A Distributed Online Multi-Agent Planning System (Dissertation Abstract)

Rafael C. Cardoso
{rafael.caue@acad.pucrs.br}

Supervisor: Rafael H. Bordini
FACIN-PUCRS
Porto Alegre - RS, Brazil

Abstract

The gap between planning and execution is still an open problem that, despite several tries from members of both automated planning and autonomous agents communities, remains without a proper general-purpose solution. We aim to tackle this problem by using a framework for the development of multi-agent systems in both the decentralised multi-agent planning stages, and the execution stages, providing a multi-agent system with capabilities to solve online multi-agent planning problems.

1 Introduction

Multi-Agent Systems (MAS) are often situated in dynamic environments where new plans of actions need to be constantly devised in order to successfully achieve the system goals. Therefore, employing planning techniques during run-time of a MAS can be used to improve agent's plans using knowledge that was not previously available, or even to create new plans to achieve some goal for which there was no known course of action at design time.

Research on automated planning has been largely focused on single-agent planning over the years. Although it is possible to adapt centralised single-agent techniques to work in a decentralised way, such as in (Crosby, Jonsson, and Rovatsos 2014), distributed computation is not the only advantage of using Multi-Agent Planning (MAP). By allowing agents to do their own individual planning the search space is effectively pruned, which can potentially decrease planning time on domains that are naturally distributed. This natural distribution also means that agents get to keep some (or even full) privacy from other agents in the system, as they might have beliefs, goals, and plans that they do not want to share with other agents. Single-agent planning can have no privacy, since the planner needs all the information available.

MAS went through a similar process of transitioning from single to multiple agents, albeit at a faster rate. Recent research, as evidenced in (Boissier et al. 2011; Singh and Chopra 2010), shows that considering other programming dimensions such as environments and organisations as first-class entities along with agents allow developers to create more robust MAS.

Thus, in this dissertation abstract we introduce the design of our Distributed Online Multi-Agent Planning System (DOMAPS). DOMAPS is composed of: i) a formalism for the representation of decentralised domains and problems in multi-agent planning, based on Hierarchical Task Network (MA-HTN); ii) a contract net protocol mechanism for goal allocation; iii) individual planning with the SHOP2 planner; and iv) the use of social laws to coordinate the agents during execution. Some preliminary results from initial experiments in a novel scenario, the floods domain, are shown.

Although approaches to online single-agent planning usually involve some kind of interleaving planning and execution (e.g., lookahead planning), in our initial approach to online multi-agent planning we focus on domains that allow agents some time to plan while the system is still in execution. DOMAPS allows for the dynamic execution of plans found during run-time, making it easy to transition from planning into execution and vice-versa, while still permitting agents to continue their execution, as long as their actions are believed to not cause any conflict with actions from a possible solution.

The remainder of the dissertation abstract is structured as follows. In the next section a discussion on multi-agent planning is presented. Section 3 introduces the initial design of the Distributed Online Multi-Agent Planning System (DOMAPS). Next, in Section 4, we describe the implementation of DOMAPS in a MAS development framework. In Section 5, we describe the floods domain, a novel domain designed for heterogeneous multi-agent systems. Some initial experiments using DOMAPS in this domain are also shown. We conclude with a discussion on related work and some concluding remarks.

2 Multi-Agent Planning

Multi-Agent Planning (MAP) has often been interpreted as two different things. Either the planning process is centralised and produces distributed plans that can be acted upon by multiple agents, or the planning process itself is multi-agent. Recently, the planning community has been favouring the concept that MAP is actually both of these things, that is, the planning process is done **by** multiple agents, and the solution is **for** multiple agents.

When considering multiple agents, the planning process gets increasingly more complicated, giving rise to several problems (Durfee and Zilberstein 2013). Actions that agents choose to make may cause an impact in future actions that the other agents could take. Likewise, if an agent knows which actions the other agents plan to take, it could change its own current choices. When dealing with multiple agents, concurrent actions are also a possibility that may require additional care.

