Time-optimal Coordination of Mobile Robots along Specified Paths

Florent Altché, Xiangjun Qian, Arnaud de la Fortelle

Time-optimal Coordination of Mobile Robots along Specified Paths

Florent Altché[1,2], Xiangjun Qian[1] and Arnaud de La Fortelle[1]

***Abstract*— In this paper, we address the problem of time-optimal coordination of mobile robots under kinodynamic constraints along specified paths. We propose a novel approach based on time discretization that leads to a mixed-integer linear programming (MILP) formulation. This problem can be solved using general-purpose MILP solvers in a reasonable time, resulting in a resolution-optimal solution. Moreover, unlike previous work found in the literature, our formulation allows an exact linear modeling (up to the discretization resolution) of second-order dynamic constraints. Extensive simulations are performed to demonstrate the effectiveness of our approach.**

I. Introduction

The deployment of autonomous mobile robots is expected to bring major benefits in many applications, and their number is likely to grow dramatically in the next decades. Therefore, the need to coordinate these robots, which means finding a way for each of them to reach its target without colliding with another robot, will become increasingly important. A vast literature on motion planning, which generalizes the problem of coordination, already exists (see *e.g.* [1]).

We consider the problem of optimally coordinating multiple robots along specified paths with variable speed under kinodynamic constraints. The simplifying fixed path assumption is notably suited for structured environments, for instance traffic intersections or warehouses, where robots are generally bound to navigate inside lanes or aisles. Various methods exist to quickly find a *feasible* solution to this problem, and many of these (see *e.g.* [2]) make use of the so-called coordination (or configuration) space introduced in [3]. However, the *optimal* coordination problem is known to be NP-hard [4], even in the absence of dynamic constraints, and few methods exist to provide a good solution with a high number of robots. As shown in [5], this algorithmic complexity stems from the implicit decision of choosing which of any two potentially colliding robots should be the first to pass, making the problem inherently combinatorial with complexity scaling as high as $2^{N(N-1)/2}$ for N robots. Exhaustive enumeration could be used for small instances (see *e.g.* [6], [7]), but would be difficult to scale up with a larger number of robots.

A possible way of handling this complexity is to prune the corresponding decision tree by removing provably non-optimal branches. Mixed-integer programming (MIP) is a widely-used framework that allows efficient handling of such combinatorial problems. General MIP problems involving arbitrary functions are very hard, but good techniques exist for a subclass of these problems, called mixed-integer second-order cone programming, or MISOCP [8]. In these problems, a convex quadratic objective function is minimized with quadratic positive semi-definite or affine constraints. A better-known subclass of MISOCP is mixed-integer linear programming (MILP), where the objective function and constraints are linear. These techniques have already been applied to trajectory planning in general, and to the coordination problem in particular.

In [9], a MILP formulation is used to compute fuel-optimal trajectories for multiple spacecrafts, while avoiding collisions and exhaust gases from other crafts. However, this model is inherently different from ours as paths are not specified in advance in this formulation, leading to a much higher computational complexity. On the opposite end of the spectrum, Wang et al. [10] use MILP to find an optimal velocity profile under piecewise-constant dynamic constraints for a train on a fixed path, but do not consider conflicts with other trains.

In [11], a mixed-integer nonlinear program is used to ensure safe separation of aircrafts evolving along fixed paths, with minimal deviation from an original flight plan. Even though dynamic constraints are not considered, nonlinearities stemming from the euclidean distance constraints render the problem very hard to solve in reasonable time, with a practical limit of 4 aircrafts.

Peng and Akella [12] consider a problem almost identical to ours. First, they propose to discretize robot paths into "collision-free" and "conflicting" segments. In a second step, by identifying the time instants when robots can enter and exit collision segments, they formulate collision avoidance constraints for every pair of robots into a nonlinear mixed-integer problem. However, the dynamic constraints are non-convex in this formulation, rendering the problem unsuitable for general-purpose solvers. To handle this difficulty, additional constraints are introduced to transform the problem into two linear subproblems that are solved as MILPs and give an upper and lower bound for the solution, but the actual optimal value is only approximated.

The problem of non-convexity of the dynamic constraints is also encountered in [7], where authors propose a convexification method using linearization along a reference speed. However, this approximation underestimates the maximum and overestimates the minimum acceleration, leading to a sub-optimal solution.

Our main contribution in this paper is the introduction of

[1] MINES ParisTech, PSL Research University, Centre for robotics, 60 Bd St Michel 75006 Paris, France [florent.altche, xiangjun.qian, arnaud.de_la_fortelle]@mines-paristech.fr

[2] École des Ponts ParisTech, Cité Descartes, 6-8 Av Blaise Pascal, 77455 Champs-sur-Marne, France

This work has received support from the European FP7 project AutoNet2030 (Grant Agreement NO. 610542).

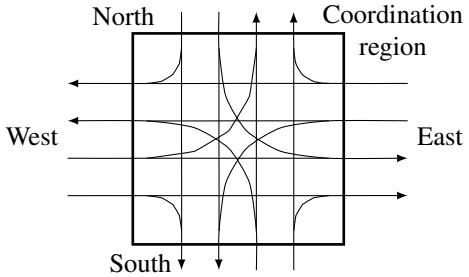


Fig. 1. Example of paths inside and outside the coordination region for a two-lane road intersection.

a new formulation of the optimal coordination problem as a MILP by discretizing over time instead of space, and using bounding polygons to transcribe safety requirements as linear constraints. This formulation allows an exact linear modeling of second-order dynamics up to the discretization resolution; solving this problem gives optimal velocity profiles for the robots under these constraints. For illustration purposes, we use the mean sojourn time as the minimization objective, but other linear optimization criteria could also be chosen.

