Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Cost-Optimal Planning by Self-Interested Agents

Raz Nissim and **Ronen I. Brafman**
Ben-Gurion University of the Negev
Be'er Sheva, Israel
raznis,brafman@cs.bgu.ac.il

Abstract

As our world becomes better connected and autonomous agents no longer appear to be science fiction, a natural need arises for enabling groups of selfish agents to cooperate in generating plans for diverse tasks that none of them can perform alone in a cost-effective manner. While most work on planning for/by selfish agents revolves around finding stable solutions (e.g., Nash Equilibrium), this work combines techniques from mechanism design with a recently introduced method for distributed planning, in order to find cost optimal (and, thus, social welfare maximizing) solutions. Based on the Vickrey-Clarke-Groves mechanisms, we present both a centralized, and a privacy-preserving distributed mechanism.

Introduction

As our world becomes better connected and more open ended, production becomes more customized, and autonomous agents no longer appear to be science fiction, a natural need arises for enabling groups of selfish agents to cooperate in generating plans for diverse tasks that none of them can perform alone in a cost-effective manner. To build a house, one requires workers with diverse skills, such as architects, lawyers, construction workers, plumbers, electricians, and more. Similarly, to organize a conference, one requires caterers, speakers, AV technicians, publicity, hotel negotiations, program committees, etc. These are just two examples of tasks that require the combined actions of diverse agents and careful planning, as one agent cannot act before other agents have achieved the preconditions for its actions. Agents involved in such tasks realize the need to collaborate with other agents both in planning and execution in order to succeed, but when doing so, they are usually motivated by the desire to maximize their own profit/utility.

Fully cooperative agents can address this problem easily by sending all information to a trusted party that will generate a plan, or by using a distributed planning algorithm, and then employing some agreed upon scheme to share any profit. Self-interested agents, on the other hand, are unlikely to agree to such a scheme for a number of reasons. First, it is easily manipulable – i.e., an agent may report incorrect information about its abilities or its costs in order to increase its share of the profit, thereby decreasing the share of the profit of agents that truthfully report their information. This can lead to a globally sub-optimal plan, discouraging agents from taking part in such a process. Second, many commercial entities are reluctant to reveal certain information to third parties. Whereas an agent must advertise in some way its public capabilities, so others will ask for its services, it prefers *not* to reveal its inner workings – its private state, how it manipulates it, and the cost of such private actions.

Conceptually, the issue of manipulation can be addressed using existing ideas. Semantically, optimal planning algorithms seek a shortest path from the initial state to a goal state in a graph that describes a transition system. In our context, edges in this graph – which correspond to actions – are controlled by different agents. A similar situation arises in network routing, where we seek to route a packet from its source to its destination using a network controlled by different agents, via a shortest path. There, Vickrey-Clarke-Groves (VCG) type mechanisms using distributed implementations of classical shortest path algorithms, such as Dijkstra's (1959), have been developed (Nisan and Ronen 2001; Feigenbaum et al. 2002). These methods provide an attractive distributed solution to this problem, which ideally, we could apply in planning problems. There is a fundamental difference, however: In planning, the transition graph is implicitly defined because the explicit graph is exponential, and thus never constructed explicitly by planning algorithms. Thus, despite their attractiveness, these methods are inapplicable in planning problems.

In this paper we present a mechanism that offers similar properties, thus addressing the problem of manipulation, maintains the privacy of information pertaining to the agents' private state, and is practically efficient. More specifically, our mechanism is distributed, strategy-proof, socially efficient, and privacy preserving. To the best of our knowledge, it is the first mechanism for planning to offer even just the first three properties. Moreover, it is based on state-of-the-art planning technology– making it relatively practical: while there is a price to selfishness, we are still able to optimally solve most solvable benchmark planning problems.

Background

We begin by describing the model used for multi-agent (MA) planning and an overview of mechanism design and

VCG mechanisms.

Planning for Self-Interested Agents

The framework we chose for our work is the MA-STRIPS model (Brafman and Domshlak 2008). This framework minimally extends the classical STRIPS problem to MA planning for *cooperative* agents. The main benefits of using this model are its simplicity, and the fact that it is easily extended to handle the case of *non-cooperative* agents.

