ALGAMES: a fast augmented Lagrangian solver for constrained dynamic games

Simon Le Cleac'h[1] · Mac Schwager[2] · Zachary Manchester[3]

Abstract

Dynamic games are an effective paradigm for dealing with the control of multiple interacting actors. This paper introduces augmented Lagrangian GAME-theoretic solver (ALGAMES), a solver that handles trajectory-optimization problems with multiple actors and general nonlinear state and input constraints. Its novelty resides in satisfying the first-order optimality conditions with a quasi-Newton root-finding algorithm and rigorously enforcing constraints using an augmented Lagrangian method. We evaluate our solver in the context of autonomous driving on scenarios with a strong level of interactions between the vehicles. We assess the robustness of the solver using Monte Carlo simulations. It is able to reliably solve complex problems like ramp merging with three vehicles three times faster than a state-of-the-art DDP-based approach. A model-predictive control (MPC) implementation of the algorithm, running at more than 60 Hz, demonstrates ALGAMES' ability to mitigate the "frozen robot" problem on complex autonomous driving scenarios like merging onto a crowded highway.

Keywords Dynamic game · Nash equilibrium · Autonomous driving

1 Introduction

Controlling a robot in an environment where it interacts with other agents is a complex task. Traditional approaches in the literature adopt a predict-then-plan architecture. First,

✉ Simon Le Cleac'h
  simonlc@stanford.edu

  Mac Schwager
  schwager@stanford.edu

  Zachary Manchester
  zacm@cmu.edu

[1] Department of Mechanical Engineering, Stanford University, Stanford, USA

[2] Department of Aeronautics and Astronautics, Stanford University, Stanford, USA

[3] The Robotics Institute, Carnegie Mellon University, Pittsburgh, USA

predictions of other agents' trajectories are computed, then they are fed into a planner that considers them as immutable obstacles. This approach is limiting because the effect of the robot's trajectory on the other agents is ignored. Moreover, it can lead to the "frozen robot" problem that arises when the planner finds that all paths to the goal are unsafe (Trautman and Krause 2010). It is, therefore, crucial for a robot to *simultaneously* predict the trajectories of other vehicles on the road while planning its own trajectory, in order to capture the reactive nature of all the agents in the scene. ALGAMES provides such a joint trajectory predictor and planner by considering all agents as players in a Nash-style dynamic game. We envision ALGAMES being run on-line by a robot in a receding-horizon loop, at each iteration planning a trajectory for the robot by explicitly accounting for the reactive nature of all agents in its vicinity.

Joint trajectory prediction and planning in scenarios with multiple interacting agents is well-described by a dynamic game. Dealing with the game-theoretic aspect of multi-agent planning problems is a critical issue that has a broad range of applications. For instance, in autonomous driving, ramp merging, lane changing, intersection crossing, and overtaking maneuvers all comprise some degree of game-theoretic interactions (Sadigh et al. 2016a, b; Fridovich-Keil et al. 2020; Dreves and Gerdts 2018; Fisac et al. 2019; Schmer-
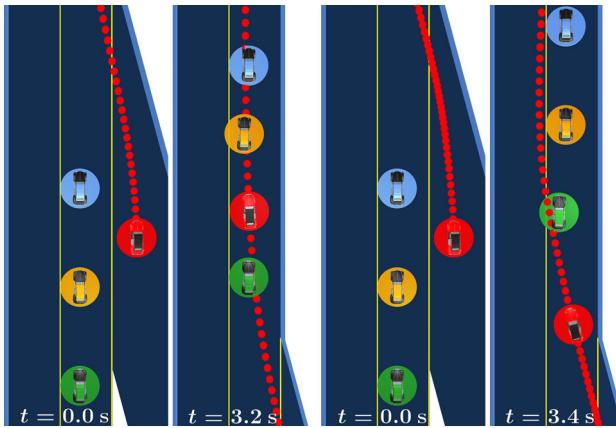
Fig. 1 A merging maneuver on a crowded highway is carried out using a receding horizon implementation of ALGAMES. On the left, the red car controlled by ALGAMES merges between the orange and green cars with minimal disruption to the traffic. On the right, the red car controlled by a non-game-theoretic MPC, is "frozen," i.e. it cannot find a feasible path. It has to slow down significantly and wait for the three cars to pass before merging

ling et al. 2018). Other potential applications include mobile robots navigating in crowds, like package-delivery robots, tour guides, or domestic robots; robots interacting with people in factories, such as mobile robots or fixed-base multi-link manipulators; and competitive settings like drone and car racing (Wang et al. 2021; Liniger and Lygeros 2019).

In this work, we seek solutions to constrained multi-player dynamic games. In dynamic games, the players' strategies are sequences of decisions. It is important to notice that, unlike traditional optimization problems, non-cooperative ga-mes have no "optimal" solution. Depending on the structure of the game, asymmetry between players, etc., different concepts of solutions are possible. In this work, we search for Nash-equilibrium solutions. This type of equilibrium models symmetry between the players; All players are treated equally. At such equilibria, no player can reduce its cost by unilaterally changing its strategy. For extensive details about the game-theory concepts addressed in this paper, we refer readers to the work of Basar and Olsder (1999).

Our solver is aimed at finding a Nash equilibrium for multi-player dynamic games, and can handle general nonlinear state and input constraints. This is particularly important for robotic applications, where the agents often interact through their desire to avoid collisions with one another or with the environment. Such interactions are most naturally represented as (typically nonlinear) state constraints. This is a crucial feature that sets game-theoretic methods for robotics apart from game-theoretic methods in other domains, such as economics, behavioral sciences, and robust control. In these domains, the agent interactions are traditionally represented in the objective functions themselves, and these games typically have no state or input constraints. In mathematics

literature, Nash equilibria with constraints are referred to as *Generalized Nash Equilibria* (Facchinei and Kanzow 2007). Hence, in this paper we present an augmented Lagrangian solver for finding Generalized Nash Equilibria specifically tailored to robotics applications.

Our solver assumes that players are rational agents acting to minimize their costs. This rational behavior is formulated using the first-order necessary conditions for Nash equilibria, analogous to the Karush–Kuhn–Tucker (KKT) conditions in optimization. By relying on an augmented Lagrangian approach to handle constraints, the solver is able to solve multi-player games with several agents and a high level of interactions at real-time speeds. Finding a Nash equilibrium for 3 autonomous cars in a freeway merging scenario takes 90 ms. Our primary contributions are:

1. A general solver for dynamic games aimed at identifying Generalized Nash Equilibrium strategies.
2. A real-time MPC implementation of the solver mitigating the "frozen robot" problem that arises in complex driving scenarios for non-game-theoretic MPC approach (Fig. 1).
3. An analysis of the non-uniqueness of Nash equilibria in driving scenarios with constraints and an assessment of the practical impact it has on players' coordination.
4. A comparison with iLQGames (Fridovich-Keil et al. 2020). ALGAMES finds Nash equilibria 3 times faster than iLQGames for a fixed constraint satisfaction criterion.

2 Related work

2.1 Equilibrium selection

Recent work focused on solving multi-player dynamic games can be categorized by the type of equilibrium they select. Several works Sadigh et al. (2016a, b), Liniger and Lygeros (2019), Yoo and Langari (2012) have opted to search for Stackelberg equilibria, which model an asymmetry of information between players. These approaches are usually formulated for games with two players: a leader and a follower. The leader chooses its strategy first, then the follower selects the best response to the leader's strategy. Alternatively, a Nash equilibrium does not introduce hierarchy between players; each player's strategy is the best response to the other players' strategies. As pointed out in Fisac et al. (2019), searching for open-loop Stackelberg equilibrium strategies can fail on simple examples. In the context of autonomous driving, for instance, when players' cost functions only depend on their own state and control trajectories, the solution becomes trivial. The leader ignores mutual collision constraints and the follower has to adapt to this strategy. This behavior can be overly aggressive for the leader (or

overly passive for the follower) and does not capture the game-theoretic nature of the problem.

Nash equilibria have been investigated in Fridovich-Keil et al. (2020), Dreves and Gerdts (2018), Wang et al. (2021), Britzelmeier et al. (2019), Di and Lamperski (2018, 2019, 2020). We also take the approach of searching for Nash equilibria, as this type of equilibrium seems better suited to symmetric, multi-robot interaction scenarios. Indeed, we have observed more natural behavior emerging from Nash equilibria compared to Stackelberg when solving for open-loop strategies.

2.2 Game-theoretic trajectory optimization

Most of the algorithms proposed in the robotics literature to solve for game-theoretic equilibria can be grouped into four types: First are algorithms aimed at finding Nash equilibria that rely on decomposition, such as Jacobi or Gauss-Siedel methods (Wang et al. 2021; Britzelmeier et al. 2019). These algorithms are based on an iterative best-response (IBR) scheme in which players take turns at improving their strategies considering the other agents' strategies as immutable (Facchinei and Kanzow 2007). This type of approach is easy to interpret and scales reasonably well with the number of players. However, convergence of these algorithms is not well understood (Facchinei and Kanzow 2007), and special care is required to capture the game-theoretic nature of the problem (Wang et al. 2021). Moreover, solving for a Nash equilibrium until convergence can require many iterations, each of which is a (possibly expensive) trajectory-optimization problem. This can lead to prohibitively long solution times.

