix P_0 (colors indicate diagonal values) in the St. Lucia example.

to find an optimal path that minimizes the *expected* cost. Furthermore, we would like the uncertainty in the estimated path cost to be low, e.g., we may require that the variance of the path cost be less than a certain prespecified threshold. To do so, we must collect a sufficient number of measurements, which may be achieved by repeatedly changing the sensor configuration over multiple iterations. Conceptually, at each iteration $\ell = 1, 2, \dots$, the sensor configuration C_ℓ^* is chosen, the threat field estimate is updated using the new measurements, and an optimal path may be computed. The main problem of interest is now reformulated as follows.

Problem 1. *Over a finite number of iterations $\ell = 1, 2, \dots, L$, find sensor configurations C_ℓ and a path π^* of minimum expected cost $\bar{\mathcal{J}}^* := \mathbb{E}[\mathcal{J}(\pi^*)]$ that satisfies $\mathbb{E}[(\mathcal{J}(\pi^*) - \bar{\mathcal{J}}^*)^2] \leq \varepsilon$, for a prespecified threshold $\varepsilon \in \mathbb{R}_{>0}$.*

We illustrate this problem by a fictional disaster scenario on the island of St. Lucia shown in Fig. 1. In this scenario, major roads are flooded and the problem is of finding a path to rescue survivors trapped in one location (nominally, the Fox Grove Inn indicated at the lower right in Fig. 1) by transporting them to a safe location (nominally, an airport indicated at the upper left in Fig. 1). A network of aerial vehicles is available to visually survey the extent of flooding and enable safe transport of the survivors through the least flooded areas. In this scenario, the time required to collect survey data must be kept to a minimum before finalizing the transport path, to which end the path-planning problem must be solved with as few surveying observations as possible.

III. COUPLED SENSOR CONFIGURATION AND PATH-PLANNING (CSCP)

The proposed method for solving Problem 1 is described by the iterative CSCP algorithm in Fig. 5. In this section, we provide the details of the steps in this algorithm and its theoretical properties. Each iteration involves three main steps: finding an optimal sensor configuration (locations and FoVs of sensors), finding GP hyperparameters, updating the threat field estimate, and finding the optimal path.

At each iteration $\ell = 0, 1, 2, \dots$, the algorithm maintains a pointwise estimated mean threat at each of the N_g grid points estimate of the threat field $\mathbf{f}_\ell \in \mathbb{R}^{N_g}$ and an estimation error covariance matrix $P_\ell \in \mathbb{R}^{N_g \times N_g}$. The expected cost of any path π is then calculated as:

$$\bar{\mathcal{J}}_\ell(\pi) = \mathbb{E}[\mathcal{J}(\pi)] = \Delta p \mathbf{f}_\ell^\top \mathbf{v}_\pi. \quad (1)$$

At each iteration, an optimal path π_ℓ^* with minimum expected cost is computed using Dijkstra's algorithm, which has computational complexity $\mathcal{O}(|V| \log |E|)$ and is efficient even for graphs large numbers of vertices and edges [32].

The path cost variance at iteration ℓ is calculated as:

$$\text{Var}_\ell(\pi) := \mathbb{E}[(\mathcal{J}(\pi) - \bar{\mathcal{J}}_\ell(\pi))^2] = (\Delta p)^2 \mathbf{v}_\pi^\top P_\ell \mathbf{v}_\pi. \quad (2)$$

A. Algorithm Initialization

The algorithm initializes “optimistically” by setting $\mathbf{f}_0 = \mathbf{0}$. The initial uncertainty in the threat estimate is quantified by initializing the estimation error covariance matrix with uniformly large values, e.g., $P = \chi \mathbf{I}_{(N_g)}$, where $\chi \gg 1$ is an arbitrary large number. Due to this “optimistic” initialization, the initial optimal path π_0^* is of minimum length, indicated by the green-colored path in Fig. 3(a).

B. Sensor Configuration

At each iteration $\ell = 0, 1, \dots$, we pose the sensor configuration problem as maximization of a *task-driven information gain (TDIG)*, which in turn we define as the reduction in cost variance of the path π_ℓ^* . We denote a sensor configuration at iteration ℓ by $C_\ell = \{s_{1\ell}, \varrho_{2\ell}, s_{2\ell}, \dots, \varrho_{N_s\ell}\}$. To define the TDIG, we must find an approximation to the posterior threat estimation error covariance due to sensor measurements. To this end, first consider the correlation matrix $\Gamma_\ell := D_\ell^{-1} P_\ell D_\ell^{-1}$, where D_ℓ is a diagonal matrix with $D_\ell[i, i] = \sqrt{P_\ell[i, i]}$. We define a reduction factor

$$\delta(C_\ell; i) := \sum_k^{N_s} \left(\sigma_k^2 \exp \left(-\frac{1}{2\varrho_k^2} (\mathbf{p}_i - \mathbf{s}_k)^\top (\mathbf{p}_i - \mathbf{s}_k) \right) \right)^{-1},$$