Definition 1. *A MA-STRIPS planning task for a set of fully cooperative agents $\Phi = \{\varphi_i\}_{i=1}^k$ is given by a 4-tuple $\Pi = (P, \{A\}_{i=1}^k, I, G)$ with the following components:*

- *P is a finite set of* atomic propositions*, $I \subseteq P$ encodes the* initial state *and $G \subseteq P$ encodes the* goal conditions.
- *For $1 \leq i \leq k$, A_i is the set of actions that the agent φ_i is capable of performing, and each action $a \in A = \bigcup A_i$ is given by its preconditions and effects, as well as its cost.*

A *solution* to a planning task is a legal plan $\pi = (a_1, \dots, a_k)$ which transforms the initial state into a state satisfying the goal conditions. A solution is *optimal* if it has minimal cost (sum of action costs) over all solutions.

Given the partitioning of the actions, we can now distinguish between *private* and *public* atoms and actions. A *private* atom of agent φ is required and affected only by the actions of φ. An action is *private* if all its preconditions and effects are private. All other actions are classified as *public*. That is, φ's private actions affect and are affected only by φ, while its public actions may require or affect the actions of other agents. Note that this implies the public action of an agent may have private preconditions and/or effects. For ease of the presentation of our algorithms and their proofs, we assume all goal conditions are *public*, thus all actions that achieve a goal condition are considered *public* as well. Our methods are easily modified to remove this assumption.

Extending MA-STRIPS from fully-cooperative agents to self-interested agents does not require changing the model itself, but rather changing how it is used, and how information is distributed to different agents. We think of the public part of public actions (i.e., public preconditions and effects) as the *interface* of the agent to the external world – the information it advertises to the external world. The private aspects of public actions, the private variables, and the private actions and their cost is information that agents do not advertise, e.g., because it reveals information about their internal operation that may be useful to competitors, or because of other privacy issues.

Since the self-interest of agents requires defining their utility functions, we make the standard assumption of *quasi-linear* utility functions, where agent φ_i's net utility, given plan π and payments $\mathcal{P}$, is

$$u_i(\pi, \mathcal{P}) = \mathcal{P}_i - cost_i(\pi)$$

where $\mathcal{P}_i$ is the payment given to agent φ_i for its participation and $cost_i(\pi)$ denotes the sum of the costs of all actions by agent φ_i in π. Finally, we assume that no collusion between agents occurs. This also implies that an agent cannot pretend to be several independent agents.

Mechanism Design and the VCG Mechanism

Mechanism design deals with the problem of optimizing some criteria (e.g. social welfare), when self-interested agents having private information are involved. In the standard (centralized) setting, agents report their private information to a "center", which solves the optimization problem and enforces the outcome.

The Vickery-Clarke-Groves (VCG) mechanism (Vickrey 1961; Clarke 1971; Groves 1973) is one of the most celebrated results of mechanism design. Very generally, given the agents' report of their private information, the center computes an optimal solution and the payments to be made to each of the agents. These payments reflect the impact each agent's participation has on other agents, and are computed based on the solutions to the *marginal* problems $\Pi_{-\varphi_i}$, in which agent φ_i is completely ignored. VCG is *strategyproof*, meaning that each agent's utility-maximizing strategy regardless of other agents' strategies and private information is to *truthfully* reveal its private information to the center. Strategyproofness implies that the agents don't need to model the behavior of others in order to compute their equilibrium strategy – *truth telling is a weakly dominant strategy*. VCG is also *efficient*, meaning that the outputted solution maximizes the total utility to agents over all possible solutions to Π. Both these properties prove useful when applying VCG to our setting of cost-optimal MA planning.

Related Work

Our work relates to two research areas: Multi-agent/ distributed planning and mechanism design for decision making in systems of self-interested agents. We now survey relevant results in both areas.

Recently, several methods for planning in the *cooperative* setting of MA-STRIPS have been presented. *Planning-First* (Nissim, Brafman, and Domshlak 2010), a distributed MA planner based on the *Planning as CSP+Planning* methodology (Brafman and Domshlak 2008), showed scalability in problems having loosely coupled agents. More recently, an approach which uses plan refinement (Torreño, Onaindia, and Sapena 2012), showed state-of-the-art performance in more tightly coupled problems. We note that both methods are privacy preserving, but do not guarantee cost-optimal solutions (Planning-First minimizes the maximal number of public actions in an agent's plan).

Tackling the strategic side of MA planning, work has been done on achieving equilibria by a strategic diagnosis of the planning scenario by the agents (Bowling, Jensen, and Veloso 2003; Ben Larbi, Konieczny, and Marquis 2007). This work attempts to find strategies (plans) for the agents which constitute a Nash Equilibrium, by performing strategic analysis of all possible agent plans. This analysis, where every strategy of every agent must be evaluated against all other strategies, makes these methods ineffective for planning, since even if plan length is bounded polynomially (not the case for many planning benchmarks), the number of available strategies is exponential. This problem pertains to all such "pure" game-theoretic approaches requiring the agents to perform some strategic evaluation of other agents.