The paper is articulated as follows. In Section II, we describe the modeling of the system of robots and the general coordination problem, and we give some theoretical insights on the problem from previous work in Section III. In Section IV, we present our formulation of the time-optimal coordination problem as a MILP problem. This formulation is then validated using computer simulation on the example of automated vehicles in an intersection, as presented in Section V. Finally, Section VI concludes the study.

II. Problem statement

We consider a set $\mathcal{N}$ of N robots evolving on predetermined paths, and we assume that coordination between robots is only needed inside a bounded region, which we call the *coordination region*. In the case of automated driving, for instance, coordination is mostly needed in the middle of road intersections, as illustrated in Fig. 1, while vehicles only need to keep a safe distance with the vehicle in front of them when they are not in the intersection. In this example, the coordination region would be chosen as the center of the intersection, including a portion of the roads leading to, and exiting from this center. Note that the predefined path assumption allows to only consider longitudinal movement of the robots, thus reducing the computational complexity, and is classical in robots coordination problems in a constrained environment [12], [13].

Each robot $i \in \mathcal{N}$ is supposed to follow a predetermined path γ_i inside the coordination region, so that we need only consider the longitudinal comportment of the robots. The dynamics of robot i are described as a double integrator:

$$(\dot{s}_i, \dot{v}_i) = (v_i, a_i) \tag{1}$$

where s_i is the curvilinear position of robot i along its path, v_i its longitudinal velocity and a_i its acceleration.

The origin of s_i is chosen so that $s_i = 0$ when the front of the robot enters the coordination region, and $s_i = s_i^{out} > 0$ when it fully exits the coordination region. s_i can therefore be interpreted as the distance traveled by robot i inside the coordination region. The velocity is assumed to be non-negative and bounded, such that $v_i \in [0, \overline{v}_i]$. The state of robot i is noted $x_i = (s_i, v_i)$; we call *trajectory* of robot i the (continuous) function mapping a given time t to the state of i at time t, noted $x_i(t) = (s_i(t), v_i(t))$. Boldface $\boldsymbol{x}$ denotes the vector $(x_1, x_2, ...)$, representing the state of the multi-robot system and $\boldsymbol{x}(t)$ is the system trajectory. To account for the dynamic constraints on the robots, we assume that the longitudinal acceleration a_i of robot i is bounded to an interval $\left[\underline{a}_i, \overline{a}_i\right]$ with $\underline{a}_i < 0 < \overline{a}_i$.

We assume that robot i enters the coordination region at time t_i^{in} with speed $v_i^{in} \in [0, \overline{v}_i]$, and is required to leave the coordination region with speed v_i^{out}; we let t_i^{out} denote the corresponding exit time. Note that v_i^{out} should be properly chosen to avoid collisions outside of the coordination region, for instance when a fast vehicle exits after a slower one.

For a pair of distinct robots i and j, we call *collision set* between i and j, noted $\mathcal{C}_{ij} \subset [0, s_i^{out}] \times \left[0, s_j^{out}\right]$ the set of positions (s_i, s_j) inside the coordination region where i and j would collide; we say that i and j are *conflicting* if $\mathcal{C}_{ij} \neq \emptyset$.

To simplify the rest of the presentation, we assume that the collision set between two robots is connected, *i.e.* we exclude the case of non-conflicting segments between two conflict segments, for instance when two paths intersect multiple times. If this is not the case, the presented results still hold provided every connected component $\mathcal{C}_{ij}^p$ of $\mathcal{C}_{ij}$ is considered individually. Moreover, we approximate the exact collision set by a (minimal) bounding polygon with edges either parallel to the horizontal or vertical axis, or to the $s_i = s_j$ line. Under our hypotheses, four possible types of conflicts can exist: robots can follow one another throughout the coordination region (a), or have crossing (b), merging (c) or diverging (d) paths, as illustrated in Fig. 2. Note that cases (c) and (d) are not mutually exclusive, and paths can merge and then diverge.

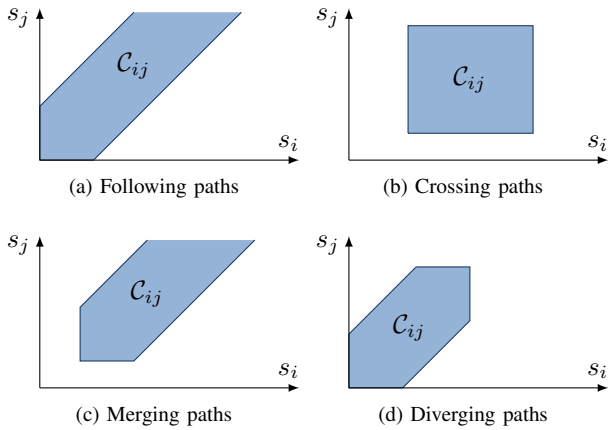
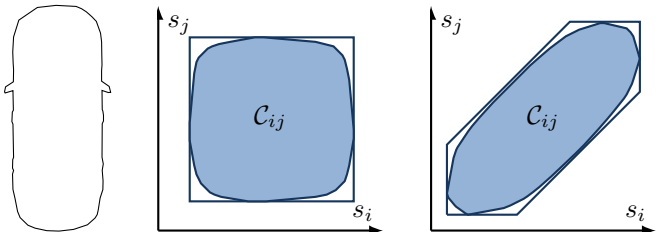


Fig. 2. Possible cases for the collision set between two conflicting rectangular robots i and j (assuming paths intersect orthogonally). The edges of $\mathcal{C}_{ij}$ are straight lines with equations $s_i = A$, $s_j = B$ or $|s_i - s_j| = C$ where A, B, C are constants.

The above bounding polygon approximation is well suited

when robots have nearly rectangular shapes, and induces a negligible loss of optimality in this case, as illustrated in Fig. 3. Note that arbitrary convex polygons can also be used, at the cost of introducing additional binary variables.



