All for One or One for All: Planning for Cooperative and Selfish Agents (Research Proposal Abstract)

Raz Nissim
Ben-Gurion University of the Negev
Be'er Sheva, Israel
raznis@cs.bgu.ac.il

Introduction

The field of planning deals with automatically producing valid action sequences, which transform a given system from its initial state to some desirable goal state. Planning is a key enabling technology for guidance and control of autonomous systems, and therefore it stands at the core of artificial intelligence research. Planning already has many applications, such as the development of self-driving cars, guidance of NASA's Mars Rovers, and even the optimization of elevators in an office building.

In the classical planning model, all actions have deterministic outcomes and the complete state of the system is known to the planner. Planning is known to be a PSPACE-hard problem, and much work has been done on identifying certain properties and problem structure which can be used to create algorithms that scale beyond current capabilities. The main focus of my work is on planning for systems having multiple agents. For example, an international shipping/logistics company is composed of multiple agents that perform the shipping tasks: pilots/airplanes that ship packages between airports, and the local drivers/trucks that ship packages within a certain locality. Intuitively, we would expect each agent to plan independently, and then to coordinate its plans with other agents, to form a global plan. Following this intuition, in the models I examine, both the planning process and the plan execution is distributed. Here, the agents must plan and coordinate in order to find an executable solution plan. I believe this best represents the practical uses of planning in multi-agent (MA) systems.

There is a long tradition of work on multi-agent planning for cooperative and non-cooperative agent teams involving centralized and distributed algorithms, often using involved models that model uncertainty, resources, and more (Szer, Charpillet, and Zilberstein 2005), and much work on how to coordinate the local plans of agents or to allow agents to plan locally under certain constraints (Cox and Durfee 2005; Steenhuisen et al. 2006; ter Mors, Valk, and Witteveen 2004; ter Mors and Witteveen 2005). However, our starting point is a more basic, and hence, we believe, more fundamental model introduced by Brafman and Domshlak (BD) which offers what is possibly the simplest model of MA planning – MA-STRIPS (Brafman and Domshlak 2008). MA-STRIPS minimally extends standard STRIPS (or PDDL) models by specifying a set of agent ids, and associating each action in the domain with one agent. Thus, essentially, it partitions the set of actions among the set of agents. We believe that this model serves as a skeleton model for most work on cooperative multi-agent systems, and that the insights gained form it can help us address more involved models, too.

BD's algorithm is based on constraint satisfaction techniques. Therefore, it can be transformed into a fully distributed algorithm simply by using a distributed CSP solver. Unfortunately, distributed CSP solvers cannot handle even the smallest instances of MA planning problems. Consequently, a dedicated algorithm, based on the ideas of BD, *Planning-First*, was developed (Nissim, Brafman, and Domshlak 2010). Unfortunately, this algorithm had trouble scaling up to problems in which each agent had to execute more than a small number of actions. (Indeed, BD's algorithm scales exponentially with the minimal number of actions per agent in the solution plan.) Recently, a new, improved algorithm, based on partial-order planning, *MAP-POP*, was developed by (Torreño, Onaindia, and Sapena 2012). Yet, this algorithm, too, leaves a serious gap between what we can solve using a distributed planner and what we can solve using a centralized planner. Moreover, *neither algorithm attempts to generate a cost-optimal plan*.

Main Challenges

Currently, the theoretical understanding of the MA planning problem is lacking. Specifically, much work is needed to understand how the complexity of planning is affected by the structure of a MA system. Such a study, using models that are expressive and general, yet simple to present and to understand, has not yet been conducted. On the practical side, current classical planning algorithms and heuristics do not utilize the special structure exhibited in MA systems. In previous work (Nissim, Brafman, and Domshlak 2010), I showed that specialized techniques for MA planning can outperform existing state-of-the-art technology when planning for MA systems. This work also showed that there is much unrealized potential in the exploitation of MA structure in planning. In general, I believe that the lack of simple and effective algorithms and of efficient frameworks for MA planning hinders both practical and theoretical advances in the field.

Beyond fully-cooperative MA systems, there is much interest in planning for systems where the agents are self-interested. In many realistic settings, agents have personal goals and costs, and are motivated to increase their net benefit. These agents may want to cooperate with one another since they have different capabilities or they find such cooperation beneficial. Planning for such systems entails dealing with different solution concepts (stability instead of cost-optimality) and changing heuristics so that they estimate stability potential of a state rather than its (cost-wise) distance from a goal. In addition, our agents must be able to compute heuristics in a distributed manner, sometimes with partial knowledge of the entire system. Existing planning techniques are incapable of handling such models.

