EURO Journal on Transportation and Logistics 10 (2021) 100058

Contents lists available at ScienceDirect

EURO Journal on Transportation and Logistics

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

A multi-agent semi-cooperative unmanned air traffic management model with separation assurance

Yanchao Liu *

Department of Industrial and Systems Engineering, Wayne State University, Detroit, MI, USA

ARTICLE INFO

Keywords:
Drone delivery
Nonlinear optimization
Air traffic management

ABSTRACT

This paper presents an air traffic management framework to enable multiple fleets of unmanned aerial vehicles to traverse dense, omni-directional air traffic safely and efficiently. The main challenge addressed here is separation assurance in the absence of full coordination and communication. In this framework, each fleet is independently managed by a routing agent, which progressively plans the non-overlapping move-ahead corridors for vehicles in the fleet by solving a nonlinear optimization model. The model is artfully designed so that agents of different fleets need not engage in complicated multilateral communications or make guesses about external vehicles' flight intents to maintain effective inter-vehicle separation. For a complex routing problem, the framework is able to support centralized fleet routing, decentralized vehicle self-routing, and any other agent-vehicle configuration in between, allowing for customized trade-off between response time and traffic efficiency. Innovative algorithmic enhancements for solving the agent's nonconvex routing problem are prescribed with detailed annotation. The effectiveness and noteworthy properties of the framework are demonstrated by several simulation experiments.

1. Introduction

Unmanned aerial vehicles (UAVs or drones, also referred as unmanned aircraft systems (UASs)) have been widely used in both military and civilian domains. As of March 2020, over 1.5 million drones have been registered with the Federal Aviation Administration (FAA), about 30% of which are commercial drones. The increasingly congested air traffic requires an appropriate level of coordination to balance the needs of safety, efficiency and equity. Conventional air traffic control approaches, which rely on human-level response and dexterity, are unable to cope with the unprecedented density and complexity of unmanned air traffic. In the future-generation UAS traffic management (UTM) paradigm, automation will be a central theme. The role of mathematical optimization is expected to shift from providing decision support for human operators to making real-time control decisions and seeing through their execution on robotic platforms in complex and uncertain environments.

To achieve the most efficient use of the airspace, a UAS's flight intent should not simply be equated to a preemptive corridor lock for an extended period of time or over an extended volume of airspace, because doing so would undermine the airspace utilization. Compared to manned flights, UAS flight missions will be more frequent and more ad-hoc. Pre-departure flight path planning will not work unless activities of other airspace users are also considered, which may also be highly dynamic and uncertain. Therefore, even a simple mission such as going from point A to point B as is the case for most delivery scenarios, is unlikely to be warranted an obstacle-free straight-line path from A to B at the most energy-efficient cruise speed. Compromises such as delayed takeoff, in-flight rerouting, forced loitering and speed adjustments have to be effectively arbitrated among different airspace users, to achieve socially efficient and equitable outcomes. Many sophisticated methods in the literature focused on addressing segregated conflict scenarios, serving as decision support tools for human aviators. Recent regulatory and technological advancements suggest that the need for a fully automated air traffic management system is imminent.

In this paper, we propose a routing optimization model to be used independently by multiple routing agents that allows real-time dynamic allocation of airspace volumes with robust mechanisms for conflict resolution. Safety standards held equal, a traffic management system built on this model can support more concurrent flights in a given volume of airspace than what existing systems are capable of, thus allowing more goods to be transported via automated aerial delivery. The intended use of the proposed system is to serve as the software core of an enterprise-serving USS under the NASA-developed UTM architecture (Kopardekar et al., 2016; Prevot et al., 2016; Jiang et al., 2016; Rios et al., 2020; McCarthy et al., 2020). It is also applicable to the

* Correspondence to: 4815 4th Street Rm 2169, Detroit, MI 48201, USA.
E-mail address: yanchaoliu@wayne.edu.

https://doi.org/10.1016/j.ejtl.2021.100058
Received 26 December 2020; Received in revised form 18 August 2021; Accepted 11 November 2021
Available online 28 November 2021

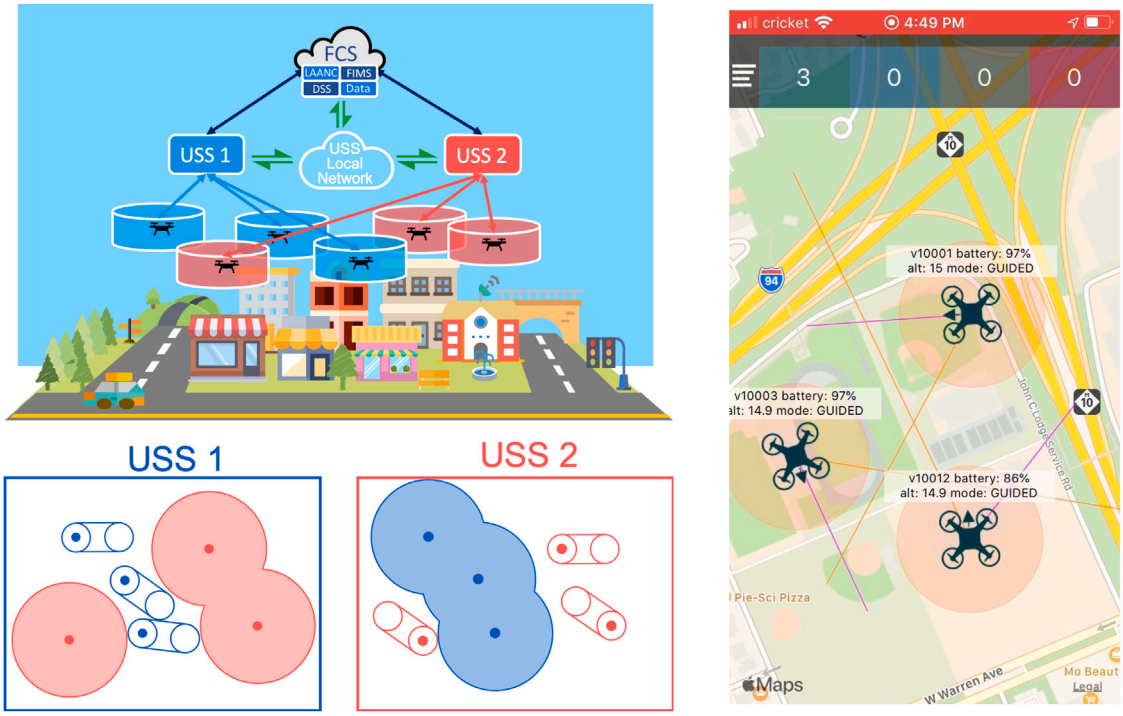

Fig. 1. Concept of the enterprise UAS service supplier (USS) and semi-cooperative trajectory management adopting a multi-agent optimization framework. Topleft: Two USS-controlled fleets operate in the same urban airspace. Bottomleft: The same traffic scenario is perceived differently by different USSs. Right: A screenshot of our USS iOS App adopting the proposed framework.

airspace service provider (ASP) or the air navigation service provider (ANSP) concepts touted by Google and Amazon (Google Inc., 2020; Amazon, 2018), respectively. An enterprise-serving USS is an entity in charge of the entire operation planning and control of the private fleet owned by an organization, such as a drone delivery company or a Drones-as-a-Service (DaaS) platform operator. Multiple USS operating in the same or adjacent geographic region(s) will share information through standard protocols and infrastructure, such as the Low Altitude Authorization and Notification Capability (LAANC) under FAA's UAS Data Exchange, the Remote ID protocol being developed by FAA and industry participants (Federal Aviation Administration, 2020), and the Discovery and Synchronization Service (DSS) in accordance with ASTM WK65041 (ASTM Subcommittee F38.02, 2020), see Fig. 1 for a concept illustration. Different from most other multi-agent concepts where each agent represents a single vehicle or robot, in the proposed system an agent represents a USS executing central control over its own fleet of UAVs. Multiple agents passively interact with each other via observing (and reacting to) the motion of each other's fleets. No complicated inter-agent communication is required to achieve harmonious traffic outcomes. UAVs within the same fleet are cooperative, while UAVs across different fleets are semi-cooperative—they share the same goal of maintaining safe separation but do not communicate flight intents or negotiate rights of way as do UAVs managed by the same USS. Central to the proposed system is a nonlinear nonconvex optimization problem to be solved independently by each agent. A solution approach leveraging commercial off-the-shelf solvers is shown to be effective for a wide range of realistic scenarios.

2. Literature review

Many new patents have been filed in recent years to address the traffic management challenges in anticipation of some unprecedented air traffic density and heterogeneity in the near future. Kopardekar (2019) proposed that a traffic management system can be managed by a commercial business, an academic institution, a government agency, or a combination thereof, depending on application scenarios. Such a system is by nature decentralized and portable. Chan et al. (2016) invented a toll system for airway usage in which aircraft are billed for their access to the airspace corridors, which are expected to become a scarce resource. Shaw et al. (2018) proposed to manage the use of the airspace via a radio access network (RAN), which implements a prioritized queuing system for drones requesting access to the airspace managed by a RAN node. Klooster et al. (2014) proposed the concept of fleet wide trajectory management systems to be installed at UAS operation control centers, which remotely plan, modify, predict, and manage an aerial vehicle's trajectory in four-dimensional (4D) airspace. In the patent by Dupray and LeBlanc (2020), several embodiments of UAV flight path planning procedures were envisioned, including the use of Dijkstra's algorithm for finding shortest paths, assessing the probability of trajectory conflicts before taking evasive maneuvers, and electronically "chaining" multiple UAVs together so that the operator only needs to directly control the lead vehicle while the other vehicles in the chain follow the lead in a specified formation. Despite the variety of new ideas, a quick search of the patent database indicated that mathematical optimization had been scantly mentioned by inventors in this space.

Optimization-based traffic management approaches have been, however, extensively studied in the academic literature. Frazzoli et al. (2001) developed a nonconvex quadratically constrained quadratic program to model the planar, multiaircraft conflict resolution problem. The overall framework consisted of centralized decision making for safety and decentralized preference optimization for efficiency, whereas the optimization objective was to minimize the deviation between the desired and conflict-avoiding heading for each aircraft. The authors furthermore presented a semidefinite programming relaxation scheme and a randomized search approach for resolving various local conflict patterns. Zhang et al. (2012) proposed a hierarchical flight planning framework that decomposes the process of seeking optimal paths for all flights in the air into two interactive decision layers. The first layer turns on/off air sectors available for use in the future time slots based on existing flight plans and traffic condition, while the second layer optimizes individual flight paths subject to the traffic rules set in the

first layer. The first layer problem and multiple second layer problems are in the form of dual decomposition of the centralized planning problem. Global optimality not being a main concern, this approach allows large-scale flight planning problems to be modeled flexibly and solved efficiently. Another two-level approach, which resulted in a large mixed integer programming (MIP) model, was presented in Dell'Olmo and Lulli (2003). Zhu and Wei (2019) developed a linear programming based approach to compute pre-departure conflict-free trajectories for autonomous aircraft. The idea of dynamic geofencing, i.e., a moving volume of reserved airspace that absorbs various uncertainty, was exploited. Tan et al. (2019) addressed the multi-drone route planning problem via evolutionary optimization algorithms, in which the total flight delay and collision possibility were minimized via a specially designed fitness function. Mukherjee and Hansen (2009) developed MIP models to support flight rerouting in response to reduced airport capacity in the context of uncertain and evolving weather conditions. The models suggested that making rerouting decisions dynamically would result in 10 to 15 percent reduction in delay cost compared to static rerouting. Pallottino et al. (2002) attempted to resolve trajectory conflicts among multiple aircraft by formulating a velocity change problem and a heading angle change problem separately, and used MIP techniques to model the or-clauses in the constraint set. The models aimed to generate a one-shot velocity or heading change so that trajectories will not overlap for the rest of the flight time in the area. To balance equity and efficiency in the presence of flight plan conflicts, Barnhart et al. (2012) developed an integer programming model to minimize a (un)fairness metric that measures the deviation from the default priority (first come first serve) in the presence of conflicts. The authors concluded the paper by recognizing that introducing optimization into the FAA's practices had been a significant challenge. This is in sharp contrast to the wealth of optimization research effort dedicated at addressing UAV application challenges, see survey papers by Otto et al. (2018), Chung et al. (2020). This observation enhanced our belief that for an emerging new system such as UTM, the resource optimization mindset and available technologies should be assessed early in the rulemaking and standardization processes, as it would be much harder and costlier to mend inefficient mandates than making them efficient in the first time.