In light of this, further work has been done on feasibility conditions of MA planning for selfish agents, aiming to find stable solutions in problems exhibiting certain structure – acyclic agent interaction graph (Brafman et al. 2009; 2010), and safe interaction, where no agent benefits from invalidating another agent's plan (Crosby and Rovatsos 2011). Furthermore, work has been done on using single-agent planners in order to perform plan improvement using best-response, in scenarios where an initial MA plan is available (Jonsson and Rovatsos 2011). We note that all the mentioned approaches do not aim to find an optimal solution.

In the field of mechanism design, much work has been done on using *centralized* incentive mechanisms for distributed systems (Ephrati and Rosenschein 1991; Parkes, Kalagnanam, and Eso 2001). The problem of lowest-cost routing on the Internet provided the motivation for work on *distributed algorithmic mechanism design* (Feigenbaum et al. 2002; Feigenbaum and Shenker 2002). This work presents distributed protocols for computing all-pairs least-cost routes, where the links are associated to strategic agents. The assumption here is that the graphs are given, and are small (one node per agent), since the algorithms require memory polynomial in graph size. This assumption means these methods cannot be used in the planning setting, where the graphs are implicit and exponentially large. The main advantage of these works was that the agents could not benefit by misreporting their costs, so no strategic analysis is required by the agents. Later work (Parkes and Shneidman 2004) discussed the robustness of the algorithm itself to manipulation, and introduced the principles required for achieving an incentive compatible distributed mechanism. These principles are discussed in further detail later on.

Based on these results, the first faithful distributed implementation of efficient social choice problems (M-DPOP) was presented (Petcu, Faltings, and Parkes 2008). M-DPOP is a distributed constraint optimization algorithm implementing the VCG mechanism. It ensures that no agent can benefit by deviating from any part of the distributed protocol. While M-DPOP can be used to solve planning problems, which can be cast as DCOPs, previous work (Nissim, Brafman, and Domshlak 2010; Nissim and Brafman 2012a) shows that distributed *planning* algorithms, especially ones using heuristic forward search, have strong computational benefits, whereas CSP-based methods can barely handle problems that require 3 or more actions per agent.

Centralized VCG Mechanism for Planning

To set the stage for our distributed mechanism, we now describe how to apply the VCG mechanism to our setting of optimal MA planning, when a trusted center exists:

Definition 2 (VCG mechanism for optimal MA planning). *Given a MA planning problem Π in which agents are self-interested and have private information, the mechanism is as follows:*

1. *Agents report their private actions and all action costs to the center.*
2. *Given the agents' reports, the center finds an optimal solution $\pi^\star$ for Π, as well as $\pi^*_{-\varphi_i}$ for each $\Pi_{-\varphi_i}$.*
3. *Each agent receives payment*

$$\mathcal{P}_i = \sum_{j \neq i} (cost_j(\pi^\star_{-\varphi_i}) - cost_j(\pi^\star))$$

In this setting, each agent reports its actions and their costs to the center, which then computes the payment rule, i.e. the payments made to each of the agents for participating in the mechanism. The payment of each of the agents is determined by the center according to the agent's *social cost* – the aggregate impact its participation has on the utilities of other agents. To compute the payments, the center needs to solve Π, as well as the marginal problems $\Pi_{-\varphi_i}$ for each $\varphi_i \in \Phi$. Note that the payments are always non-negative, since an agent's presence may only have a positive effect on solution cost. Therefore, a *pivotal* agent φ_i, i.e. one which improves solution cost by participating, will have $\mathcal{P}_i > 0$ and will be paid for its participation, whereas non-pivotal agents will pay exactly 0. This means the center will run a budget-deficit (i.e. will never make a profit), but no agent will lose by participating. This deficit can be viewed as the agents' profit from taking part in the plan. In a market where numerous agents can perform similar actions at different costs, e.g., due to efficiency differences, proximity, availability of resources, etc., the profit is related to such advantages that one agent may have over others. Note that in this schema, an agent that is essential (i.e., no solution exists without it) will receive a payment of ∞, so the underlying assumption is that no such agent exists.

As an example, consider the well-known *logistics* planning domain, which involves vehicles transporting packages to their goal location. Each of the three agents $\varphi_1 \ldots \varphi_3$ can *pickup/drop* one of two packages p_1, p_2, and can *drive* between two locations A, B. Initially, both packages and the three agents are at location A, and the goal is to move both packages to location B. Action costs differ between agents: φ_1's *pickup/drop* actions have cost 1 for p_1, and cost 2 for p_2. φ_2's *pickup/drop* actions have cost 2 for p_1, and cost 1 for p_2. φ_3's *pickup/drop* actions have cost 2 for both p_1, p_2. All drive actions have cost 1.

