SocialMapf: Optimal and Efficient Multi-Agent Path Finding With Strategic Agents for Social Navigation

Rohan Chandra, Rahul Maligi, Arya Anantula, and Joydeep Biswas, *Member, IEEE*

***Abstract*—We propose an extension to the MAPF formulation, called SocialMapf, to account for private incentives of agents in constrained environments such as doorways, narrow hallways, and corridor intersections. SocialMapf is able to, for instance, accurately reason about the urgent incentive of an agent rushing to the hospital over another agent's less urgent incentive of going to a grocery store; MAPF ignores such agent-specific incentives. Our proposed formulation addresses the open problem of optimal and efficient path planning for agents with private incentives. To solve SocialMapf, we propose a new class of algorithms that use mechanism design during conflict resolution to simultaneously optimize agents' private local utilities and the global system objective. We perform an extensive array of experiments that show that optimal search-based MAPF techniques lead to collisions and increased time-to-goal in SocialMapf compared to our proposed method using mechanism design. Furthermore, we empirically demonstrate that mechanism design results in models that maximizes agent utility and minimizes the overall time-to-goal of the entire system. We further showcase the capabilities of mechanism design-based planning by successfully deploying it in environments with static obstacles. To conclude, we briefly list several research directions using the SocialMapf formulation, such as exploring motion planning in the continuous domain for agents with private incentives.**

***Index Terms*—Multi-agent systems, path planning, mobile robots.**

I. Introduction

The multi-agent path finding (MAPF) problem corresponds to efficiently finding optimal and collision-free paths (defined as a set of waypoints or coordinates) for $k > 1$ agents in discrete $2-$D space-time. Given a bi-connected graph $\mathcal{G}$, an initial configuration S^0, containing the starting positions of all agents and a final configuration G, representing the goals of all

Manuscript received 15 October 2022; accepted 24 March 2023. Date of publication 6 April 2023; date of current version 18 April 2023. This letter was recommended for publication by Associate Editor P. Grippa and Editor M. A. Hsieh upon evaluation of the reviewers' comments. This work was supported in part by the NSF under Grants CAREER2046955, IIS-1954778, SHF-2006404, and OIA-2219236, in part by ARO under Grant W911NF-19- 2- 0333, and in part by Amazon, which has taken place in the Autonomous Mobile Robotics Laboratory (AMRL) at UT Austin. The views and conclusions contained in this document are those of the authors only. *(Corresponding author: Rohan Chandra.)*

The authors are with the Department of Computer Science, University of Texas, Austin, TX 78732 USA (e-mail: rchandra@utexas.edu; maligir@utexas.edu; anantula.arya@utexas.edu; joydeepb@cs.utexas.edu).

Digital Object Identifier 10.1109/LRA.2023.3265169

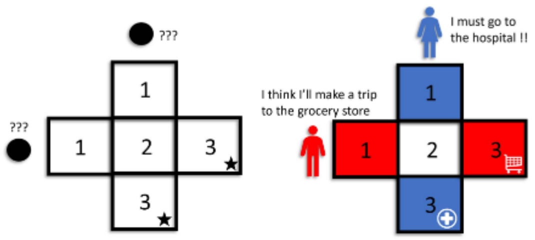
???
???
1
1
2
3
3
I must go to the hospital !!
I think I'll make a trip to the grocery store
1
1
2
3
3

Fig. 1. (left) Classical MAPF with cooperative agents and (right) MAPF in social navigation scenarios with strategic agents. Strategic agents have private incentives and strive to optimize individual utility functions, rather than a common global objective function.

the agents, the MAPF objective is to find the sequence of configurations, $\Gamma = \{S^0, S^1, \ldots, S^{t-1}, G\}$, where Γ optimizes a global objective function such as the sum-of-costs or makespan.

MAPF is a fundamental problem in robotics and control and appears frequently in many real world applications such as vehicle routing [1], [2], warehouse robotics [3], airport towing [4], and so on. Since its introduction in [5] and follow-up improvements [6], [7], [8], however, MAPF has always modeled cooperative agents that strive to optimize the global objective function. Consequently, planning under the classical MAPF formulation is typically centralized where agents simply execute the solution to the common global objective function.

In real world social navigation scenarios, however, agents are not cooperative; they are strategic, or self-interested, possessing privately-held incentives that dictate their actions [2], [9]. These agents no longer share a common global objective function and instead strive to optimize utility functions unique to each agent. Consider Fig. 1 depicting two different MAPF setups at a particular time t. On the left, we show the classical MAPF setting with two cooperative agents represented as black discs with their respective goal locations marked by ★'s. On the right, two strategic agents, Bob (red) and Alice (blue), have distinct objectives. Alice has a higher priority of going to the hospital while Bob is making a routine trip to the grocery store. In both cases, agents can fully observe the state space and step up to 3 tiles. In the MAPF setting, the cooperative agents can additionally choose to reveal their step size, or *incentive*. On the other hand, Alice's step value is not known to Bob and vice-versa, on account of them being strategic and self-interested. The scenario on the right is what we will call a SocialMapf scenario (defined formally in Section IV). Centralized MAPF solvers will detect a collision at (2, 2) and select actions for each agent that prevent the collision.

Solving a SOCIALMAPF problem implies finding a sequence $\Gamma_{\text{SOCIAL}} = \{S^0, S^1, \ldots, S^{t-1}, G\}$ that optimizes both the global objective function as well as the utility functions corresponding to each strategic agent. In this work, we cast this problem through the lens of mechanism design where the designer of the mechanism proposes a set of rules that simultaneously serves to optimize both sets of objectives.

Main Contributions

1) Our first contribution is a new framework called SOCIALMAPF that extends MAPF to include strategic, or non-cooperative, agents.
2) Our second contribution is the first polynomial-time auction-based MAPF solver to optimally and efficiently solve SOCIALMAPF, where optimality is defined in Definition III.2.

SOCIALMAPF addresses several longstanding issues in MAPF such as:

- *Decentralized planning among strategic agents:* In the most general sense, agents' incentives, which determine the next state or action, are hidden from other agents. Decentralized planning is often a desirable class of solutions due to their low computational requirements but require access to the agent incentives. Auction theory provides a solution for decentralized planning among agents with private incentives.
- *Optimality and efficiency guarantees:* Approaches for SOCIALMAPF are either optimal or efficient, but not both. One of the primary research areas in SOCIALMAPF is to develop efficient solvers within bounded sub-optimality. We propose the first optimal *and* efficient SOCIALMAPF solver for strategic agents.

