Adaptive Highways on a Grid

Hajir Roozbehani and Raffaello D'Andrea

Abstract. The primary objective of this paper is to introduce the *adaptive highways* algorithm, a path planning algorithm for vehicles moving on a grid. We consider a workspace that consists of a symmetric grid and a large number of vehicles that move on the grid to accomplish a certain task. Each vehicle is assigned the task of visiting a set of randomly selected locations, which are updated over time. The dynamics of the vehicles are described by a constrained linear double-integrator model. The objective is to find, in real time, a set of trajectories that maximize the average speed of the vehicles while ensuring safety. The trajectory optimization problem is solved locally, whereas a central entity is employed for distribution of information. Safety guarantees are provided through a space reservation mechanism. Several algorithms are presented and compared in terms of performance.

1 Introduction

This paper is primarily concerned with path planning for vehicles that move on a grid to accomplish a certain task. The objective is to move the vehicles along collision-free trajectories from their starting points to their final destinations as quickly as possible. Various algorithms are explored in this context and evaluated in terms of robustness and performance.

A reservation mechanism is used to ensure collision-free trajectories. In other words, vehicles must reserve space before using it. Collision-free navigation can then be achieved if the vehicles are always contained within their reserved areas. Hence, regardless of the algorithm used for trajectory

Hajir Roozbehani
Automatic Control Laboratory, EPFL, Switzerland
e-mail: hajir.roozbehani@epfl.ch

Raffaello D'Andrea
Institute for Dynamic Systems and Control, ETHZ, Switzerland
e-mail: rdandrea@ethz.ch

optimization, the vehicles always remain safe. Three algorithms are explored for trajectory optimization: Reservation Based Algorithm (RBA), Fixed Highways Algorithm (FHA), and Adaptive Highways Algorithm (AHA).

The proposed algorithms quantify the desirability of a given candidate path in different ways. The RBA favors trajectories that do not pass through the current space reserved by other vehicles at any given time. The FHA builds on the RBA by further imposing a predefined structure that allows the vehicles to move only in certain directions in such a way that excludes the possibility of two vehicles running into head-to-head conflicts. The AHA achieves avoiding head-to-head conflicts without imposing any predefined structure; instead, it favors paths where the traffic flows in only one direction. As a result, highways emerge and periodically change direction. This paper compares the RBA, FHA, and AHA algorithms in simulated experiments and shows that the AHA demonstrates better performance over the other methods.

In addition to its promising performance in simulations with hundreds of vehicles, the AHA is further shown to be a real-time algorithm that can be implemented on systems comprising of large number of autonomous vehicles. The Kiva Mobile Fulfillment System (Kiva MFS), is one such system. In the Kiva MFS, hundreds of robots (see Fig.1) navigate around a warehouse to pick inventory items and carry them to fixed stations for packing, leading to increased productivity [2] [3] [11]. In such systems, a robot must complete its assigned task in the shortest possible time, while avoiding conflicts with all other robots. However, finding conflict-free minimum-time paths for such large number of robots in real time is a difficult problem. The AHA is shown to be able to tackle such large-scale problems in contexts similar to that of the Kiva MFS.

The underlying path planning problem has been widely studied in the literature and dates back to early research in robotics motion planning. A recent book that covers the vast field of algorithms and their applications to path planning is referenced in [4]. Path planning for autonomous vehicles in the presence of dynamic obstacles is relatively more recent. For example, a method based on rapidly-exploring random trees [5] is presented in [1]. The algorithm is based on randomly choosing intermediary waypoints that might generate feasible trajectories to the destination. The algorithm is powerful in adversarial settings where the robots have to generate trajectories on-line in unknown environments. Path planning has also been of high interest in co-operative settings. For instance, Raffard et al. [9] have developed an iterative algorithm based on decomposition of the global problem into a sequence of tractable sub-problems. A method based on decentralized receding horizon control is presented in [10]. Safety is guaranteed through maintaining a reachable loiter pattern in all intermediate planning steps. While keeping in mind that these results can potentially be extended to large-scale systems, most of the approaches focus on applications with small number of vehicles. Real-time implementation of such approaches, which rely in one way or another on expensive iterations of an optimization process, is not a trivial task. Other

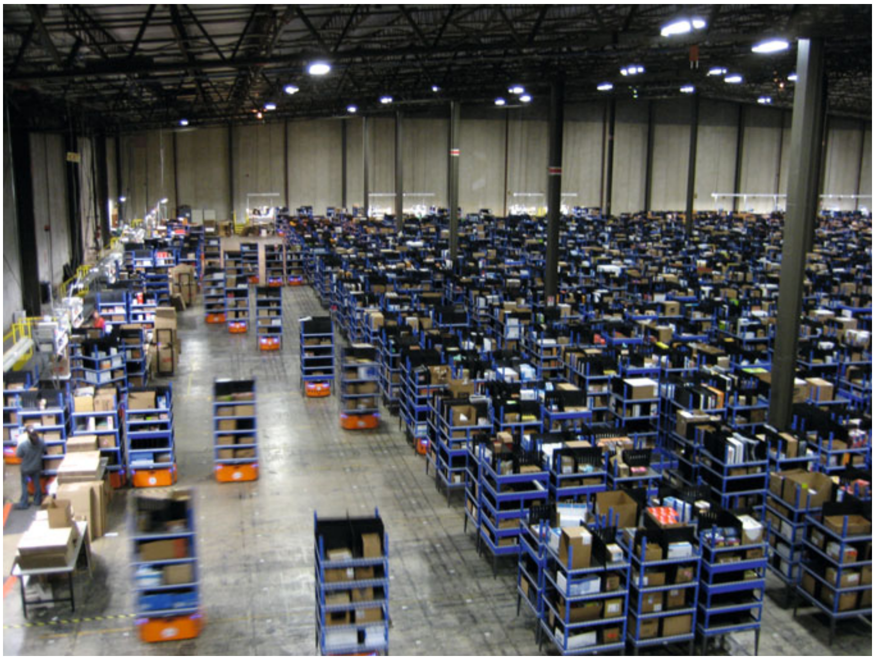


Fig. 1 A Kiva warehouse. The blue shelves hold the inventory and can be moved by the robots. The workers stay at fixed stations and the robots move around the warehouse to pick up the inventories and bring them to the stations. The underlying navigation grid can be inferred from the marks left by the robot wheels.

approaches that move toward full decentralization provide promising scalability features. These, however, come at the cost of decreased performance. For instance, a fully decentralized approach to multi-vehicle path planning is presented in [8]. Note, however, that the proposed algorithm behaves rather conservatively since it perceives any intersecting paths as a conflict without considering the time course of the vehicles on their paths.

