2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Game theoretic controller synthesis for multi-robot motion planning Part I : Trajectory based algorithms

Minghui Zhu, Michael Otte, Pratik Chaudhari, Emilio Frazzoli

***Abstract*— We consider a class of multi-robot motion planning problems where each robot is associated with multiple objectives and decoupled task specifications. The problems are formulated as an open-loop non-cooperative differential game. A distributed anytime algorithm is proposed to compute a Nash equilibrium of the game. The following properties are proven: (i) the algorithm asymptotically converges to the set of Nash equilibrium; (ii) for scalar cost functionals, the price of stability equals one; (iii) for the worst case, the computational complexity and communication cost are linear in the robot number.**

I. Introduction

Robotic motion planning is a fundamental problem where a control sequence is found to steer a robot from the initial state to the goal set, while enforcing the environmental rules. It is well-known that the problem is computationally challenging [23]. The situation is even worse for multi-robot motion planning since the computational complexity exponentially grows as the robot number.

For multi-robot motion planning, non-cooperative game theoretic controller synthesis is interesting in two aspects: *descriptive* and *perspective*. From the descriptive point of view, Nash equilibrium is desirable in inherently competitive scenarios. More specifically, Nash equilibrium characterizes the stable scenarios among inherently self-interested players where none can benefit from unilateral deviations. From the perspective point of view, non-cooperative game theoretic learning holds the promise of providing computationally efficient algorithms for multi-robot controllers where the robots are assumed to be self-interested. Although Nash equilibrium may not be socially optimal, game theoretic approaches remain useful when the computational efficiency dominates.

There have been limited results on rigorous analysis of game theoretic controller synthesis for multi-robot motion planning. The paper [19] tackles multi-robot motion planning in the framework of feedback differential games. However, it lacks of the rigorous analysis of the algorithm's convergence and computational complexity. In addition, static game theory has been used to synthesize distributed control schemes to steer multiple vehicles to stationary and meaningful configurations; e.g., in [29] for optimal sensor deployment, in [1] for vehicle routing and in [2] for target assignment.

Contributions. This paper presents the first distributed, anytime algorithm to compute open-loop Nash equilibrium for non-cooperative robots. More specifically, we consider a class of multi-robot motion planning problems where each robot is associated with multiple objectives and decoupled task specifications. The problems are formulated as an open-loop non-cooperative differential game. By leveraging the RRG algorithm in [16], iterative better response and model checking, a distributed anytime computation algorithm, namely the iNash-trajectory algorithm, is proposed to find a Nash equilibrium of the game. We formally analyze the algorithm convergence, the price of stability as well as the computational complexity and communication cost. The algorithm performance is demonstrated by a number of numerical simulations. Proofs of various theorems and lemmas are ommitted due to lack of space, please refer to [30] for the extended analysis.

Literature review. Sampling based algorithms have been demonstrated to be efficient in addressing robotic motion planning in high-dimension spaces. The Rapidly-exploring Random Tree (RRT) algorithm and its variants; e.g., in [18], [20], are able to find a feasible path quickly. Recently, two novel algorithms, PRM* and RRT*, have been developed in [16], and shown to be computationally efficient and asymptotically optimal. In [17], a class of sampling-based algorithms is proposed to compute the optimal trajectory satisfying the given task specifications in the form of deterministic μ-calculus.

Regarding the multi-robot open-loop motion planning, the approaches mainly fall into three categories: centralized planning in; e.g., [24], [28], decoupled planning in; e.g., [15], [26] and priority planning in; e.g., [9], [12]. Centralized planning is complete but computationally expensive. In contrast, decoupled and priority planning can generate solutions quicker, but are incomplete. However, the existing algorithms assume the robots are cooperative and are not directly applicable to compute Nash equilibrium where none of self-interested robots is willing to unilaterally deviate from.

Another set of relevant papers are concerned with numerical methods for feedback differential games. There have been a very limited number of feedback differential games whose closed-form solutions are known, including homicidal-chauffeur and the lady-in-the-lake games [8], [13]. The methods based on partial differential equations; e.g., in [6], [7], [27], viability theory; e.g., in [3], [4], [10]

M. Zhu is with the Department of Electrical Engineering, Pennsylvania State University, 201 Old Main, University Park, PA, 16802.
Email: muz16@psu.edu.
M. Otte, P. Chaudhari and E. Frazzoli are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge MA, 02139.
Email: ottemw@mit.edu, pratikac@mit.edu, frazzoli@mit.edu.
This research was supported in part by ONR Grant #N00014-09-1-0751 and the Michigan/AFRL Collaborative Center on Control Sciences, AFOSR grant #FA 8650-07-2-3744.

and level-set methods, e.g., in [25] have been proposed to determine numerical solutions to differential games. However, the papers aforementioned only study one-player and two-player differential games. Multi-player linear-quadratic differential games have been studied in; e.g., [8].