Given the report of actions costs, the center now finds an optimal plan having cost 6, in which φ_1, φ_2 *pickup, drive and drop* packages p_1, p_2, respectively. For the marginal problems $\Pi_{-\varphi_1}$ and $\Pi_{-\varphi_2}$, the optimal solution has cost 8. Therefore, $\mathcal{P}_1 = \mathcal{P}_2 = 8 - 3 = 5$. This gives the two pivotal agents the positive utility $u_1 = u_2 = 5 - 3 = 2$. For the non-pivotal agent φ_3, $\Pi_{-\varphi_3}$ has an optimal solution cost of 6, therefore its payment (and its utility) will be 0.

Distributed Implementation of VCG

While the centralized approach finds the optimal solution, maximizes social welfare, and ensures truthfulness of the agents, in many cases it will be impossible to find a trusted central authority. Moreover, even if one exists, this centralized approach is not *privacy preserving*, as agents must reveal their private information to the center.

A desirable alternative is a *distributed* implementation of the VCG mechanism that computes the payments, solves Π, while maintaining some sense of VCG's strategyproofness.

However, such a distributed implementation introduces new opportunities for agent manipulation, as in addition to reporting its private information untruthfully, an agent may deviate from the distributed protocol, unless the right incentives for following the protocol are provided. We present such an implementation next.

In describing our distributed implementation, we make the following assumptions[1], in addition to the ones made in the background section:

1. There is a trusted *bank* which can communicate with the agents and distribute payments.
2. The agents are rational but helpful, i.e. they are self-interested, but will deviate from a protocol only if this makes them *strictly* better off.
3. Every agent can communicate directly with all other agents.
4. The bank may rescind payment if a legal, goal-achieving plan is not found and executed.

Work by Parkes and Shneidman (2004) describes principles that guide the distribution of computation, focusing in particular on the VCG mechanism. A distributed algorithm (protocol) is said to be *ex post faithful* if it is in the best interest of every agent to truthfully follow all aspects of the algorithm (information revelation, message passing, computation, etc.) regardless of other agents' private information, given that all other agents follow the algorithm[2]. An algorithm is said to satisfy the *Partition Principle* if (1) optimal solutions are always obtained for Π and $\Pi_{-\varphi_i}$, given that all agents fully comply with the suggested distributed protocol; (2) agent φ_i cannot affect the distributed computation of the marginal problem $\Pi_{-\varphi_i}$, nor its utility from the outcome; (3) the solution found for Π is correctly executed and the payments are made to the agents. Parkes and Shneidman show that *a distributed algorithm that satisfies the partition principle is ex post faithful.*

The MAD-A* Algorithm

We now describe the distributed protocol which we use for the VCG computations. At its heart is a distributed planning algorithm: MAD-A* (Nissim and Brafman 2012a; 2012b) is a distributed, privacy-preserving variation of A*, which maintains a separate search space for each agent. Each agent maintains an *open list* of states that are candidates for expansion and a *closed list* of already expanded states. It expands the state with the minimal $f = g + h$ value in its open list. When an agent expands state s, it uses its own actions only. This means that two agents expanding the same state will generate *different* successor states.

The messages sent between agents contain the full state s, i.e., including both public and private variable values, as well as the cost of the best plan from the initial state to s found so far, the sending agent's heuristic estimate of s and the last action performed leading up to s[3]. When agent φ receives a state via a message, it checks whether this state exists in its open or closed lists. If it does not appear in these lists, it is inserted into the open list. If a copy of this state with higher g value exists, its g value is updated, and if it is in the closed list, it is reopened. Otherwise, it is discarded. Whenever a received state is (re)inserted into the open list, the agent computes its local h_φ value for this state, and assigns the maximum of its h_φ value and the h value in the received message.

Once an agent expands a solution state s, it sends s to all agents and initiates the process of verifying its optimality. When the solution is verified as optimal, the agent initiates the trace-back of the solution plan. This is also a distributed process, which involves all agents that perform some action in the optimal plan. When the trace-back phase is done, a terminating message is broadcasted.

Algorithms 1-3 depict the MAD-A* algorithm for agent φ_i. Unlike in A*, expansion of a goal state in MAD-A* does not necessarily mean an optimal solution has been found. Here, a solution is known to be optimal only if all agents prove it so. Intuitively, a solution state s having solution cost f^* is known to be optimal if there exists no state s' in the open list or the input channel of some agent, such that $f(s') < f^*$. In other words, solution state s is known to be optimal if $f(s) \leq f_{\text{lower-bound}}$, where $f_{\text{lower-bound}}$ is a lower bound on the f-value of the entire system, including all states in open lists and states in unprocessed messages.

