A Strategic Approach for Dense, Integrated, Vehicle Navigation

Mahyar R. Malekpour
NASA Langley Research Center, Hampton, Virginia 23681

Drone usage has been on the rise in recent years with applications that include parcel delivery, wildlife protection, precision farming, law enforcement, and industrial inspection, just to name a few. This paper addresses drone separation in time and distance for high volume corridors (en route) and lanes (on ground), merging as well as crossing intersections of multiple corridors/lanes. There exists a gamut of approaches to solving merging and intersection crossing problems. At one end of the extreme are the conservative yet low cost and verifiable solutions of today that deal with two drones at a time. At the other end are complex Machine Learning-based solutions with high computing requirements for fully autonomous drones of the future that are expected to handle all contentions. This paper presents a feasible and verifiable strategic approach to solving these problems that is based on distributed cooperation between the UAVs/UASs and the infrastructure. Unlike existing centralized and pre-planned approaches, the proposed solution is fully distributed and enables autonomous aircraft to decide to adjust their speed and distance with respect to the preceding aircraft, dynamically. Three phases of the strategic approach (*Prepare, Adjust, Commit*) are presented. Simulation results are presented that show the proposed approach is stable and resilient to induced perturbations and guarantees a set of fixed-wing UAVs/UASs to merge and cross intersections by adjusting their speed based on their distance to the aircraft in front of them.

Nomenclature

ACC	= Adaptive Cruise Control
ATM	= Air Traffic Management
CACC	= Cooperative ACC
CMS	= Controller Managed Spacing
FAA	= Federal Aviation Administration
IM	= Interval Management
NAS	= National Airspace System
NASA	= National Aeronautics and Space Administration
TMA-TM	= Traffic Management Advisor with Terminal Metering
UAV	= Unmanned Aerial Vehicle
sUAS	= Small Unmanned Aircraft System
V2I	= Vehicle-to-Infrastructure
V2V	= Vehicle-to-Vehicle

I. Introduction

Drone usage has been on the rise in recent years with applications that include parcel delivery, wildlife protection, precision farming, law enforcement, and industrial inspection, just to name a few. Once regulations and safety policies are put in place to allow for the widespread use of unmanned drones, the number of aircraft in the National Airspace System (NAS) is expected to skyrocket to millions, potentially congesting the airspace which increases the likelihood of separation violations and possibly incidents[1]. Currently, flight infrastructure can only support a few thousand aircraft flying over the United States National Airspace System (NAS) at any given time. A delay at one airport can send ripple effects throughout the system, causing more delays and missed connections. In air traffic control, separation is the concept of keeping an "ownship" aircraft outside a minimum distance from "intruder" aircraft to reduce the risk of the aircraft colliding, as well as preventing accidents due to secondary

factors, such as wake turbulence. Maintaining proper separation is often a safety critical property for fixed-wing drones in the airspace. This paper addresses drone separation in time and distance for high volume corridors (en route) and lanes (on ground), merging as well as crossing intersections of multiple corridors/lanes. In this paper, the term drone is applied to both Unmanned Aerial Vehicle (UAV) and small Unmanned Aircraft System (UAS) vehicles operating autonomously. There exists a gamut of approaches to solving merging and intersection crossing problems. At one end of the extreme are the conservative yet low cost and verifiable solutions of today that deal with two drones at a time. At the other end are complex Machine Learning-based solutions with high computing requirements for fully autonomous drones of the future that are expected to handle all contentions. This paper presents a feasible and verifiable strategic approach to solving these problems that is based on distributed cooperation between the drones and the infrastructure. Three phases of the strategic approach (*Prepare*, *Adjust*, *Commit*) are presented. Simulation results are presented that show the proposed approach is stable and resilient to induced perturbations and guarantees a set of fixed-wing drones to merge and cross intersections by adjusting their speed based on their distance to the aircraft in front of them while remaining in the equilibrium state. The equilibrium state is defined as the state when a set of n aircraft move at a relatively constant speed and uniform spacing from each other in a congested system. A congested system is defined as the state when at least one aircraft cannot move at its maximum allowed speed. Unlike existing centralized and pre-planned approaches, the proposed solution is fully distributed and enables autonomous aircraft to decide to adjust their speed and distance with respect to the preceding aircraft, dynamically.

Besides safe and timely arrival at a terminal, merging is also needed en route for the vehicles to navigate areas with reduced numbers of lanes/corridors and bypassing restricted areas before resuming their original paths. We use the term lane (L) to generically mean a corridor in the sky, a road, or lane of a highway on the ground. For a vehicle to merge between two vehicles in another lane there needs to be sufficient spacing between those two vehicles. Similarly, for a lane with multiple vehicles to merge into another lane, spacing between multiple vehicles in both lanes needs to be sufficiently increased to accommodate safe, seamless merging. There are many approaches to merging problem ranging from ad hoc to strategically preplanned.

Ad hoc approach may be a practical method for non-safety critical applications or when the number of vehicles in the lanes is low. In this merging approach, the vehicles are left to sort out the merging sequence among themselves, typically, at or near the merge point as is often done on ground-based roads. An important consideration is that ground-based vehicles can always just stop.

Strategically planned approaches have been proposed for safety and time critical applications, or when the number of vehicles in the lanes is high, where merging is based on a predetermined schedule, for instance, of time of arrival of the vehicles in each lane at the merge point[2-4]. Time-Based Metering is defined as a method of restricting aircraft flow by scheduling the time at which each aircraft should cross a predetermined waypoint or a runway terminal. However, despite extended strategic preplanning, a need for a tactical, last-minute solution persists. Unforeseen events and off-nominal conditions, e.g., weather, can potentially disrupt timing requirements, thus necessitating a need for a tactical, last-minute solution.

The Airborne Spacing for Terminal Arrival Routes (ASTAR), developed at the National Aeronautics and Space Administration (NASA), is a strategic approach that enables the use of efficient arrival procedures during high demand operations on a Next Generation Air Traffic Management (NextGen) environment. Three NASA technologies include: the Traffic Management Advisor with Terminal Metering (TMA-TM), which provides precise time-based scheduling in the terminal airspace; Controller Managed Spacing (CMS), which provides controllers with decision support tools that enable precise schedule conformance; and Interval Management (IM), which consists of flight deck automation that enables aircraft to either achieve or maintain a precise spacing interval behind the preceding aircraft[6,7].

Using precision scheduling, intersection crossing and merging can be anticipated and planned predeparture. However, predetermined scheduling solutions do not readily scale up for the crowded skies of the future with large numbers of aircraft even if they are unmanned or autonomous drones. Comprehensive preplanning requires extended knowledge of all flight plans, which is not available in the current NAS, and high computing power to plan safe routes, schedule time of arrivals, and meet everyone's requirements making scheduling a daunting and unfeasible task for a very large number of drones.

The advent of automotive Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) technologies, as well as an increasing interest in self-driving vehicles has sparked a lot of interest in the problem of safely conducting vehicle traffic at intersections as well as merging and spacing of the vehicles among automotive researchers. Various proposed techniques to manage vehicle traffic through intersections can generally be classified into centralized, distributed, and decentralized approaches[5,8-14].

Centralized methods rely on a single entity, often located at an intersection, to enable coordination using various schemes such as scheduling, where the entity is also relied upon to collect data from the vehicles (location, speed, etc.) and to disseminate the data to all vehicles[12-14]. Distributed approaches rely solely on the vehicles to coordinate amongst themselves as to how to navigate through an intersection or safely merge into a lane, e.g., ad hoc solutions. Decentralized approaches rely on the vehicles to navigate themselves while still relying on a centralized entity to control sequencing and scheduling. Thus far, the proposed solutions seem to limit the scope of the problem to one intersection and do not address the larger, more complex system of multiple intersections.