II. Problem formulation

Consider a team of robots, labeled by $\mathcal{V}_R \triangleq \{1, \cdots, N\}$. Each robot is associated with a dynamic system governed by the following differential equation:

$$\dot{x}_i(t) = f_i(x_i(t), u_i(t)), \tag{1}$$

where $x_i(t) \in \mathcal{X}_i \subseteq \mathbb{R}^{n_i}$ is the state of robot i, and $u_i(t) \in \mathcal{U}_i$ is the control of robot i. For system (1), the set of admissible control strategies for robot i is given by:

$$\mathcal{U}_i \triangleq \{u_i(\cdot) \,:\, [0, +\infty) \to U_i, \text{ measurable}\},$$

where $U_i \subseteq \mathbb{R}^{m_i}$. For each robot i, $\sigma^{[i]} : [0, +\infty) \to \mathcal{X}_i$ is a dynamically feasible trajectory if there are $T \geq 0$ and $u_i : [0, T] \to \mathcal{U}_i$ such that: (i) $\dot{\sigma}^{[i]}(t) = f_i(\sigma^{[i]}(t), u_i(t))$; (ii) $\sigma^{[i]}(0) = x_{\text{init}}^{[i]}$; (iii) $\sigma^{[i]}(t) \in \mathcal{X}_i^F$ for $t \in [0, T]$; (iv) $\sigma^{[i]}(T) \in \mathcal{X}_i^G$. The set of dynamically feasible trajectories for robot i is denoted by Σ_i. Note that the trajectories in Σ_i do not account for the inter-robot collisions.

Let Π_i be a finite set of atomic propositions and $\lambda_i : \mathcal{X}_i \to 2^{\Pi_i}$ associates each state in $\mathcal{X}_i$ with a set of atomic propositions in Π_i. Given a trajectory $\sigma^{[i]}$ of (1), define by $T(\sigma^{[i]})$ the set of time instances when $\lambda_i(\sigma^{[i]}(t))$ changes. The word $w(\sigma^{[i]}) \triangleq \{w_0, w_1, \cdots\}$ generated by the trajectory $\sigma^{[i]}$ is such that $w_i = \lambda_i(\sigma^{[i]}(t_i))$ where $t_0 = 0$ and $T(\sigma^{[i]}) = \{t_1, t_2, \cdots\}$.

In this paper, we consider reachability tasks where each robot has to reach an open goal set $\mathcal{X}_i^G \subset \mathcal{X}$ and simultaneously maintain the state $x_i(t)$ inside a closed constraint set $\mathcal{X}_i^F \subseteq \mathcal{X}$. As an example, let $\Pi_i = \{p_G, p_F\}$ be the set of atomic propositions. The proposition p_G is true if $x_i \in \mathcal{X}_i^G$ and similarly, p_F is true if $x_i \in \mathcal{X}_i^F$. Consider an example task specification Φ_i expressed using the finite fragment of Linear Temporal Logic, (FLTL) [21] as $\Phi_i = \mathbf{F}\ p_G \wedge \mathbf{G}\ p_F$ where $\mathbf{F}$ is the eventually operator and $\mathbf{G}$ stands for the always operator. If the word formed by a trajectory $\sigma^{[i]}$ is such that for $w(\sigma^{[i]}) = w_0, w_1, \ldots,$ there exists some w_k such that $p_G \in w_k$ and $p_F \in w_i$ for all $i \geq 0$, we say that the word $w(\sigma^{[i]})$ satisfies the LTL formula Φ_i. Let us note that FLTL formulae such as those considered here can be automatically translated into automata. The word $w(\sigma^{[i]})$ satisfies the formula if it belongs to the language of the corresponding automaton. Please refer [5] for a more thorough exposition of these concepts. Denote by $[\Phi_i] \subseteq \Sigma_i$ the set of trajectories fulfilling Φ_i. Each robot then determines a trajectory belonging to $[\Phi_i]$.

In addition to finding a trajectory that satisfies these specifications, the robot may have several other objectives such as reaching in the goal region in the shortest possible time. To quantify these objectives, we define $\texttt{Cost} : \Sigma \to \mathbb{R}^p_{\geq 0}$ as the cost functional which maps each trajectory in $\Sigma \triangleq \bigotimes_{i \in \mathcal{V}_R} \Sigma_i$[1] to a non-negative cost vector and each component of $\texttt{Cost}$ corresponds to an objective of the robots. In what follows, we assume that $\texttt{Cost}$ is continuous. In addition, the robots want to avoid the inter-robot collisions; i.e., keeping the state $x(t)$ outside the collision set $\mathcal{X}_{\text{col}}$.

The above multi-robot motion planning problem is formulated as an open-loop non-cooperative game where each robot seeks to find a trajectory which is collision-free, fulfills its task specifications and minimizes the induced cost given the trajectories of other robots. That is, given $\sigma^{[-i]} \in \Sigma_{-i}$[2], each robot i wants to find a best trajectory in the feasible set $\texttt{Feasible}_i(\Sigma_i, \sigma^{[-i]}) \triangleq \{\sigma^{[i]} \in \Sigma_i \mid \sigma^{[i]} \in [\Phi_i], \quad \texttt{CollisionFreePath}(\sigma^{[i]}, \sigma^{[-i]}) = 1\}$ where the procedure $\texttt{CollisionFreePath}$ will be defined later. The solution notion we will use is Nash equilibrium formally stated as follows:

Definition 2.1 (Nash equilibrium): The collection of trajectories $(\bar{\sigma}^{[i]})_{i \in \mathcal{V}_R} \in \Sigma$ is a Nash equilibrium if for any $i \in \mathcal{V}_R$, it holds that $\bar{\sigma}^{[i]} \in \texttt{Feasible}_i(\Sigma_i, \bar{\sigma}^{[-i]})$ and there is no $\sigma^{[i]} \in \texttt{Feasible}_i(\Sigma_i, \bar{\sigma}^{[-i]})$ such that $\texttt{Cost}(\sigma^{[i]}) \prec \texttt{Cost}(\bar{\sigma}^{[i]})$[3].

