Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Multi-Agent Pathfinding as a Combinatorial Auction

Ofra Amir
School of Engineering and Applied Sciences
Harvard University
oamir@seas.harvard.edu

Guni Sharon and **Roni Stern**
Department of Information Systems Engineering
Ben Gurion University of the Negev
{gunisharon, roni.stern}@gmail.com

Abstract

This paper proposes a mapping between multi-agent pathfinding (MAPF) and combinatorial auctions (CAs). In MAPF, agents need to reach their goal destinations without colliding. Algorithms for solving MAPF aim at assigning agents non-conflicting paths that minimize agents' travel costs. In CA problems, agents bid over bundles of items they desire. Auction mechanisms aim at finding an allocation of bundles that maximizes social welfare. In the proposed mapping of MAPF to CAs, agents bid on paths to their goals and the auction allocates non-colliding paths to the agents. Using this formulation, auction mechanisms can be naturally used to solve a range of MAPF problem variants. In particular, auction mechanisms can be applied to non-cooperative settings with self-interested agents while providing optimality guarantees and robustness to manipulations by agents. The paper further shows how to efficiently implement an auction mechanism for MAPF, utilizing methods and representations from both the MAPF and CA literatures.

Introduction

The *multi-agent path finding* (MAPF) problem is a generalization of the single-agent path finding problem to multiple agents. A common objective in MAPF is to minimize the *sum of path costs* of all agents. MAPF has many applications, including video games, traffic control (Silver 2005; Dresner and Stone 2008), robotics (Bennewitz, Burgard, and Thrun 2002; Pallottino et al. 2007; Wagner and Choset 2011), and vehicle routing (Kiesel et al. 2012).

In *Combinatorial Auctions* (CA), agents bid on bundles of items allowing them to express complex preferences over bundles. Auction mechanisms aim to find efficient allocations that maximize *social welfare*, that is, maximizing the sum of utilities of the agents. They have been applied in various domains, including airport time slot allocation (Rassenti, Smith, and Bulfin 1982), distributed query optimization (Stonebraker et al. 1994) and transportation service procurement (Song and Regan 2003).

This paper proposes a mapping between MAPF and CA, where the bundles agents bid on are paths. This mapping is carefully designed such that maximizing social welfare corresponds to minimizing the sum of path costs. The flexibility of existing CA mechanisms enables using the proposed mapping to solve a range of MAPF variants, including MAPF problems with heterogeneous agents and those with prioritization over agents. Of particular interest, CA mechanisms can be used in *non-cooperative* settings with self-interested agents that aim to minimize their own path costs rather than the sum of path costs for all agents. Using appropriate CA mechanisms and the proposed MAPF to CA mapping, we provide the first mechanism for non-cooperative MAPF that is optimal and provably robust to certain types of manipulations.

While using CA mechanisms to solve MAPF problems offers benefits, implementing auctions in the MAPF domain poses computational challenges. Specifically, agents have a possibly infinite number of paths to consider when bidding (all possible paths to their goal), and the paths (bundles) can be long (i.e., include many items). These characteristics require significant computation from the agents, in their bidding, and from the auctioneer, in its search of the optimal path allocation. To address these computational problems, we propose to use an iterative CA mechanism (Parkes 2006) in conjunction with efficient representations and methods from the MAPF literature. Iterative CA mechanisms reduce the computational burden of the agents by gradually eliciting their preferences as the auction progresses. Drawing on techniques used in ICTS, a state-of-the-art MAPF algorithm (Sharon et al. 2012c), we store the revealed preferences over bundles using a compact representation and utilize pruning rules when searching for possible path allocations. The resulting MAPF solver is shown empirically to be competitive with recent MAPF solvers.

Market-based approaches, and auctions in particular, have been previously used in multi-agent planning domain for various purposes, including task allocation (Gerkey and Mataric 2002; Hunsberger and Grosz 2000), computation of coordinated MDP policies (Bererton, Gordon, and Thrun 2003), robot exploration (Tovey et al. 2005), and resource allocation (Dolgov and Durfee 2005). In most of these works, agents bid on the tasks to perform whereas in our setting the tasks (goals) are fixed and the problem is finding non-conflicting paths to the pre-specified agents' goals.

The contributions of the paper are threefold. First, we provide a mapping between MAPF and CA. Second, we

show how this mapping can be used to solve variants of the MAPF problem, including optimal cooperative MAPF, non-cooperative MAPF and MAPF with heterogeneous agents. Third, we describe an efficient implementation of an iterative CA for MAPF.