In the remainder of this letter, we present the SOCIALMAPF framework in Section III and propose our algorithm for solving SOCIALMAPF in Section IV. We present our experiments in and analyze preliminary results in Section V. Finally, we conclude the letter and discuss limitations and future directions of research in Section VI.

II. RELATED WORK

A. Social Navigation

A recent survey on social navigation by Mavrogiannis et al. [10] highlights some of the initial efforts in the field, starting with the RHINO [11] and MINERVA [12] projects followed by [13], as well as current progress in autonomous driving, indoor navigation, and last-mile delivery. This includes siomulators like SocNavBench [14] and SEAN 2.0. [15]. Researchers also focus on analyzing social cues and human walking behavior, such as gaits [16], body language [17], and gaze [18], to facilitate social robot navigation. For instance, gaze information can help robots predict human intention and avoid collision. Body language can provide information about the presence of other people and their behavior, enabling robots to make informed decisions. Furthermore, there are also methods that attempt to model social norms such as right of way rules [19].

In motion planning, researchers address the problem of how to navigate in human crowds, while also considering their preferences and comfort levels [20]. This involves taking into account factors such as human density, speed, and direction. Additionally, the robot's motion must be natural and compliant with social norms, such as maintaining a suitable distance from other people [10]. Finally, evaluating the performance of social navigation algorithms is an active research area employing a variety of metrics including safety, efficiency, and human comfort [21], [22]. Real-world experiments are conducted to assess the algorithms in various environments and situations, and to evaluate the impact of different parameters, such as crowd size and density. Many simulators such as SocNavBench [14] and SEAN 2.0. [15] have also been proposed.

B. Game-Theoretic Motion Planning

Game-theoretic motion planning is a rapidly growing area of research in robotics, with applications in a variety of multi-robot systems. One of the main challenges in game-theoretic motion planning is to model the interactions between robots in a way that accurately captures the real-world constraints and uncertainties. Various game-theoretic models have been proposed, such as cooperative games, non-cooperative, hierarchical games, and general-sum differential games.

The literature on general-sum differential games classify existing algorithms for solving game-theoretic equilibria in robotics into four categories: 1) algorithms based on decomposition [23], [24], such as Jacobi or Gauss-Siedel methods, that are easy to interpret and scale well with the number of players, but have slow convergence and may require many iterations to find a Nash equilibrium; 2) algorithms based on dynamic programming [25], such as Markovian Stackelberg strategy, that capture the game-theoretic nature of problems but suffer from the curse of dimensionality and are limited to two players; 3) algorithms based on differential dynamic programming [26], [27], [28], [29], [30], [31], such as robust control, that scale polynomially with the number of players and run in real-time, but do not handle constraints well; and 4) algorithms based on direct methods in trajectory optimization [32], [33], [34], such as Newton's method, that are capable of handling general state and control input constraints, and demonstrate fast convergence.

C. Multi-Agent Path Finding

There is a rich history behind MAPF. Since it is impractical to discuss it all here, we refer the interested reader to a recent survey [35]. Below, we highlight the three broad categories of solutions to discrete MAPF. The first class of discrete MAPF algorithms consist of fast (worst-case time complexity is polynomial in the size of the graph) and complete solvers. A general template of this category of methods can be represented by the Kornhauser's algorithm [36], with a $\mathcal{O}(|V|^3)$ time complexity including [37], [38], [39], [40]. The second category of methods trade efficiency for optimality guarantees. While the algorithms belonging to the previous category are fast and, under certain conditions, complete, they do not provide any guarantee regarding the quality of the solution they return [6], [7], [41]. A third category of methods consist of bounded sub-optimal algorithms that return a solution whose cost is at most $1 + \epsilon$ times the cost of an optimal solution [42], [43].

Finally, we note that there exists previous approaches for MAPF with strategic agents using auctions. Amir et al. [44] use an iterative combinatorial auction called iBundle. While this is strategy-proof, the strategy space consists of multiple paths (called "bundles") and solving the auction is NP-hard. Gautier et al. [45] address the prohibitive computational costs of iBundle

SOC: sum-of-costs

TABLE I
COMPARING THE PROBLEM DESCRIPTIONS OF MAPF AND SOCIALMAPF

Parameter	MAPF	SOCIALMAPF
State space	pose, orientation	pose, orientation, incentives
Action space	{up, down, left, right, wait}	{up, down, left, right, wait}
Reward	–	$\alpha_i(u_i^t) = (t_{g,i})^{-1}$
Global Optimality	SoC, Makespan	Social Welfare
Local Optimality	–	$\Phi_i(u_i, u_{-i})$

t represents the current time-step and v_i denotes the private Incentive of α_i.

using a heuristic mechanism that trades strategy-proofness for better worst-case runtime performance. Our solution differs from this line of research by (i) bidding on individual states as opposed to bundles and (ii) providing, simultaneously, optimality as well as polynomial runtime guarantees.

III. SOCIALMAPF: FRAMEWORK

The two characteristics that underlie social navigation, missing from classical MAPF formulation are that (i) agents have different incentives and (ii) these incentives are private. Since the notion of incentives is typically ill-defined, it is best to illustrate it with an example. Consider simultaneously an ambulance carrying a critical patient to the hospital and a family making a regular trip to the nearby grocery store. The ambulance is said to possess a higher "incentive" or priority (dictated by the urgency of the situation) to reach their goals than the family en route to the grocery store. In order to extend MAPF to social navigation, we first need to model these incentives and then figure out how to perform MAPF when the incentives of the other agents are hidden. Table I compares the MAPF and SOCIALMAPF formulations.

α_i : agent Γ : trajectory
$v_i = \varphi_i(\Gamma)$ $\varphi_i : \Gamma \rightarrow \mathbb{R}^{\geq 0}$: private risk function

A. Problem Formulation

Given an arbitrary graph,[1] $\mathcal{G}$, at any particular time t, denote by S_t the current configuration of k *strategic* agents. A strategic agent is defined as follows,

Definition III.1. Strategic Agent: A strategic agent a_i is a MAPF agent with a private risk function $\varphi_i : \Gamma \longrightarrow \mathbb{R}^{\geq 0}$, where $v_i = \varphi_i(\Gamma)$ encodes the i^{th} agent's risk tolerance towards Γ.

where $\Gamma \subset |S|^T$ is the space of all finite-horizon discrete spatial configurations over the space S, and v_i is the private incentive of a_i. Trajectories for which values of v_i are closer to 0 would indicate lower risk and therefore more desirable states. The incentive parameter v_i i computed via $\varphi_i(\Gamma_i)$. In this risk function, $\varphi(\Gamma_i) = \sum_m w_m \phi_m$ is a weighted linear sum of m different features such as distance from obstacles, smoothness, distance from goal, preference to move first during conflicts, etc., where $w_m \in \mathbb{R}$ is a non-negative weight value and $\phi_m \in \mathbb{R}$ is a feature value that encodes the *quality* of the trajectory Γ_i. The weight values differ for agents with different risk tolerances. For instance, a risk-averse agent would place higher weight on the distance from obstacle than a risk-seeking agent. These incentives are computed apriori. Alternatively, one can use inverse reinforcement learning or inverse optimal control to compute these incentives via maximum likelihood estimation or distribution matching via expert trajectory techniques.