Letting computers to automatically resolve trajectory conflicts is not new even for large aircraft. Erzberger (2004) hypothesized a computer program for tactical separation assured flight environment (TSAFE) that automatically takes control of resolving close-in conflicts when the time to loss of separation breaches a critical time threshold, i.e., around 2 min. For an embodiment of TSAFE, Erzberger and Heere (2010) developed a reliable (i.e., if-then rules) two-aircraft conflict resolution algorithm based on analytical formulations of separation characteristics with bank angle constraints. Given that the computation was to be performed in a ground-based system which would then send maneuver commands to the aircraft in a conflict via ground-air data link, the authors went on to prescribe the data-link protocol requirements to ensure a reliable implementation. We postulate a cloud-air communication link through the 4G/LTE cellular network for implementing the framework proposed in this paper. While the reliability of this mode of communication deserves a separate, thorough evaluation, in this paper we build robustness measures in the routing model to account for sporadic losses of connection between the vehicle in the air and the agent's command center in the cloud.

Hoekstra et al. (2011) presented many interesting empirical results, analyses and analogies regarding free flight in a crowded airspace, where "free flight" was the combined concept of free routing and airborne separation. The author claimed that "it is right now and will be for a long time impossible to guarantee the stability or risk associated with the behavior of a large number of aircraft in any configuration. ... The large-scale effects of traffic patterns can only be studied using simulations". Our computational experience corroborates these conjectures—while there is no theoretical guarantee of traffic safety in face of uncooperative vehicle behaviors, the reliability of the framework can be assessed by simulation and statistical methods. Jang et al. (2017) envisioned a lane-based airspace structure, in which the airspace in the urban area is divided into several layers by altitude, and in each layer a number of air travel lanes are specified. Collision avoidance could then be achieved by having each vehicle keep a safe following distance, in the same way as automobiles would do in highway traffic. The model in effect reduced the spatial dimension of air traffic to one. It was demonstrated via simulation that the system-wide speed and throughput were both negatively correlated with traffic density. Our simulation experiments clearly reveal similar and even stronger results, in the context of two-dimensional omni-directional traffic which is more chaotic.

Ho et al. (2020) proposed a pre-flight deconfliction approach based on pairwise prioritization and negotiations among agents involved in trajectory conflicts. UAS service providers were allowed to plan their flight paths based on individual needs (e.g., mission and cost structure) and then would negotiate with each other to modify the initial plans to avoid conflicts. The approach emphasized fairness in the allocation of costs incurred by plan modifications. The tradeoff between efficiency and fairness in the UTM context has also been studied in Chin et al. (2020). The authors claimed that there was no all-encompassing metric of fairness whereas the allocation of resources would depend critically on the metric chosen. The tradeoff was evaluated in a rolling-horizon simulation. Zhao et al. (2019) presented a simulation tool for fast evaluation of the effects of different ATM policies on resource utilization and traffic pattern, which was applied for evaluating cost and benefits of cellular and satellite communication channels for UAV operations (Zhong et al., 2020) and for evaluating a real-time routing algorithm for small UAS (Jin et al., 2019). Recent simulation-driven approaches for assessing UTM performances can also be found in Bulusu et al. (2017), Yang et al. (2020).

Chen et al. (2015) presented a multi-agent path planning algorithm especially for drones flying in nonconvex shaped (e.g., indoor) environments, which addressed a shortcoming in an earlier work, i.e., Augugliaro et al. (2012), of applying sequential convex programming (SCP) to the multi-drone motion planning problem. Specifically, the SCP method developed in the latter paper tended to generate overly conservative constraints when applied to nonconvex environments, frequently leading to false claims of infeasibility. To overcome this problem, the authors proposed a new method that was able to incrementally tighten the constraints in the solution process, thus ensuring the feasibility of the intermediate optimization problems. The authors also experimented implementing the solution process in a decoupled fashion, in which drones (agents) solved their own trajectory optimization problems sequentially, in one pass. The decoupled implementation helped to improve the computational tractability and was demonstrated to better suit real-time trajectory planning needs.

Directly related to the present work is our prior work in Liu (2019), in which a nonlinear optimization model and a progressive solution approach were developed to route vehicles in dense air traffic with feasibility guarantee. We implemented the framework in a working software prototype to enable field tests on real drone platforms, and conducted numerous software-in-the-loop (SITL) simulations (ArduPilot Dev Team, 2020) and field tests. Several new challenges have been identified through these tests. For one, the actual speed and acceleration cannot be precisely dictated by high-level navigational commands, so the trajectory planning model cannot assume that all the UAVs under dispatch will reach the planned locations exactly on time. The model must instead be able to tolerate locational discrepancies while not compromising safety. Furthermore, in reality we must plan for such occasions when a UAV temporarily loses connection with the command center. We programmed the onboard failsafe mechanism to brake the vehicle and have the vehicle hover still (or loiter) until the connection is re-established. This seemingly innocuous action may disrupt the traffic,

cause loss of separation, and break the progressive planning process. These issues are mitigated by the new model presented in this paper.

Overall, this paper presents the following unique contributions. (1) We propose a decentralized optimization model to implement the UTM routing service for heterogeneous drone fleets; (2) We provide methods for parameterizing and solving this optimization model effectively, to guarantee a feasible chain of system states and inter-vehicle separation assurance in practice; (3) We perform extensive numeric simulations using the model to uncover important properties of the multi-agent air traffic system and of the routing model in particular, which provides guidance to UTM's real-world implementation.

3. Problem statement

We focus on transportation-purposed UAV flights, that is, flying horizontally from point A to point B as quickly as possible is a UAV's objective when it is requesting the UTM's routing service. The routing service under discussion is presumably for UAVs traversing the airspace at a given altitude. Even though the proposed mathematical model is agnostic of the space dimension (i.e., suitable for both 2D and 3D), we believe it is more realistic to leave the takeoff and landing phases of UAS flights, such as the approach sequence coordination and descend trajectory management, for a separate program, e.g., operated by the airport rather than by fleet operators/airspace users.

Consider this scenario: in a city's airspace at a given altitude (as specified by the city's airspace access rules, e.g., 300 ft above ground level), there are one or more last-mile delivery companies each operating its own fleet of UAVs (we will simply call them vehicles). Each company employs a routing agent (i.e., a computer program) to guide vehicles in its own fleet to their respective destinations. Note that this is not a one-shot task but an always-on service, to accommodate constantly emerging trip requests as the vehicles shuttle through the region performing on-demand deliveries. A vehicle is guided by periodically receiving navigational commands from its agent, whereas a command specifies the next waypoint the vehicle should travel to and at what speed. An agent is not only able to monitor the status of its own fleet, but also able to frequently query the locations of vehicles from other fleets in the airspace, via the Inter-USS network. Apart from publishing the fleet locational information to the network, agents do not engage in any bilateral or multilateral communications. There is no vehicle-to-vehicle communication either. Using available information, an agent's task is to periodically compute navigational commands so as to guide the vehicles to their destinations along the most efficient routes while maintaining sufficient inter-vehicle separation at all times.

For simplicity, we take a temporal snapshot of the vehicle locations and their intended destinations as the starting point for the multi-agent routing process, and call such a snapshot a *traffic scenario*. Vehicles that have reached their destinations during the routing process will exit the system (i.e., disappear from the radar screens of all agents), while new trip requests will be added to the system as they arise. Entrance of new vehicles into the traffic can be handled in a similar way as proposed in Liu (2019), thus will be omitted in this paper. We instead focus on designing a framework that allows multiple agents to work "semi-cooperatively" to route all vehicles to their destinations in any given traffic scenario, regardless of how many vehicles and different fleets there are in the scenario. The meaning of "semi-cooperative" will become clearer when the agent's model is discussed subsequently.

4. The agent's optimization model

In this section, we propose the main routing model from an agent's perspective and develop the model parameters that will elicit tacit agreements with other agents regarding the right of way. We call the model *SEMICO* and its formulation is given by Eqs. (1)–(6). It is a nonlinear minimization problem over a nonconvex continuous feasible set. The solution process involves solving the problem instances repeatedly, once every few seconds, with updated status data. Commercial solvers such as CONOPT will be used for local solutions.

Table 1
Notation definition.

Symbol	Definition
$u_{i,t}$	location of vehicle i at time t, in $\mathbb{R}^2$
v_i	velocity vector of vehicle i, in $\mathbb{R}^2$
$w_{i,j,t}$	loss of separation between vehicles i and j at time t, in $\mathbb{R}_+$
V_i	maximum speed of vehicle i, in $\mathbb{R}_+$
D_i	destination coordinate of vehicle i, in $\mathbb{R}^2$
$S_{i,j}$	intra-fleet separation distance between vehicles i and j, $i,j \in \mathcal{O}$
$S'_{i,j}$	inter-fleet separation distance between vehicles i and j, $i \in \mathcal{O}$, $j \in \mathcal{E}$
α_i, β	priorities and penalty factor, in $\mathbb{R}$
$\hat{u}_{i,t}$	expected location of external vehicle i at time t, in $\mathbb{R}^2$

Table 1 defines the symbols used in the model. The upper half of the table lists decision variables and the lower half lists parameters. In the text, we use the superscript $*$ after a variable to mark the solution value of that variable. The vector norm operator $\|\cdot\|$ is taken to be the l_2-norm. For instance, $\|u_{i,t} - D_i\|$ means the Euclidean distance between vehicle i's location at time t, $u_{i,t}$, and the vehicle's destination, D_i. We discretize time and let symbol $\mathcal{T}$ represent the index set of time points in the planning time frame in context. Throughout the paper, we use t_0 to denote the "current" time point at which the model is run, and use T to denote the length of the planning time frame, thus we have $\mathcal{T} = \{t_0, t_0 + 1, \ldots, t_0 + T\}$. When symbols u, v and w are mentioned without any subscript, they should be understood as vectors whose elements are defined in Table 1.

SEMICO:

$$\text{Min}_{u,v,w} \quad \sum_{i\in\mathcal{O}} \alpha_i \left(\sum_{t\in\mathcal{T}} \|u_{i,t} - D_i\| \right) + \beta \cdot \sum_{i\in\mathcal{O}, j\in\mathcal{E}} \left(\sum_{t\in\mathcal{T}} w_{i,j,t}^2 \right) \tag{1}$$

$$\text{s.t.} \quad u_{i,t} = u_{i,t-1} + V_i v_i, \ \forall i \in \mathcal{O}, t \in \mathcal{T}/\{t_0\} \tag{2}$$

$$\|v_i\| \le 1, \ \forall i \in \mathcal{O}, t \in \mathcal{T} \tag{3}$$

$$\|u_{i,t} - u_{j,t'}\| \ge S_{i,j}, \ \forall i,j \in \mathcal{O}, i < j, t, t' \in \mathcal{T} \tag{4}$$

$$\|u_{i,t} - \hat{u}_{j,t}\| + w_{i,j,t} \ge S'_{i,j}, \ \forall i \in \mathcal{O}, j \in \mathcal{E}, t \in \mathcal{T} \tag{5}$$

$$w_{i,j,t} \ge 0, \ \forall i \in \mathcal{O}, j \in \mathcal{E}, t \in \mathcal{T}. \tag{6}$$

From an agent's perspective, vehicles in the system are divided into two subsets, own fleet $\mathcal{O}$ and external fleet $\mathcal{E}$. Vehicles in the agent's own fleet are controlled by the agent, while vehicles in any external fleet are controlled by other agent(s), and their ID and location information is observable (e.g., via the remote ID and the Inter-USS infrastructure).