for each $i \in [N_g]$. Due to the reduction factor $\delta(C_\ell; i)$, the expected reduction in $P_\ell[i, i]$ reaches a maximum at each sensor location and decays elsewhere in the sensor FoV. Next, we define $Q_\pi \in \mathbb{R}^{N_g \times N_g}$ as

$$Q_\pi := \begin{cases} \delta(C_\ell; i)^{-1}, & \text{if } \mathbf{r}_\pi[i] = 1, \\ 0 & \text{otherwise,} \end{cases}$$

and we estimate the posterior diagonal variances as $\mathbf{q}_\pi[i] = D_\ell(1 - \mathbf{r}_\pi) + Q_\pi \mathbf{r}_\pi$. The posterior covariance matrix is then approximated as $P_\ell^- := \text{diag}(\sqrt{\mathbf{q}_\pi}) \Gamma_\ell \text{diag}(\sqrt{\mathbf{q}_\pi})$. Finally, we define the TDIG as:

$$h(C_\ell) := (\Delta p)^2 \mathbf{v}_\pi^\top (P_\ell - P_\ell^-) \mathbf{v}_\pi \quad (3)$$

The sensor configuration problem is of finding a configuration C_ℓ^* that maximizes the TDIG subject to the constraints

$$\mathbf{s}_k \in \mathcal{W}, \quad \varrho_k^{\min} \leq \varrho_k \leq \varrho_k^{\max}, \quad \text{for each } k \in [N_s].$$

An example of sensor placement coverage and the estimation error covariance P_0 for the initial iteration of the St.

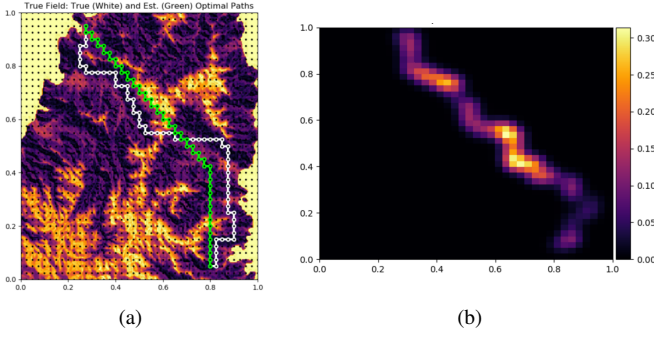


Fig. 3. St. Lucia example: (a) True threat field, true optimal path (white), and the initial path π_0^* (green). (b) Importance function at $\ell = 6$.

Lucia disaster scenario example with $N_s = 3$ and $M_k = 100$ for each $k = 1, 2, 3$, are shown in Figure 2(b).

This TDIG explicitly couples path-planning and sensor configuration, and we demonstrate in §IV that this information gain definition provides significant reductions in the number of observations required to find near-optimal paths. To the best of our knowledge this TDIG lacks convenient properties such as convexity or submodularity. Therefore, to find optimal sensor configurations that maximize this TDIG, in general a numerical optimization approach such as differential evolution [33] may be used. To reduce computational time, we propose the following heuristic method to find an initial guess for the numerical optimization.

We define an *importance function* $\alpha : V \rightarrow \mathbb{R}^{N_g}$ as

$$\alpha(i) := \sum_{n=1}^A P_{\ell-1}[n, n] \exp \left(-\frac{\|\mathbf{p}_i - \mathbf{p}_{\pi_{\ell-1}^*}[n]\|^2}{2(N_g - 1)^2} \right), \quad (4)$$

for $i \in [N_g]$, where A is the number of vertices in the path $\pi_{\ell-1}^*$. The importance function ranks grid points with high threat estimation error, and assigns importance to grid points close to the currently known optimal path $\pi_{\ell-1}^*$, as indicated in Fig. 3(b). The normalizing denominator in (4) scales at various grid resolutions the individual importance of each grid point. We find the N_s grid points $i_1^*, i_2^*, \dots, i_{N_s}^* \in [N_g]$ with the largest values of $\alpha(i_k^*)$. The following sensor configuration is then used as an initial guess for a numerical optimization algorithm to maximize the TDIG: $\mathbf{C}_{\text{guess}} := \{\mathbf{p}_{i_1^*}, \lambda, \mathbf{p}_{i_2^*}, \lambda, \dots, \mathbf{p}_{i_{N_s}^*}, \lambda\}$, where λ is arbitrarily small. In other words, the initial guess is of sensor locations at grid points with the N_s largest values of α , and arbitrarily small FoV. We use differential evolution to numerically optimize the TDIG and find the configuration $\mathbf{C}_{\ell}^*$.

