SPARCAS: A Decentralized, Truthful Multi-Agent Collision-free Path Finding Mechanism

Sankar Das[1], Swaprava Nath[1], and Indranil Saha[1]

[1]Indian Institute of Technology Kanpur, {sdas,swaprava,isaha}@iitk.ac.in

September 12, 2019

Abstract

We propose a decentralized collision-avoidance mechanism for a group of *independently controlled* robots moving on a shared workspace. Existing algorithms achieve multi-robot collision avoidance either (a) in a centralized setting, or (b) in a decentralized setting with *collaborative* robots. We focus on the setting with *competitive* robots in a *decentralized* environment, where robots may strategically reveal their information to get prioritized. We propose the mechanism **SPARCAS** in this setting that, using principles of mechanism design, ensures truthful revelation of the robots' private information and provides locally efficient movement of the robots. It is free from collisions and deadlocks, and handles a dynamic arrival of robots. In practice, this mechanism scales well for a large number of robots where the optimal collision-avoiding path-finding algorithm (M*) does not scale. Yet, **SPARCAS** does not compromise the path optimality too much. Our mechanism prioritizes the robots in the order of their 'true' higher needs, but for a higher payment. It uses *monetary transfers* which is small enough compared to the *value* received by the robots.

1 Introduction

Collision avoidance is a central problem in various multi-agent path planning applications, and the problem has been provided different solutions in different paradigms [32, 7, 10, e.g.]. In this paper, we focus on multiple *dynamically arriving* and *independently controlled* robots to move from their source to their destinations using a track network [17, 4, 40]. As many robots share the same track network, they are prone to collide with each other. To avoid that, *three* different approaches are employed in the literature.

In the first approach, offline multi-robot planning algorithms are employed to generate the collision-free paths for all the robots statically [13, 35, 41, 34, 38, 28, 29, e.g.]. However, this approach has two severe drawbacks. First, the *computation time* for generating path plans for a large number of robots may be prohibitively large. Second, they cannot deal with the *dynamic arrival* of new robots to the system without recomputing the whole plan.

In the second approach, the robots independently generate their trajectories offline without the knowledge about the trajectories of the other robots [2, 8, 20, 24, 23, 19, 25, 36, 32, 11, 7, e.g.]. Hence, the trajectories of the robots are not collision-free, but are resolved online in a decentralized manner through information exchange among the potentially colliding robots, assuming that the robots will *cooperate* with their movements. If the simultaneous movements of the robots are not possible, the robots run a distributed consensus algorithm to find a collision-free plan.

In the third approach, robots interact in an auction-like setting and bid for their makespan or demand for resource [21, 3, 22, 6]. Algorithms are designed for certain optimality objectives. However, these approaches do not allow robots to be owned and controlled by independent agents, and therefore do not ensure that these agents *bid* their privately observed information (makespan or demand of resource) *truthfully* to each other or to the planner. However, the combinatorial auction reduction of the MAPF problem [1] does satisfy truthtelling, but it is centralized and therefore has the same limitations of time complexity and dynamic arrival. The other strand of literature [33, 12] on non-cooperative robots consider other robots as dynamic obstacles and solve computationally hard mixed integer programs in a centralized manner.

We note that in many commercial settings the robots are controlled by independent operators, e.g., in autonomous vehicle movements, and the robots can potentially *manipulate* their bids for a prioritized scheduling. In this paper, we exploit the full strength of mechanism design to ensure truthtelling along with other desirable properties like collision and deadlock avoidance.

Our Approach and Results

We propose a decentralized mechanism for a group of robots that move on a shared workspace avoiding collision with each other. A collision situation arises when multiple robots try to simultaneously access a common location. The competitive robots in our setting engage in a *spot auction* and bid the amount they are willing to pay to get access to those locations. In our protocol, the robots that 'win' the auction get access to move, but for a payment. The challenge in designing such a protocol is to choose the winners and their payments such that the robots provide their private *valuations* for the access locations *truthfully*.

To this end, we consider a quasi-linear payoff model [31, Chap 10] for the robots and propose **SP**ot **A**uction-based **R**obotic **C**ollision **A**voidance **S**cheme (**SPARCAS**). We show that **SPARCAS** is *decentralized, collision-free, deadlock-free*, and *robust against entry-exit* (Claim 1). The mechanism ensures that every competitive robot reveals its private information truthfully in this spot auction (Theorem 1), does not have a positive surplus of money, and is *locally efficient* (Theorem 2).

We run extensive experiments in §6 to evaluate the performance of **SPARCAS** in practice. We show that **SPARCAS** (1) scales well with large number of robots, (2) takes much less planning and execution times compared to that of M* [38] and prioritized planning [35], (3) does differentiated prioritization of different classes of robots, and (4) handles dynamic arrivals smoothly.

We have simulated our mechanism for up to 500 robots in Python experimental setup and up to 10 TurtleBots [14] using ROS [26]. Successful simulation in ROS promises that the proposed mechanism can be implemented in a real multi-robot system.

2 Problem Setup

Define $[n] := \{1, \dots, n\}$. Let $N = [n]$ be the set of robots that are trying to travel from their sources to destinations in a *decentralized* manner over a graph $G = (V, E)$, where V is the vertex set and E is the edge set. All edges represent full-duplex paths, i.e., robots can travel on both directions on this edge following usual traffic rules. Time is discrete and is denoted by the variable t. Every edge in the graph is partitioned into slots where a robot can stand at any given time step. Every slot is uniquely numbered between $1, \dots, S$, where S is the total number of slots. We represent the k-th slot by x_k, $k \in [S]$. We call every vertex having a degree of three or more an *intersection* point, since robots moving into such vertices cannot guarantee to avoid a collision without coordinating with each other. In practice, an actual intersection will be a roundabout having a fixed number of slots. Every roundabout can hold robots equal to the number of slots at any given time and the movement of all the robots is unidirectional (e.g., counter-clockwise). An illustration of the paths and the roundabout is given in Figure 1. A vertex with degree larger than *four* can be equivalently represented using a larger roundabout that can accommodate that many number of full-duplex paths.