The agent's task is to route its own fleet of vehicles to their respective destinations along the most efficient paths, while maintaining sufficient inter-vehicle separation at all times. The agent's objective is implemented by penalizing (minimizing) two terms in (1). The first term is the total weighted distance to destination summed over all vehicles and all time points in the agent's look-ahead time frame. This term encourages minimum-length path, as is also employed in Schulman et al. (2014). The weighting factor $\alpha_i \ge 1$ represents the right-of-way priority of vehicle i among all vehicles in $\mathcal{O}$. The strategy for setting its value is discussed in Section 5.2. The second term is the total squared loss of buffer space between all own-external vehicle pairs. Penalizing this quantity incentivizes vehicles in the own fleet to keep a safe distance away from the external vehicles. Note that penalizing the squared violation can incentivize the vehicle in control to stay away equally from all external vehicles that give rise to a loss of separation with it. In contrast, if the absolute value of w were penalized, then the vehicle would not have such an incentive, causing some violation more severe than others. The parameter β is a constant to add weight to the second term. Its value determines the relative importance between separation assurance and route efficiency. In computation, as long as

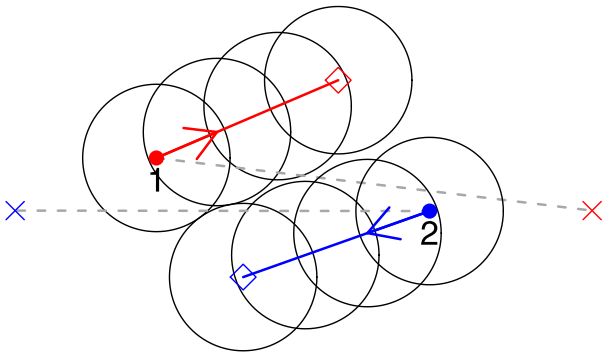

Fig. 2. Snapshot of a motion plan for two internal vehicles with a planning horizon $T = 3$. For $i \in \{1, 2\}$, solid dots mark u_{i,t_0}, arrows mark $V_i v_i$, crosses mark D_i, and hollow diamonds mark u_{i,t_0+T} which are also the target waypoints sent to the respective vehicles for execution.

a large value is set for it, the routing solution will be insensitive to its specific value. Eqs. (2) and (3) describe the linear motion of vehicles going from one waypoint to the next. Note that the velocity vector v_i is not indexed by time, thus the same velocity (direction and magnitude) is applied to the entire planning time frame. Compared to the model in Liu (2019) which allowed velocity to change across time points within the planning horizon, this constraint amounts to a restriction in the model's flexibility—less assumption is made about the controllability and certainty of the vehicles' motion in the real world. Specifically, suppose that we are at time t_0 and the planning time frame is $\mathcal{T} = \{t_0, t_0 + 1, \ldots, t_0 + T\}$. Instead of imposing a series of waypoints (one for each time point) as done in the old model, in the current model the agent commands the motion of vehicle $i \in \mathcal{O}$ by sending a single target coordinate, that is, u^*_{i,t_0+T}, or equivalently, $u_{i,t_0} + V_i T v^*_i$ derived from the difference equation (2). Given that u_{i,t_0} and V_i are both known and fixed at the time of planning, t_0, the target coordinate is totally determined by the computed speed vector v^*_i. Another benefit of using a time-invariant velocity vector is the reduced model complexity and hence faster computations.

Constraint (4) imposes trajectory separation of all pairs of vehicles in the own fleet. Any pair of vehicles i and j must remain at least $S_{i,j}$ apart along their moving directions. Compared to the old model in Liu (2019) which only checks matching time points (as would be written as $\|u_{i,t} - u_{j,t}\| \geq S_{i,j}, \forall i, j \in \mathcal{O}, t \in \mathcal{T}$), this constraint requires separation for all combinations of time points along the two trajectories. Essentially, each vehicle claims an exclusive headway corridor of T time intervals along its moving direction. The spatial length of the corridor is equal to $V_i T v_i$ for vehicle i. These treatments relieve the need for time synchronization in the execution stage, thus can effectively hedge against uncertain and off-nominal situations. For instance, in reality the ground speed may not be perfectly controlled due to inertia, drag and communication latency, so vehicles may reach their respective waypoints earlier or later than the planned time points; furthermore, if a vehicle experiences a loss of communication link it will brake and hover/loiter until communication is regained. As long as the vehicle stays in its headway corridor, traffic safety will not be jeopardized by these situations. Fig. 2 illustrates the motion plan of two internal vehicles. From the "decision-making under uncertainty" perspective, this way of handling uncertainty can be viewed as a "robust" method, whereas a "stochastic" counterpart can be found in ACAS X, in which uncertainty in the aircraft's state estimation is represented as a probabilistic distribution, and this distribution specifies where to look in a table to determine which conflict resolution advisory to provide to the pilots (Kochenderfer et al., 2012).

Constraints (5) and (6) define the loss of separation $w_{i,j,t}$ between internal (own) and external vehicles that is to be penalized in the objective function. For an internal vehicle $i \in \mathcal{O}$, keeping at least $S'_{i,j}$ away from the external vehicle $j \in \mathcal{E}$ is encouraged but not strictly enforced—violation will be penalized but will not render the model infeasible. The reason for using "soft" constraints here is that, in our experience, if hard constraints were used no algorithm could guarantee feasibility of the model without imposing additional assumptions about other agents' behavior or about the traffic density. For an extreme instance where a hostile external vehicle (predator) seeks to collide with some other vehicle (prey), nothing can stop the loss of separation between the two if the predator's maximum speed is greater than that of the prey. Maintaining the chain of (mathematical) feasibility in the solution process is highly desirable both in the algorithm evaluation and in its practical use. We will show in the next section that the soft constraints are generally conducive to the desired level of separation between internal and external vehicles. Simulation experiments in Section 7 will corroborate these conclusions.

Static obstacles and no fly zones can be handled endogenously by the SEMICO model. In the low-altitude airspace (i.e., below 400 ft), commonly encountered obstacles include tall buildings, areas claimed by other airspace users, and FAA-recognized identification areas (FRIAs) in compliance with the remote ID rule Federal Aviation Administration (2019, 2020). At a given altitude, the cross sections of their spherical or cylindrical envelopes form circular areas, see Fig. 3. From an agent's standpoint, such a circular area can be treated as a special vehicle either in $\mathcal{O}$ or in $\mathcal{E}$. Parameter setting and variable fixing for the dummy vehicles are straightforward.

4.1. Parameter setting tactics in implementation

Parameters involved in the SEMICO model fall in three categories: input data, algorithmic parameters, and user-defined parameters. Input data include the vehicle-agent membership $\mathcal{O}$ and $\mathcal{E}$, vehicle destination D_i and vehicle maximum speed V_i. Algorithmic parameters include α_i, β and $\hat{u}_{j,t}$, whose values are set by the routing algorithm to be discussed in the next Section. Here, let us look at the user-defined parameters including the length of the planning time frame T, and the separation distances $S_{i,j}$ and $S'_{i,j}$. First, how long (or how many seconds) does a modeling time interval correspond to in reality? To place an anchor point, we assume throughout the paper that SEMICO is solved at every modeling time point, that is, the current time t_0 advances one step at a time. The length (in seconds or minutes) of the unit time interval in the model is an important application-dependent parameter. Factors such as vehicle type, size and maneuverability, minimum separation headway time, desired look-ahead time, and the geographical span of the airspace under management, will come into consideration. For instance, the state-based conflict detection algorithm in the Free Flight study uses a (pretty long) look-ahead time of five minutes to allow for maneuvers that do not discount passenger comfort (Hoekstra et al., 2011), while traffic collision avoidance system (TCAS) uses a look-ahead time of 20 to 48 s (Table 2 in Federal Aviation Administration (2011)) which may result in drastic evasive maneuvers for passenger aircraft. In the airborne collision avoidance system (ACAS X), the aircraft state is re-estimated once per second, based on which the conflict resolution advisory is selected to provide to the pilots (Kochenderfer et al., 2012). In Augugliaro et al. (2012), a modeling time unit of 0.2 s was adopted, and in Chen et al. (2015), a separation distance of 0.8 m was required for each time step, since the experimental scenarios consisted of small quadcopters flying in an indoor arena.

Suppose that by user choice, each modeling time interval corresponds to three seconds and that the planning horizon is $T = 5$, then the routing process will go like this: at the current time point t_0 (let us assume that the current clock mark is 1), the agent first reads in the current locations of its own vehicles, u_{i,t_0} for $i \in \mathcal{O}$, as well as those of external vehicles, $\hat{u}_{j,t_0}$ for $j \in \mathcal{E}$, and predicts the future locations of $j \in \mathcal{E}$. Absent further information, it is reasonable to set $\hat{u}_{j,t_0+k} = \hat{u}_{j,t_0}$ for $k = 1, \ldots, T$ and $j \in \mathcal{E}$. Given these inputs, the agent then plans the fleet motion for the next five periods (15 s, or the $[1, 16]$-second

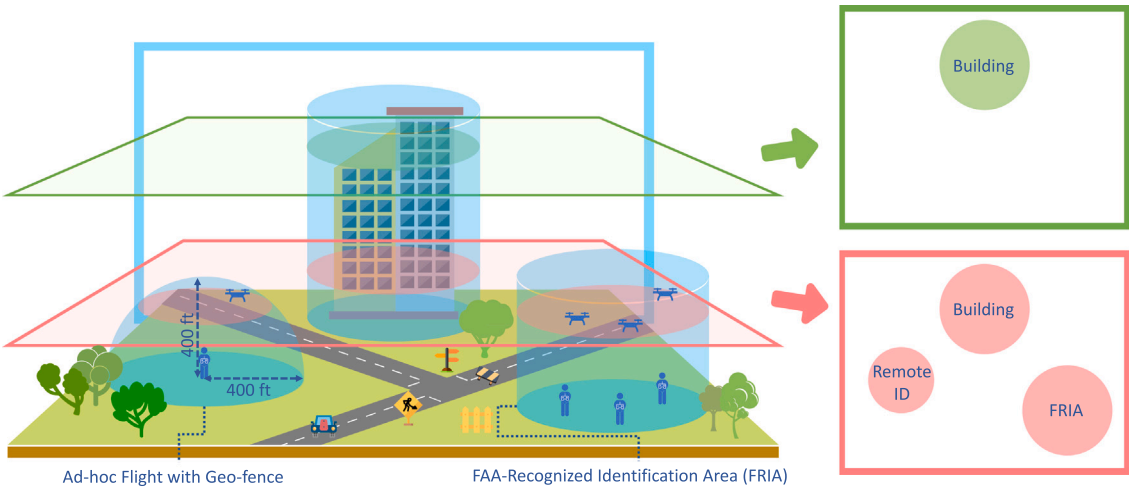

Fig. 3. Illustration of circular no-fly zones which can be easily incorporated in the SEMICO model as a special non-moving vehicles.

period) by solving SEMICO. Suppose that solving SEMICO takes two seconds of computing time, after which the agent sends the optimal waypoints u^*_{i,I_0+5} to each vehicle i in $\mathcal{O}$. The agent then waits until a total of 3 s (2 s computing time plus 1 s idle time) have elapsed to begin repeating the above process—reading in the current locations, solving SEMICO (for the next five intervals which corresponds to the [4, 19]-second interval), and sending updated waypoints. To keep the cadence, it is apparent that a modeling time interval must correspond to a real time interval that is at least longer than the solution time of SEMICO. In the next section, we will develop an algorithm that solves SEMICO consistently quickly to allow for a small unit interval length. On the other hand, updating waypoints too frequently (due to setting a small unit interval length in the model) is not suitable for long-range trips and low density traffic, for it would be a waste of computing and data-link bandwidths.

The value of T will affect the problem size, computing time and the long-term planning efficiency. A greater value of T generally allows the agent to plan farther into the future, therefore benefits the long-term planning efficiency. In the meantime, a large T would make solving SEMICO time-consuming, especially when the number of vehicles in the own fleet is also large.