C. GPR-based Field Estimation

Gaussian process regression (GPR) is a supervised machine learning technique that uses function values at training points $\mathbf{X}$ to estimate function values at test points $\mathbf{X}_*$ [34]. In the present context, for a particular sensor configuration $\mathbf{C}_{\ell}$, an updated threat estimate $\mathbf{f}_{\ell}$ and error covariance P_{ℓ} are calculated using GPR as follows. In accordance with the “optimistic” field assumption, we perform the GPR calculations assuming mean $\mathbf{0}$ because stationary kernels exhibit

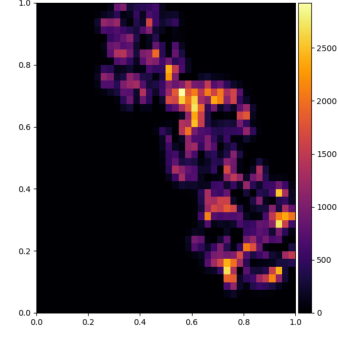


Fig. 4. Threat field estimate at the first iteration in the St. Lucia example.

$\kappa(\mathbf{x}, \mathbf{x}_*) \rightarrow 0$ as $\|\mathbf{x} - \mathbf{x}_*\| \rightarrow \infty$. Recall from §II that this sensor configuration provides $N_{\text{meas}} := \sum_{k=1}^{N_s} M_k$ measurements. We denote by $\mathbf{X} = \{\mathbf{x}_{11}, \dots, \mathbf{x}_{km}, \dots, \mathbf{x}_{N_s M_{N_s}}\}$ the set of locations where these measurements are taken. The associated measurement error covariance matrix is $\mathbf{R} := \text{diag}(\sigma_1^2, \sigma_1^2, \dots, \sigma_k^2, \sigma_k^2, \dots, \sigma_{N_s}^2)$.

First, we define a matrix kernel $K \in \mathbb{R}^{N_{\text{meas}} \times N_{\text{meas}}}$ using anisotropic radial basis functions formulated as:

$$K = \kappa(\mathbf{X}, \mathbf{X}) := \gamma^2 \exp(-\frac{1}{2}(\mathbf{x}_{k_1 m_1} - \mathbf{x}_{k_2 m_2})^T \Lambda (\mathbf{x}_{k_1 m_1} - \mathbf{x}_{k_2 m_2})) \quad (5)$$

for $k_1, k_2 \in [N_s]$, $m_1 \in [M_{k_1}]$, and $m_2 \in [M_{k_2}]$. In (5), γ^2 and Λ are hyperparameters chosen by maximizing the marginal log-likelihood defined as:

$$\log p(\mathbf{z} | \mathbf{0}, K_{\mathbf{z}}) := -\frac{1}{2} \mathbf{z}^T K_{\mathbf{z}}^{-1} \mathbf{z} - \frac{1}{2} \log |K_{\mathbf{z}}| - \frac{N_{\text{meas}}}{2} \log 2\pi,$$

where $K_{\mathbf{z}} := K + \mathbf{R}$. We use the set of grid points $\mathbf{X}_* = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{N_g}\}$ as the GPR test points. Next, we define kernels K_* and K_{**} as:

$$K_* := \kappa(\mathbf{X}, \mathbf{X}_*), \quad K_{**} := \kappa(\mathbf{X}_*, \mathbf{X}_*).$$

The joint distribution of the observations $\mathbf{z}$ and the predicted values c_* at the test points $\mathbf{f}_*$ is given by [34]:

$$\begin{bmatrix} \mathbf{z} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left(\mathbf{0}, \begin{bmatrix} K_{\mathbf{z}} & K_* \\ K_*^T & K_{**} \end{bmatrix} \right).$$

The posterior mean and covariance are $\boldsymbol{\mu}_* = K_*^T K_{\mathbf{z}}^{-1} \mathbf{z}$, and $\boldsymbol{\Sigma}_* = K_{**} - K_*^T K_{\mathbf{z}}^{-1} K_*$. Using these posteriors, we calculate the threat estimate and error covariance by a weighted least squares update based on the previous estimate and error covariance, as follows:

$$\mathbf{f}_{\ell} := (P_{\ell-1} + \boldsymbol{\Sigma}_*)^{-1} (\boldsymbol{\Sigma}_* \mathbf{f}_{\ell-1} + P_{\ell-1} \boldsymbol{\mu}_*), \quad (6)$$

$$P_{\ell} := (P_{\ell-1}^{-1} + \boldsymbol{\Sigma}_*^{-1})^{-1}. \quad (7)$$

After finding the field estimate and error covariance, an optimal path π_{ℓ}^* is found using Dijkstra’s algorithm. For the St. Lucia example, the estimate $\mathbf{f}_1$ is illustrated in Fig. 4.

D. Algorithm Termination

After finding the optimal path π_{ℓ}^* , the path cost variance $\text{Var}_{\ell}(\pi_{\ell}^*)$ is computed per (2). If $\text{Var}_{\ell}(\pi_{\ell}^*) < \varepsilon$, then the

CSCP Algorithm

- 1: Initialization: set $\ell := 0$, find $\mathbf{f}_\ell$, P_ℓ , and π_ℓ^* per §III-A.
- 2: **while** $\text{Var}_\ell(\pi_\ell^*) < \varepsilon$ **do**
- 3: Find optimal sensor configuration C_ℓ^* per §III-B.
- 4: Record measurements $\mathbf{z}$.
- 5: Increment iteration counter $\ell := \ell + 1$.
- 6: Find GPR-based threat field estimate $\mathbf{f}_\ell$ and error covariance P_ℓ per §III-C.
- 7: Use Dijkstra's algorithm to find path π_ℓ^* with minimum expected cost $\bar{\mathcal{J}}_\ell(\pi_\ell^*)$.
- 8: **end while**

Fig. 5. Pseudocode for an iterative algorithm to solve Problem 1.

algorithm terminates. Here $\varepsilon > 0$ is an arbitrarily small user-specified threshold that indicates the desired confidence in the estimated path cost, namely, smaller the value of ε , higher is the desired confidence.

