

Safe and Distributed Kinodynamic Replanning for Vehicular Networks

Kostas E. Bekris

Konstantinos I. Tsianos

Lydia E. Kavraki

Computer Science Department, Rice University, Houston, TX, 77005

Email: {bekris, konstantinos, kavraki}@cs.rice.edu

Abstract—This work deals with the problem of planning collision-free motions for multiple communicating vehicles that operate in the same, partially-observable environment in real-time. A challenging aspect of this problem is how to utilize communication so that vehicles do not reach states from which collisions cannot be avoided due to second-order motion constraints. This paper initially shows how it is possible to provide theoretical safety guarantees with a priority-based coordination scheme. Safety means avoiding collisions with obstacles and between vehicles. This notion is also extended to include the retainment of a communication network when the vehicles operate as a networked team. The paper then progresses to extend this safety framework into a fully distributed communication protocol for real-time planning. The proposed algorithm integrates of sampling-based motion planners with message-passing protocols for distributed constraint optimization. Each vehicle uses the motion planner to generate candidate feasible trajectories and the message-passing protocol for selecting a safe and compatible trajectory. The existence of such trajectories is guaranteed by the overall approach. The theoretical results have been also experimentally confirmed with a distributed simulator built on a cluster of processors and using applications such as coordinated exploration. Furthermore, the experiments show that the distributed protocol has better scalability properties when compared against the priority-based scheme.

I. INTRODUCTION

Autonomous vehicles have long been the focus of robotics research. The progress in wireless networking allows to consider groups of vehicles that operate in the same environment and use communication to coordinate their motion. Moreover, it gives rise to the idea of networks of vehicles that jointly solve a task while retaining connectivity. Using such teams of multiple, coordinating vehicles offers redundancy and robustness in the execution of many tasks (e.g. space exploration, autonomous demining). Nevertheless, the control of such systems involves multiple research challenges.

In this paper, the focus is on motion planning and coordination challenges. Given procedures for updating a vehicle's map, state and goal, the objective is to design feasible, collision-free trajectories for multiple vehicles operating in the same, partially-known environment (see Fig. 1). In particular, we deal with the safety concerns that arise due to the kinodynamic motion constraints (e.g., bounded velocity and acceleration, smooth steering) that real vehicles exhibit. We study how inter-vehicle communication [1], [2] can be utilized in this context to achieve safe motion coordination.

Part of the material presented in this paper has previously appeared in two conference publications by the same authors: "A Decentralized Planner that Guarantees the Safety of Communicating Vehicles with Complex Dynamics that Replan Online" at IROS 2007 and "A Distributed Protocol for Safe Real-Time Planning of Communicating Vehicles with Second-Order Dynamics" at ROBOCOMM 2007.

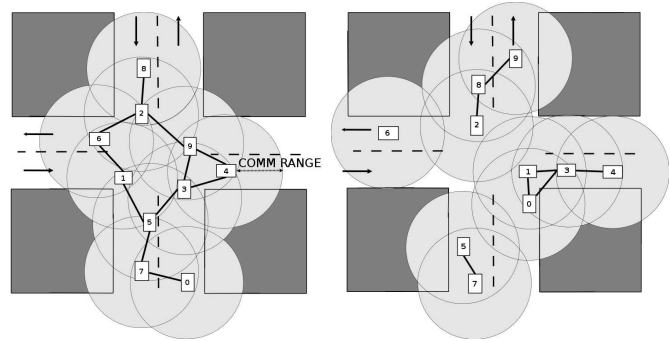


Fig. 1. Vehicles form a communication network while they move. On the left, there is one connected component while on the right vehicles have moved and multiple components have been created. Planning for such dynamic networks with centralized approaches has been studied for first-order systems [3], [4]. This paper extends these ideas by considering second order dynamics (we guarantee avoidance of Inevitable Collision States) and describing a decentralized solution using only local information.

We are interested in a solution with the following characteristics: (i) A general and abstract algorithm that is not limited to specific system dynamics or to specific types of workspaces and obstacles. (ii) A scalable, distributed solution that respects the physical limitations in sensing and communication and avoids centralized computation. (iii) A real-time algorithm, since vehicles do not typically have global knowledge of their workspace. This means that sensing, planning and execution are interleaved and there is limited amount of time to compute a partial plan towards the goal. (iv) A safe solution for systems with second-order constraints. The algorithm must provide guarantees for collision-avoidance and the retainment of a communication network if desired by the team.

A. Related Literature

Decentralized Planning - Formation Control: Multiple techniques exist for decentralized motion planning [5]. In formation control agents move while maintaining preassigned relative positions, which can be achieved with potential-fields [6], [7], [8], leader-follower approaches [9], [10] or local control laws [11]. Decentralized, navigation functions [12], [13] provide a feedback solution and can be used for vehicles with independent goals. Most of these methods focus on providing elegant stability proofs. Despite their elegance, it is difficult to apply them in general state spaces (e.g. complex obstacle and robot shapes and dynamics) [14].

Sampling-based Kinodynamic Planning: This paper investigates an alternative which is less dependent on the system's dynamics or the obstacle types. It utilizes sampling-based [15], [16], kinodynamic planning [17] popularized by al-

gorithms such as RRT [18], [19], [20]. Instead of constructing control laws given representations of the state space obstacles, such algorithms execute a search in the state space, which is mostly a computational rather than an analytical challenge. Their drawback is that they have weaker completeness properties. Optimality guarantees are also abandoned in favor of practical performance and generality.

Replanning: The original sampling-based planners were offline methods and assumed known workspaces. Planning with partial-observability requires interleaving sensing, planning and execution, where a planner is called frequently and has finite time to replan a trajectory. Replanning from scratch is possible [21] but recent methods use information from previous planning cycles to speed up the performance of replanning [22], [23], [24], [25]. There are also methods that use a roadmap to replan online [26], [27].

Replanning for Systems with Dynamics: When replanning with a sampling-based planner for a system with second-order dynamics, safety issues arise: a collision-free but partial plan may lead a vehicle to a state from which collisions cannot be avoided due to the dynamics (Inevitable Collision States (ICS) [28], [29], [30], [20]). This problem is particularly acute when multiple second-order vehicles operate in close proximity in the same environment. Similarly, a partial plan could also lead to states from which network connectivity will be inevitably lost. A framework that deals with ICS and real-time planning for a single vehicle has been recently developed in the sampling-based planning literature [28], [20].

Motion Coordination: The use of sampling-based planners on multi-agent problems is limited and it typically follows a centralized approach [3], [4]. *Centralized* planning is reliable [31] but computationally expensive due to the exponential dependency on problem dimensionality. *Decentralized* methods, such as prioritized schemes [32], plan separately for each robot and then coordinate the robots' interactions. Planning is orders of magnitudes faster but can lead to collisions [31], [3]. An important challenge is how to make decentralized planning more reliable, especially when using sampling-based planners [33]. In previous work [34], we showed that it is possible to achieve safety in a decentralized scheme that employs priorities even with second-order dynamic constraints.

Message-Passing Coordination: An alternative to priority-based schemes for decentralized solutions can be found in the AI literature. There are many techniques for distributed constraint satisfaction [35] and optimization [36] for teams of cooperating agents, such as factored Markov Decision Processes [37] and auction mechanisms [38], [39]. This paper employs message-passing, asynchronous algorithms for coordination related to loopy belief propagation, a method for distributed optimization in constraint networks [40], [41]. These message passing algorithms have been successfully applied to solve distributed inference problems in wireless sensor networks [42]. Employing such message-passing algorithms for coordinating vehicular networks results in better scalability than priority-based schemes [43].

B. Contribution

This paper describes an integration of sampling-based kinodynamic planners [14], [19] with message-passing protocols [41], [42] to distributedly control the motion of multiple communicating vehicles in simulation. It is an extension of work on safe, real-time sampling-based planning [20] to the case of multiple networked vehicles with limited communication range. It provides with theoretical safety guarantees in terms of avoiding inevitable collision or loss of connectivity states. Compared to alternative approaches for decentralized planning [11], [12], it is easily implementable on different environments and systems with various dynamics. In contrast to existing work on planning for dynamic networks, where coordination is centralized [3], the approach aims to distribute computation.

The starting point for the method is to identify information that when exchanged between the vehicles is sufficient for the planning of safe trajectories. These identified sufficient conditions dictate our approach. Each vehicle uses a sampling-based planner [20] to generate feasible trajectories that allow the existence of safe alternatives to other vehicles. For coordinating the selection of compatible trajectories between vehicles, we initially present a priority-based scheme. This allows us to provide a proof that the vehicles always have safe trajectories to follow. The priority scheme is then replaced by an asynchronous, message-passing protocol [41], [42], which provides the same theoretical safety guarantees. Among the safe solutions and given the available time, the asynchronous protocol optimizes a joint payoff function.

The proposed method has been implemented on a multi-processor simulator. Each processor models a vehicle and communicates asynchronously with other processors. The experimental results confirm the theoretical guarantees of collision avoidance and network retainment for second-order vehicles jointly exploring an unknown workspace. The distributed protocol has computational and scaling advantages when compared against the prioritized scheme [34], [43].