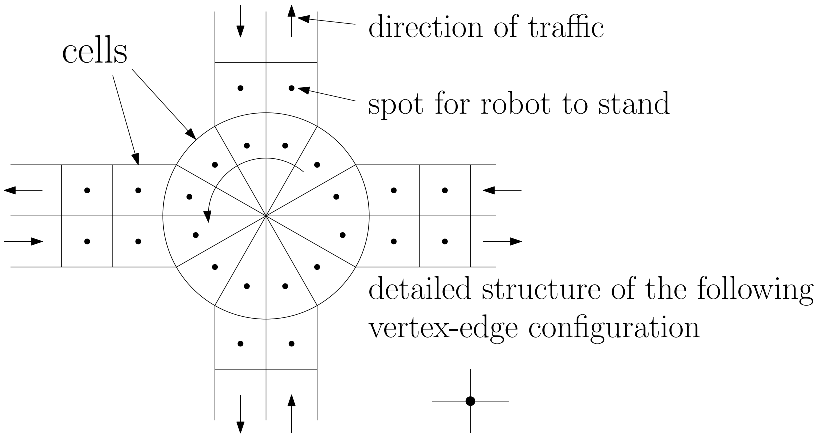

cells
direction of traffic
spot for robot to stand
detailed structure of the following
vertex-edge configuration

Figure 1: Illustration of a vertex with four incident full-duplex edges.

We assume that these robots are independently controlled (or owned) by agents who have no information about other robots' locations, sources, and destinations. Each robot has a deadline to reach its destination, which contributes to its *value* at each time step, e.g., monetary gain from the transfer of a shipment to a customer. A necessary goal of this traversal is to ensure that *no collision* occurs between the robots.

In a decentralized traversal plan of the robots, every robot decides its path based on its local information, e.g., those received through its own sensors and shared by other robots in close proximity. Let the path of robot $i \in N$ be denoted by $P_i := (x_{k_1}, x_{k_2}, \dots, x_{k_{l(i)}})$, where x_{k_j}'s are the slots of the graph leading from the source to the destination of i. When two different robots reach the same slot at the same time, a collision occurs, which each robot wants to avoid. Collision can occur (a) at a roundabout where two robots intend to move to the same empty cell, or (b) on a cell in the edge where a robot is stationary and another robot moves into the same cell. Let the current location of robot i at t is denoted by $\ell_i(t)$, and the next intended location is $n_i(t)$.

We assume that a robot's value is calculated internally by the entity that controls it by considering several factors, e.g., its proximity to the deadline, the type of objects it is carrying (high, if it is carrying a priority shipping item), and all such factors are consolidated into a real number $v_i(t)$ for robot i if it can move to its next location $n_i(t)$ at time $t+1$. The value is zero if the robot does not move at t.[1]

In this paper, our primary objective is to find a *decentralized mechanism* that (a) uses the current and intended location of the nearby robots and ensures truthful revelation of their valuations, (b) avoids collision, deadlocks, (c) maintains fairness by prioritized scheduling based on higher willingness-to-pay. We assume that the robots are capable of communicating with each other within close proximity and can compute their plans of movement. These objectives are formally defined in Section 4.

3 Decentralized Mechanisms

We develop decentralized mechanisms that avoid collision dynamically using the principles of mechanism design theory [5]. We provide two mechanisms and show how the latter improves the performance even though both of them avoid collision. Every decentralized mechanism discussed in this section is preceded by a computation of the shortest path from the source to destination for every robot.[2]

In the current setup, exactly two robots can engage in a collision scenario. A first approach to avoid collision is to develop a protocol where each robot can independently calculate a plan and move according to it. The mechanism emerging from such decentralized planning has to ensure that multiple robots do not appear at the same cell simultaneously.

Since, each robot $i \in N$ has a value $v_i(t)$ at every time step t, we want the mechanism to yield a decentralized plan for the robots such that it maximizes the sum of their values. Hence, this mechanism is economically *efficient*[3] [31, Chap 10]. In the rest of the paper, we will use the term 'efficient' to denote an allocation that maximizes the sum of the valuations of the agents. If the robots find out that they are leading to a collision, they share information of their values, e.g., $v_i(t)$ for robot i, and an efficient plan of movement is decided to allow the robot with a higher value to move and the other to wait – breaking ties arbitrarily. We provide a formal definition of 'efficiency' in Section 4.

Our first approach is a naïve decentralized mechanism, shown in Algorithm 1 at time t for robot i (the mechanism is repeated at every t for every i until each robot reaches its destination). Note that the other robot j that i may collide with, i.e., has $n_i(t) = \ell_j(t)$ or $n_i(t) = n_j(t)$, must

[1]We assume this for simplicity of the exposition. However, our algorithm works even when the value is non-zero and has the same time complexity.

[2]There are several polynomial-time algorithms to compute such a path [18, e.g.], and we therefore exclude this part from our description of the mechanism. We will, however, include the path computation time in the empirical evaluation to make a fair comparison with other collision-avoidance mechanisms.

[3]In microeconomic theory, the alternative that maximizes the *sum* of the values of all the agents is called 'efficient',

Algorithm 1 A naïve decentralized collision avoidance mechanism for robot i

```
1: Input: cells $\ell_i(t), n_i(t)$, and value $v_i(t)$ of robot $i$
2: Output: a decision for robot $i$ to STOP/GO
3: if $n_i(t) = \ell_j(t)$, for some $j \neq i$ then
4:     STOP at $t + 1$
5: else
6:     announce the tuple $(\ell_i(t), n_i(t), v_i(t))$ and receive the same from other nearby robots
7:     if $n_i(t) = n_j(t)$, for some $j \neq i$ then
8:         if $\{v_j(t) > v_i(t)\}$ or $\{v_j(t) = v_i(t)$ and $j > i\}$ then
9:             STOP at $t + 1$
10:        else
11:            GO at $t + 1$
12:    else
13:        GO at $t + 1$
```

be close to robot i and therefore can communicate with i to announce its current and next cell.

This mechanism is efficient given the value information, i.e., v_i's, of the agents at every t are accurate. Since the robots in our setup are independently controlled, these information are private to the agents. Collision mitigation in such scenarios needs a *mechanism* that *self-enforces* the agents to reveal it *truthfully*. This is important in a setting with competing robots, where a robot can overbid its value to ensure that it is given priority in every collision scenario. The naïve mechanism in the current form would fail to ensure efficiency if it uses only the reported values since the strategic robots will overbid to increase their chance of being prioritized.

This is precisely where our approach using the ideas of mechanism design [5] is useful. In mechanism design theory, it is known that in a private value setup, if the mechanism has no additional tool to penalize overbidding, only degenerate mechanisms, e.g., *dictatorship* (where a pre-selected agent's favorite outcome is selected always), are truthful [15, 30]. This negative result also holds in the robot collision setting since the Gibbard-Satterthwaite result extends to settings with cardinal values as well. A complementary analysis by Roberts [27, Thm 7.2] shows that a dictatorship result reappears under certain mild conditions in a *quasi-linear* setting (which is our current setting and is formally defined later), unless monetary transfers are allowed. Money, in these settings, is used as a means of transferring value from an agent to another, and it is shown to help by unraveling the true values of the agents. In this paper, we, therefore, use monetary transfers among the agents to ensure truthful value revelation at every round.

