Allocating Social Goals Using the Contract Net Protocol in Online Multi-Agent Planning

Rafael C. Cardoso and Rafael H. Bordini
School of Informatics (FACIN-PPGCC)
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Porto Alegre, RS, Brazil
Email: rafael.caue@acad.pucrs.br, rafael.bordini@pucrs.br

***Abstract*—Centralised planning systems generally assign goals to agents during the search for a solution to a planning problem. In a distributed multi-agent setting, this would constrain the autonomy of the agents, and violate their privacy. Thus, by using task allocation protocols, the agents themselves can compete to decide who will take each goal, then later plan individually provided coordination mechanisms are in place, giving a higher degree of autonomy and privacy during the planning process. In this paper, we propose the use of a contract net protocol mechanism to allocate social goals in multi-agent planning. Our contributions are an algorithm for determining a bid, and several bid evaluation criteria. We also define some heuristics that can be used as bids. In our algorithm, agents expand social goals using their plan library in order to collect information and formulate their bid. We also show the results of experiments in a multi-agent planning domain in order to evaluate our heuristics.**

I. Introduction

Multi-Agent Systems (MAS) are often situated in dynamic environments where new plans of action may need to be devised in order to successfully achieve the system's 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 effective course of action at design time.

Organisations in a MAS are complex entities in which agents interact in order to achieve some global purpose [1]. They provide a scope for the interactions, reduce or manage uncertainty, and coordinate agents to improve efficiency and avoid conflicts. Social goals are organisational objectives that are originated in the MAS organisation. These goals are not necessarily shared by individual agents. Organisations and social goals are especially relevant to MAS in complex, dynamic, and distributed domains. These types of domains are very similar to those that are often used in the area of Multi-Agent Planning (MAP).

In order to successfully and efficiently achieve social goals, they have to be allocated to the most appropriate agents. If the best agent to achieve the goal is not known beforehand, then it can be done, for example, by using task allocation protocols. Task allocation has seen a wide array of uses in MAS, but in this paper we use it in the context of MAP. Because we are dealing with online planning (i.e., combining planning and plan execution), new social goals can emerge (or their conditions can change) at run time.

In this paper, we use a Contract Net Protocol (CNP) mechanism to allocate social goals to agents. The main idea is to find the agent that can expand the social goal most effectively, out of all eligible agents. Agents expand social goals using their plan library in order to collect information and formulate their bid. We provide some heuristics that agents can use to determine their bid for a social goal, and also show how these heuristics can be evaluated.

When all agents taking part in the goal allocation have identical plan libraries, and they all have the same computing power, then the allocation is reduced to just assigning the goals randomly to one of these agents [2]. However, when this is not the case, our approach can lead to reasonable task allocations, and more importantly, it provides various heuristics that can be experimented with to find out the most suitable one for a particular application.

Our approach is most useful when using libraries of pre-compiled plans that are composed of: *goal*, the postcondition of the plan; *context*, the precondition of the plan; and *body*, the course of action to be executed (including actions and subgoals). Although our goal allocation algorithm ignores the context of the plan, this information can still be used by subsequent MAP mechanisms (e.g., planner, coordination). The algorithm can be adapted with minimal effort to work with other similar types of plan libraries, and it is also worth noting that many BDI-based agent programming languages use that plan structure, notably AgentSpeak [3] and its variants [4].

The remainder of this paper is structured as follows. In the next section we contextualise our approach by providing some background on MAP and CNP. Section III shows our main contribution on how to use a CNP mechanism to allocate social goals. We introduce a domain-independent algorithm for calculating CNP bids based on expanding social goals using plan libraries, discuss about the relation between agents' plans, and also provide some bid evaluation criteria. Next, in Section IV, we describe our experiments with a MAP domain, and discuss the results obtained. Afterwards, we discuss related work and end the paper with some conclusions and a brief description of future work.

II. Context

Over the years, MAP has been interpreted as two different things. Either the planning process is centralised and produces

978-1-5090-3566-3/16 $31.00 © 2016 IEEE
DOI 10.1109/BRACIS.2016.35

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.

Six phases of MAP are described in [5]: *global goal refinement*, decomposition of the global goal into subgoals; *task allocation*, use of task-sharing protocols to allocate tasks (goals and subgoals); *coordination before planning*, coordination mechanisms that avoid conflicts before planning; *individual planning*, planning algorithms that search solutions for the problem; *coordination after planning*, coordination mechanisms that solve conflicts after planning; and *plan execution*, the agents carry out the solution found.