The required inter-vehicle separation distance is best specified in terms of headway time, denoted by h. For instance, traffic safety rules could mandate that any pair of vehicles in the same altitude layer must be kept 6 s apart, meaning that it takes at least six seconds before two vehicles can crash into each other even if they travel head-to-head toward each other at their respective maximum speeds. Suppose that the maximum speeds of the two involved vehicles, 1 and 2, are both 15 m/s, then the required separation distance will be $6 \times (15+15) = 180$ m. Converting this scenario into the modeling framework by assuming 3 s per modeling time interval, we get $V_1 = V_2 = 15 \times 3 = 45$ m per time interval, and the headway time $h = 6/3 = 2$ time intervals. It is analyzed in Liu (2019) that discretizing time will cause a loss of separation in-between successive time intervals. By applying the formula thereof, we set the separation distance $S_{i,j}$ in SEMICO by

$$S_{i,j} = \sqrt{h^2 + 0.25}(V_i + V_j).$$

Continuing with the above example, $S_{1,2}$ is equal to $\sqrt{2^2 + 0.25}(45 + 45) = 185.5$ m.

For the separation buffer $S'_{i,j}$ between an internal vehicle i and an external vehicle j, extra room must be allotted for the uncontrollability of the external vehicle j (or the behavior of the other agent controlling vehicle j). Furthermore, for the pair of vehicles i and j that are controlled by two different agents, the two involved agents should adopt different buffer sizes in their respective models to form an implicit right-of-way priority order between themselves with regard to i and j in conflict. The simplest strategy to create the needed asymmetry is to prioritize the vehicles in lexicographical order. Specifically, we propose the following formula for setting $S'_{i,j}$, $i \in \mathcal{O}$, $j \in \mathcal{E}$.

$$S'_{i,j} = (\sqrt{h^2 + 0.25} + b_1 + \mathbb{1}(i < j)b_2)(V_i + V_j), \tag{7}$$

where b_1 and b_2 are positive constants, and $\mathbb{1}(\cdot)$ is the indicator function, i.e., $\mathbb{1}(E)$ is equal to 1 if the condition E is true, and equal to 0 if E is false. In this formula, b_1 represents the added buffer to account for uncertainty and b_2 represents the priority difference between i and j.

Continuing with the above numerical example with $h = 2$ and $V_1 = V_2 = 45$, now suppose that vehicles 1 and 2 are controlled by agents a1 and a2, respectively, and let us set $b_1 = b_2 = 1$, then agent a1 will have $S'_{1,2} = (\sqrt{2^2 + 0.25} + 1 + 1)(45 + 45) \approx 366$ and agent a2 will have $S'_{2,1} = (\sqrt{2^2 + 0.25} + 1 + 0)(45 + 45) \approx 276$. The implication is as follows: as soon as the planned trajectories of vehicles 1 and 2 are closer than 366 m apart, a1 will start directing vehicle 1 to make collision avoidance maneuvers. However, so long as the distance is above 276 m, a2 will still perceive the situation as being safe hence vehicle 2 will proceed as if vehicle 1 is not in its way. This asymmetry creates a desirable tacit agreement: vehicle 1 yields the right-of-way to vehicle 2, and vehicle 2 takes the courtesy to quickly pass through the conflicted area.

The workings of the SEMICO model can be construed figuratively as follows: each vehicle is a point mass surrounded by a circular forbidden area for collision avoidance, thus spatially each vehicle i is represented by a disc of radius $\sqrt{h^2 + 0.25}V_i$ and constraint (4) says that the discs of different vehicles in the same fleet should not overlap. The objective function acts as applying a force to pull the discs toward the vehicles' destinations, with the priority α_i representing the relative strength of the pulling force on vehicle i. When the imaginary discs (of radii $(\sqrt{h^2 + 0.25} + b_1 + \mathbb{1}(i < j)b_2)V_i$ and $(\sqrt{h^2 + 0.25} + b_1 + \mathbb{1}(i < j)b_2)V_j$, respectively) of an interval vehicle i and an external vehicle j overlap at time $t \in \mathcal{T}$, a strong repulsion force of magnitude $\beta w_{i,j,t}$ is applied to push the vehicles apart.

Fig. 4 visualizes four contiguous temporal snapshots (time 15 to 18) of a two-agent traffic scene, to demonstrate how the model and formula (7) works. Vehicles 1 (red) and 2 (blue), supposedly controlled by agents a1 and a2, respectively, travel head-to-head toward their destinations (marked by the crosses). By Eq. (7), vehicle 1 is more risk averse than vehicle 2 so its safety disc is larger. The planning horizon is $T = 3$ in this example. At time 15, both vehicles planned to move ahead at full speed for no loss of separation would ensue—observe that the solid red circles representing the safety discs of vehicle 1's planned

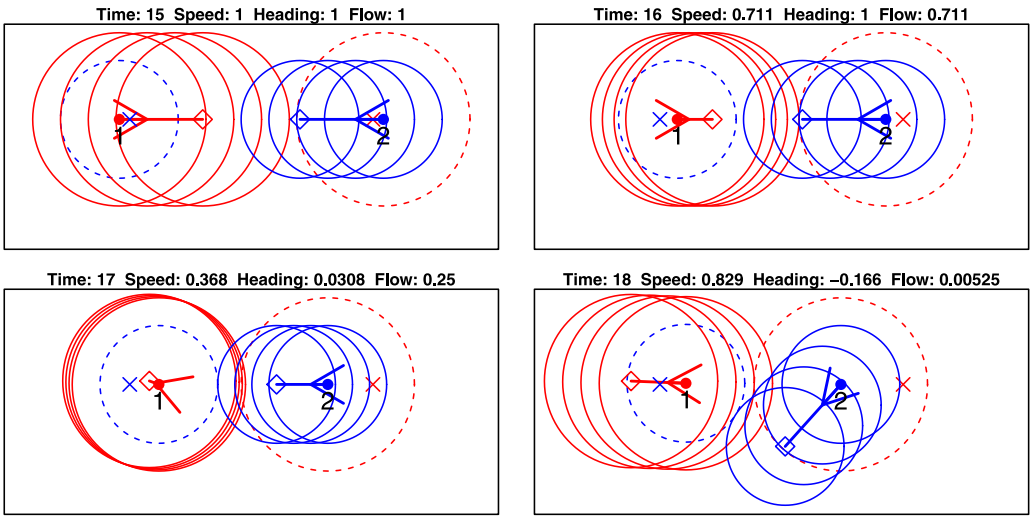

Fig. 4. Demonstration of what happens when vehicles from different fleets encounter. The two vehicles use different risk thresholds therefore form a tacit agreement for conflict resolution. Vehicle 1 is more cautious about collision hence reacts to the situation earlier, whereas vehicle 2 is less sensitive hence reacts later. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

locations and the dashed red circle representing the safety disc of vehicle 2's anticipated location (in the eyes of agent a1) do not overlap, and the same observation goes for the solid and dashed blue circles as in the eyes of agent a2. At time 16, vehicle 1 planned to reduce its speed to a level such that a loss of separation with vehicle 2 barely accrues—observe that the foremost solid red circle barely touches the dashed red circle. However, vehicle 2 still planned a full-speed move because the collision risk is not in sight yet for agent a2—the solid and dashed blue circles are still apart. At time 17, vehicle 2 also planned to reduce speed as the separation constraint kicked in for agent a2, while vehicle 1 decided to turn around and escape the conflict. At time 18, vehicle 1 planned to further ramp up speed in its escape, while vehicle 2 changed speed direction to avoid the loss-of-separation penalty. Overall, the conflict has been resolved elegantly by the two autonomous agents, and no real loss of separation has occurred—due to the added headway buffer b_1+b_2 and b_1 in agent a1's and a2's models, respectively.

5. Solution process and feasibility guarantee

SEMICO is a nonlinear programming (NLP) model, used for generating a feasible motion plan for a set of cooperative vehicles (controlled by the same agent) for the next few time periods. Each agent solves its own copy of SEMICO in a rolling horizon framework, while passively interacting with each other via the inter-fleet collision avoidance constraints (5) and the penalty terms in the objective function. We call this fashion of multi-agent interaction "semi-cooperative", explaining the name SEMICO.

The benefit of SEMICO is two-fold. On one hand, it implements the decentralized paradigm of UTM spearheaded by NASA, by allowing an arbitrary number of agents (i.e., USSs and/or UAS operators) to operate in the shared airspace. On the other hand, it provides a sound basis for promoting traffic efficiency via optimized fleet routing operations.

SEMICO is designed to be used as the routing module of an enterprise USS, to provide the routing service for a delivery fleet or for a city-wide Drones-as-a-Service (DaaS) platform. The foremost requirement for an online service is its stability and resilience—it should not easily break and when it does break, it should restore to the normal operating condition with little hassle. Translating this requirement to SEMICO, we need a robust solution process that guarantees a feasible chain of system states. The robustness is enabled by the ability to always obtain locally optimal (thus feasible) solutions to the SEMICO model and to resolve deadlock situations effectively. We will develop a solution algorithm (by leveraging off-the-shelf solver CONOPT) that fulfills this requirement. As a side note, obtaining the globally optimal solution to each run of SEMICO is unnecessary in the progressive routing framework, with detailed justification provided in Liu (2019).

5.1. Starting point strategy for feasibility guarantee

The routing process involves solving SEMICO iteratively as time increments by one unit per iteration. Let us use the notation SEMICO($\mathcal{T}$) to denote the model instance spanning a given time frame $\mathcal{T}$. The NLP solver CONOPT (Drud, 1994) has proven effective for finding good-quality local solutions for large-scale nonconvex optimization problems (Liu, 2019, 2020). In particular, the Generalized Reduced Gradient (GRG) algorithm implemented in CONOPT is a primal method with the property that, once a feasible solution is identified, the subsequent candidate solutions remain feasible until a local optimum is found (CONOPT, 2020). A feasible starting point eliminates stage one of the CONOPT routine which is minimizing infeasibility, thus saves computing time. Furthermore, a "good" feasible starting point is instrumental for finding a good local solution. We will develop a starting point strategy to realize these benefits.

The presence and behavior of external vehicles do not affect the feasibility of an agent's SEMICO instance, because any loss of separation buffer is absorbed by the variable w. So let us focus on the constraints (2)–(4) and the related variables u and v. To set the stage for discussion, given an arbitrary integer t_0 and an integer $T \geq 1$, let $\mathcal{T} = \{t_0, t_0 + 1, \ldots, t_0 + T\}$ and $\mathcal{T}' = \{t_0 + 1, t_0 + 2, \ldots, t_0 + T + 1\}$. Suppose that v_i^*, $i \in \mathcal{O}$ and $u_{i,t}^*$, $i \in \mathcal{O}$, $t \in \mathcal{T}$ constitute a feasible solution to the problem SEMICO($\mathcal{T}$), then a feasible solution to SEMICO($\mathcal{T}'$) can be constructed by simply setting v_i to 0 for all i and setting $u_{i,t}$ to u_{i,t_0+1}^* for all i and all $t \in \mathcal{T}'$. This always-feasible starting point is reassuring to have, however, initializing key variables to zero is not actually helpful for the nonlinear optimization process because it creates difficulty for finding informed descent directions. We craft a more informative starting point as follows. Let

$$v_i' := (T-1)v_i^*/T, \forall i \in \mathcal{O} \tag{8}$$

$$u_{i,t}' := u_{i,t_0+1}^* + (t - t_0 - 1)V_i v_i', \forall i \in \mathcal{O}, t \in \mathcal{T}' \tag{9}$$

then use them as the starting point to solve the problem SEMICO($\mathcal{T}'$) using CONOPT. The idea is to keep the same velocity direction as the preceding solution but reduce the speed to a level such that the new T-step trajectory falls in the same envelope as does the (feasible)

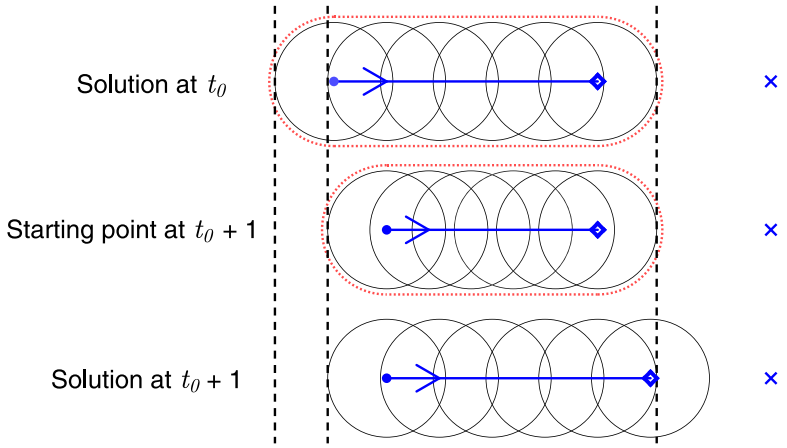

Fig. 5. Demonstration of the caterpillar-like probing motion for setting the starting point for the succeeding iteration.

trajectory computed in the preceding iteration. Fig. 5 illustrates how the motion plan is updated from one iteration to the next, i.e., from t_0 to t_0+1, as the above starting point strategy is employed. We can see that the trajectory envelope (outlined in red dashed lines) of the starting-point motion plan at $t_0 + 1$ is a subset of that of the locally optimal (thus feasible) motion plan found at t_0, therefore, the starting point is also feasible. Figuratively, the progression of the motion plan is much like a caterpillar's locomotion—squeezing muscles in an undulating wave motion to drag the tail forward, while probing ahead. Unless in extremely artificial cases (an example is provided in the appendix), the starting point constructed above is feasible for SEMICO($\mathcal{T}'$).