Robot payoff model: At every time step t, given the current position of the robots, denote the set of feasible next-time step configurations by $A(t+1)$. Hence, for every feasible configuration $a \in A(t+1)$, robot i's valuation is given by $\mathbf{val}_i(a, v_i(t)) = v_i(t)$ if robot i is allowed to move to $n_i(t)$ under a, and zero otherwise. The mechanism also recommends robot i to pay $p_i(t)$ amount of money – which can be negative too, meaning the robot *is paid*. The net payoff of robot i is

and it is known that it gives rise to a lot of other desirable properties. For a complete discussion, see [5].

given by the widely used *quasi-linear* formula [31, Chap 10]

$$\mathbf{val}_i(a, v_i(t)) - p_i(t). \tag{1}$$

A strategic robot reports its private information $v_i(t)$ to maximize its net payoff. We assume that the robots are *myopic*, i.e., consider maximizing their immediate payoff. A mechanism in such a setting is defined as follows.

Definition 1 (Mechanism) *A mechanism* $(\mathbf{f} \coloneqq [f_t]_{t=1,2,\ldots}, \mathbf{p} \coloneqq [p_{i,t}]_{i\in N, t=1,2,\ldots})$ *is a tuple which decides the* allocation *and* payment *for every robot at every time step* t*. Here the allocation function at time* t *is given by* $f_t : \mathbb{R}^n \mapsto A(t+1)$ *that decides the next location of every robot* $i \in N$*, and* $p_{i,t} : \mathbb{R}^n \mapsto \mathbb{R}$ *denotes the payment made by the robot. Hence,* $f_t(v_1(t), \ldots, v_n(t))$ *and* $p_{i,t}(v_1(t), \ldots, v_n(t))$ *denote the allocation and payment of robot* i *respectively at time step* t*.*

We assume that the robots agree to a mechanism $(\mathbf{f}, \mathbf{p})$ and compute them locally and follow the movement given by f and pay according to p to a central authority. In this paper, we consider allocation and payment functions to be stationary, i.e., independent of t.

Under this setup, the naïve decentralized mechanism (Algorithm 1) can be made truthful by adding the following *payment rule*. The robot that is prioritized pays the amount equal to the bid of the robot that stops at that time. This is the well-known *second price auction* [37], which is known to be truthful, and we run this at every instant and for every potentially colliding pair of robots.

We notice that in Algorithm 1, the robots do not simultaneously move to the same cell and therefore successfully avoid collision in a decentralized manner. However, this mechanism can lead to the following *deadlock* scenario. Suppose the intersection is of four full-duplex paths with four cells at the intersection (Figure 2). If there are two robots at the intersection and two more are attempting to enter, and both the entering robots wins the auction step (Step 8) in Algorithm 1, it will lead to the four cells at the intersection occupied with four robots. If these robots' next locations are the current locations of the robots already in the intersection, none of them can move under this mechanism (see Figure 2).

To avoid such a deadlock, a mechanism cannot allow more than $(m-1)$ robots to enter an intersection of capacity m. Our proposed decentralized mechanism **SPARCAS** (**SP**ot **A**uction-based **R**obotic **C**ollision **A**voidance **S**cheme) considers the quasi-linear payoff model, takes account of the known facts of mechanism design, and modifies the naïve mechanism to avoid such a deadlock along with other desirable properties. For a given graph G, let the set of indices of the intersection vertices be represented by K. Denote the cells of intersection vertex $k \in K$ by $I_k(V)$. **SPARCAS** modifies how the robots enter an intersection. At every time step t, for intersection k, it finds the set of robots $N_k(t)$ that are either inside the intersection or are attempting to enter the intersection. From the information shared by all the robots $(\ell_i(t), n_i(t), v_i(t)), i \in N_k(t)$, the mechanism allows every robot $i \in N_k(t)$ to find the feasible next-step configurations $A_k(t+1)$ at that intersection. The feasible configurations ensure that there are no more than $(m-1)$ robots in the intersection of capacity m. The set of feasible next-step configurations $A(t+1)$ is an union of the individual feasible configurations $A_k(t+1)$ at each intersection $k \in K$. At the non-intersection cells, every robot advances to a cell in its shortest path unless that is already

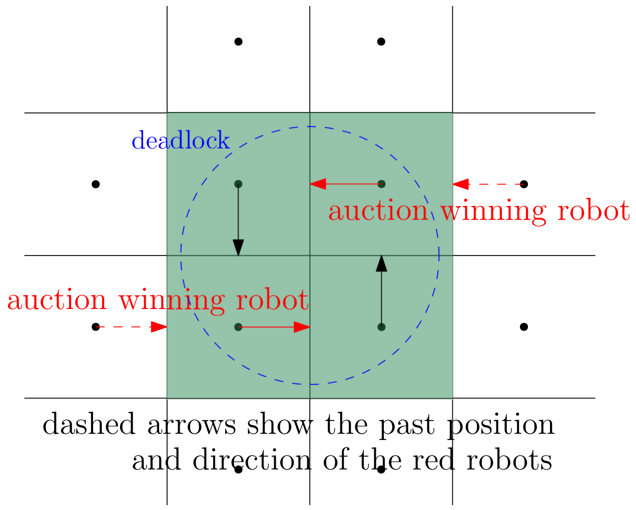

deadlock
auction winning robot
auction winning robot
dashed arrows show the past position and direction of the red robots

Figure 2: Example of a deadlock scenario. Arrows denote their intended direction of move.

occupied by robot – we keep these configurations out of the allocations (e.g., $A_k(t)$'s) since we do not need a collective decision there. The proposed mechanism picks the configuration

$$a_k \in \underset{a \in A_k(t+1)}{\operatorname{argmax}} \sum_{i \in N_k(t)} \mathbf{val}_i(a, v_i(t)). \tag{2}$$

The configuration maximizes the *social welfare* (sum of the values of all the agents) of all the robots that are either inside or are entering the intersection. Denote this configuration by $a_k^*(t+1)$. Similarly, we can define a *welfare maximizing* configuration excluding robot i as follows.

$$a_k^{N_k(t)\setminus\{i\}} \in \underset{b \in A_k^{N_k(t)\setminus\{i\}}(t+1)}{\operatorname{argmax}} \sum_{j \in N_k(t)\setminus\{i\}} \mathbf{val}_j(b, v_j(t)). \tag{3}$$

This choice maximizes the social welfare of all the robots except i. Denote this configuration by $a_k^{*,N_k(t)\setminus\{i\}}(t+1)$. Define the following expression for payment

$$\begin{aligned} p_i(t) := & \sum_{j \in N_k(t)\setminus\{i\}} \mathbf{val}_j(a_k^{*,N_k(t)\setminus\{i\}}(t+1), v_j(t)) \\ & - \sum_{j \in N_k(t)\setminus\{i\}} \mathbf{val}_j(a_k^*(t+1), v_j(t)) \end{aligned} \tag{4}$$

We are now ready to present our proposed mechanism **SPARCAS**. For agent i at time t, it is described in Algorithm 2. As before, the mechanism is repeated at every t for every i until each robot reaches its destination.