II. PROBLEM SETUP

Consider vehicles $V = \{V_1, \dots, V_v\}$ operating in a world with obstacles. Both obstacles and vehicles are rigid-bodies and there are no restrictions on their shape. Each vehicle is able to sense a local region around it and can communicate with other vehicles within a limited range. Each vehicle V_i is a dynamic system whose motion is governed by differential equations of the form:

$$\dot{x}_i(t) = f(x_i(t), u_i), \quad g(x_i(t), \dot{x}_i(t)) \leq 0 \quad (1)$$

where $x_i(t) \in \mathcal{X}_i$ represents a state, $u_i \in \mathcal{U}_i$ is a control, f, g are smooth and t is time. This paper focuses on systems with bounds both in velocity and acceleration. The dynamics of the systems we experimented with can be found in Section VII.

Given the communication limitations and states $\{x_1(t), \dots, x_v(t)\}$, the vehicles form dynamic communication links represented by a graph $G(t) = \{V(t), E(t)\}$, where $e_{ij} \in E(t)$ as long as V_i, V_j are within range. The neighbors of V_i in the graph $G(t)$ are denoted as $N_i(t)$.

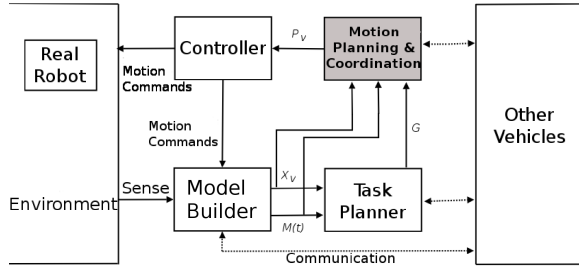


Fig. 2. The closed loop architecture and modules on a single vehicle.

A. High-Level Operation

The vehicles execute tasks which require motion. While moving, the vehicles must avoid collisions both with obstacles and with other vehicles. Since the workspace is only partially observable, the vehicles must update their world model and state estimate given new sensory information. In parallel, they must compute in real-time trajectories towards their goals. To achieve this objective, a vehicle's function is broken down into a sequence of consecutive operational cycles. The various vehicle operations, shown in Fig. 2, are executed in a pipeline over these cycles. For cycle $(t : t + dt)$ vehicle V_i executes the following:

1. Up to time t : a localization and mapping routine updates the map $M_i(t)$ and estimates future state $x_i(t + dt)$.
2. Given the map, a goal $G_i(t)$ is computed for V_i .
3. Given $M_i(t)$, $G_i(t)$ a planner must compute a plan $p(dt)$ before $t + dt$.
4. At time $t + dt$ the plan $p(dt)$ is executed at $x_i(t + dt)$.

In this work we focus on step 3, i.e. on how to utilize communication so as to provide safety guarantees when multiple vehicles operate in close proximity. Each vehicle can only communicate with neighboring vehicles given the communication constraints and exchange information. We will specify what kind of information has to be exchanged to guarantee collision avoidance. In this work, we do not deal with issues related to uncertainty in sensing and action as well as unreliable communication.

B. Motion Planning Notation

The objective of the motion planner is to compute a **plan** $p(dt)$, which is a time sequence of controls: $p(dt) = \{(u_1, dt_1), \dots, (u_n, dt_n)\}$, where $dt = \sum_i dt_i$. When a plan $p(dt)$ is executed at state $x(t)$, a vehicle will follow the **trajectory**: $\pi(x(t), p(dt))$. A trajectory is feasible, if it respects the constraint functions f and g from Eq. 1.

A state along trajectory $\pi(x(t), p(dt))$ at time $t' \in [t : t + dt]$ is denoted as $x^\pi(t')$. When a vehicle executes a plan $p(dt)$ from state $x(t)$ and consecutively executes plan $p'(dt')$, then the resulting **trajectory concatenation** will be denoted as:

$$\pi'(\pi(x(t), p(dt)), p'(dt')).$$

If two vehicles V_i, V_j at time t are not in collision with each other or with obstacles, then their corresponding states $x_i(t), x_j(t)$ are **compatible states**: $x_i(t) \asymp x_j(t)$.

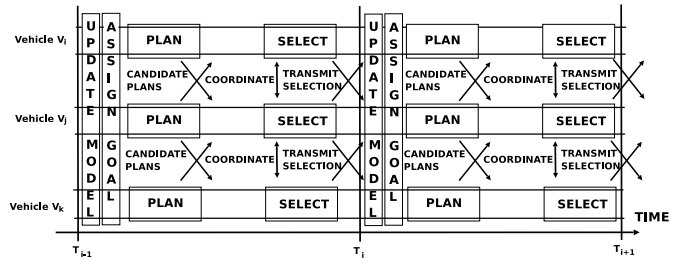


Fig. 4. The operation that a single vehicle executes in two consecutive planning cycles.

Two trajectories $\pi_i(x_i(t_i), p_i(dt_i))$ and $\pi_j(x_j(t_j), p_j(dt_j))$ are **compatible trajectories** ($\pi_i \asymp \pi_j$) if the two trajectories do not cause collisions with workspace obstacles and:

$$\forall t' \in [\max\{t_i, t_j\} : \min\{t_i + dt_i, t_j + dt_j\}] : x^{\pi_i}(t') \asymp x^{\pi_j}(t').$$

Compatible trajectories between vehicles may still lead to an inevitable collision state from which a collision cannot be avoided in the future due to second-order constraints [28]. A state $x_i(t)$ is an **Inevitable Collision State (ICS)** given the states $\{x_1(t), \dots, x_v(t)\}$ if $\forall \pi_i(x_i(t), p_i(\infty))$:

$$\exists (dt \wedge j \neq i) \text{ so that } \forall \pi_j(x_j(t), p_j(\infty)) \text{ states } x^{\pi_i}(dt) \text{ and } x^{\pi_j}(dt) \text{ are not compatible.}$$

C. Problem Definition

Given the map $M_i(t)$ and a state estimate $x_i(t + dt)$, the motion planning module of each vehicle V_i must compute before time $(t + dt)$ a plan $p_i(dt')$ so that given the trajectories of all other vehicles $\pi_j(x_j(t + dt), p_j(dt'))$ ($\forall j \neq i$):

- $\pi_i(x_i(t + dt), p_i(dt')) \asymp \pi_j(x_j(t + dt), p_j(dt'))$
- State $x_i^{\pi_i}(dt')$ is not ICS.

Vehicle V_i can only communicate with vehicles in the set $N_i(t)$. A secondary objective for the planner is to refine the quality of the selected trajectory given a measure of path quality and a goal $G_i(t)$.

III. APPROACH OVERVIEW

The proposed approach has two characteristics. Motion coordination is achieved in a decoupled manner. This feature distinguishes our approach when compared against related work on planning for communicating vehicles [3], where the vehicles forming a temporary dynamic network solve a centralized problem. Moreover, the motion planning and coordination operation of each vehicle are split into two separate steps as shown in Fig. 4:

- 1) **Generate Candidate Plans**: During the first step, the algorithm searches the state space of each vehicle V_i so as to generate a set of candidate plans \mathcal{P}_i by employing a single-vehicle planner.
- 2) **Select Compatible Plans**: During the second step, neighboring vehicles communicate by exchanging sets \mathcal{P}_i and evaluating their performance in terms of collision avoidance and task execution.

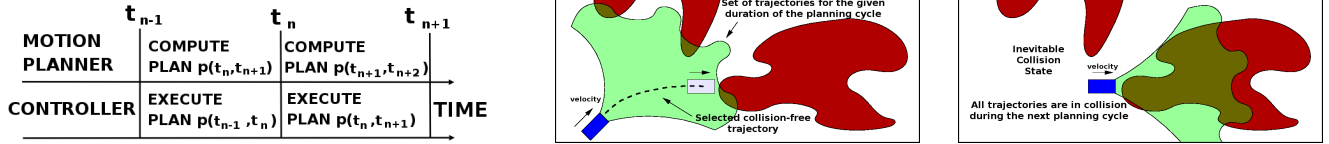


Fig. 3. (left) The robot’s synchronization scheme. (center and right) A valid plan may still lead to an ICS during the next planning cycle.

For the plan generation step, sampling-based, kinodynamic planners are particularly appropriate to search the state space and produce multiple candidate valid plans \mathcal{P}_i that are at least collision-free with the workspace obstacles. Section IV outlines the sampling-based kinodynamic planner that a single vehicle can use to replan and avoid ICS.

Two plans of different vehicles are acceptable solutions if the corresponding trajectories are compatible and they must be selected appropriately through coordination in the second step. Section V describes how such coordination can be achieved by employing a priority-based scheme. It also shows how safety is guaranteed assuming the preassigned priorities and no communication limitations. These assumptions will be later waived. In particular, in Section VI, the exchanged plans \mathcal{P}_i are viewed as actions in a discrete action space. Then the problem of distributedly selecting compatible trajectories is reduced to a distributed constraint optimization problem, for which message-passing algorithms without priorities exist [41], [42].

IV. SAFE REPLANNING FOR A SINGLE VEHICLE

In order to generate candidate plans (step 1 of the overall approach), a motion planner is employed that focuses on planning paths for a single vehicle.