Background

A MAPF problem consists of a graph $G(V, E)$ and a set of agents K. For each agent $k_i \in K$, a unique start vertex s_i, and a unique goal vertex g_i are specified. Each agent can move along the edges of the graph or wait in its location. Both move and wait actions incur a predefined cost. Costs of move and wait actions can vary with the specific edge and vertex. A solution to a MAPF problem is a set of paths, one for each agent from its start vertex to its goal, under the constraint that agents cannot collide. The task is to minimize a cumulative cost function such as the sum of the time steps required for every agent to reach its goal. Many algorithms have been developed for solving MAPF problems optimally (Standley 2010; Sharon et al. 2012c; 2012b) or suboptimally (Silver 2005; Dresner and Stone 2008; Luna and Bekris 2011; de Wilde, ter Mors, and Witteveen 2013; Barrer et al. 2014).

A CA problem is defined by a set of participating agents $K = \{1, ..., k\}$, a set of auctioned items $M = \{1, ..., m\}$, and a valuation function $v_i : 2^M \mapsto \mathbb{R}$ for each agent i that quantifies its preference over bundles of items. Agents submit bids on bundles based on their preferences, declaring the price they are willing to pay for different bundles. The auctioneer then determines an allocation of bundles to agents based on these bids such that no item is allocated to more than one agent. A CA mechanism defines the protocol of communication between the auctioneer and the agents (e.g., how bids are submitted, which bids are allowed), the process used to determine the allocation of bundles, and how the prices agents pay for bundles are computed.

CA mechanisms aim to find an allocation that maximizes *social welfare*, which is the sum of utilities for the agents. The utility for an agent is its valuation of the allocated bundles minus the price it paid to get that bundle. CAs are particularly useful when agents have non-additive valuations of items. For example, an agent might have a high valuation for a bundle that includes a flight to a vacation destination and a hotel, but will not want to buy the hotel stay if it cannot get a good price for the flight. In the MAPF domain agents also have non-additive valuation functions: a path that reaches the goal will be valued highly by an agent, while a subset of that path will not have any value for the agent. Many CA mechanisms have been proposed in the literature, differing in their auction protocols and theoretical properties (Cramton, Shoham, and Steinberg 2006).

Reducing MAPF to CA

A MAPF problem can be reduced to a CA as follows: The set of items M in the auction includes all possible states that agents can be at, i.e., all pairs of vertex and time step. The set of possible bundles in the auction is the set of feasible paths (i.e., all combinations of {vertex, time step} pairs that comprise a path). The valuation function of an agent is determined based on path costs. The reduction is summarized in Table 1.

MAPF	**CA**
Vertex-time pair	Item
Path (set of items)	Bundle
Derived from the path cost	Valuation function
Minimal sum of path costs	Maximal social welfare

Table 1: High-level view of the MAPF to CA reduction.

One of the challenges we address in this work is defining the agents' valuation function in a way that ensures a correspondence between an allocation that maximizes social welfare and a MAPF solution that minimizes the sum of path costs. If we simply set the valuation of a path to be the negative of its cost, the valuations of all paths will be negative and agents will not bid on any bundles (as their utilities would also be negative). Thus, we define the valuation function v_i for each agent i for a path p as $v_i(p) = val(g_i) - cost(p)$, where $val(g_i)$ is some positive value of reaching its goal g and $cost(p)$ denotes the cost of traversing path p. $val(g_i)$ should be set such that the social welfare is maximized only when all agents are included in the allocation (i.e., every agent is assigned with a path to its goal).

Let max_c be an upper bound on the highest path cost (over all agents) in an optimal solution (we discuss later how to obtain such a bound). Further assume that if there are several allocation that maximize social welfare, the one that includes more agents will be preferred.

Theorem 1. *Let π be a MAPF problem for which a solution exists (i.e., non-conflicting paths can be assigned), and let $R(\pi)$ be the corresponding CA problem created by the reduction, where $val(g_i) = max_c \times |K|$. Any allocation for $R(\pi)$ that maximizes social welfare corresponds to a solution to π that minimizes the sum of path costs.*

Proof: First, we show that every allocation that maximizes the social welfare must allocate a non-conflicting path to each agent. Then we show that the resulting sum of path costs is minimal.

Assume that in the allocation that maximizes the social welfare, denoted A_{max}, agent i was not included (i.e., not assigned a path). By construction, the maximal path cost for agent i cannot exceed max_c. Therefore, adding agent i to an allocation adds at least $max_c \times |K| - max_c = max_c \times (|K| - 1)$ to the utility of that allocation. Since i was excluded by A_{max}, the social welfare of all other $K-1$ agents must have increased by at least $max_c \times (|K| - 1)$ by not including i. Since the maximal cost path of a single agent is max_c, the utility of each agent could not have increased by more than max_c. Thus, the maximal added value for the other agents by not including agent i is $max_c \times (|K| - 1)$, which is not greater than the added social welfare of adding i. Therefore, either the allocation that includes i has a higher social welfare, contradicting the assumption, or it has the same social welfare, in which case it will be preferred due to the tie-breaking rule.