The SOCIALMAPF goal is to output an optimal sequence of configurations, $\Gamma_{\text{OPT}} = \{S^0, S^1, \ldots, S^{t-1}, G\}$, where S^0 and G are the initial and final configurations, such that the Γ satisfies a precise set of conditions that are explained later. A configuration S_t at any time t is specified by a list of discrete state space vectors, $s_i^t = [x_i^t, y_i^t, v_i]^\top \in \mathbb{Z}_+ \times \mathbb{Z}_+ \times \mathbb{Z}_+$ for $1 \leq i \leq k$ where x_i, y_i corresponds to a vertex occupied by a_i in $\mathcal{G}$ and v_i is the private incentive of a_i encoding the priority toward its goal g_i.

The discrete action space, identical for each agent, is given by {`up, down, left, right, wait`}. At a time-step t and from current vertex $[x_i, y_i]^\top$, an agent a_i can select from one of these actions which we denote by u_i^t; for each of the first 4 actions, each agent in SOCIALMAPF can additionally choose to step up to v_i vertices at a time. We denote the step value by $0 \leq \tau \leq v_i$. The state transition function for an agent can be specified by $s_i^{t+1} = [x_i^t \pm \tau, y_i^t, v_i]^\top$ if $u_i^t \in \{$`right, left`$\}$ or $s_i^{t+1} = [x_i^t, y_i^t \pm \tau, v_i]^\top$ if $u_i^t \in \{$`down, up`$\}$ or $s_i^{t+1} = [x_i^t, y_i^t, v_i]^\top$ if $u_i^t \in \{$`wait`$\}$.

Next, we model the global objective and agent-specific utility functions in SOCIALMAPF. A standard global objective function for the classical MAPF problem is the sum-of-costs (SoC) given by $\mathcal{F}(\Gamma) = \sum_{i=1}^{k} t_{g,i}$, where $t_{g,i}$ is the time taken by a_i to reach its goal g_i. The classical MAPF goal is to find the optimal Γ that minimizes $\mathcal{F}$. In our SOCIALMAPF setting, minimizing the SoC *alone* does not yield an optimal solution as we need to simultaneously maximize the agents' utility. Each agent a_i in SOCIALMAPF receives a reward, $\alpha_i(u_i) = (t_{g,i})^{-1}$, on reaching the goal in time $t_{g,i}$ by executing action u_i. Furthermore, since agents are strategic and cannot observe the incentives of other agents, we introduce the notion of a penalty function, $p_i(\alpha_{-i}) \in \mathbb{R}$, incurred by each agent by executing u_{-i}, where $-i$ denotes every agent except the i^{th} agent. The utility function of a_i can be written as,

$$\Phi_i(u_i, u_{-i}) = \alpha_i(u_i) - p_i(\alpha_{-i}) \tag{1}$$

At this point, we can state the definition for Γ_{OPT}:

Definition III.2: Γ_{OPT}: A sequence of transitions, $\Gamma = \{S^0, S^1, \ldots, S^{t-1}, G\}$ is optimal when the global sum-of-cost is minimal and local agent utilities (1) are maximal.

Finally, we define the model for conflict resolution in SOCIALMAPF. A conflict is defined by the tuple $\langle \mathcal{C}_{\mathcal{O}}^t, \mathcal{O}, t \rangle$, which denotes a conflict between agents belonging to the set $\mathcal{C}_{\mathcal{O}}^t$ at time t over the vertex $\mathcal{O}$ in $\mathcal{G}$. Naturally, agents must either find an alternate non-conflicting path or must move through $\mathcal{O}$ in a turn-based ordering. A turn-based ordering is defined as follows.

Definition III.3. Turn-based Orderings (σ): A turn-based ordering is a permutation $\sigma : \mathcal{C}_{\mathcal{O}}^t \rightarrow [1, k]$ over the set $\mathcal{C}_{\mathcal{O}}^t$. For any $i, j \in [1, k]$, $\sigma(a_i) = j$, equivalently $\sigma_i = j$, indicates that a_i will move on the j^{th} turn.

For a given conflict $\langle \mathcal{C}_{\mathcal{O}}^t, \mathcal{O}, t \rangle$, clearly, different permutations results in different configurations S^{t+1} in Γ. Therefore, we have $\sigma \vdash \Gamma$, where $X \vdash Y$ denotes Y can be obtained from the set of statements in X.

The conventional wisdom dictates choosing σ randomly if agents arrive at the conflict at the same time or execute first-in first-out if they arrive asynchronously [46]. And while $\sigma \vdash \Gamma_{\text{OPT}}$ for any σ in classical MAPF, randomly scheduling agents to move through $\mathcal{O}$ is sub-optimal, as we show in Section V. The reader may further recall example 1–opting for random ordering allows the ambulance to be delayed. In SOCIALMAPF, we seek a unique optimal ordering, $\sigma_{\text{OPT}} \vdash \Gamma_{\text{OPT}}$, where Γ_{OPT} simultaneously minimizes the SoC as well as maximizes (1) for all i.

[1]In this work, we assume a 4-connected $M \times N$ bi-connected graph.

