GAMECHAT: Multi-LLM Dialogue for Safe, Agile, and Socially Optimal Multi-Agent Navigation in Constrained Environments

Vagul Mahadevan, Shangtong Zhang, and Rohan Chandra
{dub5nq, xdm2bt, aar8xx}@virginia.edu, University of Virginia
Code, Videos, Proofs at **gamechat-uva.github.io/**

***Abstract*— Safe, agile, and socially compliant multi-robot navigation in cluttered and constrained environments remains a critical challenge. This is especially difficult with self-interested agents in decentralized settings, where there is no central authority to resolve conflicts induced by spatial symmetry. We address this challenge by proposing a novel approach, GAMECHAT, which facilitates safe, agile, and deadlock-free navigation for both cooperative and self-interested agents. Key to our approach is the use of natural language communication to resolve conflicts, enabling agents to prioritize more urgent tasks and break spatial symmetry in a socially optimal manner. Our algorithm ensures subgame perfect equilibrium, preventing agents from deviating from agreed-upon behaviors and supporting cooperation. Furthermore, we guarantee safety through control barrier functions and preserve agility by minimizing disruptions to agents' planned trajectories. We evaluate GAMECHAT in simulated environments with doorways and intersections. The results show that even in the worst case, GAMECHAT reduces the time for all agents to reach their goals by over 35% from a naive baseline while doubling the rate of ensuring the agent with a higher priority task reaches the goal first, from 50% (equivalent to random chance) to a 100% perfect performance at maximizing social welfare.**

I. INTRODUCTION

It is challenging for robots to plan safe, agile, and deadlock-free (situations where no robot can move toward its goal for a few seconds) trajectories in cluttered and constrained environments. First, in *decentralized* systems, we have no central authority that coordinates agents in a manner that deadlocks will be prevented or resolved. Second, with self-interested agents [1] that have conflicting objectives, we must ensure that *jerk-agent behavior* (jerk agents may break previously agreed-upon consensuses or socially compliant protocols leading to unsafe behavior) is not incentivized. Third, *symmetry* between the agents (which occurs when they are the same distance from a shared collision point and have the same velocity) must be broken in a socially optimal manner. If both move forward at the same speed they will collide, and if neither moves, they will deadlock.

A. Main Contributions

We propose a new algorithm for safe, agile, deadlock-free, and decentralized multi-robot navigation in symmetric, constrained environments (like passing through doorways and narrow hallways). Our algorithm works for both cooperative *and self-interested* agents, that is, when agents choose to optimize their own objectives. Key to this approach is a novel LLM-based communication module in which agents *automatically* engage in a natural language dialogue with each other to proactively resolve any conflicts that could arise. Any conflict resolution is then executed via a low-level dynamic game-theoretic controller. Our method, which we call GAMECHAT, has the following properties:

- **Subgame-Perfect Optimality:** GAMECHAT includes a game-theoretic strategy which yields a subgame perfect equilibrium [2] (a guarantee that we have a Nash equilibrium now and at all future times), so self-interested agents will choose to commit to our strategy, as there is no incentive to deviate from it now or in the future.
- **Welfare-Maximizing (socially optimal):** GAMECHAT is socially optimal–agents break symmetry by prioritizing more urgent tasks, maximizing social welfare.
- **Safety:** GAMECHAT uses control barrier functions (CBFs) [3], to guarantee safe trajectories.
- **Agility:** GAMECHAT is minimally invasive, that is, the speeds of the agents are decreased as little as possible, and the agents do not spatially deviate from their desired paths, preserving smooth trajectories.