In Table 1 we characterise some differences between single-agent planning and multi-agent planning. Although computation can be distributed in single-agent planning it is not commonplace, since the cases where it is actually useful are too few we omitted it from the table. And while multi-agent planning could have no privacy, even in fully cooperative domains it is fairly trivial to allow at least some sort of partial privacy. Full privacy on the other hand is quite difficult because of the coordination needs in multi-agent planning. Single-agent planning has no privacy, since the planner needs all the information available. Agents in single-agent problem formalisms are usually represented as any other object or fact of the environment. Agents in multi-agent planning are treated as first-class entities, where each agent can have its own domain and problem specification.

Table 1: Comparisons between single-agent planning (SAP) and multi-agent planning (MAP).

	computation	**privacy**	**agent abstraction**
SAP	centralised	none	objects
MAP	decentralised	partial or full	first-class entities

Durfee, in (Durfee 1999), establishes some stages of multi-agent planning, that were further extended in (Weerdt, Mors, and Witteveen 2005) and (de Weerdt and Clement 2009):

1. **Global goal refinement:** decomposition of the global goal into subgoals.
2. **Task allocation:** use of task-sharing protocols to allocate tasks (goals).
3. **Coordination before planning:** coordination mechanisms that prevent conflicts during the individual planning stage.
4. **Individual planning:** planning algorithms that search for solutions.
5. **Coordination after planning:** coordination mechanisms that correct conflicts during the individual planning stage.
6. **Plan execution:** the agents that participated in the planning process now carry out the plans.

Not all of these stages are necessary in MAP, some may even be combined into one.

3 The Distributed Online Multi-Agent Planning System

Our multi-agent planning system consists of several main components: **planning formalism** – formally describes the information from the planning domain and problem that will be used during planning; **goal allocation** – set of techniques used to allocate goals to agents; **individual planning** – the planner used during each agent's individual planning stage; and **coordination mechanism** – used before or after planning to avoid possible conflicts that can be generated during planning. DOMAPS was made to work as a general-purpose domain-independent system, and as such we expect to turn it into an open platform where many other alternatives for main components can be added, allowing the MAS designer to pick and choose the ones that work better for their particular distributed online multi-agent planning problem.

Currently, DOMAPS can be used in three different situations where agents have access to the following commands:

- **plan:** plan for a set of organisational goals in which there are no know plans.
- **replan:** plan for a specific organisational goal, either because the known plan failed, or because the agent detected a change in the environment that could potentially lead to a better solution.
- **replanall:** drop all current organisational goals and their related intentions, and start a new planning process for the organisational goals that were dropped, using up-to-date information about the environment. Useful in case everything seems to be going wrong, though this construct is slightly more difficult to automate than the other two.

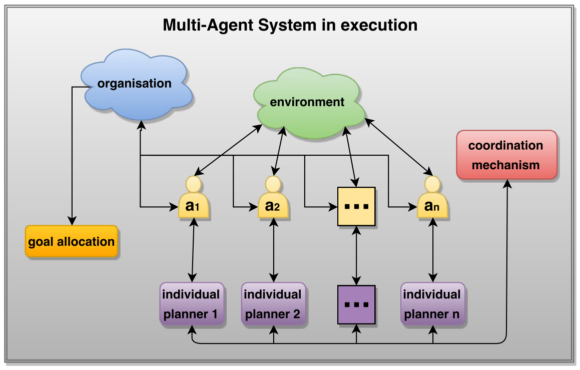

Figure 1: DOMAPS design overview.

The design overview of DOMAPS is shown in Figure 1. Multiple agents $(a_1, a_2, ..., a_n)$ interact with an environment to obtain information and carry out their actions. These agents are part of an organisation, adopting roles and following norms and receiving missions that are related to their roles, all the while pursuing the organisation's goals.

