# ALGAMES: A Fast Solver for Constrained Dynamic Games

Simon Le Cleac'h[1], Mac Schwager[2], and Zachary Manchester[2]

***Abstract*— Dynamic games are an effective paradigm for dealing with the control of multiple interacting actors. Current algorithms for solving these problems either rely on Hamilton-Jacobi-Isaacs (HJI) methods, dynamic programming (DP), differential dynamic programming (DDP), or an iterative best response approach (IBR). The first two approaches have strong theoretical guarantees; however they becomes intractable in high-dimensional real-world applications. The third approach is grounded in the success of iLQR. It is scalable, but it cannot handle constraints. Finally, the iterative best response algorithm is a heuristic approach with unknown convergence properties, and it can suffer from stability and tractability issues. This paper introduces ALGAMES (Augmented Lagrangian GAME-theoretic Solver), a solver that handles trajectory optimization problems with multiple actors and general nonlinear state and input constraints. We evaluate our solver in the context of autonomous driving on scenarios involving numerous vehicles such as ramp merging, overtaking, and lane changing. We present simulation and timing results demonstrating the speed and the ability of the solver to produce efficient, safe, and natural autonomous behaviors.**

## I. Introduction

Controlling a robot in an environment where it interacts with other actors is a complex task. Traditional approaches in the literature adopt a partitioned approach. First, predictions of other actors' trajectories are computed, then they are fed into a trajectory optimizer that considers them as immutable obstacles. This approach is limiting because the effect of the robot's trajectory on the other actors is ignored. Moreover, it can lead to the "freezing robot" problem that arises when the planner finds that all paths to the goal are unsafe [1]. The consequence is that the robot stops moving or executes unnecessary anti-collision maneuvers.

For this reason, dealing with the game-theoretic aspect of the planning problem 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 comprise some degree of game-theoretic interactions [2], [3], [4], [5], [6], [7]. Other potential applications include mobile robots navigating in crowds, like package delivery robots, tour guides, or domestic robots. Also, robots interacting with people in factories such as mobile robots, or fixed-base multi-link manipulators. Finally, it is applicable in competitive settings, e.g. drone and car racing [8], [9].

In this work, we propose to solve constrained multi-player general-sum dynamic games. In dynamic games, the players' strategies are sequences of decisions. It is important to notice that unlike traditional optimization problem, non-cooperative games have no "optimal" solution. Depending on the structure of the game, asymmetry between players, etc., one achieves different concepts of solutions. 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 his cost by unilaterally changing his strategy. For extensive details about the game-theory concepts addressed in this paper, we refer readers to the work of Bressan [10] and Basar et al. [11].

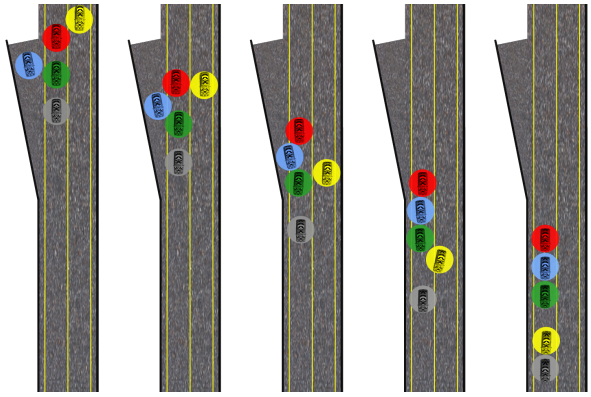

Fig. 1. Sequence of images depicting the trajectories obtained by solving for Nash equilibrium strategies. The images appear in chronological order from left to right. The vehicles achieve a ramp merging and a lane change while avoiding collision with other vehicles and respecting road boundaries.

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 interaction is most naturally, and most correctly, represented as (typically nonlinear) state constraints. This is a crucial point 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 mathematical game theory, Nash equilibria with constraints are referred to as *Generalized Nash Equilibria* [12]. 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 condition for Nash equilibria, analogous to the Karush-Kuhn-Tucker (KKT) conditions

[1]Department of Mechanical Engineering, Stanford University, Stanford, USA simonlc@stanford.edu

[2]Department of Aeronautics & Astronautics, Stanford University, Stanford, USA {schwager,zacmanchester}@stanford.edu

in optimization. By relying on an augmented Lagrangian approach to robustly handle constraints, the solver is able to solve multi-player games with numerous agents and a high level of interactions at speeds approaching real-time. Our primary contributions are:

1) A general solver for dynamic games aimed at identifying Generalized Nash Equilibrium strategies.
2) Demonstration of the solver's speed, robustness, and scalability in autonomous driving applications (Fig. 1).