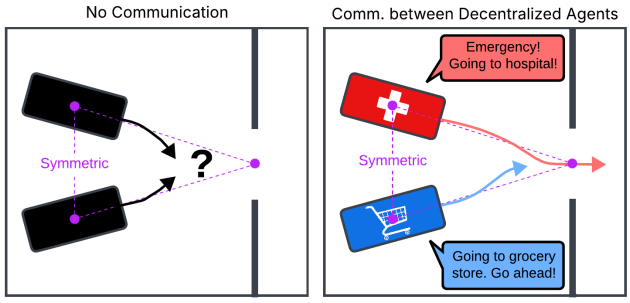

Fig. 1. Two agents head toward a **hospital** and a **grocery store** in a symmetric, constrained environment. In the left image, there is **no communication** between the agents, causing a deadlock as the agents do not know which should go first, and in the right image, GAMECHAT uses natural language communication between decentralized agents to identify their roles, thereby resolving the deadlock by prioritizing urgent tasks.

We choose natural language as for communication as it does not require a defined protocol. It also allows communication between robots and humans, which is crucial for a future where humans and robots live and work together.

II. RELATED WORKS

Collision Avoidance: To guarantee safety, one can use control barrier functions (CBFs) [3], [4] which use the forward invariance of a set. If an agent is in a safe set at some time, then it will remain in that safe set for all future times. In MPC-CBF [5], constraints are derived using CBFs and added to the receding-horizon controller to prevent collisions.
Deadlock Resolution: Symmetry between agents can result in deadlocks [6]. The simplest way to break symmetry is to rely on the environment's randomness or to randomly perturb agents [4], but this can increase total cost. Heuristic-based approaches, such as a right-hand rule [7], [8] improve over random perturbations but are more centralized. Some methods design an auction mechanism [9], [10], [11], and agents bid to cross the intersection using some bidding strategy. In reservation-based systems [12], agents must reserve slots to cross the intersection determined by the time to arrival.
Learning-based Methods: Deep reinforcement learning (DRL) has been used for multi-robot decentralized collision avoidance with local sensing [13]. Inverse reinforcement learning (IRL) has been used to infer reward functions of other agents for planning in dense crowds [14]. However,

these methods have difficulty encoding safety constraints and in out-of-distribution domains. LiveNet [15] is a recent approach that encodes safety and agility via differentiable CBFs within a neural network controller. This allows for learnable multi-robot navigation in constrained spaces.

Game-theoretic Methods: For multi-agent planning with self-interested agents, methods have been developed for distributed optimization in differential games. These algorithms solve for Nash equilibria. Some are similar to direct methods in trajectory optimization [16], [17] like Newton's method. They handle state and control input constraints and converge quickly. These algorithms yield analytical solutions that guarantee safety but not agility, and they require knowledge of the other agents' cost functions and policies/actions.

Multi-Robot Communication: In decentralized, cooperative multi-robot environments, communication methods are needed for coordination [18]. Graph Neural Networks (GNNs) have been used to communicate the features of local observations among a network of robots [19]. DMCA [20] uses RL to learn selective communication to share goal information with relevant neighbors. Further, Arul et al. [21] created an RL method that learns what information is important to communicate and when to communicate it, reducing indiscriminate broadcasting.

LLMs for Robotics: Recently, some methods use LLMs or Vision-Language Models (VLMs) in planning. Progprompt [22] uses LLMs to generate the code for robotic policies. On the multi-agent side, Garg et al. [23] use LLMs and VLMs to resolve a deadlock after it occurs by identifying a leader agent to move first. One method [24] uses VLMs to detect people and assign scores to trajectories, which induces socially compliant navigation. Most similar to our approach is RoCo [25], where robots are each equipped with an LLM instance and communicate in a natural language dialogue to agree on a plan of action. Note, however, that RoCo is fully cooperative while we investigate self-interested agents.

