Automatica 140 (2022) 110228

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

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] Waymo LLC, Mountain View, CA, USA
[c] Aerospace Engineering Dept., Worcester Polytechnic Institute, Worcester, MA, USA

ARTICLE INFO

Article history:
Received 18 August 2020
Received in revised form 11 October 2021
Accepted 24 January 2022
Available online 16 March 2022

Keywords:
Multi-agent systems
Linear temporal logic
Path-planning

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 vehicle kinematic model consists of two modes: a hover mode and a constant-speed forward motion mode with bounded steering rate. 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 & 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 (Cowlagi & Zhang, 2017; Kloetzer & Belta, 2007). 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 & Lin, 2010; Kloetzer & Mahulea, 2016; Schillinger et al., 2018b; Shoukry et al., 2017; Ulusoy et al., 2013). 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

☆ The material in this paper was partially presented at In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Kissimmee, FL, USA, January 8–12, 2018. This paper was recommended for publication in revised form by Associate Editor Michael M. Zavlanos under the direction of Editor Christos G. Cassandras.

* Corresponding author.
E-mail addresses: jfang@wpi.edu (J. Fang), zzhang@wpi.edu (Z. Zhang), rvcowlagi@wpi.edu (R.V. Cowlagi).

https://doi.org/10.1016/j.automatica.2022.110228
0005-1098/

& Kyriakopoulos, 2004; Schillinger et al., 2018a). As the number of vehicles increases, the dimension of the product abstraction increases exponentially. Sampling-based asymptotically optimal techniques are reported to construct team product abstractions with fewer states (Kantaros & Zavlanos, 2018b, 2018c).

On the other hand, several *decentralized* algorithms (Filippidis et al., 2012; Guo & Dimarogonas, 2015; Guo & Dimarogonas, 2017; Guo et al., 2016; Moarref & Kress-Gazit, 2017; Tumova & Dimarogonas, 2016) 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 & Zamani, 2016); the product of multiple such models for a team of vehicles becomes impractically large. A reduced-order product is reported (Tumova & 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 (Kantaros & Zavlanos, 2018b, 2018c; Karaman & Frazzoli, 2012; Vasile & Belta, 2013), 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 & 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 & 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 & Zhang, 2017) because reachability analyses for such vehicles typically 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 & 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 Section 1. In Section 2, we discuss the proposed algorithms for solving this problem. In Section 3, we provide illustrative numerical simulation examples, and we conclude the paper in Section 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 $\mathsf{cell}(n) \subset \mathcal{W}$. 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, $n \in [N_\mathrm{R}]$, 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`), $\Box$ (`always`), and $\mathcal{U}$ (`until`) are considered. An LTL formula is constructed using one or more atomic propositions and logical and temporal operators. A *word* $\overline{\omega} = (\omega_0, \omega_1, \dots)$ is a sequence such that each ω_n is a subset of $\{\lambda_n\}_{n=0}^{N_\mathrm{R}}$. For $m, n \in \mathbb{Z}_{\geqslant} 0$, $n \geqslant m$, the word $(\omega_m, \omega_{m+1}, \dots, \omega_n)$ is denoted $\overline{\omega}_m^n$. The *satisfaction of formula* ϕ by the word $\overline{\omega}$ is recursively defined as: (1) $\overline{\omega}$ satisfies λ_n if $\lambda_n \in \omega_0$, (2) $\overline{\omega}$ satisfies $\neg\phi$ if $\overline{\omega}$ does not satisfy ϕ, (3) $\overline{\omega}$ satisfies $(\phi_1 \wedge \phi_2)$ if $\overline{\omega}$ satisfies ϕ_1 and $\overline{\omega}$ satisfies ϕ_2, (4) $\overline{\omega}$ satisfies $(\phi_1 \mathcal{U} \phi_2)$ if there exists $n \geqslant 0$ such that $\overline{\omega}_n^\infty$ satisfies ϕ_2 and for every $m < n$, $\overline{\omega}_m^n$ satisfies ϕ_1. Each route $\mathbf{v} = (v_0, v_1, \dots) \in \mathcal{L}_\mathcal{G}$ defines a word $\overline{\omega}(\mathbf{v}) = (\omega_0, \omega_1, \dots,)$ where $\omega_\ell := \{\lambda_n \mid v_\ell = v_n\}$. The route $\mathbf{v}$ *satisfies* formula ϕ if the word $\overline{\omega}(\mathbf{v})$ satisfies ϕ.