Intuitively, none of the robots can decrease its cost by *unilaterally* deviating from a Nash equilibrium. Denote by $\Pi_{\text{NE}} \subseteq \Sigma$ the set of Nash equilibria. Note that Definition 2.1 is an extension of the standard one; e.g., in [8] where the cost functional of each player is scalar. We will compare Nash equilibrium with social (Pareto) optimum defined as follows:

Definition 2.2 (Social (Pareto) optimum): The collection of trajectories $(\bar{\sigma}^{[i]})_{i \in \mathcal{V}_R} \in \Sigma$ is socially (Pareto) optimal if there is no $(\sigma^{[i]})_{i \in \mathcal{V}_R} \in \Sigma$ such that $\bigoplus_{i \in \mathcal{V}_R} \texttt{Cost}(\sigma^{[i]}) \prec \bigoplus_{i \in \mathcal{V}_R} \texttt{Cost}(\bar{\sigma}^{[i]})$.[4]

Denote by $\Pi_{\text{SO}} \subseteq \Sigma$ the set of social optimum. Throughout this paper, we assume that $\Pi_{\text{SO}} \neq \emptyset$. In general, a Nash equilibrium may not be socially optimal. When $\texttt{Cost}$ is scalar, the gap between the set of Nash equilibrium and the set of social optimum is usually characterized by price of anarchy and price of stability in; e.g., [22].

A. Primitives

Here we define a set of primitives which will be used in the subsequent sections.

a) Sampling: The $\texttt{Sample}(A)$ procedure returns uniformly random samples from set A.

b) Local steering: Given two states x, y, the $\texttt{Steer}$ procedure returns a state z by steering x towards y for at most $\eta > 0$ distance; i.e., $\texttt{Steer}(x, y) \triangleq \text{argmin}_{z \in \mathbb{B}(x, \eta)} \|z - y\|$. In addition to this, we require that $\sigma(x, y)$, the trajectory connecting states x and y, is such that $|T(\sigma(x, y))| \leq 1$, i.e., the label $\lambda(\sigma(x, y))$ changes at most once. This property of the local steering function is called *trace inclusivity* [11].

[1] $\bigotimes$ represents the product.

[2] We use $-i$ to denote all the robots other than i.

[3] The relation $\prec$ is defined on $\mathbb{R}^p$ and given by: for $a, b \in \mathbb{R}^p$, $a \prec b$ if and only if $a_\ell \leq b_\ell$ for all $\ell \in \{1, \cdots, p\}$. Note that $\prec$ is a partial order on $\mathbb{R}^p$.

[4] $\bigoplus$ represents the summation.

c) Nearest neighbor: Given a state x and a finite set S of states, the `Nearest` procedure returns the state in S that is closest to x; i.e., $\texttt{Nearest}(S, x) \triangleq \text{argmin}_{y \in S} \|y - x\|$.

d) Near vertices: Given a state x, a finite set S and a positive real number r, the `NearVertices` procedure returns the states in S where each of them is r-close to x; $\texttt{NearVertices}(S, x, r) \triangleq \{y \in S \mid \|x - y\| \leq r\}$.

e) Path generation: Given a directed graph G with a single root and no directed cycles, the $\texttt{PathGeneration}(G)$ procedure returns the set of paths from the root to the leaf vertices.

f) Collision check of paths: Given a path σ and a set of paths Π, the $\texttt{CollisionFreePath}(\sigma, \Pi)$ procedure returns 1 if σ collides any path in Π; i.e., $\sigma(t) \in \mathcal{X}_{\text{free}} \triangleq \bigotimes_{i \in \mathcal{V}_R} \mathcal{X}_i^F \cap \overline{\mathcal{X}}_{\text{col}}$; otherwise returns 0.

g) Feasible paths: Given the path sets of Σ_i and $\sigma^{[-i]}$, $\texttt{Feasible}_i(\Sigma_i, \sigma^{[-i]})$ is the set of paths $\sigma^{[i]} \in \Sigma_i$ such that for any $\sigma^{[i]} \in \Sigma_i$, it holds that $\texttt{CollisionFreePath}(\sigma^{[i]}, \sigma^{[-i]}) = 1$.

h) Weakly feasible paths: Given the path sets of Σ_i and $\sigma^{[-i]}$, $\texttt{WeakFeasible}_i(\Sigma_i, \sigma^{[-i]})$ is a subset of $\texttt{Feasible}_i(\Sigma_i, \sigma^{[-i]})$ and consists of the paths $\sigma^{[i]}$ where for each path $\sigma^{[i]}$, there are a sequence of paths $\{\sigma_\ell^{[i]}\}$ with $\sigma_\ell^{[i]} \in \Sigma_i$ and a diminishing and non-negative sequence $\{\delta_\ell\}$ such that (i) $\sigma_\ell^{[i]}$ converges to $\sigma^{[i]}$; (ii) $\mathbb{B}(\sigma_\ell^{[i]}(t), \delta_\ell) \in \mathcal{X}_i^F$; (ii) $\|\sigma_\ell^{[i]}(t) - \sigma^{[j]}(t)\| \geq \epsilon + \delta_\ell$ for all $j \neq i$ for all t.