Safely and efficiently navigating vehicles in a system consisting of a network of lanes requires consistent and accurate data (location, speed, trajectory, etc.) on all vehicles. The accordion effect in ground traffic can be a safe solution, but it is not an efficient one. Also, fixed-wing aircraft have a non-zero minimum acceptable speed, and so safety is a paramount consideration. Since in this paper we only consider a system of vehicles operating in two dimensions (2-D), hereafter, we use the term vehicle to mean aircraft, drones, and cars. When the system is dynamic, and as the vehicles enter and exit the lanes or intersections, consistency of data (at any given time) requires large amount of real-time communication by the vehicles, which in turn introduces new challenges. In a distributed system of n vehicles, as n increases linearly, communication increases exponentially.

We introduce a decentralized approach where the task of managing traffic flow is divided into two entities: a distributed regional entity and the vehicle. A regional entity monitors and manages traffic flow dynamically and as vehicles enter/exit areas by adjusting the range of the merge zone, dynamically updating maximum speed within its merge zone, informing the vehicles of the recent adjustments to the zone and maximum speed by broadcasting short commands to the vehicles. A regional entity also coordinates its activities by communicating its status with its peers for traffic management further down the line. We present how one such regional infrastructure manages traffic flow in its region.

An outline of this paper is as follows. After presenting background, analysis of the Equilibrium algorithm is presented in Section 3, followed by a description of the generalized Equilibrium algorithm in Section 4. In Section 5, a strategic approach to solving merging and intersection crossing problems is introduced. Simulation results are presented in Section 6 that assess the feasibility of the approach followed by concluding remarks in Section 7.

II. Background

In [15] we described platoons and systems equipped with CACC where vehicles operate at a given speed for better throughput with the lead vehicle operating at a certain speed, which is determined beforehand[16]. We also presented a solution that, in contrast, does not make such assumptions. In other words, vehicles do not: (1) follow a leader in trying to match its speed as in a platoon, (2) try to achieve or maintain a pre-specified intervehicle spacing, (3) share their data with each other (no V2V communication), (4) operate based on a time schedule, and (5) have to be homogeneous. However, the vehicles are assumed to be equipped with sensors to determine their distance from the vehicle in front them. Although the vehicles adhere to a maximum speed limit set for the corridor, their eventual speed, when the system reaches equilibrium, is not predetermined and is not known to the vehicles beforehand; it is simply a function of the length of the corridor and number of the vehicles in the corridor. With minimal information at a vehicle, we presented a solution that guarantees all vehicles eventually operate at an equilibrium which is defined as traveling at a relatively constant speed and similar distances with respect to each other with traffic flowing steady and smoothly.

In this paper we present results of the analysis of the equilibrium algorithm, mainly, that (a) it is string stable, (b) the number of steps (amount of time) it takes to bring a system of n vehicles to equilibrium is no more than n^2, and (c) it lends itself to solving merging (including multiple lanes) and intersection crossing problems.

III. Analysis of Equilibrium Algorithm

For reference, we first present the algorithm. The Equilibrium algorithm is studied in the discrete domain which helps with direct transition of the algorithm to simulation and ease of analysis. We also consider this to be a base-case and assume the vehicles to be identical and leave the generalization to heterogenous vehicles to future work.

Parameters

The following parameters are used in the algorithm:

- $H > 0$, length of the corridor,
- N_i, a vehicle,
- $l \geq 0$, length of a vehicle,
- $S_{max} > 0$, maximum speed of vehicles,
- $S_{stall} > 0$, stall speed,
- $d_{safe} \geq 0$, minimum safe intervehicle distance,
- $d_{i-1,i}(t) \geq 0$, distance between N_{i-1} and N_i, where N_{i-1} is in front of N_i,
- $n \geq 0$, number of vehicles, $n < H/(l+ d_{safe})$,
- $k > 1$, gain factor, and
- $d_i(t) \geq 0$, distance N_i moves at time t, $d_i(t) = min((d_{i-1,i}(t)- d_{safe})/k, S)$.

Assumptions

- Vehicles are equipped with sensors to accurately measure their distance to the front vehicle.
- Vehicles execute the Equilibrium algorithm.
- Vehicles operate within similar speed range. We leave analysis of the general case to future work.

Requirements

- **Safe Spacing**: The minimum safe intervehicle spacing, d_{safe}, is maintained.
- **No Stall**: Vehicle's speed does not fall below a non-zero stall speed, S_{stall}.

The Equilibrium Algorithm

The algorithm as executed by N_i, for all i:

```
while (true)
{
   d_{i-1,i}(t-1) = Loc_{i-1}(t-1) - Loc_i(t-1)        // expected value ≥ d_safe
   d_i(t) = min((d_{i-1,i}(t-1) - d_safe)/k, S)        // expected value ≥ 0
   Loc_i(t) = Loc_{i-1}(t-1) + d_i(t), for all i       // expected value ≥ d_safe
}
```

Definitions:

Vehicle: N_i, a vehicle, either flying in sky (drone) or rolling on the ground (car).

Corridor: H, a one-way corridor in the sky, or a single-lane ground-based road, with a given length.

System: $\{H, n\}$, i.e., a corridor, H, with a given number of vehicles, n.

Congested: A system is said to be congested when N_i cannot navigate at the maximum allowed speed S. More formally, when $ceiling(H/(l+S)) \leq n < H/(l+d_{safe})$. For example, given $l = 4$, $S = 10$, and $H = 200$, $n \geq 15$ will result in a congested corridor. For a congested system, at equilibrium, $d_{i-1,i}(t) = (H - n(d_{safe}+l))/n \pm \varepsilon/2$, where $0 < \varepsilon << 1$.

Threshold: $n_{threshold}$, the upper bound on the number of vehicles that can navigate at their maximum allowed speed, i.e., the value of n above which the system is said to be congested. More formally, when $n_{threshold} = ceiling(H/(l+S))$.

Sparse State: A state when the number of vehicles is at or below the $n_{threshold}$.

Steady State: In a non-congested system, $n < n_{threshold}$, when the vehicles navigate at the maximum allowed speed, S, and at constant spacing from each other but potentially at different intervehicle spacing, the system is said to be in *steady state*. More formally, when $n \leq floor(H/(l+S))$.

Equilibrium

State: When n vehicles in a congested system navigate at a relatively constant speed and uniform intervehicle spacing from each other, the system is said to be in equilibrium state. In equilibrium state, the intervehicle spacing is given 4by:

$d_{i-1,i}(t) = (H - n(d_{safe}+l))/n \pm \varepsilon/2$, where $0 < \varepsilon << 1$.

Number of Steps

Several simplifying assumptions are made: (a) details of physical mechanisms of a vehicle are simplified; (b) the system is abstracted to a set of vehicles navigating in a one-way corridor, where the vehicles enter from one end and exit from another at a steady rate on a straight line by introducing a new vehicle in the corridor within a given time after another vehicle exits the corridor; and (c) two dimensional (2-D) system consisting of vehicles with same length. Furthermore, for simulation purposes, to reduce repeated memory allocation/deallocation for the vehicles entering/exiting the corridor, the exiting vehicle is reintroduced back in the system. Thus, the system is logically modeled as a circle with a set of vehicles.