Algorithm 1 MAD-A* for Agent φ_i

```
1: while did not receive true from a solution verification procedure do
2:     for all messages $m$ in message queue do
3:         process-message($m$)
4:     $s \leftarrow$ extract-min(open list)
5:     expand($s$)
```

Algorithm 2 process-message($m = \langle s, g_{\varphi_j}(s), h_{\varphi_j}(s)\rangle$)

```
1: if $s$ is not in open or closed list or $g_{\varphi_i}(s) > g_{\varphi_j}(s)$ then
2:     add $s$ to open list and calculate $h_{\varphi_i}(s)$
3:     $g_{\varphi_i}(s) \leftarrow g_{\varphi_j}(s)$
4:     $h_{\varphi_i}(s) \leftarrow max(h_{\varphi_i}(s), h_{\varphi_j}(s))$
```

To detect this situation, MAD-A* uses Chandy and Lamport's *snapshot algorithm* (Chandy and Lamport 1985), which enables a process to create an approximation of the global state of the system, without "freezing" the distributed computation. Although there is no guarantee that the computed global state actually occurred, the approximation is

[1]similar to the ones made by Petcu et al. (2008)

[2]The weakening of VCG's strategyproofness in the centralized case, to ex post faithful in the distributed case is often referred to as the *cost of decentralization*.

[3]It may appear that agents are revealing their private data because they transmit their private state in their messages. However, as will be apparent in the algorithm, other agents do not use this information in any way, nor alter it. They simply copy it to future states. Only the agent itself can change the value of its private state. Consequently, this data can be encrypted arbitrarily – it is merely used as an ID by other agents.

Algorithm 3 expand(s)

```
1: move $s$ to closed list
2: if $s$ is a goal state then
3:     broadcast $s$ to all agents
4:     initiate verification of stable property $f_{lower-bound} \geq g_{\varphi_i}(s)$
5:     return
6: for all agents $\varphi_j \in \Phi$ do
7:     if the last action leading to $s$ was public and $\varphi_j$ has a public action for which all public preconditions hold in $s$ then
8:         send $s$ to $\varphi_j$
9: apply $\varphi_i$'s successor operator to $s$
10: for all successors $s'$ do
11:     update $g_{\varphi_i}(s')$ and calculate $h_{\varphi_i}(s')$
12:     if $s'$ is not in closed list or $f_{\varphi_i}(s')$ is now smaller than it was when $s'$ was moved to closed list then
13:         move $s'$ to open list
```

good enough to determine whether a stable property currently holds in the system. A property of the system is *stable* if it is a global predicate which remains true once it becomes true. Specifically, properties of the form $f_{\text{lower-bound}} \geq c$ for some fixed value c, are stable when h is a *globally consistent* heuristic function. That is, when f values cannot decrease along a path. In our case, this path may involve a number of agents, each with its h values. If each of the local functions h_φ are consistent, and agents apply the $\max$ operator when receiving a message, this property holds.

Distributed Implementation

Consider now our distributed implementation of VCG which uses the MAD-A* algorithm, presented as Algorithm 4. In this setting, the agents participate in $|\Phi|+1$ sequential MAD-A* searches, beginning with the original problem Π, and continuing with the marginal problems $\Pi_{-\varphi_i}$ for each i. When solving the marginal problem $\Pi_{-\varphi_i}$, messages from φ_i are ignored by all agents, and no messages are sent to it. Once each of the marginal problems $\Pi_{-\varphi_i}$ are solved, each agent $\varphi_j \neq \varphi_i$ knows its local cost in π^* and in $\pi^*_{-\varphi_i}$. This is sufficient in order to send the value $\mathcal{P}_{ij}$, that is, the cost removing φ_i incurs on φ_j, to the bank. Upon receiving all these messages, the bank is able to compute $\mathcal{P}$. Since MAD-A* does not require the agents to reveal their private information, Algorithm 4 is automatically privacy preserving.