Consider routes $\mathbf{v}_i \in \mathcal{L}_\mathcal{G}$, $i \in [N_\mathrm{A}]$, associated with the motion of each vehicle in the team. Each of these paths defines a word $\overline{\omega}_i(\mathbf{v}_i) = (\omega_{0,i}, \omega_{1,i}, \dots)$, which we concatenate to define $\overline{\omega}(\mathbf{v}_1, \dots, \mathbf{v}_{N_\mathrm{A}}) := (\omega_{0,1}, \omega_{0,2}, \dots, \omega_{1,1}, \omega_{1,2}, \dots)$. Here, the rule of "concatenation" is that $\omega_{\ell,i}$ appears before $\omega_{m,k}$ in $\overline{\omega}(\mathbf{v}_1, \dots, \mathbf{v}_{N_\mathrm{A}})$ if $\ell < m$ or else if $\ell = m$ and $i < k$. The paths $(\mathbf{v}_1, \dots, \mathbf{v}_{N_\mathrm{A}})$ collectively satisfy an LTL$_{-X}$ formula ϕ if $\overline{\omega}(\mathbf{v}_1, \dots, \mathbf{v}_{N_\mathrm{A}})$ 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 $\phi = \phi^{\mathrm{S}} \wedge \phi^{\mathrm{L}}$. 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 $\Diamond(\lambda_{n_1} \wedge \Diamond(\lambda_{n_2} \wedge \Diamond(\ldots \wedge \Diamond\lambda_{n_\ell})))$, 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, $j \in [N_{\mathrm{T}}]$, i.e. $\phi^{\mathrm{L}} := \bigwedge_{j=1}^{N_{\mathrm{T}}} \phi_j$. This structural limitation on ϕ^{L} ensures that it can be decomposed into local specifications to be assigned to each agent. 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:

$$\phi_j = \Diamond\lambda_{n_1}, \qquad \phi_j = \Diamond(\lambda_{n_1} \wedge \Diamond(\lambda_{n_2} \wedge \Diamond(\ldots \wedge \Diamond\lambda_{n_\ell}))),$$
$$\phi_j = \texttt{Env\_Cond} \rightarrow \Diamond\lambda_n$$

where $n_1, n_2, \ldots \in [N_{\mathrm{R}}]$ 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 $1, \ldots, N_{\mathrm{TC}}$ and the independent tasks are labeled $N_{\mathrm{TC}} + 1, \ldots, N_{\mathrm{T}}$. The number of vehicles required to satisfy the j^{th} collaborative specification is denoted M_j. In what follows, the terms 'task j' and 'specification ϕ_j' are synonymously used.

Problem 1. Given an LTL$_{-X}$ specification ϕ over $\{\lambda_n\}_{n=0}^{N_{\mathrm{R}}}$, find traversable routes $(\mathbf{v}_1, \ldots, \mathbf{v}_{N_{\mathrm{A}}})$ with each $\mathbf{v}_i \in \mathcal{L}_{\mathcal{G}}$ such that the word $\overline{\omega}(\mathbf{v}_1, \ldots, \mathbf{v}_{N_{\mathrm{A}}})$ satisfies ϕ. □

In this problem statement, *traversability* is in the context of the vehicle models defined below. We seek a *decentralized* solution to Problem 1 where agent $i \in [N_{\mathrm{A}}]$ computes the path $\mathbf{v}_i$. Agents are assumed to communicate with each other over a static network, where the set of neighbors of agent $i \in [N_{\mathrm{A}}]$ is denoted $\mathcal{N}_i$. We assume that the network topological graph is connected. 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:

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

where the heading rate u is the control input, and $a = 1$ (resp. $a = 0$) in the forward motion mode (resp. hover mode). The set of admissible control input values is the interval $\left[-\frac{1}{\rho}, \frac{1}{\rho}\right]$, with $\rho > 0$. An admissible trajectory traces a curve with a minimum radius of turn of ρ in the forward motion mode. The parameter ρ can be different for each vehicle in the team, thereby modeling heterogeneous maneuvering capabilities (see Example 3 in Section 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.

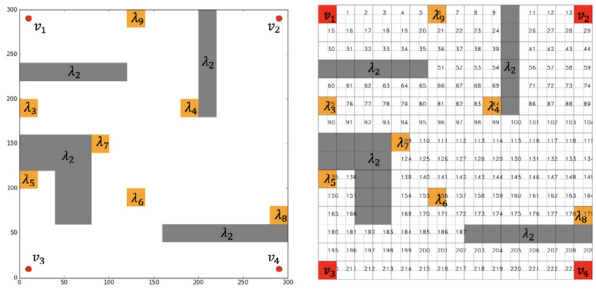
The method of *lifted graphs* is used to construct a discrete abstraction for the KCPM. Briefly, this method considers *sequences* of length $(H + 1)$ of edge transitions in the cell decomposition graph $\mathcal{G}$, where $H \geqslant 1$ is a user-specified integer, and associates these edge transition 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 $\bar{\mathcal{J}}(\mathbf{v})$. 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 & Zhang, 2017). Section 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 $\phi := \phi^{\mathrm{S}} \wedge \phi^{\mathrm{L}}$, $\phi^{\mathrm{S}} := \Box(\lambda_1 \wedge \neg\lambda_2)$, and