Next, we show that this allocation also minimizes the sum

of path costs. Let p_i^a be the path allocated to agent i in allocation a. The social welfare of a given allocation a is

$$\sum_{i=1}^{|K|} val(g_i) - cost(p_i^a)$$

The sum of path costs of the corresponding solution is

$$\sum_{i=1}^{|K|} cost(p_i^a)$$

Since $val(g_i) = max_c \times |K|$ is a constant, the allocation that maximizes social welfare also minimizes the sum of path costs.□

The observant reader may notice that the number of items in $R(\pi)$ can be infinite, as every vertex can be visited by an agent multiple times. In some cases, there is a domain specific upper bound on the time steps that can be planned for, which can be used to limit the number of {vertex, time step} items. More generally, the number of time steps required to solve any MAPF problem is known to be at most $|V|^3$ (Kornhauser, Miller, and Spirakis 1984). Thus, there is no need to consider more than $|V|^3$ time steps and the number of relevant {vertex, time step} items is at most $|V|^4$. This upper bound on the relevant time steps can also be used to set max_c as $|V|^3$ times the maximum edge cost in the graph.

Using the reduction, and based on Theorem 1, a wide range of MAPF variants can be solved with appropriate CA mechanisms, as described in the next sections.

Optimal Cooperative MAPF

A direct implication of Theorem 1 is that any MAPF problem π can be solved optimally by using an optimal CA mechanism that maximizes social welfare in $R(\pi)$. Many such mechanisms exist (Cramton, Shoham, and Steinberg 2006). As an example of how the reduction is used to solve MAPF optimally, consider the MAPF problem depicted in Figure 1. There are two agents, a_1 and a_2, each with a starting location (s_i) and a goal location (g_i). A {vertex, time step} item for vertex X and time step t is denoted by X_t (e.g., A_0 describes being at vertex A in time step 0). The cost of traversing an edge or staying in a vertex is 1. Each bundle is denoted by $b_{i,j}$ where i is the agent interested in the bundle and j is a running index. Agents' valuations of the bundles are also listed (calculated using the $val(g_i) - cost(p)$ valuation function). For simplicity, assume that max_c is 2 (any value larger than 2 would result in the same outcome).

We describe a simple sealed-bid auction mechanism where agents submit bids on each of the paths they are interested in. We further assume that agents are *truthful* and their bids reflect their actual valuations of the paths (we later discuss settings where agents are not inclined to be truthful). The auctioneer chooses a feasible allocation that maximizes its revenue. In the example problem, the maximal revenue of the auctioneer is 4, and can be obtained only when all agents are included in the optimal solution. For example, the auctioneer can achieve this revenue when assigning agent 1 with bundle $b_{1,2} = \{A_0, A_1, B_2\}$ and agent 2 with $b_{2,1} = \{C_0, B_1, D_2\}$.

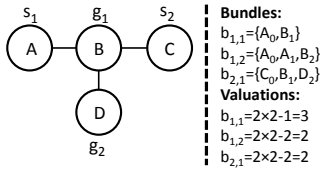

Figure 1: A MAPF problem and a corresponding CA translation.

Heterogeneous Agents and Priorities

If the agents are not homogeneous, then the cost of traversing a given path can differ between agents. For instance, a truck consumes more fuel than a motorcycle. Also, for some agents, finding a short path may be more important than for others. For example, a police car may have higher preference over regular cars and an ambulance might have an even higher preference; a service robot may be preferred over a maintenance robot. These two MAPF variants are easily handled when using CA to solve MAPF by modifying the valuation functions of agents to consider their subjective path costs. This simply requires replacing $cost(p)$ in the reduction above with $cost_i(p)$, which denotes the cost incurred by agent i for traversing p.

For example, if a truck consumes twice as much fuel as a motorcycle consumes, we can set $cost_{truck}(p) = 2 \cdot cost_{motorcycle}(p)$ in the corresponding valuation function. If an ambulance must reach its goal within a time limit, paths that take longer than the time limit will incur additional cost and thus have lower valuations. If the performance of a service robot is twice as important as that of a maintenance robot, its cost evaluation can be defined as half the cost evaluation of the latter, resulting in higher bids by the service robot for the same path.

Self-Interested MAPF

In some MAPF settings, agents may be self-interested, seeking to minimize their own costs without considering other agents' welfare. For example, traffic agents aim to minimize their own time on the road and typically do not consider the overall social welfare. Self-interested behavior by agents can lead to suboptimal outcomes and reduced social welfare (Bnaya et al. 2013). An important property of mechanisms for self-interested agents is *strategyproofness*. When using a strategyproof mechanism (also referred to as a *truthful* mechanism), agents obtain the best outcome by bidding according to their true preferences and cannot individually manipulate the mechanism by not doing so. In the context of MAPF, agents' preferences are derived from the costs they assign to paths. These costs may not be known to the other agents or to the mechanism. Without a strategyproof mechanism, a manipulative agent may choose to report its path costs untruthfully to gain a better path allocation for itself.