Algorithm 2 describes the mechanism from the individual agents' point of view. The consolidated payment collected by the trusted authority at intersection k at t from all the robots at that intersection is distributed equally to the robots that were not part of intersection k at t. Therefore, **SPARCAS** does not accumulate money from the agents.

Algorithm 2 **SPARCAS** for robot i at time t

```
1: Input: cells $\ell_i(t), n_i(t)$, and value $v_i(t)$ of robot $i$
2: Output: a decision for robot $i$ to STOP/GO
3: if $\ell_i(t), n_i(t) \notin I_l(V),\ \forall l \in K$ then
4:     if $n_i(t) = \ell_j(t)$, for some $j \neq i$ then
5:         STOP at $t + 1$
6:     else
7:         GO at $t + 1$
8: else
9:     announce the tuple $(\ell_i(t), n_i(t), \hat{v}_i(t))$ (reported value $\hat{v}_i(t)$ can be different from the true
       value $v_i(t)$) and receive the same from other nearby robots
10:    consider announced tuples of other agents who are in/heading to the same intersection:
       $(\ell_j(t), n_j(t), \hat{v}_j(t)),\ j \in N_k(t) \setminus \{i\}$, where $k$ is s.t. $n_i(t) \in I_k(V)$
11:    compute $a^*_k(t+1)$ (Equation (2)) and STOP/GO according to that recommendation in time
       $t + 1$
12:    pay $p_i(t)$ (Equation (4)) to a trusted authority (e.g., warehouse manager)
```

Locally centralized via intersection manager: **SPARCAS** does not need any synchronization among the robots except that they all follow a common clock (which is also a standard assumption in robot collision avoidance literature [38, 39, 35, e.g.]). Yet, in Section 5, we show that it satisfies many desirable properties. However, there is some computational redundancy in **SPARCAS**. Step 11 of Algorithm 2 is computed by every robot involved in the mechanism near an intersection point, and this makes the mechanism completely decentralized. A presence of an intersection manager (a trusted intermediary) at $I_k(V)$ who can collect the reported $(\ell_i(t), n_i(t), \hat{v}_i(t)), \forall i \in N_k(t)$ and compute it once and inform all the robots in $N_k(t)$ could substantially reduce the cost of computation of every robot. Though this will come at a slight compromise in the decentralized feature of **SPARCAS**, we want to make the reader informed about both the versions of the mechanism.

In the following section, we define certain desirable properties for decentralized robot collision avoidance schemes and show that **SPARCAS** satisfies all of them.

4 Design Desiderata

In a decentralized robot path planning, the agents choose their own route from the source to the destination. The collision avoidance mechanism ensures a protocol that is applied locally at a potential colliding scenario. The property desirable in such a setting is of *locally efficient* prioritization.

Definition 2 (Local Efficiency) *A robotic collision avoidance mechanism* $(\mathbf{f}, \mathbf{p})$ *is* locally efficient *if for every time* t*, every intersection* $k \in K$*, it chooses an allocation that maximizes the sum* *value of all the robots. Formally, it picks* $\forall t, \forall k \in K$

$$f(v_i(t), i \in N_k(t)) \in \underset{a \in A_k(t+1)}{\operatorname{argmax}} \sum_{i \in N_k(t)} \mathbf{val}_i(a, v_i(t)).$$

However, in a multi-agent setting, $v_i(t)$'s of the robots are unknown to the mechanism, which can only access the reported values $\hat{v}_i(t)$'s. Therefore, the following property ensures that the robots are incentivized to 'truthfully' reveal these information.

Definition 3 (Dominant Strategy Truthfulness) *A mechanism* $(\mathbf{f}, \mathbf{p})$ *is* truthful in dominant strategies *if for every t,* $v_i(t), \hat{v}_i(t), \hat{v}_{-i}(t)$[4]*, and* $i \in N$

$$\begin{aligned} &\mathbf{val}_i(f(v_i(t), \hat{v}_{-i}(t)), v_i(t)) - p_i(v_i(t), \hat{v}_{-i}(t)) \\ &\geqslant \mathbf{val}_i(f(\hat{v}_i(t), \hat{v}_{-i}(t)), v_i(t)) - p_i(\hat{v}_i(t), \hat{v}_{-i}(t)). \end{aligned}$$

The inequality above shows that if the true value of robot i is $v_i(t)$, the allocation and payment resulting from reporting it 'truthfully' maximizes its payoff irrespective of the reports of the other robots.

Since we consider mechanisms with monetary transfer, an important question is whether it generates a surplus amount of money. The following property ensures that there is neither surplus nor deficit.

Definition 4 (Budget Balance) *A mechanism* $(\mathbf{f}, \mathbf{p})$ *is* budget balanced *if for every* t *and* $(v_i(t), v_{-i}(t))$, $\sum_{i \in N} p_i(v_i(t), v_{-i}(t)) = 0$.

In the context of decentralized robot path planning, a mechanism that is robust against robot failures is highly desirable. Also, it is desirable if a robot can start its journey when other robots are already in motion, and the mechanism does not need other robots to re-compute their path plan.

Definition 5 (Entry-Exit Robustness) *A mechanism* $(\mathbf{f}, \mathbf{p})$ *is* robust against entry or exit *of the robots if the properties satisfied by the other robots' path plans are unaffected by an addition or deletion of a robot under the mechanism.*

Centralized collision avoidance mechanisms that compute the paths and recommends that to all the robots are not robust against entry or exit. With every addition or deletion of a robot, the plan has to be recomputed. It is to be noted that decentralization alone cannot guarantee entry-exit robustness. Decentralization only implies that the decisions are taken independently by each of the robots. It does not restrict the way in which they interact with each other. Based on the interaction, the plan may not be robust against entry-exit, as the following example shows.