Task allocation is one of first phases required in MAP. In this phase, the agents gain access to the goals that will be planned for. A mechanism (e.g., negotiation-based or task-sharing protocols) is used to separate and allocate goals to agents that have the best (estimated) chance of finding a potential solution.

According to Smith's original concept from his work in 1980 [6], the contract net protocol "has been developed to specify problem-solving communication and control for nodes in a distributed problem solver". In this protocol, nodes enter a negotiation process over contracts, where the winner is awarded a contract to execute the task. In our case, the tasks in questions are the social (i.e., organisational) goals.

Nodes (i.e., agents) in the CNP can assume one of two possible roles: initiator (manager) or bidder (contractor). The initiator is responsible for announcing new tasks in the CNP (i.e., creating new contracts), monitoring their activity, awarding tasks to winners, and processing the results of the task's execution. The bidders decide which contracts to take part in, determine their bids, and execute the tasks that they are awarded.

III. Using CNP for Goal Allocation

Our CNP mechanism is based on the original protocol [6], 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 is the MAS organisation. It is the organisation's role to start new contracts for social goals that have not been previously assigned to any agent. The bidders are agents taking part in the MAS organisation that also participate in the planning process.

In Listing 1, we show an overview of the specification for a social goal announcement. The *to* field allows the announcement to be sent either to all agents in the organisation (broadcast), or to a specific group of agents within the organisation. Each announcement is identified through a unique *id*. The *goal* contains both the name of the goal and the goal specification, such as preconditions for example. The *eligibility* field can be used for restricting goals to certain roles, and/or to agents that have specific plans in their plan library. Finally, the *deadline* limits the time that the initiator accepts the bids for that specific goal.

Listing 1. Goal announcement specification.

```
1 to          ::= * or group_id
2 from        ::= organisation
3 id          ::= announcement_id
4 goal        ::= goal-name, goal-spec
5 eligibility ::= role and/or plans
6 deadline    ::= timer
```

After all social goals have been allocated, that is, each social goal has a contractor agent in charge of achieving it, then the next phase of MAP can start. We assume here that every goal will eventually be allocated, meaning that there is at least one eligible agent for each social goal. The goal allocation does not guarantee that the agent will be able to find a solution (this depends on the results of the planning phase) and execute it successfully (which depends on the execution stage). Instead, goals are allocated to agents that have shown a better chance of doing so according to the chosen heuristics.

A. Social Goal Expansion

During the expansion of a social goal, our algorithm ignores the preconditions of plans, that is, we do not apply an action theory. This would involve performing lookahead, which is essentially planning, and would also have a high computing cost. Instead, we opted to go for a quick expansion of the plan library, selecting all plans that are related to the social goal, while ignoring the context of these plans. This relaxation allows for quick computation of potentially useful heuristics for the goal allocation phase.

AgentSpeak-like plan libraries often contain several recursive plans, which leads to infinite expansion of goal-plan trees [7]. Although we considered limiting or ignoring these recursive plans, we found out from our experiments that this can be a valuable information to have, and it causes no problem for our algorithm since it has a particular deadline within which to terminate. Agents with recursive plans can get stuck in a loop during execution, either because actions on real-world applications are non-deterministic, or because the MAS programmer made a mistake.

Contract nets have a deadline in order to prevent initiators to wait indefinitely for bids. In our approach, this deadline is set offline by the MAS designer for all possible types of social goals. We use it to stop the infinite expansion that can happen when agents get stuck in a loop, or when agents have large plan libraries. That is, agents expand the social goal up until they are close to that deadline, at which point they stop the expansion and use measurements of the expanded tree to form their bid value.

In Algorithm 1, we perform a breadth-first expansion of a social goal using the agent's plan library. Our algorithm collects data during expansion, which can then be used by the agent to place his bid.