The proposed iterative algorithm is outlined in Fig. 5. The properties of the CSCP algorithm are summarized in the following technical results.

Proposition 1. *The CSCP algorithm terminates in a finite number of iterations $L \in \mathbb{N}$.*

Proof. We omit a formal proof due to lack of space, instead noting that this proof relies on iterative path cost variance reduction due to Eqn. (7). $\square$

Proposition 2. *The CSCP algorithm solves Problem 1.*

Proof. We note that the path π_L^* satisfies the conditions stated in Problem 1, namely, that it has minimum expected cost $\bar{\mathcal{J}}^* = \bar{\mathcal{J}}_L(\pi_L^*)$ and $\mathbb{E}[(\bar{\mathcal{J}}_L - \bar{\mathcal{J}}^*)^2] = \text{Var}_L(\pi_L^*) < \varepsilon$ per the termination criterion enforced by Line 2 in Fig. 5. $\square$

Corollary 1. *The path π_L^* is near-optimal in the following sense. Let $\mathcal{J}^*$ denote the cost of the true optimal path. Then:*

$$P[|\bar{\mathcal{J}}_L(\pi_L^*) - \mathcal{J}^*| \leq 3\varepsilon] \geq 0.9973.$$

Proof. Due to the GPR-based field estimation and the linearity in (1), the path cost is normally distributed, and the result follows immediately from the standard normal table. $\square$

IV. RESULTS AND DISCUSSION

In this section, we provide sample results of numerical simulations of the CSCP algorithm and its comparison with traditional information-maximizing approaches.

A. Illustrative Example 1: Fictional St. Lucia Scenario

In the St. Lucia disaster scenario example, we set the workspace resolution with $N_g = 41^2$, (a separation of approximately 0.4 km between adjacent grid points) number of sensors $N_s = 3$, and the termination threshold $\varepsilon = 1$. The evolution of the expected path cost $\bar{\mathcal{J}}_\ell(\pi_\ell^*)$ and its variance $\text{Var}_\ell(\pi_\ell^*)$ over the algorithm's iterations are shown in Fig. 6. The algorithm terminates after $L = 14$ iterations. The initial path (illustrated in Fig. 3(a)) is suboptimal compared to the

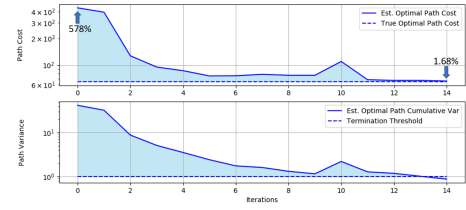


Fig. 6. St. Lucia example: (top) estimated path cost $\bar{\mathcal{J}}_\ell(\pi_\ell^*)$ with %suboptimality compared to the true optimal path cost, and (bottom) ratio $\text{Var}_\ell(\pi_\ell^*)/\varepsilon$ at all algorithm iterations ℓ .

TABLE I

SET OF PARAMETERS USED IN NUMERICAL PERFORMANCE ANALYSIS.

Parameter	Set of simulated values
# basis functions N_p	{8, 16, 32, 64, 128, 256, 512, 1024}
# sensors N_s	{1, 2, 4, 9, 16}
# grid points N_g	{11 ² , 21 ² , 31 ² , 41 ² , 51 ² }
# points in sensor FoV M_k	{4, 9, 16, 25}, for each $k \in [N_s]$
Termination threshold ε	{0.01, 0.1, 1.0}

true optimal by approximately 578%, but the final path is only 1.68% suboptimal. The threat field estimate $\mathbf{f}_L$ at the final iteration is shown in Fig. 7. Compared to the true field, notice that the estimated field remains granular (i.e. with high estimation error) in regions far away from the optimal path. Note also in Fig. 7 the high estimation error covariance (diagonal values) indicated for these “irrelevant” regions. The CSCP algorithm places sensors in areas of most relevance for the optimal path and does not “waste” sensor measurements to reduce the estimation error in regions farther away from the optimal path. The “optimistic” nature of the algorithm enables exploration of the entire workspace, as indicated by the occasional increases in $\text{Var}_\ell(\pi_\ell^*)$ seen in Fig. 6.

B. Performance Analysis on Randomly Generated Fields

We conducted a study with randomly generated threat fields in the series form $c(\mathbf{x}) = \sum_{n=1}^{N_p} \phi_n(\mathbf{x})$, where θ_n are randomly chosen coefficients, and ϕ_n are radial basis functions that cover $\mathcal{E}$. The number of bases N_p is indicative of the “richness” in spatial variations in the threat field: fields with small N_p have a few peaks and several flat regions whereas fields with very large N_p have closely spaced peaks, which may cause several flat regions.