$$\phi^{\mathrm{L}} := \Diamond\lambda_3 \wedge \Diamond\lambda_4 \wedge \Diamond\lambda_5 \wedge \Diamond\lambda_6 \wedge \Diamond(\lambda_7 \wedge \Diamond\lambda_8) \wedge \Diamond\lambda_9. \tag{2}$$

Here, $N_{\mathrm{A}} = 4$, ROIs $\lambda_3, \ldots, \lambda_9$ 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 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 $\phi^{\mathrm{L}} = \bigwedge_{j=1}^{N_{\mathrm{T}}}$ with $N_{\mathrm{T}} = 6$, $N_{\mathrm{TC}} = 2$, $\phi_1 := \Diamond\lambda_4$, $\phi_2 := \Diamond\lambda_6$, $\phi_3 := \Diamond\lambda_3$, $\phi_4 := \Diamond\lambda_5$, $\phi_5 := \Diamond(\lambda_7 \wedge \Diamond\lambda_8)$, and $\phi_6 := \Diamond\lambda_9$. □



(a) Workspace and ROIs. (b) Uniform square cells.

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

2. Decentralized route-planning

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

After the local specifications ψ_i are determined, each agent constructs a Büchi automaton, whose accepting runs are exactly the strings satisfying $\phi^{\mathrm{S}} \wedge \psi_i$. 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 (Section 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 & Zhang, 2017; Kloetzer & 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., $N_{\mathrm{TC}} = 0$. 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 $\mathbf{p}_i = (j_1, j_2, \ldots, j_N)$, with each $j_\ell \in \{N_{\mathrm{TC}} + 1, \ldots, N_{\mathrm{T}}\}$. Then we define

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

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 Section 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 Section 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 :

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

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 $\mathbf{b}_i := \mathbf{b}_i \oplus_{\mathrm{end}} j$, and $\mathbf{p}_i := \mathbf{p}_i \oplus_{n_j^*} j$, where $n_j^* := \arg\min_{n \leqslant |\mathbf{p}_i|} \bar{\mathcal{J}}(\mathbf{v}(\mathbf{p}_i \oplus_n j))$. 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 predefined 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 $\bar{\mathcal{J}}(\mathbf{v}(\mathbf{p}_i \oplus_n j)) - \bar{\mathcal{J}}(\mathbf{v}(\mathbf{p}_i))$ 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., $N_{\mathrm{TC}} > 0$, not only are such multi-agent assignments required, but the temporal synchronization of agents to achieve collaborative tasks is also required. One possible approach to the solution of this problem is based on the so-called *consensus-based grouping algorithm* (CBGA, 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 $[N_{\mathrm{T}}]$ 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 $j \in [N_{\mathrm{TC}}]$, then the agents i and k are to visit an ROI simultaneously. One of the agents, say i, will arrive "early"; to synchronize the two agents' visits, agent i must wait at this ROI until agent k arrives.

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

Bundle Construction Phase at Iteration $t \geqslant 1$

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

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

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

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

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

Consensus Phase At Iteration $t \geqslant 1$

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

Fig. 4. Consensus phase of the proposed algorithm.

2.1.3. Proposed algorithm

In the general case of Problem 1 when $N_{\mathrm{TC}} > 0$, we first execute the CBBA to assign independent tasks among the agents, and then execute the proposed algorithm to assign collaborative tasks. Informally, the proposed algorithm consists of a decentralized auction where the agents bid each other for performing collaborative tasks, while simultaneously teaming up with other agents. The agents' incentive for performing tasks is quantified by a reward function, and the incentive for agent teaming is minimization of waiting durations. Precise descriptions of all components of the proposed algorithm are provided as pseudocode in Figs. 2–4. The remainder of this subsection is a description in simpler terms, which may provide the reader a high-level understanding of the algorithm even without reading the pseudocode in detail. In Section 3, the execution of the proposed algorithm is illustrated on Example 1.

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

The element $f_{ijk}(t)$ is the *ith* 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 for independent tasks. The matrices $\mathcal{F}_i(0)$ and $\mathcal{G}_i(0)$ are initialized to zero.