(a) Robots shape (b) Crossing paths (c) Merging and diverging paths

Fig. 3. Exact shape of the collision region $\mathcal{C}_{ij}$ for (polygonal) car-like robots. The blue rectangles represent the corresponding bounding polygon.

We now define the time-optimal coordination problem as:

Definition 1: The *time-optimal coordination problem* is that of finding the optimal system trajectory $\boldsymbol{x}^*(t)$ minimizing the mean sojourn time $\frac{1}{N}\sum_{i\in\mathcal{N}}(t_i^{out}-t_i^{in})$, under the following constraints for every robot i:

- initial conditions: $x_i(t_i^{in}) = (0, v_i^{in})$
- kinematics: $\dot{s_i} = v_i \in [0, \overline{v}_i]$
- dynamics: $\dot{v_i} = a_i \in [\underline{a}_i, \overline{a}_i]$
- safety: for all $t \geq t_i^{in}$ and for every robot $j \neq i$, $(s_i(t), s_j(t)) \notin \mathcal{C}_{ij}$
- liveness: there exists $t_i^{out} < +\infty$ with $s_i(t_i^{out}) = s_i^{out}$
- exit speed: $v_i(t_i^{out}) = v_i^{out}$.

Note that in the above definition, the initial condition $x_i(t_i^{in}) = (0, v_i^{in})$ corresponding to a fixed entry time can be replaced by a fixed initial state constraint $x_i(0) = (s_i^0, v_i^0)$.

III. Theoretical analysis

For an arbitrary pair of robots with non-empty collision set, one necessarily passes before the other to avoid collisions. For two conflicting robots i and j and a given collision-free system trajectory, we say that i has priority over j if i goes before j, and we note $i \succ j$ in this case. The collection of all priorities for all pairs of robots can be encoded as a priority graph where robots are the nodes and priorities are directed edges. It has been shown in [5] that such priority graphs can be bijectively mapped to homotopy classes of trajectories for the multi-robot system. For a given continuous optimization criterion, there exists at least one optimal trajectory in any non-empty homotopy class. The global optimal trajectory $\boldsymbol{x}^*(t)$ can then be found by enumerating all optimal trajectories for all priority graphs.

Therefore, the time-optimal coordination problem has both a discrete (enumerating all priority graphs) and a continuous part (optimizing the trajectories of every robots under assigned priorities, which is a continuous optimal control problem). The discrete part of the problem is combinatorial since, for N robots inducing p pairs of conflicts, there are up to 2^p possible priority graphs. In general, p can be as high as $\frac{1}{2}N(N-1)$; however, many of these graphs can be discarded for poor performance or for being incompatible with the robots entry times. For this reason, *branch-and-bound* algorithms seem particularly well-suited to our problem as they are designed to find global optima without needing to explore the whole decision tree.

IV. Time-optimal coordination

Using the above theoretical results, we formulate the time-optimal coordination problem as a mixed-integer *linear* program (MILP) which can be solved by widely-available solvers using branch-and-bound techniques.

A. Completed Collision Set

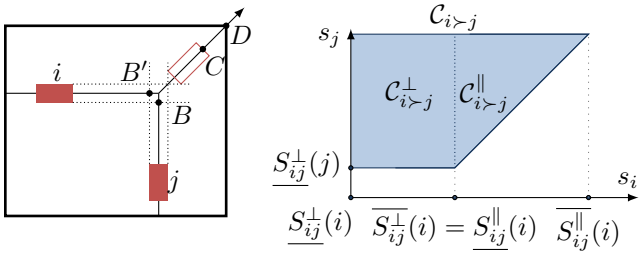
For conflicting robots i and j, we call *completed collision set* $\mathcal{C}_{i\succ j}$ the set of configurations (s_i, s_j) leading to a collision or a violation of priority $i \succ j$; mathematically, $\mathcal{C}_{i\succ j} = \mathcal{C}_{ij} + (\mathbb{R}_- \times \mathbb{R}_+)$. Since we have approximated each collision set $\mathcal{C}_{ij}$ by a minimal bounding polygon with edges parallel to the coordinate axes or to the $s_i = s_j$ lines, the completed set $\mathcal{C}_{i\succ j}$ is also a polygon with the same properties. Note that if $\mathcal{C}_{ij}$ is not connected, a completed collision set has to be defined for each connected component.

To formulate safety requirements as linear constraints, we first define a partition of $\mathcal{C}_{i\succ j}$ as $\mathcal{C}^{\parallel}_{i\succ j} \cup \mathcal{C}^{\perp}_{i\succ j}$. $\mathcal{C}^{\parallel}_{i\succ j}$ is the subset of $\mathcal{C}_{i\succ j}$ with boundary parallel to $s_i = s_j$, and $\mathcal{C}^{\perp}_{i\succ j}$ the subset with boundary parallel to the s_i axis as illustrated for the merging case in Fig. 4. Subset $\mathcal{C}^{\parallel}_{i\succ j}$ is the set of priority violations (or collisions) that could happen when j should follow i, and $\mathcal{C}^{\perp}_{i\succ j}$ is that of violations (or collisions) that could happen when j should wait for i to pass. In what follows, we note $\underline{S}^{\parallel}_{ij}(\cdot)$ and $\overline{S}^{\parallel}_{ij}(\cdot)$ the lower and upper bounds of the projection of $\mathcal{C}^{\parallel}_{i\succ j}$ on the $s_\cdot$ axis (for $\cdot = i$ or j), and we define $\underline{S}^{\perp}_{ij}(\cdot)$ and $\overline{S}^{\perp}_{ij}(\cdot)$ similarly. However, to ensure those subsets have empty intersection, we exclude $\left\{\left(\overline{S}^{\perp}_{ij}(i), s_j\right) \;:\; s_j \in [0, s_j^{out}]\right\}$ from $\mathcal{C}^{\perp}_{i\succ j}$. If either $\mathcal{C}^{\parallel}_{i\succ j}$ or $\mathcal{C}^{\perp}_{i\succ j}$ is empty, we let the corresponding lower and upper bounds be equal to 0; note that in this particular case, those subsets formally do not form a partition.