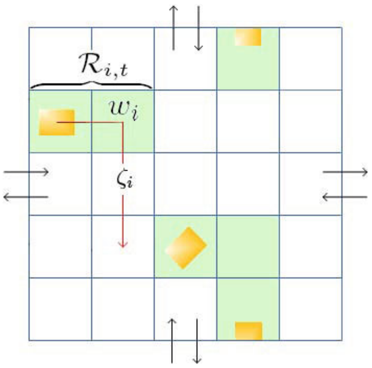
The AHA is based on predicting the potential conflicts and evaluating the expected time loss they incur. To do so, the algorithm explores an extra dimension in its search space to determine if any conflict occurs along a candidate path based on a time estimation of the available trajectories. Furthermore, the AHA algorithm distinguishes three different types of conflicts: head-to-head, head-to-side, and head-to-back. This enables it to favor more moderate conflicts (such as head-to-side) to tighter ones (such as head-to-head) on a candidate trajectory.

2 Preliminaries

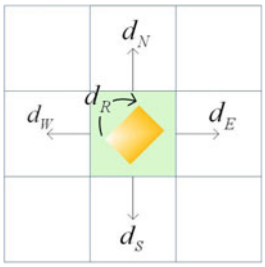
This works considers the optimization problem of finding the minimum time paths for a group of N vehicles with independent dynamics moving under both coupled and individual constraints. The vehicles are assumed to navigate on a workspace $\mathcal{W} \subset \mathbb{R}^2$ and follow target points drawn from a spatially uniform distribution. The workspace, as shown in Fig.2a, is a symmetric 2-torus. The working region is further discretized into a set of M identical partitions, called cells $\mathcal{W}_1, ..., \mathcal{W}_M$, such that $\bigcup_{j=1}^{M} \mathcal{W}_j = \mathcal{W}$ and $\mathcal{W}_i \bigcap_{i \neq j} \mathcal{W}_j = \emptyset$. Every cell $\mathcal{W}_j$ represents a unit square:

$$\mathcal{W}_j = \{(x, y) \in \mathbb{R}^2 \mid \;\; \bar{x}_j - \frac{1}{2} \leq x < \bar{x}_j + \frac{1}{2},\;\; \bar{y}_j - \frac{1}{2} \leq y < \bar{y}_j + \frac{1}{2}\}$$

with $(\bar{x}_j, \bar{y}_j) \in \{\frac{1}{2}, \frac{3}{2}, ..., \frac{\sqrt{M}}{2}\} \times \{\frac{1}{2}, \frac{3}{2}, ..., \frac{\sqrt{M}}{2}\}$ being the center of the cell. It is assumed that a vehicle can be contained within a cell. More details on the vehicles will be given later. Once a vehicle is contained in the interior of its destination cell, a new target point is assigned to it.



(a) The world model



(b) The atomic decisions

Fig. 2 *a) The world model*: The navigation map is a symmetric grid with wrap-around. If a vehicle leaves the map from one edge, it enters it from the opposite side, similar to The Game of Life or the PacMan video game from the 80's. $\mathcal{R}_{i,t}$ denotes the reserved space (see 2.4) of vehicle i at step t. ζ_i is a discrete trajectory (see 2.2). w_i denotes the current destination, which is a corner point (or breakpoint) located along the robot's trajectory. *b) The atomic decisions*: The atomic decisions either impose a rotation or guide the vehicle toward one of its neighboring cells.

2.1 Vehicle Dynamics

The position of every vehicle is denoted by $q_i = (x_i, y_i) \in \mathcal{W}$ and the heading is represented by $\theta_i \in [0, \frac{\pi}{2}]$, measured with respect to the global reference frame. We denote by $v_i \in \mathcal{V} = [-\bar{v}, \bar{v}]$ the linear velocity, and by $\omega_i \in \Omega = [-\bar{\omega}, \bar{\omega}]$ the rotational velocity of vehicle i. The admissible linear and rotational accelerations are represented by $u_i \in \mathcal{U} = [-\bar{u}, \bar{u}]$ and $\alpha_i \in \vartheta = [-\bar{\alpha}, \bar{\alpha}]$, respectively. The vehicle dynamics are modeled by a double integrator system under input and velocity constraints (see Fig. 3). It is assumed that, at any given time, the vehicle can either rotate or move in only one direction, and that transitions between horizontal ($\theta = 0$) and vertical ($\theta = \frac{\pi}{2}$) states must go through a rotational state:

$$\ddot{x}_i = u_i,\ \dot{x}_i \in \mathcal{V},\ \dot{y}_i = 0,\ \dot{\theta}_i = 0,\ \theta_i = 0,\ y_i \in \{\bar{y}_j\} \tag{1a}$$

$$\ddot{\theta}_i = \alpha_i,\ \dot{\theta}_i \in \Omega,\ \dot{x}_i = 0,\ \dot{y}_i = 0,\ (x_i, y_i) \in \{\bar{x}_j, \bar{y}_j\} \tag{1b}$$

$$\ddot{y}_i = u_i,\ \dot{y}_i \in \mathcal{V},\ \dot{x}_i = 0,\ \dot{\theta}_i = 0,\ \theta_i = \frac{\pi}{2},\ x_i \in \{\bar{x}_j\} \tag{1c}$$

These constraints are incorporated in the task specification process and fulfilled during task implementation. It is assumed throughout the paper that the input and state constraints are $\bar{u} = \bar{v} = 1$ for translational motions and $\bar{\alpha} = \bar{\omega} = 2\pi$ for rotational motions. Note that the above choice of dynamics in combination with the gridded workspace resembles the navigation scheme of the Kiva MFS shown in Fig.1.

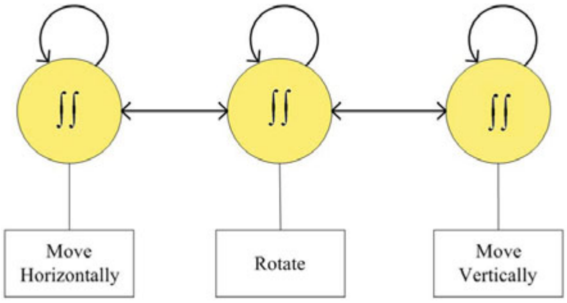


Fig. 3 The three states of Eq.1. In order to transit from moving horizontally (Eq.1a) to moving vertically (Eq.1c), the vehicle must visit an intermediary rotational state (Eq.1b).