Second, there are a variety of algorithms based on dynamic programming. In Fisac et al. (2019), a Markovian Stackelberg strategy is computed via dynamic programming. This approach seems to capture the game-theoretic nature of autonomous driving. However, dynamic programming suffers from the curse of dimensionality and, therefore, practical implementations rely on simplified dynamics models coupled with coarse discretization of the state and input spaces. To counterbalance these approximations, a lower-level planner informed by the state values under the Markovian Stackelberg strategy is run. This approach, which scales exponentially with the state dimension, has been demonstrated in a two-player setting. Adding more players is likely to prevent real-time application of this algorithm. In contrast, our proposed approach scales polynomially with the number of players (see Sect. 4.5).

Third, algorithms akin to differential dynamic programming have been developed for robust control (Morimoto and Atkeson 2003; Sun et al. 2015), and later applied to game-theoretic problems (Fridovich-Keil et al. 2020; Di and Lamperski 2018). Similar methods were applied in the stochastic (Sun et al. 2016), and belief-space planning

settings (Schwarting et al. 2021). This approach scales polynomially with the number of players and is fast enough to run real-time in a MPC fashion (Fridovich-Keil et al. 2020). However, contrary to ALGAMES, this type of approach does not natively handle constraints. Collision-avoidance constraints are typically handled using large penalties that can result in numerical ill-conditioning which, in turn, can impact the robustness or the convergence rate of the solver. Moreover, it leads to a trade-off between trajectory efficiency and avoiding collisions with other players.

Finally, algorithms that are analogous to direct methods in trajectory optimization have also been developed (Di and Lamperski 2019, 2020). An algorithm based on a first-order splitting method that is known to have a linear convergence rate was proposed by Di and Lamperski (2020). Di's experiments show convergence of the algorithm after typically 10^3 to 10^4 iterations. A different approach based on Newton's method has been proposed (Di and Lamperski 2019), but it is restricted to unconstrained dynamic games. Our solver belongs to this family of approaches. It also relies on a second-order Newton-type method, but it is able to handle general state and control input constraints. In addition, we demonstrate convergence on relatively complex problems in typically less than 10^2 iterations.

2.3 Generalized nash equilibrium problems

We focus on finding Nash equilibria for multi-player games in which players are coupled through shared state constraints (such as collision-avoidance). Therefore, these problems are instances of Generalized Nash Equilibrium Problems (GNEPs). The operations research field has a rich literature on GNEPs (Pang and Fukushima 2005; Facchinei and Pang 2006; Facchinei et al. 2009; Facchinei and Kanzow 2010; Fukushima 2011). Exact penalty methods have been proposed to solve GNEPs (Facchinei and Pang 2006; Facchinei et al. 2009). Complex constraints such as those that couple players' strategies are handled using penalties, allowing solution of multi-player games jointly for all the players. However, these exact penalty methods require minimization of nonsmooth objective functions, which leads to slow convergence rates in practice.

In the same vein, a penalty approach relying on an augmented Lagrangian formulation of the problem has been advanced by Pang and Fukushima (2005). This work, however, converts the augmented Lagrangian formulation to a set of KKT conditions, including complementarity constraints. The resulting constraint-satisfaction problem is solved with an off-the-shelf linear complementarity problem (LCP) solver that exploits the linearity of a specific problem. Our solver, in contrast, is not tailored for a specific example and can solve general GNEPs. It draws inspiration from the augmented Lagrangian formulation, which does

not introduce nonsmooth terms in the objective function, enabling fast convergence. Moreover, this formulation avoids ill-conditioning, which improves the numerical robustness of our solver.

Another solution method based on KKT conditions for constrained dynamic games has been proposed (Schwarting et al. 2019). It consists of stacking together the individual KKT conditions of all the players and solve the resulting problem as a single optimization using an off-the-shelf interior point solver (IPOPT) (Wächter and Biegler 2006). This approach has been explored with box constraints but not with more complicated inequality constraints such as collision avoidance. Indeed, when the individual problems contain inequality constraints, complementarity constraints are introduced in the stacked KKT system. Those constraints are not natively handled by interior point solvers as the KKT system has no interior at the solution. They also violate constraint qualifications that are required for well-posed, nonlinear programs (Biegler 2010). Off-the-shelf solvers typically rely on iterative relaxation schemes that may lead to solution times precluding their use in an MPC framework. ALGAMES is less general than a general-purpose optimization package such as IPOPT, but is targeted specifically at solving GNEPs involving complex inequality constraints with an augmented Lagrangian scheme that allows for fast solution times.

3 Problem statement

In the discretized trajectory-optimization setting with N time steps, we denote by n the state size, m the control-input size, x_k the state, and u_k^ν the control input of player ν at the time step k. In formulating the game, we do not distinguish between the robot carrying out the computation, and the other agents whose trajectories it is predicting. All agents are modeled equivalently, as is typical in the case of Nash-style games.

Following the formalism of Facchinei and Kanzow (2007), we consider the GNEP with M players. Each player ν decides over its control input variables $U^\nu = [(u_1^\nu)^T \ldots (u_{N-1}^\nu)^T]^T \in \mathbb{R}^{\bar{m}^\nu}$. This is player ν's strategy where m^ν denotes the dimension of the control inputs controlled by player ν and $\bar{m}^\nu = m^\nu(N-1)$ is the dimension of the whole trajectory of player ν's control inputs. By $U^{-\nu}$, we denote the vector of all the players' strategies except the one of player ν. Additionally, we define the trajectory of state variables $X = [(x_2)^T \ldots (x_N)^T]^T \in \mathbb{R}^{\bar{n}}$ where $\bar{n} = n(N-1)$, which results from applying all the control inputs decided by the players to a joint dynamical system,

$$x_{k+1} = f\left(x_k, u_k^1, \ldots, u_k^M\right) = f(x_k, u_k), \quad (1)$$

with k denoting the time-step index. The kinodynamic constraints over the whole trajectory can be expressed with $\bar{n}$ equality constraints,

$$D\left(X, U^1, \ldots, U^M\right) = D(X, U) = 0 \ \in \mathbb{R}^{\bar{n}}. \quad (2)$$

The cost function of each player is noted $J^\nu(X, U^\nu) : \mathbb{R}^{\bar{n}+\bar{m}^\nu} \to \mathbb{R}$. It depends on player ν's control inputs U^ν as well as on the state trajectory X, which is shared with all the other players. The goal of player ν is to select a strategy U^ν and a state trajectory X that minimizes the cost function J^ν. Naturally, the choice of state trajectory X is constrained by the other players' strategies $U^{-\nu}$ and the dynamics of the system via Eq. 2. In addition, the strategy U^ν must respect a set of constraints that depends on the state trajectory X as well as on the other players strategies $U^{-\nu}$ (e.g., collision-avoidance constraints). We express this with a concatenated set of inequality constraints $C : \mathbb{R}^{\bar{n}+\bar{m}} \to \mathbb{R}^{n_c}$. We model the behavior of the players by assuming that each player chooses inputs to solve its own constrained optimization problem, formally,

$$
\begin{array}{ll}
\min\limits_{X, U^1} & J^1(X, U^1) \qquad\qquad \min\limits_{X, U^M} \ J^M(X, U^M) \\
\text{s.t.} & D(X, U) = 0, \quad \cdots \quad \text{s.t.} \ D(X, U) = 0, \\
& C(X, U) \le 0, \qquad\qquad\qquad\quad C(X, U) \le 0,
\end{array}
\quad (3)
$$

Generalized Nash Equilibrium Problem—A Nash equilibrium is reached when each of these M coupled optimization problems reaches an optimal point. This set of M Problems (3), forms a GNEP because of the constraints that couple the strategies of all the players. A solution of this GNEP (a generalized Nash equilibrium), is a vector $\hat{U}$ such that, for all $\nu = 1, \ldots, M$, $\hat{U}^\nu$ is a solution to (3) with the other players' strategies fixed to $\hat{U}^{-\nu}$. This means that at an equilibrium point $\hat{U}$, no player can decrease their cost by unilaterally changing their strategy U^ν to any other feasible point.

When solving for a generalized Nash equilibrium of the game, U, we identify open-loop Nash equilibrium trajectories, in the sense that the whole trajectory U^ν is the best response to the other players' strategies $U^{-\nu}$ given the initial state of the system x_0. Thus, the control signal is a function of time, not of the current state of the system[1] x_k. However, one can repeatedly resolve the open-loop game as new information is obtained over time to obtain a policy that is closed-loop in the model-predictive control sense, as demonstrated in Sect. 7. This formulation is general enough to comprise multi-player dynamic games with nonlinear constraints on the states and control inputs. Practically, in the

---

[1] One might also explore solving for feedback Nash equilibria, where the strategies are functions of the state of all agents. This is an interesting direction for future work.

context of autonomous driving and other scenarios in multi-robot autonomy, the cost function J^ν encodes the objective of player ν, while the concatenated set of constraints, C, includes collision constraints coupled between players. We assume differentiability of the constraints and twice differentiability of the cost functions.

4 Augmented Lagrangian formulation

We propose an algorithm to solve the previously defined GNEP in the context of trajectory optimization. We express the condition that players are acting optimally to minimize their cost functions subject to constraints as an equality constraint. To do so, we first derive the augmented Lagrangian associated with Problem (3) solved by each player. Then, we use the fact that, at an optimal point, the gradient of the augmented Lagrangian is null (Bertsekas 2014). Therefore, at a generalized Nash equilibrium point, the gradients of the augmented Lagrangians of all players must be null. Concatenating this set of M equality constraints with the dynamics equality constraints, we obtain a set of equations that we solve using a quasi-Newton root-finding algorithm.

