Priority-based coordination of robots

Jean Gregoire, Silvère Bonnabel, Arnaud de La Fortelle

Priority-based coordination of robots

Jean Gregoire* Silvère Bonnabel* Arnaud de La Fortelle*†

June 4, 2013

Abstract

This paper addresses the problem of coordinating multiple robots in a common environment with positive velocity along fixed paths. We propose a priority-based framework, which is an overlay of the coordination space approach that fits well the positive velocity constraint. The theoretical developments rely on a priority graph that defines the relative order of the robots. As a primary contribution, the priority graph is proved to uniquely encode the homotopy classes of feasible trajectories, and a mathematical characterization of priority cycles leading to deadlock configurations is provided. As a secondary contribution, the framework is applied to the problem of designing an intersection management system. This system uses heuristics rules to assign priorities between vehicles, and once the the priorities are fixed the algorithm is proved to possess optimality properties. Good performance is observed in numerical experiments on an open network of intersections. More generally, the framework seems to open up new avenues of research in the field of coordination.

1 Introduction

When a fleet of robots move in a common environment, coordination must be carried out to avoid collisions. Safety is indeed the main reason why automated coordination has become an intensive field of research for applications in air traffic systems (see, e.g. [1]), railway systems (see, e.g. [2]) or autonomous intersection management (see, e.g. [3, 4]). In this paper, we consider the problem of coordinating a collection of robots, motivated by applications such as coordinating a fleet of automated guided vehicles (AGVs) in a factory, or autonomous vehicles in a fully automated transportation system.

Multiple robot coordination with unconstrained paths is a problem of high combinatorial complexity [5]. In [6], a path-velocity decomposition allowing to reduce the problem's complexity was first proposed to the authors' knowledge. In this setting, each robot is assumed to move along a predefined path and then the velocity profiles of the robots along their assigned paths are optimized. The problem is thus decomposed into a trajectory tracking problem via a low-level controler, and a high-level planning problem, the latter leading to the realm of fixed-path coordination algorithms (see, e.g. [7]).

*Mines ParisTech, Centre de Robotique, Mathématiques et Systèmes, 60 Bd St Michel 75272 Paris Cedex 06, France

†Inria Paris - Rocquencourt, IMARA team, Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay, France

The fixed-path approach has since become standard in motion planning [8, 9, 10, 11, 12]. An attempt to take into account dynamics constraints within this framework has been made in, e.g. [13, 14, 15], and [16] for distributed algorithms. The configuration of each robot boils down to its curvilinear position on its path and the configuration space of the whole system is called the coordination space. It is a n-dimensional space where n denotes the number of robots going through the intersection. To prevent collisions between robots, some configurations of the coordination space must be excluded: they constitute the so-called obstacle region. Such approaches based on the configuration space turn the motion planning problem into the geometric problem of searching a collision-free trajectory in a n-dimensional space that minimizes a certain objective function (e.g. the average travel time from the initial to the goal configuration).

In the coordination problem, the obstacle region has a cylindrical shape [7, 10], as first noticed in [9]. The papers [17, 18] study the problem of finding pareto-optimal trajectories (i.e. each robot tries to optimize its own particular objective function). They propose to first discretize the coordination space, and then to take advantage of the cylindical structure to turn the coordination space into a negatively curved discrete space. Uniqueness of locally Pareto-optimal trajectories in each class of homotopic trajectories appears then as a mere consequence of the uniqueness of geodesics linking two points in a hyperbolic space. However, enumerating all locally optima in each homotopy class to find a globally optimal trajectory is a problem of high combinatorial complexity, and the authors point out the solution proposed is of interest only with a few robots and a low degree of intersection.

In this paper, building upon (but going far beyond) our preliminary conference paper [19], we introduce an alternative approach to the coordination problem. We consider the problem of coordinating robots on fixed paths, and we impose nonnegative velocities along each path. This is a standard assumption (see, e.g. [17]) as an efficient planning algorithm is expected to slow down or in the worst case stop robots, but not have them move backwards. Such constraints are well-known to possibly lead to the so-called deadlock situations, in which vehicles "block" each other. To address the challenge of building efficient collision-free and deadlock-free trajectories, the present paper advocates the use of a priority-based framework. The usual notion of priority is mathematically captured through a priority graph.

Beyond introducing an intuitive and versatile framework for coordination of robots, the paper has four main contributions

1. The priority graph is proved to uniquely encode the homotopy classes of the coordination space described in [17, 18]. Thus, the priority graph captures the discrete combinatorial part of the problem, and allows to design efficient algorithms based on intuitive heuristic priority assignment policies.

2. The priority graph is a very suitable tool to understand the deadlock phenomenon when robots block each other. We prove that deadlocks can be prevented by avoiding certain cycles in the priority graph that we characterize mathematically. Indeed, those cycles would require robots to wait indefinitely for each other in a cyclic fashion because of cyclic priorities.

3. For a given priority graph, we solve the optimization problem in continuous time of finding a trajectory within the corresponding homotopy class that minimizes the average time spent by the robots in the intersection.

4. The proposed theoretical tools are applied to the particular problem of managing an open network of intersections. The theory is completed by introducing the intersection graph that captures links between obstacle regions. Based on the several tools at hand, a collision and deadlock-free simple algorithm running in real-time is proposed and shown to yield very satisfactory results in simulations performed on synthetic data.

The paper is organized as follows. Section 2 is mainly expository and presents the coordination space approach. Section 3 introduces our priority-based framework. Section 4 is devoted to the problem of optimizing the average travel time once priorities have been assigned. Section 5 applies the proposed tools to an intersection management problem, and the results are illustrated by numerical experiments. Finally, Section 6 is devoted to a discussion, in which a series of remarks indicate that the proposed versatile framework seems to open up new avenues in the field of coordination in robotics.

2 The coordination space approach

2.1 The fixed paths assumption

This section is mainly expository and presents the fixed-path coordination space approach (see, e.g. [7]). This approach dates back to the 1980s (see, e.g. [6]), and has since become standard (see, e.g. [20]), especially in applications to automated intersections where the environement is highly constrained and the vehicles tend to move along very similar paths in normal driving conditions.

Assumption 1 (Fixed paths)**.** *Every robot i follows a particular path γ_i and we let $x_i \in \mathbb{R}$ denote its curvilinear coordinate along the path. The state $x = (x_1, \cdots, x_n)$ indicates the configuration of all robots, and $x(t)$ denotes the evolution of the state x over time $t \in [0, T]$.*

The approach is illustrated in Figure 1. The curvilinear coordinates are normalized, so that $x \in \chi = [0, 1]^n$ where n denotes the number of robots going through the intersection (possibly varying with time). The boundedness condition on χ is rather technical but ensures that the whole intersection lies in a bounded region (somehow interactions are limited to a bounded area). The configuration space χ is known as the *coordination space* [21]. In the rest of the paper, $\{\mathbf{e}_i\}_{1 \leq i \leq n}$ denotes the canonical basis of χ.

As every robot occupies a non-empty geometric region, some states must be excluded to avoid collisions between robots.

Definition 1 (Obstacle region, Obstacle-free region)**.** *The obstacle region χ_{obs} is the open set of all collision configurations. $\chi_{\text{free}} = \chi \backslash \chi_{\text{obs}}$ denotes the obstacle-free space.*

A collision occurs when two robots occupy a same region of space, so that [7]:

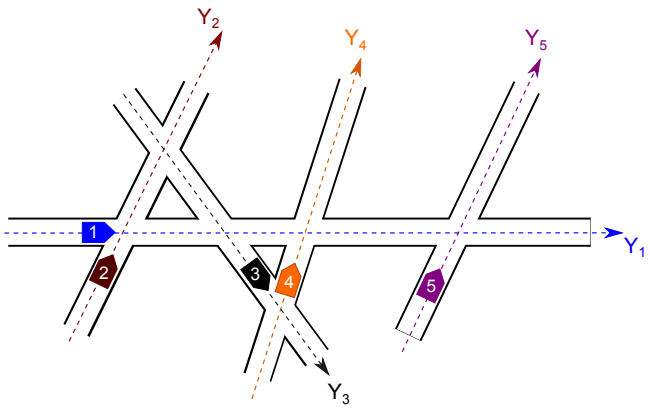

Figure 1: The fixed paths assumption. Every robot travels along an assigned path.

Property 1 (Cylindrical structure)**.** *The obstacle region can be described as the union of* $n(n-1)/2$ *open cylinders* χ^{ij}_{obs} *corresponding to as many collision pairs:* $\chi_{\text{obs}} = \cup_{i>j}\chi^{ij}_{\text{obs}}$.

Note that $\chi^{ij}_{\text{obs}} = \chi^{ji}_{\text{obs}}$ and $\chi^{ii}_{\text{obs}} = \emptyset$. Figure 2 displays the obstacle region and a collision configuration for a 2-path intersection.

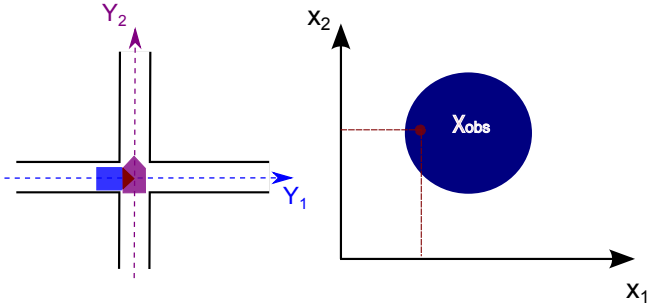

Figure 2: The left drawing depicts two paths with two robots in collision in the current configuration. The left drawing shows the obstacle region associated to the two paths in the coordination space and the collision configuration $(x_1, x_2) \in \chi_{\text{obs}}$ corresponding to the collision of the left drawing.

Assumption 2 (Cylinders convexity and regularity)**.** *Every cylinder* χ^{ij}_{obs} *has an open bounded convex cross-section (in the plane generated by* $\mathbf{e}_i$ *and* $\mathbf{e}_j$*). Moreover, the boundary of every cylinder* χ^{ij}_{obs} *is supposed to be continuous and piecewise smooth.*

This assumption is rather technical and implies that cylinders cross-sections are simply connected and excludes some cases of real intersections such as a pair of opposite turn-left. The proofs in the sequel rely on this assumption, but the results could be extended with a more relaxed assumption, leading to additional technicalities in the proofs that may complicate the paper in an undesirable way.

Cross-sections are open sets so that the complementary set is closed and hence complete. It also ensures that all cross-sections are included in the interior of $[0,1]^2$: no collision can occur for a robot at coordinates 0 or 1.

2.2 Problem formulation

The initial configuration of the robots is $x^{\text{init}} \in \chi_{\text{free}}$, and the goal region is $\chi_{\text{goal}} = \{\mathbf{1} = (1 \cdots 1)\} \subset \chi_{\text{free}}$. Moreover, robots are assumed to move with a bounded velocity in the intersection: there is a maximum velocity due to physical limitations.