III. PROBLEM FORMULATION

In this section, we define a general problem formulation. We begin with a modified Partially Observable Stochastic Game (POSG) [26], defined by the tuple: $\langle k, T, \mathcal{X}, \{\mathcal{U}^i\}, \mathcal{T}, \Omega, \{\mathcal{O}^i\}, \{\mathcal{R}^i\}\rangle$. k is the number of agents, T is the finite number of time steps in the game, and $\mathcal{X}$ is the continuous state space. The superscript $i \in \{1, \dots, k\}$ refers to the ith agent and the subscript $t \in \{0, \dots, T\}$ refers to time step t; e.g., $\mathbf{x}_t^i \in \mathcal{X}$ is the state of agent i at time t. Each agent i has a start state $\mathbf{x}_0^i$ and a set of goal states $\mathcal{X}_g \subset \mathcal{X}$. The game ends if all agents have reached their goals or T time steps have elapsed, whichever is sooner. For an agent i, $\mathcal{U}^i$ is the continuous control space containing feasible inputs. The dynamics function $\mathcal{T} : \mathcal{X} \times \mathcal{U}^i \to \mathcal{X}$ determines the state of an agent at time $t+1$ given its state and control input at time t. The set Ω is the observation space, and whenever agent i arrives at a new state, the observation function $\mathcal{O} : \mathcal{X} \to \Omega$ yields a local observation of its own and nearby agents' states. We can define a trajectory of agent i by $\Gamma^i = (\mathbf{x}_0^i, \dots, \mathbf{x}_T^i)$ and an input sequence by $\Psi^i = (u_0^i, \dots, u_{T-1}^i)$. For any time t, $C^i(\mathbf{x}_t^i) \subseteq \mathcal{X}$ is the space occupied by agent i and two robots i, j are in a collision if $C^i(\mathbf{x}_t^i) \cap C^j(\mathbf{x}_t^j) \neq \emptyset$. Each agent i strives to maximize its payoff, $\mathcal{R}^i(\Gamma^i, \Gamma^{-i})$, which represents the payoff at the end of the game for agent i if they play Γ^i and all other players take the trajectories $\Gamma^{-i} = (\Gamma^1, \dots, \Gamma^{i-1}, \Gamma^{i+1}, \dots, \Gamma^k)$. Agents are rational, meaning that they will avoid collisions at all costs, and among collision-free trajectories prefer the one which minimizes the time-to-goal.

We now define what we call a social mini-game.

Definition III.1. *A **social mini-game** (SMG) is a type of POSG. Each agent has a desired trajectory $\widetilde{\Gamma}^i$ which is what the agent would follow if it were the only agent. The crucial property that makes an SMG is that there is some time t and pair of distinct agents i, j where $C^i(\mathbf{x}_t^i) \cap C^j(\mathbf{x}_t^j) \neq \emptyset$ and $\mathbf{x}_t^i \in \widetilde{\Gamma}^i, \mathbf{x}_t^j \in \widetilde{\Gamma}^j$; i.e., following the desired trajectories would cause the agents to collide at some point in time.*

Additionally, each agent is assigned a social priority and we want to maximize social welfare [27], defined as:

Definition III.2. ***Social welfare** is defined by $\mathcal{W} = \sum_{i=1}^{k} \frac{p_i}{\tau^i(\Gamma^i)}$. $\tau^i(\Gamma^i)$ is the time-to-goal for agent i taking trajectory Γ^i, and $p^i \in \mathbb{R}^+$ is the **social priority** of agent i, expressing how important it is for agent i to reach its goal.*

Note that if $p^i > p^j$ and $\tau^i(\Gamma^i) = \tau^j(\Gamma^j)$, reducing agent i's time-to-goal would more positively impact social welfare than reducing agent j's time-to-goal by the same amount.

Next, agents will have to deviate from their desired trajectory to avoid collisions and deadlocks, but we also want to modify the desired trajectory as little as possible:

Definition III.3. *The modified trajectory $\Gamma^{i,*}$ is **minimally invasive** if it satisfies the following two properties:*

1) *$\Delta\theta_t^i = 0$ for all t (at all times, the heading of the robot does not deviate from the desired trajectory).*
2) *$\min_t |v_t^i|$ is maximized (the robot slows down as little as is necessary to prevent a collision or deadlock).*

Overall, given an SMG, our goal is to generate trajectories that are safe, deadlock-free, welfare-maximizing, and minimally invasive. We achieve this with GAMECHAT.