The planning process in DOMAPS consists of the following: a mechanism is used to separate and allocate goals to agents; up-to-date information is collected and translated into a planning formalism that the planner can understand; agents start their individual planner in the search for a solution to the set of allocated goals; agents coordinate with each other either before or after the planning process, in order to prevent the generation of any conflicts or help solve any dependencies; and finally, each agent translates the solution found by their respective planners into plans that can be added to their plan library.

3.1 Planning Formalism

We propose the Multi-Agent Hierarchical Task Network (MA-HTN) formalism, which is an extension of the centralised single-agent HTN formalism used in the SHOP2 planner (Nau et al. 2003). MA-HTN is intended for **online** multi-agent planning problems, since domain and problem information have to be collected during execution. Agents use a **translator** to parse their information about the world into domain and problem specifications that is then passed to their own individual planner.

Each agent has their own problem and domain specification. This provides a decent level of privacy on its own, since each planner only has access to their respective agent problem and domain specifications. This means that, unlike some of the other multi-agent planning formalisms, MA-HTN does not need to have privacy or public blocks. Although at some point it might be interesting to add the capability to include private goals into the planning consideration, for now we are interested only on organisational goals.

Actions from other agents can cause conflicts, either at the moment that action is executed (e.g., concurrent actions) or in the future (e.g., durative actions). Actions that can cause **conflict** have to be annotated by the MAS developer, in order for the translator to identify them. Likewise, dependencies between actions can also exist, either as a concurrent action that requires another agent or as actions that depend on the actions of other agents to happen first. These **dependency** relations also have to be annotated by the MAS developer, so that the translator can add them to the specification.

3.2 Goal Allocation

A Contract Net Protocol (CNP) mechanism is used to allocate goals to agents in DOMAPS. Our CNP mechanism is based on the original CNP design of Reid G. Smith (Smith 1980), with a few modifications in order to accommodate our needs for a goal allocation mechanism in the context of MAP. The initiator in our case will always be the organisation. It is the organisation's role to start new auctions for organisational goals that do not have any known plans on how to achieve them, or for organisational goals that have plans, but needs to be re-planned. The bidders, then, are the agents that are part of the organisation and participate during planning.

The logic of the bid depends on the rest of the mechanisms being used in DOMAPS and in the MAS development platform, but it is fair to assume that agents have the ability of checking their plan library for plans that are able to decompose, at least at some level, the goal that is being auctioned. Although domain-dependent procedures for determining the bid will provide better results, we provide a simple domain-independent general-purpose procedure that agents can use to determine their bid, shown in Algorithm 1.

The agent checks if the announcement of the goal came from the organisation and if he is eligible according to the eligibility criteria provided in the announcement, or otherwise decides not to bid. If the agent chooses to proceed with the bid, then, he keeps decomposing the goal into subtasks and incrementing the bid by 1 for each level that was successfully decomposed, either until it is close to the deadline, or it arrived in an action that could achieve the goal, or it found a dead end (in which case the bet is null).

Algorithm 1 Domain-independent algorithm for determining an agent's bid.

```
procedure bid (from, goal-name, goal-spec, eligibility, deadline)
if from $\neq$ organisation then return failure
else if I am not eligible then return failure
else
    while ((close to deadline) or (no more levels available to decompose)) do
        decompose one level of one task from goal-spec
        bid-value = bid-value++
    end while
return bid-value
end if
```

By the end of the loop the bidder agent will have the value of the bid to be sent to the initiator. The initiator allocates the goal to the agent with the lowest (not null) bid. We assume here that every goal will eventually be allocated, meaning that there is at least one agent eligible for each organisational goal.

3.3 Individual Planner

SHOP2 (Nau et al. 2003) is an HTN planner with support for the sort of anytime planning that DOMAPS requires. No modifications were made to the actual planning algorithm and search techniques of SHOP2, as the multi-agent mechanisms present in the other components proved to be enough for our initial experiments. As long as we can keep the individual planners intact, DOMAPS benefits from its multi-layered approach, making it easier to change components as we see fit, with little to no modification required in the planners.