The single vehicle approach incrementally expands a tree data structure in the vehicle’s state-time space and returns safe paths given a partially-known workspace and kinodynamic constraints [20]. There are two elements of the approach relevant to the multi-robot problem: (i) The planner’s operation is broken into consecutive replanning cycles. (ii) Within a cycle, the planner avoids ICS.

During cycle $(t_{n-1} : t_n)$, the planner uses an updated model of the world up to time t_{n-1} and an estimate of the state $x(t_n)$ at the beginning of the next planning cycle $(t_n : t_{n+1})$. Given a goal, the planner computes a new plan before t_n that will be executed during the consecutive cycle: (t_n, t_{n+1}) . This is achieved by expanding a tree data structure (*Tree*) in the vehicle’s state-time space using a sampling-based approach [20], [18], [21], [44]. From the expanded tree, a valid plan $p(t_n : t_{n+1})$ that results in the trajectory $\pi(x(t_n), p(t_n : t_{n+1}))$ must be selected.

It is not sufficient for $\pi(x(t_n), p(t_n : t_{n+1}))$ to be just collision-free, since it may lead to an ICS [28], as Fig. 3 (right) demonstrates. It is computationally intractable, however, to check if a state is truly ICS or not: all possible plans out of that state have to be examined to determine if there is an escape plan. It is sufficient, however, to take a conservative approach: if the vehicle can avoid collisions by executing a

Algorithm 1 SAMPLING-BASED PLANNER

```

Tree ← Retain valid subset of Tree from previous cycle
while (time < PlanningBudget) do
{
  Select a state  $x(t')$  on the existing Tree
  Select valid plan  $p(dt')$  given state  $x(t')$ 
  Forward propagate trajectory  $\pi(x(t'), p(dt'))$ 
  if ( $\pi$  is not collision-free with obstacles) then
    Reject  $\pi$ 
  else
    Add  $\pi$  to Tree
}

```

pre-specified “contingency” plan $\gamma(\cdot)$ out of a state x , then x is safe. In other words, state x is **safe** iff:

$$\exists \gamma(\infty) \text{ s.t. } \pi(x, \gamma(\infty)) \text{ is collision free.} \quad (2)$$

In our experiments, the contingency plan we use for car-like vehicles is a breaking maneuver that brings the car to a complete stop as fast as possible. The duration of a contingency plan depends on the vehicle’s velocity at state x and its acceleration bound. Since the planner is required to return a plan only for the period $(t_n : t_{n+1})$, only the states along the tree that occur at time t_{n+1} have to be checked whether they are safe or not. The planner implements the following invariant for all plans $p(t_n : t_{n+1})$ along the tree:

$$\pi(\pi(x(t_n), p(t_n : t_{n+1})), \gamma(\infty)) \text{ is collision-free.}$$

This means that for all plans $p(t_n : t_{n+1})$ there is a concatenation with contingency plans that leads to collision-free trajectories. With this method, a vehicle can operate in a partially-known workspace with static obstacles and can avoid collisions at all times [20].

V. SAFE DECENTRALIZED REPLANNING WITH PRIORITIES

We now move on to the case of multiple vehicles operating in the same environment. This section initially assumes vehicles that have unlimited communication range and preassigned priorities. Both of these assumptions will be waived later.

We will start from a simple extension of the single-vehicle algorithm to a coordinated approach. As Fig. 1 shows, communication links between vehicles define a graph, where the vehicles are nodes and two vehicles share an edge if the two vehicles can exchange messages. In the case of unlimited communication range this graph is complete. Suppose every vehicle has a unique global priority. We define the set N^h to represent the neighbors of vehicle V on the communication

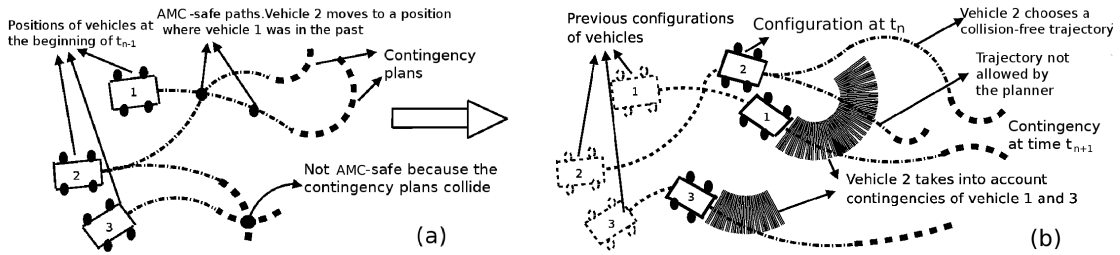


Fig. 5. (a) The lower plan for V_2 is not safe since the contingency attached to it collides with the contingency extending from the plan of V_3 . The top plan of V_2 is safe. (b) The planner of V_2 will not produce the lower trajectory because it collides with the current contingency of V_1 . The top plan is again safe.

graph with higher priorities than V and the set N^l to be the set with lower priorities. Then the simple prioritized scheme executed on each vehicle V during a single planning cycle ($t_n : t_{n+1}$) employs the following steps:

- 1) Compute a set of candidate plans \mathcal{P} of duration ($t_n : t_{n+1}$) with the single-vehicle algorithm.
- 2) Receive the selected plans \mathcal{P}^h from neighbors in N^h .
- 3) Select plan $p(t_n : t_{n+1}) \in \mathcal{P}$ that does not collide with plans in \mathcal{P}^h and best serves the goal of vehicle V .
- 4) Transmit the plan p to all neighbors N^l .

The simple extension, however, fails to produce safe trajectories for multiple reasons:

- If a cycle is completed before all higher priority plans are received, no plan p can be safely selected.
- Even if $p \in \mathcal{P}$ and \mathcal{P}^h are available on time, it may happen that no plan p is collision-free with all plans in \mathcal{P}^h due to the decentralized nature of the approach.
- Suppose p is collision-free with set \mathcal{P}^h . It may still lead to ICS given \mathcal{P}^h due to the dynamics.

The definition of a safe state from Eq. 2 is inadequate in the multi-vehicle case, where the safety of a vehicle's state depends on the states and the choices of the other vehicles. We extend the definition of safety as follows:

Safe State - Multi-vehicle case: Consider vehicles V_1, \dots, V_v that have states $x_1(t), \dots, x_v(t)$ and all vehicles $V_j, j \neq i$ execute plans $p_j(dt)$. Then state $x_i(t)$ is safe iff $\exists \gamma_i(\infty)$ so that:

$$\pi_i(x_i(t), \gamma_i(\infty)) \text{ is collision free} \wedge \forall j \neq i : \pi_i(x_i(t), \gamma_i(\infty)) \asymp \pi'_j(\pi_j(x_j(t), p_j(dt)), \gamma_j(\infty)) \quad (3)$$

Note that the trajectory $\pi_i(x_i(t), \gamma_i(\infty))$ should be compatible with the concatenation of other vehicles' plans and contingencies. Given this new definition of a safe state, we set an objective for the coordination algorithm we described earlier. To guarantee safety, it is sufficient to satisfy the following.

Invariant: For each replanning cycle ($t_n : t_{n+1}$) every vehicle V_i selects a plan $p_i(t_n : t_{n+1})$ which when executed at state $x_i(t_n)$:

- a. The resulting trajectory $\pi_i(x_i(t_n), p_i(t_n : t_{n+1}))$ is collision-free.
- b. During the current cycle ($t_n : t_{n+1}$), it is compatible with all other vehicles, $\forall j \neq i$:

$$\pi_i(x_i(t_n), p_i(t_n : t_{n+1})) \asymp \pi_j(x_j(t_n), p_j(t_n : t_{n+1})).$$

- c. It leads to state $x^{\pi_i}(t_{n+1})$ that is safe according to Eq. 3 for every choice of plans $p_j(t_{n+1} : t_{n+2})$ that the other vehicles may make during the next planning cycle.

If the Invariant holds then the algorithm will produce safe trajectories. Points a. and b. imply that there is no collision during the current cycle ($t_n : t_{n+1}$), either with static geometry or between vehicles. Point c. implies that all vehicles at the next cycle ($t_{n+1} : t_{n+2}$) have contingency plans which can be followed regardless of the other vehicles' choices. Consequently, the prioritized algorithm in the beginning of this section can be altered so that step 3 is:

- 3) Select plan $p(t_n : t_{n+1}) \in \mathcal{P}$ that satisfies the Invariant given the set \mathcal{P}^h . If no such plan exists or time is running out, execute contingency $\gamma(t_n : t_{n+1})$, which is precomputed from the previous planning cycle and collision-free due to the Invariant.