Assumption 3 (Bounded velocity)**.**

$$\forall i \in \{1...n\}, \forall t \in [0,T], |x_i'(t)| \leq v_i^{\max}$$

The coordination space approach enables to formulate very synthetically what is a feasible trajectory for the coordination problem:

Definition 2 (Feasible trajectory)**.** *A feasible trajectory for the considered problem is a right-differentiable trajectory* $x : [0,T] \to \chi_{\text{free}}$ *such that* $x(0) = x^{\text{init}}$, $x(T) \in \chi_{\text{goal}}$, *and* $\forall t \in [0,T], |x_i'(t)| \leq v_i^{\max}$.

To define a performance criterion for the motion planning problem, an objective function $c(x)$ must be introduced to compare feasible trajectories. The corresponding notion of optimality one seeks depends on the application. A standard choice for the objective function, that we will consider in the sequel, is the average time elapsed in the intersection (see, e.g. [16]). When the objective function is scalar, the optimality problem consists of finding a feasible trajectory $x^\star$ that minimizes the cost, i.e. for any feasible trajectory x, we have $c(x^*) \leq c(x)$. However, when it is vectorial, as considered in [17], the notion of optimality must be replaced with the so-called Pareto optimality.

In [17], it has been proved that the set of all feasible trajectories with fixed endpoints could be classified into homotopy classes (one element of the class is deformable to the other) in a cylindrical coordination space, and Pareto optimal trajectories are in bijection with homotopy classes. In the latter paper no semantics is proposed to describe the different homotopy classes. Our priority-based framework introduced in the next section provides such a semantics by encoding the homotopy classes thanks to the intuitive concept of priority.

3 The priority-based framework

The planned trajectory is expected to have a positive velocity for all robots through time. Indeed, it seems natural to slow down or possibly stop robots at some times to avoid collisions. In particular, for an application to intersection management, a planned trajectory with non positive velocities would produce vehicles that move backwards in the intersection area, leading to an unsafe (and likely to be inefficient) situation.

Assumption 4 (Positive velocity)**.**

$$\forall i \in \{1...n\}, \forall t \in [0,T],\ x_i'(t) \geq 0$$

Because of the positive velocity constraint, for every couple of robots with a non-empty collision region, one of the two robots necessarily passes before or after the other one. In the coordination space, the trajectory passes below or above the collision cylinder as depicted in the top drawings of Figure 3. This reflects the intuitive notion of priority, that has already been used for coordination. In this paper, we propose a novel framework based on this intuitive notion of priorities: it is an overlay of the coordination space approach that suits well the positive velocity constraint. Notably, the framework allows to underline the great impact of priority choices on the structure of the set of feasible trajectories.

3.1 The priority graph

Definition 3 (Cylinder closure)**.** *Let* $\chi_{\text{obs}}^{i\succ j}$ *denote the set defined by*

$$\chi_{\text{obs}}^{i\succ j} = \chi_{\text{obs}}^{ij} - \mathbb{R}_+\mathbf{e}_i + \mathbb{R}_+\mathbf{e}_j$$

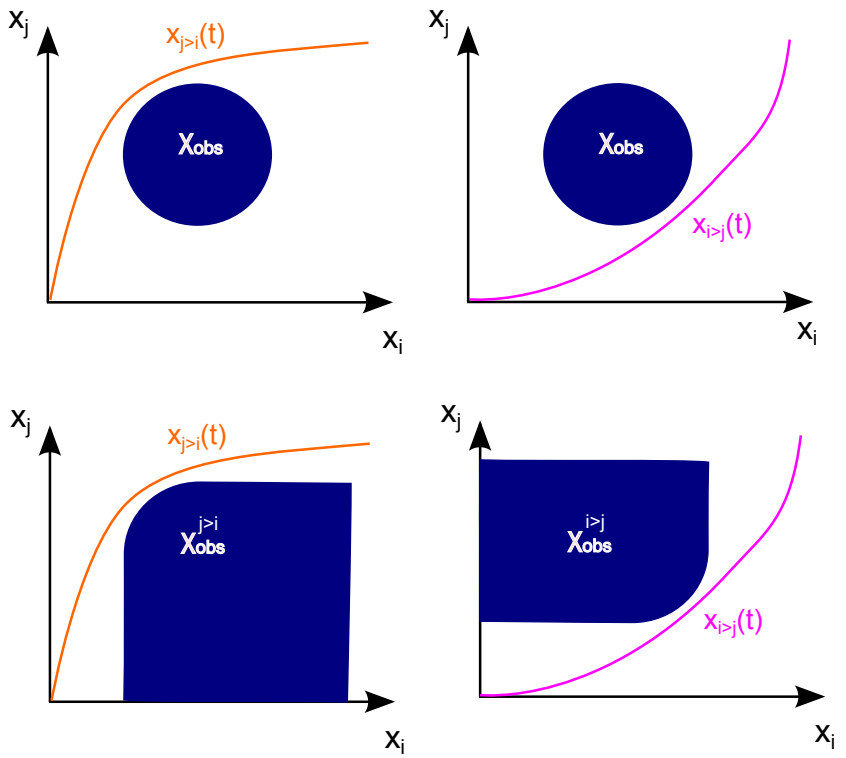

Figure 3: The top drawings represent in the plane (x_i, x_j) the obstacle region χ_{obs}, a feasible trajectory $x_{i\succ j}$ respecting priority $i \succ j$, and a feasible trajectory $x_{j\succ i}$ respecting priority $j \succ i$. The bottom drawings depict $\chi_{\text{obs}}^{i\succ j}$ and $\chi_{\text{obs}}^{j\succ i}$.

The bottom drawings of Figure 3 display $\chi_{\text{obs}}^{i\succ j}$ and $\chi_{\text{obs}}^{j\succ i}$. Thanks to the convexity hypothesis of the cylinders of χ_{obs}, we can assert that any feasible trajectory will be necessarily collision-free with respect to $\chi_{\text{obs}}^{i\succ j}$ or $\chi_{\text{obs}}^{j\succ i}$ exclusively, assuming $\chi_{\text{obs}}^{ij} \neq \emptyset$. The geometry of the coordination space thus leads us to define a natural binary relation corresponding to priority relations between

robots: a very familiar and intuitive notion in real life. We say the robot i has priority over the robot j if the associated path is collision-free with respect to $\chi_{\text{obs}}^{i\succ j}$ in the coordination space.

Definition 4 (Priority relation)**.** *A feasible trajectory x induces a binary relation $\succ$ on the set $\{1...n\}$ as follows. For $i \neq j$ s.t. $\chi_{\text{obs}}^{ij} \neq \emptyset$, $i \succ j$ if x is collision-free with $\chi_{\text{obs}}^{i\succ j}$.*

Figure 4 provides two graphical representations of priorities and highlights on a simple example that the priority relation in not necessarily an order. Note that, whereas priorities are commonly a time related concept (as introduced in [22] for example), we opt here for a topological definition of priorities in the coordination space, taking advantage of the cylindrical structure of the obstacle region.

We propose to encode the priority relations by a graph. As for any collision pair i, j we have either $i \succ j$ or $j \succ i$, the priority relation can be defined by an oriented graph G whose vertices are $\{1...n\}$.

Definition 5 (Priority graph)**.** *Given a feasible trajectory x, we call the priority graph the oriented graph G whose vertices are $\{1...n\}$ and such that $i \xrightarrow{G} j$ if $i \succ j$.*

Note that, for all $i \neq j$ there is only one arc linking i and j if $\chi_{\text{obs}}^{ij} \neq \emptyset$ or zero else. There is no arc linking i to itself. Generally speaking, there are thus potentially $2^{\frac{n(n-1)}{2}}$ possible priority graphs.

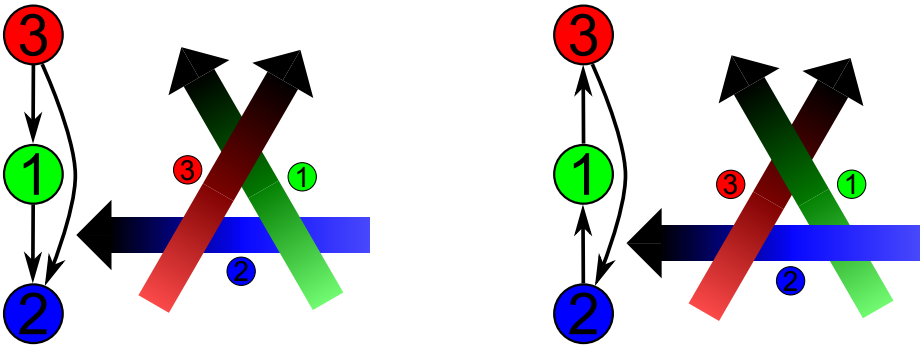

Figure 4: Two representations of priority relations. In each drawing, the relation is represented in two ways: as a complete oriented graph, where orientation yields the priority; and as trajectories over time, foreground being first, background later. The left drawing represents a relation that is an order (even a total order). The right drawing shows a relation that is *not* an order.

Note also that since we necessarily have $i \succ j$ or $j \succ i$ (exclusively), the trajectory does not pass through the intersection of regions $\chi_{\text{obs}}^{i\succ j}$ and $\chi_{\text{obs}}^{j\succ i}$. As a consequence all configurations in $\chi_{\text{obs}}^{i\succ j} \cap \chi_{\text{obs}}^{j\succ i} \supseteq \chi_{\text{obs}}$ are excluded. This operation corresponds to the South-West closure of the obstacle region proposed in [21]. Whereas this completion is sufficient to avoid deadlocks involving two robots, deadlock avoidance with more than two robots is a much more tedious task, as will be shown in the sequel.

An important result associated with the proposed framework is that the priority graph provides a semantics to describe the homotopy classes of feasible trajectories, as the priority graph is in bijective correspondance with homotopy classes. Mathematically, it can be thus stated that the priority graph is an invariant of every class.

Theorem 1 (Homotopy classes encoding)**.** *The homotopy classes of feasible trajectories are uniquely encoded by the priority graph.*

Proof. First we will prove that two homotopic feasible trajectories (thus having the same endpoints) necessarily have the same graph. Indeed, suppose two homotopic feasible trajectories x^1 and x^2 in χ_{free} have two distinct priority graphs. Say that $i \succ j$ for x^1 and $j \succ i$ for x^2. It means that in the plane (x_i, x_j), the trajectory x^1 is below χ_{obs}^{ij}, whereas the trajectory x^2 is above χ_{obs}^{ij}. As $\chi_{\text{obs}}^{ij} \neq \emptyset$ and $\forall i \in \{1...n\},\ x_i' \geq 0$, any continuous transformation transforming x^1 into x^2 will inevitably collide χ_{obs}^{ij} (by the intermediate value theorem). As a consequence, x^1 and x^2 are not homotopic in χ_{free}, which yields a contradiction.