## II. Related Work

### A. Equilibrium Selection

Recent work focused on solving multi-player dynamic games can be categorized by the type of equilibrium they select. Several works [2], [3], [9], [13] opted for the search of Stackelberg equilibria, which models an asymmetry of information between players. It is usually formulated for games with two players, a leader and a follower. The leader chooses his 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 [6], searching for open-loop Stackelberg equilibrium strategies can fall flat 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 can ignore 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.

Search for Nash equilibria has been investigated in [4], [5], [8], [14]. 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.

### B. Game-Theoretic Trajectory Optimization

Most of the algorithms proposed in the robotics literature to solve for game-theoretic equilibria can be grouped into three types. First, algorithms relying on a decomposition method such as Jacobi or Gauss-Siedel methods [8], [14], [15]. These algorithms are based on an iterative best response scheme. All the players take turns at improving their strategies, considering the other agents' strategies as immutable. It is aimed at finding Nash equilibria. This type of approach is easy to interpret and handles games with numerous players well. However, convergence of these algorithms is not well understood [12], and special care is required to capture the game-theoretic nature of the problem [8]. 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 [6], 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 relies on a simplified dynamics model coupled with a coarse discretization of the state and input space. 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 only been demonstrated in a two-player setting. Adding more players would prevent real-time application of this algorithm. Our proposed approach, on the contrary, scales reasonably with the number of players (see Fig. 4).

Finally, algorithms akin to differential dynamic programming have been developed for robust control [16] and later applied to game-theoretic problems [4]. This approach scales polynomially with the number of players and is potentially fast enough to run real-time in a model predictive control (MPC) fashion. However, this approach does not handle constraints. Collision-avoidance constraints are typically handled using large penalties that can result in numerical ill-conditioning and a brittle solver. Moreover, it leads to a trade-off between trajectory efficiency and avoiding collisions with other players. This approach seems questionable in the autonomous driving context. Our approach, however, can enforce nonlinear state and input constraints in a rigorous way.

### C. Generalized Nash Equilibrium Problems

As mentioned above, we focus on finding Nash equilibria for multi-player games in which players are coupled through shared state constraints (such as collision-avoidance constraints). Therefore, these problems are instances of Generalized Nash Equilibrium Problems (GNEPs). The operations research field has a rich literature on GNEPs [17], [18], [19], [20], [21]. Exact penalty methods have been proposed to solve GNEPs [18], [19]. Complex constraints such as those that couple players' strategies are handled using penalties. This allows solution of multi-player games jointly for all the players, while still being able to reason about complex constraints. However, these exact penalty methods require minimization of nonsmooth objective functions, which turns out to be slow in practice. In the same vein, a penalty approach relying on an augmented Lagrangian formulation of the problem has been advanced by Pang et al. [17]. 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, on the contrary, is not tailored for a specific example and can solve general GNEPs. It draws inspiration from this augmented Lagrangian formulation, which does not introduce nonsmooth terms in the objective function so the solution can be found quickly. Moreover, this formulation

avoids ill-conditioning, which makes our solver numerically robust.

## III. Problem Statement

Following the formalism of Facchinei [12], we consider the GNEP with $M$ players. Each player $\nu$ controls the variables $p^\nu \in \mathbb{R}^{n_p}$. We denote by $\boldsymbol{p}$ the concatenated vector of the individual decision variables,

$$\boldsymbol{p} = \begin{bmatrix} (p^1)^T & \dots & (p^M)^T \end{bmatrix}^T, \tag{1}$$

with dimension $q = Mn_p$. By $\boldsymbol{p}^{-\nu}$, we denote the vector of all the players' decision variables except those of player $\nu$. The cost function of each player is noted $J^\nu : \mathbb{R}^q \to \mathbb{R}$. It depends on player $\nu$'s variables $p^\nu$ as well as on all the other players' variables $\boldsymbol{p}^{-\nu}$. The goal of player $\nu$ is to select a strategy $p^\nu$ that minimizes his cost function $J^\nu$, given the other players' strategies $\boldsymbol{p}^{-\nu}$. In addition, the strategy $p^\nu$ must belong to a set $P^\nu(\boldsymbol{p}^{-\nu})$, and we express this constraint with a concatenated set of inequality constraints $C^\nu : \mathbb{R}^q \to \mathbb{R}^{n_c}$. Formally,