IV. PROPOSED ALGORITHM TO SOLVE SOCIALMAPF

We present a solver that addresses the two issues presented in the previous section. For the first issue (*"agents have incentives"*), SOCIALMAPF models agent incentives through an auction program. Specifically, agents autonomously submit bids to the auction program based on their incentives. Depending on the bids made, the agents receive rewards and make payments. The tradeoff between the reward and payment is controlled via the incentive parameter, which prevents agents from bidding infinitely high values. For the second issue (*"incentives are private"*), agents solve (2) independent of the actions of other agents, that is, there exist a dominant strategy for each agent regardless of other agents' actions. Therefore, agents do not require knowledge of other agents' incentives.

m: # of items v_i: private valuation
k: # of agents g_i: allocation rule h_i: payment rule

A. Background on Auction Theory

Auctions are a game-theoretic mechanism that are used extensively in economic applications like online advertising [47]. In an auction, there are m items to be allocated among k agents. Each agent a_i has a private valuation v_i and submits a bid b_i to receive at most one item dictated by an allocation rule g_i. The valuation v_i is the "dollar amount" that a_i would pay for that item assuming there was no competitor. g_i is a rule that decides a winner for the item for *e.g.*, agent with the highest bid. A strategy is defined as an n dimensional vector, $b = (b_i \cup b_{-i})$, representing the bids made by every agent. b_{-i} denotes the bids made by all agents except a_i. The quasi-linear utility u_i incurred by a_i is given as follows,

$$\Phi_i(b) = v_i g_i(b) - \overset{\text{payment term}}{h_i} \tag{2}$$

In the equation above, the quantity on the left represents the total utility for a_i which is equal to gain value of the allocated goods $v_i g_i(b)$ minus a payment term h_i. We refer the reader to Chapter 3 in [47] for a derivation and detailed analysis of (2). The performance of an auction is measured by the social welfare of the entire system comprising of the k agents, which is defined as

social welfare

$$\mathcal{W}(b) = \sum_{i=1}^{k} v_i g_i(b) \tag{3}$$

The primary objective in auction theory is to determine an allocation rule g and a payment rule h such that there exist b^* for which both $u_i(b)$ for all i and $\mathcal{W}(b)$ are maximized. Unfortunately, simply establishing existence is insufficient; we should be able to compute or determine b^*. A strategy-proof or incentive-compatible auction yields $b_i^* = v_i$.

B. Algorithm

We leverage the global optimality in terms of SoC of search-based MAPF solutions such as CBS and, in this section, extend their core functionality to include local agent-level optimality (1). Formally, we identify two phases of a SOCIALMAPF solver. The top level phase is the motion planning phase where agents make progress by stepping towards their goal states along a pre-computed trajectory cost map. The bottom level phase is the conflict resolution phase which resolves conflicts over shared goal states. Since search-based methods are susceptible to uncertainty in the edge costs [48] and, as we show later in Section V-D, result in collisions and increased time-to-goal, we rely on variants of artificial potential field (APF) methods to step towards the goal. The conflict resolution phase employs

M_c: map c: goal state

an auction to determine an optimal priority ordering in which agents should pass through the conflicted states. We describe these two stages below:

1) Phase 1: The Motion Planning Phase: We begin by creating a map $\mathcal{M}_c$ corresponding to each goal state c. Each tile (indicated by i, j denoting the ith row and jth column) in $\mathcal{M}_c$ is assigned a potential value using the A^* algorithm that encodes the number of tiles between the $\mathcal{G}[i,j]$ and the goal c. Although this formulation is designed for any number of goals, we assume $|c| = 1$ for simplicity.

2) Phase 2: The Conflict Resolution Phase: During a conflict $\langle \mathcal{C}_\mathcal{O}^t, \mathcal{O}, t\rangle$, suppose $\mathcal{C}_\mathcal{O}^t = \{a_1, a_2\}$. Then, by design, either $u_1^t \in \{\texttt{up, down, left, right}\}, u_2^t = \texttt{wait}$ or $u_2^t \in \{\texttt{up, down, left, right}\}, u_1^t = \texttt{wait}$. A particular alternative yields a specific turn-based ordering σ. In the above example, $\sigma = \sigma_1\sigma_2$ where $\sigma_1 = 1, \sigma_2 = 2$ or $\sigma_1 = 2, \sigma_2 = 1$. Crucially, with cooperative agents as in classical MAPF, every value of σ results in the the same outcome, but this is not so in SOCIALMAPF, where we turn to mechanism design in order to determine σ_{OPT}.

More specifically, we run an auction, (g, h), with a given allocation rule g and payment rule h. Since we are operating in simulation, we assume agents have some form of "digital currency". Agents a_i bid on the values of σ_i with the following allocation rule g:

1) Given set $\mathcal{C}_\mathcal{O}^t$ of agents conflicted over state $\mathcal{O}$ at time t and their corresponding bids b_i, initialize $q \leftarrow 0$.
2) Sort $\mathcal{C}_\mathcal{O}^t$ in decreasing order of b_i.
3) Do the following for $|\mathcal{C}_\mathcal{O}^t|$ steps:
   a) Increment q by 1.
   b) Let a_i be the the first element in $\mathcal{C}_\mathcal{O}^t$.
   c) Set $\sigma_i = q$.
   d) $\mathcal{C}_\mathcal{O}^t \leftarrow \mathcal{C}_\mathcal{O}^t \setminus \{a_i\}$.
4) Repeat steps $1-3$ while $\mathcal{C}_\mathcal{O}^t$ is non-empty.
5) Return

$$\sigma_{\text{SocialMapf}} = \sigma_1\sigma_2\ldots\sigma_{|\mathcal{C}_\mathcal{O}^t|} \tag{4}$$

allocation rule

To summarize the algorithm, the agent with the highest bid is allocated the highest priority and is allowed to move first, followed by the second-highest bid, and so on. Upon moving on the qth turn, the agent receives a reward of α_q and makes a payment h_i according to the following payment rule,

$$h_i(b) = \sum_{j=q}^{|\mathcal{C}_\mathcal{O}^t|} b_{j+1}\left(\alpha_j - \alpha_{j+1}\right) \tag{5}$$

The payment rule is the "social cost" of reaching the goal ahead of the agents arriving on turns $q+1, q+2, \ldots, q+\mathcal{C}_\mathcal{O}^t$. The overall utility function for a_i, $\Phi_i(b)$, is given by the following equation, utility function

$$\Phi_i(b) = v_i\alpha_q - \sum_{j=q}^{|\mathcal{C}_\mathcal{O}^t|} \overset{\text{agents' bids}}{b_{j+1}}\left(\alpha_j - \alpha_{j+1}\right) \tag{6}$$

where $b = [b_1, b_2, \ldots, b_{\mathcal{C}_\mathcal{O}^t}]^\top$ represent the agent bids. The gain term $v_i\alpha_q$ denotes a_i's time reward on reaching the goal on the qth turn. Note that the gain term in (6) $(v_i\alpha_q)$ implicitly encourages agents to adopt shorter paths (greater time rewards) but not necessarily the shortest path. Such a formulation recognizes that social navigation is not equivalent to traversing the shortest paths, as typically formulated in the classical MAPF. To obtain the global objective function for a sequence Γ, we first reformulate the SoC as

$$\widehat{\mathcal{F}}(\Gamma) = \sum_{i=1}^{k} v_i t_{g,i} \tag{7}$$

Minimizing (7) is the same as minimizing the SoC since the vector $[v_1, v_2, \ldots, v_k]^\top$ is constant. Equivalently, using the A.M.-G.M.-H.M[2] inequality, we can choose to maximize $\sum_{i=1}^{k} v_i (t_{g,i})^{-1}$. Using $\alpha_i = (t_{g,i})^{-1}$, we have,

$$\Gamma^* = \arg_\Gamma \min \widehat{\mathcal{F}} = \arg_\Gamma \max \sum_{i=1}^{k} v_i \alpha_i \tag{8}$$

which represents the social welfare, $\mathcal{W}$, of an auction with k agents that receives rewards α_i. We have, at this point, derived the existence of $\sigma_{\text{SOCIALMAPF}} \vdash \Gamma$ with (6) and (8) corresponding to the agent utility function and social welfare of a sequence Γ. We now prove Γ is optimal.