Strategic Manipulation in MAPF

To illustrate a scenario where manipulation is possible in MAPF, and agents have an incentive to lie about their preferences, consider the graph shown in Figure 2. s_i and g_i are the start and goal vertices of the agents. Each edge is labeled with its cost. We describe a simple mechanism that

can be used in order to adapt any centralized optimal MAPF algorithm to a setting with unknown path costs: First, agents report their path costs. Then, an optimal MAPF algorithm is run to find the optimal allocation of paths.

If the agents report their costs truthfully, this mechanism will result in the following allocation: a_1 will get the path that goes through X (cost of 2.5) and a_2 the path that goes through Y (cost 2). This will result in a sum of path costs of 4.5. Note, however, that a_1 would have preferred the path that goes through Y as it only costs 2. If a_1 is self-interested, it might try to manipulate the outcome by reporting its costs untruthfully. Specifically, it could report that its cost for the edge from s_1 to X is 4 instead of 1.5. As a result, a social welfare maximizing mechanism would allocate to a_1 the path that goes through Y and will allocate a_2 the path that goes through Z, believing that these paths, with a total cost of 4.7, are the optimal allocation. Thus, this simple mechanism is not strategyproof.

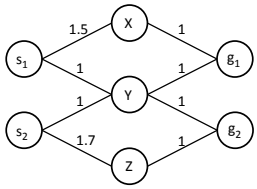

Figure 2: An example MAPF problem where an agent might gain by reporting its preferences untruthfully.

Strategyproof Mechanisms

The auction literature typically considers self-interested settings and has developed strategyproof mechanisms. An example of such an auction mechanism that is robust to the type of manipulation exhibited above is the *Vickrey–Clarke–Groves* (VCG) auction (Clarke 1971; Vickrey 1961; Groves 1973). Informally, when using VCG, the price an agent i pays for an allocated bundle b is based on the "harm" its participation in the auction caused to other agents. Formally, prices are defined as follows. Let a^* denote the allocation chosen in the auction. Each agent i pays $\sum_{i \neq j} v_j(a^*_{-i}) - \sum_{i \neq j} v_j(a^*)$, where $v_j(a^*_{-i})$ is the value agent j would get if agent i had not participated, and $v_j(a^*)$ is the value j obtains from the current allocation.

Returning to the example in Figure 2, assume that both a_1 and a_2 get a value of 10 for achieving their goal and incur a cost for traversing a path p to the goal based on the edges' costs, i.e., $v(p) = 10 - cost(p)$. Then, $v_1(s_1, X, g_1) = 10 - 2.5 = 7.5$, while $v_1(s_1, Y, g_1) = 10 - 2 = 8$. Similarly, the values for a_2 are 8 for the path through Y and 7.3 for the path through Z. If the agents bid truthfully, the auctioneer will allocate the path s_1, X, g_1 to a_1, and the path s_2, Y, g_2 to a_2, as this allocation maximizes the auctioneer's revenue ($7.5 + 8 = 15.5$). a_1 will pay $v_2(a^*_{-1}) - v_2(a^*) = 0$ (it did not cause any "harm"); a_2 will pay $v_1(a^*_{-2}) - v_1(a^*) = 0.5$. The utilities for the agents are thus $u_1 = 10 - 2.5 - 0 = 7.5$ and $u_2 = 10 - 2 - 0.5 = 7.5$.

As mentioned above, VCG is strategyproof, meaning that individual agents cannot manipulate the outcome in a useful way by bidding untruthfully (Vickrey 1961; Clarke 1971; Groves 1973). To illustrate, assume that a_1 would have bid only 5 on the path s_1, X, g_1, and 8 for the path s_1, Y, g_1, and that a_2 bids truthfully as before. The auctioneer will allocate the path s_1, Y, g_1 to a_1 and the path s_2, Z, g_2 to a_2. So a_1 was indeed able to manipulate the outcome. However, such a manipulation does not benefit a_1 due to the Vickrey payment scheme: a_1 will now need to pay $v_2(a^*_{-1}) - v_2(a^*) = 8 - 7.3 = 0.7$. Its utility will thus be $10 - 2 - 0.7 = 7.3$, which is lower than its utility when reporting truthfully. Therefore, the agent has no incentive to lie.

To the best of our knowledge, the only mechanism proposed for self-interested MAPF is the Iterative Taxation Framework (ITF) by Bnaya et al. (2013). This mechanism assigns "taxes" to pairs of location and time, driving agents to follow paths with higher-social welfare rather than selling paths to agents as done in auction mechanisms. ITF has several drawbacks compared to optimal, strategyproof auction mechanisms. First, it is not guaranteed to maximize the social welfare. Second, it is not strategyproof: the mechanism requires knowledge of the start and goal vertices for each agent. A single agent can potentially manipulate the mechanism by reporting its start and goal vertices untruthfully in a way that would result in lower taxes on the path it prefers. Third, it does not support heterogeneous agents as it requires that agents have equal costs over edges and that the mechanism knows these costs.