4.1 Individual optimality

First, without loss of generality, we suppose that the vector C is actually the concatenated set of inequality and equality constraints, i.e., $C = [C_i^T \ C_e^T]^T \in \mathbb{R}^{n_{ci}+n_{ce}}$, where $C_i \leq 0$ is the vector of inequality constraints and $C_e = 0$ is the vector of equality constraints (e.g. terminal state equality constraints). To embed the notion that each player is acting optimally, we formulate the augmented Lagrangian associated with Problem (3) for player ν. The dynamics constraints are handled with the Lagrange multiplier term $\mu^\nu \in \mathbb{R}^{\bar{n}}$, while the other constraints are dealt with using both a multiplier and a quadratic penalty term specific to the augmented Lagrangian formulation. As a motivation for this differential treatment; one typically handles inequality and highly nonlinear equality constraints with an augmented Lagrangian formulation for its improved robustness. We denote by $\lambda \in \mathbb{R}^{n_c}$ the Lagrange multipliers associated with the vector of constraints C; $\rho \in \mathbb{R}^{n_c}$ is a penalty weight;

$$L^\nu(X, U) = J^\nu + \mu^{\nu T} D + \lambda^T C + \frac{1}{2} C^T I_\rho C. \tag{4}$$

I_ρ is a diagonal matrix defined as,

$$I_{\rho,kk} = \begin{cases} 0 & \text{if } C_k(X, U) < 0 \ \wedge \ \lambda_k = 0, \ k \leq n_{ci}, \\ \rho_k & \text{otherwise,} \end{cases} \tag{5}$$

where $k = 1, \ldots, n_{ci} + n_{ce}$ indicates the kth constraint. It is important to notice that the Lagrange multipliers μ^ν associated with the dynamics constraints are specific to each player ν, but the Lagrange multipliers and penalties λ and ρ are common to all players. Given the appropriate Lagrange multipliers μ^ν and λ, the gradient of the augmented Lagrangian with respect to the individual decision variables $\nabla_{X,U^\nu} L^\nu = G^\nu$ is null at an optimal point of Problem (3). The fact that player ν is acting optimally to minimize J^ν under the constraints D and C can therefore be expressed as follows,

$$\nabla_{X,U^\nu} L^\nu(X, U, \mu^\nu) = G^\nu(X, U, \mu^\nu) = 0. \tag{6}$$

It is important to note that this equality constraint preserves coupling between players since the gradient G^ν depends on the other players' strategies $U^{-\nu}$.

4.2 Root-finding problem

At a generalized Nash equilibrium, all players are acting optimally and the dynamics constraints are respected. Therefore, to find an equilibrium point, we have to solve the following root-finding problem,

$$\begin{aligned} \min_{X,U,\mu} \quad & 0, \\ \text{s.t.} \quad & G^\nu(X, U, \mu^\nu) = 0, \quad \forall \ \nu \in \{1, \ldots, M\} \\ & D(X, U) = 0. \end{aligned} \tag{7}$$

We note that the set of constraints, C, is embedded in G and implicitly handled through augmented Lagrangian penalties. Therefore, we do not need to add the primal feasibility constraint, $C(X, U) \leq 0$, to Problem 7. We use Newton's method to solve the root-finding problem. We denote by G the concatenation of the augmented Lagrangian gradients of all players and the dynamics constraints, $G(X, U, \mu) = [(G^1)^T, \ldots, (G^M)^T, D^T]^T$, where $\mu = [(\mu^1)^T, \ldots, (\mu^M)^T]^T \in \mathbb{R}^{\bar{n}M}$. We compute the first-order derivative of G with respect to the primal variables X, U and the dual variables μ that we concatenate in a single vector $y = [(X)^T, (U)^T, (\mu)^T]$,

$$H = \nabla_{X,U,\mu} G = \nabla_y G. \tag{8}$$

Newton's method allows us to identify a search direction δy in the primal-dual space,

$$\delta y = -H^{-1} G. \tag{9}$$

We couple this search direction with a backtracking line-search (Nocedal and Wright 2006) given in Algorithm 1 to ensure local convergence to a solution using Newton's

Method (Nocedal and Wright 2006) detailed in Algorithm 2.

Algorithm 1 Backtracking line-search

1: **procedure** LINESEARCH(y, G, δy)
2:    **Parameters**
3:    $\alpha = 1$,
4:    $\beta \in (0, 1/2)$,
5:    $\tau \in (0, 1)$,
6:    **Until** $||G(y + \alpha \delta y)||_1 < (1 - \alpha\beta)||G(y)||_1$ **do**
7:       $\alpha \leftarrow \tau\alpha$
8:    **return** α

Algorithm 2 Newton's method for root-finding problem

1: **procedure** NEWTON'SMETHOD(y)
2:    **Until** Convergence **do**
3:       $G \leftarrow [(\nabla_{X,U^1} L^1)^T, \ldots, (\nabla_{X,U^M} L^M)^T, D^T]^T$
4:       $H \leftarrow \nabla_y G$
5:       $\delta y \leftarrow -H^{-1}G$
6:       $\alpha \leftarrow$ LINESEARCH(y, G, δy)
7:       $y \leftarrow y + \alpha\delta y$
8:    **return** y

Algorithm 3 ALGAMES solver

1: **procedure** ALGAMES(y_0, ρ_0)
2:    **Initialization**
3:    $\rho \leftarrow \rho^{(0)}$,
4:    $\lambda \leftarrow 0$,
5:    $\mu^\nu \leftarrow 0$, $\forall \nu$
6:    $X, U \leftarrow X^{(0)}, U^{(0)}$
7:    **Until** Convergence **do**
8:       $y \leftarrow$ NEWTON'SMETHOD(y)
9:       $\lambda \leftarrow$ DUALASCENT(y, λ, ρ),   ▷ Eq. 10
10:      $\rho \leftarrow$ INCREASINGSCHEDULE(ρ),   ▷ Eq. 11
11:    **return** y

4.3 Augmented Lagrangian updates

To obtain convergence of the Lagrange multipliers λ, we update them with a dual-ascent step. This update can be seen as shifting the value of the penalty terms into the Lagrange multiplier terms,

$$\lambda_k \leftarrow \begin{cases} \max(0, \lambda_k + \rho_k C_k(X, U)) & k \leq n_{ci}, \\ \lambda_k + \rho_k C_k(X, U) & n_{ci} < k \leq n_{ci} + n_{ce}. \end{cases} \tag{10}$$

We also update the penalty weights according to an increasing schedule, with $\gamma > 1$:

$$\rho_k \leftarrow \gamma\rho_k, \quad \forall k \in \{1, \ldots, n_c\}. \tag{11}$$

4.4 ALGAMES

By combining Newton's method for finding the point where the dynamics is respected and the gradients of the augmented Lagrangians are null with the Lagrange multiplier and penalty updates, we obtain our solver ALGAMES (Augmented Lagrangian GAME-theoretic Solver) presented in Algorithm 3. The algorithm, which iteratively solves the GNEP, requires as inputs an initial guess for the primal-dual variables $y^{(0)}$ and initial penalty weights $\rho^{(0)}$. The algorithm outputs the open-loop strategies of all players X, U and the Lagrange multipliers associated with the dynamics constraints μ.

4.5 Algorithm complexity

Following a quasi-Newton approximation of the matrix H (Nocedal and Wright 2006), we neglect some of the second-order derivative terms associated with the constraints. Indeed, these terms involve third-order tensors which require significant amounts of computation per iteration; while only marginally increasing the progress made per iteration. Therefore, the most expensive part of the algorithm is the Newton step defined by Eq. 9. By exploiting the sparsity pattern of the matrix H, we can solve Eq. 9 using a back-substitution scheme akin to solving a Riccati recursion with complexity $O(N(n + m)^3)$. The complexity is cubic in the number of states n and the number of control inputs m, which are typically linear in the number of players M. Therefore, the complexity of one iteration of the algorithm is $O(NM^3)$. We evaluate the scalability of ALGAMES for a varying number of players in a drone navigation scenario (Fig. 2b) on 5 different dynamical systems (unicycle, bicycle, quadrotor etc.) with varying number of states and control inputs (Fig. 2a). The timing results are shown in Fig. 2c.

For Newton-type methods, the overall *algorithm complexity* is the same as the *iteration complexity* since the number of iterations required to converge to an optimal solution is independent of the problem size; provided that the initial guess is close to an optimal solution (Facchinei and Kanzow 2007). The same reasoning about complexity holds for iLQGames. On the other hand, IBR has a smaller iteration complexity, $O(NM)$, it is linear in terms of the number of players. However, the overall algorithm complexity depends on rate of convergence to a Nash equilibrium. In theory, convergence is guaranteed under extremely restrictive conditions (Facchinei and Kanzow 2007). In practice, we have observed low convergence rates on a simple problem where the only constraints were linear kinodynamics constraints (Fig. 2d). ALGAMES is 4–10 times faster than IBR for a typical convergence threshold (residual $< 10^{-4}$.) Thus, even with a large number of players, ALGAMES remains significantly faster than IBR.