The above analysis as well as formula (8) and (9) assumes that the motion plan is executed precisely, i.e., the starting location of vehicle i for the planning cycle $\mathcal{T}'$ is exactly u^*_{i,t_0+1} as planned in the previous cycle $\mathcal{T}$. In practice, the starting location is populated by the vehicle's GPS location. There are always discrepancies between the planned location and the actual location, which come from measurement errors, inertial lags and flight controller's delayed response to waypoint commands. Accordingly, the starting point for $\mathcal{T}'$ should be revised as follows.

$$v'_i = \frac{\|u^*_{i,t_0+T} - \hat{u}_{i,t_0+1}\|}{\|u^*_{i,t_0+T} - u^*_{i,t_0}\|} v^*_i,\ \forall i \in \mathcal{O} \tag{10}$$

$$u'_{i,t} = \hat{u}_{i,t_0+1} + (t - t_0 - 1)V_i v'_i,\ \forall i \in \mathcal{O}, t \in \mathcal{T}' \tag{11}$$

where $\hat{u}_{i,t_0+1}$ is the actual location of vehicle i at time $t_0 + 1$. Formula (8) and (9) are used in simulation experiments while formula (10) and (11) are used in software implementation.

5.2. Deadlock resolution

Deadlock refers to a situation in which multiple vehicles block the way of one another and settle at a locally optimal solution having all involved vehicles come to a complete stop. Its occurrence is mainly attributed to the nonconvexity of the SEMICO model as well as to the limited horizon of the planning session. A simplest deadlock scenario is when two (or more) vehicles having comparable priorities approach a common destination simultaneously. It is analogous to simultaneously pulling two (or more) balls toward a same anchor point using elastic cords. It can be imagined that the balls will stabilize at pushing against each other, none able to align its center to the anchor point.

Deadlocks cannot be resolved endogenously by the optimization model without substantially complicating the formulation or making extra assumptions. For instance, mixed integer programming (MIP) models have been attempted in Zhou et al. (2020) to address the sequencing and routing challenges of landing multiple aerial vehicles at corridor-constrained vertiports, and pairwise prioritization and negotiation procedures were used in Ho et al. (2020). Here, we develop a deadlock detection and correction method in the progressive motion planning framework that employs the SEMICO model as a building block.

Let $\mu^*_{i,j,t,t'}$ be the Lagrangian multiplier (or dual variable) for the corresponding constraint (4) at the local optimal solution (u^*, v^*, w^*) of SEMICO, then by the first-order necessary condition for optimality the following equation holds,

$$\left(\|u^*_{i,t} - u^*_{j,t'}\| - S_{i,j}\right)\mu^*_{i,j,t,t'} = 0,\ \forall i, j \in \mathcal{O}, i \neq j,\ t, t' \in \mathcal{T}.$$

This means that when $\mu^*_{i,j,t,t'} > 0$, the separation constraint between vehicles i and j is binding at the time points t and t' along their respective planned trajectories. When this happens for any $t, t' \in \mathcal{T}$, i.e., when $\max_{t,t'\in\mathcal{T}} \mu^*_{i,j,t,t'} > 0$, we say that vehicles i and j are in a *binding cluster*. The notion of binding cluster is transitive, that is, if i and j are in a binding cluster, and j and k are in a binding cluster, then all three of them are in the same binding cluster, even if $\max_{t,t'\in\mathcal{T}} \mu^*_{j,k,t,t'} = 0$.

After each solve of the SEMICO model, the following steps are implemented to prevent and resolve deadlocks.

1. Set $\alpha_i = 1$, $\forall i \in \mathcal{O}$, and identify all distinct binding clusters.
2. For each binding cluster C:
   (a) Pick the smallest (in lexicographical order) vehicle and set its priority to 100. Mathematically, set $\alpha_{i'} = 100$ where $i' = \min_{i\in C} i$.
   (b) If all vehicles in the cluster stopped moving, then negate the sign of their priorities except for the one picked in the above step. Mathematically, if $\max_{i\in C} \|v^*_i\| < \epsilon$, then set $\alpha_i = -1$ for all $i \in C/\{i'\}$.

Remarks: (1) The idea in step 2(a) is to give one vehicle a distinctively higher priority than the other vehicles in the binding cluster, to make that vehicle traverse quickly, therefore resolve the cluster deadlock one vehicle at a time. The reason for using the lexicographical order is for simplicity and for consistency across different iterations. It is important to pick the same vehicle over the iterations until the binding cluster dissolves, rather than alternating between different vehicles. Lexicographical order can achieve this consistency without extra bookkeeping. Other prioritizing schemes can be used in practice as long as the consistency in vehicle selection is preserved. (2) Step 2(b) is a more dramatic measure to unravel deadlocks. The idea is to temporarily "loosen up" the tightly packed cluster, to allow room for the trapped high-priority vehicle to escape. One might wonder why having a high priority alone is insufficient to guarantee the high-priority vehicle's ability to cut through the clustered congestion. The fundamental cause is the inclusion of t_0 in the set $\mathcal{T}$ in constraint

(4) and the fact that u_{i,t_0}, $i \in \mathcal{O}$, assumes a fixed value (current location, which is read in). Suppose that vehicles i and j are in a conflict and Step 2(a) is triggered to set $\alpha_i = 1$ and $\alpha_j = 100$. One would then expect vehicle i to move away (back off or move sideways, depending on the attack angle of vehicle j) in order to make way for vehicle j. In fact, the constraints $\|u_{i,t_0} - u_{j,t'}\| \geq S_{i,j}$, for $t' \in \mathcal{T}$, may prevent that from happening, because vehicle j is locked out of vehicle i's current location u_{i,t_0} no matter where vehicle i decides to move—in this case vehicle i does not receive the needed incentive to act as wanted. In general, whenever the low-priority vehicle's actual location (not the planned location at a future time) is in the way of the high-priority vehicle, the prioritization method will fail to serve its purpose of proactively resolving conflicts. In such situations, Step 2(b) is a necessary backup procedure to rectify stalemated deadlocks. Admittedly, this is a heuristic approach. However, it has been tested effective in numerical simulations of very high-density traffic scenarios. We have not encountered a deadlock situation that cannot be resolved by the above method.

After each iteration, the agent scans all vehicles in $\mathcal{O}$ and removes those that have reached their destination. Specifically, the agent updates $\mathcal{O} \leftarrow \mathcal{O}/\{i : \|u^*_{i,t_0+T} - D_i\| \leq r\}$, where r is a small positive number representing the proximity threshold. This step mirrors what is actually done in practice as vehicles exit the routing system. When two (or more) vehicles have destinations that are too close to each other, a situation also observed in Section 6.4 of Dechering (2019), the above prioritization method works without a problem.

5.3. Reducing unnecessary corridor locks to effectivize prioritization

As analyzed above, constraint (4) enforced over time combinations of $(t_0, t_0+1), \ldots, (t_0, t_0+T)$ and $(t_0+1, t_0), \ldots, (t_0+T, t_0)$ may nullify the prioritization strategy. The rationale for enforcing these constraints is to prevent nearby vehicles from running into a vehicle's safety disc in case the vehicle suddenly comes to a halt (brake and hover, as a failsafe action triggered by the vehicle's onboard computer). In fact, we can remove certain time combinations over which the constraint is enforced, so as to reinstate the effectiveness of prioritization without compromising safety. The key point here is to exploit the fact that a vehicle can move at most one step before re-planning occurs, so a vehicle that (by motion plan) needs to travel two steps to reach a point cannot possibly reach (or violate separation constraint with) the point before the next re-planning. This means that constraint (4) over $(t_0, t_0+2), \ldots, (t_0, t_0+T)$ and $(t_0+2, t_0), \ldots, (t_0+T, t_0)$ can be safely relinquished. As a side note, constraints $\|u_{i,t_0} - u_{j,t_0}\| \geq S_{i,j}$, for $i, j \in \mathcal{O}, i < j$, are also redundant, because u_{i,t_0}, $i \in O$, are not really variables—their values are fixed by actual status data read in at the beginning of the current planning horizon. In summary, constraint (4) is implemented as follows.

$$\|u_{i,t} - u_{j,t'}\| \geq S_{i,j},\ i, j \in \mathcal{O}, i < j,\ t, t' \in \mathcal{T}/\{t_0\}$$
$$\|u_{i,t_0} - u_{j,t_0+1}\| \geq S_{i,j},\ i, j \in \mathcal{O}, i < j$$
$$\|u_{i,t_0+1} - u_{j,t_0}\| \geq S_{i,j},\ i, j \in \mathcal{O}, i < j.$$

Compared to the original version, the above implementation makes the prioritization strategy more effective in preventing deadlocks, as has been verified by numerical experiments.

6. Performance evaluation metrics

The foremost benefit of using optimization-based route planning method is to maximize traffic efficiency, particularly when the airspace becomes a scarce resource due to high traffic density. We adopt the following metrics to quantify traffic efficiency. For an individual vehicle i, the instantaneous speed efficiency, heading efficiency and flow efficiency at an arbitrary time point t_0 are defined as follows,

$$\eta_i^{\text{spd}} = \|v_i\|,$$
$$\eta_i^{\text{hdg}} = \frac{v_i^T (D_i - u_{i,t_0})}{\|D_i - u_{i,t_0}\|},$$
$$\eta_i^{\text{flow}} = \eta^{\text{spd}} \eta^{\text{hdg}}.$$

The corresponding efficiency metrics for the entire trip of a vehicle, for all vehicles in a fleet, or for the entire traffic scenario consisting of all fleets, can be calculated by summing the above quantities over appropriate sets of time or vehicles.

Since the separation between vehicles of different fleets is enforced as soft constraints in the SEMICO model, the effectiveness of the soft constraints needs to be monitored after the fact. Questions such as how frequently and how much the underlying separation requirement is violated must be answered. To do so, after each routing iteration, we calculate the following quantities for each pair of vehicles i and j, $i < j$, that are controlled by different agents.

$$E_{ij} = \max\{h(V_i + V_j) - \|u^*_{i,t_0} - u^*_{j,t_0}\|, 0\},$$
$$\delta_{ij} = \mathbb{1}(E_{ij} > 0),$$
$$\rho_{ij} = \frac{E_{ij}}{h(V_i + V_j)}.$$

Here, E_{ij} is the distance by which the separation requirement between i and j is violated at t_0, δ_{ij} records whether a violation has occurred or not, and ρ_{ij} is the relative extent (i.e., $100\rho_{ij}$ percent) of the violation. These metrics can be summarized over relevant time periods and selected sets of vehicles for customized reports.