Simulations are conducted in MATLAB. Table 1 is a summary of simulations of $H = 200$, $d_{safe} = l$, $S = [2 .. 4]$, and $n = 20$ to 25; thus, expecting intervehicle distance of 10 to 8, respectively, at equilibrium. Each simulation is run over 100 times with the vehicles starting at random positions on the circular lane (L) for each run. As shown in Fig. 1 and the last column in the table indicate, the number of steps it takes to reach equilibrium asymptotically approaches n^2. Formal analysis of this result is left to future work. Fig. 1 is a depiction of the simulation results of the first row of Table 1, where for visual purposes each simulation run is shown in a different color. The vertical axis is the minimum distance to the front vehicle, over all vehicles, at the end of each simulation step. The horizontal axis is the number of steps to reach equilibrium. This result lends itself to determining how long it takes for a system to return to equilibrium after experiencing perturbation. (As elaborated in the following sections, this result lends itself to solving merging and intersection crossing problems.)

For a system of $\{H, n\}$, the number of steps it takes to reach the equilibrium state is empirically determined to be less than n^2, within a small epsilon, $\varepsilon = 0.015\%$, of the expected intervehicle distance that is needed to account for oscillation in intervehicle distances imposed by the feedback loop in the algorithm.

Table 1. Simulations of various n.

n	H	Intervehicle distance	num steps < n^2
20	200	10.00	296 .. 372 < 400
21	200	9.52	294 .. 401 < 441
22	200	9.10	262 .. 427 < 484
23	200	8.67	212 .. 449 < 529
24	200	8.33	44 .. 475 < 576
25	200	8.00	81 .. 461 < 625

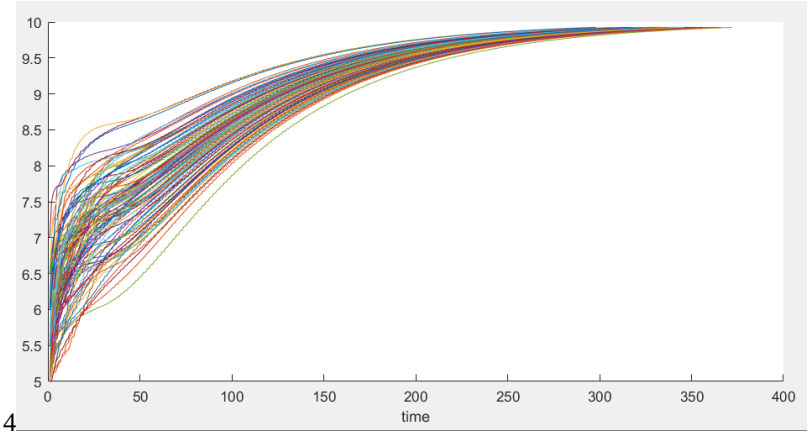


Fig. 1. 100 simulations, n = 20, random initial state, y-axis is maximum intervehicle spacing.

Stability Analysis

Several aircraft following each other are said to form a string. A string is said to be stable if the spacing error does not grow as a function of the vehicle's position in the string, ensuring that small perturbations early in the string do not develop into unacceptable behavior later in the string [7,16].

In [15], analysis of string stability of the algorithm was left as future work which is presented in this section. This analysis is necessary since it is directly related to intervehicle spacing requirements and is elaborated in the following sections.

We assume vehicles are fixed-wing and operate within a minimum and maximum speed without stalling. We believe, an approach that addresses fixed-wing vehicles is also applicable to rotary-wing vehicles that can hover but not necessarily the other way around.

Study of stability is conducted via simulations in MATLAB. Extensive simulations have shown that the proposed algorithm is resilient to perturbation. We define perturbation as a disturbance that causes a change in the behavior of a vehicle for a limited time when system is in the equilibrium state. Perturbations are induced in the system in two different ways: either (a) by decreasing the speed, or (b) by varying the default value of k, for one of the vehicles during the simulations.

The change in k can be either increasing or decreasing from its default value of 2, which in [15] was empirically established to be optimal. Below are examples of a system of $n = 12$ vehicles that was subjected to various perturbations. The effect of change in speed of a vehicle is depicted in Fig. 2a and 2b, where the vehicle's maximum speed was reduced by 50% for a limited time. In Fig. 3a and 3b, k is increased from its default value of 2 to 3 and then back to 2 for a limited time. In Fig. 4a and 4b, k is decreased from its default value of 2 to 1.5 (note: the system does not reach equilibrium when $k < 1.5$) and back to 2. Note that activities of the vehicles are shown in different colors. Simulation results show the equilibrium algorithm is resilient to perturbation of any kind and the system bounces back to equilibrium rather quickly. Furthermore, from the details provided in figures 2 through 5, the farther a vehicle is from the disturbed vehicle, the less it is affected by the perturbation. In other words, perturbation effect diminishes at it trickles down the string. Also, from figures 2 and 3, a decrease in value of k results in speeding up, i.e., adjusting the value of k provides another mechanism to control overall speed of a corridor.

Another way of perturbing a system is to abruptly insert a new vehicle into the string without violating safe intervehicle distances. This type of perturbation is an abstract model of merging a vehicle with a string even though an abrupt insertion of a vehicle is not a desirable behavior as merging is expected to take place over some nonzero time interval. Nonetheless, from figures 5a and 5b, even when a new vehicle is abruptly inserted in the string, the system bounces back to the equilibrium state and, thus, the algorithm remains string stable. Also, as seen in the figures 2 to 5, the perturbation effect diminishes as it reaches farther vehicles down the string. Furthermore, although for perturbations resulting from sudden changes in the speed, or k, the system returned to equilibrium relatively quickly, perturbation due to inserting (adding) a new vehicle into the midst had more lasting effect and, thus, took longer to reach the equilibrium state. This is partly due to severity of the abruptness of the perturbation and partly for its permanent effect (the inserted vehicle remained in the string) which increased the number of vehicles in the string and added demand on the same real state, thus, reducing the intervehicle spacing at the next equilibrium state which affected all vehicles in the system.

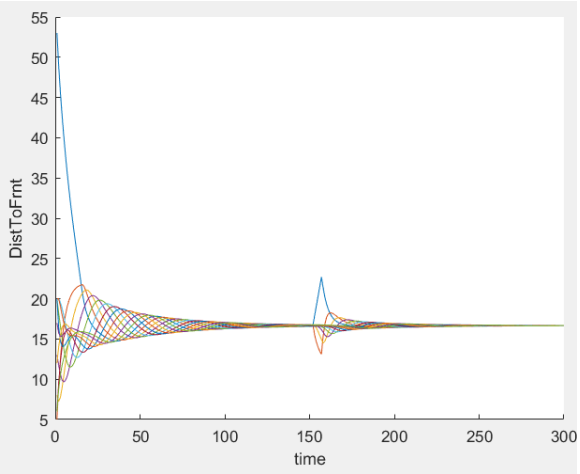


Fig. 2a. Perturbation, max speed 10 to 5 to 10.

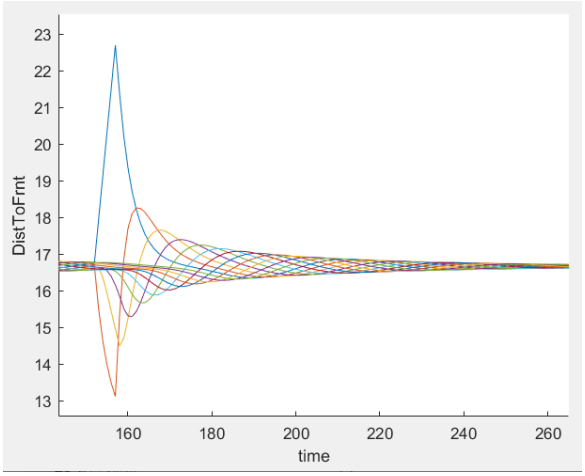


Fig. 2b. Detail.