2.2 Trajectories

The motion characteristics of the vehicle can be abstracted into a finite state model. This is useful for the path planning algorithms in Section 4. Continuous variables q_i and θ_i can be mapped into the discrete state variable $\sigma^i = \{(\mathrm{x}^i, \mathrm{y}^i), \mathrm{R}^i\}$. The pair $(\mathrm{x}^i, \mathrm{y}^i)$ represents the cell position of q_i (i.e., the center of the cell that contains q_i). For example, $(0.75, 0.5)$ is mapped to $(\frac{1}{2}, \frac{1}{2})$. Furthermore, the discrete rotational state R^i is defined by:

$$\mathrm{R}^i = \begin{cases} \mathrm{H} & 0 \le \theta_i < \frac{\pi}{4} \\ \mathrm{V} & \frac{\pi}{4} \le \theta_i \le \frac{\pi}{2} \end{cases}$$

In other words, a vehicle is perceived to be in a horizontal/vertical state H/V if its orientation is closer to the horizontal/vertical axis. The set of admissible decisions is denoted by $\mathcal{D} = \{d_N, d_S, d_E, d_W, d_R, d_\emptyset\}$. The atomic decisions presented in Fig.2b define discrete steps to move either horizontally $\{d_E, d_W\}$ or vertically $\{d_N, d_S\}$, to rotate d_R, or to wait and do nothing $d_\emptyset$. Naturally, the discrete states evolve according to the adopted decisions. This can be described with a discrete dynamical system:

$$\sigma^i_{k+1} = \mathcal{F}(\sigma^i_k, d^i_k) \tag{2}$$

For example, $\mathcal{F}(\frac{1}{2}, \frac{1}{2}, \mathrm{H}, d_E) = (\frac{3}{2}, \frac{1}{2}, \mathrm{H})$. Furthermore, note that vertical/horizontal moves are not allowed when the vehicle is facing horizontally/vertically.

A discrete trajectory ζ_i (see Fig.4), which leads vehicle i from its starting coordinate to its target, is defined as follows:

Definition 1. A discrete trajectory,

$$\zeta_i = \big\{(\sigma^i_0, t^i_0, d^i_0), (\sigma^i_1, t^i_1, d^i_1), .., (\sigma^i_n, t^i_n, d_\emptyset)\big\}$$

is a finite sequence whose elements are the vehicle's discrete position and orientation σ^i_k, a time estimate t^i_k, and an atomic decision d^i_k with $d^i_n = \emptyset$.

2.3 Central Hub

Although the trajectory optimization problem is solved locally, the vehicles can rely on a central entity for communication purposes. We assume that an omniscient central hub $\mathcal{H}$ exists that is aware of the reserved space and discrete trajectory of every vehicle. This central hub can be considered as a blackboard on which every vehicle can write about its intentions and status. Furthermore, the central hub imposes a fixed ordering on the vehicles for communicating to the hub, reserving space, and path planning. Note that

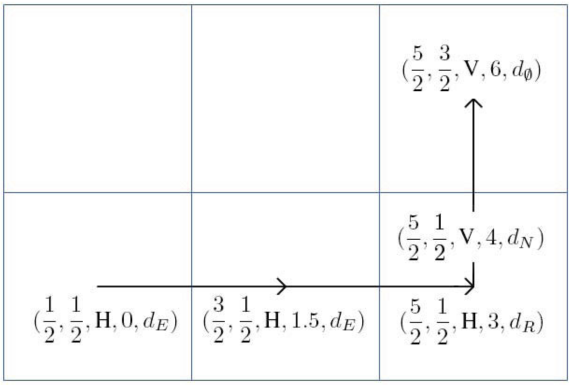

$(\frac{5}{2},\frac{3}{2},\mathrm{V},6,d_\emptyset)$
$(\frac{5}{2},\frac{1}{2},\mathrm{V},4,d_N)$
$(\frac{1}{2},\frac{1}{2},\mathrm{H},0,d_E)$
$(\frac{3}{2},\frac{1}{2},\mathrm{H},1.5,d_E)$
$(\frac{5}{2},\frac{1}{2},\mathrm{H},3,d_R)$

Fig. 4 A discrete trajectory. The vehicle starts in a horizontal state $\mathrm{R}_0^i = \mathrm{H}$ at time $t_0^i = 0s$. Given the constraints on the maximum velocities and accelerations, it follows that rotations and horizontal/vertical moves take $1s$. Note that each acceleration/deceleration takes an extra $0.5s$.

the information stored in the hub contains discrete states of the vehicles and is updated at a relatively slow rate. Hence, the communication and memory requirements of the central hub are modest. Distribution of information can be achieved in a decentralized fashion as well, by exploiting geometrical distribution of the vehicles.

2.4 Ensuring Collision-Free Paths

A reservation mechanism ensures that the vehicles move on collision-free paths. In short, the mechanism requires the vehicles to reserve space as they move. The reserved space of each vehicle (the green areas in Fig.2a) is a sequence of cells located on its trajectory that contain the vehicle itself. As long as a cell is reserved by some vehicle, no other vehicle can use it. Furthermore, it is required that each vehicle can stop within its reserved area without requiring any more reservations. Hence, the vehicles are always contained within their exclusive reserved areas, ensuring safe navigation. In addition, once a vehicle is finished using its reserved space, it releases that space so that other vehicles can use it.

To express the reservation mechanism more formally, we let the collision region $P_{i,t} \subset \mathcal{W}$ indicate the smallest union of cells that fully contains vehicle i at time t. Collision avoidance requires that $P_{j,t} \bigcap_{i \neq j} P_{i,t} = \emptyset$ for all t. At any time $t \in [t_k^i, t_{k+1}^i]$, the collision region is a subset of the cells corresponding to $\{\sigma_k^i, \sigma_{k+1}^i\}$. If the vehicles could instantly transfer into a stationary state in their $P_{i,t}$ zones, safety would be guaranteed by making the $P_{i,t}$ zone always unreachable to $j \neq i$. A conflict zone comprised of all cells on the discrete trajectory that lie on vehicle i's path until it can fully stop is considered, in order to ensure safety under dynamical constraints,. This gives a lower bound on the number of cells that each vehicle must reserve. In the example shown in Fig.4, the conflict zone at $t = 0s$ is the cell centered at $(\frac{1}{2}, \frac{1}{2})$ and at $t = 0.5s$ is the set of cells centered at $(\frac{1}{2}, \frac{1}{2})$ and $(\frac{1}{2}, \frac{3}{2})$. As mentioned earlier, it is required that every vehicle i be completely contained inside its reserved area $\mathcal{R}_{i,t}$ at all times. Therefore, collision-free navigation is guaranteed if $\mathcal{R}_{j,t} \bigcap_{i \neq j} \mathcal{R}_{i,t} = \emptyset$.