$$\begin{aligned} \min_{p^\nu} \quad & J^\nu(p^\nu, \boldsymbol{p}^{-\nu}), \\ \text{s.t.} \quad & C^\nu(p^\nu, \boldsymbol{p}^{-\nu}) \leq 0. \end{aligned} \tag{2}$$

A solution of the GNEP (a generalized Nash equilibrium), is a vector $\bar{\boldsymbol{p}}$ such that for all $\nu = 1, \dots, M$, $\bar{p}^\nu$ is a solution to (2) with the other players' strategies fixed to $\bar{\boldsymbol{p}}^{-\nu}$. This means that at an equilibrium point $\bar{\boldsymbol{p}}$, no player can decrease his cost by unilaterally changing his strategy $p^\nu$ to any other feasible point.

In the discretized trajectory optimization setting with $N$ time steps, we denote by $n$ the state size, $m$ the control input size, $x_t^\nu$ the state, and $u_t^\nu$ the control input of player $\nu$ at the time step $t$. In this context, the decision variables of each player $p^\nu$ designate the primal variables associated with this player. They are the sequences of states $X^\nu = [(x_1^\nu)^T \dots (x_N^\nu)^T]^T$ and control inputs $U^\nu = [(u_1^\nu)^T \dots (u_{N-1}^\nu)^T]^T$ of player $\nu$, i.e.

$$p^\nu = \begin{bmatrix} (X^\nu)^T & (U^\nu)^T \end{bmatrix}^T. \tag{3}$$

Thus, when solving for a generalized Nash equilibrium of the game $\boldsymbol{p}$, we identify open-loop Nash equilibrium trajectories, in the sense that the control signal is a function of time, not of the state variables of the players. 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. The cost function $J^\nu$ encodes the objective of player $\nu$. The concatenated set of constraints $C^\nu$ includes dynamics constraints and, in the context of autonomous driving, collision constraints coupled between players. This formulation is general enough to comprise multi-player general-sum dynamic games with nonlinear constraint on the states and control inputs.

## IV. Augmented Lagrangian Formulation

We propose an algorithm to solve the previously defined GNEP in the context of trajectory optimization. We express the fact that players are acting optimally to minimize their cost functions under constraints as an equality. To do so, we first derive the augmented Lagrangian associated with (2) solved by each player. Then, we use the fact that, at an optimal point, the gradient of the augmented Lagrangian is null [22]. Therefore, at a generalized Nash equilibrium point, the gradients of the augmented Lagrangians of all players must be null. This is a set of $M$ equality constraints that we solve using a quasi-Newton root-finding algorithm.

### A. Individual Optimality

First, without loss of generality, we suppose that the vector $C^\nu$ is actually the concatenated set of inequality and equality constraints, i.e. $C^\nu = [C_i^{\nu T} C_e^{\nu T}]^T \in \mathbb{R}^{n_{ci}+n_{ce}}$, where $C_i^\nu \leq 0$ is the vector of inequality constraints and $C_e^\nu = 0$ is the vector of equality constraints. To embed the notion that each player is acting optimally, we formulate the augmented Lagrangian associated with (2) for player $\nu$. We denote by $\lambda^\nu \in \mathbb{R}^{n_c}$ the Lagrange multipliers associated with the vector of constraints $C^\nu$; $\rho^\nu \in \mathbb{R}^{n_c}$ is a penalty weight vector.

$$L^\nu(p^\nu, \boldsymbol{p}^{-\nu}) = J^\nu + \lambda^{\nu T} C^\nu + \frac{1}{2} C^{\nu T} I_{\rho^\nu} C^\nu. \tag{4}$$