Model	State dim.	Control dim.
Quadrotor	12	4
Double Integrator 3D	6	3
Bicycle	4	2
Unicycle	4	2
Double Integrator 2D	4	2

(a)
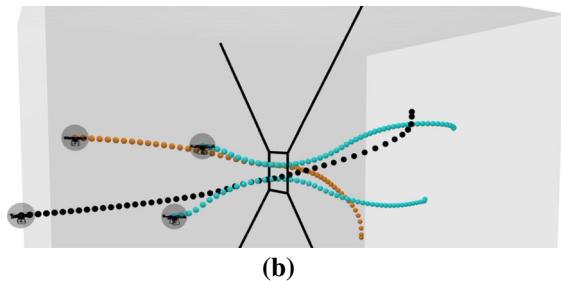
(b)
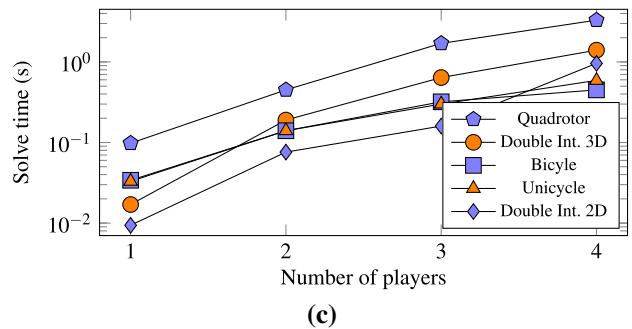
(c)
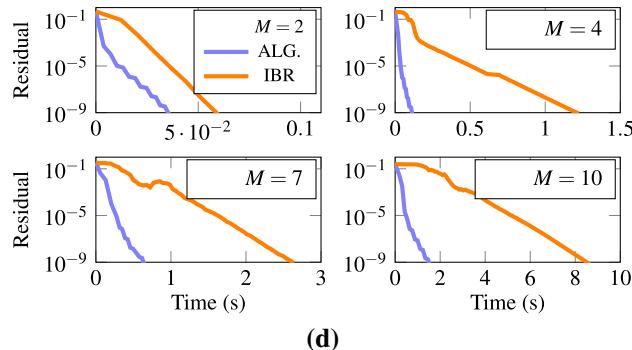
(d)

Fig. 2 We evaluate the scalability of ALGAMES for an increasing number of players and compare its performance to IBR

4.6 Algorithm discussion

Here we discuss the inherent difficulty in solving for Nash equilibria in large problems, and explain some of the limitations of our approach. First of all, finding a Nash equilibrium is a non-convex problem in general. Indeed, it is known that even for single-shot discrete games, solving for exact Nash equilibria is computationally intractable for a large number of players (Daskalakis et al. 2009). It is, therefore, not surprising that, in our more difficult setting of a dynamic game in continuous space, no guarantees can be provided about finding an exact Nash equilibrium. Furthermore, in complex interaction spaces, constraints can be highly non-linear and nonconvex. This is the case in the autonomous driving context with collision-avoidance constraints. In this setting, one cannot expect to find an algorithm working in polynomial time with guaranteed convergence to a Nash equilibrium respecting constraints. On the other hand, *local* convergence of Newton's method to open-loop Nash equilibria has been established in the unconstrained case (that is, starting sufficiently close to the equilibrium, the algorithm will converge to it) (Di and Lamperski 2019). Our approach solves a sequence of unconstrained problems via the augmented Lagrangian formulation. Each of these problems, therefore, has guaranteed *local* convergence. However, as expected, the overall method has no guarantee of global convergence to a generalized Nash equilibrium.

Second, our algorithm requires an initial guess for the state and control input trajectories X, U and the dynamics multipliers μ. Empirically, we observe that choosing $\mu = 0$ and simply rolling out the dynamics starting from the initial state x_0 without any control was a sufficiently good initial guess to get convergence to a local optimum that respects both the constraints and the first-order optimality conditions. For a detailed empirical study of the convergence of ALGAMES and its failure cases, we refer to Sects. 5.5 and 5.6.

Finally, even for simple linear-quadratic games, the Nash equilibrium solution is not necessarily unique. In general, an entire subspace of equilibria exists. In this case, the matrix H in Equation 9 will be singular. In practice, we regularize this matrix so that large steps δy are penalized, resulting in an invertible matrix H.

5 Simulations: design and setup

We choose to apply our algorithm in the autonomous driving context. Indeed, many maneuvers like lane changing, ramp merging, overtaking, and intersection crossing involve a high level of interaction between vehicles. We assume a single car is computing the trajectories for all cars in its neighborhood, so as to find its own trajectory to act safely among the group. We assume that this car has access to a relatively good estimate of the surrounding cars' objective functions. Such an estimate could, in principle, be obtained by applying inverse optimal control on observed trajectories of the surrounding cars.

In a real application, the car would compute its strategy as frequently as possible in a receding-horizon loop to adapt to unforeseen changes in the environment. We demonstrate the feasibility of this approach on complex driving scenarios where a classical predict-then-plan architecture fails to overcome the "frozen robot" problem.

5.1 Autonomous driving problem

Constraints Each vehicle in the scene is an agent of the game. Our objective is to find a generalized Nash equilibrium trajectory for all of the vehicles. These trajectories have to be dynamically feasible. The dynamics constraints at time step k are expressed as follows,

$$x_{k+1} = f\left(x_k, u_k^1, \ldots, u_k^M\right). \tag{12}$$

We consider a nonlinear unicycle model for the dynamics of each vehicle. A vehicle state, x_k^v, is composed of a 2D position, a heading angle and a scalar velocity. The control input u_k^v is composed of an angular velocity and a scalar acceleration. In addition, it is critical that the trajectories respect collision-avoidance constraints. We model the collision zone of the vehicles as circles of radius r. The collision constraints between vehicles are then simply expressed in terms of the position $\tilde{x}_k^v$ of each vehicle,

$$(2r)^2 - ||\tilde{x}_k^v - \tilde{x}_k^\omega||_2^2 \leq 0, \quad \forall \, v, \omega \in \{1, \ldots, M\}, v \neq \omega. \tag{13}$$

We also model boundaries of the road to force the vehicles to remain on the roadway. This means that the distance between the vehicle and the closest point, q, on each boundary, b, has to remain larger than the collision-circle radius, r,

$$r^2 - ||\tilde{x}_k^v - q_b||_2^2 \leq 0, \quad \forall \, b, \, \forall \, v \in \{1, \ldots, M\}. \tag{14}$$

In summary, based on reasonable simplifying assumptions, we have expressed the driving problem in terms of non-convex and non-linear coupled constraints.

Cost Function We use a quadratic cost function penalizing the use of control inputs and the distance between the current state and the desired final state x_f of the trajectory. We also add a quadratic penalty on being close to other cars,

$$\begin{aligned}
J^v(X, U^v) = \sum_{k=1}^{N-1} \frac{1}{2}(x_k - x_f)^T Q (x_k - x_f) + \frac{1}{2} u_k^{v\,T} R u_k^v \\
+ \frac{1}{2}(x_N - x_f)^T Q_f (x_N - x_f) \\
+ \sum_{k=1}^{N} \sum_{\omega \neq v} \gamma \left(\max \left(0, ||\tilde{x}_k^v - \tilde{x}_k^\omega||_2 - \eta \right) \right)^2,
\end{aligned} \tag{15}$$

η controls the distance at which this penalty is "activated", and γ controls its magnitude.

(a)

(b)
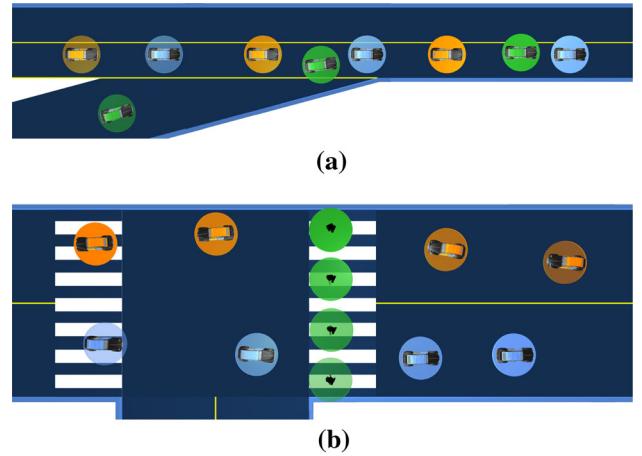
Fig. 3 Two driving environments are considered: a ramp merging scenario (top) and an intersection crossing scenario (bottom)

5.2 Comparison to iLQGames

In order to evaluate the merits of ALGAMES, we compare it to iLQGames (Fridovich-Keil et al. 2020) which is a DDP-based algorithm for solving general dynamic games. Both algorithms solve the problem by iteratively solving linear-quadratic approximations that have an analytical solution (Basar and Olsder 1999). For iLQGames, the augmented objective function $\hat{J}^v$ differs from the objective function, J^v, by a quadratic term penalizing constraint violations,

$$\hat{J}^v(X, U) = J^v(X, U) + \frac{1}{2} C(X, U)^T I_\rho C(X, U). \tag{16}$$

where I_ρ is defined by,

$$I_{\rho,kk} = \begin{cases} 0 & \text{if } C_k(X, U) < 0, \ k \leq n_{ci}, \\ \rho_k & \text{otherwise.} \end{cases} \tag{17}$$

Here ρ is an optimization hyperparameter that we can tune to satisfy constraints. For ALGAMES, the augmented objective function, L^v, is actually an augmented Lagrangian, see Eq. 4. The hyperparameters for ALGAMES are the initial value of $\rho^{(0)}$ and its increase rate γ defined in Eq. 11.