From a single vehicle point of view, the optimal reservation policy for vehicle i is to request a space that contains its entire trajectory. However, this prevents other vehicles from using the available cells. Hence, for performance reasons, the request from the central hub contains just enough space so that the vehicle can be safe while moving at maximum speed. Every vehicle sets a candidate goal point $\tilde{g}_i \in \mathbb{R}^2$:

$$\tilde{g}_i = q_i + ({}^{\bar{v}^2}\!/_{2\bar{u}} + 2h\bar{v} + \frac{1}{2})\text{sgn}(w_i - q_i).$$

where h is a discrete sampling time and w_i denotes the current destination. Note that since diagonal moves are not allowed, the vector $\text{sgn}(w_i - q_i)$ always has one zero element. Next, the vehicle evaluates its next target point:

$$g_i = \min_{r \in \{\tilde{g}_i, w_i\}} \{\|q_i - r\|\}.$$

When the next target point is not included in the vehicle's reserved area $g_i \notin \mathcal{R}_{i,t}$, a request is sent to the central hub. The maximum number of cells to be requested is then:

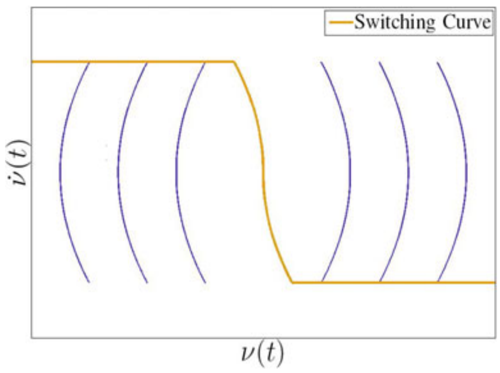
$$\text{ceil}({}^{\|q_i - g_i\|}\!/_{l})$$

Given that $\bar{v} = \bar{u} = 1$, the reserved area $\mathcal{R}_{i,t}$, with $t \in [t^i_k, t^i_{k+1}]$, is a subset of the cells corresponding to $\{\sigma^i_k, \sigma^i_{k+1}, \sigma^i_{k+2}\}$. Also, note that during rotations we have: $w_i = q_i$ and the vehicle is safe by remaining in its own cell. After receiving a response packet from the central hub, the reserved space $\mathcal{R}_{i,t}$ is updated and the next target point is modified such that it can be contained within the reserved area. Therefore, $g_i = \min_{r \in \mathcal{R}_{i,t}} \{\|w_i - r\|\}$. At this time, the vehicle also releases the unneeded portion of its reserved space.

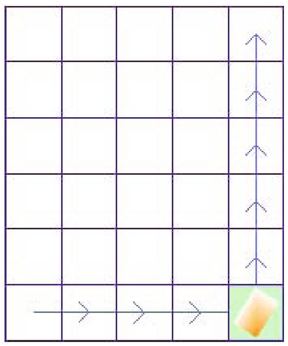
The above scheme ensures that the vehicles remain within their reserved spaces at all times.

2.5 Problem Statement

Problem 1: Consider a group of N vehicles that follow the dynamics given in (2), can communicate with a central entity $\mathcal{H}$, and comply with the rules of the reservation mechanism. Given a fixed ordering defined by the central hub, find for each $i \in N$, a discrete trajectory ζ^*_i that minimizes the required time to reach its destination.



(a) Phase portrait of the minimum-time path



(b) The generated trajectory

Fig. 5 The continuous, minimum-time trajectory: *a) Phase portrait of the minimum-time path:* The optimal controller leads the vehicle toward its current destination with at most two switches, which occur on the switching curve (the orange line). *b) The generated trajectory:* A vehicle, its minimum-time trajectory, and its reserved zone are shown in an obstacle free environment.

3 Control Architecture

A two-layer control architecture is proposed. At the higher layer, a discrete trajectory is extracted from a discrete search algorithm implemented on a local planner. Then, a low-level controller provides a continuous execution in order to implement and preserve the properties of the discrete trajectory. Note that the planners act centrally in the sense of information retrieval, but solve the path planning problem locally based on the information they have about other vehicles.

Once the discrete trajectory is decided by the planner, the vehicle looks for break points in the trajectory, where it must come to a full stop. Hence, every vehicle heads to a current destination $w_i \in \mathcal{W}$ with the intent of reaching a final destination denoted by $f_i \in \mathcal{W}$. The vehicles adopt a bang-bang control [6] policy in order to traverse between current destinations as well as to rotate.

The minimum time trajectories for a single vehicle scenario are shown in Fig. 5. As can be seen from the figure, the optimal controller has no more than two switching points. Note that each degree of freedom is controlled independently and the above figure corresponds to a general coordinate ν_i. Each vehicle traverses on the minimum time trajectories with the maximum acceleration until it reaches its switching curve. At this point, the vehicle starts coasting at maximum possible speed. Once the turning point is reached, the vehicle moves toward its current destination with maximum deceleration. Interested readers can refer to [7] for a thorough analysis of the adopted control policy. Note that in the presence of multiple vehicles the discrete trajectories extracted from the planner might be suboptimal. Nevertheless, the optimal policy at the controller level remains the same.

4 Path Planning Algorithms

This section is concerned with presenting three path planning algorithms in the scope of *Problem 1.* In short, the algorithms estimate the required time for a transition in discrete states, and further compute a feasible combination of transitions that minimizes the accumulated estimated time to reach a final state. The estimated time for transitions varies from one algorithm to another.

In the RBA (Reservation Based Algorithm), in addition to the traveled distance, the current reserved areas of the vehicles are taken into account. For each vehicle, the algorithm penalizes a candidate path for passing through a cell reserved by other vehicles based on the current distance between the vehicle and the reserved cell. In the FHA (Fixed Highways Algorithm), in addition to accounting for reservations, a highway structure dividing the workspace into odd and even lanes is imposed. The vehicles are allowed to move in only one direction on each lane. Hence, the vehicles never encounter one another in a head-to-head situation. In other words, the algorithm prunes any subset of the search space that may potentially lead to head-to-head conflicts. The drawback is that a large portion of the search space, which could otherwise be exploited by the vehicles, is omitted in a conservative way. The AHA (Adaptive Highways Algorithm), however, is based on estimating whether a conflict occurs on a candidate path or not. This is achieved by comparing the time estimations and atomic decisions on the candidate path with those available in the trajectories of other vehicles. By taking the time and decisions into account, the AHA is able to omit only unpromising candidate paths rather than a large subset of the search space. In the rest of this section, we provide details on how the algorithms work.