IV. METHODOLOGY

Here, we describe GAMECHAT in detail. First, we discuss technical details of our environment and give an overview of the approach. Next, we present how we implemented the LLM dialogue between agents for prioritizing tasks. Finally, we explain our game-theoretic control strategy and prove that it results in a subgame perfect equilibrium.

A. Environment Details

Our environment contains two agents running single-integrator unicycle dynamics. The state $(x, y, \theta) \in \mathcal{X}$ consists of 2D position and heading, and the control inputs $(v, \omega) \in \mathcal{U}^i$ consist of both linear and angular velocity. Ultimately, each agent i is trying to maximize its payoff $\mathcal{R}^i$. We implement this by defining a cost function $\mathcal{J}^i$ and using Model Predictive Control (MPC) to solve the following receding horizon optimization problem:

$$\left(\Gamma^{i,*}, \Psi^{i,*}\right) = \underset{(\Gamma^i, \Psi^i)}{\operatorname{argmin}} \sum_{t=0}^{T-1} \mathcal{J}^i(\mathbf{x}_t^i, u_t^i) + \mathcal{J}_T^i(x_T^i) \tag{1a}$$

$$\text{s.t. } \mathbf{x}_{t+1}^i = \mathcal{T}(\mathbf{x}_t^i, u_t^i), \forall t \in \{0, \dots, T-1\} \tag{1b}$$

$$C^i(\mathbf{x}_t^i) \cap C^j(\mathbf{x}_t^j) \neq \emptyset, \forall t \tag{1c}$$

$$u_{\min} \leq u_t^i \leq u_{\max}, \forall t \tag{1d}$$

$$\mathbf{x}_T^i \in \mathcal{X}_g^i. \tag{1e}$$

At each time step, agent i (though the function $\mathcal{O}^i$) observes the position, heading, and velocity of the other

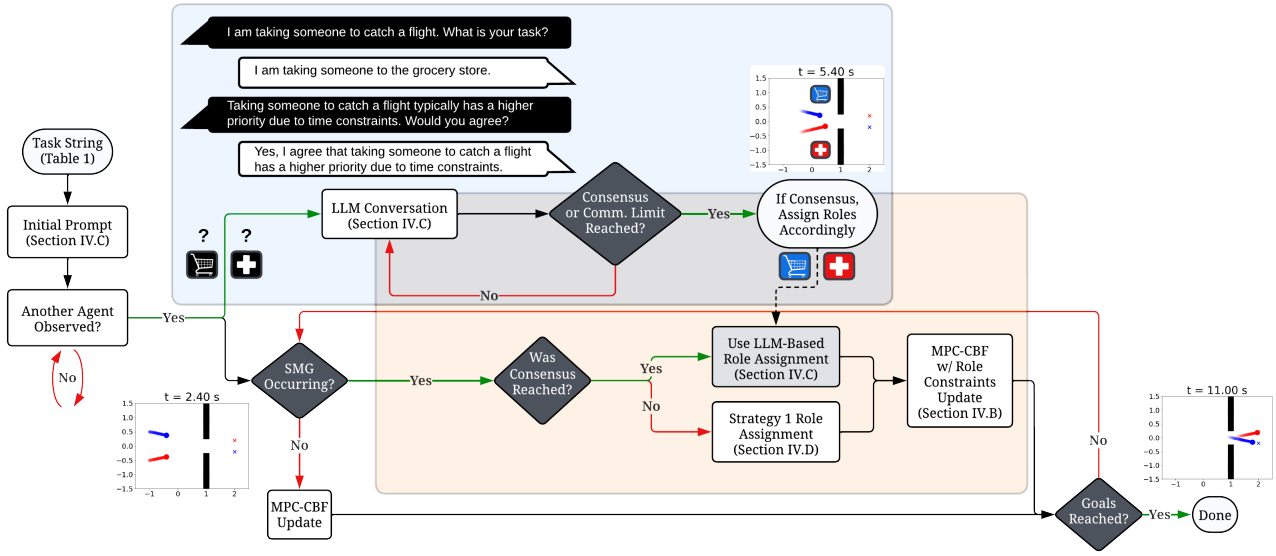

Fig. 2. Flow chart describing the logical flow of GAMECHAT. Blue box represents nodes involved in our novel LLM-based communication module. Orange box represents nodes handling a social mini-game (some LLM nodes are partially covered, representing that the agent may or may not be in those nodes during an SMG, as the communication could finish before an SMG begins or occur concurrently).