To prove uniqueness, consider now a given graph G associated to a feasible trajectory. We are going to show that all trajectories respecting G are homotopic to a special trajectory. Homotopy defining an equivalence relation, this will suffice to conclude. Introduce the trajectory $x^\star$ respecting the priority graph G defined in Theorem 3 below (essentially, this trajectory consists of moving forward the robots at maximum velocity until meeting the boundary of the closured collision regions and then moving along it until maximum velocity is again feasible). The result of Theorem 3 is twofold: 1- the trajectory x^* is feasible, and 2- any component of any trajectory respecting the priority graph G and having the same endpoints as x^*, cannot overtake the same component of $x^\star$. The idea of the proof is that any feasible trajectory is somehow a delayed version of $x^\star$. Suppose x is a feasible trajectory with priority graph G. We will show it is deformable into x^* by considering the auxiliary trajectory x^α defined as follows:

$$\forall i \in \{1...n\}, x_i^\alpha(t) = \min\left(x_i(t+\alpha), x_i^\star(t)\right)$$

The trajectory clearly respects the initial condition as $x_i^\alpha(0) = x_i(0) = x_i^\star(0)$ as well as the kinetic constraints, reaches the goal region and is collision-free for all $\alpha > 0$. Moreover, $x_i^0(t) = x_i(t)$. As the goal is reached in finite time, there exists a time $T > 0$ for which $x_i(T) = 1$ for all i. Thus, there exists $\alpha_0 \leq T$ such that $x_i^{\alpha_0}(t) \geq x_i^\star(t)$ for all i (see Figure 5). As a result, $(\alpha, t) \mapsto x^\alpha(t)$ is a function H such that $H(0,t) = x(t)$ and $H(\alpha_0, t) = x^\star(t)$. Moreover it is continuous because taking the minimum is a continuous operation. By definition of homotopy, we can assert that x and $x^\star$ are homotopic. As a consequence, trajectories respecting a given priority graph G are homotopic. It results that homotopy classes are uniquely encoded by the priority graph. □

The above theorem asserts that the homotopy classes are uniquely encoded by the priority graphs. It raises the question whether every priority graph can be mapped to a non-empty class of homotopic trajectories, i.e. is feasible.

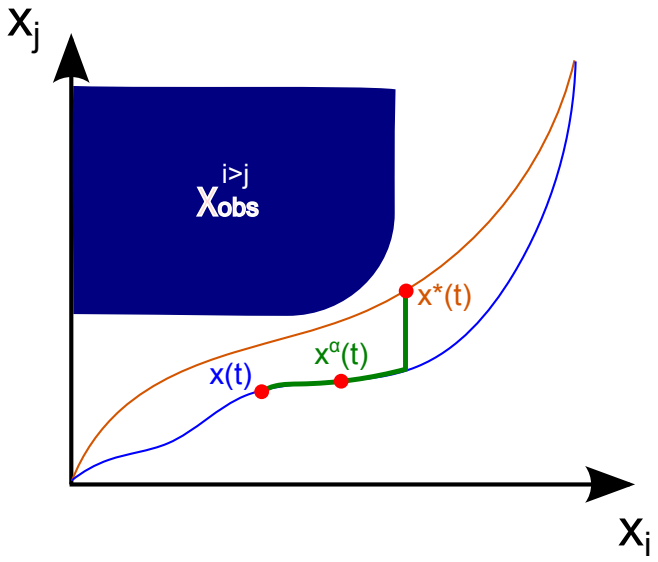

Figure 5: The continuous transformation changing any trajectory respecting a priority graph G into the optimal trajectory $x\star$ respecting this priority graph.

3.2 Priorities' feasibility and deadlocks

In the example of the left drawing of Figure 6, some priorities are obviously not feasible, i.e. they do not encode a non-empty homotopy class of feasible trajectories. Indeed, if the the robot 2 passes before the robot 1 and the robot 2 passes before the robot 3, the robot 1 must pass before the robot 3, because the three paths have a common intersection point. Hence, for this particular situation, the priority graph cannot be cyclic and the priority relation must be an order relation to be feasible. If the assigned priorities are cyclic, it will result in a deadlock situation, each robot waiting for the other to pass. This example shows that there is a strong link between priorities' feasibility and deadlocks. Loosely speaking, a priority graph is feasible, if the assigned priorities do not result inevitably in a deadlock configuration. In the following, we revisit and we characterize the familiar notion of deadlock configuration, by tying it to a priority graph. It allows to provide a necessary and sufficient condition for priorities' feasibility.

Definition 6 (Deadlock configuration)**.** *Given a priority graph G and a configuration $x^0 \in \chi$, x^0 is a deadlock configuration if there exists a group of robots such that each robot of the group is at the boundary of either a collision set or a G-priority violation with another robot of the group.*

Note that the above definition is conservative and excludes configurations in which robots must slide on each other to move forward. This is a pathological case that should be excluded in practice even if a feasible trajectory exists.

Figure 7 depicts examples of deadlock configurations. The two examples of Figure 7 underline the strong link between priorities' cyclicity and deadlocks. Consider the 3-robot deadlock of the left drawing of Figure 7. In this example, robot 2 cannot move forward because of robot 3, robot 1 cannot move forward

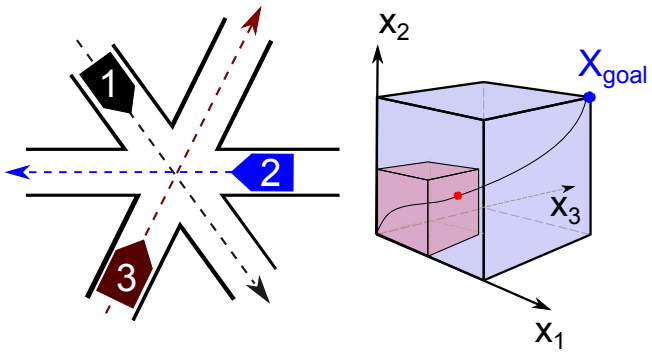

Figure 6: The left drawing represents a 3-path intersection with a common intersection point and 3 robots going through the intersection. The right drawing depicts what happens in the coordination space when cyclic priorities are assigned. Any feasible trajectory respecting cyclic priorities should stay in the red box $[0, \frac{1}{2}]^3$, but reaching the goal obviously requires to cross this box.

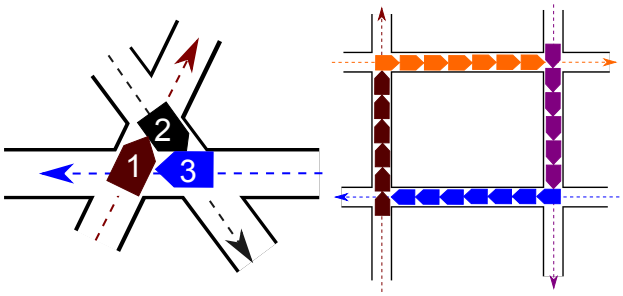

Figure 7: Two examples of deadlock configurations. On the left side, 3 robots are implicated in the deadlock. On the right side, the deadlock is caused by a priority cycle involving much more robots. In both examples, each robot of the group is at the boundary of a collision or a priority violation with another robot of the group.

because of robot 2 and robot 3 cannot move forward because of robot 1. As a consequence, the configuration x of the system is at the boundary of $\chi_{\text{obs}}^{3\succ 2}$, $\chi_{\text{obs}}^{2\succ 1}$ and $\chi_{\text{obs}}^{1\succ 3}$, i.e. $x \in \partial\chi_{\text{obs}}^{3\succ 2} \cap \partial\chi_{\text{obs}}^{2\succ 1} \cap \partial\chi_{\text{obs}}^{1\succ 3}$, where $\partial\chi_{\text{obs}}^{i\succ j}$ denotes the boundary of a $\chi_{\text{obs}}^{i\succ j}$, as depicted in Figure 8. This example motivates the following lemma that yields the set of deadlock configurations associated to a cycle of priorities:

Lemma 1 (Deadlock configurations associated to a cycle of priorities)**.** *The configurations* $\chi_{\text{deadlock}}^{\mathcal{C}} := \bigcap_{i \xrightarrow{\mathcal{C}} j} \partial\chi_{\text{obs}}^{i\succ j}$ *associated to any cycle* $\mathcal{C}$ *in the priority graph are all necessarily deadlocks.*

Proof. The group of robots composed of the nodes of $\mathcal{C}$ respects the condition of Definition 6. Indeed, for all $i \xrightarrow{\mathcal{C}} j$, robot i is at the boundary of a collision or a priority violation with robot j by definition. □

The above lemma sketches the role of priority cycles in deadlock formation. The following theorem provides a necessary and sufficient condition for priori-

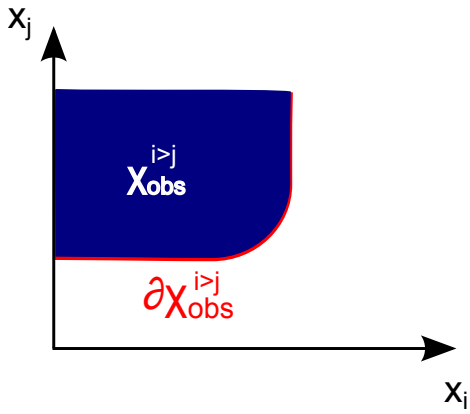

Figure 8: The boundary of a fixed-priority collision cylinder

ties' feasibility, i.e. ensuring that assigned priorities will not result inevitably in a deadlock configuration, and confirms the key role of priority cycles.

Theorem 2 (Priorities' feasibility)**.** *A priority graph G encodes a non-empty homotopy class of feasible and deadlock-free trajectories if and only if for all cycles $\mathcal{C}$ in G, $\chi^{\mathcal{C}}_{\text{deadlock}} := \bigcap_{i \xrightarrow{c} j} \partial\chi^{i\succ j}_{\text{obs}} = \emptyset$, in which case the priority graph G is said to be feasible.*

Proof. Suppose that for all cycles $\mathcal{C}$ in G, $\chi^{\mathcal{C}}_{\text{deadlock}} = \emptyset$, then there exists a feasible trajectory. It will be constructed the proof of Theorem 3.