4.1 Description of Cost-to-Go

The time required for transition from a state σ_k^i to a neighbor state σ_{k+1}^i is denoted by $\mathcal{L}(\sigma_k^i, d_k^i)$. A lower bound on $\mathcal{L}(.,.)$ is given by:

$$\mathcal{L}(\sigma_k^i, d_k^i) = \tau_{min}(\mathcal{F}(\sigma_k^i, d_k^i), \sigma_k^i)$$

with $\tau_{min}(\mathcal{F}(\sigma_k^i, d_k^i), \sigma_k^i) = 1/\bar{v}$ if the vehicle is to move either horizontally or vertically, i.e., $\mathrm{R}_k^i = \mathrm{R}_{k+1}^i$. If a rotation must be performed, a minimum of $\tau_{rot}(\mathcal{F}(\sigma_k^i, d_k^i), \sigma_k^i) = 2\sqrt{\pi/2\bar{\alpha}}$ must be considered in the cost. Therefore, if the transition is compatible with the safety requirements posed by the reservation mechanism, the transition time is restricted only by the kinematic and dynamic constraints. Furthermore, the heuristic cost-to-go to the terminal state σ_f^i is simply:

$$\lambda(\sigma_k^i, \sigma_f^i) = \frac{1}{\bar{v}} d_{man}(\sigma_k^i, \sigma_f^i) + \tau_{rot}(\sigma_f^i, \sigma_k^i)$$

where $d_{man}(\sigma_k^i, \sigma_f^i)$ is the manhattan distance to the destination cell and $\tau_{rot}(\sigma_f^i, \sigma_k^i)$ is a lower bound on the potential rotations that might take place in the future. Here, we account only for one such rotation. Moreover, the stage cost is defined as follows:

$$\phi(\sigma_0^i, \sigma_k^i) = \sum_{j=0}^{k} \mathcal{L}(\sigma_j^i, d_j^i) + \lambda(\sigma_k^i, \sigma_f^i)$$

With the above ingredients, a standard A* algorithm can be applied to find the shortest path, i.e., to minimize $\phi(\sigma_0^i, \sigma_f^i)$.

4.2 Reservation Based Algorithm

The minimum-time optimization problems considered for each vehicle in *Problem 1* are coupled through the reservations. This has several implications on the design of the algorithms. In fact, algorithms that react only myopically to avoid collisions with other vehicles, fail to demonstrate an adequate level of performance in multi-vehicle settings. This can be understood by noting that, in the presence of multiple vehicles, the system may get stuck in a local minima or in an oscillatory trajectory. Fig. 6 shows a scenario where the vehicles myopically account for the position of each other without taking the reserved areas or future decisions of each other into account. In this case, the transition cost penalizes trajectories that pass through the current cells of other vehicles:

$$\mathcal{L}(\sigma_k^i, d_k^i) = \tau_{min}(\mathcal{F}(\sigma_k^i, d_k^i), \sigma_k^i) + \delta(\sigma_k^i, d_k^i, \sigma_0^{-i}) \tag{3}$$

where σ_0^{-i} denotes the initial discrete states of other vehicles $j \neq i$ at the start of the planning, which is always assumed to be at $t = 0s$ for convenience. In this algorithm, $\delta(.,.,.)$ is defined as follows:

$$\begin{cases} \delta(\sigma_k^i, d_k^i, \sigma_0^j) = \hat{\delta} & \text{if discrete position of } \sigma_k^i = \text{discrete position of } \sigma_0^j \\ \delta(\sigma_k^i, d_k^i, \sigma_0^j) = 0 & \text{otherwise} \end{cases}$$

where $\hat{\delta}$ is a non-zero constant. In other words, if the node σ_k^i expanded at the k-th layer of the search algorithm is occupied by another robot, the transition cost increases by a constant. It can be shown that there exists a critical cost δ_c such that for $\hat{\delta} < \delta_c$ the vehicles can get stuck in a local minima and for $\hat{\delta} \geq \delta_c$ they oscillate between two cells. If the vehicles take the reserved spaces

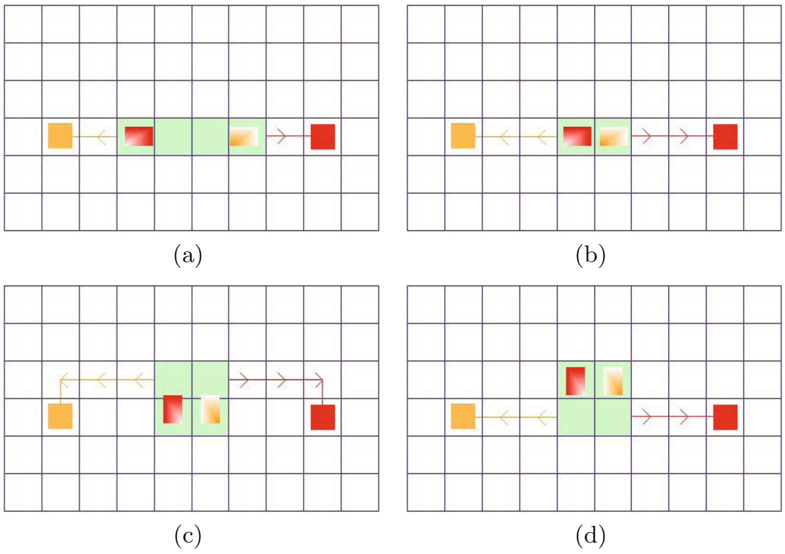


Fig. 6 A two vehicle scenario with myopic planning. The big squares are the destination cells, the rectangles are the vehicles, the arrows are the trajectories, and the green cells are the reservations. The red vehicle (on the left) is heading to the red square (on the right). Note that the reservations cover the arrows. *a)* The vehicles move toward their destinations. *b)* The vehicles encounter a head-to-head conflict and cannot further reserve the space needed to move on their trajectories. They start re-planning. *c)* First, the red vehicle plans a trajectory around the orange one. The trajectory goes up and continues to the right as shown in the figure. The orange vehicle starts planning immediately after the red vehicle and avoids its current position. Consequently, both vehicles go up. *d)* They meet again and the scenario repeats. Therefore, the vehicles demonstrate vertical oscillations. Re-planning can not resolve the oscillation deadlock.

of one another into account, the oscillation deadlocks are resolved as shown in Fig.7a. The transition cost is modified to:

$$\mathcal{L}(\sigma_k^i, d_k^i) = \tau_{min}(\mathcal{F}(\sigma_k^i, d_k^i), \sigma_k^i) + \delta(\sigma_k^i, d_k^i, \mathcal{R}_{-i,0}) \tag{4}$$

where

$$\begin{cases} \delta(\sigma_k^i, d_k^i, \mathcal{R}_{j,0}) = \hat{\delta} & \text{if discrete position of } \sigma_k^i \in \mathcal{R}_{j,0} \\ \delta(\sigma_k^i, d_k^i, \mathcal{R}_{j,0}) = 0 & \text{otherwise} \end{cases}$$

In addition, the utilized information $\mathcal{R}_{-i,0}$ is valid only within a short time horizon close to the start of planning. The reserved areas that are located at a relatively far distance from σ_0^i are likely to have been released by the time the vehicle wants to use them. For this reason, an estimation horizon $\hat{e}$ can be introduced in the planner such that:

$$\delta(.,.,.) = \begin{cases} \delta(.,.,.) & \text{dist}(\sigma_0^i, \sigma_k^i) \leq \hat{e} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

However, the performance may still be substantially influenced by conflicts. This can be better understood by realizing that, in this scheme, up to four equally optimal trajectories might exist. Hence, if a vehicle can predict that a conflict might occur on one of its shortest paths, it can choose an alternative, conflict-free trajectory. Moreover, the second order nature of the vehicles' dynamics penalizes switches on the control input. Hence, predicting the conflicts, especially in scenarios which involve a head-to-head encounter similar to Fig. 6, enables the vehicles to traverse with a smaller number of switchings on their control trajectory.