Example **1 (Decentralized but not Entry-Exit Robust)** *Consider the following decentralized version of the prioritized planning algorithm. Before beginning to move, the robots send their identities on a common channel (assume that there is a wired broadcast channel connecting all the starting*

[4]We use the subscript $-i$ to denote all the agents except agent i, therefore, $v_{-i} := (v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_n)$.

positions), but this common channel is not available when they are on the move. The highest priority robot computes its path and broadcasts it (the priority order is fixed beforehand and is a common knowledge of all the robots). After listening to that plan, the second highest priority robot plans its path considering the former as a dynamic obstacle and broadcast, and the process continues for robots of the next priorities. After all robots compute their plans in this decentralized *manner, they leave their parking slots and follow their pre-decided (but decentralized) plan of movement. This scheme is clearly not robust against entry-exit. A newly joined robot does not have the earlier broadcast messages and hence cannot plan its path. Also if that robot has a higher priority than some of the robots that are moving already, then those lesser priority robots cannot change their path plan.*

In the following section, we show that our proposed mechanism satisfies all these properties. In Section 6, we consider a real warehouse setting and exhibit the performance of **SPARCAS** in practice.

5 Theoretical guarantees

The property of truthfulness is important in the multi-agent setting since it ensures that the allocation decision is taken on the *true* values of $v_i(t)$'s and the actual locally efficient allocations were done.

Theorem 1 **SPARCAS** *is dominant strategy truthful.*

Proof: This proof is a standard exercise in the line of the proof for Vickery-Clarke-Groves (VCG) mechanism [37, 9, 16]. **SPARCAS** follows the VCG allocation and payment locally at every intersection calculated by the robots independently. Hence, the payoff of robot i at intersection k is given by (for brevity of notation, we hide the time argument in every function and write a_k^* as $a_k^*(v_i, v_{-i})$)

$$\begin{aligned}
&\mathbf{val}_i(a_k^*(v_i, \hat{v}_{-i}), v_i) - p_i(v_i, \hat{v}_{-i}) \\
&= \sum_{j \in N_k} \mathbf{val}_j(a_k^*(v_i, \hat{v}_{-i}), \hat{v}_j) - \sum_{j \in N_k \setminus \{i\}} \mathbf{val}_j(a_k^{*, N_k \setminus \{i\}}, \hat{v}_j) \\
&\geqslant \sum_{j \in N_k} \mathbf{val}_j(a_k^*(\hat{v}_i, \hat{v}_{-i}), \hat{v}_j) - \sum_{j \in N_k \setminus \{i\}} \mathbf{val}_j(a_k^{*, N_k \setminus \{i\}}, \hat{v}_j) \\
&= \mathbf{val}_i(a_k^*(\hat{v}_i, \hat{v}_{-i}), v_i) - p_i(\hat{v}_i, \hat{v}_{-i}).
\end{aligned}$$

The first equality is obtained by writing and reorganizing the expression for p_i. The inequality holds since by definition $\sum_{j \in N_k} \mathbf{val}_j(a_k^*(v_i, \hat{v}_{-i}), \hat{v}_j) \geqslant \sum_{j \in N_k} \mathbf{val}_j(a_k, \hat{v}_j)$ for every a_k; in particular, we chose $a_k^*(\hat{v}_i, \hat{v}_{-i})$. The last equality is obtained by reorganizing the expressions again. ■

SPARCAS redistributes the generated money from a particular intersection k to the robots who are not part of it at that time step (see the paragraph following Algorithm 2). Clearly, this does not affect the truthfulness properties. The generated surplus before redistributing can be shown to be always non-negative.[5] The allocation of **SPARCAS**, given by Equation (2), maximizes the social welfare at every intersection. Therefore, it is locally efficient. Hence we get the following theorem.

Theorem 2 **SPARCAS** *is budget balanced and locally efficient.*

SPARCAS satisfies certain properties by construction. The following claim summarizes them and we explain it below.

Claim **1** **SPARCAS** *is decentralized, collision-free, deadlock-free, and robust against entry or exit.*

In **SPARCAS**, each robot near an intersection point does message passing, compute the allocation and then move synchronously according to it. Each of the robots at an intersection computes the same allocation which keeps one block in the intersection empty (see the definition of the allocation set A_k). Thus the robots avoid collision and deadlock, and the mechanism is completely decentralized.

Note that **SPARCAS** depends only on the values reported by the robots that are already in an intersection or are about to enter it. A newly entered robot can at most take part in such an interaction, but cannot change the way other robots in other intersections interact with each other. Hence, the properties that those robots were satisfying before the entry of this robot continue to be satisfied. Hence we get the claim.

In the following section, we investigate the performance of **SPARCAS** in real-world scenarios.

6 Experiments

While **SPARCAS** satisfies several desirable properties of a collision avoidance mechanism, its scalability, time complexity to find a collision-avoiding path, and differentiated treatment with different classes of robots are not theoretically captured in the previous sections. This is why an experimental study is called for. Note that the mechanisms that fall in the third set of approaches in §1 either only use the bidding part of the auctions and not the payment which is essential to guarantee truthfulness in an auction [21, 3, 22, 6] or is a reduced combinatorial auction of the centralized algorithm [1]. Either way they are incomparable with **SPARCAS**, which is truthful and decentralized. We compare **SPARCAS**with two widely used protocols for multi-agent path finding – (a) M* [38]: optimal, but has a significant time complexity, and (b) prioritized planning [35]: suboptimal, but has low time complexity.

To evaluate **SPARCAS** experimentally, we use a 2-D rectangular workspace representing a road network. An example of such a workspace of size 16×16 is shown in Figure 3. A robot picks and delivers an object from and to a cell in the service area. Each robot follows the traffic

[5] The intuition for this claim is that absence of a robot reduces congestion, which leads to the other robots being allowed to move. This increases the sum of the values of the other robots. Therefore, we skip a formal proof of this fact.

rules in the road network, and moves along with the directions of the arrows from the source to the destination. It can change its direction only within an intersection cell, but its options are restricted by the available traffic directions in that particular intersection cell.

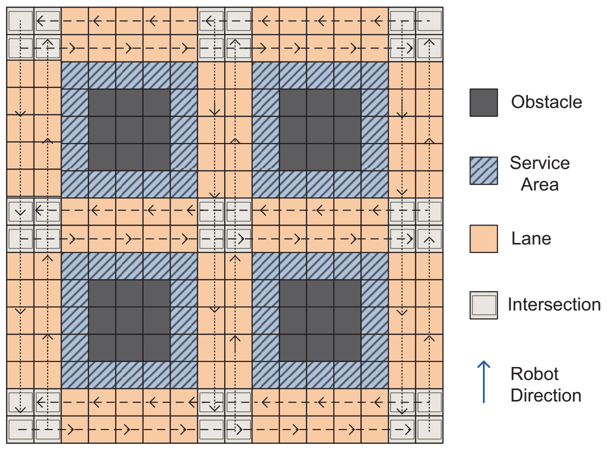


Figure 3: Illustration of a 16×16 workspace.