Theorem IV.1. $\sigma_{\text{SOCIALMAPF}} \vdash \Gamma_{\text{OPT}}$: If $\sigma_{\text{SOCIALMAPF}} \leftarrow (g, h)$ where (g, h) are defined by (4) and (5), then $\sigma_{\text{SOCIALMAPF}} \vdash \Gamma_{\text{OPT}}$.

Proof: From Definition 3.2, Γ is optimal when the global sum-of-cost is minimal and local agent utilities (1) are maximal. Equations (6) and (8) demonstrate that the SoC and (1) correspond to the utility and welfare functions of an arbitrary auction. Therefore, to prove that $\sigma_{\text{SOCIALMAPF}} \vdash \Gamma_{\text{OPT}}$, it is sufficient to show that (g, h) is welfare-maximizing and strategy-proof. Theoretical analysis of the auction (g, h) in [49], [50] showed that the auction specified by (g, h) in (6) is a strategy-proof, welfare-maximizing auction with the following strictly dominant strategy: $b_i^* = v_i$. □

For an agent at any time step t and state s, the solver steps towards a state that has a lower cost following the A* path, which is optimal. We assume agents always move at every time step (unless they are forced to wait during the conflict resolution phase). If there is a conflict at some next state, then the motion planner first checks if either agent can be assigned a different next state with an identical cost. If so, that agent is reassigned and the other agent is allocated the original next state. If, however, neither agent can be reassigned, then we enter the conflict resolution phase. The conflict resolution is performed via the auction and always outputs a priority order. This is also how our approach prevents deadlocks.

V. Experiments and Discussion

Our experiments answer the following questions: (i) how does mechanism design-based planning perform in SOCIALMAPF? (ii) is mechanism design-based planning strategy-proof? (iii), can this new class of mechanism design-based solvers extend to increasingly complex SOCIALMAPF environments, for *e.g.* in the presence of obstacles? and (iv) can current optimal search-based MAPF algorithms solve SOCIALMAPF?

We used Python to perform all experiments on an Intel(R) i7 CPU at 2.20 GHz with 8 cores and 16 GB RAM. For experiments that involve comparing runtime, we set a maximum timeout for each trial as 20 seconds.

A. Mechanism Design-Based Planning in SocialMAPF

We test our new auction-based SOCIALMAPF solver in the same doorway, hallway, and intersection environments and compare the time taken to reach the goal as well as the number of collisions. Fig. 3(a)–(c) clearly show that our proposed approach

[2]Arithmetic Mean-Geometric Mean-Harmonic mean.

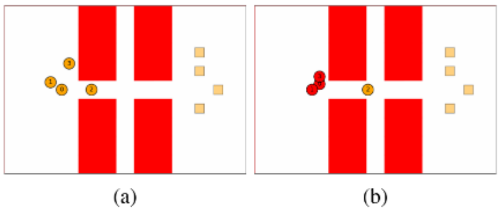

Fig. 2. Multi-agent path finding with strategic agents in a narrow hallway scenario. The yellow circles and squares denote the agents and their respective goal locations. The red regions are obstacles. (a) Initial time step. (b) Agents 0, 1, 3 collide.

TABLE II
COMPARING OUR APPROACH TO LAZYCBS AND PRIORITY-BASED SEARCH (PBS)

Method	10a	20a	30a	40a	50a
LazyCBS [41]	0.2 ms	0.4 ms	0.9 ms	1.0 ms	20 ms
PBS [51]	1.0 ms	3.0 ms	4.0 ms	10 ms	fail
Our solver	0.04 ms	0.9 ms	1.0 ms	7.0 ms	30 ms

yields 0 collisions and can scale up to $\mathcal{O}(n^2)$ agents where n refers to the size of the grid. We visualize our approach in Fig. 5(a). We vary the number of agents from 4 to 50 and average the results over 100 trials. The solid color lines represent the mean value with the shaded regions representing the confidence intervals. We note that while the runtime is still excessive compared to modern MAPF approaches [42], we emphasize that we have not implemented any performance-enhancing heuristic optimization, which is a direction of future work. The goal was to highlight the fact that in social navigation scenarios, where existing MAPF methods take over 60 seconds to find a solution for 5 or more agents, auction-based planners take up $\frac{1}{4}\times$ the time for $10\times$ the number of agents.

B. Mechanism Design-Based Planning is Strategy-Proof

One of the main challenges with MAPF with strategic agents is to simultaneously minimize the cost to the overall system-wide objective (8) and maximize each agent's utility functions (6). In this section, we empirically confirm that mechanism design (more specifically, auctions) satisfies both objectives.

In Fig. 4(a)–(c), we compare the system welfare for auction-based planning versus random ordering for the doorway, narrow hallway, and corridor intersection scenarios. As before, we vary the number of agents from 4 to 50 and average the results over 100 trials. The solid color lines represent the mean value with the shaded regions representing the confidence intervals. The blue line corresponds to the system welfare when the bidding strategy is optimally according to Theorem 4.1. The red line, on the other hand, corresponds to the system welfare when the bidding strategy is chosen randomly. The linear relationship with increasing number of agents is expected since the social welfare is a monotonically increasing linear function.