4.3 Fixed Highways Algorithm

A highway structure can be imposed on the RBA explained above. This leads to a navigation scheme prohibiting head-to-head encounters such as the scenario in Fig.7. In the FHA, the lanes are numbered with integers and the vehicles are allowed to move in only one direction on each lane based on the lane number being odd or even.

Although the highway structure resolves the head-to-head conflicts in a subtle way, the vehicles travel a longer distance on average. Furthermore, deadlocks similar to Fig. 8a may still happen. In order to circumvent the deadlock, the vehicles exploit the modified cost defined in either Eq. 3 or Eq. 4 combined with the estimation horizon defined in Eq. 5. The transition cost $\hat{\delta}$ must be high enough to force the vehicles to deviate from their paths in a head-to-side conflict. However, if there is no deadlock, the deviation

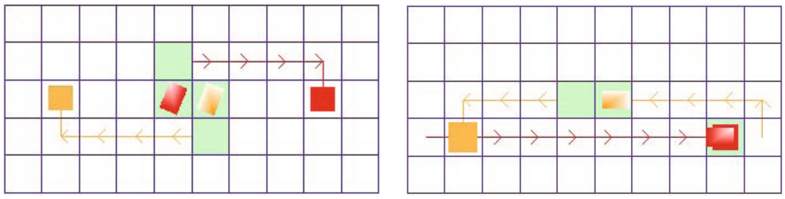

(a) Reservation Based Algorithm (b) Adaptive Highways Algorithm

Fig. 7 Comparing the RBA with the AHA in a two vehicle scenario. *a) Reservation Based Algorithm:* The vehicles first meet in a head-to-head situation and start re-planing similar to Fig. 6b. The vehicles proceed to reserving space immediately after planning. Hence, the symmetry in the conflict is broken, which leads to the resolution of the deadlock. However, the conflict has an adverse effect on the performance of both vehicles. *b) Adaptive Highways Algorithm:* The red vehicle plans first and the orange vehicle plans after it. Once the red vehicle plans a straight trajectory to its destination, the orange vehicle takes a detour to avoid any head-to-head conflicts.

diminishes the performance of the algorithm in head-to-side encounters (see Fig.9) where deviation is likely to be suboptimal.

4.4 Adaptive Highways Algorithm

In the AHA algorithm, each vehicle attempts to predict the future conflicts with other vehicles along a candidate path based on the list of trajectories (cell positions, decisions, and time estimates) stored in the central hub. The inclusion of time estimates in the algorithm requires augmenting a time element to the states of the search space, which further induces a discrete search on a hyper graph with a time dimension. To augment the states by time, a discrete planning step can be defined to incrementally increase the state of the vehicle as a function of time during the planning. Note that the dynamic constraints can be used to significantly reduce the dimension of the search problem and make the algorithm faster.

The actual time required for any transition through the states of a trajectory depend on whether or not the vehicle can reserve the corresponding cells. Moreover, the availability of the space $\mathcal{R}_{i,k}$ requested at the k-th layer of a trajectory ζ_i depends on ζ_j for $j \neq i$, more conveniently denoted by ζ_{-i}. Hence, by comparing the pairs $\{\sigma_k^i, t_k^i, d_k^i\}$ on a candidate path with those in ζ_{-i} during planning, we can more accurately determine if any conflict occurs along the candidate path. The transition cost can then be defined as follows:

$$\mathcal{L}(\sigma_k^i, d_k^i) = \tau_{min}(\mathcal{F}(\sigma_k^i, d_k^i), \sigma_k^i) + \delta(\sigma_k^i, d_k^i, \zeta_{-i}) \tag{6}$$

Note that the direction of the expected encounter can be inferred from the decision elements of the trajectories. Since the time loss due to a conflict depends on the direction of the conflict, it is likely that one has to distinguish the constraints from each other, depending on the direction of the expected encounters. For instance, the optimal cost for head-to-head encounters is likely to be higher than that of head-to-side encounters since it must be high enough to encourage deviations as can be seen in Fig.7b.

4.5 Congestion Avoidance

The cost can be further adapted to account for static vehicles. The main idea is to scan the level of traffic at each cell and associate an integral gain with the cells that contain static vehicles. A first order system is considered for the congestion equation:

$$\dot{c}(\sigma) = \frac{1}{\tau}(-c(\sigma) + c_{\max}\hat{c}_{in}(\sigma))$$

$$\hat{c}_{in}(\sigma) = \begin{cases} 1 & \text{if } v_i = 0 \text{ and } i \text{ is located at } \sigma \\ 0 & \text{otherwise} \end{cases}$$

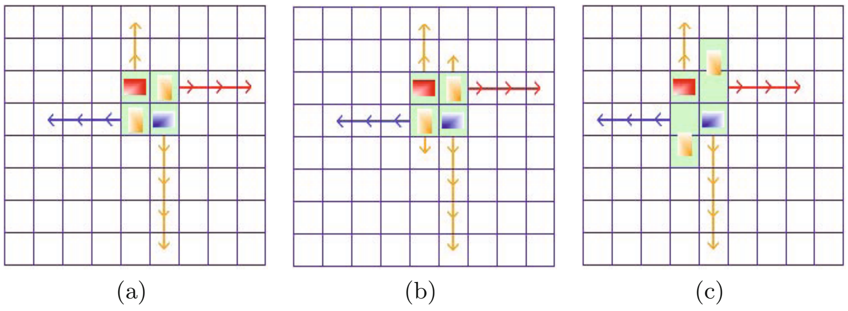


Fig. 8 A deadlock scenario resolved with the AHA. *a)* Four vehicles are put in a deadlock scenario. Each vehicle blocks the path of its counter-clockwise neighbor. (The blue vehicle blocks the path of the orange vehicle, which itself blocks the path of the red one, etc.). *b)* The vehicles re-plan. The two orange vehicles plan on moving backwards one step and heading to their destinations after the others pass. *c)* The conflict is resolved. Note that the solution is based on accurate timing between the vehicles. Solving the same deadlock involves deviations in both the RBA and the FHA.