One hundred trials with randomly chosen coefficients were conducted for each combination of the parameters listed in Table I, for a total of 240,000 simulations. In what follows we describe the impact of each of these parameters on the average path cost suboptimality and on the number of iterations required for termination. In §IV-C, we compare the performance of the CSCP algorithm in these simulations to traditional information-maximizing sensor placement. Some plots are omitted for lack of space, and are available in the following repository: github.com/Chase1325/CSCP_Public.

Suboptimality: The suboptimality of the paths resulting from the CSCP algorithm, averaged over all 240,000 simulations is 0.436%, which supports the claim that the CSCP

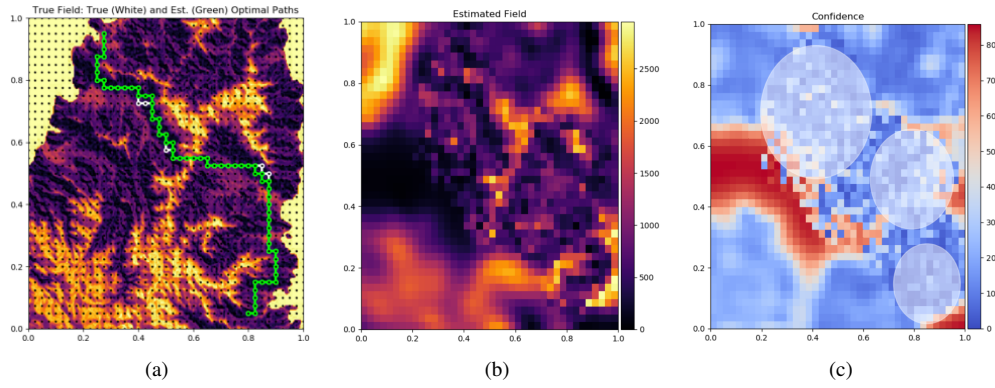


Fig. 7. St. Lucia example: (a) estimated optimal path π_L^* in green compared to true optimal in white, (b) threat estimate f_L and (c) diagonal elements of estimation error covariance and sensor placement at the final iteration.

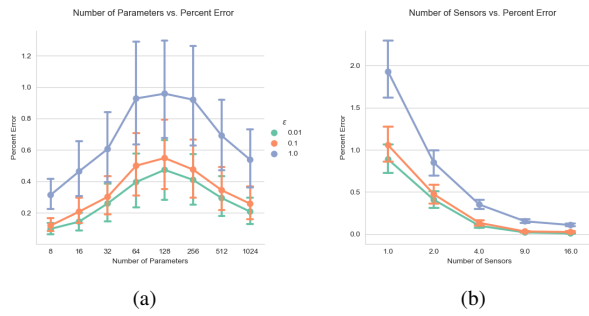


Fig. 8. Average suboptimality of the resultant path for varying N_p and N_s .

algorithm provides near-optimal paths. The impacts of the various parameters in Table I are as follows.

Figure 8(a) shows the impact of the number of basis functions N_p . The average suboptimality (percentage error averaged over all other parameters) remains below 1.0% for any value of N_p with $\epsilon = 1$ and below 0.5% with $\epsilon = 0.01$. For extreme values of $N_p \geq 512$ or $N_p \leq 16$, the average error decreases below 0.3% with $\epsilon = 0.01$.

Similarly, Fig. 8(b) shows the impact of the number of sensors N_s . Increasing the number of sensors rapidly decreases the overall error, and with $N_s \geq 4$ the suboptimality (percentage error averaged over all other parameters) lies below 0.5% for all values of ϵ . Even in the extreme case of $N_s = 1$, the suboptimality remains below 2% with $\epsilon = 1$ and below 1% with $\epsilon = 0.01$. In this context, we note that the average error decreases also with an increasing number of measurements M_k taken within a sensor FoV (plot omitted). Finally, the workspace resolution, i.e., the number of grid points N_g is found to have no significant impact on the average error when $\epsilon = 0.1$ or $\epsilon = 0.01$ (plot omitted).

Number of iterations: The average over all simulations of the number of iterations L required for the algorithm to terminate is 6.88, which supports the claim that the CSCP algorithm is computationally efficient.

Figure 9(a) shows that the number of iterations remains essentially constant with N_p , except in cases with small values of $N_p \leq 16$. This discrepancy is explained by

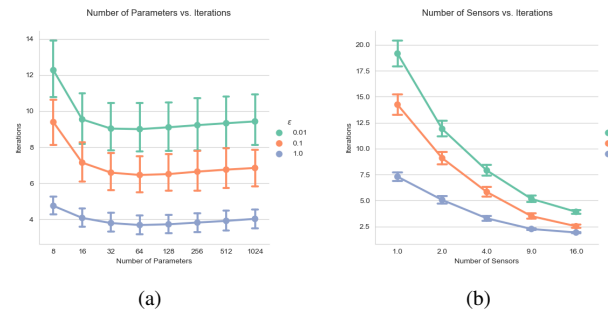


Fig. 9. Average number of algorithm iterations, for varying N_p and N_s .