At each iteration $t \in \mathbb{N}$, let $n_{\mathrm{last}}(t)$ denote the sequential position in $\mathbf{p}_i(t)$ of the last collaborative task. At $t = 1$, when no collaborative tasks are yet assigned, $n_{\mathrm{last}}(1) = 0$. In the bundle construction phase, each agent i updates $\mathbf{b}_i(t)$, $\mathbf{p}_i(t)$, $\mathcal{F}_i(t)$, and $\mathcal{G}_i(t)$ by optimally inserting available tasks into $\mathbf{b}_i(t)$ and $\mathbf{p}_i(t)$. To this end, for each $j \in [N_{\mathrm{TC}}]$ that is not present in $\mathbf{b}_i(t)$, an optimal position for insertion n_j^* is computed (Line 8 in Fig. 2). Note that sequential positions only *after* $n_{\mathrm{last}}(t)$ are considered, due to which rewards for tasks already assigned do not change due to insertion of task j.

The reward $f_{iji}(t)$ is updated based on the traversal cost $\bar{\mathcal{J}}(\mathbf{v}(\mathbf{p}_i \oplus_{n_j^*} j|_j))$ of partial completion of $\mathbf{p}_i \oplus_{n_j^*} j$ (Line 9 in Fig. 2). By partial completion we mean the sequential execution of tasks in $\mathbf{p}_i$ until the n_j^{*th} task. The set of agents $\mathbf{u}_j$ assigned to collaborative task j and the agents' waiting duration τ_j are updated per the calculations Δ_{ij} described in Fig. 3. 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 $\|\mathbf{g}_{ij}(t)\|_1 \geqslant M_j$, a sufficient number of agents are currently assigned to task j, but this assignment may not be optimal in terms of the agents' 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). Informally, this step in the bundle construction phase ensures that the waiting durations of agents in the team is minimal.

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 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 $t \geqslant 1$, 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 $s \geqslant t$.*

Proof. If an agent i is assigned to collaborative task $j^\dagger$, let $\tau^i_{j^\dagger}$ denote the duration for which agent i is required to wait to synchronize with other agents assigned to this task. By definition in Fig. 3,

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

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

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

It follows that $\tau_{\mathbf{u},j^\dagger}(t) \leqslant \tau_{\mathbf{u},j^\dagger}(s)$ for every $s \geqslant t$, i.e., the maximum waiting duration for any set of agents assigned to the collaborative task $j^\dagger$ does not increase over the algorithm's iterations. As shown in Choi et al. (2009), the property (5) also implies that for any two sets $\mathbf{u}_1, \mathbf{u}_2 \subset [N_A]$ with $|\mathbf{u}_1| = |\mathbf{u}_2| = M_{j^\dagger}$,

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

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

Proposition 3. *Let δ be the diameter[1] 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 $t \leqslant N_{TC}\delta$,

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

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 $j^\dagger_1 \in [N_{TC}]$, and $\mathbf{u}^*_{j^\dagger_1}$ does not change after iteration t.

Similarly, after at most $(N_{TC}-1)\delta$ further iterations, another task $j^\dagger_2 \in [N_{TC}]\backslash\{j^\dagger_1\}$ meets the condition in the statement of Lemma 2, and consequently $\mathbf{u}^*_{j^\dagger_2}$ 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. □

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 Proposition 3, 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, $\tau_{\mathbf{u}^*_j,j}(t) \leqslant \tau_{\mathbf{u},j}(t)$ 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 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. □

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 & Zhang, 2017), the worst-case complexity of this computation is $\mathcal{O}\left[|\mathcal{G}|N_T 4^H\left(4+\log(|\mathcal{G}|N_T 4^H)\right)\right]$. Therefore, by Proposition 3, the worst-case computational complexity of each agent is $\mathcal{O}\left[\delta N^2_{TC}|\mathcal{G}|N_T 4^H\left(4+\log(|\mathcal{G}|N_T 4^H)\right)\right]$. This result shows a significant innovation over the state-of-the-art because the computational complexity does not depend on the number of agents.

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).

[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.

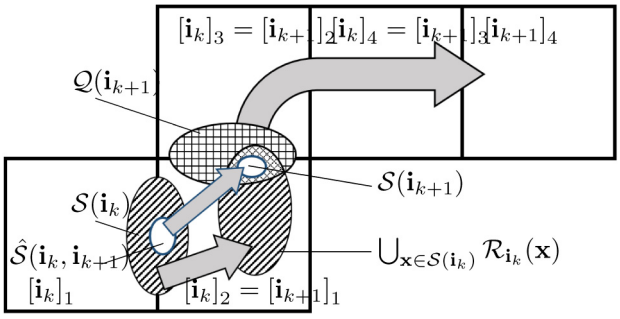


Fig. 5. Illustration (with $H = 3$) of edge cost assignment in the lifted graph.