agent along with the position of the obstacles in the environment. The other agent and obstacles are treated as circular and control barrier functions are created for each one. The inequalities generated from these CBFs are added in the next cycle of MPC as collision avoidance constraints (we refer the reader to [18] for a detailed background on CBFs).

Our social mini-game $\mathcal{S}$ has a structure where both agents have a straight-line desired trajectory to their respective goals. Both agents must pass through the collision point $\mathcal{Q}$, which models a doorway or the central point of a tight intersection. When both agents detect the existence of $\mathcal{S}$, each agent i is a distance d_i from $\mathcal{Q}$. We model them as thin, car-like objects of length l and zero width. Both agents also share the same velocity constraint $v_{\max}$.

B. Technical Approach

Our overall technical approach is diagrammed in Figure 2. First, at the earliest time t when the agents observe each other (i.e. $\mathbf{x}_t^j \in \mathcal{O}^i(\mathbf{x}_t^i)$ and $\mathbf{x}_t^i \in \mathcal{O}^j(\mathbf{x}_t^j)$), the agents begin sending messages back and forth until consensus is reached or enough messages are sent without an agreement. Based on the consensus, the agents assign themselves *leader* or *follower* roles, determining which agent will pass through $\mathcal{Q}$ first. If an SMG is detected before the dialogue comes to a consensus (or if no agreement is reached), the agents will default to and continue executing Strategy 1.

At each time step, the agents check to see if they are in an SMG (equivalent to detecting an imminent collision). If roles were not already assigned through communication, the agents will assign themselves roles using Strategy 1 until a consensus is reached, based on which they may change their roles. The leader moves at $v_{\max}$ toward the goal and the follower slows down and reaches $\mathcal{Q}$ as the leader leaves it. We enforce the follower's slowdown by lowering the linear velocity upper bound (1d) from $v_{\max}$ to $\frac{d_i v_{\max}}{l+d_j}$.

Theorem IV.1. GAMECHAT *yields minimally invasive trajectories (see Definition III.3).*

C. LLMs for Priority Determination

Each agent has access to an LLM (`gpt-4o-mini` [28]) instance through the OpenAI API and a task in natural language by way of an initial prompt. We tested with three types of agents. In decreasing order of priority they are: hospital agents, airport agents, and grocery agents. When the agents observe each other, they take turns sending messages; they receive the last message from the other agent, add it to the dialogue history, and query the LLM (since we are using the OpenAI API, there is network latency, but each message is generated within one second) for a reply to send back. The robots keep moving around the environment while communicating and will stop conversing if they reach a consensus as to which agent's task has the higher priority or each robot has sent four messages with no agreement. An example conversation is shown in Figure 2.

If the agents come to a consensus, knowing which agent's task has a higher priority gives a way to break symmetry. It also allows an agent with a slightly higher TTQ reach the goal first, which would maximize social welfare. We ensured the agents had distinct types instead of attempting to rank the priority of different task strings within the same category.

Theorem IV.2. *In a symmetric social mini-game ($TTQ_i = TTQ_j$ and $\tau^i(\widetilde{\Gamma}^i) = \tau^j(\widetilde{\Gamma}^j)$) where, without loss of generality, $p^i > p^j$, the modified trajectories $\Gamma^{i,*}$ and $\Gamma^{j,*}$ generated by* GAMECHAT *maximize social welfare.*

D. Game Theoretic Strategy (Strategy 1)

In the event that communication is not possible, the agents are not able to come to an agreement, or the agents end up in an SMG while they are still in the process of communicating (consensus has not yet been reached), we need a control strategy for the agents to fall back on. Here, we describe our proposed strategy (hereafter called *Strategy 1*):