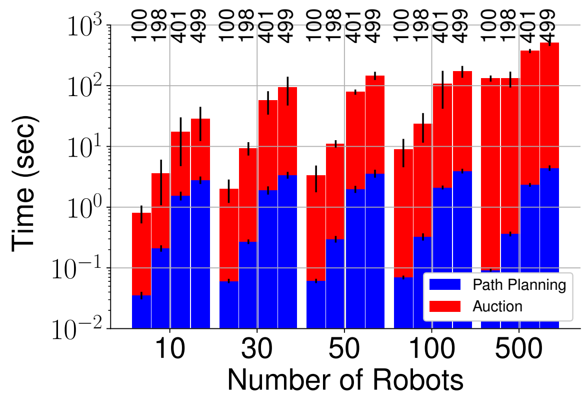


Figure 4: Path planning and auction times of **SPARCAS**. The numbers on the bars show the workspace width for each number of robots.

We have implemented **SPARCAS** in Python.[6] The simulations have been performed in a 64-bit Ubuntu 14.04 LTS machine with Intel(R) Core TM i7-4770 CPU @3.40 GHz $\times$ 20 processors and 128 GB RAM. Each run for a specific number of robots and the workspace size is performed 20 times to calculate the average of the result. The source and goal locations of the robots are selected independently and uniformly at random from the cells of the service area.

A robot is generated uniformly at random from one of the three classes: *economy, regular, premium* having weights 0.02, 0.065, and 0.2 respectively.[7] At every time instant of the experiments, the *valuation* of a robot entering an intersection is assumed to be $(t_{\text{wait}} + 1) \times w$, where t_{wait} is the wait time of the robot till that instant and w is the weight as described above.

6.1 Scalability

We evaluated the scalability of **SPARCAS** on workspaces of four different sizes: 100×100, 198×198, 401×401 and 499×499,[8] and for different number of robots between 10 and 500. Figure 4 shows how the computation time of **SPARCAS** varies with the number of robots and the size of the workspace. The computation time of **SPARCAS** has two components: (a) time for an offline path computation and (b) the time required to run the spot auctions (given by Algorithm 2) online. In our experiments, we assumed that the robots compute their paths to reach the destination using A* algorithm [18]. As this computation can take place in parallel, we count the *maximum* of the plan computation times for all the robots as the offline computation

[6]All codes are available at **https://bit.ly/2lx0fHk**

[7]Weights are increasing with a rough multiplying factor of 3.25.

[8]These numbers ensure a regular pattern of the workspace of Figure 3.

time (part (a)). We assume that each plan execution slot is preceded by a *mini-slot* when the auction can take place at any intersection. We take the product of the duration of the mini-slot dedicated for auction and the maximum number of steps to finish the plan execution for any robot (the makespan) to find the total time spent in spot auction (part (b)). The duration for the mini-slot has been decided based on extensive simulation with different reasonably-sized workspaces and robot populations.

6.2 Comparison with static multi-robot planning algorithms

We compare the performance of **SPARCAS** with that of M* [38] and prioritized planning [35], two state-of-the-art multi-robot path planning algorithms. The original version of M* ensures the optimality of the generated multi-robot plan in terms of the total cost (sum of the time for all the robots to reach destination). However, a variant of M* called *inflated* M* can produce a sub-optimal plan faster than the original M*. We compare **SPARCAS** with inflated M* too. The prioritized planning algorithm also does not provide any optimality guarantee, but can generate collision-free paths much more time-efficiently than both versions of M*.

We compare **SPARCAS**, M*, inflated M*, and prioritized planning for a 100×100 workspace and different number of robots upto 500 robots. We have set a timeout of 1200s. If the computation does not finish before the timeout, we consider the timeout duration as the computation time of the algorithms (which is a lower bound). The experimental results are shown in Figure 5. The lines connect through the average values of the planning times, while the actual planning times for 20 runs for every number of robots are scattered according to their values in the figure. The results show that **SPARCAS** outperforms all the other three algorithms in terms of computation time. M* and inflated M* algorithms do not scale beyond 75 robots for this timeout.

We also compare the makespan (maximum time among all robots to reach destination) and average path execution time (the average of the individual path lengths) for the mechanisms for a workspace of size 100×100 in Figure 6. The method of the plot is identical to Figure 5. The numbers on top of the bars show the percentage of successful completion (not hitting timeout). The results show that there is not much difference in the execution part of these mechanisms and that **SPARCAS** provides a path which is very close in length to that of the optimal path generated by M*.

6.3 Delays of different priority classes

We compare how long robots of different classes (economy, regular, and premium) take to reach their destinations. The first and second subplots of Figure 7 show the average waiting times and payments respectively of the different classes of robots for a workspace size of 100×100. We see that for a reasonable congestion, **SPARCAS** prioritizes the higher classes for a greater payment.

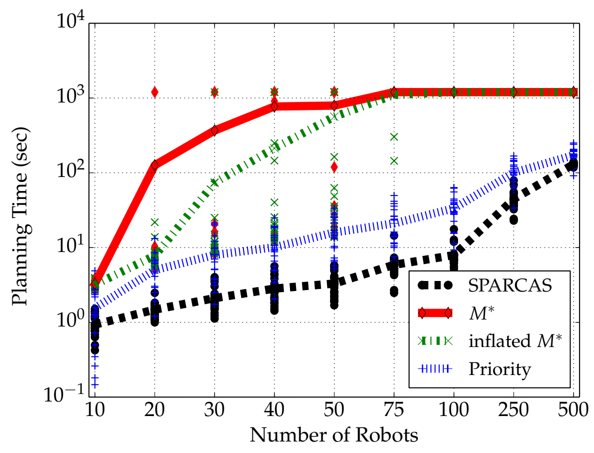


Figure 5: The y-axis shows the total planning time (in sec) of the different algorithms and the x-axis shows the number of robots. For **SPARCAS**, the planning time is the sum of the offline path computation and online spot auction times. Workspace size 100×100.