An Iterative Combinatorial Auction for MAPF

The CA problem resulting from our reduction has several important characteristics. First, a large number of bundles needs to be considered by each agent as the number of possible paths grows exponentially with the cost. Second, the sizes of bundles are potentially large as agents may need to traverse long paths before reaching their goal. These characteristics limit the type of CA mechanism that can be used to solve MAPF problems in practice, in either cooperative or non-cooperative settings.

In *sealed-bid* auctions, such as VCG, each agent needs to bid over all bundles that it may be interested in. In MAPF, this means that an agent i would need to find all paths from s_i to g_i and place bids on them. The number of paths between two vertices in a graph may be exponential in the path length. Moreover, to find an allocation that maximizes its revenue, the auctioneer will need to check the cross product of these potentially exponential number of bids. Thus, a sealed-bid mechanism such as VCG is not feasible for solving MAPF problems in practice.

Iterative CA mechanisms were designed to alleviate the computational problem of eliciting agents' preferences over bundles (Parkes 2006). In contrast to sealed-bid auctions in which agents submit bids for all their possibly desired bundles, in iterative CAs agents are not required to report their valuations for all bundles. Instead, the auction consists of multiple rounds, gradually revealing agents' valuations for bundles. Agents can adjust their bids over time and are not required to determine their exact valuation for every bundle at the onset of the auction. Prices are raised in every round according to the demand for items and agents can submit

new bids as prices change. For example, in a MAPF problem agents might start by bidding on their shortest paths, and then bid on longer paths when prices for shorter paths increase. With this mechanism, agents do not have to compute all of their possible paths ahead of time. They only perform this computation when they are required to.

We describe an implementation of a particular iterative CA, called *iBundle* (Parkes 1999), to solve the MAPF problem. iBundle was chosen because it maximizes social welfare and is strategyproof. For simplicity, we describe the basic iBundle mechanism (Parkes 1999) which was only proven to be strategyproof in some scenarios. However, the iBundle extend and adjust mechanism (Parkes and Ungar 2002) can be similarly used to ensure strategyproofness.

Each round of iBundle is composed of three stages:

1. *Bidding*: agents place bids on their desired bundles. Agents are allowed to place XOR bids, meaning that they would like to get any of the bundles they bid on but not more than one of them.
2. *Winner determination*: the auctioneer determines a provisional allocation of bundles to agents. This is the allocation that maximizes the auctioneer's revenue and is feasible, i.e., no item is allocated to more than one agent.
3. *Price update*: the auctioneer updates the "ask prices" of bundles based on their demand.

The auction terminates when (a) the provisional allocation includes at least one bundle for each of the agents , or (b) all agents place exactly the same set of bids in successive iterations[1]. Next, we describe each of the auction stages in more detail, and show how to implement them efficiently using techniques from the MAPF literature. As a running example, consider the MAPF problem depicted in Figure 3(a)[2]. For every agent i, s_i and g_i denote its start and goal vertices. Agents incur a cost of 1 for moving to a neighboring vertex or staying at their current vertex.

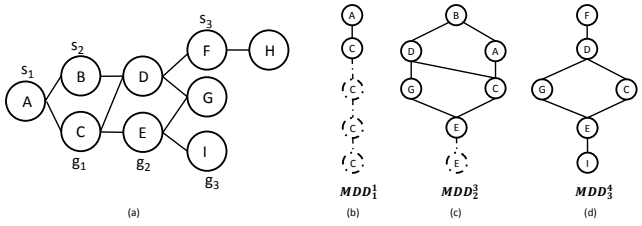

Figure 3: (a) An example MAPF problem with three agents, a_1, a_2 and a_3. s_i and g_i mark the start and goal locations for and agent i; (b) MDD representing a_1's shortest path; (c) MDD representing a_2's shortest paths; (d) MDD representing a_3's shortest paths.

Bidding

When deciding which paths (bundles) to bid on, agents need to consider two factors: the path cost, denoted by $cost(p)$ and the path's current ask price in the auction, denoted $ask(p)$. Agents are assumed to follow a *myopic best response* strategy (Parkes 1999). According to this strategy, agents bid on bundles that maximize their utility given the current prices and the costs of the paths (i.e., their valuation of a path minus its price).

When using this strategy, agents bid on their best bundles at each iteration given the current prices and always place bids with the current price (i.e., they do not consider raising the price before it has been raised by the auctioneer). It has been shown that myopic best response is a dominant strategy for an agent given that other agents are implementing this strategy (Parkes 2001). In MAPF, an agent's best response is to bid on all paths that minimize $cost(p) + ask(p)$ at their current ask price ($ask(p)$).