Background

A MA-STRIPS problem for a set of agents $\Phi = \{\varphi_i\}_{i=1}^k$ is given by a 4-tuple $\Pi = \langle P, \{A_i\}_{i=1}^k, I, G\rangle$, where P is a finite set of propositions, $I \subseteq P$ and $G \subseteq P$ encode the initial state and goal, respectively, and for $1 \leq i \leq k$, A_i is the set of actions agent φ_i is capable of performing. Each action $a = \langle \mathrm{pre}(a), \mathrm{eff}(a)\rangle$ is given by its preconditions and effects. A plan is a solution to Π iff it is a solution to the underlying STRIPS problem obtained by ignoring the identities of the agent associated with each action. Since each action is associated with an agent, a plan tells each agent what to do and when. In different planning contexts, one might seeks special types of solutions. For example, in the context of planning games (Brafman et al. 2009), *stable* solutions are sought. We focus on cooperative multi-agent systems, seeking either a (standard) solution or a cost-optimal solution.

The partitioning of the actions to agents yields a distinction between private and public propositions and actions. A *private* proposition 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's actions, while its public actions may require or affect the actions of other agents. For ease of the presentation of our algorithms and their proofs, we assume that all actions that achieve a goal condition are considered *public*. Our methods are easily modified to remove this assumption.

In a distributed system of fully-cooperative agents privacy is not an issue, and so the distinction between private and public actions is not essential, although it can be exploited for computational gains (Brafman and Domshlak 2008). But there are settings in which agents collaborate on a specific task, but prefer not to reveal private information about their local states, their private actions, and their cost. We will refer to algorithms that plan without revealing this information as *privacy preserving* (distributed) planning algorithms. More specifically, in a privacy-preserving algorithm the only information available about an agent to others is its set of public actions, projected onto public atoms. This can be viewed as the interfaces between the agents. Information about an agent's private actions and private aspects of a public action are known to the agent only.

Table 1: Comparison of MA greedy best-first search and MAP-POP. Solution cost, running time (in sec.) and the number of sent messages are shown. *"X"* denotes the problem wasn't solved after one hour.

	#	Solution cost		Runtime		Messages	
problem	agents	MAFS	MAP-POP	MAFS	MAP-POP	MAFS	MAP-POP
Logistics6-0	3	25	25	**0.06**	60.6	**470**	1050
Logistics7-0	4	**36**	37	**0.2**	233.3	**2911**	4898
Logistics8-0	4	31	31	**0.16**	261	**940**	4412
Logistics9-0	4	36	36	**1.02**	193.3	**2970**	3168
Logistics10-0	5	**45**	51	**0.43**	471	**2097**	14738
Logistics11-0	5	**54**	X	**2.7**	X	**14933**	X
Logistics12-0	5	**44**	45	**1.3**	1687	**4230**	28932
Logistics13-0	7	**87**	X	**0.9**	X	**5140**	X
Logistics14-0	7	**68**	X	**0.67**	X	**2971**	X
Logistics15-0	7	**95**	X	**0.74**	X	**6194**	X
rovers5	2	**22**	24	**0.13**	18.7	**84**	323
rovers6	2	**37**	39	**0.07**	18.2	**27**	313
rovers7	3	18	18	**0.07**	44.1	**225**	490
rovers8	4	**26**	27	**0.2**	744	**937**	12102
rovers9	4	38	**36**	**0.82**	222	**380**	4467
rovers10	4	**38**	X	**0.41**	X	**271**	X
rovers11	4	37	**34**	**0.34**	132.5	**299**	2286
rovers12	4	21	**20**	**0.09**	34.4	435	**410**
rovers13	4	**49**	X	**0.15**	X	**472**	X
rovers14	4	**31**	35	**0.42**	443.8	**310**	7295
rovers15	4	46	**44**	**0.33**	164	**252**	2625
rovers16	4	**44**	X	**0.27**	X	**552**	X
rovers17	6	**52**	X	**0.57**	X	**628**	X
satellites7	4	22	22	**0.23**	28.8	**248**	543
satellites8	4	26	26	**0.21**	40.7	**133**	678
satellites9	5	30	**29**	**0.35**	93.3	**397**	1431
satellites10	5	30	**29**	**0.41**	65.9	**355**	942
satellites11	5	31	31	**0.69**	51	**514**	904
satellites12	5	**43**	49	**1.1**	76.9	**390**	1240
satellites14	6	44	**43**	**1.8**	123.4	**721**	1781
satellites15	8	**63**	X	**3.9**	X	**1507**	X
satellites16	10	56	56	**6.6**	481.2	**2279**	4942
satellites17	12	49	49	**6.7**	2681	**2172**	26288

Main Achievements and Results

I now give a short overview of my published and ongoing results.