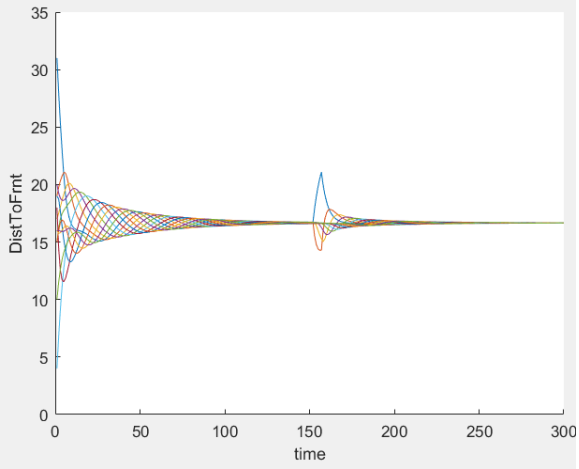


Fig. 3a. Perturbation, k = 2, 3, 2.

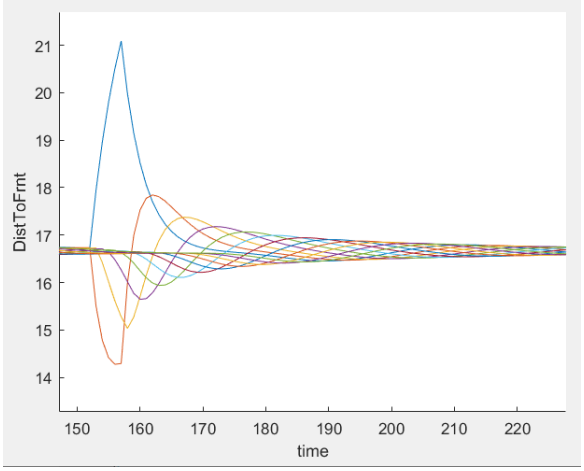


Fig. 3b. Detail.

An increase in value of k results in slowing down as is evident by comparing the above figures.

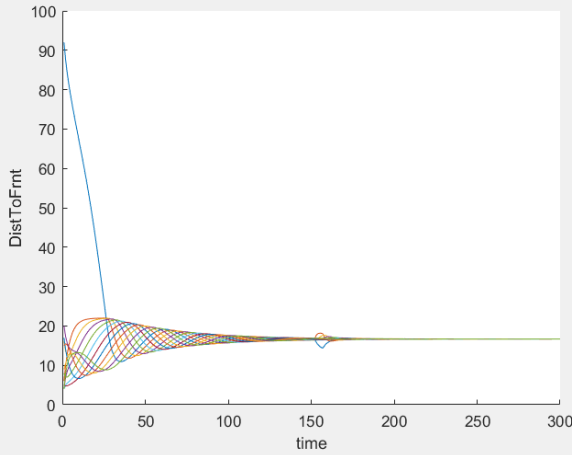


Fig. 4a. Perturbation, k = 2, 1.6, 2.

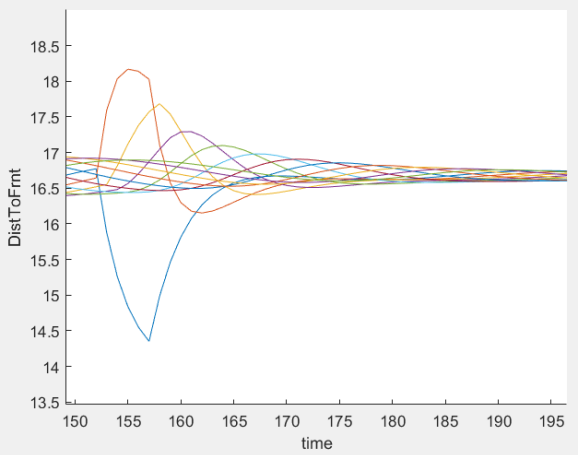


Fig. 4b. Detail.

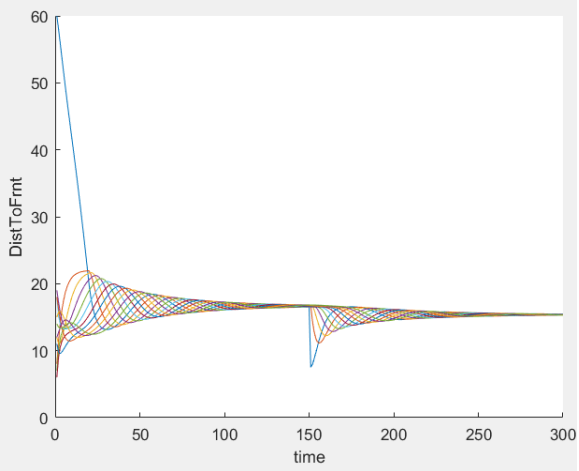


Fig. 5a. Perturbation, abrupt insertion of a vehicle.

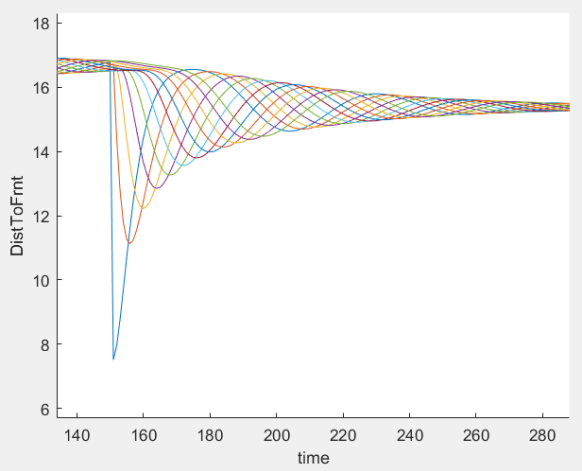


Fig. 5b. Detail.

Given the stability analysis and the Equilibrium algorithm's tolerance to various perturbations, the algorithm is generalized as it lends itself to solving merging and intersection crossing problems and is presented in the following section. Furthermore, although the algorithm is shown to be stable to perturbations, abrupt insertion of a vehicle in a string is an undesirable merging behavior and potentially unsafe. In the following sections we present a strategic approach for a smooth, seamless merging of vehicles from one lane into another and crossing intersections safely.

IV. Generalized Algorithm

The generalized Equilibrium algorithm as executed by N_i, for all i:

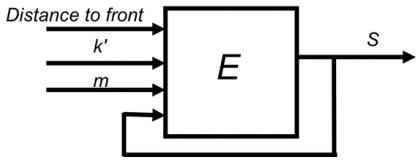
while (true)
{

$d_{i-1,i}(t\text{-}1) = Loc_{i-1}(t\text{-}1) - Loc_i(t\text{-}1)$ // *expected value* $\geq d_{safe}$

$d_i(t) = min((d_{i-1,i}(t\text{-}1) - \mathbf{m}.d_{safe})/k, S)\mathbf{k'}$ // *expected value* ≥ 0

$Loc_i(t) = Loc_{i-1}(t\text{-}1) + d_i(t)$, *for all i* // *expected value* $\geq d_{safe}$

}



Two new parameters are introduced in the Equilibrium algorithm, namely, m and k', which are highlighted in **bold**, where $m \geq 1$ and $k' = k/(k + \delta)$, where $\delta \geq 0$. $m > 1$ increases intervehicle spacing while $k' \leq 1$, i.e., $\delta > 0$, reduces maximum speed of the lane resulting in a gradual phase difference between the two lanes. With m and k' chosen properly, the vehicles in the two lanes will be sufficiently spread apart and out of phase with each other at the merge point, thus, resulting in smooth merging, potentially without a need for a last-minute speed or phase adjustment, assuming the vehicles have same length. Generalization to account for various vehicle lengths is left to future work. Hereafter, by Equilibrium algorithm, or simply algorithm, we mean generalized Equilibrium algorithm.