An important remark is that $\mathcal{C}^{\parallel}_{i\succ j}$ and $\mathcal{C}^{\perp}_{i\succ j}$ are both connected and left invariant by translation of a vector from $\{0\} \times \mathbb{R}_+$. Therefore, with this decomposition, if $\underline{S}^{\parallel}_{ij}(i) \leq s_i \leq \overline{S}^{\parallel}_{ij}(i)$, condition $(s_i, s_j) \notin \mathcal{C}_{i\succ j}$ is equivalent to $s_j \leq s_i - a_{ij}$ where a_{ij} is a constant corresponding to a following distance and a potential offset of curvilinear abscissa. If $\underline{S}^{\perp}_{ij}(i) \leq s_i < \overline{S}^{\perp}_{ij}(i)$, condition $(s_i, s_j) \notin \mathcal{C}_{i\succ j}$ is equivalent to $s_j \leq \underline{S}^{\perp}_{ij}(j)$. If s_i is not in either of these intervals, collisions or priority violations cannot occur. Note that, if needed, a finer decomposition could be used to better approximate the exact shape of the collision set, by using bounding convex polygons with edges having slopes different than 0 or 1, and defining one subset of $\mathcal{C}_{i\succ j}$ per edge.

B. Discretization

If priorities have been chosen, finding the best trajectories in continuous time and space is equivalent to a nonlinear time-optimal control problem in a non-convex space, which



(a) Merging paths (b) Corresponding collision set

Fig. 4. Illustration of the completed collision set $\mathcal{C}_{i\succ j}$ in the merging case, and the corresponding values of $\underline{S}^{\perp}_{ij}$, $\overline{S}^{\perp}_{ij}(i)$, $\underline{S}^{\parallel}_{ij}(i)$ and $\overline{S}^{\parallel}_{ij}(i)$. $s_j = \underline{S}^{\perp}_{ij}(j)$ when j reaches B in Fig. 4a, and $s_i = \underline{S}^{\parallel}_{ij}(i)$ when i reaches C. Similarly, $s_i = \underline{S}^{\perp}_{ji}(i)$ when i reaches B'. Also note that the equation for the lower-right boundary of the collision set is $s_i \geq s_j + \underline{S}^{\parallel}_{ij}(i) - \underline{S}^{\perp}_{ij}(j)$.

is usually difficult to solve. Several authors, among which [12] and [7], have used spatial discretization to overcome this difficulty and formulate a simpler problem. In this setting, paths are divided into segments and the average speed v_{avg} and occupancy time t_{occ} for each robot in each segment are chosen as variables. One of the major issues arising from this discretization is that the second-order dynamics of the robot imply nonlinear (and non-convex) relations between the length ℓ of a segment and these two variables. Indeed, these constraints are written as $\ell = v_{avg} t_{occ}$ which translates to $\ell \leq v_{avg} t_{occ} \wedge \ell \geq v_{avg} t_{occ}$; one of those inequalities defines a non-convex set in the (v_{avg}, t_{occ}) plane that has to be approximated in order to be treated in most optimization frameworks, thus degrading solution quality.

Instead of using spatial segments, we propose a temporal discretization which, to the best of our knowledge, has not yet been used to solve the coordination problem with a fixed-paths assumption. In what follows, we note $\tau > 0$ the fixed duration of a time step; for an integer $k \geq 0$ and a robot i, we note $s_i^k = s_i(k\tau)$, and use similar notations for v_i. We note K the maximum number of time steps in the computation.

C. Variables

Besides variables s_i^k and v_i^k, we need to introduce a few supplementary variables, all of them binary. For two conflicting robots i and j, we let $\pi_{ij} = 1$ if and only if (iff) $i \succ j$, and for all time steps k we define several variables ε_{ij} that indicate if, at time step k, robots i and j have entered and/or exited $\mathcal{C}^{\perp}_{i\succ j}$ and $\mathcal{C}^{\parallel}_{i\succ j}$.

Specifically, we let $\varepsilon^{\bullet,in}_{ij}(i,k) = 1$ iff $s_i^k \geq \underline{S}^{\bullet}_{ij}(i)$ and $\varepsilon^{\bullet,out}_{ij}(i,k) = 1$ iff $s_i^k \geq \overline{S}^{\bullet}_{ij}(i)$, where $\bullet$ is either $\parallel$ or $\perp$, and we define similarly $\varepsilon^{\bullet,in}_{ij}(j,k) = 1$ iff $s_j^k \geq \underline{S}^{\bullet}_{ij}(j)$ and $\varepsilon^{\bullet,out}_{ij}(j,k) = 1$ iff $s_j^k \geq \overline{S}^{\bullet}_{ij}(j)$.

We also introduce, for every robot, the variables $\mu_i^k = 1$ iff $s_i^k \geq 0$ and $\sigma_i^k = 1$ iff $s_i^k \geq s_i^{out}$, respectively indicating if the robot has entered and exited the coordination region at time step k.

D. Objective function

As stated earlier, we wish to minimize the average sojourn time, which is equivalent (since entry times are prescribed) to minimizing the average exit times. Since our formulation does not use time as a variable, we use

$$\mathcal{O} = \frac{1}{N} \sum_{\substack{i=1\dots N \\ k=0\dots K}} \sigma_i^k \tag{2}$$

for our objective function. $\mathcal{O}$ is the average number of time steps spent after exiting the coordination region, which approximates (to the duration of a time step τ) the total amount of time spent after exiting the coordination region, divided by τ. More precisely, $\mathcal{O} = \frac{1}{N} \sum_{i=1\dots N} (K - k_i^{out} + 1)$ where k_i^{out} is the time step at which robot i exits the coordination region, and thus maximizing $\mathcal{O}$ is equivalent to minimizing the average value of k_i^{out}.