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

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

Every element $\mathbf{i} \in V_H$ is an ordered $(H+1)$-tuple of the elements of V, and this tuple corresponds to a sequence of successively adjacent cells. We denote by $[\mathbf{i}]_k$ the k^{th} element of $\mathbf{i}$, for $k < m \leqslant H+1$. Let E_H be a set of all pairs $(\mathbf{i}, \mathbf{j})$, with $\mathbf{i}, \mathbf{j} \in V_H$, such that $[\mathbf{i}]_k = [\mathbf{j}]_{k-1}$, for every $k = 2, \ldots, H+1$, and $[\mathbf{i}]_1 \neq [\mathbf{j}]_{H+1}$. The *lifted graph* $\mathcal{G}_H$ is defined as the directed graph (V_H, E_H).

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 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 $\mathcal{R}_{\mathbf{i}}(\mathbf{x})$ 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 $\hat{\mathcal{S}}(\mathbf{i}, \mathbf{j}) := \{\mathbf{x} \in \mathcal{S}(\mathbf{i}) : \mathcal{R}_{\mathbf{i}}(\mathbf{x}) \cap \mathcal{Q}(\mathbf{j}) \neq \varnothing\}$ (see Fig. 5.)

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

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

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

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:

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

Efficient algorithms for computing the sets $\mathcal{R}(\cdot)$, $\mathcal{S}(\cdot)$, and $\mathcal{Q}(\cdot)$ for the KCPM are available (Cowlagi & Zhang, 2017). Any route in $\mathcal{G}$ with finite H-cost is guaranteed to be traversable (Cowlagi & 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, 12 GB 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 $\psi_4 :=$ 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 Fig. 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 $\psi_4 :=$ null.

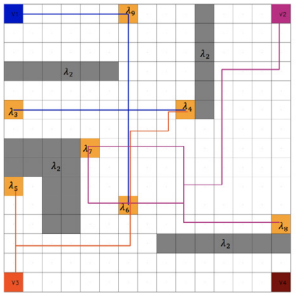
Fig. 6(a) illustrates the corresponding route for each vehicle. Vehicle 1 starts at the blue-colored ROI and then moves to ROIs λ_9, λ_6, λ_4 and λ_3 sequentially. Vehicle 2 starts at the magenta-colored ROI and then moves to ROIs λ_6, λ_7 and λ_8 sequentially. Vehicle 3 starts at the red-colored ROI 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: $\psi_1 = \Diamond(\lambda_9 \wedge \Diamond(\lambda_4 \wedge \Diamond\lambda_3))\psi_2 = \Diamond(\lambda_4 \wedge \Diamond\lambda_6)$, $\psi_3 = \Diamond(\lambda_5 \wedge \Diamond(\lambda_6 \wedge \Diamond(\lambda_7 \wedge \Diamond\lambda_8))\psi_4 = \Diamond\lambda_6$

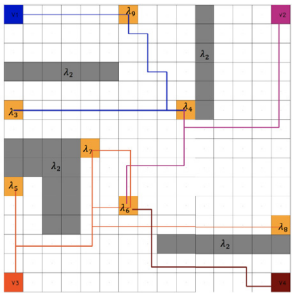
The route traveled by each vehicle is shown in Fig. 6(b). In this case, the sum of lengths of all the vehicles' routes is 94 units. However, the waiting duration for vehicles 2, 3, and 4 is 2, 6, and 10 time units respectively, i.e. a total of 18 time units compared to 4 time units due to the proposed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

Table 1
Proposed algorithm execution in Example 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

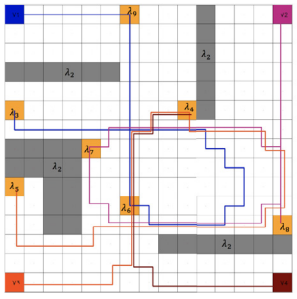


(a) Proposed method.

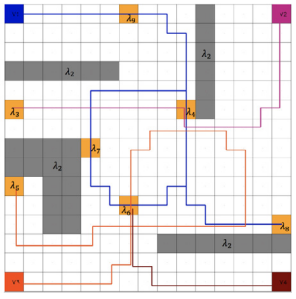


(b) E-CBGA.

Fig. 6. Routes of all vehicles in Example 1: vehicles 1–4 indicated in blue, magenta, red, and maroon, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Proposed method.



(b) E-CBGA.