V. A Strategic Approach

For vehicles to autonomously tackle merging and intersection crossing problems and navigate the crowded sky of the future on their own, they need two things: awareness of own location and real-time knowledge of the traffic flow. A map and GPS type services can provide location awareness, assuming near 100% availability and sufficient accuracy of GPS services. Although vision, radar, and other sensors on vehicles are needed to gather accurate local knowledge, they are not going to be the primary sources to provide the needed global view of the system. Real-time knowledge of the traffic requires a more global view that can either be acquired via direct V2V communication between all vehicles (without considering V2V communication range for now), or provided to the vehicles externally, e.g., by a global and central entity that is somehow aware of all vehicles' status. For a distributed autonomous navigation of the future national airspace with thousands of drones flying, the amount of communication required for V2V communication will increase exponentially as the number of vehicles grows linearly, making a full awareness of the status of all vehicles impractical, if not infeasible. (This is besides the fact that V2V range is limited to no more than 300 meters and aircraft cover more than 300 meters in a short time.) Additionally, the vehicles need to have processing capability for such high volume of data to reach consensus among themselves to negotiate a safe and feasible sequence prior to merging and intersection crossing. A

challenging task in its own turn given the dynamic nature of the network of corridors and the vehicles' locations in each corridor at any given time. (Reaching a distributed consensus is a complex, open-ended problem for a static network of entities and a topic of ongoing research.) Such tasks are in the purview of an external entity.

A global central entity to gather and manage the overall traffic flow also requires a capability to communicate with all vehicles which becomes impractical as the size of the network increases. However, implemented as a set of distributed collaborative V2Is that provide local knowledge and communication means to the vehicles, on the one hand, and with other V2Is using an Infrastructure-To-Infrastructure (I2I) backbone, as shown in Fig. 6, on the other hand, the distributed global entity can alleviate the exorbitantly high-volume communication need.

A V2I device serves as a building block of a cooperative distributed system is assumed to be capable of (1) gathering and maintaining local information on the status of the vehicles as they enter and exit its region (using various sensors, e.g., vision, radar, V2V), (2) communicating with the vehicles in its region, and (3) communicating with other V2Is. Combining local knowledge and a summary of neighboring V2Is' status information, a V2I determines proper values for the algorithm parameters and broadcasts timely commands, i.e., m and k', to the vehicles in its own region. It also shares a summary of its status to the neighboring V2Is via the I2I backbone to provide a smooth, efficient, and high throughout traffic flow in the entire system. By broadcasting of m and k' to the vehicles as they enter/exit its region, V2I informs the vehicles when and how much they need to adjust their intervehicle spacing and phase differences in preparation for a seamless merge.

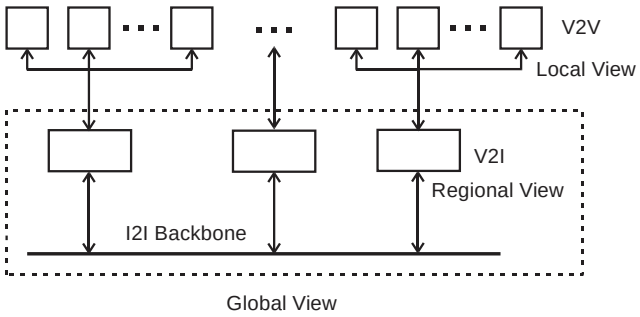


Fig. 6. A distributed system of V2I.

In addition to executing the generalized Equilibrium algorithm, a V2I employs a three-prong strategy to manage a smooth traffic flow in its own region.

A Three-Prong Strategy - How The Algorithm Lends Itself To Solving Merging And Intersection Crossing Problems

This approach assumes the V2I execute the Equilibrium algorithm and is capable of broadcasting messages (m and k') to the vehicles while they are within its specified range. The presented approach to solving merging and intersection crossing problems consists of three phases: *Prepare*, *Adjust*, *Commit*.

Prepare, Phase I – In preparation to merge or cross an intersection, a new value of m is broadcast to all vehicles instructing them to begin increasing their intervehicle spacing. The new value of m is based on the spacing needed for a safe merge or intersection crossing, where $m = f(l, n, H, g)$ and g is the status of neighboring V2Is. The V2I transmits its new value of g to neighboring V2Is. Specific function defining g is left to future work.

Adjust, Phase II – To ensure the vehicles in different lanes are out of phase with each other prior to merging/intersection crossing, a new value of k' is broadcast to the vehicles in the lower priority lanes. The new value of k' is based on the spacing needed for a safe merge or intersection crossing, where $k' = f(l, n, H, g)$.

Commit, Phase III – The vehicles are instructed to merge or cross the intersection. If the lanes are in parallel proximity, $\theta = 0$, with sufficient spacing between them, merging can be done by multiple vehicles at one time. This feature requires broadcasting additional parameter, $q \geq 1$, signaling a platoon of vehicles to seamlessly merge with another, assuming by merging two platoons we do not mean appending one behind the other.

With algorithm parameters, m and k', chosen properly, the vehicles in two lanes will be sufficiently spread apart and out of phase with each other at the merge/intersection point so that executing the *Commit* step does not induce undue perturbation in the string. Although not explicitly stated in the algorithm, upon completion of merge/intersection crossing and as the vehicles enter another zone, they are instructed to resume operation with their default values for the algorithm parameters, i.e., $m = 1$ and $k' = 1$. Also, to guarantee adequate phase shift, anticipate unforeseen scenarios, and as a safety measure, there is always a need for a last-minute tactical adjustment. Direct V2V communication (assume sufficient range) is a practical means for the vehicles to communicate with

their neighboring vehicles, acquire more accurate knowledge of each other's status, and make last-minute adjustment to their speed to be sufficiently out of phase with each other.

The parameter *m*, *k'*, and *g* are short and concise messages broadcast by V2I costing far less communication bandwidth than exchanging V2V messages between all vehicles. In the simulations, for instance, $m = 5l$ because minimum intervehicle spacing of $2l$ was sufficient for merging purposes. Similarly, $k' = 0.9456$ to result in sufficient phase differences between the two lanes.

In the following section we show that in the absence of unforeseen scenarios, with this three-step approach, vehicles in a V2I region merge and cross intersection interweaving in time and space. One of the challenges encountered in analyzing V2I behavior was modeling challenges associated with merging and intersection crossing as two separate cases, which was resolved as explained below.

Intersection Crossing Is Special Case Of Merging

Upon merging, the vehicles share a lane for a time interval until they depart to different lanes from the shared space. When the shared space and the amount of time that space is occupied are relatively limited, merging becomes synonymous to occupying a common space at an intersection which is typically assumed to be momentary. Thus, we view the intersection crossing problem as a special case of the merging problem where the vehicles alternatively share a limited space for a short time. Therefore, hereafter, we focus on merging, in the general case, where a vehicle, or the entire lane (string), interweaves into another lane in time and space.

VI. Simulation

Earlier in the Equilibrium algorithm description *H* was defined as the length of the corridor. In a distributed system of V2Is, depending on the length the corridor and capabilities of the V2Is, the corridor may be divided into multiple regions managed by as many V2Is with each V2I region, in turn, consisting of multiple zones (*Z*). For studying behavior of a V2I behavior executing the algorithm, *H* indicates the length of the region managed by a V2I, *Z* indicates the length of a merge zone, within *H*, i.e., $Z_i \leq H$. however, for the simulations, we only consider one zone *Z*, where *Z* is measured from where the increased intervehicle spacing factor, *m*, and the phase shift factor, *k'*, go into effect until the merge point. The prepare-to-merge buffer area, *P*, is defined as part of *Z* from when the vehicles engage in the last-minute preparation, by adjusting their phase shift and tweaking speed, until the merge point using V2V. In other words, *P* reflects V2V communication range.