E. Constraints

Many of the constraints needed for the coordination problem are conjunctions (noted $\wedge$), or logical implications ($\Rightarrow$); we use the binary variables introduced in the previous section as indicators, and use a "big-M" formulation [14] to enforce those constraints. In what follows, left-hand side terms are variables, and right-hand side terms are problem parameters.

a) Binary variables: To ensure the additional binary variables are set to the correct value, the following helper conditions are enforced for all $0 \leq k \leq K$, every robot $i \in \mathcal{N}$ and every robot $j \in \mathcal{N}$ conflicting with i:

$$\mu_i^k = 0 \Rightarrow s_i^k \leq 0 \tag{h1}$$
$$\mu_i^k = 1 \Rightarrow s_i^k \geq 0 \tag{h2}$$
$$\sigma_i^k = 0 \Rightarrow s_i^k \leq s_i^{out} \tag{h3}$$
$$\sigma_i^k = 1 \Rightarrow s_i^k \geq s_i^{out} \tag{h4}$$
$$\pi_{ij} + \pi_{ji} = 1 \tag{h5}$$

Constraints similar to (h1)-(h2) are used for $\varepsilon^{\bullet,in}_{ij}(i,k)$, $\varepsilon^{\bullet,in}_{ij}(j,k)$, $\varepsilon^{\bullet,out}_{ij}(i,k)$ and $\varepsilon^{\bullet,out}_{ij}(j,k)$, where $\bullet$ is either $\perp$ or $\parallel$. Constraint (h5) ensures that exactly one of the variables π_{ij} and π_{ji} is set to 1. Note that strict inequalities cannot be enforced in a MILP framework; therefore, the above constraints do not specify the value of each indicator variables at its point of discontinuity. This limitation, however, has little effect on the solutions.

b) Initial and terminal values, bounds: To account for the initial values of the variables and the different bounds on the problem, we use the following boundary constraints for all $0 \leq k \leq K-1$ and each robot i:

$$\mu_i^k = 0 \Rightarrow v_i^{k+1} = v_i^{in} \tag{b1}$$
$$\sigma_i^{k+1} = 1 \Rightarrow v_i^k = v_i^{out} \tag{b2}$$
$$s_i^0 = -v_i^{in} t_i^{in} \tag{b3}$$
$$s_i^K \geq s_i^{out} \tag{b4}$$
$$v_i^0 = v_i^{in} \tag{b5}$$
$$v_i^k \in [0; \overline{v}_i] \tag{b6}$$

Conditions (b1), (b3) and (b5) enforce the initial constraint $x_i(t_i^{in}) = (0, v_i^{in})$, (b2) the final speed constraint, (b6) enforces the bounds on speed and (b4) ensures the liveness constraint, as all robots are required to exit in finite time.

c) Kinodynamic constraints: We assume that robots use a constant acceleration during each time step. Under this assumption, we enforce the kinodynamic constraints using the conditions, for all $0 \leq k \leq K-1$ and every robot i:

$$\sigma_i^k = 0 \Rightarrow s_i^{k+1} - s_i^k - \frac{1}{2}\tau\left(v_i^{k+1} + v_i^k\right) = 0 \quad \text{(k1)}$$

$$\sigma_i^k = 0 \Rightarrow v_i^{k+1} - v_i^k \leq \overline{a}_i\tau \quad \text{(k2)}$$

$$\sigma_i^k = 0 \Rightarrow v_i^{k+1} - v_i^k \geq \underline{a}_i\tau \quad \text{(k3)}$$

Condition (k1) enforces the kinematic constraints (under the constant acceleration assumption); (k2) and (k3) account for the dynamic constraints. Note that the constant acceleration hypothesis could be relaxed using third-order dynamics, or more general second-order cone programming, as

$$x_i^k - v_i^k\tau + \frac{\overline{a}_i\tau}{\underline{a}_i - \overline{a}_i}y_i^k - \frac{1}{2(\underline{a}_i - \overline{a}_i)}\,{y_i^k}^2 \leq \frac{\overline{a}_i\,\underline{a}_i\tau^2}{2(\underline{a}_i - \overline{a}_i)}$$

$$x_i^k - v_i^k\tau + \frac{\underline{a}_i\tau}{\overline{a}_i - \underline{a}_i}y_i^k - \frac{1}{2(\overline{a}_i - \underline{a}_i)}\,{y_i^k}^2 \geq \frac{\overline{a}_i\,\underline{a}_i\tau^2}{2(\overline{a}_i - \underline{a}_i)}$$

where $x_i^k = s_i^{k+1} - s_i^k$ and $y_i^k = v_i^{k+1} - v_i^k$. A justification for this extension can be found in [15] which shows that these constraints exactly describe the set of reachable positions and speeds under second-order integrator dynamics with bounded acceleration in a given time τ.