i) Strongly feasible paths: Given the path sets of Σ_i and $\sigma^{[-i]}$), $\texttt{StrongFeasible}_i(\Sigma_i, \sigma^{[-i]})$ is a subset of $\texttt{Feasible}_i(\Sigma_i, \sigma^{[-i]})$ and consists of the paths where for each path $\sigma^{[i]}$, there are a sequence of paths $\{\sigma_\ell^{[i]}\}$ with $\sigma_\ell^{[i]} \in \Sigma_i$, a diminishing and non-negative sequence $\{\delta_\ell\}$ and $\delta > 0$ such that (i) $\sigma_\ell^{[i]}$ converges to $\sigma^{[i]}$; (ii) $\mathbb{B}(\sigma_\ell^{[i]}(t), \delta_\ell) \in \mathcal{X}_i^F$; (ii) $\|\sigma^{[i]}(t) - \sigma^{[j]}(t)\| \geq \epsilon + \delta$ for all $j \neq i$.

III. iNash-trajectory Algorithm

In this section, we propose the iNash-trajectory Algorithm to solve the open-loop game defined above. It is followed by the algorithm analysis and discussion.

A. Algorithm statement

The iNash-trajectory Algorithm leverages the RRG algorithm in [16], iterative better response and model checking, and informally stated as follows. At each iteration k, each robot i samples $\mathcal{X}_i$ once, and adds the new sample $x_{\text{rand}}^{[i]}$ to its vertex set $V_k^{[i]}$. Robot i extends its previously generated graph $G_{k-1}^{[i]}$ towards the new sample $x_{\text{rand}}^{[i]}$ via local steering, and obtains the new graph $G_k^{[i]}$. After they finish the construction of the new graphs, the active robots play a game on their product graph for *one* round in a *sequential* way. Robot i is active at time k if its goal set is reachable through some path of $G_k^{[i]}$. The active robot with the least index, say i, first chooses a smaller-cost and feasible path on $G_k^{[i]}$ by performing the `BetterResponse` procedure in Algorithm 3. Then robot i informs all other robots the new path. After that, the active robot with the second least index, say j, performs the better response to update its path on $G_k^{[j]}$ and the new path is sent to other robots. The remaining active robots sequentially update the planned paths on their own graphs and announce the new paths to others. Once all the active robots finish the path updates, the game terminates for the current iteration k. At the next iteration $k+1$, the same steps are repeated. The iNash-trajectory Algorithm is formally stated in Algorithm 1.

Algorithm 1: The iNash-trajectory Algorithm

1 **for** $i = 1 : N$ **do**
2 $\quad V^{[i]}(0) \leftarrow x_{\text{init}}^{[i]}$;
3 $\quad E^{[i]}(0) \leftarrow \emptyset$;
4 $A_k \leftarrow \emptyset$;
5 $k \leftarrow 1$;
6 **while** $k < K$ **do**
7 $\quad$ **for** $i = 1 : N$ **do**
8 $\quad\quad x_{\text{rand}}^{[i]} \leftarrow \texttt{Sample}(\mathcal{X}_i)$;
9 $\quad\quad G_k^{[i]} \leftarrow \texttt{Extend}(G_{k-1}^{[i]}, x_{\text{rand}}^{[i]})$;
10 $\quad$ **for** $i \in \mathcal{V}_R \setminus A_{k-1}$ **do**
11 $\quad\quad$ **if** $V_k^{[i]} \cap \mathcal{X}_i^G \neq \emptyset$ **then**
12 $\quad\quad\quad A_k \leftarrow A_{k-1} \cup \{i\}$;
13 $\quad$ **for** $i \in A_k$ **do**
14 $\quad\quad \tilde{\sigma}_k^{[i]} = \sigma_{k-1}^{[i]}$;
15 $\quad$ **for** $i = 1 : N$ **do**
16 $\quad\quad$ **if** $i \in A_k$ **then**
17 $\quad\quad\quad \Pi_k^{[i]} \leftarrow \{\{\sigma_k^{[j]}\}_{j \in A_k, j<i}, \{\tilde{\sigma}_k^{[j]}\}_{j \in A_k, j>i}\}$;
18 $\quad\quad\quad \sigma_k^{[i]} \leftarrow \texttt{BetterResponse}(G_k^{[i]}, \Pi_k^{[i]})$;
19 $\quad k \leftarrow k + 1$;

The iNash-trajectory algorithm is an anytime algorithm; i.e., assuming a solution can be found within the allotted planning time, then it is continually improved while planning time remains

B. Discussion

The `Extend` procedure is similar to that in the RRG algorithm [16] with the difference that the edges leaving from the new sample are not added. Instead, $G_k^{[i]}$ is identical to the auxiliary graph G_n used in the proof of Theorem 38 in [16] for RRT*. Notice that $G_k^{[i]}$ is a directed graph and does not include any directed circle. So there are a finite number of paths for the root to reach any leaf vertex and the `PathGeneration` procedure in Algorithm 3 is well-defined.