Algorithm 4 Selfish-MAD-A*

```
1: Run MAD-A* on $\Pi$ with all agents in $\Phi$ to find $\pi^\star$
2: for all $\varphi_i \in \Phi$ do
3:     Run MAD-A* on $\Pi_{-\varphi_i}$ to find $\pi^*_{-\varphi_i}$
4:     Agents $\varphi_j \neq \varphi_i$ compute $\mathcal{P}_{ij} = cost_j(\pi^\star_{-\varphi_i}) - cost_j(\pi^\star)$ and send it to the bank.
5:     The bank computes $\mathcal{P}_i = \sum_{j \neq i} \mathcal{P}_{ij}$
6: Bank pays $\mathcal{P}_i$ to every agent $\varphi_i$, and broadcasts $\pi^\star$ to all agents.
```

Proof of Correctness

In order to prove the correctness of algorithm 4, we must prove that it satisfies the partition principle (Parkes and Shneidman 2004). Therefore, we must show that (1) MAD-A* computes optimal solutions to Π and $\Pi_{-\varphi_i}$ for all $\varphi_i \in \Phi$, given that all agents comply with the protocol; (2) No agent can affect the solution of its respective marginal problem, nor its utility from the outcome; (3) the optimal solution of Π is correctly executed and all payments are made.

Condition (1) follows immediately from the correctness and optimality of MAD-A*, which requires only that the heuristic used by each of the agents is consistent. The first part of condition (2) holds since agent φ_i does not even participate in the solving of $\pi^*_{-\varphi_i}$, and therefore, cannot influence it in any way. For the second part, agent φ_i's utility $u_i(\Pi)$ is given by

$$\mathcal{P}_i - \text{cost}_i(\pi^\star) = \sum_{j \neq i} (\text{cost}_j(\pi^\star_{-\varphi_i}) - \text{cost}_j(\pi^\star)) - \text{cost}_i(\pi^\star)$$
$$= \text{cost}(\pi^\star_{-\varphi_i}) - \text{cost}(\pi^\star)$$

Since φ_i cannot influence $\text{cost}(\pi^\star_{-\varphi_i})$, it can have a positive influence on its utility only by decreasing $\text{cost}(\pi^\star)$. As $\pi^\star$ is already optimal, any such decrease means either that other agents' costs have been decreased, which will cause the plan to fail (some agent will not agree to such a solution) and the payments to be rescinded, or that φ_i misreported its own costs, which does not change its actual utility. Condition (3) follows directly from our assumption that there exists a trusted bank, which can rescind payments if a legal plan is not executed.

We have shown that Selfish-MAD-A* satisfies the partition principle, and therefore, it is *ex post faithful*.

Optimizing Selfish-MAD-A* by Multigoal Search

In the distributed implementation presented in the previous section, each of the $|\Phi|+1$ problems is solved in isolation. It is clear that when these problems are solved, the generated search spaces overlap, causing some states to be generated (and evaluated) multiple times. For example, in $\Pi_{-\varphi_i}$, all reachable states s having $f(s) \leq \text{cost}(\pi^\star_{-\varphi_i})$ must be expanded. In Π, all these states are reachable, and will be expanded as well.

In order to prevent the duplication of effort Selfish-MAD-A* entails, we propose a *multigoal* variation of MAD-A*, which finds optimal solutions to all $|\Phi|+1$ problems in a single run and on a single (distributed) search space. The main idea is to additionally identify each state s with the set of agents participating in the (currently) best plan leading up to s. If s can be reached via two paths having different participating agents sets, s is duplicated. When a goal state $s^\star$ is reached, it is treated as a candidate solution for $\Pi_{-\varphi_i}$ if φ_i does not participate in the path leading to $s^\star$. Termination detection remains unchanged, except that it is performed for each marginal problem, as well as for Π. $s^\star$ is known to be optimal for marginal problem $\Pi_{-\varphi_i}$, if $f(s^\star) \leq f^i_{\text{lower-bound}}$, where $f^i_{\text{lower-bound}}$ is a lower bound on the f-value of all states for which φ_i does not participate

Table 1: Comparison of centralized A*, centralized VCG, and two versions of distributed VCG. Running time (in sec.) and the number of generated states are shown.

	Time (sec.)				Generated states			
Problem	A*	Cen	Dis	OptDis	A*	Cen	Dis	OptDis
rovers6 (2)	278	282.5	324	290	34M	35M	114M	102M
rovers7 (3)	0.7	2.78	6.62	5.1	62271	316553	2262129	2104573
rovers12 (4)	1	16.7	102.1	67.2	55783	1306970	29412049	24506767
satellite5 (3)	0.06	0.06	2.78	1.1	2817	2961	1928245	932233
satellite6 (3)	0.4	0.4	4.67	2.16	39182	39384	3311637	1763072
satellite7 (4)	2.96	2.98	31.9	11.04	246762	246902	25675038	10208671
transport2 (2)	0.01	0.02	0.2	0.12	166	290	10626	8874
transport3 (2)	3.68	12.1	42.7	33.2	27354	120256	2583063	2207614
transport4 (2)	40.4	52.2	1036	712	112824	154469	24543906	22787598
zenotravel8 (3)	0.06	0.08	0.66	0.32	725	1009	46282	26560
zenotravel9 (3)	20.8	45.4	835.7	611	227670	538352	44768124	35520708
zenotravel10 (3)	56.8	375	X	X	539895	4287493	X	X
zenotravel11 (3)	2.22	2.4	22.4	14.8	24094	26332	1490640	1126475