Consequently, now we need to answer the question of: *how to produce and select plans $p(t_n : t_{n+1})$ that satisfy the Invariant.* To guarantee the Invariant it is sufficient for any selected plan at step 3 of the algorithm to satisfy:

Condition 1: As in the single-vehicle case, the concatenation of plan $p_i(dt)$ with a contingency plan $\gamma_i(\infty)$ must be collision-free:

$$\pi'(\pi(x(t_n), p(t_n : t_{n+1})), \gamma(\infty)) \text{ is collision-free.} \quad (4)$$

Condition 2: The concatenation of plan $p_i(dt)$ with a contingency plan $\gamma_i(\infty)$ must be compatible with the contingency plans $\gamma_j(\infty)$ of other vehicles:

$$\forall j \neq i : \pi'_i(\pi_i(x_i(t_n), p_i(dt)), \gamma_i(\infty)) \asymp \pi_j(x_j(t_n), \gamma_j(\infty)) \quad (5)$$

Condition 3: The concatenation of plan $p_i(dt)$ with a contingency plan $\gamma_i(\infty)$ must be compatible with the concatenations of plans $p_j(dt)$ of other vehicles with their contingency plans $\gamma_j(\infty)$:

$$\forall j \neq i : \pi'_i(\pi_i(x_i(t_n), p_i(dt)), \gamma_i(\infty)) \asymp \pi'_j(\pi_j(x_j(t_n), p_j(dt)), \gamma_j(\infty)) \quad (6)$$

Theorem: Assume the Invariant is satisfied during planning cycle ($t_{n-1} : t_n$) for all vehicles. Then if each vehicle V_i selects a plan $p_i(t_n : t_{n+1})$ that satisfies Eq. 4, 5 and 6 or selects an available contingency plan, then the Invariant will also hold during the next planning cycle ($t_n : t_{n+1}$).

Proof: We will have to show that the three points of the Invariant are satisfied during the next planning cycle ($t_n : t_{n+1}$). There are two cases. Either the algorithm manages to produce and select a plan $p_i(t_n : t_{n+1})$ that satisfies Eq. 4, 5 and 6 or selects a contingency plan. We will treat these two cases separately:

1) Assume such plan $p_i(t_n : t_{n+1})$ has been found. Because the plan satisfies Eq. 4 and Eq. 6, points a. and b. of the Invariant are satisfied, respectively. Point c. is more complicated. The state $x(t_{n+1})$ that the vehicle will reach after executing $p_i(t_n : t_{n+1})$ must have the property that it is safe according to Eq. 3. The application of the contingency plan γ_i at state $x(t_{n+1})$ will result in a collision-free path according to Eq. 4, so one of the two specifications of Eq. 3 is satisfied. State $x(t_{n+1})$ has to be safe, however, for any choice of plans $p_j(t_{n+1} : t_{n+2})$ that the other vehicles will make during the next planning cycle. There are again two possible cases for the nature of plans another vehicle V_j can follow during cycle ($t_{n+1} : t_{n+2}$):

a. Assume vehicle V_j computes a plan $p_j(t_{n+1} : t_{n+2})$ that satisfies the conditions. Then due to Eq. 5, this plan is compatible with the contingency of V_i during that cycle:

$$\pi_i(x^{\pi_i}(t_{n+1}), \gamma_i(\infty)) \asymp \pi'_j(\pi_j(x^{\pi_j}(t_{n+1}), p_j(t_{n+1} : t_{n+2})), \gamma_j(\infty))$$

b. Assume vehicle V_j resorts to a contingency during cycle ($t_{n+1} : t_{n+2}$). Due to Eq. 6, however, the contingency of V_j is by construction compatible with the contingency of V_i :

$$\pi_i(x^{\pi_i}(t_{n+1}), \gamma_i(\infty)) \asymp \pi_j(x^{\pi_j}(t_{n+1}), \gamma_j(\infty))$$

In any case, Eq. 3 is satisfied for state $x^{\pi_i}(t_{n+1})$, which means that the third point of the Invariant is also satisfied for the next planning cycle.

2) Assume that vehicle V_i has to resort to a contingency. The inductive hypothesis is that the Invariant holds during the current cycle, so the state $x(t_n)$ is safe according to Eq. 3 for every choice of plans of other vehicles. From Eq. 3 the points a. and b. of the Invariant trivially hold for the trajectory that follows the contingency plan. In order to show that the state $x(t_{n+1})$ reached after the application of the contingency plan $\gamma_i(t_n : t_{n+1})$ is safe according to Eq. 3 we can follow exactly the same reasoning as above. From Eq. 4 the trajectory $\pi_i(x(t_{n+1}), \gamma_i(\infty))$ will be collision-free and will also be compatible given any choice the other vehicles will make due to Eq. 5 and 6. \square

A. Priority-based Algorithm

We describe here how we can algorithmically satisfy the specified conditions within a prioritized scheme. The algorithm in pseudo-code is given in Algorithm 2. Fig. 5 provides an illustration of how a vehicle (V_2) uses information from its neighbors (V_1, V_3) to respect conditions 2 and 3.

To satisfy the second condition, each vehicle V_i must be aware of the contingency plans of other vehicles V_j at state $x(t_n)$ during planning cycle ($t_{n-1} : t_n$). These contingency

Algorithm 2 PRIORITY-BASED SCHEME for V_i

Identify set of neighbors $N = N^h \cup N^l$

(Exchange contingencies)

for all $j \in N$ **do**

Send contingency $\gamma_i(\infty)$ to V_j

Receive contingency $\gamma_j(\infty)$ from V_j

(Planning: satisfies conditions 1,2)

$HN \leftarrow N^h$ (high priority neighbor set)

Select *PlanningBudget* according to priority

$Tree \leftarrow$ Retain valid subset of *Tree* from previous cycle

while ($time < PlanningBudget$) $\wedge HN \neq \emptyset$ **do**

{

(Sampling-Based Kinodynamic Planning)

Select an existing trajectory sample s from *Tree*

Select plan $p(dt)$ and state $x(t)$ on s

Propagate trajectory $\pi(x(t), p(dt))$

(Cond. 1: Avoid ICS with obstacles)

if ($\pi(x(t), p(dt))$ is not **collision-free**) **then**

Reject π

else

if ($t < t_{n+1}$) \wedge ($t + dt > t_{n+1}$) **then**

(path intersects next cycle t_{n+1})

if ($\pi(\pi(x(t), p(dt)), \gamma(\infty))$ not collision-free) **then**

(Leads to ICS with obstacles)

Reject π

(Cond. 2: Compatibility with $\gamma_j(\infty)$)

for all $j \in N$ and while π is not rejected **do**

if ($\pi(x(t), p(dt)) \not\asymp \pi_j(x_j(t_n), \gamma_j(\infty))$) **then**

(Does not respect Eq. 5)

Reject π

(Receive high priority plans)

if (message arrived from $j \in HN$) **then**

Receive selected plan $p_j^*(t_n : t_{n+1})$

Receive contingency $\gamma_j^*(\infty)$ at $x_j(t_{n+1})$

Remove j from *HN*

}

(Path Selection: satisfies req. 3)

$p_i^* \leftarrow \gamma_i(\infty)$ (safe from previous round)

$P' \leftarrow$ Extract all plans $p'_i(t_n : t_{n+1})$ from *Tree*

for all $p'_i \in P'$ and while ($time < PlanningCycle$) **do**

for all $j \in N^h$ **do**

if (Eq. 6 does not hold for $p'_i, p'_j(t_n : t_{n+1}), \gamma_j^*(\infty)$) **then**

Reject p'_i

if p'_i is not rejected and p'_i better than p_i^* **then**

$p_i^* \leftarrow p'_i(t_n : t_{n+1})$

(Transmit selected plan)

for all $j \in N^l$ **do**

Send selected plan p_i^* to V_j

Send contingency $\gamma_i^*(\infty)$ at $x_i(t_{n+1})$ to V_j

plans have already been computed by each V_j during the previous step. This information can be communicated at the beginning of each cycle. After exchanging contingency plans, the sampling-based, kinodynamic planner is invoked. It generates a tree data structure of feasible trajectories in the state-space that are collision-free and avoids ICS with obstacles in the beginning of the consecutive planning cycle (Eq. 4). The planner considers also in collision all the trajectories that intersect the contingencies of other vehicles to satisfy Eq. 5.

The third condition specifies that when a vehicle makes a decision it must inform the other vehicles so that pairs of plans satisfy Eq. 6. This exchange of information can follow the vehicles' priorities. Vehicle V_i with the highest priority computes a solution plan $p_i(t_n : t_{n+1})$ from the motion planner and the accompanying contingency plan that could be executed at state $x^{\pi_i}(t_{n+1})$. V_i transmits its solution to lower priorities vehicles, which must now come up with a plan that respects Eq. 6 given V_i 's choice. Every vehicle waits to receive the choices of all vehicles with higher priorities before selecting its own plan. If a plan that respects Eq. 6 is available from the tree data structure computed by the motion planner, then it is selected and transmitted to the lower priority vehicles. If no such plan is found, the available contingency plan is selected and transmitted. If time is running out (variable *PlanningCycle* in Algorithm 1) and not all higher priority vehicles have send their plans, then a contingency plan is again selected.

Note that the prioritized scheme imposes a total ordering over all the vehicles. When the communication graph is complete this results in the lowest priority vehicle having to wait for all the previous vehicles to select plans. The high priority vehicle has to transmit its selection early enough (variable *PlanningBudget* in Algorithm 1) so that the sequence of selected plans reaches all vehicles within the planning cycle along a chain of priorities. *Note that even if the PlanningBudget is not sufficiently long so that all vehicles communicate, the vehicles still do not collide in our setup.* The vehicles will end up selecting contingency plans, which correspond to stopping maneuvers and they will stop safely. This undesired effect is less pronounced when vehicles have limited communication as Fig. 6 shows, because vehicles that do not communicate do not effect one another. We have, however, addressed this advantage by proposing a fully distributed approach, described in Section VI, that guarantees the satisfaction of the three conditions without priorities.