Many parameters can be used to tweak the SHOP2 planner. Perhaps the most relevant to DOMAPS is the parameter that guides which kind of search that will be made, of which the possible values are:

- **first:** depth-first search that stops at the first plan found.
- **shallowest:** depth-first search for the shallowest plan, or the first such plan if there are more than one.
- **id-first:** iterative-deepening search that stops at the first plan found.

3.4 Coordination Mechanism

Social laws can coordinate agents by placing restrictions on the activities of the agents within the system. The purpose of these restrictions is twofold: it can be used to prevent some destructive interaction from taking place; or it can be used to facilitate some constructive interaction.

The design of social laws is domain-dependent, and we require them to be supplied by a designer offline. Thus, while the social laws are provided before planning, we do not directly use them during individual planning. Instead, we take advantage of the capabilities provided by the

MAS development framework, that we used to implement DOMAPS, in order to apply social laws in coordination after planning.

In the original model of Shoham and Tennenholtz (Shoham and Tennenholtz 1995), social laws were used to restrict the activities of agents so as to ensure that all individual agents are able to accomplish their personal goals. We follow a similar idea, although agents here aim to achieve organisational goals, and thus, are naturally compelled to follow the social laws that are present in the system.

We formally define social laws in our model as:

Definition 1 Given a set of agents Ag, a set of actions Ac, a set of states S, a set of preconditions P, and a set of options Θ, a *social law* is a tuple (ag,ac,s,P,Θ) where $ag \in Ag$, $ac \in Ac$, and $s \in S$.

A social law sl constrains a specific action ac of agent ag, considered to be a possible point of conflict (as established in the operator description, as shown in the MA-HTN formalism), when the state s satisfies each precondition $\rho_i \in P$, giving the agent all possible options $\theta_i \in \Theta$. Although not explicitly present in this model, the *null* action (e.g., do nothing) can be a possible option, but in order for it to be viable it needs to have been established as an action (operator) in the MAS.

4 Multi-Agent System Integration

DOMAPS is an online system, and, as mentioned before, that implicates the use of planning techniques whilst the MAS is running. Therefore, we need a MAS development platform in order to properly implement and evaluate DOMAPS. We chose to use the **JaCaMo**[1] (Boissier et al. 2011) framework as the MAS development platform, since it contains all of the programming abstractions that DOMAPS requires – **organisation**, **environment**, and **agent** abstractions.

JaCaMo combines three separate technologies into a framework for MAS programming that makes use of multiple levels of abstractions, enabling the development of robust MAS. Each technology (Jason, CArtAgO, and Moise) was developed separately for a number of years and are fairly established on their own when dealing with their respective abstraction level (agent, environment, and organisation).

To illustrate the run-time of DOMAPS when the `domaps.plan` internal action is executed, consider the overview provided in Figure 2. When an agent executes `domaps.plan`, it goes through phase 1 and activates the contract net protocol to allocate the organisational goals between the agents. Then, in phase 2, each agent knowledge about the world is passed to a MA-HTN translator, that sends the information needed to SHOP2 for the individual planning that takes place in phase 3. The solution found by each agent's planner goes back through the MA-HTN translator again, translating the solution into AgentSpeak plans. Finally, the solution is carried out by the agents in accordance to the social laws (phase 5) that are associated with actions from the solution that can cause conflicts.