5.3 Timing experiments

We evaluate the performance of both algorithms in two scenarios (see Fig. 3) with the number of players varying from two to four. To compare the speed of both algorithms, we set the termination criterion as a threshold on constraint violations $C \leq 10^{-3}$. The timing results averaged over 100 samples are presented in Table 4a. First, we notice that both algorithms achieve real-time or near-real-time performance on complex autonomous driving scenarios (the horizon of the solvers is fixed to 5*s*).

Scenario	# Players	ALGAMES	iLQGames
Ramp	2	**38 ± 10ms**	104 ± 23*ms*
Merging	3	**89 ± 14ms**	197 ± 15*ms*
	4	860 ± 251*ms*	**705 ± 209ms**
	2	**50 ± 11ms**	752 ± 168*ms*
Intersection	3	**116 ± 22ms**	362 ± 93*ms*
	4	**509 ± 33ms**	1905 ± 498*ms*

(a)
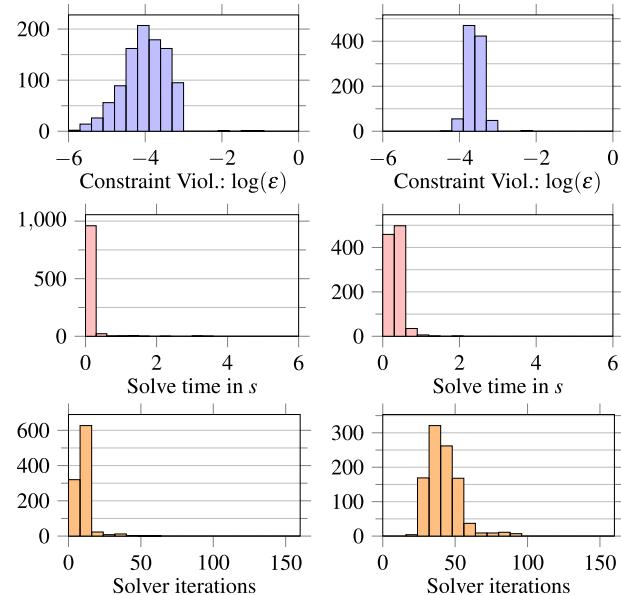
(b)

Scenario	Frequency	$\mathbb{E}[\delta t]$	$\sigma[\delta t]$
Ramp Merging	69 Hz	14 ms	72 ms
Intersection	66 Hz	15 ms	66 ms

(c)

Fig. 4 We compare ALGAMES and iLQGames, both in terms of solve time (top) and ability to reliably enforce constraints (middle). Additionally, we evaluate the update frequency of ALGAMES in the MPC setting (bottom)

We observe that the speed performance of ALGAMES and iLQGames are comparable in the ramp merging scenario. For this scenario, we tuned the value of the penalty for iLQGames to $\rho = 10^2$. Notice that for all scenarios the dimensions of the problem are scaled so that the velocities and displacements are all the same order of magnitude. For the intersection scenario, we observe that the two-player and four-player cases both have much higher solve times for iLQGames compared to the 3-player case. Indeed, in those two cases, we had to increase the penalty to $\rho = 10^3$, otherwise the iLQGames would plateau and never reach the constraint satisfaction criterion. This, in turn, slowed the

algorithm down by decreasing the constraint violation convergence rate.

5.4 Discussion

The main takeaway from these experiments is that, for a given scenario, it is generally possible to find a suitable value for ρ that will ensure the convergence of iLQGames to constraint satisfaction. With higher values for ρ, we can reach better constraint satisfaction at the expense of slower convergence rate. In the context of a receding horizon implementation (MPC), finding a good choice of ρ that would suit the whole sequence of scenarios encountered by a vehicle could be difficult. In contrast, the same hyperparameters $\rho^{(0)} = 1$ and $\gamma = 10$ were used in ALGAMES for all the experiments across this paper. This supports the idea that, thanks to its adaptive penalty scheme, ALGAMES requires little tuning.

While performing the timing experiments, we also noticed several instances of oscillatory behavior for iLQGames. The solution would oscillate, preventing it from converging. This happened even after an adaptive regularization scheme was implemented to regularize iLQGames' Riccati backward passes. Oscillatory behavior was not seen with ALGAMES. We hypothesize that this is due to the dual ascent update coupled with the penalty logic detailed in Eqs. 5 and 10, which add hysteresis to the solver.

5.5 Monte Carlo analysis

To evaluate the robustness of ALGAMES, we performed a Monte Carlo analysis of its performance on a ramp merging problem. First, we set up a roadway with hard boundaries as pictured in Fig. 3a. We position two vehicles on the roadway and one on the ramp in a collision-free initial configuration. We choose a desired final state where the incoming vehicle has merged into the traffic. Our objective is to generate generalized Nash equilibrium trajectories for the three vehicles. These trajectories are collision-free and cannot be improved unilaterally by any player. To introduce randomness in the solving process, we apply a random perturbation to the initial state of the problem. Specifically, we perturb x_0 by adding a uniformly sampled noise. This would typically correspond to displacing the initial position of the vehicles by $\pm 1m$, changing their initial velocity by $\pm 3\%$ and their heading by $\pm 2.5°$.

We observe in Fig. 4b, that ALGAMES consistently finds a satisfactory solution to the problem using the same hyperparameters $\rho^{(0)} = 1$ and $\gamma = 10$. Out of the 1000 samples 99.5% converged to constraint satisfaction $C \leq 10^{-3}$ while respecting the optimality criterion $||G||_1 < 10^{-2}$. By definition, $||G||_1$ is a merit function for satisfying optimality and dynamics constraints. We also observe that the solver converges to a solution in less than $0.2s$ for 96% of the samples.

These empirical data tend to support the fact that ALGAMES is able to solve the class of ramp merging problem quickly and reliably.

For comparison, we present in Fig. 4b the results obtained with iLQGames. We apply the same constraint satisfaction criterion $C \leq 10^{-3}$. We fixed the value of the penalty hyper-parameter ρ for all the samples as it would not be a fair comparison to tune it for each sample. Only 3 samples did not converge with iLQGames, this is a performance comparable to ALGAMES for which 5 samples failed to converge. However, we observe that iLQGames is 3 times slower than ALGAMES with an average solve time of 350 ms compared to 110 ms and require on average 4 times more iterations (9 against 41).

5.6 Solver failure cases

The Monte Carlo analysis allows us to identify the typical failure cases of our solver, i.e. the cases where the solver does not satisfy the constraints or the optimality criterion. Typically in such cases, the initial guess, which consists of rolling out the dynamics with no control, is far from a reasonable solution. Since the constraints are ignored during this initial rollout, the car at the back can overtake the car at the front by driving through it. This creates an initial guess where constraints are strongly violated. Moreover, we hypothesize that the tight roadway-boundary constraints tend to strongly penalize solutions that would 'disentangle' the car trajectories because they would require large boundary violation at first. Therefore, the solver gets stuck in this local optimum where cars overlap each other. Sampling several initial guesses with random initial control inputs and solving in parallel could reduce the occurrence of these failure cases. Also, being able to detect, reject, and re-sample initial guesses when the initial car trajectories are strongly entangled could also improve the robustness of the solver.

6 Non-uniqueness of nash equilibria

A Nash equilibrium corresponds to a situation where all players are acting optimally given the other players' strategies. This is a way for players to compete in a coordinated fashion without communication. However, if the Nash equilibrium is non-unique, the coordination is ambiguous and players have to decide individually which Nash equilibrium to follow. This can lead to inconsistencies. The non-uniqueness of Nash equilibrium solutions has been observed in practical robotics applications such as autonomous driving (Peters et al. 2020). Peters et al. identified isolated clusters of solutions in unconstrained Nash equilibrium problems and proposed an estimation method to improve players' coordination. In this section, we detail several underlying causes

of non-uniqueness that arise in practical robotics scenarios. Additionally, we present the behavior of ALGAMES in such circumstances.

6.1 Linear-quadratic dynamic games

Linear-quadratic (LQ) dynamic games are an important building block for optimization algorithms relying on sequential-quadratic approximations such as ALGAMES or iLQGames (Fridovich-Keil et al. 2020). The conditions for the existence and uniqueness of a Nash equilibrium have been extensively studied (Basar 1976; Abraham and Kulkarni 2019). In the continuous-time setting, Eisele characterized the different solution regimes for the LQ game, including non-existence and non-uniqueness (Eisele 1982). In the discrete-time setting, the open-loop Nash equilibrium problem is equivalent to a static quadratic game (i.e., a one-step game). For such problems, the Nash equilibrium solutions can either be non-existant, can form an affine subspace, or be a single point in the case of a unique solution.