Suppose now that for a cycle $\mathcal{C}$ in G, we have $\chi^{\mathcal{C}}_{\text{deadlock}} \neq \emptyset$. We are going to build a closed box in the coordination space capturing any trajectory which respects those priorities. To do so, take $x^0 \in \chi^{\mathcal{C}}_{\text{deadlock}}$. For each $i \in \text{nodes}(\mathcal{C})$, consider $K^0_j = \{x \in \chi : x_j = x^0_j \text{ and } 0 \leq x_i \leq x^0_i \text{ for } i \neq j\}$. By definition of $\chi^{i\succ j}_{\text{obs}}$ we have

$$x^0 \in \partial\chi^{i\succ j} \Longrightarrow K^0_j \subset (\chi^{i\succ j}_{\text{obs}} \cup \partial\chi^{i\succ j}_{\text{obs}})$$

As $\forall i \xrightarrow{\mathcal{C}} j, x^0 \in \partial\chi^{i\succ j}_{\text{obs}}$, applying the latter result yields:

$$\forall i \xrightarrow{\mathcal{C}} j, K^0_j \subset (\chi^{i\succ j}_{\text{obs}} \cup \partial\chi^{i\succ j}_{\text{obs}})$$

Any trajectory respecting the subset of priorities defined by $\mathcal{C}$ cannot cross $\bigcup_{i \xrightarrow{c} j} (\chi^{i\succ j}_{\text{obs}} \cup \partial\chi^{i\succ j}_{\text{obs}})$ (see Figure 3). As a consequence, the feasible trajectory cannot cross the smaller set $\bigcup_{i \xrightarrow{c} j} K^0_j$, which is equal to $\bigcup_{j \in \text{nodes}(\mathcal{C})} K^0_j$, as every node is involved in a cycle.

In the configuration space restricted to the coordinates which appear in $\mathcal{C}$, $\bigcup_{j \in \text{nodes}(\mathcal{C})} K^0_j$ is the set of upper faces of the semi-open rectangular parallelepiped $[0, x^0)$. As a result, in this restricted configuration space, the trajectory is confined in $[0, x^0)$, which is a closed box, apart at the point x^0 which lies on the boundary, but the trajectory cannot pass through x^0 because it is a deadlock configuration. Thus, finally, there is no feasible deadlock-free trajectory respecting the priority graph G. □

From a theoretical viewpoint, the latter theorem gives insight into the strong relationship between deadlocks and priorities, as it proves the set of priority graphs G such that $\chi^{\mathcal{C}}_{\text{deadlock}} = \emptyset$ for all cycles $\mathcal{C}$ in G is in bijective correspondance with the set of homotopy classes of feasible deadlock-free trajectories. The fact that cycles in priorities may result in deadlocks had been known for long [23, 24, 25], and preventing cycles in priorities has been a naive and conservative way to avoid deadlocks. The latter theorem refines this knowledge by characterizing (only) the problematic cycles, and puts it on firm mathematical grounds.

4 Coordinating robots with assigned priorities

The priority graph captures the combinatorial discrete part of the coordination problem. As a result, when priorities are assigned, optimization is a much easier task because it boils down to a continuous problem: finding an optimal trajectory over a set of homotopic trajectories.

In the following, we provide a solution to the problem of finding an optimal trajectory respecting a priority graph, i.e. coordinating robots with assigned priorities, for a particular but meaningful objective function. The problem and its solution being in continuous time, the optimal solution is not readily implementable. We thus propose a discrete-time algorithm that approaches the optimal solution as the time-step decreases.

Note that, the question at hand can be linked to the problem of finding Pareto optima as described in [18], where a discrete-time version of the optimal trajectory below has been already proposed and referred to as left-greedy trajectory. In the latter paper, its optimality follows from the fact the state-space is discrete and negatively curved. In the present paper we propose a more elementary proof in continuous time. In continuous time, the definition is however slightly different as it strongly relies on properties of the priority graph.

4.1 Optimal trajectory for assigned priorities

In the following, we suppose that the priority graph G is fixed so the problem boils down to finding an optimal trajectory satisfying the priorities assigned in G. As an optimality criterion, we propose to focus on the the particular but meaningful cost function $c(x) = \frac{1}{n}\sum_{i=1}^{n} T_i = \frac{1}{n}\sum_{i=1}^{n} x_i^{-1}(1)$ with $x_i^{-1}(1)$ denoting the first date at which x_i reaches coordinate 1 and $T_i = x_i^{-1}(1)$ being the exit time for the robot i. Such objective functions penalizing the average time elapsed to reach the goal region are standard (see, e.g. [16]). The considered optimality problem consists of finding a feasible trajectory $x^\star$ respecting priorities defined by G that minimizes the average exit time of the robots, that is, $c(x^*) \leq c(x)$ for any feasible trajectory x with priority graph G.

Algorithm 1 provides the optimal velocity vector for any configuration x^0. The algorithm belongs to the so-called "bug familly" (see [26]). The idea is to allow the robots to move at maximum speed along their paths, i.e. $x'(t) = v^{\max}$, until the boundary of a fixed priority collision cylinder $\partial\chi^{j\succ i}_{\text{obs}}$ is reached. In this case, the boundary is followed at maximum possible speed. When $x(t)$ belongs to $\partial\chi^{i\succ j}_{\text{obs}}$ and $i \xrightarrow{G} j$, $x'_j(t)$ must be lower or equal to $x'_i(t)\left.\frac{\partial x_j}{\partial x_i}\right|_{\partial\chi^{i\succ j}_{\text{obs}}}(x(t))$, where

$\frac{\partial x_j}{\partial x_i}\Big|_{\partial\chi_{\text{obs}}^{i\succ j}}$ denotes the partial derivative of x_j in the direction x_i along $\partial\chi_{\text{obs}}^{i\succ j}$. Otherwise, x would inevitably collide $\chi_{\text{obs}}^{i\succ j}$ just after time t. In order to maximize the velocity of robot j, we thus let: $x_j'(t) = \min(v_i^{\max}, x_i'(t)\,\frac{\partial x_j}{\partial x_i}\Big|_{\partial\chi_{\text{obs}}^{i\succ j}}(x(t)))$. The following algorithm outputs a feasible optimal velocity for any given input configuration. The underlying principle is illustrated in Figure 9.

Algorithm 1 Maximum velocity

Input: x^0, feasible G

function MAXIMUMVELOCITY

$G' := \{i \xrightarrow{G} j : x^0 \in \partial\chi_{\text{obs}}^{i\succ j}\}$

for $j \in \text{nodes}(G')$ from higher to lower priority **do**

$v_j^0 \leftarrow \min\left(v_j^{\max}, \min_{i \xrightarrow{G'} j} v_i^0\,\frac{\partial x_j}{\partial x_i}\Big|_{\partial\chi_{\text{obs}}^{i\succ j}}(x^0)\right)$

end for

return v^0

end function

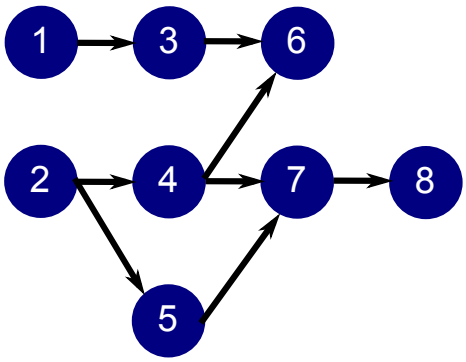

Figure 9: An example of directed acyclic graph (DAG). Suppose that this graph is the sub-graph G' as defined in Algorithm 1. Algorithm 1 will assign every velocity v_i^0 for example in the order defined by the labelling, i.e. from v_1^0 to v_8^0. Robots at the roots will be assigned a maximal velocity: $v_2^0 = v_2^{\max}$ and $v_1^0 = v_1^{\max}$. When computing v_7^0, for instance, v_4^0 and v_5^0 will be already defined, and v_7^0 will be computed as follows: $v_7^0 = \min(v_7^{\max}, v_4^0\,\frac{\partial x_7}{\partial x_4}\Big|_{\partial\chi_{\text{obs}}^{4\succ 7}}, v_5^0\,\frac{\partial x_7}{\partial x_5}\Big|_{\partial\chi_{\text{obs}}^{5\succ 7}})$

Theorem 3 (Optimality for assigned priorities)**.** *Given a feasible priority graph G and an initial configuration x^{init}, the trajectory $x^\star$ defined by the differential equation $x^{*\prime}(t) = v(x^*(t), G)$ with initial condition x^{init} and velocity profile returned at all times by Algorithm 1 is optimal.*

Proof. The proof is based on the following steps: the solution is shown 1- to be well-defined 2- to reach the goal in finite time 3- to be optimal.

1- The velocity profile output by Algorithm 1 is uniquely defined because there is no cycle $\mathcal{C}$ such that $\bigcap_{i \xrightarrow{\mathcal{C}} j} \partial\chi_{\text{obs}}^{i\succ j} \neq \emptyset$. Indeed, as a consequence, the sub-graph G' of G retaining only edges $i \to j$ such that $x^0 \in \partial\chi_{\text{obs}}^{i\succ j}$ is acyclic.

G' is a directed acyclic graph (DAG); as a result, it can be traversed from higher priorities to lower priorities. Moreover, the returned velocity does not depend on the order in which nodes are traversed because at line 4, v_j^0 depends only on the projected velocities of higher priority nodes pointing to j (see Figure 9). The maximal solution $x^\star$ of the proposed differential equation exists because the boundary of every cylinder χ_{obs}^{ij} is supposed to be continuous and piecewise smooth (see Assumption 2).

2- There is always at least one robot at maximal velocity (the one(s) such that no robot has priority over it), so it reaches coordinate 1 in finite time. Once it has exited the intersection, there is at least another robot at maximal velocity unless the goal has been reached. As there is a finite number of robots, a simple induction ensures $x^\star$ reaches χ_{goal} in finite time.