For example, in the first iteration of an auction for the problem shown in Figure 3(a), a_1's best response bid includes its shortest path, $[\langle A, t_1\rangle, \langle C, t_2\rangle]$. a_2's best response is to place a XOR bid on its three shortest paths:$[\langle B, t_1\rangle, \langle A, t_2\rangle, \langle C, t_3\rangle, \langle E, t_4\rangle]$; $[\langle B, t_1\rangle, \langle D, t_2\rangle, \langle C, t_3\rangle, \langle E, t_4\rangle]$; $[\langle B, t_1\rangle, \langle D, t_2\rangle, \langle G, t_3\rangle, \langle E, t_4\rangle]$. Similarly, a_3's best response includes all of its shortest paths with $cost(p) = 4$.

As prices increase, agents begin to consider longer paths. Specifically, their best response should include all paths that reach the goal with a minimum value of $cost(p) + ask(p)$. For example, given that all paths with $cost(p) = 3$ that a_2 can traverse are priced at 1 ($ask(p) = 1$) and paths with $cost(p) = 4$ are priced at 0, a_2 will bid on all paths with $cost(p) = 3$ at price 1 and all paths with $cost(p) = 4$ at price 0, as they have the same total cost ($cost(p) + ask(p) = 4$).

As the auction progresses, agents will consider bidding on paths with higher costs. The number of paths reaching a goal with a given cost can be exponential in the cost. This growth in the number of bundles agents bid on is atypical of auction problems and requires a compact representation to represent bids. Fortunately, a compact representation for a set of paths was proposed in the MAPF literature (Sharon et al. 2012c). For a given cost c, the set of paths from s_i to g_i can be compactly represented as a multi-valued decision diagrams (MDDs) (Srinivasan et al. 1990), where decision nodes are {vertex, time step} pairs that are part of the paths from s_i to g_i of the corresponding cost. MDDs provide a compact representation as their size is at most $|V| \times c$ (Sharon et al. 2012c). Agents bid on complete MDDs rather than explicitly representing each path (each MDD can be seen as a XOR bid that includes all paths represented by it).

Figure 3(b-d) shows MDDs for the agents' shortest paths to their goals. An MDD representing the paths of agent a with cost c is denoted MDD_a^c. For example, MDD_2^3 (Figure 3(c)) describes all the paths that a_2 can take from its start vertex to its goal at a cost of 3. Each level of the MDD includes all of the vertices that a_2 might be at in a specific time step when traversing a path of cost 3 to the goal. An edge connects an MDD node with nodes in the next level representing vertices that can be arrived at from the vertex represented by the MDD node. For example, at level 1, a_2 could be at node A or D. From node D it could move to either node C or G. To ensure agents do not conflict after they

[1]Condition (b) assumes that agents have deterministic bidding strategies. Thus, if they submit the same bids in consequent rounds, the same allocation will be chosen and prices will not change. Consequently, agents will keep repeating their bids (Parkes 1999).

[2]For a complete description of the auction process for this example problem see Amir et al. (2014).

reach their goals, we align the MDDs by extending them to the length of the longest MDD by replicating the goal node (as shown in dashed lines in the figure).

Winner Determination

At the end of each round of the auction, a *provisional allocation* is determined. The provisional allocation is chosen such that it maximizes revenue and allocates non-conflicting bundles for the set of agents included in the allocation. Ties are broken in favor of allocations that include more agents. At a high-level, the winner determination (WD) problem consists of the following components: (1) generating candidate allocations based on agents' bids; (2) checking whether the allocation is feasible; (3) computing the revenue of each allocation, and (4) choosing the provisional allocation.

For example, in the first round of the auction for the problem shown in Figure 3(a), each agent bids on the MDD corresponding to its shortest paths (Figure 3(b-d)). The possible candidate allocations include all combinations of bundles that agents bid on: (1) Allocating a bundle to one of the agents only. E.g., allocating a_1 its requested bundle ($[\langle A, t_1\rangle, \langle C, t_2\rangle]$) and not allocating any bundle to a_2 and a_3; (2) allocating bundles to a pair of agents. E.g., allocating to a_1 its requested bundle ($[\langle A, t_1\rangle, \langle C, t_2\rangle]$) and allocating to a_2 one of its requested bundles ($[\langle B, t_1\rangle, \langle D, t_2\rangle, \langle C, t_3\rangle, \langle E, t_4\rangle]$), or (3) allocating each of the agents one of their requested bundles.