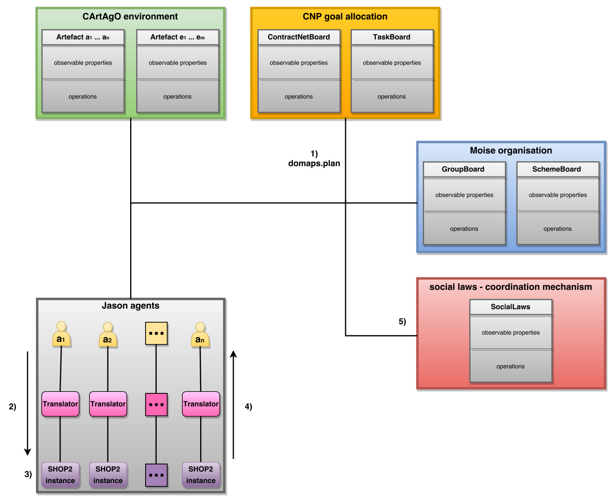

Figure 2: DOMAPS run-time overview of the `domaps.plan` internal action.

5 The Floods Domain

The lack of robust and complex multi-agent domains led us to design a new domain, in order to best exploit the advantages of MAP and MAS. The inspiration for this specific domain came from a real-world scenario, taken from another project that we currently participate. It is a multidisciplinary and inter-institutional project that focuses on using information technology (e.g., a team of autonomous multi-robots) to help mitigate and prevent natural disasters. This scenario is specifically targeted at flood disasters, often caused by intense hydro-meteorological hazards that can lead to severe economic losses, and in some extreme cases even deaths.

Our domain, the Floods domain, is based on that real-world scenario. In the floods domain, a team of autonomous and heterogeneous robots are dispatched to monitor flood activity in a region with multiple areas that are passive of floods. All of the goals come from the Centre for Disaster Management (CDM) that is located in the region being monitored. The CDM is usually operated by humans, but in our JaCaMo+DOMAPS implementation we simulate them by using agents, capable of creating dynamic goals during run-time.

In Figure 3, we show the elements that compose the Floods domain. The domain takes place in a particular region, which is divided into several interconnected *areas*. Movement through the region occurs from traversing these areas. *Flood* events are common in the region, especially during heavy-rain. These floods can be observed from specific areas in the region. The areas can be connected by a *water path*, that can be traversed by naval units, and/or by a *ground path*, that can be traversed by ground units. *Water*

[1] `http://jacamo.sourceforge.net/`.

sample can be requested to be collected from certain areas. During flood events, *victims* may be detected and in need of assistance. The *CDM* establishes a base of operations in one of the areas in the region.

Finally, the naval units are composed of *USVs* that can move through areas connected by water paths, collect water samples, and take pictures of flood events. Meanwhile, the *UGVs* are ground units that are able to move through areas connected by ground paths, take pictures of flood events, and provide assistance to victims by transporting first-aid kits to first responders close by. The robots can only perceive other robots that are in the same area.

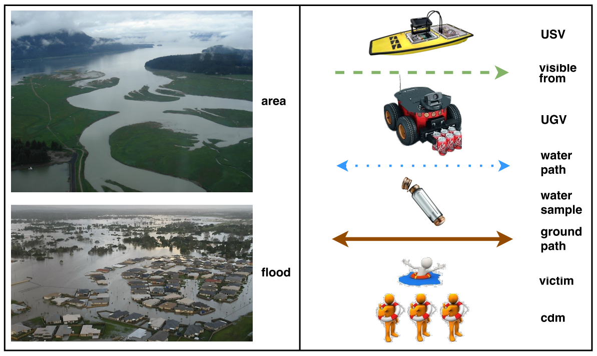

Figure 3: Elements from the Floods domain.

5.1 Experiments

For these initial experiments, we maintained the number of agents and focused on increasing the number of goals. It seems that there is a relation between the number of goals and the number of agents. For most domains, having the number of goals equal to the number of agents, and assuming that each agent is capable of solving its associated goal, appears to result in faster planning times. As the number of goals surpasses the number of agents, the planning time approximates to that of single-agent SHOP2. In Table 2 we show the some initial experiments on this domain for small problems with 4, 8, 16, and 32 goals.