Where $I_{\rho^\nu}$ is a diagonal matrix defined as,

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

where $k = 1, \dots, n_{ci} + n_{ce}$ indicates the $k^{\text{th}}$ constraint. Given the appropriate Lagrange multipliers $\lambda^\nu$, the gradient of the augmented Lagrangian with respect to the individual primal variables $\nabla_{p^\nu} L^\nu = G^\nu$ is null at an optimal point of (2). The fact that player $\nu$ is acting optimally to minimize $J^\nu$ under the constraints $C^\nu$ can therefore be expressed as follows,

$$\nabla_{p^\nu} L^\nu(p^\nu, \boldsymbol{p}^{-\nu}) = G^\nu(p^\nu, \boldsymbol{p}^{-\nu}) = 0. \tag{6}$$

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

### B. Root-Finding Problem

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

$$\begin{aligned} \min_{\boldsymbol{p}} \quad & 0, \\ \text{s.t.} \quad & G^\nu(p^\nu, \boldsymbol{p}^{-\nu}) = 0, \ \forall\, \nu \in \{1, \dots, M\}. \end{aligned} \tag{7}$$

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 $G(\boldsymbol{p}) = [(G^1)^T, \dots, (G^M)^T]^T$. We compute the first order derivative of $G$ with respect to all primal variables, $H = \nabla_{\boldsymbol{p}} G$. Newton's

method allows us to identify a search direction $\delta \boldsymbol{p}$ in the primal variables space,

$$\delta \boldsymbol{p} = -H^{-1}G. \tag{8}$$

We couple this search direction with a backtracking line-search [23] detailed in Algorithm 1 to ensure local convergence to a solution using Newton's Method [23] presented in Algorithm 2.

**Algorithm 1** Backtracking line-search

**procedure** LINESEARCH($\boldsymbol{p}, G, \delta \boldsymbol{p}$)
  **Parameters**
  $\alpha = 1$,
  $\beta \in (0, 1/2)$,
  $\tau \in (0,1)$,
  **Until** $||G(\boldsymbol{p} + \alpha \delta \boldsymbol{p})||_2 < (1 - \alpha\beta)||G(\boldsymbol{p})||_2$ **do**
    $\alpha \leftarrow \tau\alpha$
  **return** $\alpha$

**Algorithm 2** Newton's method for root-finding problem

**procedure** NEWTON'SMETHOD($\boldsymbol{p}$)
  **Until** Convergence **do**
    $G \leftarrow [(\nabla_{p^1} L^1)^T, \ldots, (\nabla_{p^M} L^M)^T]^T$
    $H \leftarrow \nabla_{\boldsymbol{p}} G$
    $\delta \boldsymbol{p} \leftarrow -H^{-1}G$
    $\alpha \leftarrow$ LINESEARCH($\boldsymbol{p}, G, \delta \boldsymbol{p}$)
    $\boldsymbol{p} \leftarrow \boldsymbol{p} + \alpha \delta \boldsymbol{p}$
  **return** $\boldsymbol{p}$

**Algorithm 3** ALGAMES solver

**procedure** ALGAMES($\boldsymbol{p}_0, \rho_0$)
  **Initialization**
  $\lambda^\nu \leftarrow 0$, $\forall \nu$
  $\rho^\nu \leftarrow \rho_0$, $\forall \nu$
  $\boldsymbol{p} \leftarrow \boldsymbol{p}_0$
  **Until** Convergence **do**
    $\boldsymbol{p} \leftarrow$ NEWTON'SMETHOD($\boldsymbol{p}$)
    $\lambda^\nu \leftarrow$ DUALASCENT($\boldsymbol{p}, \lambda^\nu, \rho^\nu$), $\forall \nu$
    $\rho^\nu \leftarrow$ INCREASINGSCHEDULE($\rho^\nu$), $\forall \nu$
  **return** $\boldsymbol{p}$

### C. Augmented Lagrangian Updates

To obtain convergence of the Lagrange multipliers $\lambda^\nu$, 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,

$$\lambda_k^\nu \leftarrow \begin{cases} \max(0, \lambda_k^\nu + \rho_k^\nu C_k^\nu(\boldsymbol{p})) & k \leq n_{ci}, \\ \lambda_k^\nu + \rho_k^\nu C_k^\nu(\boldsymbol{p}) & n_{ci} < k \leq n_{ci} + n_{ce}. \end{cases} \tag{9}$$

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

$$\rho_k^\nu \leftarrow \gamma \rho_k^\nu, \ \forall k \in \{1, \ldots, n_c\}. \tag{10}$$

### D. ALGAMES

By combining Newton's method for finding the point where 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 variables $\boldsymbol{p}_0$ and initial penalty weights $\rho_0$. The algorithm outputs the primal variables $\boldsymbol{p}$ containing the open-loop strategies of all players. Finding a Nash equilibrium is a non-convex problem in general. There is, therefore, no guarantee about convergence to the global optimum, and our algorithm requires a reasonable initial guess to converge.