Fig. 7. Routes of all vehicles in Example 2: vehicles 1–4 indicated in blue, magenta, red, and maroon, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Example 2. We consider Example 1 again, but each vehicle is now modeled by the KCPM. The minimum radius of turn is set to 3 units, where one unit is the side of a square cell. The lifted graph algorithm is applied with $H = 4$ to calculate the rewards for each agent, and to compute routes satisfying each agents' local specifications. The resultant local specifications for each agent are: $\psi_1 := \Diamond(\lambda_9 \wedge \Diamond(\lambda_6 \wedge \Diamond\lambda_3))$, $\psi_2 := \Diamond(\lambda_7 \wedge \Diamond\lambda_8)$, $\psi_3 := \Diamond(\lambda_6 \wedge \Diamond(\lambda_4 \wedge \Diamond\lambda_5))$, $\psi_4 := \Diamond(\lambda_6 \wedge \Diamond\lambda_4)$. The routes of the vehicles to satisfy these local specifications are shown in Fig. 7(a). First, in comparison with Example 1, notice in Fig. 6(a) that the routes of

all vehicles involve several sharp turns, which cannot be executed by a vehicle with minimum turn radius of 3 units. By contrast, in Fig. 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 Example 1, whereas the proposed algorithm results in longer paths, the total waiting duration is significantly lower. □

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

The routes of the vehicles to satisfy these local specifications are shown in Fig. 8(a). First, note that these local specifications and routes are different compared to those in Example 2, although the global specification is the same. For example, the route planned for vehicle 1 (blue) is significantly different because of the larger minimum turn radius in this example. In comparison, the vehicles routes associated with the task assignments by 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. □

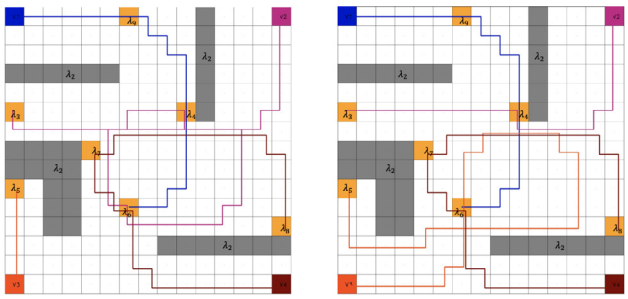
Example 4. The specification $\phi = \phi^S \wedge \phi^L$ in this example involves the operator `implies` denoted $\rightarrow$, which enables vehicular motion conditioned on the environment. The safety and liveness specifications are $\phi^S = \Box(\lambda_1 \wedge \neg\lambda_2)$, and $\phi^L = (\wedge_{m=3}^{9}\Diamond\lambda_m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

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

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. □



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

Fig. 8. Routes of all vehicles in Example 3: vehicles 1–4 indicated in blue, magenta, red, and maroon, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

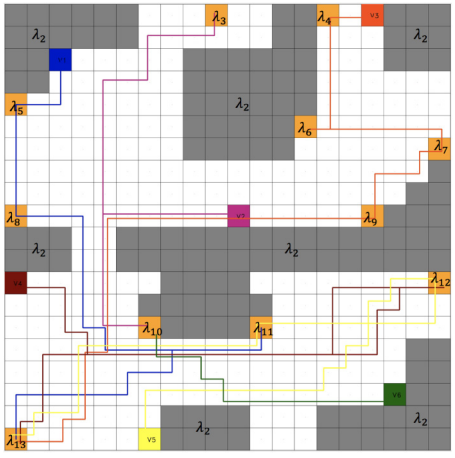


Fig. 9. Routes of all vehicles in Example 4: vehicles 1–6 indicated in blue, magenta, red, maroon, yellow, and green respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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 Example 4 and $\phi^L := (\wedge_{m=3}^{7}\lambda_m) \wedge (\Diamond\lambda_8 \wedge ((\texttt{Found\_T})_8 \rightarrow \Diamond\lambda_{15})) \wedge (\wedge_{n=9}^{11}\lambda_n) \wedge (\wedge_{k=12}^{14}(\Diamond\lambda_k \wedge ((\texttt{Found\_T})_k \rightarrow \Diamond\lambda_{15})))$ with $M_9 = 2$ and $M_i = 3$ for $i = 10, \ldots, 14$.