The results are shown in regards to time spent planning, and the number of state expansions and inferences that were made during planning. These results do not depict any of the run-time features of DOMAPS, as we are still investigating how to evaluate it as a whole, and considering what evaluation parameters that could be used both for planning and for execution.

It is clear that our approach would be faster than regular SHOP2, since we are assigning goals to agents previously, while SHOP2 needs to expand states during planning in order to try different assignments. The real advantage that these initial experiments show relate to the number of expansions and inferences, showing DOMAPS does much fewer, even if adding all the agents, than SHOP2. The individual planning approach taken in DOMAPS can discard many of the predicates that are usually used to assign tasks between different objects, remember that agents in SHOP2 are no different than any other object from the planning formalism. By

Table 2: Initial experiment results.

	DOMAPS			**SHOP2**
	usv1	usv2	ugv1	
floods 4 pl. time	0.001	0.001	0.001	0.004
exp.	8	8	15	65
inf.	13	13	21	186
floods 8 pl. time	0.001	0.001	0.002	0.011
exp.	15	15	29	129
inf.	21	21	37	360
floods 16 pl. time	0.002	0.002	0.004	0.033
exp.	29	29	57	257
inf.	37	37	69	708
floods 32 pl. time	0.003	0.003	0.005	0.095
exp.	57	57	113	513
inf.	69	69	133	1404

using agents as first-class abstractions during planning we are free of the use of these predicates. These results should also be scalable, which we aim to prove in future experiments. Experiments for increasing the number of agents, and also for increasing the number of predicates, are already underway.

6 Related Work

There has been several surveys over the years describing advancements in particular areas of planning. Of interest and related to this research there are, for example: in (desJardins et al. 1999), a survey on distributed online (continual) planning is presented, with the state of the art in distributed and online planning at the time (1999), and a design for a distributed online planning paradigm; a survey (Meneguzzi and De Silva 2013) that presents a collection of recent techniques (2013) used to integrate single-agent planning in BDI-based agent-oriented programming languages, focusing mostly on efforts to generate new plans at run-time; and two multi-agent planning surveys, in 2005 (Weerdt, Mors, and Witteveen 2005) and 2009 (de Weerdt and Clement 2009), describing several approaches taken towards multi-agent planning over the last few years.

In (Nissim and Brafman 2014), the authors propose a heuristic forward search for classical multi-agent planning that respects the natural distributed structure of the system, preserving agent privacy. According to their experiments, their system showed the best performance in regards to planning time and communication, as well as the quality of the solution in most cases, when compared to other offline multi-agent planning systems.

FLAP (Sapena, Onaindia, and Torreño 2015) is a hybrid planner that combines partial-order plans with forward search and uses state-based heuristics. FLAP implements a parallel search technique that diversifies the search. Unlike the other planners, FLAP exploits delaying commitment to the order in which actions are applicable. This is done to achieve flexibility, reducing the need of backtracking and minimizing the length of the plans by promoting the parallel execution of actions. These changes come at an increase in computational cost, though it allows FLAP to solve more problems than other partial-order planners.

In (Clement, Durfee, and Barrett 2007), multi-agent planning algorithms and heuristics are proposed to exploit summary information during the coordination stage in order to speed up planning. The authors claim that by associating summary information with plans' abstract operators it can ensure plan correctness, even in multi-agent planning, while still gaining efficiency and not leading to incorrect plans. The key idea is to annotate each abstract operator with summary information about all of its potential needs and effects. This process often resulted in an exponential reduction in planning time compared to a flat representation. Their approach depends on some specific conditions and assumptions, and therefore cannot be used in all domains, i.e., it is not a general-purpose system.