## V. 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. Our game-theoretic planner could improve performance in these interactive tasks compared to traditional planners that do not consider coupled interactions among all the 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. In a real application, this computation would be repeated as frequently as possible in an MPC fashion.

### A. Autonomous Driving Problem

*1) Constraints:* Each vehicle in the scene is an actor of the game. Our objective is to find a generalized Nash equilibrium trajectory for each vehicle. These trajectories have to be dynamically feasible. The dynamics constraints at time step $t$ are expressed as follows,

$$x_{t+1}^\nu = f(x_t^\nu, u_t^\nu). \tag{11}$$

Although the solver is able to deal with nonlinear constraints arising from complex dynamics models, we consider only double-integrator dynamics. A vehicle state $x_t^\nu$ is composed of a 2D position and a 2D velocity. The control input $u_t^\nu$ is the 2D acceleration. The dynamics constraints can be expressed as,

$$x_{t+1}^\nu - Ax_t^\nu - Bu_t^\nu = 0. \tag{12}$$

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 $\tilde{x}_t^\nu$ of each vehicle,

$$r^2 - ||\tilde{x}_t^\nu - \tilde{x}_t^\mu||_2^2 \leq 0, \quad \forall \ \nu, \mu \in \{1, \ldots, M\}, \nu \neq \mu. \tag{13}$$

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, $r$,

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

Finally, we enforce a final state constraint on a subset of the state dimensions. With this constraint we can enforce, for instance, a final velocity or a final position of the vehicle along a particular direction. In summary, based on reasonable simplifying assumptions, we have expressed the driving problem in terms of both linear individual constraints and non-convex coupled constraints.

*2) 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 $x_f^v$ of the trajectory,

$$J^v(p^v) = \sum_{t=1}^{N} \frac{1}{2}(x_t^v - x_f^v)^T Q(x_t^v - x_f^v) + \frac{1}{2} u_k^{vT} R u_k^v. \quad (15)$$

This cost function only depends on the decision variables $p^v$ of vehicle $v$. Players' behaviors are coupled only through collision constraints. We could also add terms depending on other vehicles' strategies, such as a congestion penalty.

### B. Driving Scenarios

We test our solver on three different driving scenarios involving strong interactions between vehicles:

*1) Ramp Merging:* First, we set up a roadway with hard boundaries as pictured in Fig. 1 to demonstrate a ramp-merging maneuver. We position multiple vehicles on the roadway in a collision-free initial configuration. We choose a reasonable desired final state where the incoming vehicle has merged into the traffic. We purposefully place numerous players in a relatively confined space to maximize the level of interaction between players. Our objective is to generate generalized Nash equilibrium trajectories for all the vehicles. These trajectories are collision-free and cannot be improved unilaterally by any player.

*2) Lane Changing:* The objective for each vehicle is to change lanes while avoiding collisions (Fig. 6). This situation is challenging because it involves a high level of negotiation [7] between drivers in a real-world setting, which results in strongly coupled trajectories.

*3) Overtaking:* A fast vehicle is placed behind a slower one (Fig. 7). The faster vehicle performs an overtaking maneuver to maintain its desired speed.

## VI. Simulations: Results

### A. Robustness, Speed, and Scalability

*1) Robustness:* To get a better understanding of the algorithm, we plot the L1-norm of the concatenated gradients of the individual Lagrangians $||G||_1$ and the condition number of the second order derivative matrix $H$ in Fig. 2. It corresponds to the ramp merging experiment presented in Figure 5. The gradient curve, similar to a sawtooth wave, surges in value due to dual-ascent updates that impact the value of the Lagrangians. The root-finding algorithm gets the gradient to converge towards zero in few iterations. We observe that the condition number of $H$ incrementally increases after each penalty update. However, it remains in a reasonable range during the solve. This reasonable conditioning of the numerical problem, combined with the consistent behavior

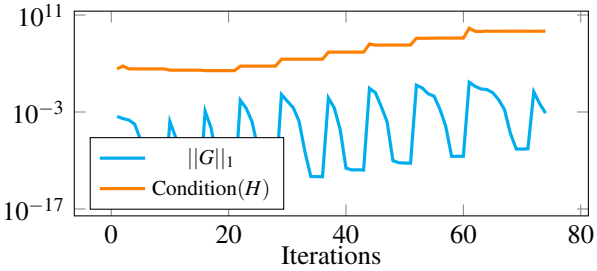


Fig. 2. Behavior of the L1-norm of the gradient $G$ and the condition number of the matrix $H$ during the solve.