3- Suppose the trajectory $x^\star$ as defined in the theorem is not optimal. Then there exists a distinct trajectory $\tilde{x}$ such that $c(\tilde{x}) < c(x^\star)$. Let T^0 be the first time at which a component of $\tilde{x}$ becomes (strictly) greater than the same component of $x^\star$ after T^0: $\forall i \in \{1...n\}, \forall t \leq T^0, x_i^\star(t) \geq \tilde{x}_i(t)$ and $\forall t \in (T^0; T^0 + \epsilon)$, $x_j^\star(t) < \tilde{x}_j(t)$ for some $\epsilon > 0$. If there is no $i \xrightarrow{G} j$ such that $x(T^0) \in \partial\chi_{\text{obs}}^{i \succ j}$, then ${x'}_j^\star(t) = v_j^{\max}$ at time T^0 and $\tilde{x}'_j(t)$ should be greater than $v_j^{\max}$, which is impossible. As a result, there exists a vertex $i \xrightarrow{G} j$ such that x^* follows $\partial\chi_{\text{obs}}^{i \succ j}$ after T^0. If $\tilde{x}_j$ becomes greater than $x_j^\star$ after T^0, it means the velocity associated to the trajectory $\tilde{x}$ is pointing strictly inside $\chi_{\text{obs}}^{i \succ j}$, which is impossible as it has been assumed to be collision-free. The trajectory $x^\star$ is thus optimal. □

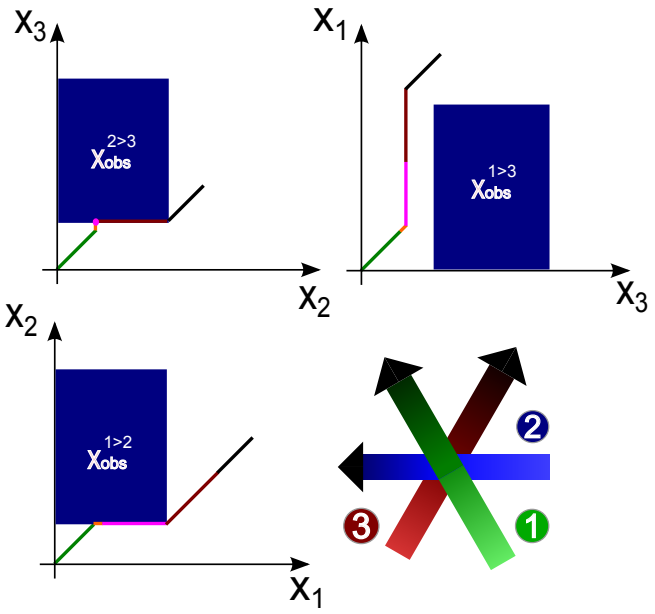

Figure 10: The bottom-right drawing depicts a 3-path intersection where priorities are fixed to $1 \succ 2$, $2 \succ 3$ and $1 \succ 3$. The three other plots visualize in each plane (x_i, x_j) the obstacle region and the optimal trajectory of Theorem 3. The color of the trajectory changes when a collision pair leaves or reaches the boundary of the obstacle region.

Figure 10 illustrates the trajectory defined in Theorem 3 for a 3-path-intersection.

4.2 Algorithm in discrete time

The optimal trajectory $x^\star$ of Theorem 3 has been defined as the solution of a differential equation in continuous time, and is thus not implementable numerically. We propose here an easy-to-implement algorithm in discrete time that approaches the optimal trajectory as the time-step decreases.

The main idea of Algorithm 2 is to use a bang-bang type control, and to discretize the time with a time-step ΔT. The robots travel at maximum speed along the paths, i.e. $x_i'(t) \leftarrow v_i^{\max}$ and $x_i(t+\Delta T) \leftarrow x_i(t) + v_i^{\max}\Delta T$ (see line 9) until a boundary is reached. When the boundary of a fixed priority collision cylinder $\partial\chi_{\text{obs}}^{j\succ i}$ is approached at a distance less than $v_j^{\max}\Delta T$ in the direction j (see lines 5 and 6), the robot j is merely stopped for the next time-step, i.e. $x_j'(t) \leftarrow 0$ and $x_j(t+\Delta T) \leftarrow x_j(t)$ (see line 7).

Algorithm 2 The collision avoidance algorithm with fixed priorities

Input: $x^{\text{init}}, \chi_{\text{obs}}, \chi_{\text{goal}}, G$
  function FIXEDPRIORITIESOPTIMALTRAJECTORY
    $x(0) \leftarrow x^{\text{init}}$
    while $x(T) \notin \chi_{\text{goal}}$ **do**
      for $i \in \{1...n\}$ **do**
5: $\hat{x} \leftarrow x(T) + v_i^{\max}\Delta T\mathbf{e}_i$
        if $\exists j \xrightarrow{G} i$ s.t. $\hat{x} \in \chi_{\text{obs}}^{j\succ i}$ **then**
          $x_i(T+\Delta T) = x_i(T)$
        else
          $x_i(T+\Delta T) = x_i(T) + v_i^{\max}\Delta T$
10: **end if**
      end for
      $T \leftarrow T + \Delta T$
    end while
    return $x : [0, T] \to \chi$
15: **end function**

5 Application to intersection management

Due to the promises in autonomous cars design, automated intersection management has attracted much interest recently [27, 28, 29, 30, 3]. Two main goals motivate the research in this topic. The first one is to avoid accidents due to collisions that occur mainly at intersections and because of human errors. Indeed, human error is a leading factor in over 90% of all road accidents [31, 32]. The second one is to enhance road traffic efficiency, given that intersections represent bottlenecks in the traffic network.

Traditionally, intersection management systems are formalized as a multi-agent system and heuristics algorithms are used to optimize the traffic through the intersection [3, 33]. When vehicles request to go through the intersection, they are assigned a planned trajectory by the intersection manager, and they

need to follow the planned trajectory to ensure collision-freeness, deadlock-freeness and efficiency. Such methods have proved their ability to reduce traffic congestion [20] and the risk of accidents [34].

In the context of Intelligent Transportation Systems, due to unpredictable events or cooperation with human-driven vehicles for example, following a planned velocity is not robustly feasible and replanning is of high complexity. With our priority-based framework, safety is ensured as long as the priority graph is respected: it is a less restrictive condition thus providing robustness.

As already mentioned, the number of potential priority graphs is $2^{n(n+1)/2}$. Enumerating all possible graphs, retaining the feasible ones, and computing the cost associated to the (locally) optimal trajectories very quickly becomes of prohibitive numerical complexity when the number of cars in the intersection n grows. On the other hand, a naive algorithm which would eliminate all cycles would possibly lead to very suboptimal trajectories. As a result, a high-level planning algorithm is needed to construct feasible graphs of interest in which some cycles are allowed to appear, as long as they lead to deadlock-free trajectories in accordance with Theorem 2. This algorithm can be based on several heuristics depending on the application.

Regarding the application to intersection management, we propose to build high-level heuristics upon a very natural and intuitive concept: the notion of storage capacity of the intersection between two successive collision zones, and a companion mathematical tool, the intersection graph.

5.1 The intersection graph and the capacity concept

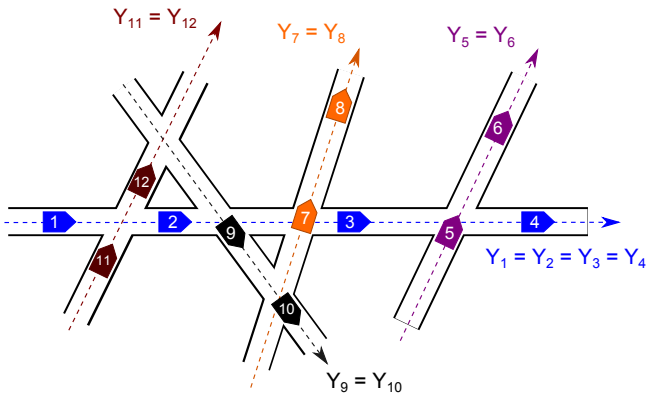

Figure 11: Vehicles at an intersection area. Vehicles travelling along the same path are depicted with the same color.

A key difference in intersection management compared to general coordination of robots with fixed paths is that vehicles in the same lane going in the same direction travel along the same path. Let Γ denote the set of paths of the intersection, $\forall i \in \{1...n\}$, $\gamma_i \in \Gamma$, as depicted in Figure 11. This justifies the definition of the collision region associated to two distinct paths:

Definition 7 (Collision region)**.** *Given two paths* (Γ_a, Γ_b) *(possibly the same),* $K_{ab} \subset [0,1]^2$ *denotes the set of positions* (x_a, x_b) *such that a vehicle on* Γ_a *at* x_a *and a vehicle on* Γ_b *at* x_b *collide.*

Note that for $\gamma_i = \Gamma_a$ and $\gamma_j = \Gamma_b$, K_{ab} is the cross-section of χ_{obs}^{ij} (in other words the projection onto a 2-dimensional plane), and we define similarly $K_{a\succ b}$ and $\partial K_{a\succ b}$, as depicted in Figure 12.

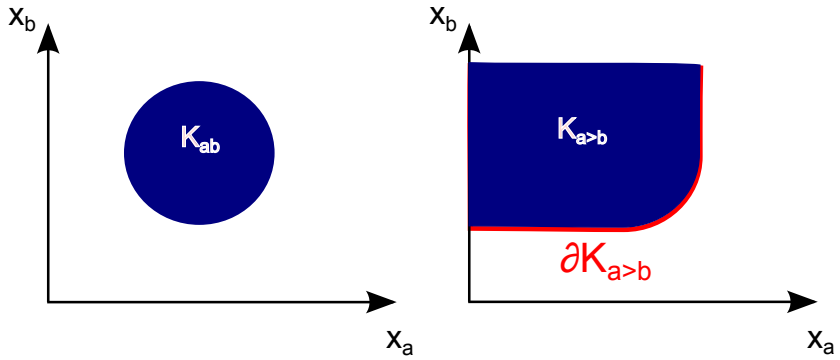

Figure 12: The collision region, the fixed-priority collision region and its boundary.

In the following, we introduce the intersection graph and the concept of storage capacities that enable to build very efficient priority assignment rules in the particular context of intersection management.

The intersection graph Essentially, the intersection graph is a graph representing the sequence of collision points that vehicles cross through a given intersection in a synthetic way.

Definition 8 (Intersection graph)**.** *The intersection graph is the graph of nodes* $\{K_{ab}\}_{a\neq b}$ *and edges defined as follows:*

$$\left\{ K_{ab} \to K_{bc} : a, b, c \text{ distinct}, \inf_{x\in\partial K_{b\succ a}} x_b \leq \inf_{x\in\partial K_{b\prec c}} x_b \right\}$$

More prosaically, there is an arrow from K_{ab} to K_{bc} if a vehicle on path Γ_b crosses intersection area $\Gamma_b \cap \Gamma_c$ after crossing $\Gamma_a \cap \Gamma_b$. Figure 13 shows a 3-path intersection and the associated intersection graph.

The storage capacity concept The intersection graph enables to decompose the paths of the intersection into sections that lie between two collision regions. It is of high interest to note that a limited number of vehicles can be stored in these sections without blocking the traffic on the intersecting paths. For example, in the intersection depicted in Figure 14, at most 5 vehicles can be stored in the section $K_{ab} \to K_{bc}$. If more vehicles enter this section, e.g. 6 vehicles, as depicted in the right drawing of Figure 14, the traffic is obviously blocked on path Γ_a.

In the following formal definition of the capacity concept, for simplicity's sake, we assume that all vehicles have the same shape and size. The definition could easily be extended to take into account the different sizes of the vehicles.