Kovacs proposed a recent extension for PDDL3.1 that enables the description of multi-agent planning problems (Kovacs 2012). It copes with many of the already discussed open problems in multi-agent planning, such as the exponential increase of the number of actions, but it also approaches new problems such as the constructive and destructive synergies of concurrent actions. Although only the formalism is provided (it is not yet implemented in any system), the ideas expressed by Kovacs are enticing, making it an interesting candidate to add to DOMAPS planning formalisms.

7 Conclusion

In this dissertation abstract we described the design of a Distributed Online Multi-Agent Planning System (DOMAPS). Specifying each of its main components: *i)* the planning formalism – we introduced the MA-HTN formalism, a multi-agent variation of the traditional single-agent HTN formalism; *ii)* the goal allocation mechanism – by using a contract net protocol, the agents that participate in the planning stage can pre-select the goals that they believe to be more appropriate to them, this pre-planning can cut the planning time considerably in domains with very heterogeneous agents; *iii)* the individual planner – the SHOP2 planner is used in each agent for individual planning, so as to make the most of the HTN-like structure of the plan library in Jason agents; *iv)* the coordination mechanism – employment of social laws to coordinate the agents during run-time in order to avoid possible conflicts made during planning.

Initial experiments and experience with DOMAPS has presented enough positive incentives to pursue solutions for the limitations and to provide improvements for the system overall.

References

Boissier, O.; Bordini, R. H.; Hübner, J. F.; Ricci, A.; and Santi, A. 2011. Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*.

Clement, B. J.; Durfee, E. H.; and Barrett, A. C. 2007. Abstract reasoning for planning and coordination. *Journal of Artificial Intelligence Research (JAIR)* 28:453–515.

Crosby, M.; Jonsson, A.; and Rovatsos, M. 2014. A single-agent approach to multiagent planning. In *21st European Conf. on Artificial Intelligence (ECAI'14)*.

de Weerdt, M., and Clement, B. 2009. Introduction to Planning in Multiagent Systems. *Multiagent Grid Syst.* 5(4):345–355.

desJardins, M. E.; Durfee, E. H.; Ortiz, C. L.; and Wolverton, M. J. 1999. A survey of research in distributed, continual planning. *AI Magazine* 20(4).

Durfee, E. H., and Zilberstein, S. 2013. Multiagent planning, control, and execution. In Weiss, G., ed., *Multiagent Systems 2nd Edition*. MIT Press. chapter 11, 485–545.

Durfee, E. H. 1999. Distributed problem solving and planning. In *Mutliagent systems*. MIT Press. 121–164.

Kovacs, D. L. 2012. A multi-agent extension of pddl3.1. In *Proceedings of the 3rd Workshop on the International Planning Competition (IPC)*, ICAPS-2012, 19–27.

Meneguzzi, F., and De Silva, L. 2013. Planning in BDI agents: a survey of the integration of planning algorithms and agent reasoning. *The Knowledge Engineering Review* FirstView:1–44.

Nau, D.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. Shop2: An htn planning system. *Journal of Artificial Intelligence Research* 20:379–404.

Nissim, R., and Brafman, R. I. 2014. Distributed heuristic forward search for multi-agent planning. *J. Artif. Intell. Res. (JAIR)* 51:293–332.

Sapena, O.; Onaindia, E.; and Torreño, A. 2015. FLAP: applying least-commitment in forward-chaining planning. *AI Commun.* 28(1):5–20.

Shoham, Y., and Tennenholtz, M. 1995. On social laws for artificial agent societies: Off-line design. *Artif. Intell.* 73(1-2):231–252.

Singh, M., and Chopra, A. 2010. Programming multiagent systems without programming agents. In Braubach, L.; Briot, J.-P.; and Thangarajah, J., eds., *Programming Multi-Agent Systems*, volume 5919 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 1–14.

Smith, R. G. 1980. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput.* 29(12):1104–1113.

Weerdt, M. D.; Mors, A. T.; and Witteveen, C. 2005. Multiagent planning: An introduction to planning and coordination. Technical report, Handouts of the European Agent Summer.