	$i=1$	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 Example 4, this specification[2] 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. □

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

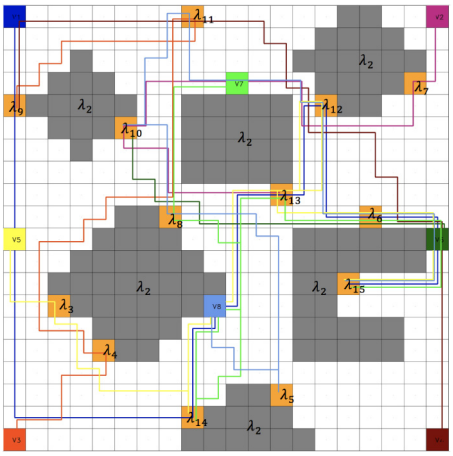


Fig. 10. Example 5: routes of vehicles 1–8 indicated in dark-blue, magenta, red, maroon, yellow, dark-green, light-green, and light-blue respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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.
Example 1	77.78%	3.191%	6
Example 2	58.33%	17.60%	13
Example 3	45.45%	−5.98%	13
Example 4	66.67%	33.15%	11
Example 5	N/A	N/A	14
Example 6	97.68%	12.55%	16

Another simulation example, which we denote **Example 6,** with $N_A = 10$ and $N_T = 13$ is provided in Fang (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 Example 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 Proposition 3, 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 $N_{TI}\delta$ 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/LKOVzu36PwI. 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

Belta, C., Yordanov, B., & Gol, E. A. (2017). *Formal methods for discrete-time dynamical systems*. Springer International Publishing.

Chen, Y., Ding, X. C., & Belta, C. (2011). Synthesis of distributed control and communication schemes from global LTL specifications. In *2011 50th IEEE conference on decision and control and european control conference* (pp. 2718–2723).

Choi, H. L., Brunet, L., & How, J. P. (2009). Consensus-Based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, *25*(4), 912–926.

Cowlagi, R. V., & Zhang, Z. (2017). Route guidance for satisfying temporal logic specifications on aircraft motion. *Journal of Guidance, Control, and Dynamics*, *40*(2), 390–401. http://dx.doi.org/10.2514/1.G001829.

Duret-Lutz, A. (2013). Manipulating LTL formulas using spot 1.0. In *Lecture notes in computer science*: *vol.8172, Proceedings of the 11th international symposium on automated technology for verification and analysis* (pp. 442–445). Hanoi, Vietnam: Springer, http://dx.doi.org/10.1007/978-3-319-02444-8_31.

Emerson, E. A. (1990). Temporal and modal logic. In *Formal models and semantics* (pp. 995–1072). Elsevier.

Fang, J. (2020). *Interactive route-planning and mobile sensing with a team of multiple robotic vehicles to satisfy linear temporal logic specifications*. (Ph.D. thesis), Worcester, MA, USA: Worcester Polytechnic Institute, URL https://digital.wpi.edu/show/8336h474t.

Fang, J., Zhang, Z., & Cowlagi, R. V. (2018). 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.

Filippidis, I., Dimarogonas, D. V., & Kyriakopoulos, K. J. (2012). Decentralized multi-agent control from local LTL specifications. In *2012 IEEE 51st IEEE conference on decision and control* (pp. 6235–6240).

Guo, M., & Dimarogonas, D. V. (2015). Multi-agent plan reconfiguration under local LTL specifications. *International Journal of Robotics Research*, *34*(2), 218–235.

Guo, M., & Dimarogonas, D. V. (2017). Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks. *IEEE Transactions on Automation Science and Engineering*, *14*(2), 797–808.

Guo, M., Tumova, J., & Dimarogonas, D. V. (2016). Communication-free multi-agent control under local temporal tasks and relative-distance constraints. *IEEE Transactions on Automatic Control*, *61*(12), 3948–3962.

Huang, A. S., Olson, E., & Moore, D. C. (2010). Lcm: Lightweight communications and marshalling. In *2010 IEEE/RSJ international conference on intelligent robots and systems* (pp. 4057–4062). IEEE.