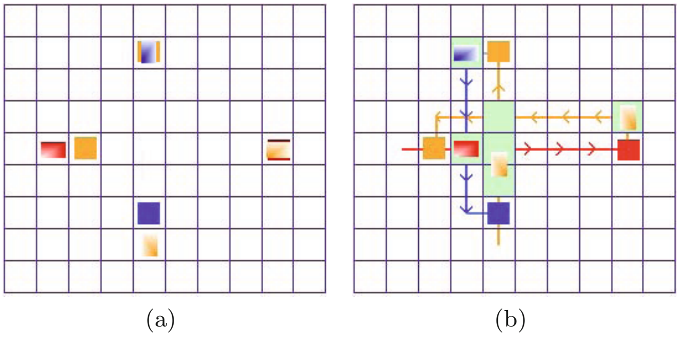


Fig. 9 A four vehicle scenario using the AHA. The big squares are the destinations and the rectangles are the vehicles. *a)* The vehicles face a mixture of head-to-head and head-to-side conflicts *b)* The vehicles deviate to avoid head-to-head conflicts and wait on their paths to resolve head-to-side conflicts.

requiring two more parameters τ, c_{max} to be optimized. The transition cost is further modified to:

$$\mathcal{L}(\sigma_k^i, d_k^i) = \tau_{min}(\mathcal{F}(\sigma_k^i, d_k), \sigma_k^i) + \delta(\sigma_k^i, d_k^i, \zeta_{-i}) + c(\sigma_k^i)$$

The main impact of the congestion avoidance is to add robustness to the proposed algorithms.

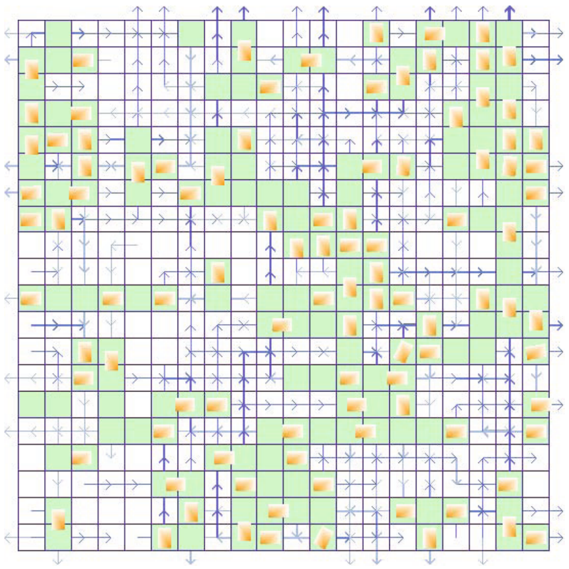


Fig. 10 A view of the platform in operation: the green areas are the reserved spaces and the arrows represent the trajectories. The thickness of the lines are proportional to number of trajectories passing through a certain cell.

5 Results

We developed a C++ platform (see Fig. 10) to study the performance of the proposed algorithms. The simulation environment is a test-bed that captures the path planning aspects of a large-scale multi vehicle system. To examine the efficacy of the proposed algorithms, we conducted several systematic experiments. Results are provided in this section. Fig. 11 demonstrates the performance of different algorithms presented above. Experiments were carried out on a 20×20 grid, on systems ranging from twenty to two hundred vehicles in size. The parameters of each algorithm were optimized at each density using a line search algorithm. It should be pointed out that, in all algorithms, if a vehicle does not move for a certain amount of time $\hat{t}$, it replans again to avoid congestions. As can be seen from the results, the AHA demonstrates superiority over the other methods. Hence, we consider it as the main approach for further analysis. The parameters required to be optimized for the algorithms are presented in Table 1 followed by a description for each parameter.

Furthermore, the algorithm is promising from the computation time point of view. As shown in Table 2, the algorithm performs seven times faster than real time in an experiment on a 50×50 grid with 625 vehicles, corresponding to a density of 0.25. The computation times provided in Table 2 can give some insight on the real-time implementability of the algorithms. Experimental

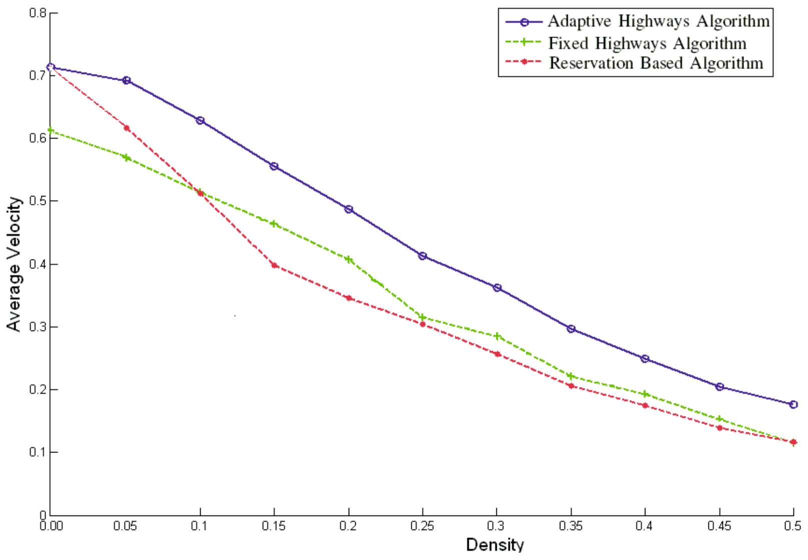


Fig. 11 Comparing the performance of different algorithms in simulation.

Table 1 The parameters of the algorithms. The optimization is done based on the average speed of the vehicles after 200 seconds of simulated time using a line search algorithm at a density of 0.25 on a 20×20 grid. $\hat{t}$ is the maximum wait time before re-planing as explained earlier in this section. τ and c_{max} are the time constant and maximum cost of congestion. $\hat{\delta}$ is the cost associated with conflicts in the RBA and the FHA. $\hat{e}$ is the estimation horizon. $\hat{\delta_h}$ is the cost associated with head-to-head conflicts in the AHA. As expected, this value is higher than $\hat{\delta_s}$, which corresponds to head-to-side conflicts. $\hat{\delta_0}$ corresponds to head-to-back encounters where vehicles plan to move toward the same direction. An optimal value of 0 implies that the algorithm encourages platooning.

Algorithm	$\hat{t}$	τ	$\hat{e}$	c_{max}	$\hat{\delta}$	$\hat{\delta_h}$	$\hat{\delta_s}$	$\hat{\delta_0}$
RBA	$2s$	$1s$	2	15	10	-	-	-
FHA	$2s$	$1s$	1	15	8	-	-	-
AHA	$2s$	$1s$	-	10	-	13	2	0

evidence suggests that the complexity of the algorithm grows almost linearly with respect to the number of the vehicles and the size of the grid.