B. Limited Communication

In this section we will waive the assumption that the vehicles have unlimited communication, so that dynamic networks are formed and dissolved as the vehicles move towards their goals, as in Fig. 1. One advantage and two complications arise in this case for the proposed methodology.

The advantage is that the flow of information resulting from the priority-based algorithm is no longer a chain by default. In the previous setup, where all the vehicles communicate one with another, the vehicle with the smallest priority has

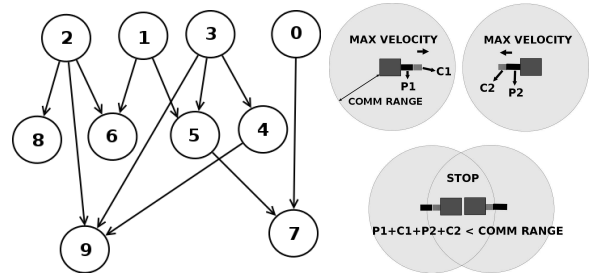


Fig. 6. (left) For the dynamic network in Fig. 1 the above DAG shows the transmission of selected plans p by high priority vehicles to lower priority vehicles - low number denote high priority. (right) Two vehicles that enter each other's comm range at maximum velocity, cannot collide if after finishing their plans they execute their contingency plans.

to wait for all the other ones to pick their plan before it can make a decision. When vehicles have limited communication, however, the chain of dependence for the units with low priority tends to be shorter. For example, the Directed Acyclic Graph (DAG) in Fig. 6 (left), represents the partial ordering defined by the priorities of the dynamic vehicular network displayed in Fig. 1(left). The DAG structure allows for the plan selection step to be executed in parallel on many vehicles even with a prioritized scheme. For example, vehicles 2, 1, 3 and 0 can immediately select their plan without having to wait for any unit of higher priority. The decision making procedures for vehicles 4 and 8 depend on different units and can potentially take place in parallel. At the same time, there are the following two complications to consider:

- 1) Is the Invariant still satisfied or do collisions occur between vehicles because of limited communication?
- 2) Is it possible to retain a communication network as the vehicles move?

This section deals with the first complication and the following section will address the second one. In the general case, collisions will occur between vehicles when communication is limited. The problem is that two vehicles that do not communicate can approach one another during one replanning cycle and collide before they have an opportunity to compare their plans and exchange contingencies.

Despite the above fact, the Invariant can still be guaranteed by imposing limits on the maximum velocity of the vehicles. For the following discussion, assume a simple communication model, which is also the one employed in the experimental section of this work. The model specifies that two vehicles V_i and V_j can communicate as long as states x_i and x_j bring each vehicle inside the communication range of the other. This simple model ignores the effects of obstacles and defines the communication region of a vehicle as a disk centered at the vehicle. If all vehicles have the same communication range C_r then the following is true:

$$V_i \text{ and } V_j \text{ communicate iff: } \|x_i, x_j\| < C_r$$

where the operation $\|\cdot, \cdot\|$ computes the distance between two states after projecting them in the Euclidean space (i.e., ignoring orientation, velocities and other state parameters).

For this simple radial communication model, Fig. 6 (right) shows the extreme case to take into account so as to satisfy the Invariant: two vehicles are just outside their communication range and they move with maximum velocity towards one another. This implies that, in the worst case, they will keep getting closer with maximum velocity for an entire cycle before being able to communicate. When they communicate, there are two possible outcomes. If they manage to find compatible plans, they can execute them safely. Otherwise, they must execute contingency plans. In order for the Invariant to be satisfiable, the resulting contingency plans must not lead to a collision, which means that the vehicles have enough time to break and come to a complete stop before colliding. Consequently, the following has to be true:

- **Sufficient Communication Condition 1:** the distance a vehicle covers when it moves at maximum velocity for one planning cycle and then applies a contingency plan until it comes to a complete stop, must be less than half of the communication range C_r .

This is a conservative condition as the case described above is the worst case scenario. However, it guarantees that the vehicles have enough time to communicate and once they communicate, they have enough time to come to a complete stop by employing their contingency plans. For some realistic parameters for car-like vehicles (communication range: 100m, breaking deceleration 10m/sec^2 , planning cycle 1sec) the allowable maximum velocity is sufficiently high (approx. 80 Km/h or 50 mph).

The proposed coordination framework does not depend on the radial communication model described above. It can also accommodate more complex communication constraints as long as the following condition is satisfied:

- **Sufficient Communication Condition 2:** Given two states x_i and x_j , there must be a deterministic function $f_c : \mathcal{X} \times \mathcal{X} \rightarrow [0, 1]$ that can compute whether or not vehicles V_i and V_j can communicate from states x_i and x_j .

Given this definition, line-of-sight constraints can also be accommodated as long as unexplored space is treated as an obstacle. Alternatively, the proposed framework can utilize techniques that build radio signal strength maps [45] as a preprocessing step, to define which pairs of states inside the environment allow the vehicles to communicate. In this way, realistic communication challenges, such as multi-path effects can also be taken into account. The above requirement, however, implies that the communication properties of an environment are stationary and do not change over time.

C. Retaining Connectivity

Given the existence of function f_c , it is possible now to describe how to maintain a communication network as the vehicles move inside the environment. Assume the vehicles form a communication graph as in Fig. 1(left) and the objective is to retain connectivity. One way to guarantee connectivity is to require that each vehicle V maintains at each replanning cycle

all the communication links with higher priority neighbors N^h . This may not be efficient, however, as it may force the vehicles to come close one to another. This solution also forces the vehicular network to retain the same topology throughout its operation.

However, to retain connectivity, it is sufficient to only retain the communication links along a spanning tree of the original communication graph $G(t) = \{V(t), E(t)\}$. A spanning tree of the graph $G(t)$ is the graph $T(t) = \{V(t), E_T(t)\}$, where $E_T(T) \subset E(t)$ so that $T(t)$ is connected and there are no loops (assuming $G(t)$ is connected). Distributed algorithms for computing approximate spanning trees have been studied extensively in wireless networks literature as they appear in multiple problems [46]. There are also many established protocols for the computation of spanning trees [47]. A distributed algorithm for the computation of $T(t)$ is employed in the beginning of every planning cycle. It is sufficient to compute an approximate spanning tree, one that also contain loops. The inclusion of unnecessary edges does not effect the network's connectivity, it only makes the topology less dynamic.

Given the approximate spanning tree $T(t) = \{V(t), E_T(t)\}$, the planning algorithm must then guarantee that the vehicles do not choose trajectories that will break the communication links $e_{ij} \in E_T(t)$ along $T(t)$. To achieve this objective we have to update the definition of compatible states.

Compatible States: Two states $x_i(t), x_j(t)$ for vehicles V_i, V_j are compatible [$x_i(t) \asymp x_j(t)$] if:

- 1) neither V_i or V_j is in collision with obstacles at states $x_i(t)$ and $x_j(t)$ respectively,
- 2) V_i and V_j do not collide one with the other at states $x_i(t)$ and $x_j(t)$,
- 3) and $\|x_i, x_j\| < C_r$ if $e_{ij} \in E_T(t)$.

Similarly, the definitions of a compatible trajectory and of a safe state have to be updated given the above definition of compatible states.

Consequently, the necessary changes in Algorithm 2 so as to provide network connectivity are the following:

- In the beginning of the planning cycle, compute which communications links have to be maintained between vehicles by distributedly computing an approximate spanning tree $T(t)$ [46].
- Add an extra check for conditions 2 and 3 given the updated definition of a compatible state.

Given the above changes and since the vehicles move, the communication graph changes over time (Fig. 10 (right)). Consequently, the spanning tree recomputed in every cycle also changes over time. This allows the network to achieve different topology if necessary. Note that for an edge to be considered as a valid communication link, it must be attainable during a planning cycle, given the dynamic motion constraints. For example, if two vehicles are moving away one from the other with high velocity, this might be a communication link that cannot be maintained due to the dynamic constraints, and has to be excluded from the spanning tree $T(t)$.