1) If there will be no collision on the rest of the desired trajectory $\widetilde{\Gamma}^i$, the agent sets linear velocity to $v_{\max}$.
2) Let TTQ_i denote the time agent i would take to reach $\mathcal{Q}$ (for now, assume there is asymmetry so $\text{TTQ}_i \neq \text{TTQ}_j$).
3) If $\text{TTQ}_i < \text{TTQ}_j$, then agent i simply chooses $v_{\max}$ as its linear velocity, taking the leader role.
4) If $\text{TTQ}_i > \text{TTQ}_j$, agent i selects a velocity so it reaches $\mathcal{Q}$ at the instant that agent j completely clears $\mathcal{Q}$. For our environment, this means agent i will choose a linear

TABLE I
PERFORMANCE OF THE VARIOUS CONTROL METHODS IN DOORWAY SCENARIO.

Method	(↓) # Coll.	(↓) # DLs	(↑) % CP	(↓) Hi Pri. TTG (s)	(↓) Makespan (s)	(↑) Min v (m/s)	(↓) Δ Path (m)
MPC-CBF [5]	0	18	N/A	N/A	N/A	N/A	N/A
SMG-CBF [18]	0	0	50	11.300 ± 1.289	12.400 ± 0.873	0.118 ± 0.003	0.009 ± 0.002
GAMECHAT (no LLM)	**0**	**0**	50	10.733 ± 0.566	**11.267 ± 0.194**	0.200 ± 0.088	0.010 ± 0.001
GAMECHAT Ground Truth	0	0	100	10.400 ± 0.291	11.533 ± 0.194	0.227 ± 0.025	0.009 ± 0.001
GAMECHAT	**0**	**0**	**100**	**10.467 ± 0.388**	11.600 ± 0.291	**0.228 ± 0.030**	**0.001 ± 0.001**

velocity of $\frac{d_i v_{\max}}{l+d_j}$ until reaching $\mathcal{Q}$, from which point it will choose $v_{\max}$. It takes the follower role.

Theorem IV.3. *If both agents follow Strategy 1 at all times, then we have a subgame perfect equilibrium.*

V. EXPERIMENTS AND RESULTS

There are three key questions we address. First, in the absence of communication, does Strategy 1 perform better than baseline methods? Second, when we incorporate communication, how often do the conclusions of the LLMs match up with the true priorities? Finally, how do the communicative and noncommunicative methods compare?

We tracked several metrics. First, we counted the number of collisions and deadlocks. We next examined metrics involving priority: the percent of scenarios where the higher-priority agent got to $\mathcal{Q}$ first and the average time to goal for the higher priority agent. Finally, we tracked the makespan, defined as the total duration of the scenario (also the slower agent's time to goal) and metrics involving invasiveness: the slower agent's minimum velocity prior to reaching $\mathcal{Q}$ (higher is better since it indicates less significant slowdown) and the average deviation from the desired straight-line paths.

The metrics are displayed in Table I. Note that MPC-CBF, despite being safe from collisions, frequently failed to complete the scenario since it lacks deadlock resolution capabilities. All other methods, however, were always able to avoid collisions and deadlocks, and they also had little path deviation, indicating smoothness. Also, we ran a hardcoded experiment (from here on referred to as the hardcoded baseline) where the first agent needed to reach $\mathcal{Q}$ before the other agent was permitted to begin moving. It obtains a makespan of $18.4s$ in the doorway environment.

Noncommunicative Methods: We observe that GAMECHAT (no LLM) is more agile than SMG-CBF as it causes the second agent to slow down less (see Min v in Table I). GAMECHAT (no LLM) has the lowest makespan of all the methods, at the expense of not considering priority at all. It struggles to identify a leader and follower until some time has passed (see Figure 3), which increases makespan and reduces its min v as one agent has to slow down greatly when it suddenly realizes it is the follower only after it gets very close to the doorway. However, overall, our game-theoretic control strategy outperforms existing methods.