The tree structure returned by RRT* in [16] is more computationally efficient than the graph $G_k^{[i]}$ in our algorithm. However, the rewiring step in RRT* induces that $G_{k-1}^{[i]}$ may not be a subgraph of $G_k^{[i]}$. This property is crucial for the algorithm convergence to Nash equilibria in the next section. To verify if $(\sigma^{[i]} \cap G_k^{[i]}) \in [\Phi_i]$ on Line 4 of the `BetterResponse` procedure, we check if the sequence $\xi_i = (\sigma^{[i]} \cap G_k^{[i]})$ satisfies Φ_i by translating Φ_i into the corresponding Buchi automaton.

C. Analysis

In this section, we analyze the asymptotic optimality, computational complexity, and communication cost of the

Algorithm 2: The `Extend` Procedure

$V \leftarrow V_{k-1}^{[i]}$;
$E \leftarrow E_{k-1}^{[i]}$;
$x_{\text{nearest}} \leftarrow \texttt{Nearest}(E, x_{\text{rand}}^{[i]})$;
$x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}}^{[i]})$;
if $\texttt{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
  $\mathcal{X}_{\text{near}} \leftarrow \texttt{NearVertices}(E, x_{\text{new}}, \min\{\gamma(\frac{\log k}{k})^{\frac{1}{n}}, \eta\})$;
  $V \leftarrow V \cup \{x_{\text{new}}\}$;
  for $x_{\text{near}} \in \mathcal{X}_{\text{near}}$ **do**
    if $\texttt{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
      $E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}})\}$;
return $G = (V, E)$

Algorithm 3: The `BetterResponse` Procedure

$\mathbb{P}_k^{[i]} \leftarrow \texttt{PathGeneration}(G_k^{[i]})$;
$\mathbb{P}_{\text{f}}^{[i]} \leftarrow \emptyset$;
for $\sigma^{[i]} \in \mathbb{P}_k^{[i]}$ **do**
  if $\texttt{CollisionFreePath}(\sigma^{[i]}, \Pi_k^{[i]}) ==$
  1 && $(\sigma^{[i]} \cap G_k^{[i]}) \in [\Phi_i]$ **then**
    $\mathbb{P}_{\text{f}}^{[i]} \leftarrow \mathbb{P}_{\text{f}}^{[i]} \cup \{\sigma^{[i]}\}$;
$\sigma_{\min}^{[i]} \leftarrow \sigma_{k-1}^{[i]}$;
for $\sigma^{[i]} \in \mathbb{P}_{\text{f}}^{[i]}$ **do**
  if $\texttt{Cost}(\sigma^{[i]}) \prec \texttt{Cost}(\sigma_{\min}^{[i]})$ **then**
    $\sigma_{\min}^{[i]} \leftarrow \sigma^{[i]}$;
    Break;
return $\sigma_{\min}^{[i]}$

iNash-trajectory Algorithm. Before doing that, we first prove the existence of Nash equilibrium.

Lemma 3.1 (Existence of Nash equilibrium): It holds that $\Pi_{\text{SO}} \subseteq \Pi_{\text{NE}}$ and Π_{NE} is non-empty.

Let $\hat{\Sigma}$ be the set of limit points of $\{\sigma_k^{[i]}\}_{i \in \mathcal{V}_R}$. We are ready to show the convergence of the iNash-trajectory Algorithm.

Theorem 3.1 (Asymptotic optimality): There is $\bar{c}_\ell^{[i]} \geq 0$ such that $\texttt{Cost}_\ell(\bar{\sigma}^{[i]}) = \bar{c}_\ell^{[i]}$ for any $\{\bar{\sigma}^{[i]}\}_{i \in \mathcal{V}_R} \in \hat{\Sigma}$. In addition, any limit point $\{\bar{\sigma}^{[i]}\}_{i \in \mathcal{V}_R}$ is a Nash equilibrium.

Next, we will analyze the computational complexity of the algorithm in terms of the `CollisionFreePath` procedure. Let θ_n to be the total number of calls to the `CollisionFreePath` procedure at iteration n.

Lemma 3.2 (Computational complexity): It holds that $\theta_n \leq \bigoplus_{i \in \mathcal{V}_R} |\mathbb{P}_k^{[i]}|$, where $\mathbb{P}_k^{[i]}$ is defined in Algorithm 3.

In Lemma 3.2, the quantity $|\mathbb{P}_k^{[i]}|$ is independent of the robot number. So the worst computational complexity of the iNash-trajectory grows linearly in the robot number. It is in contrast to the exponential dependency in centralized path planning. The computational efficiency comes with the non-cooperative game theoretic formulation where each robot myopically responds to others. Note that a Nash equilibrium may not be socially optimal for the robot team.

Let ϑ_n to be the number of exchanged paths in iteration n. The following lemma shows the worst communication cost is linear in the robot number.

Lemma 3.3 (Communication cost): $\vartheta_n \leq 2N$.