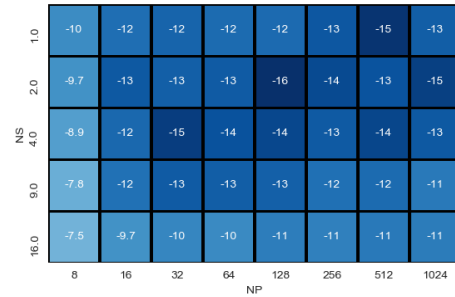


Fig. 10. Reduction in number of iterations due to the CSCP algorithm with $N_g = 51$, $M_k = 4$, and $\epsilon = 0.01$.

the structure of the threat field with small N_p , namely, several peaks separated by several flat regions, due to which there may be several possible paths with costs close to the optimum through different regions of the environment.

Figure 9(b) shows the intuitive result that the number of iterations rapidly decreases with increasing N_s . Similarly, the number of iterations also decreases with increasing values of M_k (plot omitted). With $N_s \geq 3$, an average of less than 10 iterations are required for the algorithm to terminate, for any $\epsilon \leq 1$. Finally, we note the number of iterations L increases with increasing N_g (plot omitted).

C. Comparisons with Information-Maximizing Approaches

In each of the numerical simulations described in §IV-B, we compared the result of the CSCP algorithm to an

information-maximizing (IM) sensor placement approach. This IM approach finds sensor configurations to maximize the reduction in $\text{tr}(P_{\ell-1})$. Compared to the proposed sensor placement based on the TDIG, the IM approach is entirely decoupled from the path-planning problem and attempts to estimate the field with uniformly low error in all spatial regions. We compared the number of iterations required by the CSCP algorithm to that required by the IM approach to terminate with an identical termination criterion, namely $\text{Var}_{\ell}(\pi_{\ell}^*) \leq \varepsilon$. Note that the total number of sensor measurements is $L \sum_{k=1}^{N_s} M_k$; i.e., the total number of measurements is proportional to the number of iterations

The average (over the aforesaid 240,000 simulations) number of iterations required by the proposed algorithm is 38.36% less than that required by the IM approach.

This means that the CSCP algorithm finds near-optimal paths with significantly fewer measurements. In certain cases, the number of measurements required is reduced by *an order of magnitude*. For example, Fig. 10 shows the average reduction in the number of iterations due to the CSCP algorithm for $N_g = 51$, $M_k = 4$, and $\varepsilon = 0.01$, with varying N_s and N_p . Most of these numbers are greater than 10, which indicates one order of magnitude reduction in the number of measurements, because the total number of sensor measurements is proportional to the number of iterations. Figure 10 indicates larger reductions with increasing N_p , i.e., in threat fields with “richer” spatial variations. Figure 10 also indicates that this advantage of the CSCP algorithm remains consistent regardless of the number of sensors. Finally, we note that the number of iterations decreases with increasing N_g (plot omitted; available at the aforesaid repository).

We did not compare the CSCP algorithm to sensor placement techniques that exploit submodularity of mutual information [17] because these techniques address pointwise measurements and do not consider a FoV with non-zero area.

D. Illustrative Example 2: Minefield

An application of the CSCP algorithm is finding a safe path through a minefield. Ground penetrating radar sensors for detecting improvised explosive devices have the promise of being able to detect explosives buried out of sight. The United States Army has developed the Airobotics Optimus drone capable of performing such a task [35]. To emulate this application, we consider a threat field as illustrated in Fig. 11 (top left image). The salient features of this threat field are several small regions (“mines”) with extremely large threat values. To execute the CSCP algorithm on this field, we set $N_g = 41^2$, $N_s = 5$ and $M_k = 16$ for each $k \in [5]$.

The threat field estimates, optimal paths, and error covariances (diagonal values) resulting from the CSCP algorithm and the IM approach are shown in Fig. 11. Both approaches find optimal “mine-free” paths, but the CSCP algorithm requires half as many iterations (i.e., measurements) to do so. Note in Fig. 11 that the CSCP algorithm avoids unnecessarily exploring much of the top left region indicated by high estimation error, and results in a field estimate that is highly accurate in some regions and highly inaccurate in others. By

contrast, the IM approach attempts to estimate the field with uniformly low estimation error in all regions, due to which it requires twice as many measurements.

V. CONCLUSIONS

In this paper we propose a coupled sensor configuration and path-planning method in unknown environments. This method depends on a novel task-driven information gain (TDIG) metric, which quantifies the notion of information of “most relevance” to the path-planning problem. The CSCP algorithm places sensors that maximizes the TDIG, i.e., in locations that maximize uncertainty reduction in the path cost. By removing the traditional separation between sensing and planning, the CSCP algorithm significantly reduces the number of sensor measurements required to find near-optimal paths. This property of the CSCP algorithm can benefit applications where sensor data is difficult, expensive, and/or time-consuming to acquire.