However, not all candidate allocations are *feasible* as some allocations consist of bundles that include the same item. For example, allocating $[\langle B, t_1\rangle, \langle A, t_2\rangle, \langle G, t_3\rangle, \langle E, t_4\rangle]$ to a_2 and $[\langle F, t_1\rangle, \langle D, t_2\rangle, \langle C, t_3\rangle, \langle E, t_4\rangle, \langle I, t_5\rangle]$ to a_3 is infeasible, as they both include $\langle E, t_4\rangle$. Thus, the candidate allocations need to be checked for feasibility. Finally, of all feasible allocations, the one that maximizes revenue is chosen. Agents that received bundles in the provisional allocation are termed "happy" while agents who did not receive any bundle are termed "unhappy".

The winner determination problem is computationally hard (Sandholm 2002). The pathfinding domain poses additional computational challenges for the winner determination process. First, the set of items is very large due to the time dimension. Second, agents have relative valuations for MDDs. Thus, if agents have unlimited budgets, they will keep bidding on all paths they bid on in previous iterations, resulting in a large number of bids and an increasing number of candidate allocations to be considered. Finally, checking whether an allocation is feasible requires solving a sub-problem of the complete pathfinding problem, namely searching for non-conflicting paths given agents' MDDs.

To address these challenges, a Branch and Bound search can be used to determine the provisional allocation. This approach is similar to that proposed by Sandholm (2002) with some modifications to allow it to work with MDDs rather than searching the space of individual bundles. Feasibility of candidates can be done by searching the MDDs using special prunning rules introduced by Sharon et al. (Sharon et al. 2012c). Computing the revenue of an allocation is straightforward (summing the prices of the MDDs in the allocation).

Price Update

Price updating in iBundle works as follows: initial prices for all bundles are 0. At the end of each round, the prices of bundles that "unhappy" agents bid on are raised to the maximal amount bid on the bundle by an unhappy agent plus ϵ, where ϵ is a parameter of the auction. The MDDs compactly represent all the paths (bundles) agents bid on. Thus, the price of the entire MDD can be increased directly rather than considering each path separately.

For example, recall that at the end of the first iteration of an auction for the problem shown in Figure 3(a), the provisional allocation includes two agents. Let us assume that paths for a_1 and a_3 were arbitrarily chosen to be included in the provisional allocation. Then, the price for the paths a_2 bid on, namely MDD_2^3, is raised by ϵ.

The choice of ϵ affects the efficiency and optimality of the auction. For a small enough ϵ value, iBundle is known to terminate with the optimal allocation when agents use the best response bidding strategy. However, using smaller values of ϵ leads to more iterations of the auction and thus sometimes a larger value is preferred to speed-up the auction (giving up optimality). In MAPF, it is sufficient to set ϵ to the minimal edge cost to preserve optimality, because a lower increment in price will not lead to a change in agents' bids.

Experimental Results

We conducted an empirical evaluation of the proposed approach on a range of standard MAPF benchmarks (Sturtevant 2012), comparing it with state-of-the-art optimal MAPF algorithms. In most cases iBundle performed better than CBS (Sharon et al. 2012a) but worse than ICTS (Sharon et al. 2012c), demonstrating its competitiveness with existing algorithms in the cooperative setting. Importantly, in contrast with prior MAPF algorithms, the proposed approach provides strategyproofness guarantees when applied to non-cooperative settings. For more details about the empirical evaluation, see Amir et al. (2014).

Conclusion

This paper showed a mapping between MAPF problems and CA, where allocations that maximize social welfare correspond to optimal MAPF solutions. This mapping allows using CA mechanism to solve a range of MAPF variants. In particular, we show how CA mechanisms enable optimally solving MAPF with heterogeneous agents, prioritized agents, and self-interested agents. The resulting mechanism is to our knowledge the first strategyproof mechanism for the non-cooperative MAPF problem. The paper outlined several computational challenges for solving MAPF problems with CA mechanisms, and showed that iterative CA mechanisms can be adapted to resolve them. Bridging between auctions and MAPF opens the door for further integration of ideas from these two separately studied areas.

Acknowledgments

We thank David Parkes for helpful discussions and invaluable guidance on auction mechanisms. We thank Barbara Grosz and Ariel Felner for their feedback on the paper.

References

Amir, O.; Sharon, G.; and Stern, R. 2014. An empirical evaluation of a combinatorial auction for solving multi-agent pathfinding problems. Technical Report TR-05-14, Harvard University.

Barrer, M.; Sharon, G.; Stern, R.; and Felner, A. 2014. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *SOCS*.

Bennewitz, M.; Burgard, W.; and Thrun, S. 2002. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robotics and Autonomous Systems* 41(2-3):89–99.

Bererton, C.; Gordon, G. J.; and Thrun, S. 2003. Auction mechanism design for multi-robot coordination. In *Advances in Neural Information Processing Systems*, None.