LLM Performance and Welfare Maximization: We measured the average duration of the LLM conversation as $2.767s$ with a standard deviation of $0.441s$. At first, the LLMs had difficulty with the task of conversing and determining priority (e.g., task forgetting, hallucinating false instructions, forgetting the consensus, etc.). But after carefully modifying the prompts, these problems became less frequent, to the point that GAMECHAT performed as well as when agents were already assigned the correct roles (GAMECHAT Ground Truth). The correct consensus was always reached since by default, LLMs are designed to be very honest and cooperative. Since GAMECHAT was reaches the correct ordering of priorities in the consensuses, it maximizes social welfare (see Theorem IV.2). Even in the asymmetric case where the higher priority agent starts farther away from $\mathcal{Q}$, GAMECHAT can have it go first, something the noncommunicative methods cannot do (see Figure 3).

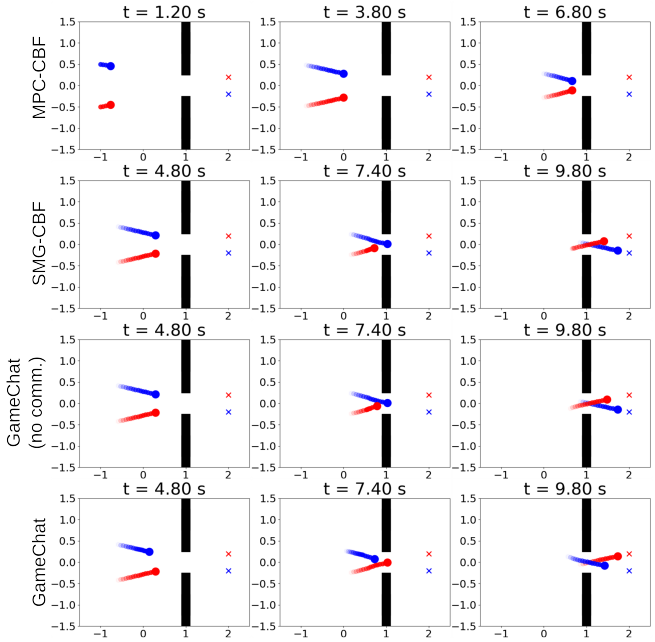

Fig. 3. Trajectories generated by noncommunicative methods in the symmetric doorway. **Blue** is a **grocery** agent and **red** is a **hospital** agent.

Communicative vs. Noncommunicative Methods: GAMECHAT (no LLM) has a lower makespan than the communicative methods. However, since it does not account for priorities at all, it (along with the other noncommunicative methods) only has a 50% correct priority rate (the same as guessing), while the communicative methods give up a small amount of time in exchange for a doubling in accurately ordering the priorities, which is a worthwhile exchange as it greatly increases social welfare.

VI. CONCLUSION AND FUTURE WORK

We presented GAMECHAT, a new approach for safe, agile, and socially optimal control in multi-robot constrained environments with self-interested agents. Agents communicate with each other by querying their LLMs to generate messages. Agents fall back on a game-theoretic strategy if they do not reach consensus. We demonstrate our approach's effectiveness in simulated constrained environments.

An interesting future direction is to address the challenge of agents lying in communication. One could also work on fine-tuning LLMs for better performance on the task (which we did not focus on since their performance out-of-the-box exceeded expectations) or running local models for lower latency. Adding more types of agents and more nuanced/complex backstories would be interesting as it could help us find the failure point of using LLMs to determine relative priority. Finally, we hope to bring GAMECHAT onto physical robots to validate real-world performance.

REFERENCES

[1] C. Witteveen and M. de Weerdt, "Multi-agent planning for non-cooperative agents.," in *AAAI Spring Symposium: Distributed Plan and Schedule Management*, p. 169, 2006.

[2] R. Selten, "Spieltheoretische behandlung eines oligopolmodells mit nachfrageträgheit: Teil i: Bestimmung des dynamischen preisgleichgewichts," *Zeitschrift für die gesamte Staatswissenschaft/Journal of Institutional and Theoretical Economics*, no. H. 2, pp. 301–324, 1965.