Multi-Agent Forward Search

We now describe a distributed variant of forward best-first search (MAFS). This algorithm 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 value in its open list. When an agent expands state s, *it uses its own operators only.* This means two agents expanding the same state will generate *different* successor states.

Since no agent expands all relevant search nodes, messages must be sent between agents, informing one agent of open search nodes relevant to it expanded by another agent. Agent φ_i characterizes state s as relevant to agent φ_j if φ_j has a public operator whose public preconditions (the preconditions φ_i is aware of) hold in s. In that case, Agent φ_i will send s to Agent φ_j.

As we will see, this general and simple scheme – apply your own actions/operators only and send relevant generated nodes to other agents – can be used to distribute other search algorithms. However, there are various subtle points pertaining to message sending and termination that influence the correctness and efficiency of the distributed algorithm, as we will see later.

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, and the sending agent's heuristic estimate of s. 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 value for this state, and then can choose between/combine the value it has calculated and the h value in the received message. If both heuristics are known to be admissible, for example, the agent could choose the maximal of the two estimates.

Once an agent expands a solution state s, it sends s to all agents and awaits their confirmation. For simplicity, and in order to avoid deadlock, once an agent either broadcasts or confirms a solution, it is not allowed to create a new solution. If a solution is found by more than one agent, the one with lower cost is chosen, and ties are broken by choosing the solution of the agent having the lower ID. When the solution is confirmed by all agents, 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 and the solution is outputted.

The MAFS Algorithm

Algorithms 1-3 depict the MAFS algorithm for agent φ_i. An empirical comparison of MAFS and current state-of-the-art distributed planner MAP-POP is shown in Table 1.

Algorithm 1 MAFS 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:     $h_{\varphi_i}(s) \leftarrow max(h_{\varphi_i}(s), h_{\varphi_j}(s))$
```

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 $s$ as a solution
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
```

Optimal MAFS

MAFS as presented, is not an optimal planning algorithm. It can, however, be slightly modified in order to achieve optimality. We now describe these modifications, which result in a MA variation of A* we refer to as MA-A*.

As in A*, the state chosen for expansion by each agent must be the one with the lowest $f = g + h$ value in its open list. In MA-A*, therefore, *extract-min* must return this state.

Unlike in A*, expansion of a goal state in MAFS does not necessarily mean an optimal solution has been found. In our case, 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 (which includes all states in all open lists, as well as states in messages that have not been processed, yet).

To detect this situation, we use 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 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 state via a message (known as *pathmax*), this property holds[1].

[1]Although recent work (Holte 2010) shows that pathmax does not necessarily make a *bona-fide* consistent heuristic, pathmax does

We note that for simplicity of the pseudo-code we omitted the detection of a situation where a goal state does not exist. This can be done by determining whether the stable property *"there are no open states in the system"* holds, using the same *snapshot algorithm*.

Applying the knowledge gained by implementing MA-A*, I created the first MA planning framework, MA-FD. The basis for MA-FD is Fast-Downward (FD), which is currently the leading system for centralized planning. I believe my framework will provide researchers with fertile ground for developing new search techniques and heuristics for MA planning. I am hopeful that having a simple and effective MA planning framework will increase interest in MA planning research, as Fast-Downward has done for centralized planning.

Partition-Based Pruning for Optimal Planning

MA-A*'s empirical success was surprising, and led me to examine its theoretical properties in depth. It is now clear that MA-A* is not a shallow parallelization or distribution of A*. Rather, it is structure-aware, using the distinction between local and globally relevant actions and propositions to focus the work of each agent, dividing both the search space and actions among the agents, and exploiting symmetries that arise from the MA structure. It is now clear that some of the ideas that make MA-A* successful, can also be applied to centralized planning. One concrete example of this is a technique for symmetry exploitation in centralized search, which has been published last year (Nissim, Apsel, and Brafman 2012).