7. Simulation experiments

In this section, we present numerical experiments to demonstrate noteworthy properties of the SEMICO model and agent behaviors. All experiments were performed on a Dell Precision Tower 8520 with an Intel(R) Core(TM) i9-9900X CPU @ 3.50 GHz, 16.0 GB RAM and Windows 10 Enterprise Operating System. We used GAMS 30.2.0 and the CONOPT solver for modeling and solution. Default solver options were used and the solver is single-threaded.

This is how a traffic scenario consisting of N vehicles was generated. The origin and destination (OD) coordinates (abscissa and ordinate) of each of the N vehicles is randomly placed in a 500×500 square area, subject to the following acceptance conditions: (1) the origins of any pair of vehicles must be sufficiently apart so as not to violate the separation constraints; (2) the OD distance should not be too short, i.e., must be greater than 150. Vehicle maximum speeds are randomly sampled from $\{8, 12, 16\}$. Throughout the experiments, we set $T = 3$, $h = 1$, $\beta = 100$, $b_1 = 1$ (except in Section 7.2) and $b_2 = 1$. Video demonstration of some experimental results can be found at https://youtu.be/vgHVcQ_i5RU.

7.1. The price of semi-cooperativeness

We want to understand how the number of agents in a UTM system affects the traffic performance. To do so, we first apply the SEMICO framework to three scenarios having 10, 30 and 50 vehicles, respectively. Each scenario is run multiple times with different numbers of agents. In multi-agent cases, each agent is set to control the same number of vehicles. The vehicle-to-agent assignment is made sequentially. For example, in the 10-vehicle scenario when five agents are tested, each agent controls two vehicles: a1 controls vehicles 1 and 6, a2 controls vehicles 2 and 7, and so forth. Fig. 6 demonstrates the traffic scene at time 1 (i.e., vehicles' starting positions and directions) as well as the trajectory traces generated from the one-agent solutions. Full results are summarized in Table 2. The column headers are explained as follows. Detour and Delay are the (actual/oracle) type of ratios to quantify efficiency. Specifically, Detour is calculated as the actual travel distance of all vehicles divided by the total straight-line OD distance; Delay is calculated as the total travel time of all vehicles

divided by the total time distance between ODs, whereas the latter is calculated assuming the vehicles travel at their respective maximum speeds along the straight OD line. Higher values of Detour and Delay indicate lower traffic efficiency. nViol is the number of violations to the inter-vehicle separation requirement accrued during the entire routing process (i.e., sum of δ_{ij} over all $(i,j), i < j$ and t), and pViol is the maximum proportion of violation ever observed (i.e., max of ρ_{ij} over all $(i,j), i < j$ and t). aCPU and mCPU are the average and maximum computing time to solve an agent's instance of SEMICO, in seconds. Note that different agents can solve their respective SEMICO problems simultaneously in different threads, that is why the per-agent computing time is relevant here. nPriority and nNegate are the total number of invocations of Step 2(a) and Step 2(b), respectively, in the deadlock resolution procedure.

Key observations include: (1) The SEMICO model is generally effective—all cases were solved successfully and only a few minor separation violations occurred at relatively high traffic density. (2) For the same traffic scenario, using more agents leads to faster routing decisions but also reduces the overall traffic efficiency. This reflects the typical tradeoff between efficiency and flexibility in the design of UTM systems. Vehicles are *fully cooperative* when they are controlled by the same agent, and are *semi-cooperative* when controlled by different agents. We call the reduced traffic efficiency caused by the increased number of agents the *price of semi-cooperativeness*, which is an important factor to consider in the design of UTM systems. As the experiments have shown, the SEMICO framework is able to cope with the whole efficiency-flexibility spectrum, that is, for the same traffic scenario, any agent-vehicle assignment scheme can be accommodated. (3) Deadlocks arise, hence must be resolved, more frequently when more vehicles are controlled by each agent, and when the traffic density is high. Deadlocks, by design, will not arise in decentralized routing scenarios where each agent controls one vehicles.

The SEMICO framework can also be used for decomposing the time-consuming task of routing a large fleet of vehicles into multiple smaller tasks to be handled in parallel by a choice number of routing agents. For the above instance, routing 50 vehicles in a dense traffic can take up to 13.6 s per iteration, which may be unbearably long for some applications. If these vehicles were assigned to five routing agents, then the per-iteration time could be reduced to less than one second. This would amount to a 74% efficiency loss (i.e., Detour from 1.16 to 2.02) in exchange for a 18x speedup, a worthwhile trade in certain applications.

To make a more vivid contrast between centralized and decentralized routing outcomes, we make a stylized example, in which 20 vehicles start in a ring-shaped formation and each vehicle's destination is the opposite point along the diameter of the ring, see Fig. 7. We run two extreme configurations. One configuration has a single agent centrally controlling all vehicles, and the other configuration has 20 agents, each controlling a single vehicle. The resulting flight traces are visualized in Fig. 8. The upper part of the figure shows the centralized case and the lower part shows the decentralized case. The left plot of each case flattens the temporal dimension so each distinct point in the plot represents the 2D location of a vehicle (distinguished by color) at a time point; the right plot of each case presents the continuous time dimension along the vertical axis to demonstrate the non-overlapping 3D trajectories over space and time. We can see that when the traffic is centrally managed, the available space is exploited more aggressively—vehicles are packed tightly together when passing through congested areas. This is because no additional buffer space is needed to account for external uncertainty. In contrast, when vehicles route themselves independently in a busy traffic, more inter-vehicle distance is maintained, leading to less efficient use of the airspace.

7.2. Adjusting the safety buffer to cope with uncooperative vehicles

Fleet managers that use the SEMICO model to route their fleets are not required to assume that other vehicles in the airspace are subject to the same routing (behavioral) model. In reality, there may be free-flying vehicles, or vehicles that simply follow pre-determined flight paths with little regard to other airspace activities. As long as these vehicles emit their real-time location coordinates (e.g., via the remote ID infrastructure), SEMICO agents are able to cope with them. To demonstrate this capability, we introduce several uncooperative vehicles into a random 30-vehicle traffic scene. While the 30 vehicles are being routed by the SEMICO framework, the uncooperative vehicles simply travel along straight lines to their destinations at full speed, not engaging in active deconfliction. It is then totally up to the SEMICO agents to detect and avoid these "blind vehicles".

We experimented with three values for the inter-fleet buffer time b_1, i.e., $b_1 \in \{1,2,3\}$, and four values for the number of blind vehicles (Bv) introduced into the traffic scene, i.e., 2, 4, 6 and 8. The main metrics of interest include the loss of separation metrics nViol and pViol, and the traffic efficiency metrics Detour and Delay, when the traffic is managed by different numbers of agents (Ag). The outcomes are listed in Table 3, organized in a way to accommodate comparisons from different viewpoints. We make two observations here. First, in all cases increasing the buffer b_1 is an effective method to mitigate the risk brought by uncooperative vehicles. In particular, when $b_1 = 3$ any number (up to 8) of uncooperative vehicles failed to cause loss of separation, regardless of how decentralized the routing process is. Second, centralized routing (Ag = 1) is more capable of coping with uncooperative external vehicles than multi-agent routing. Everything else held equal, centralized routing results in fewer loss-of-separation incidences and smaller detour and delay factors.

7.3. Multi-agent routing performances at different densities

We test the routing performance using traffic scenarios consisting of 30 to 100 vehicles, whereas for each scenario 20 random cases were generated. We normalize the workload of agents across different scenarios by assigning 10 vehicles to each agent. For example, three agents were utilized for the 30-vehicle scenarios, four agents were utilized for the 40-vehicle scenarios, and so forth. A total of 160 cases were tested, having a total mission distance of about 3.1 million meters. All these cases were successfully solved, with an average per-agent per-iteration computing time of 0.38 s. Table 4 lists summary statistics by the number of vehicles, including the mean (Mn) and standard deviation (SD) of detour, delay and average per-agent computing time in seconds. The Max CPU column lists the maximum per-agent computing time (in seconds) ever observed in all the 20 test cases, so it reliably reflect the upper bound of computing time one can expect for an instance of the given size. Fig. 9 plots the detour factor and computing time v.s. traffic density. The traffic density is defined as the total area of vehicles' safety discs divided by the area of the traffic scene, $500 \times 500 = 250,000$. We can see that traffic density causes routing inefficiency (Detour) to grow almost linearly, while the average computing time exhibits an accelerating growth as density increase. The delay v.s. density relation is more or less of the same pattern as the relation between detour and density, so its plot is omitted here.

8. Conclusion

While centralized traffic management is most conducive to efficient point-to-point air trips, the practical reality demands a more flexible architecture in which multiple private fleets can operate independently, yet harmoniously, in a shared airspace. Sufficient separation is the most fundamental operating requirement shared among all airspace users, but how to maintain the desired level of separation in a decentralized operation environment is a core challenge to be addressed. This paper

Table 2
Solution summary of different multi-agent routing scenarios.

Vehicles	Agents	Detour	Delay	nViol	pViol	aCPU	mCPU	nPriority	nNegate
10	1	1.09	1.31	0	0	0.12	0.27	44	2
	2	1.11	1.31	0	0	0.1	0.19	31	0
	5	1.25	1.53	0	0	0.1	0.28	0	0
	10	1.24	1.51	0	0	0.09	0.22	0	0
30	1	1.08	1.31	0	0	0.94	11.11	90	4
	2	1.53	1.91	0	0	0.24	1.63	175	7
	5	1.65	2.03	0	0	0.13	0.43	132	0
	10	1.71	2.11	0	0	0.12	0.42	38	0
	15	1.89	2.35	0	0	0.12	0.45	13	0
	30	1.89	2.3	0	0	0.11	0.43	0	0
50	1	1.16	1.47	0	0	2.89	13.58	364	48
	2	1.84	2.25	0	0	0.71	3.93	519	23
	5	2.02	2.55	0	0	0.22	0.74	356	6
	10	2.34	3.01	0	0	0.19	0.44	217	1
	25	2.26	2.88	0	0	0.17	0.66	48	0
	50	2.14	2.88	0	0	0.16	0.51	0	0
70	1	1.2	2.14	0	0	5.74	47.08	753	417
	2	2.19	3.07	0	0	1.91	13.12	1145	247
	7	3.08	4.53	0	0	0.41	2.09	866	11
	10	3.37	4.9	0	0	0.38	1.04	907	1
	35	3.36	5.07	1	0.01	0.32	0.71	240	1
	70	3.43	5.14	0	0	0.31	0.79	0	0
100	1	1.18	1.78	0	0	29.1	248.92	951	343
	2	2.06	2.83	0	0	6.79	42.22	1729	188
	5	2.79	4.07	0	0	1.14	7.54	1780	41
	10	3.29	4.94	0	0	0.76	2.18	1393	18
	20	3.52	5.18	3	0.09	0.68	1.29	837	10
	50	3.47	5.11	22	0.26	0.59	1.31	230	1