Algorithm 4: The iOptimalControl Algorithm

for $i = 1 : N$ **do**
  $V^{[i]}(0) \leftarrow x_{\text{init}}^{[i]}$;
  $E^{[i]}(0) \leftarrow \emptyset$;
$A_k \leftarrow \emptyset$;
$k \leftarrow 1$;
while $k < K$ **do**
  for $i = 1 : N$ **do**
    $x_{\text{rand}}^{[i]} \leftarrow \texttt{Sample}(\mathcal{X}_i)$;
    $G_k^{[i]} \leftarrow \texttt{Extend}(G_{k-1}^{[i]}, x_{\text{rand}}^{[i]})$;
  for $i \in \mathcal{V}_R \setminus A_{k-1}$ **do**
    if $V_k^{[i]} \cap \mathcal{X}_i^G \neq \emptyset$ **then**
      $A_k \leftarrow A_{k-1} \cup \{i\}$;
  $(\sigma_k^{[i]})_{i \in A_k} \leftarrow \texttt{OptimalTrajectory}(\bigotimes_{i \in A_k} G_k^{[i]})$;
  $k \leftarrow k + 1$;

Algorithm 5: The `OptimalTrajectory` Procedure

for $i \in A_k$ **do**
  $\mathbb{Q}_k^{[i]} \leftarrow \texttt{PathGeneration}(G_k^{[i]})$;
for $i \in A_k$ **do**
  $\mathbb{P}_{\text{f}}^{[i]} \leftarrow \emptyset$;
  for $\sigma^{[i]} \in \mathbb{Q}_k^{[i]}$ **do**
    if $\texttt{CollisionFreePath}(\sigma^{[i]}, \mathbb{Q}_k^{[-i]}) ==$
    1 && $(\sigma \cap \bigotimes_{i \in A_k} G_k^{[i]}) \in [\Phi]$ **then**
      $\mathbb{P}_{\text{f}}^{[i]} \leftarrow \mathbb{P}_{\text{f}}^{[i]} \cup \{\sigma^{[i]}\}$;
$\sigma_{\min} \leftarrow \texttt{Sample}(\bigotimes_{i \in A_k} \mathbb{P}_{\text{f}}^{[i]})$;
for $\sigma \in \bigotimes_{i \in A_k} \mathbb{P}_{\text{f}}^{[i]}$ **do**
  if $\bigoplus_{i \in A_k} \texttt{Cost}(\sigma^{[i]}) \prec \bigoplus_{i \in A_k} \texttt{Cost}(\sigma_{\min}^{[i]})$ **then**
    $\sigma_{\min} \leftarrow \sigma$;
return $\sigma_{\min}$

D. Comparison

In order to demonstrate the scalability of iNash-trajectory Algorithm, we consider the benchmark algorithm, the iOptimalControl Algorithm. The key difference between the iOptimalControl and iNash-trajectory Algorithms is that a centralized authority in the iOptimalControl Algorithm determines a social optimum on the product graph at each iteration. In the iOptimalControl Algorithm, we use the notation $(\sigma \cap \bigotimes_{i \in A_k} G_k^{[i]}) \in [\Phi]$ for $(\sigma^{[i]} \cap G_k^{[i]}) \in [\Phi_i]$ for all $i \in A_k$.

The following theorem guarantees the asymptotic optimality of the iOptimalControl Algorithm.

Theorem 3.2 (Asymptotic optimality): Any limit point $\{\bar{\sigma}^{[i]}\}_{i \in \mathcal{V}_R}$ is a social optimum.

Next, we will analyze the computational complexity of the algorithm in terms of the `CollisionFreePath` procedure. Let θ'_n to be the total number of calls to the `CollisionFreePath` procedure at iteration n.

Lemma 3.4 (Computational complexity): It holds that $\theta'_n = \bigotimes_{i \in \mathcal{V}_R} |\mathbb{Q}_k^{[i]}|$, where $\mathbb{Q}_k^{[i]}$ is defined in Algorithm 5.

The above lemma shows that the computational complexity exponentially grows vs. robot number. Table I summarizes the comparison of the iOptimalControl and iNash-trajectory

Algorithms where the prices of anarchy and stability are compared for the case $p = 1$. In particular, the price of stability (POS) [22] is the ratio between the minimum additive cost function value in Π_{NE} and that of one in Π_{SO}, and defined as follows:

$$\mathrm{POS} = \frac{\inf_{\sigma \in \Pi_{\mathrm{NE}}} \bigoplus_{i \in \mathcal{V}_R} \mathtt{Cost}(\sigma^{[i]})}{\bigoplus_{i \in \mathcal{V}_R} \mathtt{Cost}(\bar{\sigma}^{[i]})},$$

for any $\bar{\sigma} \in \Pi_{\mathrm{SO}}$. By Lemma 3.1, we know $\Pi_{\mathrm{SO}} \subseteq \Pi_{\mathrm{NE}}$ and thus POS is equal to 1. On the other hand, the price of anarchy (POA) [22] is the ratio between the maximum additive cost function value in Π_{NE} and that of one in Π_{SO}, and given by:

$$\mathrm{POA} = \frac{\sup_{\sigma \in \Pi_{\mathrm{NE}}} \bigoplus_{i \in \mathcal{V}_R} \mathtt{Cost}(\sigma^{[i]})}{\bigoplus_{i \in \mathcal{V}_R} \mathtt{Cost}(\bar{\sigma}^{[i]})},$$

for any $\bar{\sigma} \in \Pi_{\mathrm{SO}}$. The value of POA depends on a number of factors; e.g., the obstacle locations, the dynamic systems and so on. It is interesting to find a lower bound on Π_{SO} given more information as in; e.g., [14], and utilize mechanism design to eliminate the price of anarchy.