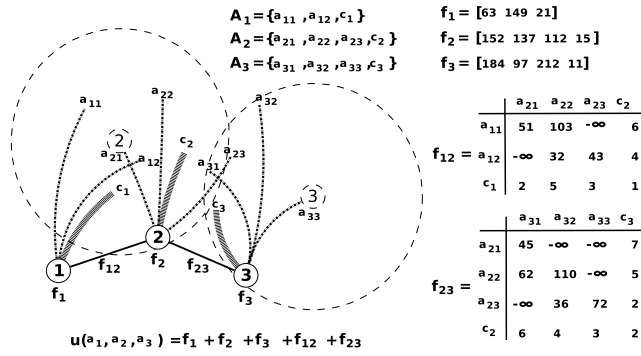


Fig. 7. A simple coordination graph, the action sets A_1, A_2, A_3 , the atomic and pairwise payoffs $f_1, f_2, f_3, f_{12}, f_{23}$ and the global utility function u

VI. SAFE DISTRIBUTED REPLANNING FOR VEHICULAR NETWORKS

As mentioned before, the priority-based methodology exhibits a major drawback. In the worst case and depending on the connectivity of the network, the vehicle with the lowest priority has to wait for every other vehicle to select a plan before taking its own decision. This implies that vehicles with low priorities often do not receive the choices of their higher priority neighbors within the duration of the planning cycle and they have to resort to contingency plans. The proposed safety conditions, however, do not depend on the prioritized scheme. They allow the implementation of message-passing protocols for distributed constraint optimization. The coordination problem can be modeled with coordination graphs [37] and can be solved with distributed message passing algorithms based on *belief propagation* [40], such as the *max-plus* algorithm [41]. In this section, we describe an approach that employs these abstractions.

In the formalization of coordination graphs, we assume we have n agents, and each agent i has to select an action a_i out of a finite action set A_i . Generally the goal is to find the optimal action vector $\bar{a}^* = (a_1^*, \dots, a_n^*)$ that maximizes a global utility function $u(\bar{a})$. The utility has a structure captured by a coordination graph $CG = (V, E)$. On every node of CG we define a function $f_i(a_i)$ called *atomic payoff*. An atomic payoff describes how well each action serves the goal of the agent corresponding to that node. On the edges e_{ij} of CG we define *pairwise payoff* functions $f_{ij}(a_i, a_j)$ that indicate how good for the team are pairs of actions of interacting agents (see Fig. 7 for an example). The global utility is assumed to depend only on the unary and pairwise payoff functions as follows:

$$u(\bar{a}) = \sum_i f_i(a_i) + \sum_{e_{ij} \in E(CG)} f_{ij}(a_i, a_j)$$

Max-plus is a distributed message passing algorithm that attempts to compute an optimal action vector using only local computations and communication for every agent. While the algorithm is running, each agent i chooses a neighboring agent j on CG , collects and adds all incoming messages from other

agents in its neighborhood, and sends a new message to j that is computed by the following formula:

$$m_{ij}(a_j) = \max_{a_i} \{f_i(a_i) + f_{ij}(a_i, a_j) + \sum_{k \in N(i), k \neq j} m_{ki}\} \quad (7)$$

At any time during the execution, the agents can compute a *marginal* function $g_i(a_i) = f_i(a_i) + \sum_{k \in N(i)} m_{ki}$. Maximizing g_i provides the best possible action a'_i for agent i with respect to messages from other agents. $u(a'_1, \dots, a'_n)$ is an approximation to the optimal u .

In order to adapt this formulation in our framework, each vehicle can be viewed as a node in the coordination graph CG . Two nodes share an edge in the graph if the corresponding vehicles can communicate or if they can potentially collide. The discrete set of actions of the max-plus algorithm corresponds to the set of candidate plans \mathcal{P}_i . The atomic payoffs $f_i(p_i)$, where $p_i \in \mathcal{P}_i$ can be computed by evaluating how close each trajectory takes vehicle V_i to its goal G_i . In the evaluation of the pairwise payoffs we must also express whether two trajectories are compatible or not:

$$f_i(p_i(dt), p_j(dt)) = -\infty \text{ if } \pi_i(x_i(t+dt), p_i(dt)) \neq \pi_j(x_j(t+dt), p_j(dt)) \quad (8)$$

where $p_i \in \mathcal{P}_i$ and $p_j \in \mathcal{P}_j$. If the two plans p_i and p_j are compatible then the pairwise payoff can be assigned a positive value that depends on other properties that have to be optimized. Consequently, it is necessary before the coordination step for the neighboring vehicles to exchange the sets of candidate plans so as to compute the pairwise payoff functions.

The pairwise payoffs can be computed in a distributed way that balances the burden among vehicles. Each vehicle computes one row for every pairwise payoff matrix that it is involved in, in a cyclic order. Then each vehicle transmits this row to its neighbors. In the computation of f_{ij} if $i \leq j$ then i computes rows r_1, r_2, \dots and j computes in reverse $r_{max}, r_{max-1}, \dots$, until the whole array is completed. In this way, a vehicle that has many neighbors is not overwhelmed and its computational overhead is outsourced to neighbors with a smaller number of payoff matrices to compute.

Message-Passing Algorithm

We describe here how the new message-passing approach can satisfy the safety conditions in a distributed way. The algorithm is provided in pseudo-code in Algorithm 3.

As in the priority-based algorithm, each vehicle V_i must be aware of the contingency plans of neighboring vehicles V_j at state $x(t+dt)$. This information is communicated at the beginning of each cycle between neighbors and the sampling-based, kinodynamic planner is invoked to construct the *Tree*. Next, the set of candidate plans \mathcal{P}_i is constructed from *Tree*. For each plan in this set, the corresponding contingency is attached to it and the unary payoff $f_i(p_i)$ is evaluated. Then, neighboring vehicles exchange their candidate plans and compute pairwise payoffs. Instead of using the simple form of compatibility as in Eq. 8, however, we must use Eq. 6,

Algorithm 3 SAFE AND DISTRIBUTED REPLANNING

Identify set of neighbors N_i

(Exchange contingencies)

for all $j \in N_i$ **do**Send contingency $\gamma_i(\infty)$ to V_j Receive contingency $\gamma_j(\infty)$ from V_j

(Planning: satisfies conditions 1,2)

 $Tree \leftarrow$ Retain valid subset of $Tree$ from previous cycle**while** ($time < PlanningBudget$) **do**

{

Select a state $x(t')$ on the existing $Tree$ Select valid plan $p(dt')$ Propagate trajectory $\pi(x(t), p(dt'))$ **if** ($(\pi(x(t), p(dt')), \gamma(\infty))$) is not **collision-free**) **then**Reject π **else****for all** $j \in N_i$ and while π not rejected **do****if** ($\pi(x(t), p(dt')) \neq \pi_j(x_j(t_n), \gamma_j(\infty))$) **then**Reject π

}

(Evaluate and exchange candidate plans)

 $\mathcal{P}_i \leftarrow$ plans of duration dt from $Tree$ **for all** $p_i \in \mathcal{P}_i$ **do**

{

Attach contingency plan $\gamma_i(\infty)$ to $p_i(dt)$ Evaluate unary payoff $f_i(p_i)$ for $p_i \in \mathcal{P}_i$ given G_i

}

for all $j \in N_i$ **do**

{

Send set \mathcal{P}_i to V_j Receive set \mathcal{P}_j from V_j

(Take Cond. 3 into account)

for all $p_i \in \mathcal{P}_i$ **do****for all** $p_j \in \mathcal{P}_j$ **do****if** ($\pi'_i(\pi_i(x_i(t_n), p_i(dt)), \gamma_i(\infty))$) \neq $\pi'_j(\pi_j(x_j(t_n), p_j(dt)), \gamma_j(\infty))$) **then**Compute payoff $f_{ij}(p_i, p_j)$ given the goals G_i, G_j **else** $f_{ij}(p_i, p_j) = -\infty$

}

(Coordination)

Enter into asynchronous message-passing to optimize:

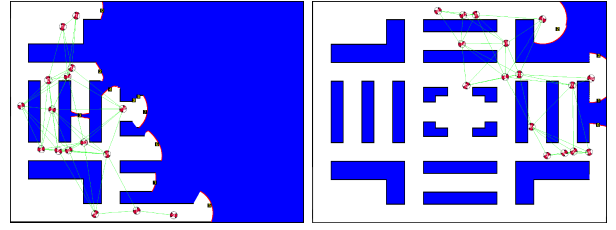
 $u(\bar{p}) = \sum_i f_i(p_i) + \sum_{e_{ij} \in E(CG)} f_{ij}(p_i, p_j)$ Stop protocol before time $t + dt$ Select plan \bar{p}_i that maximized $u(\bar{p})$ **for all** $j \in N_i$ **do****if** (\bar{p}_i incompatible with \bar{p}_j) **then**Select contingency γ_i as the next action

Fig. 8. Two snapshots of 16 vehicles exploring the labyrinth environment, while retaining a vehicular network.

which takes into account the concatenation with contingency plans. Given the definition of unary and pairwise payoffs the asynchronous message-passing protocol is initiated and the vehicles start exchanging messages. When the algorithm runs out of time, vehicle transmit to their neighbors their final action selections. Max-plus is incomplete, so if two neighboring vehicles have selected incompatible trajectories then one of the two vehicles switches to the contingency plan. This is a very fast adjustment step, which guarantees that the third condition is always satisfied. In the experiments section we show that max-plus has to resort to the contingency plan less often than priority-based schemes.

The secondary goal is to find among the safe solutions one that maximizes the global utility u , within the allocated amount of time. Because the algorithm does not monotonically increase the global utility, it must periodically compute u and keep track of the action vector that produced the maximum value. However, no single agent has all the available information to compute u . In order to achieve an efficient, distributed computation of the utility we use a minimum spanning tree of CG . We use again distributed protocols for computing minimum spanning trees on graphs using local information such as the distance between agents [46]. Given the minimum spanning tree structure, an arbitrary vehicle acting as a root of the tree initiates the process:

Down pass The root transmits a signal to compute u . Each node passes the signal down the tree.