Hunt, S., Meng, Q., Hinde, C., & Huang, T. (2014). A consensus-based grouping algorithm for multi-agent cooperative task allocation with complex requirements. *Cognitive Computation*, *6*(3), 338–350.
Kantaros, Y., & Zavlanos, M. M. (2018a). Distributed optimal control synthesis for multi-robot systems under global temporal tasks. In *Proceedings of the 9th ACM/IEEE*. dl.acm.org.
Kantaros, Y., & Zavlanos, M. M. (2018b). STyLuS*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems. CoRR.
Kantaros, Y., & Zavlanos, M. M. (2018c). Temporal logic optimal control for large-scale multi-robot systems: 10400 states and beyond. In *2018 IEEE conference on decision and control* (pp. 2519–2524).
Karaman, S., & Frazzoli, E. (2012). Sampling-based algorithms for optimal motion planning with deterministic μ-calculus specifications. In *2012 american control conference* (pp. 735–742).
Karimadini, M., & Lin, H. (2010). Synchronized task decomposition for two cooperative agents. In *2010 IEEE conference on robotics, automation and mechatronics* (pp. 368–373).
Kloetzer, M., & Belta, C. (2007). Temporal logic planning and control of robotic swarms by hierarchical abstractions. *IEEE Transactions on Robotics*, *23*(2), 320–330.
Kloetzer, M., & Mahulea, C. (2016). Multi-robot path planning for syntactically co-safe LTL specifications. In *2016 13th international workshop on discrete event systems* (pp. 452–458).
Loizou, S. G., & Kyriakopoulos, K. J. (2004). Automatic synthesis of multi-agent motion tasks based on LTL specifications. In *Decision and control, 2004. CDC. 43rd IEEE conference on, Vol.1* (pp. 153–158).
Moarref, S., & Kress-Gazit, H. (2017). Decentralized control of robotic swarms from high-level temporal logic specifications. In *2017 international symposium on multi-robot and multi-agent systems* (pp. 17–23). http://dx.doi.org/10.1109/MRS.2017.8250926.
Rungger, M., & Zamani, M. (2016). Scots: A tool for the synthesis of symbolic controllers. In *Hybrid systems computation and control*.
Schillinger, P., Bürger, M., & Dimarogonas, D. V. (2018a). Decomposition of finite LTL specifications for efficient multi-agent planning. In *Distributed autonomous robotic systems* (pp. 253–267). Springer.
Schillinger, P., Bürger, M., & Dimarogonas, D. V. (2018b). Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. *The International Journal of Robotics Research*, *37*(7), 818–838.
Shoukry, Y., Nuzzo, P., Balkan, A., Saha, I., Sangiovanni-Vincentelli, A. L., Seshia, S. A., Pappas, G. J., & Tabuada, P. (2017). 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* (pp. 1132–1137). http://dx.doi.org/10.1109/CDC.2017.8263808.
Stavrou, D., Timotheou, S., Panayiotou, C. G., & Polycarpou, M. M. (2018). Optimizing container loading with autonomous robots. *IEEE Transactions on Automation Science and Engineering*, [ISSN: 1545-5955] *15*(2), 717–731. http://dx.doi.org/10.1109/TASE.2017.2679485.
Toth, P., & Vigo, D. (2002). *The vehicle routing problem*. Philadephia, USA: SIAM.
Tumova, J., & Dimarogonas, D. V. (2016). Multi-agent planning under local LTL specifications and event-based synchronization. *Automatica*, *70*, 239–248.
Ulusoy, A., Smith, S. L., Ding, X. C., Belta, C., & Rus, D. (2013). Optimality and robustness in multi-robot path planning with temporal logic constraints. *International Journal of Robotics Research*, *32*(8), 889–911.
Vasile, C. I., & Belta, C. (2013). Sampling-based temporal logic path planning. In *2013 IEEE/RSJ international conference on intelligent robots and systems*.
Zavlanos, M. M., Spesivtsev, L., & Pappas, G. J. (2008). A distributed auction algorithm for the assignment problem. In *Decision and control, 2008. CDC 2008. 47th IEEE conference on* (pp. 1212–1217).

Jie Fang is a Quality Engineer at MathWorks Inc., Natick, MA, USA. She received her Ph.D. degree in Mechanical Engineering from Worcester Polytechnic Institute in 2020 and her MS degree in Mechanical Engineering from the University of Florida in 2014. Her research interests include on the connected vehicle system including task allocation, route planning and estimation, and multi-vehicle collaboration in unknown environment.

Zetian Zhang is a Software Engineer at Waymo LLC, Mountain View, CA, USA. He received his MS degree in Mechanical Engineering and Ph.D. degree in Aerospace Engineering from Worcester Polytechnic Institute in 2013 and 2019, respectively, and his BS degree in Energy and Thermal Engineering from the University of Shanghai for Science and Technology in 2011. His research is focused primarily on autonomous vehicles, motion planning and optimal control.

Raghvendra V. Cowlagi is Associate Professor of Aerospace Engineering at Worcester Polytechnic Institute, Worcester, MA, USA. He obtained a Ph.D. degree in Aerospace Engineering from Georgia Tech, worked as a postdoctoral researcher at MIT, and worked as a member of the research staff at Aurora Flight Sciences Corp., Cambridge, MA. His research interests are in the area of motion planning, control, and estimation for the autonomy of mobile vehicles. He serves as Associate Editor for Aerospace Science and Technology, the ASME Journal of Autonomous Vehicles and Systems, and the IEEE-CSS Conference Editorial Board. He is a recipient of the AFOSR Young Investigator Award (2016) and the AFRL Summer Faculty Fellowship (2021).