d) Safety constraints: Using the previously-defined indicator variables ε and the results from Section IV-A, we translate the safety constraints as follows: for every pair of conflicting robots (i,j) and for all $0 \leq k \leq K-1$:

$$\pi_{ij} = 1 \wedge \varepsilon_{ij}^{\perp,out}(i,k) = 0 \Rightarrow \varepsilon_{ij}^{\perp,in}(j,k+1) = 0 \quad \text{(s1)}$$

$$\pi_{ij} = 1 \wedge \varepsilon_{ij}^{\parallel,in}(j,k) = 1 \wedge \varepsilon_{ij}^{\parallel,out}(i,k) = 0 \Rightarrow s_i^{k+1} - s_j^{k+1} \geq a_{ij} \quad \text{(s2)}$$

$$\pi_{ij} = 1 \wedge \varepsilon_{ij}^{\parallel,in}(j,k) = 1 \wedge \varepsilon_{ij}^{\parallel,out}(i,k) = 0 \Rightarrow s_i^{k+1} - s_j^{k+1} + \frac{\tau}{2}\left(v_i^{k+1} - v_j^{k+1}\right) \geq a_{ij} \quad \text{(s3)}$$

with $a_{ij} = d_\parallel + \underline{S}_{ij}^{\parallel}(i) - \underline{S}_{ij}^{\perp}(j)$, in which $d_\parallel$ is a following distance (from front of the follower to rear of the leader) between two robots. The term $\underline{S}_{ij}^{\parallel}(i) - \underline{S}_{ij}^{\perp}(j)$ in a_{ij} accounts for the potential offset in curvilinear position between two robots in the case of merging paths (and vanishes in other cases), as illustrated in Fig. 4. Condition (s1) can be phrased as "if i has priority and has not yet passed the collision set, then j cannot go in". The additional condition (s3) is used to ensure that no collision occurs between two time steps; for the same reason, (s1) involves $\varepsilon_{ij}^{\perp,out}(i,k)$ and $\varepsilon_{ij}^{\perp,in}(j,k+1)$. This approach can be seen as a systematization of the concept of spatio-temporal trajectory envelopes presented in [13], [16], where the temporal extent of these envelopes is automatically adjusted during resolution.

Note that the above formulation can be used in the case of a collision set with multiple connected components, by introducing a set of variables π and ε, and of parameters $\underline{S}$ and $\overline{S}$ for each of these components. This method allows simultaneous resolution over multiple conflict areas, thus ensuring the solution is a global optimum and not comprised of several local optima, which would be the case if each connected component was treated separately.

F. Optimization problem

To simplify notations, we note X the tuple of all the variables described above. The optimization problem (in which indices have been omitted for readability) of finding

$$\max_X \quad \mathcal{O}(X) \qquad (3)$$
$$\text{s.t.} \quad \text{(h1)} - \text{(h5)},\ \text{(b1)} - \text{(b6)},\ \text{(k1)} - \text{(k3)},\ \text{(s1)} - \text{(s3)}$$

either gives a solution to the discretized time-optimal coordination problem, or is infeasible. Specifically, infeasibilities can either be caused by the choice of a too small value for the number of time steps K, or because the initial states of the robots do not allow a safe passage through the intersection.

The sub-optimality caused by the time discretization vanishes as the time step goes to zero. Since the limit on computation time effectively sets a lower bound on the time step duration, it is desirable to choose a value providing good quality solutions in reasonable time. This issue will be discussed in Section V. However, an additional artifact arises from time discretization with a finite resolution, as objective function $\mathcal{O}$ does not distinguish solutions with sojourn times differing by less than the duration of a time step for at least one robot. To correct this issue, we make use of the fact that maximizing the speed of a robot allows to minimize its (non-discretized) sojourn time. Therefore, adding an "averaged normalized speed" term $\frac{1}{NK}\sum_{i,k}\frac{v_i^k}{\overline{v}_i}$ to function $\mathcal{O}$ allows to choose, among all solutions of (3), the one with highest average speed and thus smallest average sojourn time. Note that the weighting of the added term ensures this solution is still optimal for problem (3), and the modified objective function has been used in our simulations.

Moreover, our formulation could also be used for continuous arrivals of robots in real-time. Let T be an upper bound for the computation time for N_T robots: at a given time t we consider the set $\mathcal{N}_t$ of robots entering the coordination region between times t and $t+T$, and we assume $|\mathcal{N}_t| \leq N_T$. If optimal trajectories have been assigned for the robots of $\mathcal{N}_{t-T}$ and taking those as constraints for the robots of $\mathcal{N}_t$, we can compute optimal trajectories for those robots before they reach the entry of the coordination region. Constraint (b1) ensures those trajectories remain feasible by that time and can therefore be assigned to the robots of $\mathcal{N}_t$. Note that this time-receding method may, however, cause sub-optimalities compared to considering all robots at once.

Also note that optimizing robots speed profiles for minimum sojourn time comes in pair with reducing safety margins to a minimum. For actual applications, this may cause problems in the case of unexpected events such as failure of a robot, which could make other robots unable to avoid a collision. Future work will study how to balance efficiency with the ability to cope with contingencies, to improve the robustness of the coordination.

V. Simulation results

The use of the above optimization problem to find a time-optimal coordination has been validated by computer simulation on the example of autonomous vehicles in the intersection of Fig. 1. The simulation is based on the free traffic modeling tool SUMO [17] and uses its path generation algorithm to compute collisions sets.

Vehicles are generated either deterministically (first simulation), or using random Poisson arrival times with normally-distributed entry speeds truncated to a minimum and a maximum speed (second and third simulations). Optimization problem (3) (using the modified objective function) is then run into the commercial MILP solver Gurobi [18], using its Python interface to generate the constraints. Lastly, if the problem is feasible, the solution trajectories are simulated in SUMO using the TraCI interface to verify that they do not generate collisions.

In all simulations, vehicles are modeled as rectangles of $5\,\mathrm{m}$ length by $2\,\mathrm{m}$ width, with $[\underline{a}_i, \overline{a}_i] = [-3, +4]\ \mathrm{m\,s^{-2}}$. The exit speed for all $i \in \mathcal{N}$ is set as $v_i^{out} = \overline{v} = 15\,\mathrm{m\,s^{-1}}$, which ensures the absence of collisions outside of the coordination region. The entry speed v_i^{in} is deterministically chosen in the first simulation and is normally distributed with average $12\,\mathrm{m\,s^{-1}}$ and standard deviation $3\,\mathrm{m\,s^{-1}}$, truncated to $[10, 15]\ \mathrm{m\,s^{-1}}$ in the second and third simulations.