Proof Sketch We focus on the two-player case, the result can easily be extended to the M-player case. We denote, s^ν and J^ν, the strategy and quadratic cost function of player ν,

$$J^\nu(s^1, s^2) = \frac{1}{2} \begin{bmatrix} s^1 \\ s^2 \end{bmatrix}^T \begin{bmatrix} Q_{1,1}^\nu & Q_{1,2}^\nu \\ Q_{2,1}^\nu & Q_{2,2}^\nu \end{bmatrix} \begin{bmatrix} s^1 \\ s^2 \end{bmatrix} + \begin{bmatrix} s^1 \\ s^2 \end{bmatrix}^T \begin{bmatrix} q_1^\nu \\ q_2^\nu \end{bmatrix} + c^\nu. \tag{18}$$

The first-order necessary conditions for optimality of a Nash equilibrium, $\hat{e} = (\hat{s}^1, \hat{s}^2)$, can be written as an affine equation,

$$\begin{bmatrix} Q_{1,1}^1 + Q_{2,1}^2 & Q_{2,2}^2 + Q_{1,2}^1 \end{bmatrix} \begin{bmatrix} \hat{s}^1 \\ \hat{s}^2 \end{bmatrix} + \begin{bmatrix} q_1^1 \\ q_2^2 \end{bmatrix} = 0. \tag{19}$$

The second-order necessary conditions are independent of the Nash equilibrium point considered. They require positive semi-definiteness of the matrices $Q_{\nu,\nu}^\nu$, for all $\nu \in \{1, 2\}$. Therefore, any point, e, in the affine subspace defined by Eq. 19, will respect both the first-order and second-order necessary conditions for optimality. □

In case of a unique Nash Equilibrium, ALGAMES converges in one Newton iteration to the solution. When the Nash equilibrium solutions form an affine subspace, ALGAMES converges to the point in the subspace closest to the initial guess. This is due to the regularization added to the Jacobian of the KKT condition, H, defined in Eq. 8.

6.2 Isolated nash equilibria

We have seen that an LQ game can generate an affine Nash equilibrium subspace. Thus, it cannot lead to multiple isolated Nash equilibria. However, in general, a dynamic game

(a)

(b)
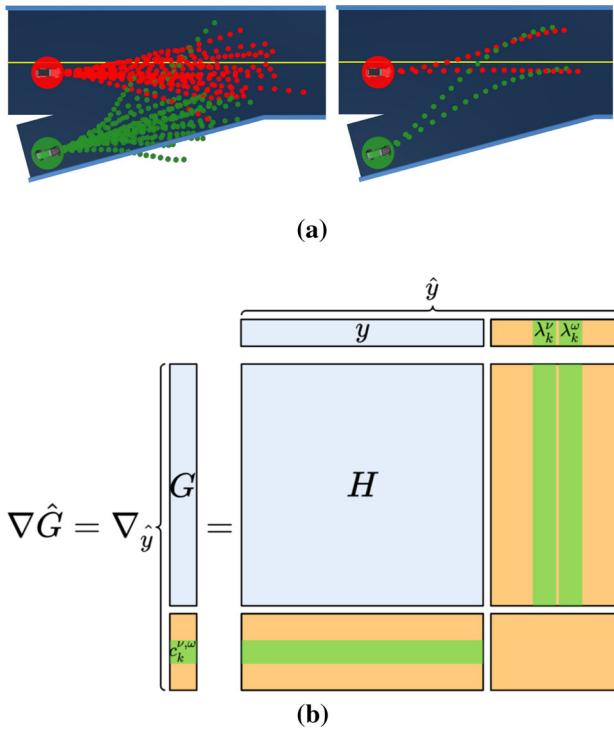
Fig. 5 We illustrate isolated Nash equilibria (top), and non-isolated Nash equilibrium solutions stemming from an underdetermined KKT system (bottom)

can admit multiple isolated Nash equilibria as highlighted by Peters et al. (2020). In unconstrained autonomous driving scenarios, they generally appear when collision-avoidance costs are introduced. These costs are non-convex and introduce a coupling between the players' strategies. Typically, these isolated Nash equilibria correspond to "topologically" different driving strategies. For instance, in a ramp merging scenario, the merging vehicle can merge in front of or behind an incoming vehicle (Fig. 5a). The equilibrium point to which ALGAMES converges is typically the closest to the initial guess, thanks to the regularization scheme. This is a desirable property, especially in the MPC setting , because it prevents the re-planned trajectory from oscillating between different Nash equilibria.

6.3 Generalized nash equilibrium

Identifying the solution set of a GNEP remains a major challenge as pointed out by Fisher et al. in an extensive survey (Fischer et al. 2014). In general, the solution set of the GNEP can be constituted of one or many isolated points or even non-isolated points. Theoretical results in this domain often rely on strong assumptions, such as convexity of the feasible set, absence of shared constraints, or decoupled cost functions (Dreves and Gerdts 2018). All these assumptions could be violated in a typical robotic scenario. Indeed, collision-

avoidance constraints are shared and non-convex. Similarly, collision-avoidance costs or congestion terms introduce coupling between the players' costs.

We explore the structure of the generalized Nash equilibrium (GNE) solutions in the presence of shared collision-avoidance constraints. We denote, $c_k^{\nu,\omega} : \mathbb{R}^n \to \mathbb{R}$, the collision-avoidance constraint between player ν and player ω at time step k. For each collision-avoidance constraint $c_k^{\nu,\omega}$, we introduce two Lagrange multipliers $\lambda_k^\nu \in \mathbb{R}$ and $\lambda_k^\omega \in \mathbb{R}$; one for each player. We remark that, for a single constraint, we add two Lagrange multipliers. We denote, N_c, the number of collision-avoidance constraints. By concatenating these constraints with the residual vector G, we add N_c entries and N_c rows to its Jacobian. We denote $\hat{G}$ and $\hat{H}$ the "augmented" residual vector and Jacobian matrix. We need to differentiate the residual $\hat{G}$, with respect to the $2N_c$ Lagrange multipliers associated with the collision constraints. This adds $2N_c$ columns to the Jacobian $\hat{H}$. Thus, the Jacobian is an underdetermined linear system, with N_c more columns than rows (Fig. 5b). However, only active collision-avoidance constraints should be included in the Jacobian. Therefore, the Jacobian only has N_a more columns than rows, where N_a denotes the number of active collision-avoidance constraints. Thus, the nullspace of the underdetermined linear system $\hat{H}$ is at least of dimension N_a (Fig. 5b). Consequently, the solution set of a GNEP can potentially be composed of non-isolated points and could span in multiple dimensions locally around a known equilibrium point.

We explore this nullspace at a Nash equilibrium point by slightly disturbing the current generalized Nash equilibrium in one of the nullspace's directions (Fig. 6a). We obtain a continuum of GNE. Additionally, Fig. 6b, c, present the two main directions in which the solution can drift while remaining a GNE. The nullspace was of dimension 17, which corresponds to the number of active constraints at the equilibrium point. Yet, we notice that most of the trajectory variability is captured by a limited number of eigenvectors. We remark that the two principal eigenvectors have an elegant interpretation: they both favor one vehicle over the others. Additionally, they nicely show how disturbing the trajectory of one player influences the trajectories of the other players through the collision constraints. Finally, by combining these two eigenvectors and stepping in the resulting direction, one could favor any of the three vehicles.

6.4 ALGAMES' convergence to normalized nash equilibrium

We observe that the multipliers λ_k^ν and λ_k^ω associated with the shared constraint, $c_k^{\nu,\omega}$, are equal at every iteration of the solver. Indeed, we can assume that the multipliers are initialized with the same value (typically zero). Moveover, these multipliers are updated with identical dual ascent updates,

(a)

(b)

(c)
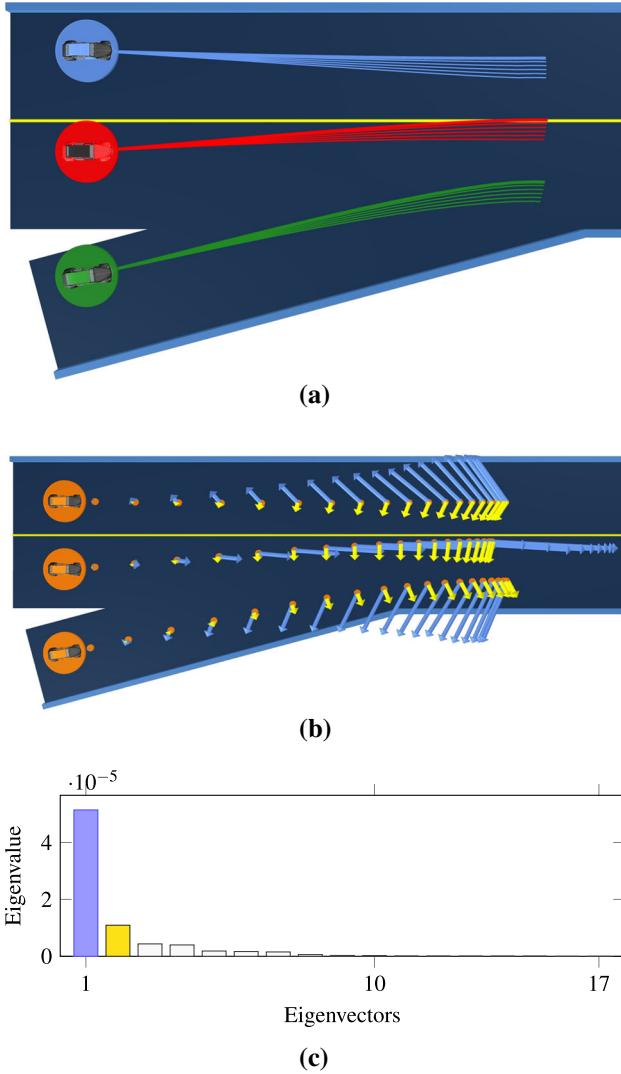
Fig. 6 We explore the generalized equilibrium subspace in one direction and obtain a sequence of GNE (top). Additionally, we represent the two principal vectors along which the trajectories can evolve while remaining a GNE (middle and bottom)