This technique requires as input a partition of the set of actions. Given such a partition, it allows us to prune actions roughly as follows: after performing an action a, if this action affects only actions from its own partition (i.e., by supplying or destroying their pre or prevail conditions) then the next action should be from that same partition. The power of this method depends on the quality of the partition used. In some domains natural partitions suggest themselves. For example, when there is a natural notion of agents in a domain (e.g., trucks in Logistics, satellites in Satellite, etc.), then it is natural to partition the set of actions to sets consisting of actions involving a specific agent. However, most domains offer no obvious partition, and hence one of our contributions is a principled and efficient automated domain decomposition method.

As an example, consider a logistics problem consisting of two trucks, and a partition of the actions such that for $i \in \{1, 2\}$, all actions of $truck_i$ are in $\mathcal{A}_i$. When expanding state s, for which the creating action was $a = move(truck_1, loc_1, loc_2)$, all actions corresponding to $truck_2$ are pruned. Since a was performed in order to achieve some precondition for $truck_1$'s public *load/unload* action at loc_2, PB pruning focuses the search effort on applying that action.

Generally, in optimal plans, private actions are executed only to enable some public action, since otherwise, the private action can be removed and the plan is not optimal. Therefore, when pruning the actions of other partitions after applying a private action a, the search effort focuses on achieving the cause for applying a in the first place. The reader may have observed that in optimal search in general, when reaching state s via a private action $a \in \mathcal{A}_i$, all public actions of $\mathcal{A}_i$ constitute a disjunctive action landmark in state s.

Since, in general, planning problems do not exhibit MA structure, in order to use PB pruning we must partition the actions of the problem. As an exponential number of partitions exist, some measure of partition quality is required. Using PB pruning, action pruning is performed only when the first of two consecutive actions is private and the second is one belonging to a different partition. We introduce the notion of **symmetry score** (Γ), which measures the probability of such a sequence appearing in the search:

$$\Gamma(\{\mathcal{A}\}_{i=1}^{k}) = \sum_{i=1}^{k}(pr(a \in \mathcal{A}_i \text{ and } a \text{ is private}) * pr(a \notin \mathcal{A}_i))$$

Γ is, of course, an approximation, since it regards the probability of each action appearing at any point in the search as equal. However, our work shown a high correlation between Γ and the effectiveness of PB pruning. Empirical results which depict the effect partition-based (PB) pruning has on coverage, time, and search space size are shown in Table 2.

Future Challenges

MA-A* showed empirical success in problems exhibiting a natural MA structure. An important question for future work is whether (and how) problems that do not exhibit such structure could be reformulated as MA problems. Such reformulation would enable us to parallelize the search using MA-A*, and solve problems faster. I am currently exploring this problem using graph partition algorithms, and some promising results have already been achieved. In addition, further theoretical analysis of these decompositions has shown that with minimal changes to any centralized algorithm, significant pruning of the search space can be achieved by using methods very similar to MA-A*, without affecting optimality. Investigating and finding more techniques that are used in the MA setting, and can be useful in centralized planning as well, is therefore a key challenge.

Another key challenge is finding a way to deal with incomplete knowledge of the agents. Specifically, how can an accurate heuristic be computed with only partial knowledge of the problem. Solving this issue is crucial for improving performance in the distributed,privacy preserving setting.

Since my main aim is to create planners for real-world MA systems, my methods must be extended beyond the fully cooperative setting. I plan on extending my methodologies for solving fully cooperative MA planning, to more realistic models, involving such selfish, yet willing to cooperate agents. Although these models require different solution concepts and heuristics, I believe that many of the techniques used in MAFS can help formulate an efficient planning algorithm for such systems. In addition, I have ongoing work which aims to use mechanism design tools such as the Vickrey-Clarke-Groves mechanism, in order to "force" util-

ensure that f-values along a path are non-decreasing.

Table 2: Coverage and comparison of A*+$h_{lm\text{-}cut}$ using our 3 pruning rules. Time and node ratios are relative to the baseline planner.