Definition 9 (Capacity)**.** *Given three distinct paths* Γ_a*,* Γ_b *and* Γ_c*, the capacity of the section* $K_{ab} \to K_{bc}$ *of the intersection graph is the maximum num-*

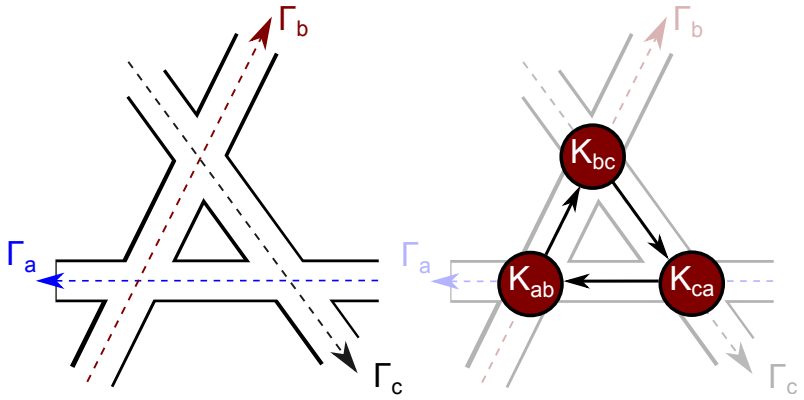

Figure 13: An example of intersection graph associated to a 3-path intersection. Note that the intersection graph here is cyclical.

ber of vehicles that can be stored on Γ_b *between coordinates* $\max_{x\in K_{ab}} x_b$ *and* $\min_{x\in K_{bc}} x_b$.

Note that, when some sections of the intersection graph are occupied beyond their capacity, it can lead to very inefficient situations as depicted in the left drawing of Figure 15. As vehicle 1 has created an over capacity in section $K_{ab} \to K_{bc}$, the traffic is blocked on Γ_a, and vehicles 2, 3, ... and 7 are blocked: it is very inefficient. When such a situation is propagated through the intersection, it can lead to a deadlock configuration as depicted in the right drawing of Figure 15.

In the following, we present the principle of dynamic priority assignment that consists of assigning priorities one after the other, leading to the realm of real-time algorithms, allowing one to tackle the challenging case of an open intersection with varying number of vehicles. Then, we introduce heuristics rules based on the occupancy of the intersection graph meant to assign priorities without creating inefficient situations.

5.2 Dynamic priority assignment

In this application, we propose to iteratively build the trajectory in the coordination space and to dynamically update the priority graph. This will lead to algorithms that are suited to real time, and open intersections. The basic principle is as follows. Consider the example of Figure 16 with only two vehicles on two distinct paths. The algorithm just moves forward the vehicles without assigning priorities until $(x_1(t_0) + v_1^{\max}\Delta T, x_2(t_0)) \in K^{2\succ 1}$. At time t_0, priorities $1 \succ 2$ and $2 \succ 1$ are still both possible. The algorithm can choose to let vehicle 1 move forward and assign priority $1 \succ 2$ (as in the example of Figure 16), or to slow down vehicle 1 and assign priority $2 \succ 1$. Time t_0 thus appears as a key instant when a decision has to be made. In the following, we opted for the natural decision to let vehicle 2 move forward, unless it violates a rule among a set of rules designed to avoid ending up in inefficient or deadlock configurations. When the vehicle 2 is authorized to move forward, the priority graph must be updated: $2 \succ 1$.

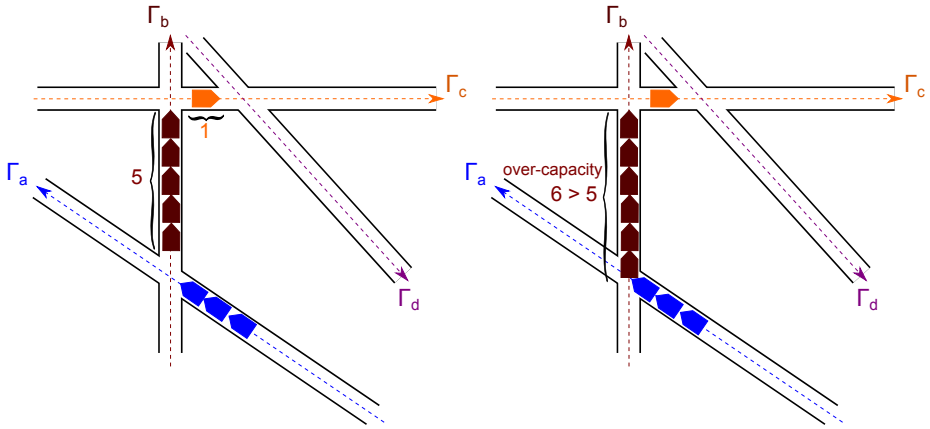

Figure 14: Definition of the capacity concept. At most 5 vehicles can be stored in the section $K_{ab} \to K_{bc}$, and only 1 in the section $K_{bc} \to K_{cd}$. On the left drawing, 6 vehicles have entered the section $K_{ab} \to K_{bc}$ whereas the capacity of this section is 5. The traffic is blocked on path Γ_a.

The inefficiency of acyclic priority assignment Before going any further into the proposed algorithm, we would like briefly discuss two naive approaches to assign priorities iteratively. First of all, note that traffic lights constitute a very naive approach to assign priorities. It leads to very safe situations, and the rule assignment is easily understood by any driver. However, it is evidently very suboptimal especially at low density, at least in the framework of this paper, where a central planner is allowed to assign the vehicles' velocities at all times. Then, a somehow more sophisticated approach but still naive consists of eliminating all cycles in the priority graph. This basic rule ensures deadlock-freeness but is suboptimal as Theorem 2 proves that acyclic priority graphs can be deadlock-free. The key role of cycles in the deadlock prevention problem of resource allocations systems is well-known [23, 24, 25].

Consider for instance the situations depicted in Figure 17: a vehicle can be stopped at the intersection because of a resulting cycle, whereas there would be no deadlock if it passed, which is a source of inefficiency. The storage capacities of edges of the intersection graph play a key role in detecting and preventing this kind of situations.

5.3 Efficient heuristic rules for deadlock-free priority assignment

In this section, we advocate that the most important rule to follow is to avoid over capacity propagation through the intersection as depicted in Figure 18. Mathematically it boils down to avoiding the creation of two adjacent over capacity sections. Moreover, when vehicles enter in a section beyond its capacity, they will have to enter the next section in order to free the current section, otherwise the over capacity will stay for ever. That is why we introduce a propagation mechanism: the first vehicles of the current section will request to enter in the next section.

In the actual implementation, the rule described in Figure 18 can be a little

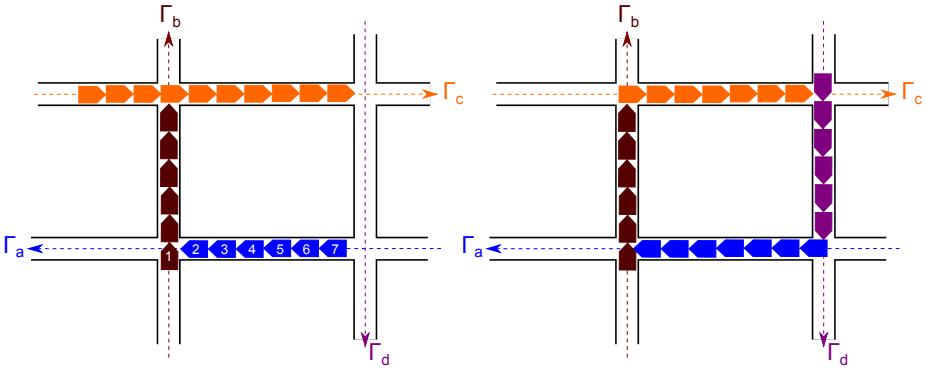

Figure 15: The left drawing depicts an example of inefficiency due to over capacity in the section $K_{ab} \rightarrow K_{bc}$ of the intersection graph, and the right drawing represents a deadlock configuration due to over capacity propagation.

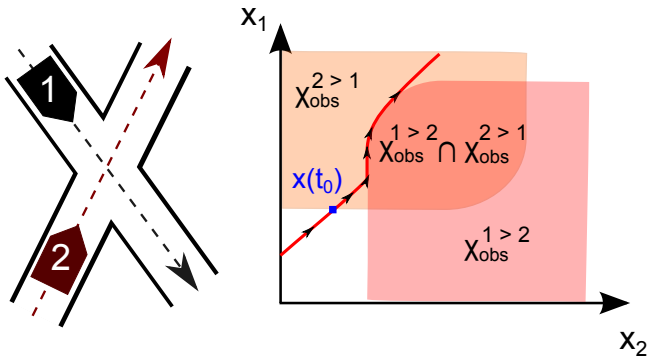

Figure 16: An example illustrating the dynamic priority assignment approach. When computing iteratively the trajectory, the priority between vehicles 1 and 2 needs to be assigned when the boundary of one the fixed-priority obstacle regions is reached, i.e. at $x(t_0)$.

relaxed in order to anticipate the fact that a section is about to be freed, i.e. the over capacity is about to disappear. It enables to grant the permission to enter the collision region in certain cases although the following section presents an over-capacity, leading to substantial gains in the efficiency. A final rule that leads to important gains in efficiency consists of clustering some vehicles travelling along the same path. Each vehicle in a cluster of vehicles following each other is allowed to to request the entry in a given section at the same time as the first vehicle of the group. It enables to create clusters that share the same priorities when travelling through the intersection.

5.4 Numerical experiments

The algorithms presented in this paper have been implemented into a simulator coded in Java and have proved their ability to run in real-time. Only straight paths are implemented and all vehicles are supposed to be circle-shaped with a common diameter. They clearly illustrate the benefits of the introduced frame-

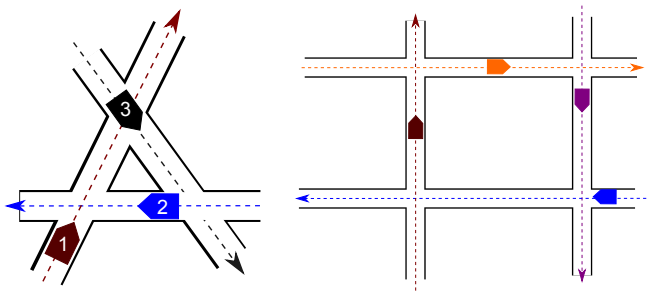

Figure 17: Two examples of inefficiency of the acyclic priorities approach. In both cases, a vehicle is not authorized to enter the intersection because it would cause a cycle the priority graph. However, it would be much more efficient to let it enter the intersection.