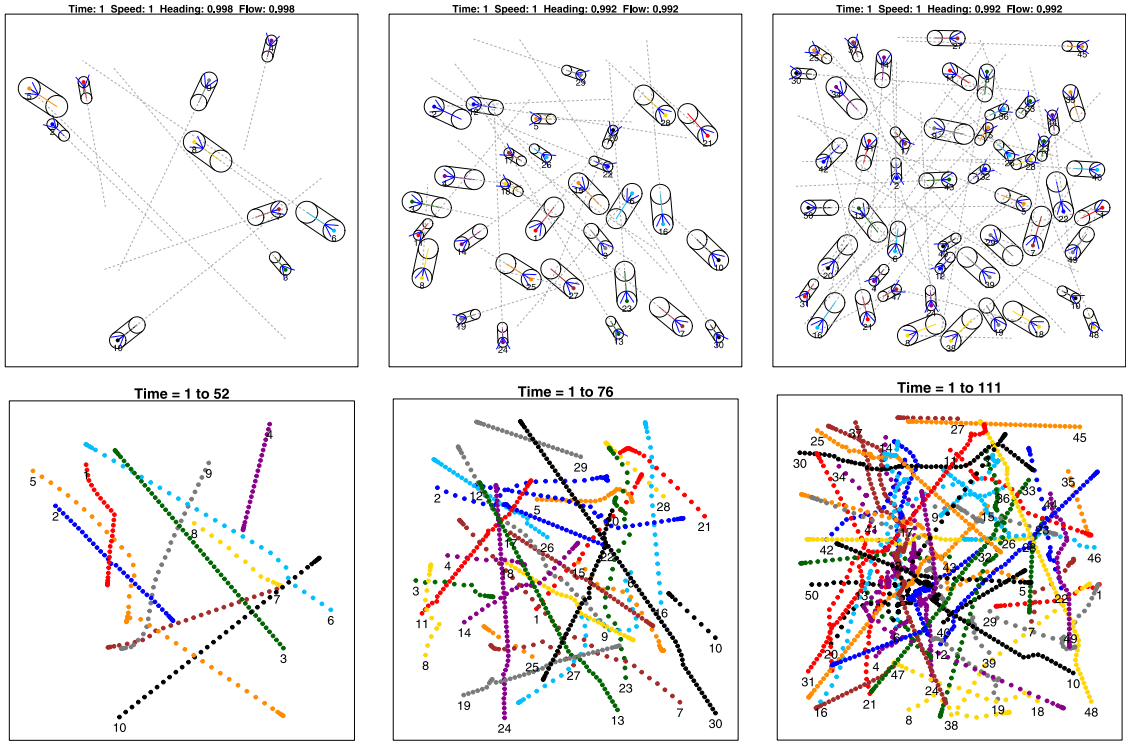

Fig. 6. Traffic scenarios consisting of 10, 30 and 50 vehicles (upper) and trajectory traces generated by the centralized (one-agent) solution.

has demonstrated a remarkably simple, scalable and effective solution to this problem, predicated on limited information sharing enabled by the remote ID protocol. Simply put, each fleet internally solves its own routing optimization problem, and by observing some global ordering convention, such as ordering vehicles alphabetically by their unique IDs, inter-fleet trajectory conflicts can be resolved most of the time—no complicated inter-fleet communication or negotiation is needed, and no assumption about other fleets' behavior is required.

The routing optimization framework boasted several elements designed to work in practical software, including the corridor lock for coping with locational variance and contingent conditions, the caterpillar-like heuristic for finding useful feasible solutions, the Lagrangian-based

Table 3
Performance of the routing system's response to uncooperative vehicles in traffic.

	Ag/Bv	$b_1 = 1$				$b_1 = 2$				$b_1 = 3$			
		2	4	6	8	2	4	6	8	2	4	6	8
nViol	1	0	0	0	12	0	0	0	2	0	0	0	0
	2	3	7	0	0	0	0	0	1	0	0	0	0
	5	0	2	2	2	0	0	2	0	0	0	0	0
	10	0	1	3	2	0	0	0	0	0	0	0	0
	15	0	2	2	5	0	0	0	0	0	0	0	0
	30	0	10	5	1	0	0	0	0	0	0	0	0
pViol	1	0	0	0	0.79	0	0	0	0.15	0	0	0	0
	2	0.26	0.20	0	0	0	0	0	0.15	0	0	0	0
	5	0	0.06	0.06	0.06	0	0	0.18	0	0	0	0	0
	10	0	0.01	0.23	0.01	0	0	0	0	0	0	0	0
	15	0	0.06	0.06	0.16	0	0	0	0	0	0	0	0
	30	0	0.21	0.26	0.01	0	0	0	0	0	0	0	0
Detour	1	1.13	1.18	1.25	1.26	1.19	1.28	1.37	1.44	1.26	1.39	1.56	1.60
	2	1.47	1.52	1.68	1.50	1.83	2.03	1.94	1.78	2.36	2.21	2.24	2.17
	5	1.63	1.84	1.86	1.87	2.52	2.54	2.26	2.46	3.30	3.24	2.96	2.95
	10	1.94	1.95	2.02	2.25	2.67	3.03	2.48	2.87	3.59	3.22	3.42	4.06
	15	1.88	2.09	1.92	2.27	2.62	2.78	2.82	2.75	3.55	4.07	3.31	3.57
	30	1.95	2.03	2.11	2.21	2.46	2.89	2.86	2.52	3.70	3.70	3.47	3.40
Delay	1	1.42	1.52	1.59	1.63	1.58	1.64	1.68	1.83	1.65	1.81	1.88	2.02
	2	1.83	2.00	2.08	1.83	2.36	2.64	2.43	2.31	3.16	2.96	2.83	2.79
	5	2.06	2.29	2.29	2.26	3.33	3.04	2.91	3.06	4.30	4.52	3.70	3.53
	10	2.47	2.42	2.46	2.61	3.61	3.78	3.25	3.54	4.85	4.55	4.32	5.08
	15	2.34	2.55	2.37	2.75	3.37	3.77	3.47	3.40	4.74	5.58	4.31	4.74
	30	2.43	2.51	2.59	2.75	3.08	3.76	3.77	3.18	5.46	4.94	4.64	4.95

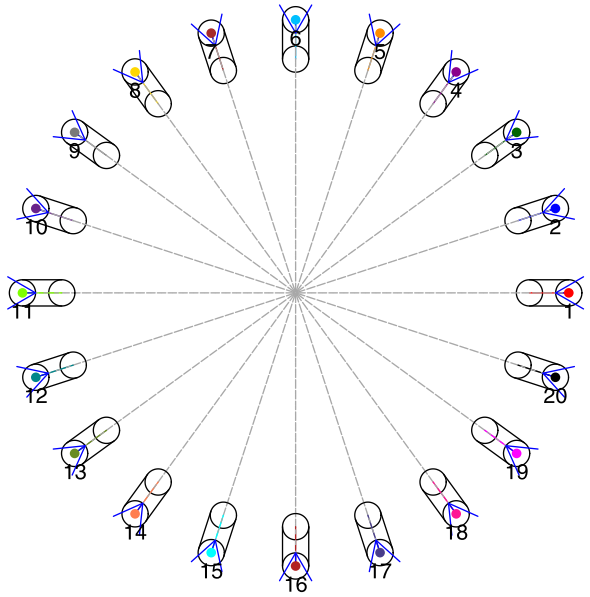

Fig. 7. A stylized scenario with 20 vehicles starting in a ring-shaped formation, each aiming to reach the opposite position on the ring.

Table 4
Summary of randomized computational results.

Vehicles	Detour		Delay		Avg CPU		Max CPU
	Mn	SD	Mn	SD	Mn	SD	
30	1.61	0.08	2.06	0.11	0.25	0.01	1.12
40	2.01	0.17	2.71	0.31	0.29	0.01	1.53
50	2.21	0.25	3.13	0.42	0.35	0.02	2.09
60	2.40	0.15	3.48	0.35	0.42	0.02	5.13
70	2.75	0.22	4.08	0.49	0.52	0.06	4.01
80	2.94	0.23	4.48	0.52	0.53	0.06	4.27
90	3.13	0.29	4.90	0.58	0.68	0.05	5.80
100	3.21	0.31	5.15	0.79	0.87	0.09	16.54

predictive deadlock detection mechanism, and the two-step deadlock resolution strategy. The framework has been validated to work as intended in a variety of simulation experiments.

In both the patent and academic literature, conceptual and theoretical works greatly outnumber evidence-based analyses. However, in an emerging, regulation-bound industry, concrete data and empirical analysis from field trials (or authentic simulations of field operations) may be more valuable and more convincing to regulators and investors alike. Therefore, meaningful future work can include implementing the framework on real UAV fleets to understand and improve its real-world performance, and establishing true-to-life virtual environments to speed up the evaluation of business model hypotheses, concepts of operations and technological innovations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The research is supported in part by the National Science Foundation, USA via grant number 1944068, and in part by the State of Michigan via the Michigan Translational Research and Commercialization (MTRAC), USA grant for advanced transportation. The author would like to thank the associate editor and the two anonymous referees for their constructive suggestions which helped improve the quality of the paper. Zhenyu Zhou helped make Figures 1, 3 and A.10 and his assistance is appreciated.

Appendix. A crafted case for which the caterpillar-like starting point method loses feasibility

The case is demonstrated in Fig. A.10. Two vehicles travel along parallel paths from left to right. At time t_0, the top vehicle is at point A and its planned future locations are B, C and D; the bottom vehicle is at point E and its planned future locations are F, G and H. Their planned speeds are the same (and apparently the top vehicle is not at its maximum speed). Their plans constitute a feasible solution because

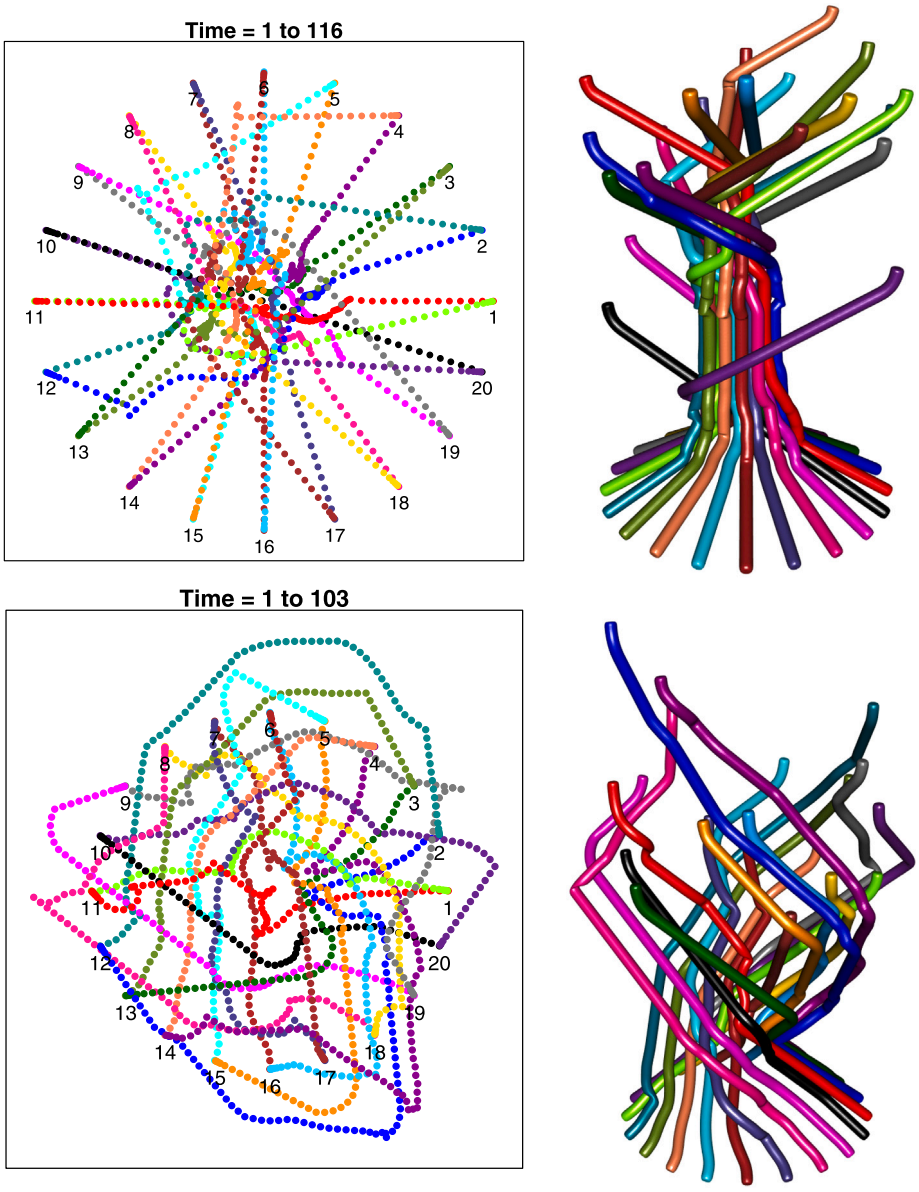

Fig. 8. Flight traces of the 20-vehicle case. Upper: Centralized planning. Lower: Decentralized planning. In the 3D plots, time advances upward along the vertical axis.