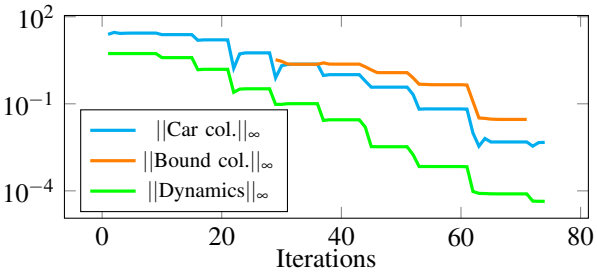


Fig. 3. Convergence of the constraint satisfaction during the solve for the 3-vehicle ramp merging example. The plotted constraints are collision avoidance between vehicles, collision avoidance with road boundaries, and dynamic feasibility. The end of the solve is triggered when all the constraints are satisfied up to a threshold $\varepsilon = 10^{-2}$.

of the root-finding method, exhibits the robustness of the solver. We also plot constraint satisfaction in Fig. 3, where we observe a linear convergence of the maximum constraint violations.

*2) Scalability:* Scalability of the algorithm to scenarios with more than two players is highly desirable. Indeed, driving problems like lane merging often involve 3 or 4 players. We solved for five-second trajectories ($N = 25$ time steps) while increasing number of actors from 2 to 8 on one core of an AMD Ryzen 2950x processor. Fig. 4 demonstrates near-real-time performance on scenarios with 2 and 3 players (2.5s and 6.1s respectively). The solving time increases reasonably with the number of players. Compared to other approaches that scale exponentially with the number of players [6], our method is still tractable for up to 8 players (one-minute solve time for a 5s trajectory).

### B. Results From Driving Scenarios

*1) Ramp Merging:* As pictured in Fig. 5, we observe that the merging vehicle in blue is squeezing between the other two vehicles. They are adapting their trajectories to let the blue vehicle merge smoothly. We also see that the blue vehicle is taking a trajectory that accommodates the other vehicles by squeezing against the ramp boundary represented by a black line on Fig. 5. The algorithm demonstrates near real-time performance as it takes 6.1s to solve the 5s trajectory on one core of an AMD Ryzen 2950x processor.

We then test the solver on a more complex problem with 5 vehicles. In addition to the 3 ramp-merging vehicles, we add 2 vehicles and one of them is performing a lane change. Fig. 1 presents the highly coupled trajectories that we obtain. We observe that the red and green vehicles are slightly nudging on their right to accommodate for the yellow vehicle

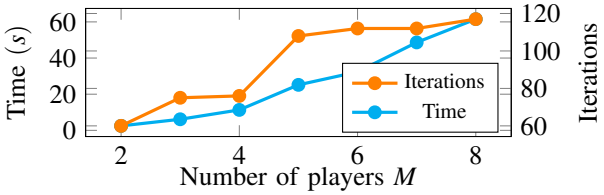


Fig. 4. Solving time and number of iterations required to reach the constraint satisfaction threshold vs. the number of players. All scenarios from 2 to 8 players are of comparable complexity. They include one ramp merging and multiple lane changes when the number of players is sufficient.

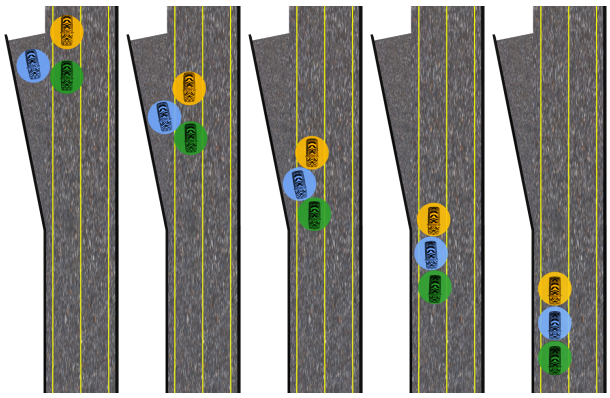

Fig. 5. Ramp merging scenario: the blue vehicle successfully merges on the highway by squeezing between the other two vehicles.

overtaking them. This example demonstrates the robustness of the solver to complex multi-player interactions as well as its scalability. This 5s trajectory is solved in 25.1s.