Simulations were performed on a personal computer running on a $3.60\,\mathrm{GHz}$ Intel Core i7-4790 CPU with $16\,\mathrm{GB}$ of RAM. A replay of some of our simulations is available in the accompanying video submission[1].

A. Microscopic simulation

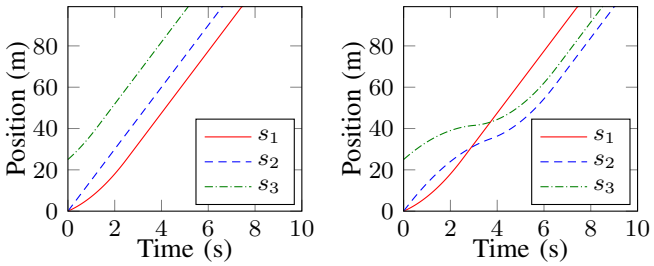
We first demonstrate the ability of our method to find the global optimum on a simple example with three vehicles in the case of the intersection displayed in Fig. 1: vehicle 1 goes from south to west, vehicle 2 from west to east and vehicle 3 from north to south. Vehicles initially start at $(s_1^0, s_2^0, s_3^0) = (0, 0, 25)\ \mathrm{m}$ with speeds $(v_1^{in}, v_2^{in}, v_3^{in}) = (5, 15, 10)\ \mathrm{m\,s^{-1}}$. Fig. 5a shows the globally optimal trajectories for each vehicle, which lies in the homotopy class represented by priorities $3 \succ 2$, $2 \succ 1$, $3 \succ 1$. For comparison purposes, Fig. 5b shows the (locally) optimal trajectories when sub-optimal priorities $1 \succ 3$, $3 \succ 2$, $1 \succ 2$ are enforced. The optimum average sojourn time found by our algorithm is $6.5\,\mathrm{s}$, and the example sub-optimal one is $8.4\,\mathrm{s}$.

B. Influence of time step duration

The discretization time step has a double effect on the solution: first, we assume constant acceleration during one time step. Second, the safety constraints require that one robot of each conflicting pair leaves the conflict area one time step before the other can enter.

In Fig. 6 we show the average optimality loss caused by choosing larger time step durations for a random set of 85 initial configurations of 15 vehicles. For each instance, the resolution-optimal average time t_{opt}^{τ} is computed for time step durations τ ranging from $0.125\,\mathrm{s}$ to $5\,\mathrm{s}$. The relative loss of optimality is computed as $\frac{t_{opt}^{\tau} - t_{opt}^{0.125}}{t_{opt}^{0.125}}$. Interestingly, the averaged values fit closely to an affine function with slope 7.2% per second for the above set of parameters. The loss of optimality remains less than 6% when the time step is smaller than $1\,\mathrm{s}$; moreover, the solution of (3) converges as the time step duration vanishes.

The kinodynamic parameters, *i.e.* the maximum speed $\overline{v}$ and the acceleration bounds $\underline{a}_i$ and $\overline{a}_i$ influence the magnitude of the optimality loss. Fig. 7 shows a comparison of the losses of optimality for three different scenarios, namely "reference", "lower speed" and "higher acceleration". Reference parameters are, as before, $(\overline{v}, \underline{a}_i, \overline{a}_i) = (15\,\mathrm{m\,s^{-1}}, -3\,\mathrm{m\,s^{-2}}, 4\,\mathrm{m\,s^{-2}})$. The same set of 85 initial configurations is used for those three scenarios, and we



(a) Optimal priorities (b) Suboptimal priorities

Fig. 5. Optimal trajectories within given priority classes, corresponding to a global (left) and local (right) optimum

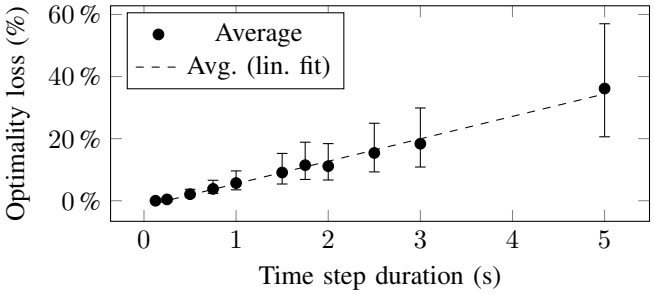


Fig. 6. Average relative optimality loss (compared to a $0.125\,\mathrm{s}$ time step), depending on time step duration, for 85 instances of 15 vehicles. Error bars correspond to 1 standard deviation for instances above or below average.

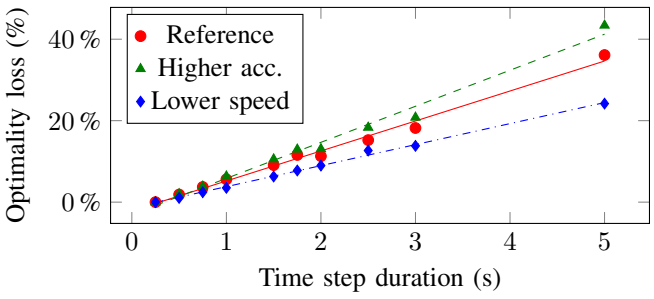


Fig. 7. Average relative optimality loss (compared to a $0.25\,\mathrm{s}$ time step), depending on time step duration, for 85 instances of 15 vehicles with reference parameters (red), lower maximum speed of $10\,\mathrm{m\,s^{-1}}$ (blue) and higher absolute values of accelerations $[\underline{a}_i, \overline{a}_i] = [-6, +8]\ \mathrm{m\,s^{-2}}$ (green).