For simulation and analysis purposes, we consider one V2I, as a base-case, managing traffic for two lanes, *L1* and *L2*, merging into one with *L2* having higher priority than *L1*. As mentioned earlier, intersection crossing is viewed as a special case of merging. Simulation of more than two lanes are left to future work.

Simulation parameters in Table 2 are kept constant, when applied. *m*, *k'*, and cooperativeness of the two lanes are the changing parameters while *n* and *Z* are kept constant. Simulation under dynamically adjusting *Z* as *n* fluctuates are left to future work. Initial speed, spacing, and min/max speed of the vehicle are known. The vehicles are assumed to operate within their minimum and maximum allowed speeds without stalling since only fixed-wing vehicles are considered. The scenarios examine the effect of absence and presence of the Equilibrium algorithm while the vehicles are noncooperative and cooperative.

Table 2. Simulation parameters

Parameter	Description	Value
l	Vehicle length	3
d_{safe}	Min safe distance	l
S_{max}	Max speed	4
S_{stall}	Stall speed	2
P	Prepare-to-merge buffer, i.e., V2V range	30 l
n	Number of vehicles in Z	8
m	Intervehicle spacing factor	5 l
δ		0.115
k'	Phase shift factor	0.9456
k	Gain factor	2
B	Obstacle/Intersection Crossing length	15 l
R	Merge buffer	5 l

Initial conditions for all scenarios in the simulations are the same. The vehicles in both lanes start from a parked position (zero speed) and at the same distance from the merge point but long before reaching the merge zone *Z*. Their sole task is to move forward as fast as they can, navigating around an obstacle that is blocking *L1* by merging in *L2* until they pass the obstacle and switch back into their original lanes and move forward. $k' = 0.9456$ was empirically determined to result in sufficient phase differences between the two lanes. The constant parameters in Table 2 were chosen for visual purposes and animation during simulation. Determination of proper values for m and k' are left to future work. When the length of the obstacle is small, resumption of navigation in the original lane after the merge is synonymous to crossing an intersection.

The merge buffer, *R*, is modeled after ground-based highways where the buffer is meant to adjust speed and phase differences and use visual cues as the final adjustment before merging into the ongoing traffic. This buffer starts from the designated merge point to a stop point beyond which vehicles have to stall. As the vehicles enter a V2I zone, they are instructed to adjust their intervehicle spacing and maximum speed. When preparing to merge (within $P+R$), the vehicles rely on their V2V communication for last-minute final speed and phase adjustment (detect and avoid scheme). If Z and $P+R$ are chosen properly, there may not be a need for V2V communication and a possible last-minute conflict resolution scheme. These parameters are depicted in Fig. 7.

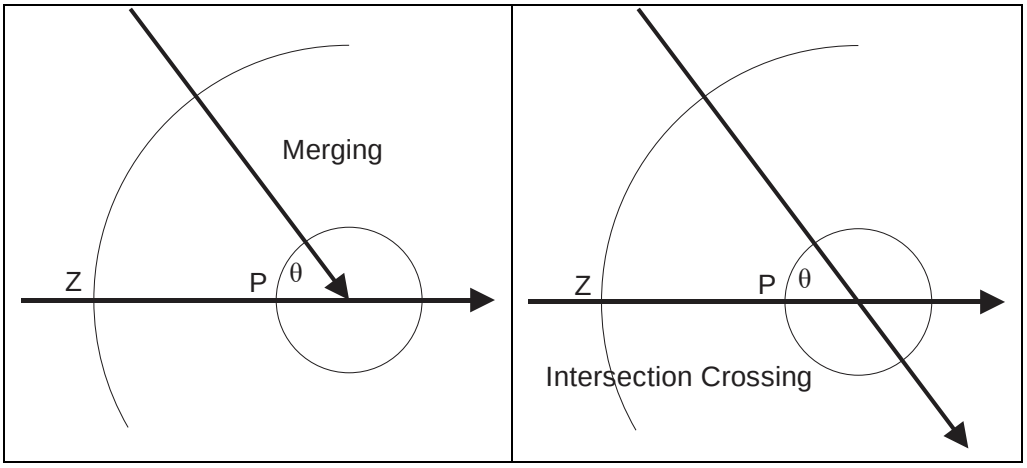


Fig. 7. Merging and intersection crossing models, $\theta \geq 0$.

To show the effectiveness of the Equilibrium algorithm, we first simulate a scenario where the algorithm is disabled, thus, the vehicles effectively operate in ad hoc mode. In the following scenarios behavior of the system of non-cooperative or cooperative vehicles are examined. The following ad hoc scenario, in the absence of V2V communication between the vehicles, the system is modeled after a two-lane ground-based, i.e., $P = 0$ and $R > 0$, roads and *L1* and *L2* are assumed to be parallel to each other, $\theta = 0$. Thus, the vehicles have local knowledge and awareness of themselves and their immediate neighbors in both lanes. Note, in ad hoc mode, *m* and *k'* do not play a role.

	Algorithm	*Z*	*P*	*R*	*m*	*k'*	Cooperative
L1	Ad hoc	-	0	15	1	1.0	No
L2	Ad hoc	-	0	15	1	1.0	No

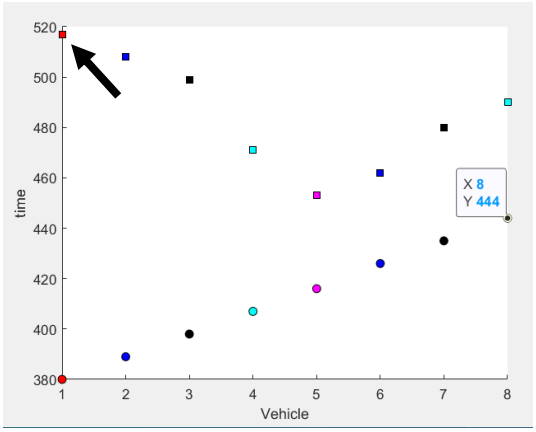


Fig. 8a. Timeline comparison of ad hoc merging of two noncooperative lanes.

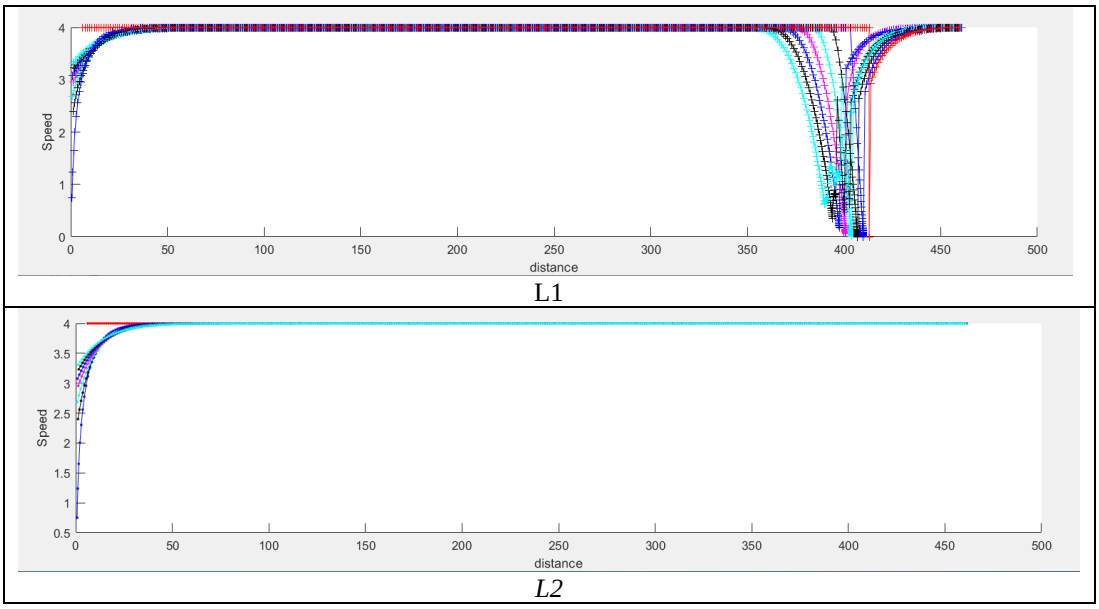


Fig. 8b. Speed comparison of ad hoc merging of two noncooperative lanes.