[3] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*, pp. 3420–3431, Ieee, 2019.

[4] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[5] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *2021 American Control Conference (ACC)*, pp. 3882–3889, IEEE, 2021.

[6] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed autonomous robotic systems: The 10th international symposium*, pp. 203–216, Springer, 2013.

[7] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, online collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[8] Y. Chen, M. Guo, and Z. Li, "Recursive feasibility and deadlock resolution in mpc-based multi-robot trajectory generation," *arXiv preprint arXiv:2202.06071*, 2022.

[9] N. Suriyarachchi, R. Chandra, J. S. Baras, and D. Manocha, "Gameopt: Optimal real-time multi-agent planning and control for dynamic intersections," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2599–2606, IEEE, 2022.

[10] D. Carlino, S. D. Boyles, and P. Stone, "Auction-based autonomous intersection management," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pp. 529–534, IEEE, 2013.

[11] R. Chandra, R. Maligi, A. Anantula, and J. Biswas, "Socialmapf: Optimal and efficient multi-agent path finding with strategic agents for social navigation," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3214–3221, 2023.

[12] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of artificial intelligence research*, vol. 31, pp. 591–656, 2008.

[13] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 6252–6259, IEEE, 2018.

[14] R. Chandra, H. Karnan, N. Mehr, P. Stone, and J. Biswas, "Towards imitation learning in real world unstructured social mini-games in pedestrian crowds," *arXiv preprint arXiv:2405.16439*, 2024.

[15] S. Gouru, S. Lakkoju, and R. Chandra, "Livenet: Robust, minimally invasive multi-robot control for safe and live navigation in constrained environments," *arXiv preprint arXiv:2412.04659*, 2024.

[16] S. Le Cleac'h, M. Schwager, and Z. Manchester, "Algames: a fast augmented lagrangian solver for constrained dynamic games," *Autonomous Robots*, vol. 46, no. 1, pp. 201–215, 2022.

[17] B. Di and A. Lamperski, "Newton's method and differential dynamic programming for unconstrained nonlinear dynamic games," in *2019 IEEE 58th conference on decision and control (CDC)*, pp. 4073–4078, IEEE, 2019.

[18] R. Chandra, V. Zinage, E. Bakolas, P. Stone, and J. Biswas, "Deadlock-free, safe, and decentralized multi-robot navigation in social mini-games via discrete-time control barrier functions," *arXiv preprint arXiv:2308.10966*, 2023.

[19] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 11785–11792, IEEE, 2020.

[20] S. H. Arul, A. S. Bedi, and D. Manocha, "Dmca: Dense multi-agent navigation using attention and communication," *arXiv preprint arXiv:2209.06415*, 2022.

[21] S. H. Arul, A. S. Bedi, and D. Manocha, "When, what, and with whom to communicate: Enhancing rl-based multi-robot navigation through selective communication," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7695–7695, IEEE, 2024.

[22] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530, IEEE, 2023.

[23] K. Garg, S. Zhang, J. Arkin, and C. Fan, "Foundation models to the rescue: Deadlock resolution in connected multi-robot systems," *arXiv preprint arXiv:2404.06413*, 2024.

[24] D. Song, J. Liang, A. Payandeh, X. Xiao, and D. Manocha, "Socially aware robot navigation through scoring using vision-language models," *arXiv e-prints*, pp. arXiv–2404, 2024.

[25] Z. Mandi, S. Jain, and S. Song, "Roco: Dialectic multi-robot collaboration with large language models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 286–299, IEEE, 2024.

[26] E. A. Hansen, D. S. Bernstein, and S. Zilberstein, "Dynamic programming for partially observable stochastic games," in *AAAI*, vol. 4, pp. 709–715, 2004.

[27] J. C. Harsanyi, "Cardinal welfare, individualistic ethics, and interpersonal comparisons of utility," *Journal of political economy*, vol. 63, no. 4, pp. 309–321, 1955.

[28] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, *et al.*, "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.