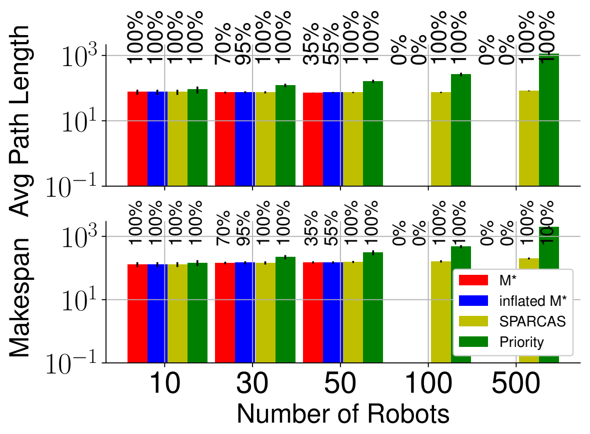
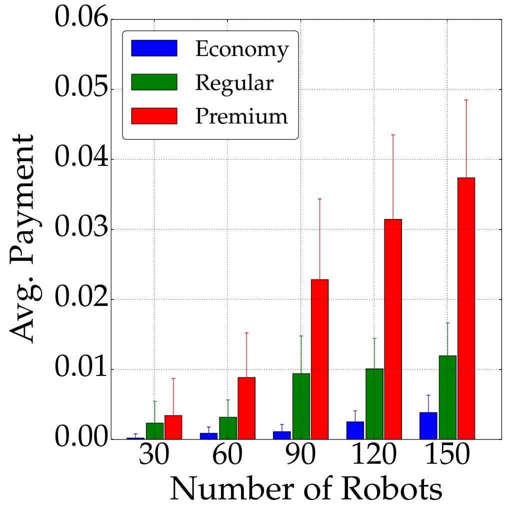


Figure 6: Average path length and makespan for different mechanisms. The numbers on the bar denote the percentage of cases where the mechanism could find a path within the timeout period.



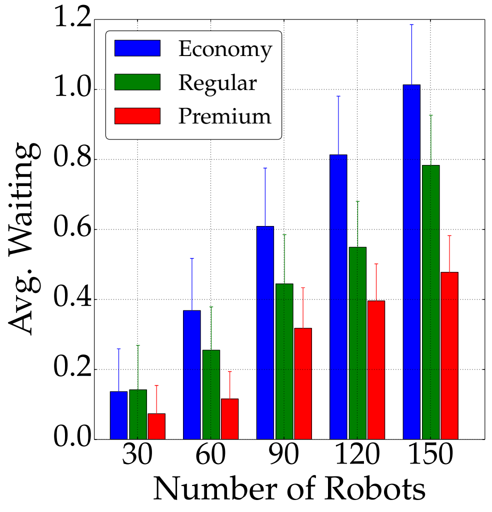


Figure 7: Average waiting time (in sec) and payments under **SPARCAS** for different classes of robots.

6.4 Experiments with ROS

We have simulated **SPARCAS** for up to 10 TurtleBots [14] on ROS [26]. We have used a workspace of size 14×14 (similar to Figure 3) with each grid cell of length 1m. The linear and the angular velocity of the TurtleBots have been chosen in such a way that each motion primitive takes $6.5\mathrm{s}$ for execution. The maximum time for running an auction (decides the length of a mini-slot) observed in all our experiments is about $60\mathrm{ms}$. The video of our experiments is submitted as a supplementary material.

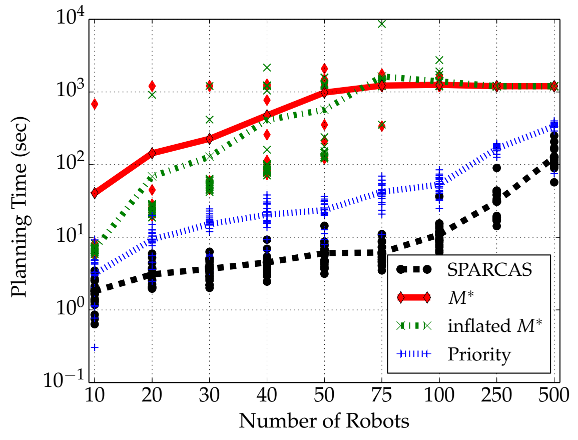
We also ran experiments to find out (a) the payments under **SPARCAS**, and (b) the capability to handle dynamic arrival of robots. The payments are sufficiently small and the dynamic arrivals are gracefully handled in **SPARCAS**. The details of these experiments are in the supplemental material.

6.5 Capability of handling dynamic robot arrival

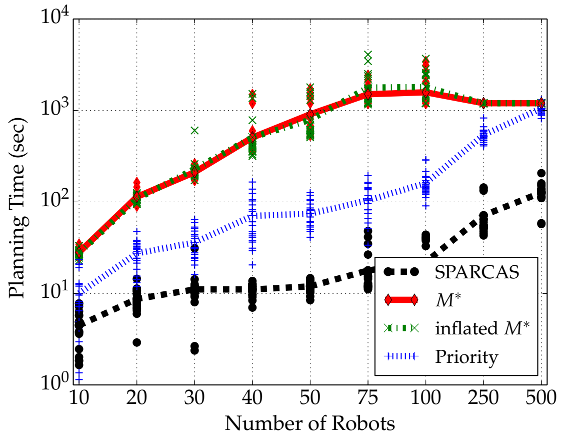
In this section, we study the robustness of **SPARCAS** against dynamic robot arrivals. The setup remains similar to the previous subsection – we mention the differences as follows. We partition the total number of robots into two groups of equal size for this evaluation. The first group of robots arrive at the beginning. The rest 50% of the robots arrive independently and uniformly at random within the time interval of zero and the length of the workspace. In **SPARCAS**, a newly arrived robot computes its own path by using the A* algorithm [18], and starts to follow that path immediately. However, in M*, a newly arrived robot requests path to the centralized M* path planner. The M* planner collects such requests and waits for re-planning until the number of the newly arrived robots exceeds a predefined threshold. In our experiments, this threshold is set to 2. During the re-planning, the M* planner considers the current locations of the previously arrived robots as their source locations and excludes the robots which already have reached their respective goal locations. For prioritized planning, the robots that arrive later are considered to have a lower priority than the robots arrived already, and compute their path considering the previous robots' positions as dynamic obstacles. Figure 8 shows the results of the experiments for two different workspace sizes: 100×100 and 198×198.

6.6 Payments under SPARCAS

SPARCAS is different from the other collision avoiding mechanisms in its use of payments which ensure truthful participation of the competitive robots. Therefore, it is a natural question to find out how large the payments are in comparison with the valuations received by the robots. In our experiment with a workspace size of 100×100, 95.12% of the robots were never needed to pay. For the number of robots: $10, 20, 30, 40, 50$ and 75, the tuple of average valuation and average payment of the robots were $(1.9, 7.5 \times 10^{-4})$, $(2.16, 23.75 \times 10^{-4})$, $(2.2, 47.5 \times 10^{-4})$, $(2.33, 47.5 \times 10^{-4})$, $(2.41, 63 \times 10^{-4})$, and $(2.56, 134.7 \times 10^{-4})$ respectively. Similarly, another important question is whether the robots encounter a negative payoff, i.e., payment more than valuation. In the experiment, we find that the answer to this is negative. Hence the robots have incentives to voluntarily participate in this mechanism. We also find, quite expectedly as was argued in Footnote 5, that the payments by the robots are always non-negative.



(a) Workspace size 100×100



(b) Workspace size 198×198

Figure 8: The y-axis shows the total planning time (in sec) of the algorithms and the x-axis shows the number of robots. For **SPARCAS**, the planning time is the sum of the offline path computation and online spot auction times.