Bnaya, Z.; Stern, R.; Felner, A.; Zivan, R.; and Okamoto, S. 2013. Multi-agent path finding for self interested agents. In *Sixth Annual Symposium on Combinatorial Search*.

Clarke, E. H. 1971. Multipart pricing of public goods. *Public choice* 11(1):17–33.

Cramton, P.; Shoham, Y.; and Steinberg, R. 2006. *Combinatorial auctions*. MIT press.

de Wilde, B.; ter Mors, A. W.; and Witteveen, C. 2013. Push and rotate: cooperative multi-agent path planning. In *AAMAS*, 87–94. International Foundation for Autonomous Agents and Multiagent Systems.

Dolgov, D., and Durfee, E. 2005. Computationally-efficient combinatorial auctions for resource allocation in weakly-coupled mdps. In *AAMAS*, 657–664.

Dresner, K. M., and Stone, P. 2008. A multiagent approach to autonomous intersection management. *J. Artif. Intell. Res.(JAIR)* 31(1):591–656.

Gerkey, B. P., and Mataric, M. J. 2002. Sold!: Auction methods for multirobot coordination. *Robotics and Automation, IEEE Transactions on* 18(5):758–768.

Groves, T. 1973. Incentives in teams. *Econometrica: Journal of the Econometric Society* 617–631.

Hunsberger, L., and Grosz, B. J. 2000. A combinatorial auction for collaborative planning. In *ICMAS*, 151–158. IEEE.

Kiesel, S.; Burns, E.; Wilt, C. M.; and Ruml, W. 2012. Integrating vehicle routing and motion planning. In *ICAPS*.

Kornhauser, D.; Miller, G.; and Spirakis, P. 1984. Coordinating Pebble Motion On Graphs, The Diameter Of Permutation Groups, And Applications. In *Symposium on Foundations of Computer Science*, 241–250. IEEE.

Luna, R., and Bekris, K. E. 2011. Push and swap: Fast cooperative path-finding with completeness guarantees. In *IJCAI*.

Pallottino, L.; Scordio, V. G.; Bicchi, A.; and Frazzoli, E. 2007. Decentralized cooperative policy for conflict resolution in multivehicle systems. *IEEE Transactions on Robotics* 23(6):1170–1183.

Parkes, D. C., and Ungar, L. H. 2002. An ascending-price generalized vickrey auction. Technical report, Harvard University.

Parkes, D. C. 1999. i bundle: an efficient ascending price bundle auction. In *Proceedings of the 1st ACM conference on Electronic commerce*, 148–157. ACM.

Parkes, D. C. 2001. An iterative generalized vickrey auction: Strategy-proofness without complete revelation. In *Proceedings of AAAI Spring Symposium on Game Theoretic and Decision Theoretic Agents*, 78–87.

Parkes, D. 2006. Iterative combinatorial auctions. *Combinatorial auctions* 41–77.

Rassenti, S. J.; Smith, V. L.; and Bulfin, R. L. 1982. A combinatorial auction mechanism for airport time slot allocation. *The Bell Journal of Economics* 402–417.

Sandholm, T. 2002. Algorithm for optimal winner determination in combinatorial auctions. *Artificial intelligence* 135(1):1–54.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2012a. Conflict-based search for optimal multi-agent path finding. In *AAAI*.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2012b. Meta-agent conflict-based search for optimal multi-agent path finding. In *SOCS*.

Sharon, G.; Stern, R.; Goldenberg, M.; and Felner, A. 2012c. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*.

Silver, D. 2005. Cooperative pathfinding. In *AIIDE*, 117–122.

Song, J., and Regan, A. 2003. Combinatorial auctions for transportation service procurement: The carrier perspective. *Transportation Research Record: Journal of the Transportation Research Board* 1833(1):40–46.

Srinivasan, A.; Ham, T.; Malik, S.; and Brayton, R. K. 1990. Algorithms for discrete function manipulation. In *IEEE International Conference on Computer-Aided Design (ICCAD)*, 92–95.

Standley, T. S. 2010. Finding optimal solutions to cooperative pathfinding problems. In *AAAI*.

Stonebraker, M.; Devine, R.; Kornacker, M.; Litwin, W.; Pfeffer, A.; Sah, A.; and Staelin, C. 1994. An economic paradigm for query processing and data migration in mariposa. In *Parallel and Distributed Information Systems, 1994., Proceedings of the Third International Conference on*, 58–67. IEEE.

Sturtevant, N. 2012. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games* 4(2):144 – 148.

Tovey, C.; Lagoudakis, M. G.; Jain, S.; and Koenig, S. 2005. The generation of bidding rules for auction-based robot coordination. In *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*. Springer. 3–14.

Vickrey, W. 1961. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance* 16(1):8–37.

Wagner, G., and Choset, H. 2011. M*: A complete multirobot path planning algorithm with performance bounds. In *IROS*, 3260–3267. IEEE.