Up pass Each node i :

- 1) Collects partial payoff values and actions from children.
- 2) Maximizes marginal g_i and chooses best action a_i .
- 3) Adds its contribution to the global payoff u .
- 4) Sends new partial payoff and actions to its parent.

Down pass The root adds up all partial payoffs so as to compute and maximize u . The optimal value of u^* and actions \bar{a}^* are transmitted down the tree.

This computation is fast since the utility computation messages can be interleaved with normal max-plus messages [41].

VII. EXPERIMENTAL RESULTS

Setup: We tested our algorithm on a distributed simulator that we developed and ran on an XD1 Cray cluster. The planner for each vehicle is running on a different processor and operates under time limitations imposed by a server that simulates ground truth. All data exchange is done via simple send and receive messages using sockets.

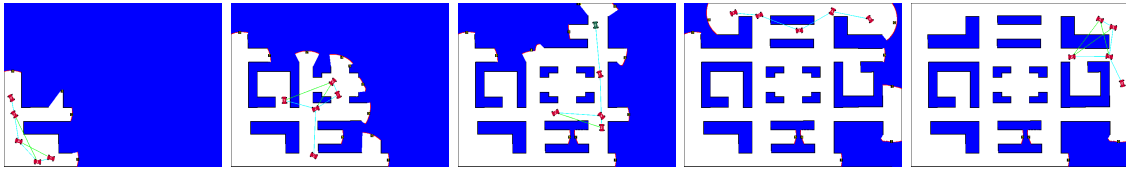


Fig. 9. Snapshots from an experiment in scene “labyrinth” with 5 vehicles, communication range at 25% of the scene width and sensing range at 15%.

Nr Vehicles	Cond1		Cond1 & Cond2		Cond1 & Cond3		All Conditions	
	1 st failure (sec)	success %	1 st failure (sec)	success %	1 st failure(sec)	success %	1 st failure(sec)	success %
2	287.10	10%	293.25	37.37%	113.10	0%	N/A	100%
4	21.00	0%	141.07	12.00%	21.53	0%	N/A	100%
8	3.67	0%	24.16	0%	4.31	0%	N/A	100%
16	3.00	0%	23.10	0%	3.00	0%	N/A	100%

TABLE I
PROBABILITY THAT NETWORKS OF CAR-LIKE VEHICLES SUCCEEDED TO EXPLORE WHEN DIFFERENT SAFETY CONDITIONS ARE MET.

The simulated vehicular networks have been tested in three different environments. *Rooms* and *Random* are seen in Fig. 10. The first represents a structured environment with rooms and corridors, while the second is an unstructured environment. *Labyrinth* (Fig. 8) is a difficult scene that contains multiple narrow passages. Two types of vehicles have been tested, car-like robots, for which the dynamic equations are shown in Fig. 10, and differential drive robots with bounded acceleration. The car-like robots obey velocity bounds: $|V| \leq 3.5m/s$, and acceleration bounds: $\alpha \leq 0.8m^2/s$ as well as steering bounds: $|s| \leq 1deg/m$, $|t| \leq 4deg/s$. Vehicles have limited sensing and communication ranges. For contingencies, deceleration maneuvers were used. The framework allows for plugging in other types of dynamics and contingencies.

We present here results from an application that combines many of the constraints we are interested in testing our algorithm. The vehicles have to solve a coordinated exploration task while retaining a network and avoiding collisions. They are initially located at the bottom left corner of the environment, close one to another, but at collision-free states forming a network with a single component. During each replanning cycle a simulated model builder and a task planner transmit to the vehicles the updated map and set of goals. The goals correspond to frontiers of the unexplored space and are assigned greedily so that large frontiers which are close to vehicles are being considered first. Experiments with up to 32 vehicles have been conducted.

We compare the max-plus algorithm against a simpler prioritized scheme described in more detail in [34]. In that scheme, the vehicles have unique global priorities. The planning step is the same as here. For the plan selection, each vehicle

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{s} \end{pmatrix} = \begin{pmatrix} \cos \theta \cdot \cos s \cdot V \\ \sin \theta \cdot \cos s \cdot V \\ \sin s \cdot v \\ \alpha \\ t \end{pmatrix}$$



ROOMS



RANDOM

Fig. 10. The state update equations for the car-like vehicles and scenes “rooms” and “random”.

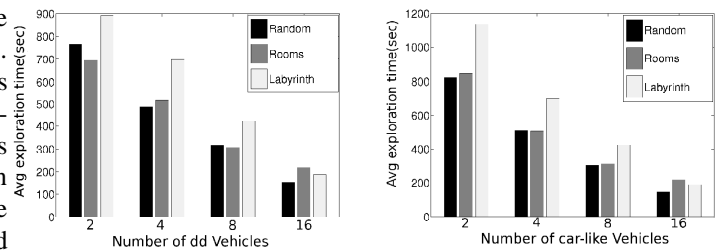


Fig. 12. Scalability results for three scenes: Random, Rooms, Labyrinth. Left: DD robots, Right: Car-Like robot

receives the choices of higher priority vehicles and then tries to choose its own plan so that it is compatible the higher priority neighbors. If no such plan exists, the vehicle chooses the contingency plan. At last the vehicle transmit its selection to its lower priority neighbors.

Feasibility: Table I exhibits the importance of the safety conditions in decoupled replanning. We measure the time (in seconds), that the vehicles can move without colliding with each other when Cond. 2 and/or 3 (those necessary for safe multi-vehicle planning) are relaxed. The numbers reported show the time at which the first collision or loss of network connectivity occurs. The problem is so constrained for multiple vehicles, that often collisions cannot be avoided already since the 2nd replanning loop. The results are averaged out of 10 runs and are shown in columns labeled *failure*. If either one of the two conditions is absent, the vehicles will collide. When all conditions are enabled, then as expected, there is no failure. The columns labeled *success*, measure the percentage of successful exploration of the whole space without collisions. As we see, for small teams of 2 or 4 vehicles, there were some cases where the vehicles completed the task without one or both of the conditions. This is to be expected since the chances of an encounter are lower for such small teams.

Contingency Plans: One important advantage of the max-plus algorithm is that it avoids the use of priorities in coordination. In priority-based schemes, lower priority vehicles are overly constrained by the choices of higher priority agents. This may lead to frequent selection of contingency plans. We have

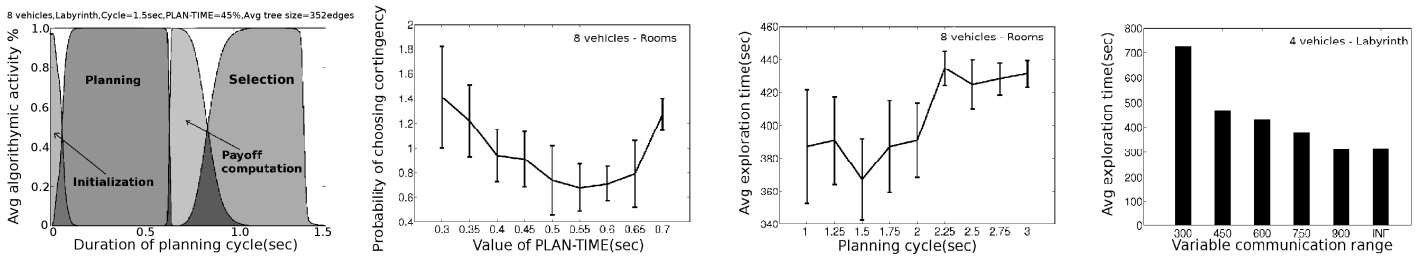


Fig. 11. Average activity profile during a cycle (left) and dependence on (from second to forth): CYCLE_DURATION, PLAN_TIME, and maximum communication range.

experimented in scenes *Labyrinth* and *Rooms* for networks with 16 and 32 vehicles. Table II presents the number of times contingencies were selected using the simple prioritized scheme and max-plus. For 32 vehicles, max-plus chooses contingency plans considerably fewer times. Additionally, the results from the prioritized scheme have higher variation between different scenes and team sizes, while max-plus is much more consistent.

Scalability: The scalability properties of the algorithm are presented in Fig. 12, which provides the average running times (10 runs per case) to complete exploration in the three scenes for car-like and DD vehicles. Increasing teams size from 2 to 16 results in 5 to 6 times faster exploration. This is a very encouraging result given that the simulated systems are very constrained, both due to network and kinodynamic constraints. Moreover, there is no significant variation in the performance of the algorithm when applied to systems with different dynamics.

Performance and Parameter Dependence: Fig. 11(left) shows the average activity profile of a vehicle during each cycle. The algorithm utilizes most of the replanning cycle in useful computations but the payoff computation takes up a non-trivial amount of time. In the selection step, max-plus does not let the processor idle, and in most cases is able to find an optimal or near optimal solution. The latter is confirmed by the second figure where we see the probability of executing contingency plans as a function of the portion of the planning cycle allocated to the motion planner (PLAN-TIME). This probability is very low (0.7 – 1.5%). Moreover, there is an optimum value at around 55%. For small PLAN-TIME, the planner has little time to produce enough plans, while for larger ones max-plus has not enough time to make a selection and the contingency is selected for safety reasons. The third figure shows that increasing the duration of the planning cycle can result in performance deterioration. The last figure shows

that as the communication range increases, four vehicles finish the exploration faster, which is expected.