Acknowledgement: This research is funded in part by AFOSR grant #FA9550-17-1-0028.

REFERENCES

- [1] M. A. Demetriou and D. Uciński, “State estimation of spatially distributed processes using mobile sensing agents,” in *Proceedings of the 2011 American Control Conference*, 2011, pp. 1770–1776.
- [2] R. Madankan, S. Pouget, P. Singla, M. Bursik, J. Dehn, M. Jones, A. Patra, M. Pavolonis, E. B. Pitman, T. Singh, and P. Webley, “Computation of probabilistic hazard maps and source parameter estimation for volcanic ash transport and dispersion,” *Journal of Computational Physics*, vol. 271, pp. 39–59, 2014.
- [3] J. Cortés, S. Martinez, T. Karatas, and F. Bullo, “Coverage control for mobile sensing networks,” *IEEE Transactions on Robotics and*, vol. 20, no. 2, pp. 243–255, 2004.
- [4] S. Martinez, “Distributed interpolation schemes for field estimation by mobile sensor networks,” *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 491–500, 2010.
- [5] Z. Wu, M. Fu, Y. Xu, and R. Lu, “A distributed Kalman filtering algorithm with fast finite-time convergence for sensor networks,” *Automatica*, vol. 95, pp. 63–72, 2018.
- [6] D. Uciński, “Sensor network scheduling for identification of spatially distributed processes,” *International Journal of Applied Mathematics and Computer Science*, vol. 22, no. 1, pp. 25–40, 2012.
- [7] M. Demetriou, N. Gatsonis, and J. Court, “Coupled controls-computational fluids approach for the estimation of the concentration from a moving gaseous source in a 2-d domain with a Lyapunov-guided sensing aerial vehicle,” *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 853–867, 2013.
- [8] S. Das and J. M. Moura, “Consensus+ innovations distributed Kalman filter with optimized gains,” *IEEE Transactions on Signal Processing*, vol. 65, no. 2, pp. 467–481, 2016.
- [9] S. Martinez, J. Cortes, and F. Bullo, “Motion coordination with distributed information,” *IEEE Control Systems Magazine*, pp. 75–88, August 2007.
- [10] S. Martinez and F. Bullo, “Optimal sensor placement and motion coordination for target tracking,” *Automatica*, vol. 42, no. 4, pp. 661–668, 2006.
- [11] L. Wang, J. Markdahl, Z. Liu, and X. Hu, “Decentralized cooperative tracking subject to motion constraints,” *Automatica*, vol. 96, pp. 121–126, 2018.
- [12] N. Adurthi and P. Singla, “Information driven optimal sensor control for efficient target localization and tracking,” in *Proceedings of the 2014 American Control Conference*, Portland, OR, USA, June 4–6 2014, pp. 610–615.
- [13] N. Farmani, L. Sun, and D. Pack, “Optimal UAV sensor management and path planning for tracking multiple mobile targets,” in *ASME Dynamic Systems and Control Conference*, vol. 46193. American Society of Mechanical Engineers, 2014, p. V002T25A003.