*2) Lane Changing:* In Fig. 6, we see the 5s trajectory computed in 2.3s in the lane changing scenario. The orange vehicle starts behind the blue one, but with a higher desired speed. To respect its desired speed while achieving lane change, it passes the blue vehicle before changing lanes.

*3) Overtaking:* The five-second overtaking trajectory computed in 4.2s is presented in Fig. 7. We simply place a vehicle with high speed behind a slower vehicle. The cost function is strongly penalizing deviation from the desired speed along the roadway axis and encouraging the vehicles to end their trajectories in the right lane. This is sufficient to trigger an overtaking maneuver by the faster vehicle. The slower vehicle slightly nudges towards the boundary of the road to accommodate the overtaking vehicle's trajectory.

*4) Rich Autonomous Behavior:* Our solver can handle cost functions that depend on the decision variables of all the players $\boldsymbol{p}$. However, we tested our solver with a cost function only dependent on the individual decision variables of each player $p^v$. In this simple setting, the open-loop Stackelberg equilibrium is trivial [2], [3]. The leader chooses his strategy first, ignoring collision constraints. The follower then selects a strategy that has to account for the collision constraints considering the leader's trajectory as immutable. On the contrary, our approach converges to trajectories that account for the individual objective of all players while sharing the responsibility of avoiding collisions, even with "egoistic" cost functions. The conclusion from these experiments is that solving for Nash equilibrium apparently produces natural-looking trajectories.

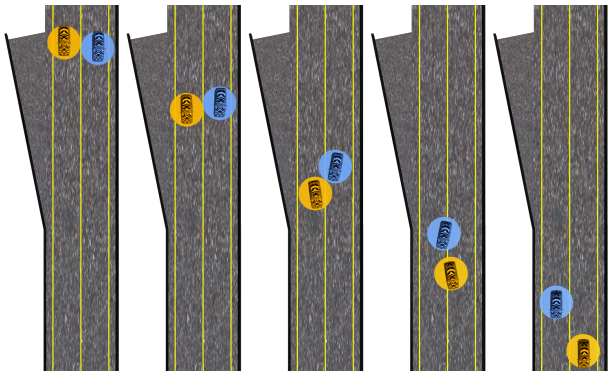

Fig. 6. Lane changing scenario: the two vehicles successfully change lanes.

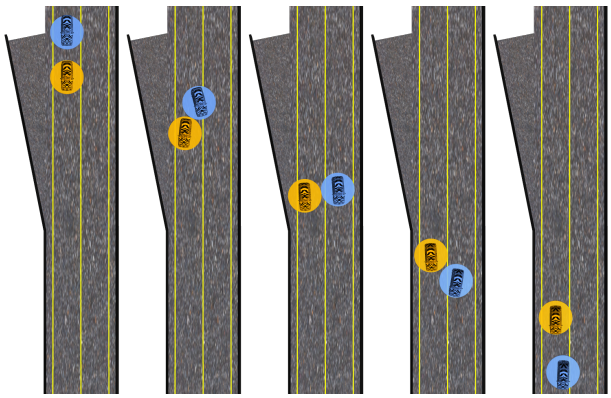

Fig. 7. Overtaking scenario: successful overtaking maneuver.

## VII. Conclusions

We have introduced a new algorithm for finding Nash equilibrium trajectories. We demonstrated the speed and robustness of the solver on complex autonomous driving scenarios including nonlinear and non-convex constraints. We have shown near-real-time performance for up to 3 players. In a real-world driving application, replanning would be performed as frequently as possible to give a feedback policy in the sense of MPC. Parallelizing the computation of both the sparse matrix $H$ and the gradient $G$ should lead to large reductions in solution time, enabling true real-time performance in many realistic scenarios. Indeed, these computations are decomposable across the number of players and the number of time steps. We intend to exploit this enticing property in future work.