TABLE I: The comparison of the iOptimalControl and iNash-trajectory Algorithms

	iOptimalControl	iNash-trajectory
Solution Notion	Social optimum	Nash equilibrium
Solution Feasibility	Yes	Yes
Price of stability	N/A	One
Price of anarchy	N/A	Unknown
Coordination	High	Low
Asymptotic optimality	Yes	Yes
Computational complexity	Exponential	Linear

IV. Experiments

We perform two experiments in simulation to evaluate the performance of iNash. The first involves 8 circular robots moving in an environment with randomly generated obstacles (Figure 2-left), while the second involves 6 robots in a traffic intersection scenario (Figure 2-right); both involve state spaces consisting of first order dynamics and time. Robots are holonomic discs with radii of 0.5 meters.

We compare iNash to two prioritized methods that are not guaranteed to return a Nash-Equilibrium, but are arguably similar to our proposed algorithm. The first is the standard prioritized approach from [12]. Each robot builds its own random graph; then robots select their paths in order such that the path of robot i does not conflict with robots $1, \ldots, i-1$. The second is an any-time version of the prioritized method. Each time robot i finds a better path that does not conflict with the paths of robots $1, \ldots, i-1$, then for $j = i+1, i+2, \ldots$ (in order) robot j must choose a new path that does not conflict with robots $1, \ldots, j-1$. This differs from our algorithm (where new paths must respect the paths of all other robots), and the solution is not guaranteed to converge to a Nash Equilibrium.

For experiments we consider the task specifications for each robot to be of the form $\Phi_i = \mathbf{F}\ p_{G_i} \wedge \mathbf{G}\ p_F$, i.e., each robot tries to reach a different goal region in the shortest possible distance while respecting the same constraint set $\mathcal{X}^F$. The automaton consists of two states (see Fig. 1).

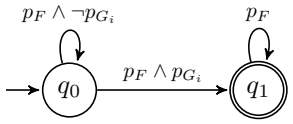

Fig. 1: Automaton for $\Phi_i = \mathbf{F}\ p_{G_i} \wedge \mathbf{G}\ p_F$.

Discussion of Experimental Results: Experimental results are summarized in tables II-III. In iNash all robots tend to bear the burden of conflict resolution similarly, on average. This contrasts with the prioritized methods, in which robot's with lower IDs have shorter paths and reach the goal more frequently than robots with higher IDs. The result that some iNash paths are longer than the prioritized paths is expected, given that in iNash robots act in their own self interest.

V. Conclusions

This paper discusses a class of multi-robot motion planning problems where each robot is associated with multiple-objectives and independent class specifications. We formulated the problem as an open-loop, non-cooperative differential game and proposed a distributed, anytime algorithm to compute the Nash equilibrium. Techniques from Rapidly-exploring Random Graphs and iterative better response were used to provide convergence guarantees and analyse the price of stability as well as the communication cost of the algorithm. We also presented results of simulation experiments that demonstrate the efficiency and anytime nature of the algorithm. Future directions include coupled task specifications of robots and algorithms which can eliminate the price of anarchy.

References

[1] A. Arsie, K. Savla, and E. Frazzoli. Efficient routing algorithms for multiple vehicles with no explicit communications. *IEEE Transactions on Automatic Control*, 54(10):2302–2317, 2009.

[2] G. Arslan, J. R. Marden, and J. S. Shamma. Autonomous vehicle-target assignment: A game theoretic formulation. *ASME Journal on Dynamic Systems, Measurement, and Control*, 129(5):584–596, 2007.

[3] J.P. Aubin. *Viability theory*. Springer, 2009.

[4] J.P. Aubin, A. Bayen, and P. Saint-Pierre. *Viability theory: New directions*. Springer-Verlag, New York, 2011.

[5] Christel Baier, Joost-Pieter Katoen, et al. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.

[6] M. Bardi and I. Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Birkhäuser, 1997.

[7] M. Bardi, M. Falcone, and P. Soravia. Numerical methods for pursuit-evasion games via viscosity solutions. *Annals of the International Society of Dynamic Games*, 4:105 – 175, 1999.

[8] T. Basar and G. Olsder. *Dynamic noncooperative game theory*. SIAM Classics in Applied Mathematics, 1999.

[9] S. J. Buckley. Fast motion planning for multiple moving robots. In *IEEE International Conference on Robotics and Automation*, pages 322–326, May 1989.

[10] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre. Set-valued numerical analysis for optimal control and differential games. *Annals of the International Society of Dynamic Games*, 4(1):177 – 247, 1999.

[11] Luis I Reyes Castro, Pratik Chaudhari, Jana Tumova, Sertac Karaman, Emilio Frazzoli, and Daniela Rus. Incremental sampling-based algorithm for minimum-violation motion planning. In *Proc. of 52nd IEEE Conference on Decision and Control*, 2013.

[12] M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica*, 2(6):477–521, 1987.