In addition, in Fig. 4(d), we plot the utility curves for 4 agents performing SOCIALMAPF in the narrow hallway scenario as depicted in Fig. 2. On the vertical axis, we plot the utility values corresponding to bids (horizontal axis) made by each agent. The black circles indicate the agents' true incentives. We observe that bidding the true incentive yields the maximum utility (highest points on each corresponding curve). Note that in simulated settings, the private incentives can be chosen arbitrarily. In real

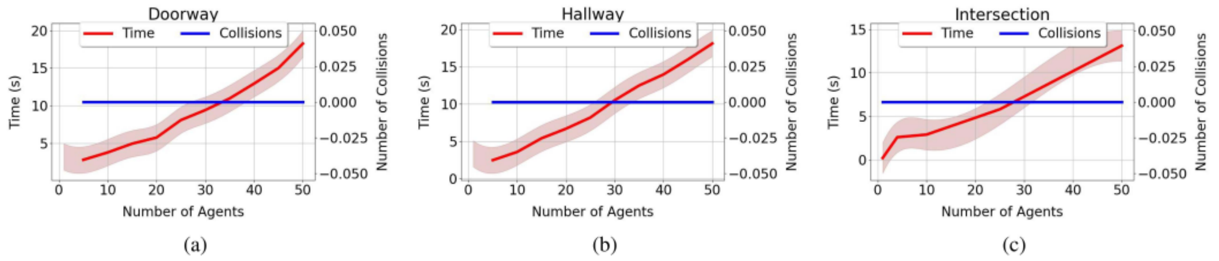

Fig. 3. Auction-based planning in SOCIALMAPF : We measure the runtime and number of collisions with respect to the number of agents. Auction-based planners take up $\frac{1}{4}\times$ the time for $10\times$ the number of agents compared to MAPF solvers in SOCIALMAPF settings (Refer to Fig. 6). (a) Doorway Scenario. (b) Hallway Scenario. (c) Intersection Scenario.

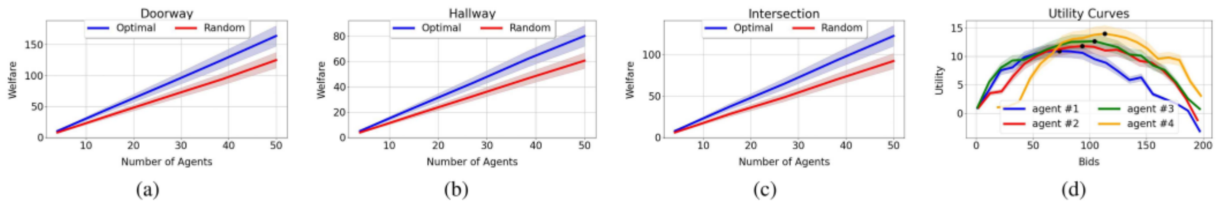

Fig. 4. Auction-based planning is strategy-proof: In Fig. 4(a)–(c), we compare the social welfare of the system using an auction planner (blue line) with that computed using CBS-random (red line). In Fig. 4(d), we plot the utility curves for 4 agents as a function of their bids. The black points denote an agent's private incentive v_i.

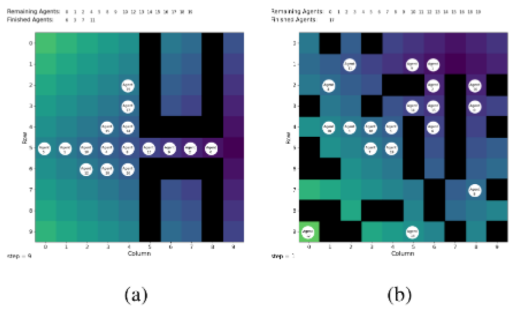

Fig. 5. Visualizations: (left) auction-based planning in the narrow hallway scenario for many strategic agents and (right) auction-based planning in the presence of static obstacles. The darker colors indicate tiles that are closer to the goal with dark purple being the goal point. Lighter colors, correspondingly, indicate tiles farther from the goal.

world navigation with robots, the incentives would need to be more carefully assigned and is a direction of future work.

C. Extending Mechanism Design-Based Planning to SOCIALMAPF With Static Obstacles

We test mechanism design-based planning in SOCIALMAPF environments with static obstacles randomly placed throughout the 10×10 grid as visualized in Fig. 5(b). As before, we benchmark the runtime and number of collisions by averaging over 100 trials each. We vary the number of agents from 4 to 15 never exceeding more than 5 seconds in each case. We vary the number of obstacles from 10 to 25 and observe 0 collisions. Note that both the initial configurations of the agents as well as the obstacles are random. We observe that auction-based planning has no trouble in scaling in terms of number of obstacles and number of agents.

D. Optimal Search-Based Algorithms in SOCIALMAPF

We investigate the performance of an optimal search-based algorithm such as conflict-based search (CBS) [6] in SOCIALMAPF. Since CBS is originally designed for the classical MAPF formulation with cooperative agents with full observability, we model uncertainty for the private incentives in the CBS implementation by normally distributing the edge costs in the planning stage and allowing multi-hop path traversal during the execution stage. In Fig. 6, the columns correspond to the doorway, narrow hallway, and corridor intersection, respectively. We benchmark the CBS algorithm (rows 1 and 2) and its variants, CBS-random (rows 3 and 4) and CBS-high priority (rows 5 and 6). In CBS-random, conflicted nodes with equal costs for the child nodes are selected randomly (as opposed to first-come first-serve) whereas in CBS-high priority, the agent with the higher priority is revealed and is always expanded. The first, third, and fifth rows of Fig. 6 test these solvers as the number of strategic agents increases and the second, fourth, and sixth rows test the effect of the constrained geometry of the environment (specifically, we increase the gap size of the doorway or hallway). In all experiments, we average the results over 100 trials. The solid color lines represent the mean value and the associated shaded regions represent the confidence intervals.

In general, we observe that CBS and its variants scale poorly as the number of agents increases and no longer guarantee collision-free trajectories. For more than 4 agents, search-based methods fail to return a solution within the allotted time (20 seconds). Furthermore, we observe that CBS incurs up to 50 collisions in the intersection environment. To test the effect of the social navigation environment, we vary the gap size from 1 (narrow gap) to 9 (open environment) and, as expected, observe that collisions are more when the environment is constrained (smaller gap sizes). Since in this experiment, we keep the number of the agents fixed at 3, we do not observe variations in the runtime.

Finally, existing priority-based MAPF such as priority-based search [51] assume the availability of an optimal priority. That is, these methods do not solve the priority determination sub-problem. Our approach solves, simultaneously, both the priority determination sub-problem as well as the resulting MAPF problem. We compare with LazyCBS and Priority-based Search (PBS) [51] in Table II.