The results we obtained from ALGAMES are promising as they seem to let the vehicles share the responsibility for avoiding collisions, leading to seemingly natural 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 this solver could be a very efficient tool to generate trajectories in situations where the level of interaction between players is strong.

## Acknowledgments

This work was supported in part by DARPA YFA award D18AP00064 and NSF NRI award 1830402.

## REFERENCES


[1] Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 797–803, Taipei, October 2010. IEEE.

[2] Dorsa Sadigh, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan. Planning for Autonomous Cars that Leverage Effects on Human Actions. In *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 2016.

[3] Dorsa Sadigh, S. Shankar Sastry, Sanjit A. Seshia, and Anca Dragan. Information gathering actions over human internal state. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 66–73, Daejeon, South Korea, October 2016. IEEE.

[4] David Fridovich-Keil, Ellis Ratner, Jennifer C Shih, Anca D Dragan, and Claire J Tomlin. Iterative Linear Quadratic Approximations for Nonlinear Multi-Player General-Sum Differential Games. page 8, 2019.

[5] Axel Dreves and Matthias Gerdts. 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(1):326–342, January 2018.

[6] Jaime F. Fisac, Eli Bronstein, Elis Stefansson, Dorsa Sadigh, S. Shankar Sastry, and Anca D. Dragan. Hierarchical Game-Theoretic Planning for Autonomous Vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9590–9596, Montreal, QC, Canada, May 2019. IEEE.

[7] Edward Schmerling, Karen Leung, Wolf Vollprecht, and Marco Pavone. Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9, Brisbane, QLD, May 2018. IEEE.

[8] Riccardo Spica, Davide Falanga, Eric Cristofalo, Eduardo Montijano, Davide Scaramuzza, and Mac Schwager. A Real-Time Game Theoretic Planner for Autonomous Two-Player Drone Racing. *arXiv:1801.02302 [cs]*, January 2018.

[9] Alexander Liniger and John Lygeros. A Noncooperative Game Approach to Autonomous Racing. *IEEE Transactions on Control Systems Technology*, pages 1–14, 2019.

[10] Alberto Bressan. Noncooperative Differential Games. A Tutorial. *Department of Mathematics, Penn State University*, page 81, 2010.

[11] Tamer Basar and Geert Jan Olsder. *Dynamic noncooperative game theory*, volume 23. Siam, 1999.

[12] Francisco Facchinei and Christian Kanzow. Generalized Nash equilibrium problems. *4OR*, 5(3):173–210, September 2007.

[13] Je Hong Yoo and Reza Langari. 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*, pages 499–508, Fort Lauderdale, Florida, USA, October 2012. ASME.

[14] Andreas Britzelmeier, Axel Dreves, and Matthias Gerdts. Numerical solution of potential games arising in the control of cooperative automatic vehicles. In William S. Levine and Richard Stockbridge, editors, *2019 Proceedings of the Conference on Control and Its Applications*, pages 38–45, Philadelphia, PA, January 2019. Society for Industrial and Applied Mathematics.

[15] Mingyu Wang, Zijian Wang, John Talbot, and Mac Schwager. Game Theoretic Planning for Self-Driving Cars in Competitive Scenarios. page 9.

[16] Jun Morimoto and Christopher G Atkeson. Minimax Differential Dynamic Programming: An Application to Robust Biped Walking. *Advances in neural information processing systems*, pages 1563–1570, 2003.

[17] Jong-Shi Pang and Masao Fukushima. Quasi-variational inequalities, generalized Nash equilibria, and multi-leader-follower games. *Computational Management Science*, 2(1):21–56, January 2005.

[18] Francisco Facchinei and Jong-Shi Pang. Exact penalty functions for generalized Nash problems. *Large-scale nonlinear optimization*, pages 115–126, 2006.

[19] Francisco Facchinei and Christian Kanzow. Penalty methods for the solution of generalized Nash equilibrium problems (with complete test problems). page 45, 2009.

[20] Francisco Facchinei, Andreas Fischer, and Veronica Piccialli. Generalized Nash equilibrium problems and Newton methods. *Mathematical Programming*, 117(1-2):163–194, March 2009.

[21] Masao Fukushima. Restricted generalized Nash equilibria and controlled penalty algorithm. *Computational Management Science*, 8(3):201–218, August 2011.

[22] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[23] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.