VIII. CONCLUSIONS

This paper presents a novel integration of sampling-based planners with message-passing protocols for the distributed solution of planning problems that involve vehicles with dynamic constraints. It extends work on safe, real-time sampling-based planning to the case of multiple communicating vehicles. The method provides safety guarantees in terms of collision avoidance as well as in terms of retaining a connected communication network. The algorithm has been implemented on a distributed simulator and the results on vehicles with acceleration constraints confirm the safety properties of the approach in a workspace exploration application. A comparison over priority-based schemes shows that the distributed protocol offers improved scalability.

The proposed method allows for plugging in other types of dynamic constraints and can also be integrated with higher-level approaches for distributed task assignment and distributed state estimation. Two important directions for improving the current framework is to study the effects of sensing uncertainty and limited communication reliability. We aim towards addressing these issues and possible extensions in future work.

IX. ACKNOWLEDGMENTS

Work on this paper has been supported in part by NSF 0308237, 0615328 and 0713623. The computational experiments were run on equipment obtained by CNS 0454333, and CNS 0421109 in partnership with Rice University, AMD and Cray. The authors would also like to thank the anonymous reviewers and the organizing committee of ROBOCOMM 2007 for their comments and their invitation to MONET.

REFERENCES

- [1] X. Yang, L. Liu, N. H. Vaidya, and F. Zhao, "A vehicle-to-vehicle communication protocol for cooperative collision warning," in *MOBIQUITOUS-04*, 22-26 Aug. 2004.
- [2] D. S. R. C. D. Home, "http://www.leearmstrong.com/dsrc/dsrchomeset.htm."
- [3] C. Clarc, S. Rock, and J.-C. Latombe, "Dynamic networks for motion planning in multi-robot space systems," in *Intl. Symp. of Artificial Intelligence, Robotics and Automation in Space*, 2003.
- [4] M. C. Clark, T. Bretl, and S. Rock, "Applying kinodynamic randomized motion planning with a dynamic priority system to multi-robot space systems," in *Aerospace Conference*, 2002.

	Rooms		Labyrinth	
	16	32	16	32
Prioritized	3.61 %	24.5 %	1.35 %	8.42 %
Max-plus	0.98 %	2.26 %	3.04 %	4.84 %

TABLE II
AVERAGE PERCENTAGE OF CYCLES THAT V_i EXECUTES CONTINGENCY.

- [5] R. M. Murray, "Recent reseach in cooperative control of multi-vehicle systems," (submitted) *ASME Journal of Dynamic Systems, Measurement, and Control*, 2007.
- [6] P. Ogren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in distributed environments," *IEEE Tr. on Aut. Control*, vol. 49, no. 8, pp. 1292–1302, 2004.
- [7] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Tr. on Aut. Control*, vol. 51, no. 3, pp. 401–420, 2006.
- [8] G. A. S. Pereira, A. K. Das, V. Kumar, and M. F. M. Campos, "Decentralized motion planning for multiple robots subject to sensing and communication constraints," in *Work. on Multi-Robot Systems*, 2003.
- [9] H. G. Tanner, G. J. Pappas, and V. Kumar, "Leader-to-formation stability," *IEEE TRA*, vol. 20, no. 3, June 2004.
- [10] M. Egerstedt, X. Hu, and A. Stotsky, "Control of mobile platforms using a virtual vehicle approach," *IEEE Transactions on Automated Control*, vol. 46, no. 4, pp. 1777–1782, November 2001.
- [11] L. Pallotino, V. G. Scordio, E. Frazzoli, and A. Bicchi, "Decentralized and scalable conflict resolution strategy for multi-agent systems," in *Int. Symp. on Mathematical Theory of Networks and Systems*, 2006.
- [12] D. V. Dimarogonas, K. J. Kyriakopoulos, and D. Theodorakatos, "Totally distributed motion control of sphere world multi-agent systems using decentralized navigation functions," *ICRA*, 2006.
- [13] S. Loizou, D. Dimarogonas, and K. Kyriakopoulos, "Decentralized feedback stabilization of multiple nonholonomic agents," in *ICRA*, vol. 3, 2004, pp. 3012–3017.
- [14] S. LaValle, *Planning Algorithms*. Cambridge university Press, 2006.
- [15] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE TRA*, vol. 12, no. 4, pp. 566–580, Aug. 1996.
- [16] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Boston: MIT Press, 2005.
- [17] B. R. Donald, P. G. Xavier, J. F. Canny, and J. H. Reif, "Kinodynamic motion planning," *Journal of ACM*, vol. 40, no. 5, pp. 1048–1066, 1993.
- [18] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *IJRR*, vol. 20, no. 5, pp. 378–400, May 2001.
- [19] A. M. Ladd and L. E. Kavraki, "Motion planning in the presence of drift, underactuation and discrete system changes," in *RSS I*, Cambridge, MA, June 2005.
- [20] K. E. Bekris and L. E. Kavraki, "Greedy but safe replanning under kinodynamic constraints," in *ICRA*, Rome, Italy, April 2007.
- [21] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *IJRR*, vol. 21, no. 3, pp. 233–255, 2002.
- [22] J. Bruce and M. Veloso, "Safe multi-robot navigation within dynamic constraints," *Proc. of the IEEE*, 2006.
- [23] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with rrts," in *ICRA*, 2006.
- [24] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite rrts for rapid replanning in dynamic environments," in *IEEE Int. Conf. on Robotics and Automation, ICRA-07*, 2007.
- [25] R. Gayle, K. R. Klingner, and P. G. Xavier, "Lazy reconfiguration forest: An approach for planning with multiple tasks in dynamic environments," in *ICRA*, Rome, April 10-14 2007, pp. 1316–1323.
- [26] J. v. d. Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *ICRA*, May 2006.
- [27] M. Kallman and M. Mataric, "Motion planning using dynamic roadmaps," in *ICRA-04*, vol. 5, 2004.
- [28] T. Fraichard and H. Asama, "Inevitable collision states - a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.
- [29] S. Petti and T. Fraichard, "Partial motion planning framework for reactive planning within dynamic environments," in *ICRA*, Barcelona, Spain, September 2005.
- [30] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *AIAA Journal of Guidance and Control*, vol. 25, no. 1, pp. 116–129, 2002.
- [31] G. Sanchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *ISRR*, 2003, pp. 404–417.
- [32] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning for teams of mobile robots," *Robotics and Autonomous Systems*, vol. 41, no. 2, pp. 89–99, 2002.
- [33] M. Saha and P. Ito, "Multi-robot motion planning by incremental coordination," in *IROS*, 2006.
- [34] K. E. Bekris, K. I. Tsianos, and L. E. Kavraki, "A decentralized planner that guarantees the safety of communicating vehicles with complex dynamics that replan online," in *IROS (submitted)*, 2007.
- [35] M. Yokoo and K. Hirayama, "Algorithms for distributed constraint satisfaction: A review," *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 2, pp. 189–212, 2000.
- [36] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, "An asynchronous complete method for distributed constraint optimization," *Artificial Intelligence Journal*, vol. 161, no. 1-2, pp. 149–180, 2005.
- [37] C. Guestrin, D. Koller, and R. Parr, "Multiagent planning with factored mdps," in *NIPS-14*. MIT Press, 2002.
- [38] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multi-robot coordination," *IEEE TRA*, vol. 18, no. 5, pp. 758–786, Oct 2002.
- [39] M. B. Diass, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: a survey and analysis," *Proc. of the IEEE*, vol. 94, no. 7, pp. 1257–1270, July 2006.
- [40] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
- [41] J. R. Kok and N. Vlassis, "Collaborative multiagent reinforcement learning by payoff propagation," *Journal of Machine Learning Research*, vol. 7, pp. 1789–1828, 2006.
- [42] K. Plarre and P. R. Kumar, "Extended message passing algorithm for inference in loopy gaussian graphical models," *Ad Hoc Networks*, vol. 2, pp. 153–169, 2004.
- [43] K. E. Bekris, K. I. Tsianos, and L. E. Kavraki, "A distributed protocol for safe real-time planning of communicating vehicles with second-order dynamics," in *ROBOCOMM*, 2007.
- [44] A. M. Ladd and L. E. Kavraki, "Fast tree-based exploration of state space for robots with dynamics," in *WAFR*, 2005, pp. 297–312.
- [45] M. A. Hsieh, V. Kumar, and C. J. Taylor, "Constructing radio signal strength maps with multiple robots," in *IEEE Inter. Conference on Robotics and Automation*, vol. 4, New Orleans, LA, April 2004, pp. 4184–4189.
- [46] G. Singh and A. J. Bernstein, "A highly asynchronous minimum spanning tree protocol," *Distributed Computing*, vol. 8, no. 3, pp. 151–161, March 1995.
- [47] P. Radia, "An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN," *ACM SIGCOMM Computer Communication Review*, vol. 15, no. 4, pp. 44–53, 1985.