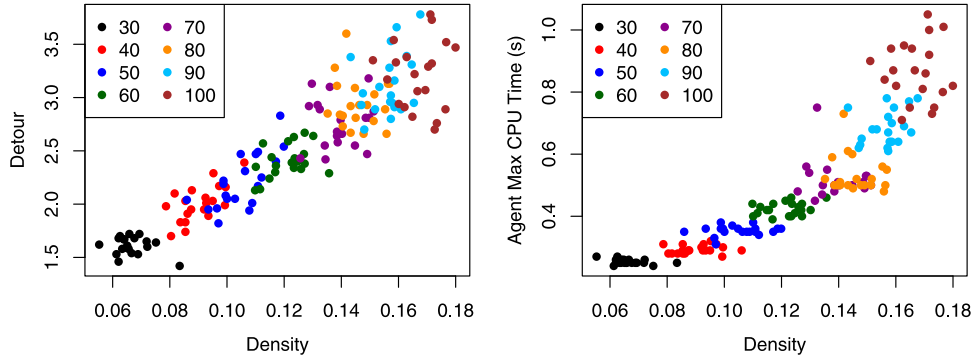

Fig. 9. Effects of traffic density on detour and computing time. Each point is a test case. The number of vehicles is distinguished by the point color, and shown in the legend. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

none of the circles centered at $\{A, B, C, D\}$ overlaps with any of the circles centered at $\{E, F, G, H\}$. Now suppose the vehicles follow their plans and move one step ahead, to point B and point F, respectively. According to the formula (10) and (11), the starting-point plan at t_0+1 for the top vehicle's planned future locations are B', C' and D, and for the bottom vehicle's planned future locations are F', G' and H. We

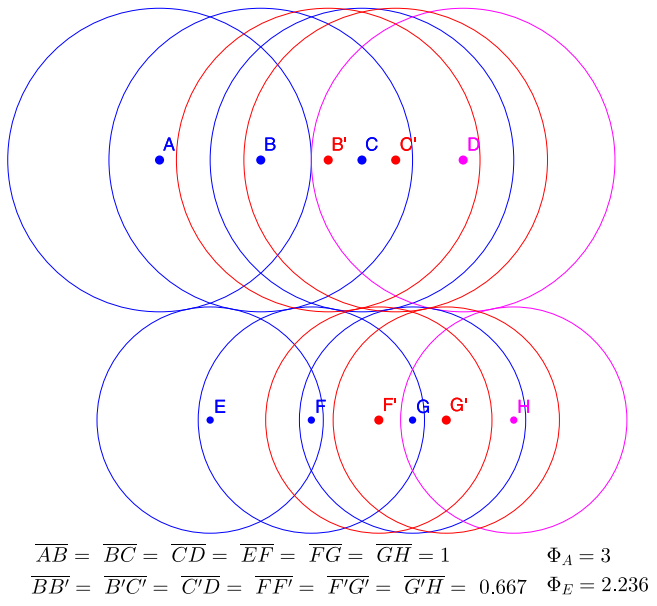

Fig. A.10. A crafted case for which the caterpillar-like starting point method looses feasibility.

can see that the two vehicles' planned locations at certain non-matching future time points actually violate the separation constraints, because the corresponding circles slightly overlap. The example demonstrates that the starting point strategy is just a heuristic. Nonetheless, such an extreme corner case almost never occurs in reality, and even when it does occur, the infeasible starting point should not be a big problem because CONOPT solver's phase 1 is usually quite effective for restoring feasibility from a reasonable non-trivial starting point.

References

Amazon, 2018. Amazon Prime Air. URL: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011. Accessed on March 18, 2020.

ArduPilot Dev Team, 2020. SITL simulator (software in the loop). Online Manual.

ASTM Subcommittee F38.02, 2020. Standard specification for remote ID and tracking.

Augugliaro, F., Schoellig, A.P., D'Andrea, R., 2012. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1917–1922.

Barnhart, C., Bertsimas, D., Caramanis, C., Fearing, D., 2012. Equitable and efficient coordination in traffic flow management. Transp. Sci. 46 (2), 262–280.

Bulusu, V., Sengupta, R., Polishchuk, V., Sedov, L., 2017. Cooperative and non-cooperative UAS traffic volumes. In: 2017 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 1673–1681.

Chan, A.K., Jesse R. Cheatham, I., Duncan, W.D., Hwang, E.Y., Hyde, R.A., Pan, T.S., Tegreene, C.T., Wood, V.Y.H., 2016. System and method for management of airspace for unmanned aircraft. Elwha LLC, U.S. Patent US9508264B2.

Chen, Y., Cutler, M., How, J.P., 2015. Decoupled multiagent path planning via incremental sequential convex programming. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). pp. 5954–5961.

Chin, C., Gopalakrishnan, K., Evans, A., Egorov, M., Balakrishnan, H., 2020. Tradeoffs between efficiency and fairness in unmanned aircraft systems traffic management. In: 9th International Conference on Research in Air Transportation.

Chung, S.H., Sah, B., Lee, J., 2020. Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. Comput. Oper. Res. 123, 105004.

CONOPT, 2020. CONOPT Solver Manual in GAMS Documentation.

Dechering, M.J., 2019. Traffic management of small-unmanned aerial systems in an urban environment. (Ph.D. thesis). University of Cincinnati, p. 97,

Dell'Olmo, P., Lulli, G., 2003. A new hierarchical architecture for Air Traffic Management: Optimisation of airway capacity in a Free Flight scenario. European J. Oper. Res. 144 (1), 179–193.

Drud, A.S., 1994. CONOPT - A large-scale GRG code. ORSA J. Comput. 6 (2), 207–216.

Dupray, D.J., LeBlanc, F.W., 2020. Unmanned aerial vehicles. US Patent US10586464B2.

Erzberger, H., 2004. Transforming the NAS: The next generation air traffic control system. NASA/TP–2004-212828.

Erzberger, H., Heere, K., 2010. Algorithm and operational concept for resolving short-range conflicts. Proc. Inst. Mech. Eng. 224, 225–243.

Federal Aviation Administration, 2011. Introduction to TCAS II version 7.1.

Federal Aviation Administration, 2019. Remote identification of unmanned aircraft systems. Technical Report, (2019–28100), Federal Register.

Federal Aviation Administration, 2020. UAS Remote Identification. https://www.faa.gov/uas/research_development/remote_id/.

Frazzoli, E., Mao, Z.-H., Oh, J.-H., Feron, E., 2001. Resolution of conflicts involving many aircraft via semidefinite programming. J. Guid. Control Dyn. 24 (1).

Google Inc., 2020. Google UAS airspace system overview. Accessed June 2020.

Ho, F., Geraldes, R., Gonçalves, A., Rigault, B., Sportich, B., Kubo, D., Cavazza, M., Prendinger, H., 2020. Decentralized multi-agent path finding for UAV traffic management. IEEE Trans. Intell. Transp. Syst. 1–12.

Hoekstra, J., Ruigrok, R.C., van Gent, R.N., 2011. Free flight in a crowded airspace? In: Air Transportation Systems Engineering. American Institute of Aeronautics and Astronautics, Inc., pp. 533–545.

Jang, D.-S., Ippolito, C.A., Sankararaman, S., Stepanyan, V., 2017. Concepts of airspace structures and system analysis for UAS traffic flows for urban areas. In: AIAA Information Systems-AIAA Infotech @ Aerospace.

Jiang, T., Geller, J., Ni, D., Collura, J., 2016. Unmanned aircraft system traffic management: Concept of operation and system architecture. International Journal of Transportation Science and Technology 5 (3), 123–135.

Jin, Z., Zhao, Z., Luo, C., Basti, F., Solomon, A., Gursoy, M.C., Caicedo, C., Qiu, Q., 2019. Simulation of real-time routing for UAS traffic management with communication and airspace safety considerations. In: 2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC). pp. 1–10.

Klooster, J.K., Ren, L., Hochwarth, J.K.U., 2014. Method and system for aerial vehicle trajectory management. European Patent EP2503530B1.

Kochenderfer, M., Holland, J., Chryssanthacopoulos, J., 2012. Next generation airborne collision avoidance system. Linc. Lab. J. 19, 17–33.

Kopardekar, P., 2019. Unmanned aircraft systems traffic management. U.S. Patent US10332405B2.

Kopardekar, P., Rios, J., Prevot, T., Johnson, M., Jung, J., III, J.E.R., 2016. Unmanned aircraft system traffic management (UTM) concept of operations. In: 16th AIAA Aviation Technology, Integration, and Operations Conference.

Liu, Y., 2019. A progressive motion-planning algorithm and traffic flow analysis for high-density 2D traffic. Transp. Sci. 53 (6), 1501–1525.

Liu, Y., 2020. A note on solving DiDi's driver-order matching problem. Optim. Lett. 1–17.

McCarthy, T., Pforte, L., Burke, R., 2020. Fundamental elements of an urban UTM. Aerospace 7 (7: 85).

Mukherjee, A., Hansen, M., 2009. A dynamic rerouting model for air traffic flow management. Transp. Res. B 43 (1), 159–171.

Otto, A., Agatz, N., Campbell, J., Golden, B., Pesch, E., 2018. Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. Networks 72 (4), 411–458.

Pallottino, L., Feron, E., Bicchi, A., 2002. Conflict resolution problems for air traffic management systems solved with mixed integer programming. IEEE Trans. Intell. Transp. Syst. 3 (1), 3–11.

Prevot, T., Rios, J., Kopardekar, P., III, J.E.R., Johnson, M., Jung, J., 2016. UAS traffic management (UTM) concept of operations to safely enable low altitude flight operations. In: 16th AIAA Aviation Technology, Integration, and Operations Conference.

Rios, J., Aweiss, A., Jung, J., Homola, J., Johnson, M., Johnson, R., 2020. Flight demonstration of unmanned aircraft system (UAS) traffic management (UTM) at technical capability level 4. In: AIAA AVIATION 2020 Forum.

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., Abbeel, P., 2014. Motion planning with sequential convex optimization and convex collision checking. Int. J. Robot. Res. 33 (9), 1251–1270.

Shaw, V., Cu, Z., Dowlatkhah, S., 2018. Intelligent drone traffic management via radio access network. U.S. Patent US9940842B2.

Tan, Q., Wang, Z., Ong, Y.-S., Low, K.H., 2019. Evolutionary optimization-based mission planning for UAS traffic management (UTM). In: 2019 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 952–958.

Yang, X., Egorov, M., Evans, A., Munn, S., Wei, P., 2020. Stress testing of UAS traffic management decision making systems. In: AIAA AVIATION 2020 FORUM.

Zhang, W., Kamgarpour, M., Sun, D., Tomlin, C., 2012. A hierarchical flight planning framework for air traffic management. Proc. IEEE 100 (1), 179–194.

Zhao, Z., Luo, C., Zhao, J., Qiu, Q., Gursoy, M.C., Caicedo, C., Basti, F., 2019. A simulation framework for fast design space exploration of unmanned air system traffic management policies. In: 2019 Integrated Communications, Navigation and Surveillance Conference (ICNS). pp. 1–10.

Zhong, C., Zhao, Z., Luo, C., Gursoy, M.C., Qiu, Q., Caicedo, C., Basti, F., Solomon, A., 2020. A cost-benefit analysis to achieve command and control (C2) link connectivity for beyond visual line of sight (BVLOS) operations. In: 2020 Integrated Communications Navigation and Surveillance Conference (ICNS). pp. 2D1–1–2D1–14.

Zhou, Z., Chen, J., Liu, Y., 2020. Optimized landing of drones in the context of congested air traffic and limited vertiports. IEEE Trans. Intell. Transp. Syst. 1–11.

Zhu, G., Wei, P., 2019. Pre-departure planning for urban air mobility flights with dynamic airspace reservation. In: AIAA Aviation 2019 Forum.