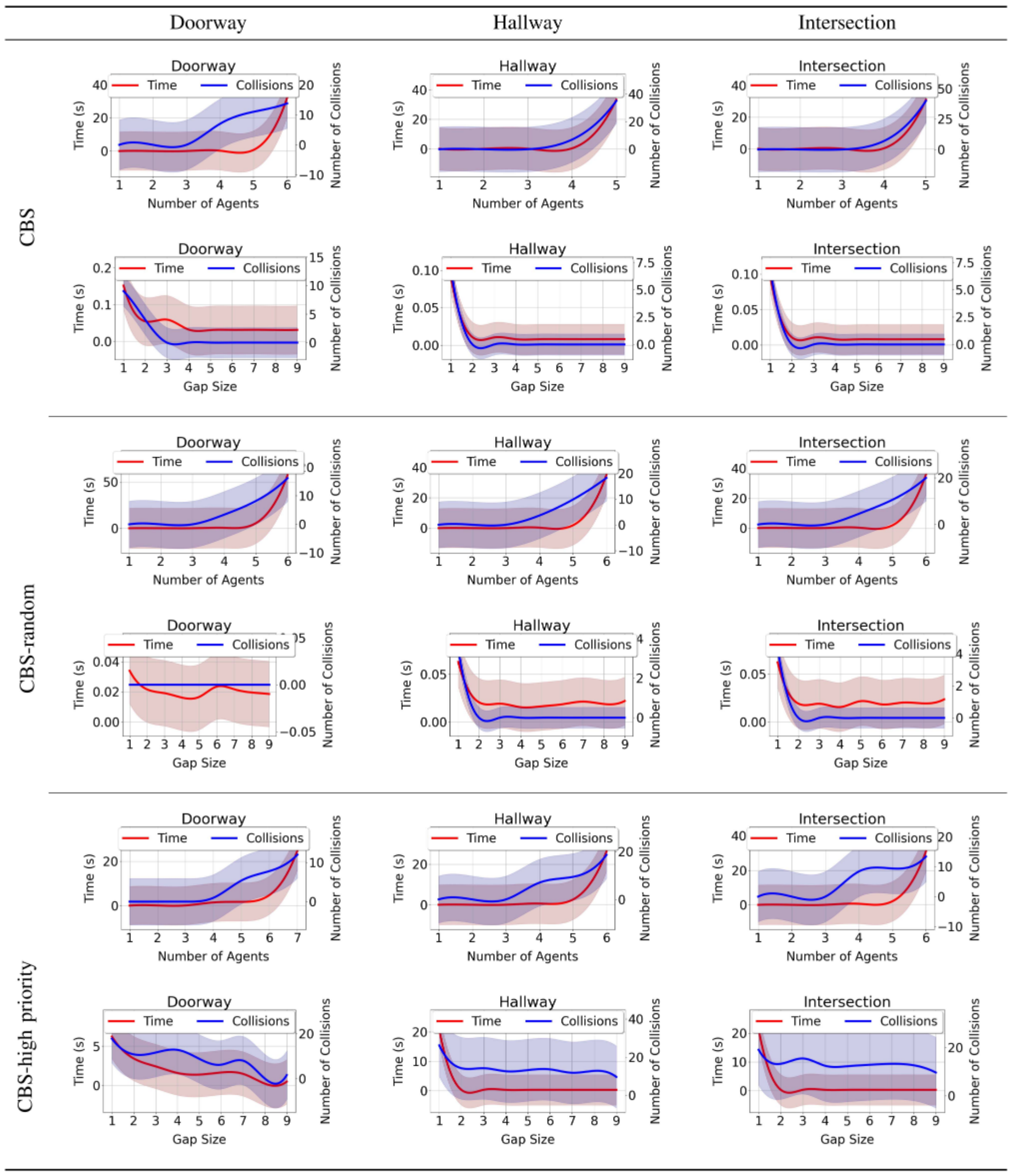

Fig. 6. Optimal search-based planning in SocialMapf : We analyze optimal search-based MAPF solvers such as conflict-based search (CBS) [6] and its variant CBS-random [43] in the SocialMapf regime. See Section V-D for a detailed analysis.

VI. Conclusion, Limitations, and Future Work

We presented a new framework for multi-agent path finding (MAPF) in social navigation scenarios like doorways, elevators, hallways, and corridor intersections among agents with private incentives. We show that existing search-based MAPF solvers are unable to model these unknown incentives and result in collisions due to the inherent uncertainty in agent intentions. Our solution consists of an auction-based approach where we resolve conflicts by incentivising agents to reveal their true intents and executing a conflict resolution protocol based on these true intents. We show that our auction planner results in zero collisions and is 20× more efficient than search-based solvers.

Our work aims to promote research in both artificial intelligence and robotics, revealing previously unnoticed links to other areas of research, such as economics and mechanism design. In each domain, there are fundamental open problems and research directions that need to be solved in order to achieve actual physical robots navigating among humans in the real world. We outline several independent research themes below that build upon the SocialMapf framework. First, SocialMapf solvers to continuous space-time domains, specifically addressing the need to redefine collisions. SocialMapf solvers must also model uncertainty in agent state-spaces and go beyond Gaussian distributions. An important direction is also to perform human incentive estimation via sophisticated techniques such as inverse reinforcement learning. Finally, SocialMapf solvers should explore hybrid combinations of artificial potential field and search-based approaches.

REFERENCES

[1] A. Mavrogiannis, R. Chandra, and D. Manocha, "B-GAP: Behavior-rich simulation and navigation for autonomous driving," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4718–4725, Apr. 2022.

[2] R. Chandra, M. Wang, M. Schwager, and D. Manocha, "Game-theoretic planning for autonomous driving among risk-aware human drivers," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 2876–2883.

[3] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Mag.*, vol. 29, no. 1, pp. 9–9, 2008.

[4] R. Morris et al., "Planning, scheduling and monitoring for airport surface operations," in *Proc. Workshops 30th AAAI Conf. Artif. Intell.*, 2016, pp. 608–614.

[5] D. Silver, "Cooperative pathfinding," in *Proc. AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2005, vol. 1, pp. 117–122.

[6] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, 2015.

[7] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 195, pp. 470–495, 2013.

[8] E. Boyarski et al., "ICBS: Improved conflict-based search algorithm for multi-agent pathfinding," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 223–225.

[9] R. Chandra et al., "Forecasting trajectory and behavior of road-agents using spectral clustering in graph-LSTMs," *IEEE Robot. Automat. Lett.*, vol. 5, no. 3, pp. 4882–4890, Jul. 2020.

[10] C. Mavrogiannis et al., "Core challenges of social robot navigation: A survey," 2021, *arXiv:2103.05668*.