Given that there is a limited number of vehicles per lane, even with ad hoc model, eventually all vehicles move beyond the merge point and around the obstacle. However, in this scenario, because the vehicles in *L2* have higher priority and are noncooperative, the first vehicle in *L1* cannot pass nor merge and so stalls at the end of the merge buffer *R*, and thus, becomes the last one (taking the longest time) to change lanes as pointed to by the arrow in Fig. 8a. Due to multiple vehicles stalling, the original sequence of the vehicles is not maintained, a prevalent phenomenon on ground-based highways.

The following scenario is similar to the previous one except that the vehicles in both lanes are cooperative, however, a vehicle allows the one in the other lane to pass and merge only if it is, even slightly, ahead of it.

	Algorithm	Z	P	R	m	k'	Cooperative
L1	Ad hoc	-	0	15	1	1.0	Yes
L2	Ad hoc	-	0	15	1	1.0	Yes

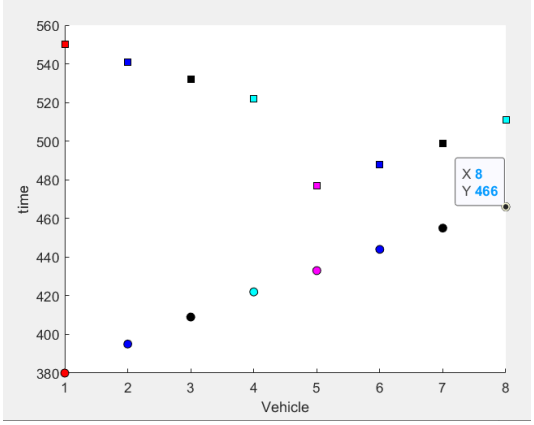


Fig. 9a. Timeline comparison of ad hoc merging of two cooperative lanes.

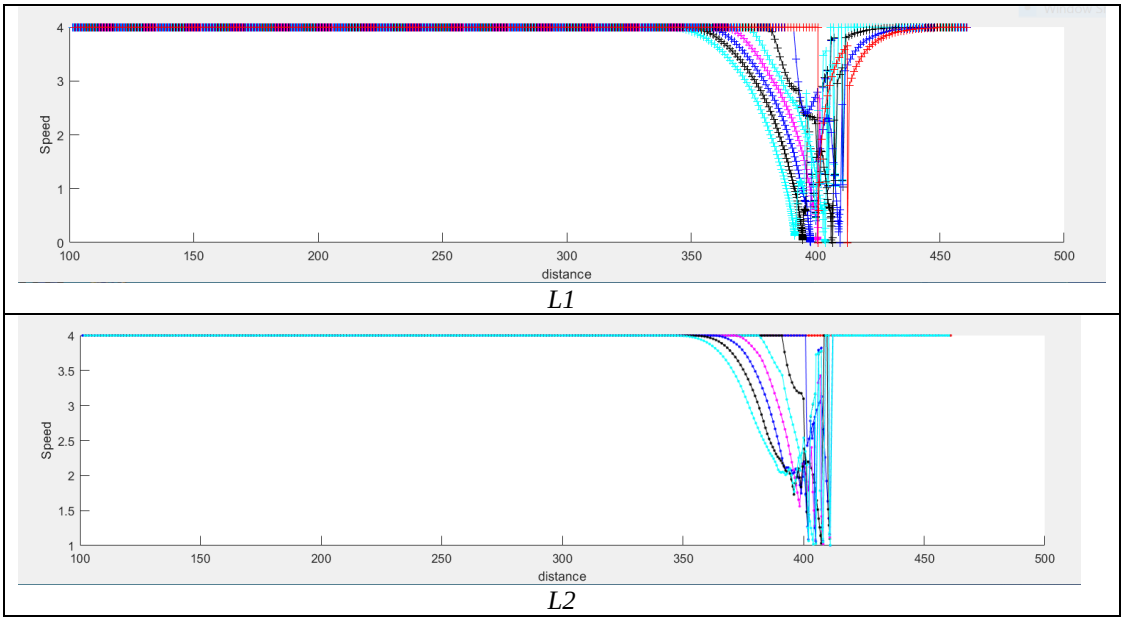


Fig. 9b. Speed comparison of ad hoc merging of two cooperative lanes.

In this scenario, although slight improvement in timeline of the activities is seen in the lower priority lane, *L1*, cooperation has increased activities in the higher priority lane, *L2*. Stalling however remained prevalent. In Fig. 9a, the timeline of the last vehicle in *L2* increased from 444 to 466 when compared to Fig. 8a.

In the following scenario, Equilibrium algorithm is in effect, $m = 5l$, and the vehicles in both lanes are signaled to increase their intervehicle spacing as they enter Z and are given ample time and space to do so, $Z >> 0$. The lane with lower priority, *L1*, has a slightly reduced maximum speed within Z as indicated by $k' < 1$ resulting in the vehicles in *L1* being sufficiently out of phase with those in *L2* prior to reaching the merge point, thus, eliminating a need for a last-minute adjustment using V2V for instance and so, $P = 0$ and $R = 0$. However, for safety measures, in a possible implementation, P and R are expected to have nonzero values. This scenario applies to $\theta \geq 0$.

	Algorithm	*Z*	*P*	*R*	*m*	*k'*	Cooperative
L1	Extended Equilibrium	170	0	0	5*l*	0.95	Yes
L2	Extended Equilibrium	170	0	0	5*l*	1.0	Yes

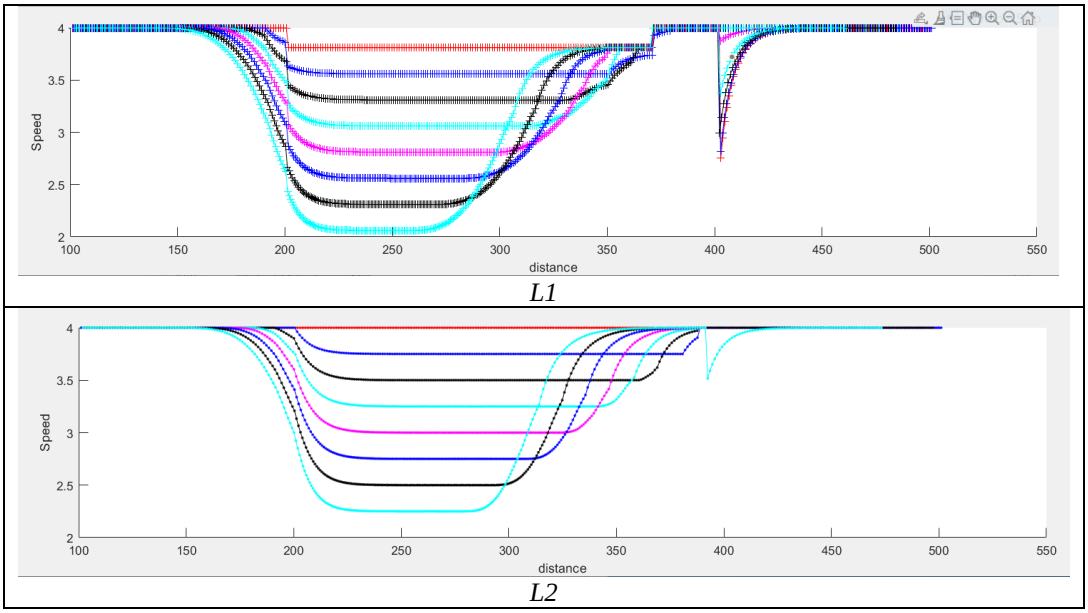


Fig. 10. Speed comparison of Extended Equilibrium merging of two cooperative lanes.