	Coverage				Total Time			Search Time			Expanded			Generated		
Domain	A*	PB	T	PBT	PB	T	PBT	PB	T	PBT	PB	T	PBT	PB	T	PBT
airport	27	27	27	27	0.96	**1.01**	0.97	1.01	1.01	**1.03**	1	1	1	**1.03**	1	**1.03**
blocks	28	28	28	28	1	1	**1.01**	1	1	**1.01**	1	1	1	1	1	1
depot	7	7	7	7	1.03	1	**1.04**	1.04	1	**1.04**	1	1	1	**1.1**	1	**1.1**
driverlog	13	13	13	13	1.2	**1.45**	1.24	1.2	**1.45**	1.25	1.13	1.1	**1.24**	1.49	1.62	**1.68**
freecell	15	15	15	15	0.94	**1.02**	0.94	1.01	**1.02**	1.01	1	1	1	1	1	1
grid	2	2	2	2	0.97	**1.01**	0.97	1.01	**1.02**	1.01	1	1	1	1	1	1
gripper	6	6	6	6	1	1.01	**1.02**	1	1.01	**1.02**	1	1	1	**1.01**	1	**1.01**
logistics00	20	20	20	20	2.15	1	**2.16**	2.15	1	**2.16**	**1.52**	1	**1.52**	**2.88**	1	**2.88**
logistics98	6	6	6	6	**3.74**	1	3.71	**3.78**	1	3.75	**2.02**	1	**2.02**	**4.65**	1	**4.65**
miconic	141	141	141	141	0.82	**1.01**	0.83	0.99	1.02	**1.02**	1	1	1	1	1	1
mprime	**23**	20	22	20	0.74	**0.99**	0.75	1.03	0.99	**1.05**	**1.01**	1	**1.01**	**1.15**	1	**1.15**
mystery	15	15	15	15	0.63	0.98	0.64	1	0.98	**1.01**	1	1	1	**1.01**	1	**1.01**
openstacks	7	7	7	7	1	1	1	1	1	1	1	1	1	1	1	1
pathways-noneg	5	5	5	5	1.65	1.02	**1.7**	1.65	1.02	**1.7**	**1.31**	1	**1.31**	**1.87**	1	**1.87**
pipes-notankage	16	16	16	16	1.04	1.01	**1.05**	1.09	1.01	**1.1**	1	1	1	**1.09**	1	**1.09**
pipes-tankage	9	9	9	9	1	1	**1.01**	1.02	1	**1.04**	1	1	1	**1.04**	1	**1.04**
psr-small	49	49	49	49	1	**1.52**	0.99	1	**1.52**	1	1	**1.48**	1	1.08	**1.51**	1.08
rovers	7	**8**	7	**8**	2.61	1.04	**2.64**	2.63	1.04	**2.66**	**1.78**	1	**1.78**	3.23	1.02	**3.24**
satellite	7	**12**	7	**12**	13.74	1.22	**14.09**	14.55	1.22	**14.91**	**6.85**	1.04	**6.85**	19.18	1.35	**19.33**
tpp	6	6	6	6	1.03	1	**1.04**	**1.04**	1	**1.04**	**1.02**	1	**1.02**	**1.37**	1	**1.37**
transport	11	11	11	11	**1.5**	1	**1.5**	**1.54**	1	**1.54**	**1.24**	1	**1.24**	**1.79**	1	**1.79**
trucks	9	9	9	9	0.96	0.99	0.95	1	0.99	1	1	1	1	**1.01**	1	**1.01**
zenotravel	12	**13**	12	**13**	**2.7**	0.99	2.68	**2.81**	0.99	2.79	**1.54**	1	**1.54**	**2.86**	1	**2.86**
Total/Geometric Mean	441	**445**	440	**445**	1.339	1.047	**1.350**	1.424	1.048	**1.437**	1.226	1.023	**1.231**	1.558	1.055	**1.567**

itarian agents to cooperate and perform distributed optimal planning.

References

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.

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

Cox, J. S., and Durfee, E. H. 2005. An efficient algorithm for multiagent plan coordination. In *AAMAS*, 828–835. ACM.

Holte, R. C. 2010. Common misconceptions concerning heuristic search. In *SOCS*.

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.

Steenhuisen, J. R.; Witteveen, C.; ter Mors, A.; and Valk, J. 2006. Framework and complexity results for coordinating non-cooperative planning agents. In *MATES*, 98–109.

Szer, D.; Charpillet, F.; and Zilberstein, S. 2005. Maa*: A heuristic search algorithm for solving decentralized pomdps. In *UAI*, 576–590.

ter Mors, A., and Witteveen, C. 2005. Coordinating self interested autonomous planning agents. In *BNAIC*, 383–384.

ter Mors, A.; Valk, J.; and Witteveen, C. 2004. Coordinating autonomous planners. In *IC-AI*, 795–.

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