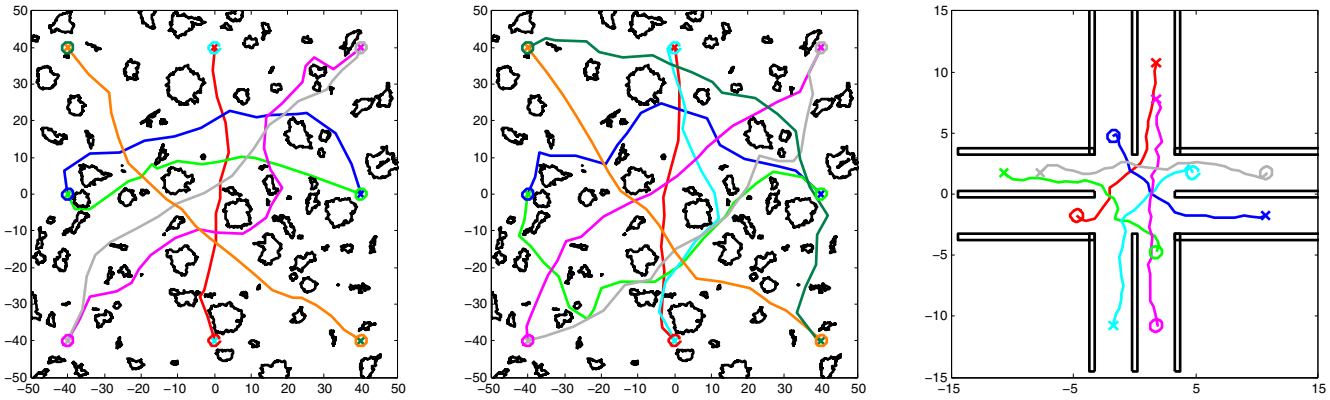

Fig. 2: Experimental environments. Obstacles are outlined in black, paths are colored lines. Robots start at 'O's and end at 'X's ('O's are drawn 3x the robot radii to help visualization). **Left/Center:** 8 robots in a randomly generated environment; the Nash Equilibrium in the left trial allows 6 of 8 robots to reach their goals, while all 8 reach their goals in the center trial. **Right:** 6 Robots at an intersection and the paths corresponding to a Nash equilibrium where all robots reach their goals

TABLE II: Experimental Results, Random Environment

	Mean path length over 20 trials (ratio vs. socially optimal length)								Total times reached goal (of 20)							
Algorithm	Robot ID								Robot ID							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
iNash (any-time)	1.295	1.343	1.324	1.301	1.166	1.293	1.224	1.202	20	19	18	20	16	15	18	17
Prioritized	1.084	1.149	1.263	1.316	1.228	1.326	1.312	1.349	20	20	18	19	20	18	18	15
Prioritized (any-time)	1.081	1.129	1.153	1.200	1.126	1.168	1.163	1.204	20	20	18	20	19	20	20	20

TABLE III: Experimental Results, Intersection Environment

	Mean path length over 20 trials (ratio vs. socially optimal length)						Total times reached goal (of 20)					
Algorithm	Robot ID						Robot ID					
	1	2	3	4	5	6	1	2	3	4	5	6
iNash (any-time)	1.168	1.228	1.245	1.243	1.201	1.166	11	14	12	14	16	15
Prioritized	1.107	1.238	1.384	1.492	1.339	1.300	19	18	19	13	12	4
Prioritized (any-time)	1.085	1.212	1.165	1.248	1.136	1.251	19	18	20	19	20	15

[13] R. Isaacs. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. Dover, 1999.

[14] R. Johari and J.N. Tsitsiklis. Efficiency loss in a network resource allocation game. *Mathematics of Operations Research*, 29(3):407–435, 2004.

[15] K. Kant and S.W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research*, 5(3):72 – 89, 1986.

[16] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846 – 894, 2011.

[17] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning with deterministic mu-calculus specifications. In *2012 American Control Conference*, pages 735–742, Montréal, Canada, December 2012.

[18] J.J. Kuffner and S.M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *IEEE Conference on Robotics and Automation*, pages 995–1001, 2000.

[19] S.M. LaValle and S.A. Hutchinson. Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation*, 14(6):912–925, 1998.

[20] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[21] Zohar Manna. *Temporal verification of reactive systems: safety*, volume 2. Springer, 1995.

[22] N. Nisan, T. Roughgarden, E. Tardos, and V.V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[23] J.H. Reif. Complexity of the mover's problem and generalizations. In *The 20th Annual IEEE Conference on Foundations of Computer Science*, pages 421–427, 1979.

[24] G. Sanchez and J.C. Latombe. On delaying collision checking in prm planning – application to multi-robot coordination. *The International Journal of Robotics Research*, 21:5 – 26, 2002.

[25] J.A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Science*. Cambridge University Press, 1996.

[26] T. Simeon, S. Leroy, and J.-P. Lauumond. Path coordination for multiple mobile robots: a resolution-complete algorithm. *IEEE Transactions on Robotics and Automation*, 18(1):42 – 49, 2002.

[27] P.E. Souganidis. Two-player, zero-sum differential games and viscosity solutions. *Annals of the International Society of Dynamic Games*, 4(1):69 – 104, 1999.

[28] E.K. Xidias and N.A. Aspragathos. Motion planning for multiple non-holonomic robots: a geometric approach. *Robotica*, 26:525–536, 2008.

[29] M. Zhu and S. Martínez. Distributed coverage games for energy-aware mobile sensor networks. *SIAM Journal on Control and Optimization*, 51(1):1–27, 2013.

[30] M. Zhu, M. Otte, P. Chaudhari, and E. Frazzoli. Distributed anytime game-theoretic learning for multi-robot motion planning - Part I : Trajectory based algorithms. http://arxiv.org/abs/1402.2708.