7 Discussions

We presented a scalable decentralized collision avoidance mechanism for a *competitive* multi-robot system using ideas of mechanism design. We prove that it is truthful, efficient, deadlock-free, and budget-balanced. We exhibit experimentally that it is scalable to hundreds of robots, and can handle dynamic arrival without compromising the path-length optimality too much. In future, we would like to extend the algorithm to multi-intersection 'traffic-jam's and conduct experiments on a real multi-robot system.

References

[1] Ofra Amir, Guni Sharon, and Roni Stern. Multi-agent pathfinding as a combinatorial auction. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[2] K. Azarm and G Schmidt. Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation. In *IEEE international conference on robotics and automation*, volume 4, pages 3526–3533, 1997.

[3] Curt Bererton, Geoffrey J Gordon, and Sebastian Thrun. Auction mechanism design for multi-robot coordination. In *Advances in Neural Information Processing Systems*, pages 879–886, 2004.

[4] R. Bogue. Growth in e-commerce boosts innovation in the warehouse robot market. *Industrial Robot: An International Journal*, 43(6):583–587, 2016.

[5] Tilman Börgers. *An introduction to the theory of mechanism design.* Oxford University Press, USA, 2015.

[6] Jan-P Calliess, Daniel Lyons, and Uwe D Hanebeck. Lazy auctions for multi-robot collision avoidance and motion control under uncertainty. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 295–312. Springer, 2011.

[7] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 285–292. IEEE, 2017.

[8] L. Chun, Z. Zheng, and W Chang. A decentralized approach to the conflict-free motion planning for multiple mobile robots. In *IEEE international conference on robotics and automation (ICRA)*, volume 2, pages 1544–1549, 1999.

[9] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[10] Ankush Desai, Indranil Saha, Jianqiao Yang, Shaz Qadeer, and Sanjit A. Seshia. DRONA: a framework for safe distributed mobile robotics. In *Proceedings of the 8th International Conference on Cyber-Physical Systems, ICCPS 2017, Pittsburgh, Pennsylvania, USA, April 18-20, 2017*, pages 239–248, 2017.

[11] V.R. Desaraju and J. P. How. Decentralized path planning for multi-agent teams with complex constraints. *Autonomous Robots*, 32(4):385–403, 2012.

[12] Aparna Dhinakaran, Mo Chen, Glen Chou, Jennifer C Shih, and Claire J Tomlin. A hybrid framework for multi-vehicle collision avoidance. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2979–2984. IEEE, 2017.

[13] M. Erdmann and T. Lozano-Perez. On multiple moving objects. In *ICRA*, volume 3, pages 1419–1424, 1986.

[14] Willow Garage. Turtlebot. *Website: http://turtlebot.com/*, 2011.

[15] A. Gibbard. Manipulation of voting schemes. *Econometrica*, 41:587–602, 1973.

[16] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[17] E. Guizzo. Three engineers, hundreds of robots, one warehouse. *IEEE Spectrum*, 45(7):26–34, 2008.

[18] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2):100–107, 1968.

[19] G. Hoffmann and C. Tomlin. Decentralized cooperative collision avoidance for acceleration constrained vehicles. In *IEEE conference on decision and control (CDC)*, pages 4357–4363, 2008.

[20] M. Jager and B. Nebel. Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, volume 3, pages 1213–1219, 2001.

[21] Michail G Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton J Kleywegt, Sven Koenig, Craig A Tovey, Adam Meyerson, and Sonal Jain. Auction-based multi-robot routing. In *Robotics: Science and Systems*, volume 5, pages 343–350. Rome, Italy, 2005.

[22] Ernesto Nunes and Maria Gini. Multi-robot auctions for allocation of tasks with temporal constraints. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[23] R. Olfati-Saber, J. Fax, and R. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[24] L. Pallottino, V. Scordio, and A Bicchi. Decentralized cooperative conflict resolution among multiple autonomous mobile agents. In *IEEE conference on decision and control (CDC)*, volume 5, pages 4758–4763, 2004.

[25] O. Purwin, R. DfiAndrea, and J Lee. Theory and implementation of path planning by negotiation for decentralized agents. *Robotics and Autonomous Systems*, 56(5):422–436, 2008.

[26] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Y. Ng. ROS: an open-source robot operating system. In *Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA)*, 2009.

[27] Kevin Roberts. *The Characterization of Implementable Choice Rules*, chapter Aggregation and Revelation of Preferences, pages 321–348. North Holland Publishing, 1979.

[28] Indranil Saha, Rattanachai Ramaithitima, Vijay Kumar, George J. Pappas, and Sanjit A. Seshia. Automated composition of motion primitives for multi-robot systems from safe LTL specifications. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, pages 1525–1532, 2014.

[29] Indranil Saha, Rattanachai Ramaithitima, Vijay Kumar, George J. Pappas, and Sanjit A. Seshia. Implan: Scalable incremental motion planning for multi-robot systems. In *7th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2016, Vienna, Austria, April 11-14, 2016*, pages 43:1–43:10, 2016.

[30] M. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.

[31] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations.* Cambridge University Press, 2008.

[32] Jamie Snape, Jur Van Den Berg, Stephen J Guy, and Dinesh Manocha. Smooth and collision-free navigation for multiple robots under differential-drive constraints. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4584–4589. IEEE, 2010.

[33] Ryo Takei, Haomiao Huang, Jerry Ding, and Claire J Tomlin. Time-optimal multi-stage motion planning with guaranteed collision avoidance via an open-loop game formulation. In *2012 IEEE International Conference on Robotics and Automation*, pages 323–329. IEEE, 2012.

[34] Matthew Turpin, Nathan Michael, and Vijay Kumar. Capt: Concurrent assignment and planning of trajectories for multiple robots. *International Journal of Robotic Research*, 33(1), 2014.

[35] J. van den Berg and M. Overmars. Prioritized motion planning for multiple robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2217–2222, 2005.

[36] P. Velagapudi, K. Sycara, and P. Scerri. Decentralized prioritized planning in large multirobot teams. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4603–4609, 2010.

[37] W. Vickrey. Counter speculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961.

[38] Glenn Wagner and Howie Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *IROS*, pages 3260–3267, 2011.

[39] Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1 – 24, 2015.

[40] M. Wulfraat. Is Kiva systems a good fit for your distribution center? an unbiased distribution consultant evaluation. http://www.mwpvl.com/html/kiva_systems.html. Accessed: October 2016.

[41] Jingjin Yu and Steven M. LaValle. Planning optimal paths for multiple robots on graphs. In *ICRA*, pages 3612–3617, 2013.