[1]Also available at `https://youtu.be/RiW2OFsdHOY`

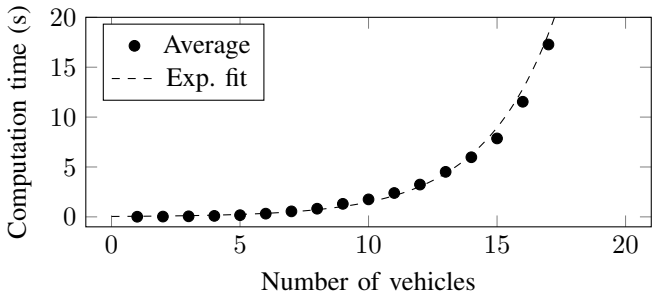


Fig. 8. Averaged computation time over 10 instances for a time step duration of 1 s and a varying number of vehicles.

only vary the kinodynamic parameters; in the lower speed scenario, initial speeds are also reduced by 33%. We find that an increase of time step duration causes higher losses of optimality in instances with more dynamic robots (higher speeds or higher absolute values of acceleration bounds) than those with less dynamic ones. As a result, the time step duration can be adapted to the dynamic characteristics of robots; for instance using longer time steps for slower robots.

C. Computation time

To measure the computation time required to solve problem (3) for the intersection displayed in Fig. 1, we run the simulator for different numbers of vehicles with a fixed time step of 1 s (Fig. 8). Computation time remains below 1 s for up to 8 vehicles, which would make our approach suitable for real-world applications. Note that the same 30 s time horizon is considered across all simulations to provide fair comparison, while a shorter horizon could be used for problems involving fewer robots, thus reducing computation time; moreover, the presented MILP problem has been formulated for clarity rather than execution speed, notably by introducing redundant variables. As a result, performance can likely be further improved to process more robots in the same time frame.

VI. Conclusion

This article presents a new approach to compute time-optimal trajectories for the coordination of mobile robots in a structured environment, taking into account kinodynamic constraints. Assuming each robot follows a predetermined path and using well-chosen bounding polygons, it is possible to formulate this problem as a mixed-integer linear program. By using temporal instead of spatial discretization, our approach allows an exact modeling of second-order integrator dynamics with piecewise-constant acceleration. The formulated problem mixes the discrete choice of relative priorities between conflicting robots with the continuous optimization of speed profiles respecting these priorities.

Extensive computer simulations have been run, on the example of a road intersection with autonomous vehicles, using the open-source traffic simulator SUMO and commercial MILP solver GUROBI. These simulations demonstrate the ability of our approach to compute optimal, collision-free trajectories in reasonable time.

Using consumer-grade hardware, the proposed method can treat up to 8 robots in less than a second. It is therefore suitable for real-time use, for instance for robots in an automated warehouse or autonomous vehicles at an intersection. In situations where computation time is more important than optimality, our formulation could also be used to design more efficient heuristics, with provable performance bounds, or to assess potentially dangerous situations where no safe trajectories exist. More general quadratic programming techniques could allow to use more complex objective functions, and relax the constant acceleration assumption while still providing an exact modeling of the dynamic constraints, opening many perspectives for future research.

References

[1] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

[2] T. Siméon, S. Leroy, and J. P. Laumond, "Path coordination for multiple mobile robots: A resolution-complete algorithm," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 42–49, 2002.

[3] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Transactions on Computers*, vol. 100, pp. 108–120, 1983.

[4] P. Dasler and D. M. Mount, "On the complexity of an unregulated traffic crossing," *CoRR*, vol. abs/1505.00874, 2015.

[5] J. Gregoire, "Priority-based coordination of mobile robots," *arXiv preprint*, vol. arXiv:1306.0785, 2013.

[6] H. Chitsaz, J. O'Kane, and S. LaValle, "Exact Pareto-optimal coordination of two translating polygonal robots on an acyclic roadmap," *IEEE International Conference on Robotics and Automation.*, pp. 3981–3986 Vol.4, 2004.

[7] N. Murgovski, G. R. de Campos, and J. Sjoberg, "Convex modeling of conflict resolution at traffic intersections," in *IEEE Conference on Decision and Control (CDC)*, Dec 2015, pp. 4708–4713.

[8] F. Alizadeh and D. Goldfarb, "Second-Order Cone Programming," *Math. Programming - Ser. B.*, vol. 95, no. 3 – 51, pp. 3–51, 2003.

[9] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 755–764, 2002.

[10] Y. Wang, B. De Schutter, B. Ning, N. Groot, and T. J. J. van den Boom, "Optimal trajectory planning for trains using mixed integer linear programming," in *IEEE Conference on Intelligent Transportation Systems (ITSC)*, oct 2011, pp. 1598–1604.

[11] S. Cafieri and N. Durand, "Aircraft deconfliction with speed regulation: New models from mixed-integer optimization," *Journal of Global Optimization*, vol. 58, no. 4, pp. 613–629, 2014.

[12] J. Peng, "Coordinating Multiple Robots with Kinodynamic Constraints Along Specified Paths," *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 295–310, apr 2005.

[13] F. Pecora, M. Cirillo, and D. Dimitrov, "On mission-dependent coordination of multiple vehicles under spatial and temporal constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, oct 2012, pp. 5262–5269.

[14] A. Richards and J. How, "Mixed-integer programming for control," *Proceedings of the American Control Conference*, pp. 2676–2683, 2005.

[15] J. Johnson and K. Hauser, "Optimal acceleration-bounded trajectory planning in dynamic environments along a specified path," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 2035–2041.

[16] M. Cirillo, F. Pecora, H. Andreasson, T. Uras, and S. Koenig, "Integrated Motion Planning and Coordination for Industrial Vehicles," in *International Conference on Automated Planning and Scheduling*, 2014, pp. 463–471.

[17] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.

[18] Gurobi Optimization, Inc., "Gurobi optimizer reference manual," http://www.gurobi.com, 2015.