defined in Equation 10,

$$\lambda_k^{v(0)} = \lambda_k^{\omega(0)} \tag{20}$$

$$\lambda_k^{v(t+1)} = \max\left(0, \lambda_k^{v(t)} + \rho_k c_k^{v,\omega}(x_k)\right) \tag{21}$$

$$\lambda_k^{\omega(t+1)} = \max\left(0, \lambda_k^{\omega(t)} + \rho_k c_k^{v,\omega}(x_k)\right) \tag{22}$$

$$\implies \lambda_k^{v(t+1)} = \lambda_k^{\omega(t+1)} \tag{23}$$

Here, t denotes the iteration index. A consequence of this trivial recursion is that the multipliers associated with the same shared constraint are equal at the solution. Therefore, if ALGAMES converges, it converges to a Normalized Nash Equilibrium (NNE) in the sense of Rosen (1965). An NNE is a GNE with the additional requirement that the multipliers associated with shared constraints are equal. This reason-

ing was applied by Dreves to a potential reduction method (Dreves 2011). We transcribe it in the augmented Lagrangian context. At an NNE, because the multipliers are equal, the price to pay for violating the collision-avoidance constraint is the same for both players. This can be interpreted as enforcing a notion of "fairness" between the players in addition to optimality. One interesting characteristic of NNE, compared to GNE, is that they are not subject to the nullspace issue described in Sect. 6.3. Indeed, thanks to the additional constraints enforcing equality between multipliers, active constraints no longer introduce more columns than rows in the KKT system.

7 MPC implementation of ALGAMES

In this section, we propose an MPC implementation of the algorithm that provides us with a feedback policy instead of an open-loop strategy and demonstrates real-time performance. We compare this MPC to a non-game-theoretic baseline on a crowded ramp merging which is known to be conducive to the "frozen robot" problem.

7.1 MPC feedback policy

The strategies identified by ALGAMES are open-loop Nash equilibrium strategies. They are sequences of control inputs. On the contrary, DDP-based approaches like iLQGames solve for feedback Nash equilibrium strategies that provide a sequence of control gains. In the MPC setting, we can obtain a feedback policy with ALGAMES by updating the strategy as fast as possible and only executing the beginning of the strategy. This assumes a fast update rate of the solution. To support the feasibility of the approach, we implemented an MPC on the ramp merging scenario described in Fig. 3a. There are 3 players constantly maintaining a 40 time step strategy with 3 s of horizon. We simulate 3 s of operation of the MPC by constantly updating the strategies and propagating noisy unicycle dynamics for each vehicle. We compile the results from 100 MPC trajectories in Table 4c. We obtain a 69 Hz update frequency for the planner on average. We observe similar performance on the intersection problem defined in Fig. 3b, with an update frequency of 66 Hz.

7.2 "Unfreezing" the robot

To illustrate the benefits of using ALGAMES in a receding-horizon loop, we compare it to a non-game-theoretic baseline MPC. With this baseline, the prediction step and the planning step are decoupled. Specifically, each agent predicts the trajectories of the surrounding vehicles by propagating straight, constant velocity trajectories. Then, each agent plans for itself assuming these predicted trajectories are immutable

obstacles. We test these two controllers on a challenging scenario where a vehicle has to merge on a crowded highway as presented in Fig. 1. We perform a Monte Carlo analysis by uniformly sampling the initial state, x_0, around a nominal state with perturbations corresponding to a ± 2.5 m longitudinal displacement, ± 25 cm lateral displacement, $\pm 3°$ in angular displacement for each car. Given the initial state, the vehicle on the ramp should be able to merge between the blue and orange cars or the orange and green cars, taking the 2nd and 3rd place respectively. However, waiting for all cars to pass before merging into 4th place is not a desirable behavior. Indeed, with such a policy, the merging vehicle has to slow down significantly and could get stuck on the ramp if the highway does not clear. We run ALGAMES in a receding horizon loop and the baseline MPC to generate 6-s trajectories for 100 different initial states. We record the position of the merging vehicle at the end of the simulation and compile the results in Fig. 7a.

We observe that the "frozen robot" problem occurs with the baseline MPC for 85% of the simulations. An interpretation of this result is that the vehicle on the ramp cannot find a merging maneuver that is not colliding with its constant-velocity trajectory predictions. Since there is no feasible merging maneuver, the only option left is to wait for the other vehicles to pass before merging.

On the contrary, by running ALGAMES in a receding-horizon loop, the vehicle merges into traffic in 2nd or 3rd place in 96% of the simulations (Fig. 7a). ALGAMES avoids the "frozen robot" pitfall in most cases by gradually adjusting its velocity to merge with minimal disruption to the traffic (Fig. 1).

7.3 Non-uniqueness of nash equilibria in practice

We assess the effect of the non-uniqueness of Nash equilibria in the MPC context. We focus on the coordination issue, that players may face when there exists multiple Nash equilibria. In our experiment, each car independently runs ALGAMES as an MPC policy. Each car plans for itself and predicts the other vehicles' trajectories. We purposefully provide each player with a very different initial guess, in order to generate a mismatch between the Nash equilibrium solution that each player converges to. We simulate this on a ramp merging scenario with two players. In this scenario, an example of Nash equilibrium mismatch could be that both players think they let the oher player go first (Fig. 7b). The results, presented in Fig. 7c, suggest that most of the mismatches disappear rapidly after the initialization, i.e both players converges to the same Nash equilibrium. This can happen, for instance, when one Nash equilibrium is no longer feasible because it violates the bounds on the control inputs or the boundaries of the road. This positive results mitigates the concern caused by the potential occurrence of non-unique

(a)

Predicted: ● ● ● ● ●   Planned: ▬▬▬

(b)

(c)
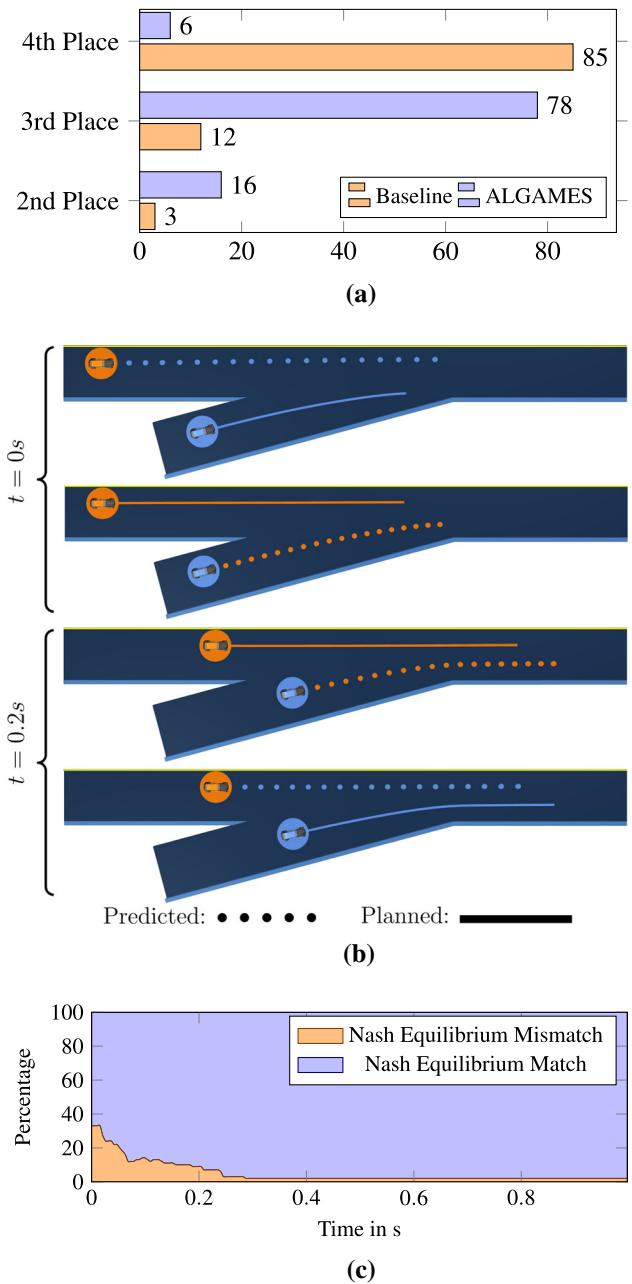
Fig. 7 We evaluate the ability of ALGAMES to avoid the "frozen robot" problem and to handle Nash equilibrium non-uniqueness

Nash equilibria. Nevertheless, it is also important to analyze the failure cases, where the two Nash equilibrium solutions found by the two players do not coincide. Typically, in these cases, each player's solution remains fairly constant and does not oscillate between multiple equilibria. In such circumstances, it would be appropriate to estimate the equilibrium that the other player is following in order to switch to this equilibrium. Peters et al. demonstrated the feasibility of this approach in similar scenarios, using a particle filter (Peters et al. 2020).

8 Conclusions

We have introduced a new algorithm for finding constrained Nash equilibrium trajectories in multi-player dynamic games. We demonstrated the performance and robustness of the solver through a Monte Carlo analysis on complex autonomous driving scenarios including nonlinear and non-convex constraints. We have shown real-time performance for up to 4 players and implemented ALGAMES in a receding-horizon framework to give a feedback policy. We empirically demonstrated the ability of ALGAMES to mitigate the "frozen robot" problem in comparison to a non-game-theoretic receding horizon planner. The results we obtained from ALGAMES are promising, as they seem to let the vehicles share the responsibility for avoiding collisions, leading to natural-looking trajectories where players are able to negotiate complex, interactive traffic scenarios that are challenging for traditional, non-game-theoretic trajectory planners. For this reason, we believe that ALGAMES could be a very efficient tool to generate trajectories in situations where the level of interaction between players is strong. Our implementation of ALGAMES is available at https://github.com/Robotic ExplorationLab/ALGAMES.jl.