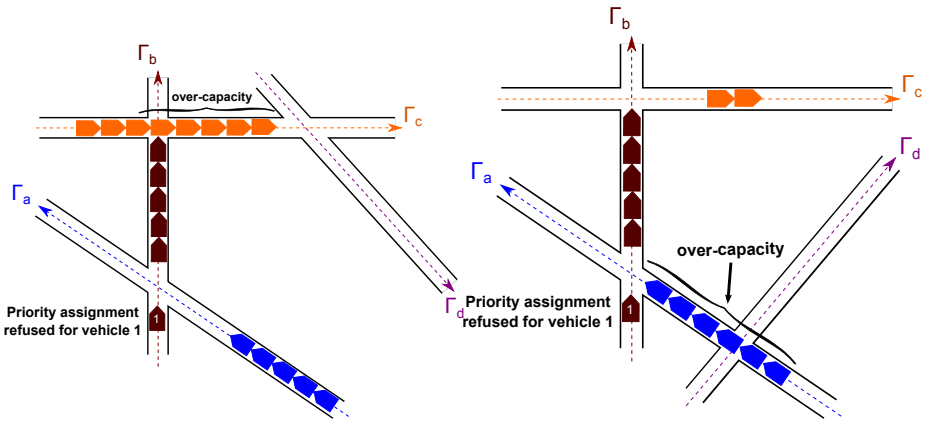

Figure 18: The principle of over capacity propagation avoidance. In both drawings, the vehicle 1 is stopped because otherwise, it would result in two adjacent over-capacity sections.

work, not only for the case of an usual simple intersection, but also for the more complex case of a network of intersections, where an acyclic priority graph is evidently far from optimal, because of cycles involving different intersections that should be allowed most of the time.

The proposed algorithm is compared with a naive acyclic priority assignment policy. In order to evaluate the performance of each algorithm, the following criteria are proposed. First, the increase in travel time in percentage measures the average ratio $(\Delta T_{travel} - \underline{\Delta T}_{\text{travel}})/(\underline{\Delta T}_{\text{travel}})$ over vehicles, where ΔT_{travel} is the actual travel time of the vehicle through the intersection and $\underline{\Delta T}_{\text{travel}}$ is the ideal travel time of the vehicle in the absence of other vehicles, that is the length divided by the maximum velocity. This quantity thus represents the increase in travel time due to other vehicles. We propose to plot it against the input traffic flow (in percentage), that represents the ratio between the actual input traffic flow and a continuous traffic flow on each path taken as reference (i.e. an input traffic flow of $v_a^{\max}/D$ on each path Γ_a where D denotes the diameter of the circle-shaped vehicles). The vehicles are generated randomly at

a constant rate over time.

Note that 50% is a natural upper bound for the performance of the intersection manager of the 4-path intersection. Indeed, when the traffic flow is very high, the best strategy is to alternate the traffic in each direction, so that only two of the four paths can output vehicles, it yields a 50% traffic flow ratio as depicted in the left drawing of Figure 21.

Case of a simple 4-paths intersection Simulations were carried out for a simple 4-path intersection depicted in the left drawing of Figure 19. Figure 20 presents a performance comparison between the proposed algorithm and a basic acyclic priority assignment policy.

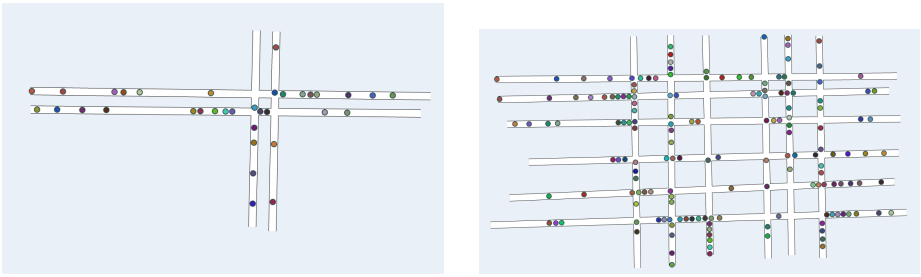

Figure 19: Screen-shot of the intersections used for the presented simulation results.

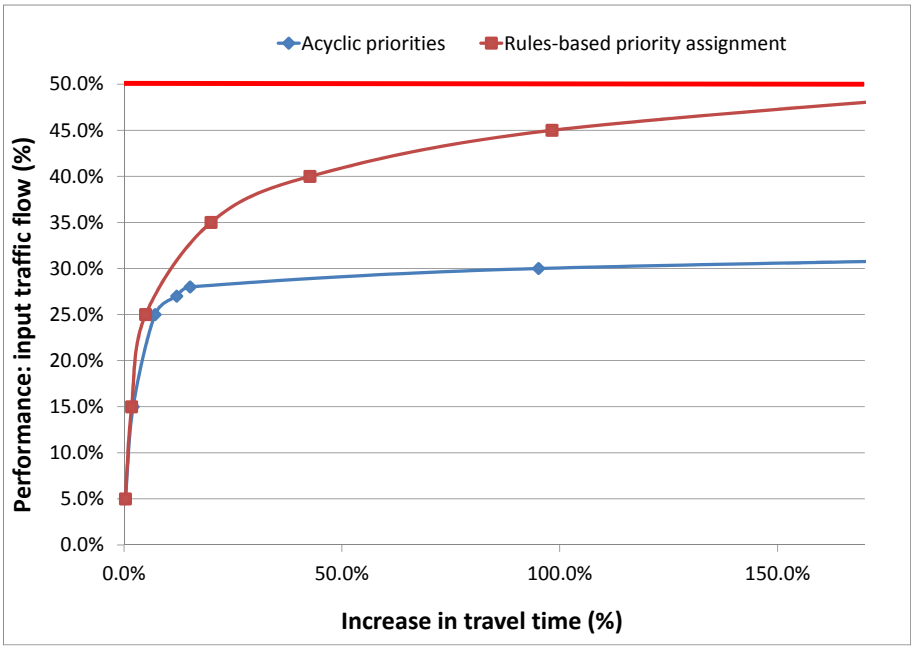

Figure 20: Performance comparison of priority assignment using rules based on intersection graph occupancy and acyclic priorities assignment for a 4-path intersection.

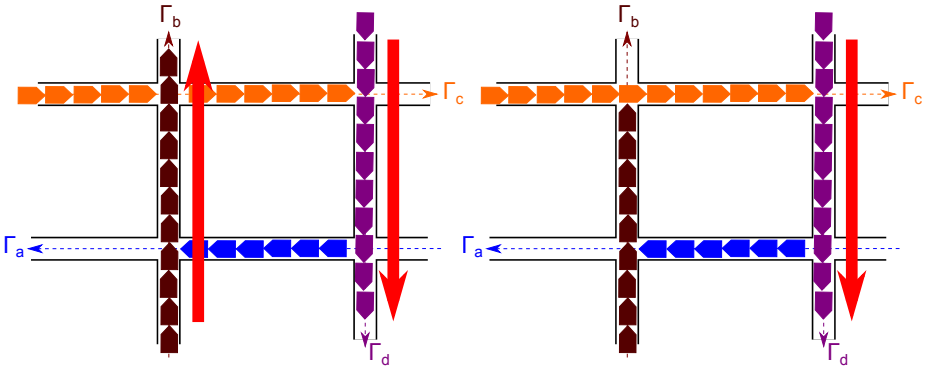

Figure 21: Saturation traffic flow of 50% for the rules-based priority assignment algorithm (left drawing) and of 25% for the acylic priorities algorithm (right drawing).

Figure 20 shows that the proposed algorithm yields very good performances as even for pretty high traffic flow, such as 35%, the average delay is below 40% in travel time increase. Moreover, it obviously outperforms the acyclic priorities approach. Also, it asymptotically approaches the upper bound on the performance when the increase in travel time grows to infinity. This is a remarkable feature, as the algorithm that consists only of avoiding cycles saturates for an input traffic flow around 30%, which can be explained in the following way: this algorithm carries out maximally permissive deadlock avoidance, and inefficient situations such as the one depicted in Figure 15 occur and propagate in three adjacent sections. As a result, when the traffic flow is high, only one path of the four paths can output vehicles, as depicted in the right drawing of Figure 21; it yields a saturation traffic flow of around 25%. Note that it illustrates that maximally permissive deadlock avoidance (see, e.g. [35]) in the context of intersections management is not a good strategy in terms of traffic flow efficiency.

Case of a network of intersections Simulations have also been carried out for a large scale intersection, such as the one depicted in the right drawing of Figure 19. The performance results for this intersection are depicted in Figure 22. Good performance is observed as even with a pretty high input traffic flow such as 20%, the average delay is lower than 10%, using the heuristic rules based on intersection graph occupancy. The performance of the acyclic priorities policy is particularly low in such a large scale intersection, because of the inefficient situations as depicted in Figure 17. The video attached to this paper[1] shows how vehicles are coordinated at such a large scale intersection area using the heuristic rules. One can easily notice how vehicles take care about not propagating over capacity. This video and the performance results of Figure 22 demonstrate that our naive heuristics rules guarantee an efficient scalable coordination of the vehicles.

[1]available at `http://www.youtube.com/watch?v=wStwIl9dtYI`

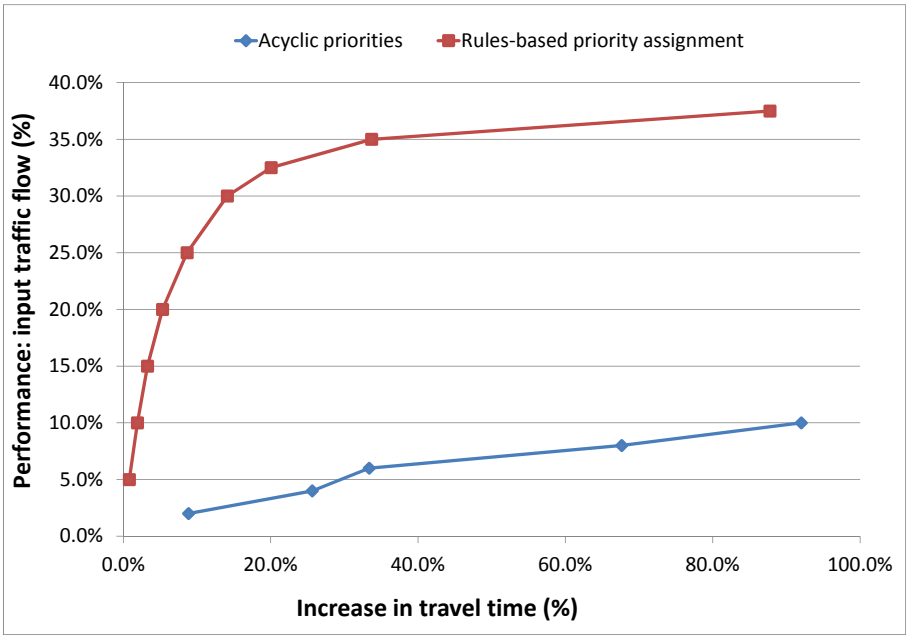

Figure 22: Performance comparison of priority assignment using rules based on intersection graph occupancy and acyclic priorities assignment for network of intersections.