in their currently best plan. Once all $|\Phi| + 1$ solutions are verified, the algorithm terminates. To further optimize the algorithm, states that are irrelevant for remaining marginal searches can be pruned, reducing computational effort.

Since all $|\Phi| + 1$ are now solved on a single, albeit distributed, search space, this presents a new possibility of manipulation, having φ_i influencing the solution of $\Pi_{-\varphi_i}$. To avoid this, and to retain property (2) of the partition principle, a state arriving via message from φ_i is automatically considered to have φ_i in its participating agents, and thus never to be considered as a candidate solution for $\Pi_{-\varphi_i}$.

Empirical Evaluation

We performed an empirical evaluation on several MA planning benchmark domains taken from the International Planning Competition . We compare the computational effort required by centralized A* and our two mechanisms (centralized and distributed). We refer to the overhead of computing the centralized mechanism (compared to centralized A*) as the *cost of selfishness*, and to the computational overhead of distributing the mechanism as the *cost of privacy*. We show that empirically, these costs are not high, providing evidence of the feasibility of our methods.

Table 1 depicts the running time and the number of generated states of A* (solving the underlying classical planning problems with full knowledge) and the 3 VCG mechanisms – Centralized (Cen), Selfish-MAD-A* (Dis), and optimized multigoal Selfish-MAD-A* (OptDis). All planners were implemented on top of the Fast-Downward planning system (Helmert 2006) and use the state-of-the-art LM-cut heuristic (Helmert and Domshlak 2009). The number of agents is given in parentheses in the Problem column. We note that the two centralized configurations use the pruning method described by Nissim et al. (2012), which simulates the computational benefits of MAD-A* in centralized search. This is done so MAD-A* wouldn't have an advantage, giving an accurate view of the costs of selfishness and privacy. Running time was limited to 30 minutes and memory to 4GB.

In most cases, the cost of selfishness (overhead incurred by the centralized computation of the VCG payments) is not high – most problems are solved in roughly the same time, with a few exceptions solved 4 and 16 times slower. This is because solving the marginal problems is fairly easy once an optimal solution has been found. However, the cost of privacy is much higher – most problems are solved 4 times slower than centralized VCG, with one problem solved 18 times slower. This is mainly because the privacy of information hurts the heuristic quality, an important factor in the effectiveness of forward search (heuristic) algorithms. However, given that the distributed configurations solve a more difficult problem than the centralized ones, this slowdown is expected and acceptable. Comparing the two distributed approaches, we see that the optimized multigoal version dominates Selfish-MA-A* w.r.t. both runtime and generated nodes. It is clear that elimination of the duplicate effort has a positive effect computation-wise.

Discussion

We described a distributed, strategy-proof, socially efficient, and privacy preserving mechanism for planning by a group of self-interested agents. This result contributes to a small, but growing body of research on this topic. One important advantage of our mechanism is its efficiency. Although there is a clear computational cost to selfishness, we can optimally solve the same order of problems as state-of-the-art optimal planning algorithms. This is in stark contrast to most alternatives: work on stable planning relies on CSP-based algorithms discussed by Nissim, Brafman and Domshlak (2010), that have difficulty handling plans that require more than 3 public actions per agent and do not generate optimal plans, whereas work that seeks equilibria either does not consider the issue of privacy, assumes truthfulness ignoring the issue of manipulation, or requires costly computations.

The VCG mechanism is not without faults. Its main weakness is the the fact that the side payments that the plan beneficiary has to pay the agents can be very substantial in theory. This makes the mechanism suitable in two contexts: 1) Competitive environments where there are a number of different agents with similar capabilities and similar costs, in which case the payments agents receive are not significantly beyond their true cost. Thus, in the example of building construction, there is a large number of providers for every possible task, some more efficient or appropriate (e.g., because their proximity to the site lowers their costs), but all in the same ball park. 2) Plan beneficiary for which the benefit of optimality far outweighs the cost paid. For example, road construction by local government, where the time to complete the project outweighs other considerations. How to address the issue of overcharging is the subject of much current work on related problems (Elkind, Sahai, and Steiglitz 2004; Karlin, Kempe, and Tamir 2005; Singer 2010), although none of the current results are applicable to the planning settings. We hope to examine this question in future work.