References

Abraham, M. P., & Kulkarni, A. A. (2019). New results on the existence of open loop Nash equilibria in discrete time dynamic games via generalized Nash games. *Mathematical Methods of Operations Research, 89,* 157–172.

Basar, T. (1976). On the uniqueness of the Nash solution in linear-quadratic differential games. *International Journal of Game Theory, 5,* 65–90.

Basar, T., & Olsder, G. J. (1999). *Dynamic Noncooperative Game Theory* (Vol. 23). SIAM.

Bertsekas, D. P. (2014). *Constrained optimization and Lagrange multiplier methods*. Academic press.

Biegler, L. T. (2010). *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. Society for Industrial and Applied Mathematics.

Britzelmeier, A., Dreves, A., & Gerdts, M. (2019). Numerical solution of potential games arising in the control of cooperative automatic vehicles. In S. Levine & R. Stockbridge (Eds.) *2019 Proceedings of the conference on control and its applications* (Philadelphia, PA) (pp. 38–45). Society for Industrial and Applied Mathematics.

Daskalakis, C., Goldberg, P. W., & Papadimitriou, C. H. (2009). The complexity of computing a Nash equilibrium. *SIAM Journal on Computing, 39*(1), 195–259.

Di, B., & Lamperski, A. (2018). Differential dynamic programming for nonlinear dynamic games. arXiv:1809.08302 [math].

Di, B., & Lamperski, A. (2019). Newton's method and differential dynamic programming for unconstrained nonlinear dynamic games. In *2019 IEEE 58th conference on decision and control (CDC)*, (Nice, France) (pp. 4073–40780. IEEE.

Di, B., & Lamperski, A. (2020). First-order algorithms for constrained nonlinear dynamic games. arXiv:2001.01826 [cs, eess].

Dreves, A. (2011). *Globally convergent algorithms for the solution of generalized nash equilibrium problems*. Ph.D. Thesis, Julius-Maximilians-Universitat Wurzburg, Wurzburg.

Dreves, A., & Gerdts, M. (2018). A generalized Nash equilibrium approach for optimal control problems of autonomous cars: A generalized Nash equilibrium approach for optimal control problems of autonomous cars. *Optimal Control Applications and Methods, 39,* 326–342.

Eisele, T. (1982). Nonexistence and nonuniqueness of open-loop equilibria in linear-quadratic differential games. *Journal of Optimization Theory and Applications, 37,* 443–468.

Facchinei, F., Fischer, A., & Piccialli, V. (2009). Generalized Nash equilibrium problems and Newton methods. *Mathematical Programming, 117,* 163–194.

Facchinei, F., & Kanzow, C. (2007). Generalized Nash equilibrium problems. *4OR, 5,* 173–210.

Facchinei, F., & Kanzow, C. (2010). Penalty methods for the solution of generalized Nash equilibrium problems. *SIAM Journal on Optimization, 20,* 2228–2253.

Facchinei, F., & Pang, J.-S. (2006). Exact penalty functions for generalized Nash problems. *Large-Scale Nonlinear Optimization,* 115–126.

Fisac, J. F., Bronstein, E., Stefansson, E., Sadigh, D., Sastry, S. S., & Dragan, A. D. (2019). Hierarchical game-theoretic planning for autonomous vehicles. In *2019 International conference on robotics and automation (ICRA)*, (Montreal, QC, Canada) (pp. 9590–9596). IEEE.

Fischer, A., Herrich, M., & Schönefeld, K. (2014). Generalized Nash equilibrium problems—Recent advances and challenges. *Pesquisa Operacional, 34,* 521–558.

Fridovich-Keil, D., Ratner, E., Peters, L., Dragan, A. D., & Tomlin, C. J. (2020). Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In *2020 IEEE international conference on robotics and automation (ICRA)*, (Paris, France) (pp. 1475–1481). IEEE.

Fukushima, M. (2011). Restricted generalized Nash equilibria and controlled penalty algorithm. *Computational Management Science, 8,* 201–218.

Liniger, A., & Lygeros, J. (2019). A noncooperative game approach to autonomous racing. *IEEE Transactions on Control Systems Technology,* 1–14.

Morimoto, J., & Atkeson, C. G. (2003). Minimax differential dynamic programming: An application to robust biped walking. *Advances in Neural Information Processing Systems,* 1563–1570.

Nocedal, J., & Wright, S. (2006). *Numerical optimization*. Springer.

Pang, J.-S., & Fukushima, M. (2005). Quasi-variational inequalities, generalized Nash equilibria, and multi-leader-follower games. *Computational Management Science, 2,* 21–56.

Peters, L., Fridovich-Keil, D., Tomlin, C. J., & Sunberg, Z. N. (2020). Inference-based strategy alignment for general-sum differential games. arXiv:2002.04354 [cs, eess].

Rosen, J. B. (1965). Existence and uniqueness of equilibrium points for concave N-person games. *Econometrica, 33,* 520.

Sadigh, D., Sastry, S. S., Seshia, S. A., & Dragan, A. (2016a). Information gathering actions over human internal state. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, (Daejeon, South Korea) (pp. 66–73). IEEE.

Sadigh, D., Sastry, S., Seshia, S. A., & Dragan, A. D. (2016b). Planning for autonomous cars that leverage effects on human actions. In *Robotics: science and systems XII, robotics: science and systems foundation*.

Schmerling, E., Leung, K., Vollprecht, W., & Pavone, M. (2018). Multimodal probabilistic model-based planning for human-robot interaction. In *2018 IEEE international conference on robotics and automation (ICRA)*, (Brisbane, QLD) (pp. 1–9). IEEE.

Schwarting, W., Pierson, A., Alonso-Mora, J., Karaman, S., & Rus, D. (2019). Social behavior for autonomous vehicles. *Proceedings of the National Academy of Sciences, 116,* 24972–24978.

Schwarting, W., Pierson, A., Karaman, S., & Rus, D. (2021). Stochastic dynamic games in belief space. *IEEE Transactions on Robotics,* 1–16.

Sun, W., Theodorou, E. A., & Tsiotras, P. (2015). Game theoretic continuous time differential dynamic programming. In *2015 American control conference (ACC)*, (Chicago, IL, USA) (pp. 5593–5598). IEEE.

Sun, W., Theodorou, E. A., & Tsiotras, P. (2016). Stochastic game theoretic trajectory optimization in continuous time. In *2016 IEEE 55th conference on decision and control (CDC)*, (Las Vegas, NV, USA) (pp. 6167–6172). IEEE.

Trautman, P., & Krause, A. (2010). Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 IEEE/RSJ international conference on intelligent robots and systems*, (Taipei) (pp. 797–803). IEEE.

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming, 106,* 25–57.

Wang, M., Wang, Z., Talbot, J., Gerdes, J. C., & Schwager, M. (2021). Game-theoretic planning for self-driving cars in multivehicle competitive scenarios. *IEEE Transactions on Robotics,* 1–13.

Yoo, J. H., & Langari, R. (2012). Stackelberg game based model of highway driving. In *Volume 1: Adaptive control; advanced vehicle propulsion systems; aerospace systems; autonomous systems; battery modeling; biochemical systems; control over networks; control systems design; cooperativ*, (Fort Lauderdale, Florida, USA) (pp. 499–508). ASME.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Simon Le Cleac'h is a graduate student with the Robotic Exploration Lab at Carnegie Mellon University and with the Multi-Robot Systems Lab at Stanford University. He received his B.S. in Engineering from Ecole Centrale Paris in 2016 and his M.S. in Mechanical Engineering from Stanford University in 2019. His research interests include real-time motion-planning, game-theoretic optimization, and inverse optimal control.

Mac Schwager is an assistant professor with the Aeronautics and Astronautics Department at Stanford University. He obtained his B.S. degree in 2000 from Stanford University, his MS degree from MIT in 2005, and his Ph.D. degree from MIT in 2009. He was a postdoctoral researcher working jointly in the GRASP lab at the University of Pennsylvania and CSAIL at MIT from 2010 to 2012, and was an assistant professor at Boston University from 2012 to 2015. He received the NSF CAREER award in 2014, the DARPA YFA in 2018, and a Google faculty research award in 2018, and the IROS Toshio Fukuda Young Professional Award in 2019. His research interests are in distributed algorithms for control, perception, and learning in groups of robots, and models of cooperation and competition in groups of engineered and natural agents.

Zachary Manchester is an assistant professor in the Robotics Institute at Carnegie Mellon University and founder of the Robotic Exploration Lab. He received a Ph.D. in aerospace engineering in 2015 and a BS in applied physics in 2009, both from Cornell University. He was a postdoctoral fellow in the Agile Robotics Lab at Harvard from 2015 to 2017 and an assistant professor at Stanford from 2018 to 2020. He received the NASA Early Career Faculty Award in 2018 and a Google Faculty Research Award in 2020. His research interest include numerical optimization, control and estimation with applications to aerospace and robotic systems with challenging nonlinear dynamics.