6 Discussion

The key contribution of the present paper is the introduction and a mathematical study of the priority graph. It is a topological invariant encoding simply the homotopy classes for trajectories with non-negative velocities. Moreover, priorities give a semantics for these classes that enables the practitioner to build fast and efficient heuristics for motion planing in spite of its inherent complexity. The framework was applied to intersection management. It was coupled with a high-level planning heuristic and has proved to yield an efficient real-time coordination algorithm for intersection management. The framework proves versatile, and seems to open up new avenues in the field of robots coordination. Indeed in the following, we would like to provide a list of concluding remarks and points that deserve attention, for which the use of the proposed framework seems relevant. The several ideas seem to confirm the potential strength of the framework, but exploring each one would be naturally beyond the scope of the present paper.

Introducing dynamic constraints In the present paper, vehicles are allowed to start and stop instantly. However, in concrete applications, and in particular for an application to intersection management, vehicles have inertia. Actually, introducing dynamic constraints in our algorithms is a not-so-hard task, as explained in the companion paper [36]. Therein, the main point is anticipation of braking possibilities of vehicles (i.e. slowest trajectory at a given moment) to assign priorities.

Distribution The field of distributed control and multi-agent systems has attracted considerable interest over the last decade, notably for their good properties in term of robustness or scalability. We anticipate that two main avenues of research could build upon the interplay between the realm of distributed algorithms and the priority-based framework. First of all, priorities could be assigned by a central planning algorithm, and the choice of a particular trajectory over the corresponding homotopy class could be transferred to the vehicles in a distributed manner. Basically, this is the principle beneath traffic lights in intersection management: scheduling is centralized while dynamics is distributed. The central planner could take into account several criteria to assign priorities, such as time elapsed since a waiting vehicle has arrived, inertia of each vehicle, that is cars versus trucks, energy spent by each vehicle in order to adapt its velocity, emergency prioritary vehicles.

Distributing priority assignment is more challenging yet feasible. The priority graph is a very synthetic structure that could itself be constructed iteratively in a distributed way. A very basic distributed algorithm would be a first-come first-served policy: each vehicle waits for the others to exit the collision region before moving forward. Obviously this would work only for simple intersections and few vehicles. This lead is more exploratory, but it may be possible to develop distributed coordination systems and communication protocols that assign priorities in a distributed way.

Robustness and account of uncertainty Since the priority graph is a topological invariant, it is robust against small perturbations in trajectories. Therefore we believe our framework enables to develop more robust coordination algorithms taking into account several sources of uncertainty. Indeed, in the present framework, once a feasible priority graph has been found, in order to prevent collisions from occurring one only needs to check whether the priority graph is respected at all times. Compared to motion planning algorithms that can ensure safety only if the planned trajectory $x(t)$ is followed (possibly with an error margin), our framework potentially allows for a much wider range of uncertainties, as essentially the only requirement is that the actual trajectory remains in the homotopy class. Moreover, when assigning priorities, it should be possible to take into account uncertainties in the control and the sensors available in each particular vehicle, in the spirit of [37]. For example, priority assignment policies that are efficient but need a very reliable motion control will be reserved to very reliable automated cars, whereas cars with less motion control accuracy or even human-driven cars could be assigned priorities that are easier to execute.

Rule-based priority assignment The priority graph not only provides a simple coding for the homotopy classes of feasible trajectories; it provides also a semantics that is very intuitive, human-understandable. The consequence is two-fold. First simplicity opens new avenues to develop easy-to-implement and fast algorithms than can reason on a very synthetic discrete object: the priority graph. Second, semantics enables to develop intuitive heuristic rules to assign priorities, as illustrated by the application of Section 5.

Intersection management Finally, priority-based intersection management builds a bridge between multiple robot motion planning using the coordination space and heuristic algorithms used for intersection (and more generally traffic) management. Motion planning based on the configuration space in robotics comes with very efficient tools, such as the machinery of sampling-based methods like Rapidly exploring Random Trees. Casting some traffic management problems into this usual motion-planning framework of robotics seems very promising as illustrated by the results of Section 5. Thus, the next step is to deploy such algorithms on robots and intelligent vehicles.

Finally, one sees that priority-based coordination of robots can be very likely extended to real dynamical systems — even with uncertainties; it could be hybridized with some distribution techniques to inherit their good scalability and robustness properties; its semantics and its simplicity may lead to better algorithm design especially in intersection management. For all these reasons we believe that priority-based coordination will be concretely applied. This is one of the next issues we will address: deploy such algorithms on robots and intelligent vehicles.

References

[1] C. Tomlin, G. Pappas, and S. Sastry, "Conflict resolution for air traffic management: a study in multiagent hybrid systems," *Automatic Control, IEEE Transactions on*, vol. 43, pp. 509 –521, apr 1998.

[2] I. Sahin, "Railway traffic control and train scheduling based on inter-train conflict management," *Transportation Research Part B: Methodological*, vol. 33, no. 7, pp. 511 – 534, 1999.

[3] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of Artificial Intelligence Research*, vol. 31, pp. 591–656, March 2008.

[4] L. Alvarez and R. Horowitz, "Safe platooning in automated highway systems," Institute of Transportation Studies, Research Reports, Working Papers, Proceedings qt1v97t5w1, Institute of Transportation Studies, UC Berkeley, Jan. 1997.

[5] J. Hopcroft, J. Schwartz, and M. Sharir, "On the complexity of motion planning for multiple independent objects: *pspace*-hardness of the 'warehouseman's problem'," *The International Journal of Robotics Research*, vol. 3, no. 4, pp. 76–88, 1984.

[6] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986.

[7] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006. Available at http://planning.cs.uiuc.edu/.

[8] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[9] S. M. LaValle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 3, pp. 2847 –2852 vol.3, apr 1996.

[10] S. Leroy, J. P. Laumond, and T. Simeon, "Multiple path coordination for mobile robots: A geometric algorithm," in *In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI*, pp. 1118–1123, 1999.

[11] T. Lozano-Perez, "Spatial planning: A configuration space approach," 1980.

[12] T. Fraichard and C. Laugier, "Planning movements for several coordinated vehicles," in *Intelligent Robots and Systems '89. The Autonomous Mobile Robots and Its Applications. IROS '89. Proceedings., IEEE/RSJ International Workshop on*, pp. 466 –472, sep 1989.

[13] S. Akella and S. Hutchinson, "Coordinating the motions of multiple robots with specified trajectories," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 1, pp. 624 – 631 vol.1, 2002.

[14] J. Peng and S. Akella, "Coordinating multiple robots with kinodynamic constraints along specified paths," in *International Journal of Robotics Research*, pp. 221–237, Springer-Verlag, 2002.

[15] M. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Automated vehicle-to-vehicle collision avoidance at intersections," in *Proceedings of World Congress on Intelligent Transport Systems*, 2011.

[16] Y. Guo and L. Parker, "A distributed and optimal motion planning approach for multiple mobile robots," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 3, pp. 2612 –2619, 2002.

[17] R. Ghrist, J. M. O'Kane, and S. M. LaValle, "Computing pareto optimal coordinations on roadmaps," *The International Journal of Robotics Research*, vol. 12, pp. 997–1012, 2005.

[18] R. Ghrist and S. M. Lavalle, "Nonpositive curvature and pareto optimal coordination of robots," *SIAM Journal on Control and Optimization*, vol. 45, pp. 1697–1713, November 2006.

[19] J. Gregoire, S. Bonnabel, and A. de La Fortelle, "Optimal cooperative motion planning for vehicles at intersections," in *Navigation, Perception, Accurate Positioning and Mapping for Intelligent Vehicles, Workshop, 2012 IEEE Intelligent Vehicles Symposium*, 2012.

[20] K. Dresner and P. Stone, "Multiagent traffic management: a reservation-based intersection control mechanism," in *Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004. Proceedings of the Third International Joint Conference on*, pp. 530 –537, july 2004.

[21] P. O'Donnell and T. Lozano-Periz, "Deadlock-free and collision-free coordination of two robot manipulators," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pp. 484 –489 vol.1, may 1989.

[22] S. J. Buckley, "Fast motion planning for multiple moving robots," in *Proceedings IEEE International Conference on Robotics & Automation*, pp. 322–326, 1989.

[23] M. Lawley and S. Reveliotis, "Deadlock avoidance for sequential resource allocation systems: Hard and easy cases," *IEEE Transactions on Automatic Control*, vol. 46, pp. 1572–1583, 2001.

[24] E. G. Coffman and M. J. Elphick, "System deadlocks," *Computing Surveys*, vol. 3, pp. 67–78, 1971.

[25] M. Jäger and B. Nebel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," in *In IROS*, pp. 1213–1219, 2001.

[26] V. J. Lumelsky and A. A. Stepanov, *Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape*, pp. 363–390. New York, NY, USA: Springer-Verlag New York, Inc., 1990.

[27] J. A. Misener, "Cooperative intersection collision avoidance system (cicas): signalized left turn assist and traffic signal adaptation," tech. rep., 2010.

[28] O. Mehani and A. de La Fortelle, "Trajectory planning in a crossroads for a fleet of driverless vehicles," in *Proceedings of the 11th international conference on Computer aided systems theory*, EUROCAST'07, (Berlin, Heidelberg), pp. 1159–1166, Springer-Verlag, 2007.

[29] A. Colombo and D. Del Vecchio, "Efficient algorithms for collision avoidance at intersections," *Hybrid Systems: Computation and Control*, 2012.

[30] I. Zohdy and H. Rakha, "Optimizing driverless vehicles at intersections," in *10th ITS World Congress Vienna, Austria*, October 2012.

[31] J. Treat, N. Castellan, R. Stansifer, R. Mayer, R. Hume, D. Shinar, S. McDonald, and N. Tumbas, *Tri-level Study of the Causes of Traffic Accidents: Final Report. Volume I: Causal Factor Tabulations and Assessments*. 1977.

[32] NCSA, "National center for statistics and analysis, traffic safety facts 2003," tech. rep., U.S. DOT, Washington, DC, 2004.

[33] V. Hirankitti and J. Krohkaew, "An agent approach for intelligent traffic-light control," in *Modelling Simulation, 2007. AMS '07. First Asia International Conference on*, pp. 496 –501, march 2007.

[34] K. Dresner and P. Stone, "Mitigating catastrophic failure at intersections of autonomous vehicles," in *AAMAS Workshop on Agents in Traffic and Transportation*, (Estoril, Portugal), pp. 78–85, May 2008.

[35] S. Reveliotis and E. Roszkowska, "On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems," *Automatic Control, IEEE Transactions on*, vol. 55, no. 7, pp. 1646–1651, 2010.

[36] J. Gregoire, S. Bonnabel, and A. de La Fortelle, "Introducing dynamic constraints in priority-based intersection management," in *52nd IEEE Conference on Decision and Control*, 2013.

[37] J. Van Den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.