[11] W. Burgard et al., "Experiences with an interactive museum tour-guide robot," *Artif. Intell.*, vol. 114, no. 1-2, pp. 3–55, 1999.

[12] S. Thrun et al., "Probabilistic algorithms and the interactive museum tour-guide robot Minerva," *Int. J. Robot. Res.*, vol. 19, no. 11, pp. 972–999, 2000.

[13] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 797–803.

[14] A. Biswas, A. Wang, G. Silvera, A. Steinfeld, and H. Admoni, "Socnavbench: A grounded simulation testing framework for evaluating social navigation," *ACM Trans. Hum.-Robot Interaction*, vol. 11, no. 3, pp. 1–24, 2022.

[15] N. Tsoi et al., "SEAN 2.0: Formalizing and generating social situations for robot navigation," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 11047–11054, Oct. 2022.

[16] V. V. Unhelkar, C. Pérez-D'Arpino, L. Stirling, and J. A. Shah, "Human-robot co-navigation using anticipatory indicators of human walking motion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 6183–6190.

[17] A. E. Patla, A. Adkin, and T. Ballard, "Online steering: Coordination and control of body center of mass, head and body reorientation," *Exp. brain Res.*, vol. 129, pp. 629–634, 1999.

[18] J. Hart, R. Mirsky, X. Xiao, and P. Stone, "Incorporating gaze into social navigation," 2021, *arXiv:2107.04001*.

[19] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1343–1350.

[20] R. Mirsky, X. Xiao, J. Hart, and P. Stone, "Conflict avoidance in social navigation–A survey," 2021, *arXiv:2106.12113*.

[21] Y. Gao and C.-M. Huang, "Evaluation of socially-aware robot navigation," *Front. Robot. AI*, vol. 8, pp. 420–441, 2022.

[22] R. Mirsky, X. Xiao, J. Hart, and P. Stone, "Prevention and resolution of conflicts in social navigation–a survey," 2021, *arXiv:2106.12113*.

[23] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager, "Game-theoretic planning for self-driving cars in multivehicle competitive scenarios," *IEEE Trans. Robot.*, vol. 37, no. 4, pp. 1313–1325, Aug. 2021.

[24] A. Britzelmeier, A. Dreves, and M. Gerdts, "Numerical solution of potential games arising in the control of cooperative automatic vehicles," in *Proc. Conf. Control Appl.*, 2019, pp. 38–45.

[25] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 9590–9596.

[26] W. Schwarting, A. Pierson, S. Karaman, and D. Rus, "Stochastic dynamic games in belief space," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 2157–2172, Dec. 2021.

[27] W. Sun, E. A. Theodorou, and P. Tsiotras, "Game theoretic continuous time differential dynamic programming," in *Proc. Amer. Control Conf.*, 2015, pp. 5593–5598.

[28] W. Sun, E. A. Theodorou, and P. Tsiotras, "Stochastic game theoretic trajectory optimization in continuous time," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 6167–6172.

[29] J. Morimoto and C. G. Atkeson, "Minimax differential dynamic programming: An application to robust biped walking," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 1563–1570.

[30] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin, "Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 1475–1481.

[31] B. Di and A. Lamperski, "Differential dynamic programming for nonlinear dynamic games," 2018, *arXiv:1809.08302*.

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

[33] B. Di and A. Lamperski, "Newton's method and differential dynamic programming for unconstrained nonlinear dynamic games," in *Proc. IEEE 58th Conf. Decis. Control*, 2019, pp. 4073–4078.

[34] B. Di and A. Lamperski, "Local first-order algorithms for constrained nonlinear dynamic games," in *Proc. IEEE Amer. Control Conf.*, 2020.

[35] R. Stern, "Multi-agent path finding–an overview," *Artif. Intell.*, vol. 11866, pp. 96–115, 2019.

[36] D. M. Kornhauser, G. Miller, and P. Spirakis, "Coordinating pebble motion on graphs, the diameter of permutation groups, and applications," Master's thesis, M.I.T., Dept. Electr. Eng. Comput. Sci., Cambridge, MA, USA, 1984.

[37] R. Luna and K. E. Bekris, "Efficient and complete centralized multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 3268–3275.

[38] B. De Wilde, A. W. Ter Mors, and C. Witteveen, "Push and rotate: A complete multi-agent pathfinding algorithm," *J. Artif. Intell. Res.*, vol. 51, pp. 443–492, 2014.

[39] Q. Sajid, R. Luna, and K. Bekris, "Multi-agent pathfinding with simultaneous execution of single-agent primitives," in *Proc. Int. Symp. Combinatorial Search*, pp. 88–96, 2012.

[40] P. Surynek, "A novel approach to path planning for multiple robots in bi-connected graphs," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 3613–3619.

[41] G. Gange, D. Harabor, and P. J. Stuckey, "Lazy CBS: Implicit conflict-based search using lazy clause generation," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2019, vol. 29, pp. 155–162.

[42] J. Li, W. Ruml, and S. Koenig, "EECBS: A bounded-suboptimal search for multi-agent path finding," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, pp. 12353–12362.

[43] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proc. 7th Annu. Symp. Combinatorial Search*, 2014, pp. 19–27.

[44] O. Amir, G. Sharon, and R. Stern, "Multi-agent pathfinding as a combinatorial auction," in *Proc. 20th AAAI Conf. Artif. Intell.*, 2015.

[45] A. Gautier, A. Stephens, B. Lacerda, N. Hawes, and M. J. Wooldridge, "Negotiated path planning for non-cooperative multi-robot systems.," in *Proc. AAMAS*, 2022, pp. 472–480.

[46] Z. Zhong, M. Nejad, and E. E. Lee, "Autonomous and semiautonomous intersection management: A survey," *IEEE Intell. Transp. Syst. Mag.*, vol. 13, no. 2, pp. 53–70, Summer 2021.

[47] T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2016.

[48] J. J. Chung, A. J. Smith, R. Skeele, and G. A. Hollinger, "Risk-aware graph search with dynamic edge cost discovery," *Int. J. Robot. Res.*, vol. 38, no. 2-3, pp. 182–195, 2019.

[49] R. Chandra and D. Manocha, "GamePlan:Game-theoretic multi-agent planning with human drivers at intersections, roundabouts, and merging," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 2676–2683, Apr. 2022.

[50] N. Suriyarachchi, R. Chandra, J. S. Baras, and D. Manocha, "Gameopt: Optimal real-time multi-agent planning and control at dynamic intersections," in *Proc. IEEE 25th Int. Conf. Intell. Transp. Syst.*, 2022, pp. 2599–2606.

[51] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 7643–7650.