Fig. 12 shows the robustness of the AHA with respect to individual vehicle failures. The conducted experiment concerns the performance of the AHA in the presence of inactive vehicles, i.e., the vehicles that do not move and do not interact with the central hub. In the studied case every vehicle becomes inactive once in every five missions and stays inactive for one mission time. The mission time is the average time required for the vehicles to move from a starting cell to a destination cell.

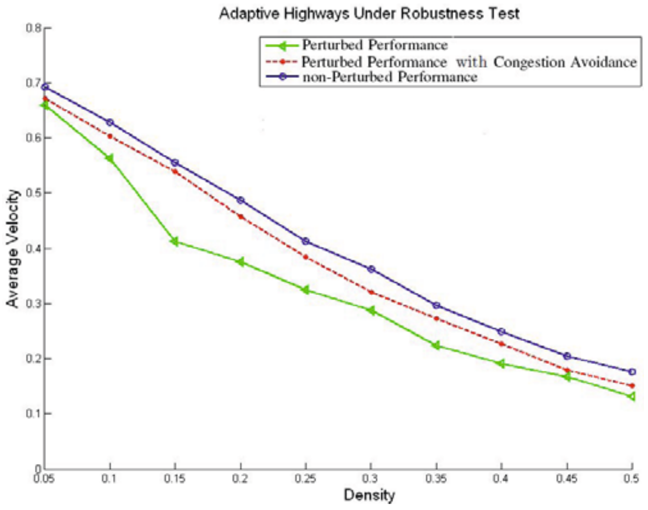


Fig. 12 The robustness test. The congestion avoidance scheme can help retrieve the performance in the presence of vehicle failures. Note that the performance is averaged over the working vehicles, and that the inactive vehicles are not counted. The small loss in performance is inevitable as the congestion is perceived with some delay.

Table 2 Comparing the running time of the algorithms. This experiment lasts 200 seconds of simulated time and is conducted on a laptop with 2.2GHZ Intel CPU and 2GB memory. The provided numbers are the ratio of the simulation time with respect to the running time of each algorithm. The simulation is carried out in a 50×50 environment with 625 vehicles, corresponding to a density of 0.25. The experiment suggests that the AHA can be considered as a real time algorithm, although it is slower than its fixed counterpart.

Algorithm	Simulation Time/Real Time
RBA	50
FHA	28
AHA	7

6 Discussion

While the AHA is shown to be promising in the studied context, there is no claim on the global optimality of the algorithm. Simple scenarios can be constructed to show that policies extracted from the algorithm correspond to a local minima of the global optimization problem. Fig.13a shows such an example. Iterative planning is further introduced here to deal with such cases (see Fig.13b). In this case, the vehicles plan in multiple iterations. Initially, the costs (δ_h,δ_s, and δ_0) are set to zero and a trajectory is generated and

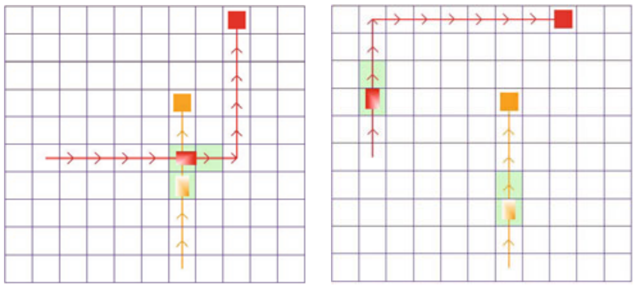

(a) Adaptive Highways Algorithm (b) Adaptive Highways Algorithm with iterative planning

Fig. 13 The role of iterative planning. *a) Adaptive Highways Algorithm:* The red vehicle, which has two equally good paths to reach its destination, plans first. The orange vehicle plans next. The solution requires the orange vehicle to wait in the middle of its trajectory until the red vehicle passes. Observe that the fixed planning order in the AHA leads to sub-optimal strategies. *b) Adaptive Highways Algorithm with iterative planning :* Each vehicle plans twice. In the first iteration, all costs are zero. Therefore, each vehicles plans as if there is no other vehicle around, leading to the same trajectories as those in Fig.13a. In the second iteration, the cost for the head-to-side conflict increases to 1. The red vehicle predicts the conflict and chooses the alternative conflict-free trajectory. The orange vehicles keeps its previous trajectory, which is now conflict-free. In this scenario, iterative planning yields the globally optimal solution.

stored in the central hub for each vehicle. The costs are then increased by some constant and this process repeats until the costs reach their optimal value.

Finally, despite its relative complexity compared to other algorithms studied in this work, the AHA is still a real-time algorithm that can be implemented on systems comprised of large number of autonomous vehicles such as the Kiva MFS. The flexibility of the algorithm to work with a scalable number of vehicles can be further exploited to design even larger systems through distribution of computation and communication load. Furthermore, the robustness test suggests that the AHA algorithm, when combined with congestion avoidance, demonstrates a high level of robustness with respect to vehicle failures–a feature that is of high interest in practical applications.

References

1. Frazzoli, E., Dahleh, M.A., Feron, E.: Real-time motion planning for agile autonomous vehicles. AIAA Journal of Guidance, Control, and Dynamics 25(1), 116–129 (2002)
2. Guizzo, E.: Three engineers, hundreds of robots, one warehouse. IEEE Spectrum 45(7), 22–29 (2008)

3. Kator, C.: Staples' robotic retrievers offer new take on break-pack picking. In: Modern Materials Handling (2007)
4. Lavalle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
5. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. In: Proceedings of the 1999 IEEE International Conference on Robotics and Automation, Denver, CO (August 2000)
6. Nagy, T.K., D'Andrea, R., Ganguly, P.: Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle. Robotics and Autonomous Systems 46, 47–64 (2004)
7. Purwin, O., D'Andrea, R.: Trajectory generation and control for four wheeled omnidirectional vehicles. Robotics and Autonomous Systems 54(1), 13–22 (2006), doi:10.1016/j.robot.2005.10.002
8. Purwin, O., D'Andrea, R., Lee, J.W.: Theory and implementation of path planning by negotiation for decentralized agents. Robotics and Autonomous Systems (2007), doi:10.1016/j.robot.2007.09.020.
9. Raffard, R.L., Tomlin, C.J., Boyd, S.P.: Distributed optimization for cooperative agents: Application to Formation Flight. In: 43rd IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas, December 14-17 (2004)
10. Schouwenaars, T., How, J., Feron, E.: Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, August 16-19. Providence, Rhode Island (2004)
11. Wurman, P.R., D'Andrea, R., Mountz, M.: Coordinating hundreds of cooperative, autonomous vehicles in warehouses. AI Magazine 29(1), 9–19 (2008)