Acknowledgments

The authors were partly supported by the Paul Ivanier Center for Robotics Research and Production Management, and the Lynn and William Frankel Center for Computer Science.

References

Ben Larbi, R.; Konieczny, S.; and Marquis, P. 2007. Extending classical planning to the multi-agent case: A game-theoretic approach. In *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, ECSQARU, 731–742.

Bowling, M. H.; Jensen, R. M.; and Veloso, M. M. 2003. A formalization of equilibria for multiagent planning. In *IJCAI*, 1460–1462.

Brafman, R. I., and Domshlak, C. 2008. From one to many: Planning for loosely coupled multi-agent systems. In *ICAPS*, 28–35.

Brafman, R. I.; Domshlak, C.; Engel, Y.; and Tennenholtz, M. 2009. Planning games. In *IJCAI*, 73–78.

Brafman, R. I.; Domshlak, C.; Engel, Y.; and Tennenholtz, M. 2010. Transferable utility planning games. In *AAAI*.

Chandy, K. M., and Lamport, L. 1985. Distributed snapshots: Determining global states of distributed systems. *ACM Trans. Comput. Syst.* 3(1):63–75.

Clarke, E. H. 1971. Multipart pricing of public goods. *Public Choice* 2:19–33.

Crosby, M., and Rovatsos, M. 2011. Heuristic multiagent planning with self-interested agents. In *AAMAS*, 1213–1214.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271.

Elkind, E.; Sahai, A.; and Steiglitz, K. 2004. Frugality in path auctions. In *SODA*, 701–709.

Ephrati, E., and Rosenschein, J. S. 1991. The Clarke Tax as a consensus mechanism among automated agents. In *AAAI*, 173–178.

Feigenbaum, J., and Shenker, S. 2002. Distributed algorithmic mechanism design: recent results and future directions. In *DIAL-M*, 1–13. ACM.

Feigenbaum, J.; Papadimitriou, C. H.; Sami, R.; and Shenker, S. 2002. A BGP-based mechanism for lowest-cost routing. In *PODC*, 173–182. ACM.

Groves, T. 1973. Incentives in Teams. *Econometrica* 41:617–631.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *ICAPS*.

Helmert, M. 2006. The Fast Downward planning system. *J. Artif. Intell. Res. (JAIR)* 26:191–246.

ICAPS. The International Planning Competition. http://www.plg.inf.uc3m.es/ipc2011-deterministic/.

Jonsson, A., and Rovatsos, M. 2011. Scaling up multiagent planning: A best-response approach. In *ICAPS*.

Karlin, A. R.; Kempe, D.; and Tamir, T. 2005. Beyond VCG: Frugality of truthful mechanisms. In *FOCS*, 615–626.

Nisan, N., and Ronen, A. 2001. Algorithmic mechanism design. *Games and Economic Behavior* 35(1-2):166–196.

Nissim, R., and Brafman, R. I. 2012a. Multi-agent A* for parallel and distributed systems. In *AAMAS*, 1265–1266.

Nissim, R., and Brafman, R. I. 2012b. Multi-agent A* for parallel and distributed systems. In ICAPS *Workshop on Heuristics and Search for Domain-Independent Planning*, 43–51.

Nissim, R.; Apsel, U.; and Brafman, R. I. 2012. Tunneling and decomposition-based state reduction for optimal planning. In *ECAI*, 624–629.

Nissim, R.; Brafman, R. I.; and Domshlak, C. 2010. A general, fully distributed multi-agent planning algorithm. In *AAMAS*, 1323–1330.

Parkes, D. C., and Shneidman, J. 2004. Distributed implementations of Vickrey-Clarke-Groves mechanism. In *AAMAS*, 261–268. IEEE Computer Society.

Parkes, D. C.; Kalagnanam, J.; and Eso, M. 2001. Achieving budget-balance with Vickrey-based payment schemes in exchanges. In *IJCAI*, 1161–1168. Morgan Kaufmann.

Petcu, A.; Faltings, B.; and Parkes, D. C. 2008. M-DPOP: Faithful distributed implementation of efficient social choice problems. *J. Artif. Intell. Res. (JAIR)* 32:705–755.

Singer, Y. 2010. Budget feasible mechanisms. In *FOCS*, 765–774.

Torreño, A.; Onaindia, E.; and Sapena, O. 2012. An approach to multi-agent planning with incomplete information. In *ECAI*, volume 242, 762–767. IOS Press.

Vickrey, W. 1961. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance* 8–37.