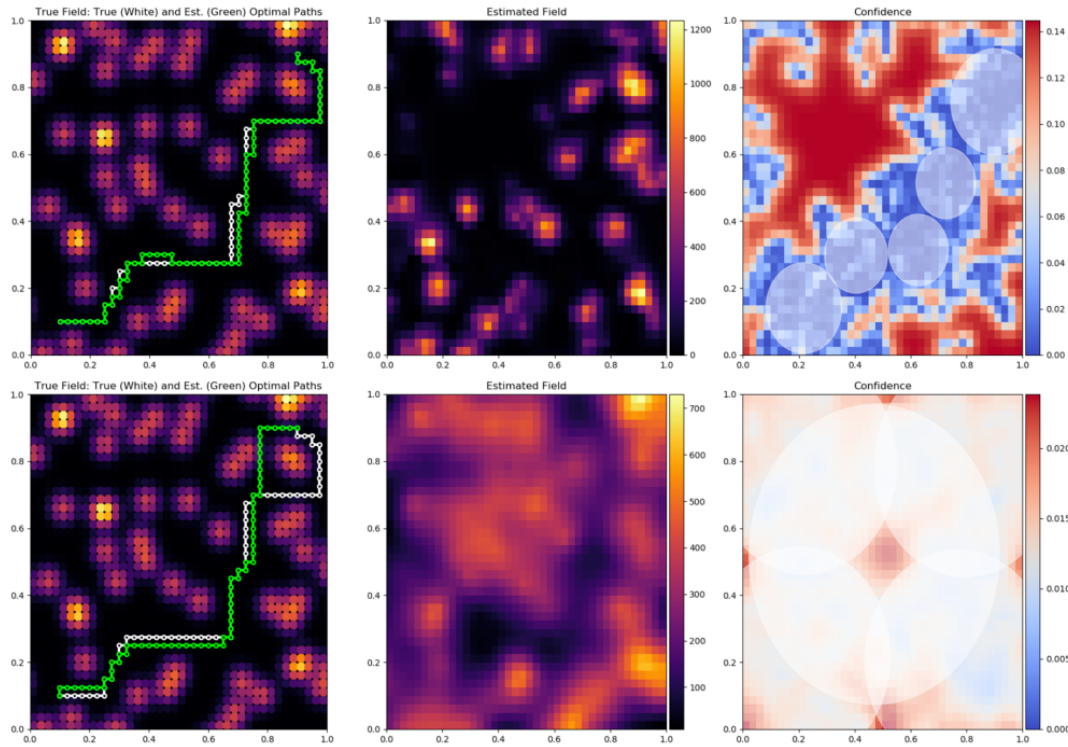


Fig. 11. Minefield example: (top) CSCP optimal path, threat field estimate, and estimation error covariance diagonal values, and (bottom) same quantities for IM approach.

- [14] T. M. Cheng and A. V. Savkin, "Decentralized control of multi-agent systems for swarming with a given geometric pattern," *Computers & Mathematics with Applications*, vol. 61, no. 4, pp. 731–744, 2011.
- [15] A. L. Alfeo, M. G. Cimino, N. De Francesco, A. Lazzeri, M. Lega, and G. Vaglini, "Swarm coordination of mini-uavs for target search using imperfect sensors," *Intelligent Decision Technologies*, vol. 12, no. 2, pp. 149–162, 2018.
- [16] D. Cochran and A. O. Hero, "Information-driven sensor planning: Navigating a statistical manifold," *2013 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2013 - Proceedings*, no. 0, pp. 1049–1052, 2013.
- [17] 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, pp. 235–284, 2008.
- [18] C. K. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in neural information processing systems*, 1996, pp. 514–520.
- [19] D. J. MacKay, "Introduction to Gaussian processes," *NATO ASI Series F Computer and Systems Sciences*, vol. 168, pp. 133–166, 1998.
- [20] C. E. Rasmussen and K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2005.
- [21] C. Stachniss, C. Plagemann, A. J. Lilienthal, and W. Burgard, "Gas distribution modeling using sparse gaussian process mixture models," in *International Conference on Robotics Science and Systems, Robotics: science and systems, 2008, Zürich, Switzerland, June 25-28, 2008*, vol. 4. MIT Press, 2008, pp. 310–317.
- [22] L. Merino, F. Caballero, J. R. Martínez-de Dios, J. Ferruz, and A. Ollero, "A cooperative perception system for multiple uavs: Application to automatic detection of forest fires," *Journal of Field Robotics*, vol. 23, no. 3-4, pp. 165–184, 2006.
- [23] D. Gu and H. Hu, "Spatial gaussian process regression with mobile sensor networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1279–1290, 2012.
- [24] C. Kreucher, A. O. Hero, and K. Kastella, "A comparison of task driven and information driven sensor management for target tracking," in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 4004–4009.
- [25] B. S. Cooper and R. V. Cowlagi, "Interactive planning and sensing in unknown static environments with task-driven sensor placement," *Automatica*, vol. 105, pp. 391–398, Jul 2019.
- [26] H. Kurniawati, T. Bandyopadhyay, and N. M. Patrikalakis, "Global motion planning under uncertain motion, sensing, and environment map," *Autonomous Robots*, vol. 33, no. 3, pp. 255–272, 2012.
- [27] S. Chakravorty and R. Saha, "Simultaneous planning localization and mapping: A hybrid Bayesian/frequentist approach," in *2008 American Control Conference*. IEEE, 2008, pp. 1226–1231.
- [28] D. Duvenaud, J. Lloyd, R. Grosse, J. Tenenbaum, and G. Zoubin, "Structure discovery in nonparametric regression through compositional kernel search," in *International Conference on Machine Learning*, 2013, pp. 1166–1174.
- [29] A. B. Abdesslem, N. Dervilis, D. J. Wagg, and K. Worden, "Automatic kernel selection for gaussian processes regression with approximate Bayesian computation and sequential Monte Carlo," *Frontiers in Built Environment*, vol. 3, p. 52, 2017.
- [30] M. Blum and M. A. Riedmiller, "Optimization of Gaussian process hyperparameters using Rprop," in *European Symposium on Artificial Neural Networks*. Citeseer, 2013, pp. 339–344.
- [31] G. C. Cawley and N. L. Talbot, "Preventing over-fitting during model selection via bayesian regularisation of the hyper-parameters," *Journal of Machine Learning Research*, vol. 8, no. Apr, pp. 841–861, 2007.
- [32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press, 2001.
- [33] J. Lampinen, "A constraint handling approach for the differential evolution algorithm," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, vol. 2, 2002, pp. 1468–1473 vol.2.
- [34] K. P. Murphy, *Machine learning: a probabilistic perspective*. Cambridge, MA, USA: MIT press, 2012.
- [35] C. Lo, "Explosives detection: the world's first sniffer drone," Online, 2020. [Online]. Available: <https://www.army-technology.com/features/featureexplosives-detection-the-worlds-first-sniffer-drone-5729390>