For *L1*, *k'* = 0.9456, results in an approximately 5% reduction in maximum speed compared to *L2* with *k'* = 1, which is not necessarily an optimum value. As shown in Fig. 10, by gradual adjustment and dissemination of *m* and *k'*, as the vehicles navigate through *Z*, much smoother preparation and merge is achievable without any of the vehicle stalling, i.e., $S(t) > S_{stall}$. Analysis of dynamic adjustment of *m* and *k'* is left as future work.

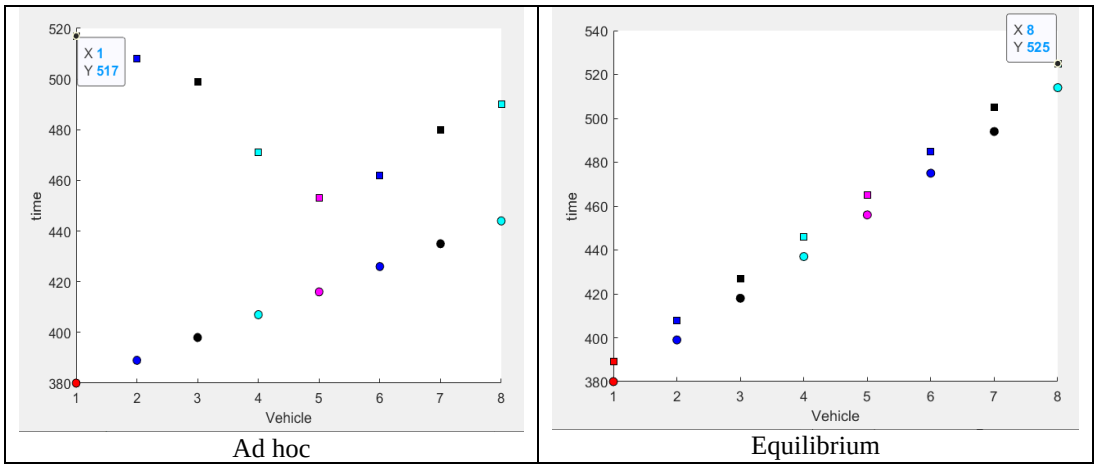


Fig. 11. Comparison of above scenarios.

Given the limited number of vehicles in the above scenarios, as their comparison in Fig. 11 indicates, in both the absence (ad hoc) and presence of the Equilibrium algorithm, all vehicles eventually merged and resumed on their original paths at roughly the same amount of time. What is different is the quality of the flight, lack of stalling, maintaining sequence of vehicles in both lanes, predictability of traffic flow, and a smooth and orderly merging of the two lanes when the vehicles cooperate and adhere to the Equilibrium algorithm.

VII. Conclusion

This paper addressed vehicle (drone) separation in time and distance in high volume corridors (en route) and lanes (on ground) as well as merging and intersection crossing of multiple lanes with a large number of autonomous fixed-wing vehicles navigating the predicted crowded skies of the future. An Equilibrium algorithm was presented and shown to be string stable in maintaining intervehicle separation. Resilience to induced perturbations lend the algorithm to solving merging and intersection crossing problems and was thus generalized. A decentralized, collaborative, distributed system of regional entities, V2Is, was presented that monitor and manage traffic flow dynamically and as the vehicles enter/exit in their corresponding regions. Each V2I adjusts the range of the merge zone, dynamically updates maximum speed within its merge zone, informs the vehicles of the adjustments to its zone and updated maximum speed of vehicles by broadcasting short commands to the vehicles. By recognizing that intersection crossing is a special case of the merging problem, this paper focused on the merging problem. Analysis of one such V2I under various scenarios executing the generalized Equilibrium algorithm and a three-prong strategy is presented. Simulation results were presented that showed the proposed approach is stable and resilient to induced perturbations and guarantees a set of fixed-wing vehicles will safely merge and cross intersections by adjusting their speed based on their distance to the aircraft in front of them while remaining in the equilibrium state. The proposed three-prong strategy is scalable to merging multiple vehicles at one time as well as merging of one platoon with another platoon.

References

[1] FAA Aerospace Forecasts Fiscal Years 2021-2041 (https://www.faa.gov/).

[2] Abbott, T.: *An Overview of a Trajectory-Based Solution for En Route and Terminal Area Self-Spacing to Include Parallel Runway Operations*, Nasa/cr-2011-217194, 2011.

[3] Abbott, T.: *An Overview of a Trajectory-Based Solution for En Route and Terminal Area Self-Spacing: Fourth Revision*, Nasa/cr-2013-218044, 2013.

[4] Balachandran, S., Muñoz, C., Consiglio, M., Feliú, M., Patel, A.: *Independent Configurable Architecture for Reliable Operation of Unmanned Systems with Distributed On-Board Services*, Proceedings of the 37th Digital Avionics Systems Conference (DASC 2018), London, England, UK, 2018.

[5] Balachandran, Manderino, C., S., Muñoz, C., Consiglio, M.: *A Decentralized Framework to Support UAS Merging and Spacing Operations in Urban Canyons*, Proceedings of 2020 International Conference on Unmanned Aircraft Systems (ICUAS 2020), 2020.

[6] Baxley, B., Swenson, H., Prevot, T., and Callantine, T.: *NASA's ATM technology demonstration-1: Integrated Concept of arrival operations*, Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st, IEEE, 2012, pp. 3B1–1.

[7] Swieringa, K. A.: *The String Stability of a Trajectory-Based Interval Management Algorithm in the Midterm Airspace*, 15th AIAA Aviation Technology, Integration, and Operations Conference. 2015.

[8] Dresner, and P. Stone: *Multiagent traffic management: An improved intersection control mechanism*, in Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems. ACM, 2005, pp. 471–477.

[9] Bruni, L., Colombo, A., and Del Vecchio, D.: *Robust multi-agent collision avoidance through scheduling*, in Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on. IEEE, 2013, pp. 3944–3950.

[10] Miculescu, D., and Karaman, S.: *Polling-systems-based autonomous vehicle coordination in traffic intersections with no traffic signals*, arXiv preprint arXiv:1607.07896, 2016.

[11] Colombo, A., and Del Vecchio, D.: *Efficient algorithms for collision avoidance at intersections*, in Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control. ACM, 2012, pp. 145–154.

[12] Zhang, Y. J., Malikopoulos, A. A. , and Cassandras, C. G.: *Optimal control and coordination of connected and automated vehicles at urban traffic intersections*, in American Control Conference (ACC), 2016. IEEE, 2016, pp. 6227–6232.

[13] Zhao, L., Malikopoulos, A., and Rios-Torres, J.: *Optimal control of connected and automated vehicles at roundabouts: An investigation in a mixed-traffic environment*, arXiv preprint arXiv:1710.11295, 2017.

[14] Lee, J., and Park, B.: *Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment*, IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 1, pp. 81–90, 2012.

[15] Malekpour, M.R.: *Achieving Equilibrium for Dense, Integrated Vehicle Navigation*, AIAA SciTech 2021 Forum, virtual, January 2021.

[16] Omae, M., Fukuda, R., Ogitsu, T., Chiang, W.: *Spacing Control of Cooperative Adaptive Cruise Control for Heavy-Duty Vehicles*, 7th IFAC Symposium on Advances in Automotive Control, The International Federation of Automatic Control, September 4-7, 2013.