Algorithm 1 Breadth-first expansion of a social goal.

```
1: function expand(goal, deadline)
2: Plans ← relevant_plans(goal)
3: if Plans = ∅ then
4:     return "not eligible"
5: end if
6: n_plans, n_actions, m_depth ← 0
7: m_width ← |Plans|
8: Subplans ← ∅
9: while there exists plan ∈ Plans do
10:    if time() ≥ deadline then
11:        return (n_plans, n_actions, m_depth, m_width)
12:    end if
13:    n_plans ← n_plans + 1
14:    n_actions ← n_actions + count_actions(plan)
15:    Goals ← goals(plan)
16:    Subplans ← Subplans ∪ relevant_plans(Goals)
17:    Plans ← Plans \ {plan}
18:    if Plans = ∅ then
19:        if |Subplans| > m_width then
20:            m_width ← |Subplans|
21:        end if
22:        m_depth ← m_depth + 1
23:        Plans ← Subplans
24:        Subplans ← ∅
25:    end if
26: end while
27: return (n_plans, n_actions, m_depth, m_width)
```

The expansion starts with the social goal as the root of the tree. The agent uses a $relevant\_plans$ function that returns all plans in the agent's plan library that can be used to achieve that particular goal. If the set of such $Plans$ is initially empty, then the agent is not eligible to bid for this contract. Otherwise, the information used for determining the agent's bid is initialised: n_plans is the total number of plans that were expanded; $n\_actions$ is the total number of actions found in all of the plans that were expanded; m_depth is the maximum depth of the tree; and m_width is the maximum width of the tree, which initially receives the cardinality of the $Plans$ set, indicating the initial width of the tree. The $Subplans$ set is initially empty.

Parameter $deadline$ is the maximum time after the start of expansion that the algorithm can run for. It is expected that when calling the $expand$ function, the deadline given for the algorithm is sufficiently before the CNP deadline so that the agent has time to communicate its bid to the organisation manager (the protocol initiator).

At the start of a plan's expansion, the agent uses a function $time()$ to get the time that has passed since the start of the $expand$ function, and checks if that value is greater than or equal to $deadline$. If it is, then the algorithm stops the expansion and returns its measurements that will be used to formulate a bid.

While there remains any $plan$ in the $Plans$ set, we increase the counter of total plans expanded, add the number of actions found in the body of the plan to the total number of actions, and assign all the subgoals found in the body of the $plan$ to the $Goals$ set. Then, the agent uses again the $relevant\_plans$ function to get all relevant plans but now for each of the subgoals in that set.

The $plan$ that was expanded is removed from the $Plans$ set. If this was the last plan and the $Plans$ set is now empty, then the agent checks if the cardinality of the $Subplans$ set is higher than our current m_width, in which case the cardinality of the $Subplans$ set becomes the current maximum width of the tree. After that, we increase the maximum depth counter, and move the $Subplans$ set to the $Plans$ set (this effectively give us breadth-first search but doing it in this particular way allows us to make all the measurements we need).

When both sets are empty, or the deadline is past, the algorithm returns a multi-valued bid, a 4-tuple with relevant information collected during the expansion of the goal-plan tree for the given social goal.

B. Plan Trees

We opted for not saving the whole goal-plan tree. Instead, we update the measurements that will be part of the bid and we currently use as bid evaluation criteria, and save only the plans that need to be immediately expanded. This makes the whole expansion process faster and use less memory, which is good enough for the heuristics we need at this initial stage of allocating goals in MAP. The tree being generated is similar to goal-plan trees, which are tree structure of goals whose children are the plans that achieve it, and the children of a plan are subgoals. For the purpose of comparing goal-plan trees between agents as used in our approach, we can omit goals and subgoals, since we currently do not count the number of goals for bidding purposes.

When comparing plan trees between agents, we can classify them according to their topology into four different types: *recursive distinct plan tree*, when a recursive plan was used and plan trees are sufficiently distinct from each other; *non-recursive distinct plan tree*, when no recursive plan was used and plan trees are sufficiently distinct from each other; *recursive similar plan tree*, when a recursive plan was used and plan trees are similar or identical to each other; and *non-recursive similar plan tree*, when no recursive plan was used and plan trees are similar or identical to each other.

In Figure 1, we can observe all four comparisons between plan trees. Figure 1(a) shows the recursive distinct plan tree comparison, where plan trees contain recursive plans and the trees are different from each other, which is evident by comparing the maximum width of both trees: 3 for *agent1* and 1 for *agent2*. Figure 1(b) shows the non-recursive distinct plan tree, where plan trees do not contain any recursive plans and the threes are once again different from each other. Figure 1(c) shows the recursive similar plan tree, where plan trees contain recursive plans and the trees are very similar to one another (same maximum width). Figure 1(d) shows the non-recursive

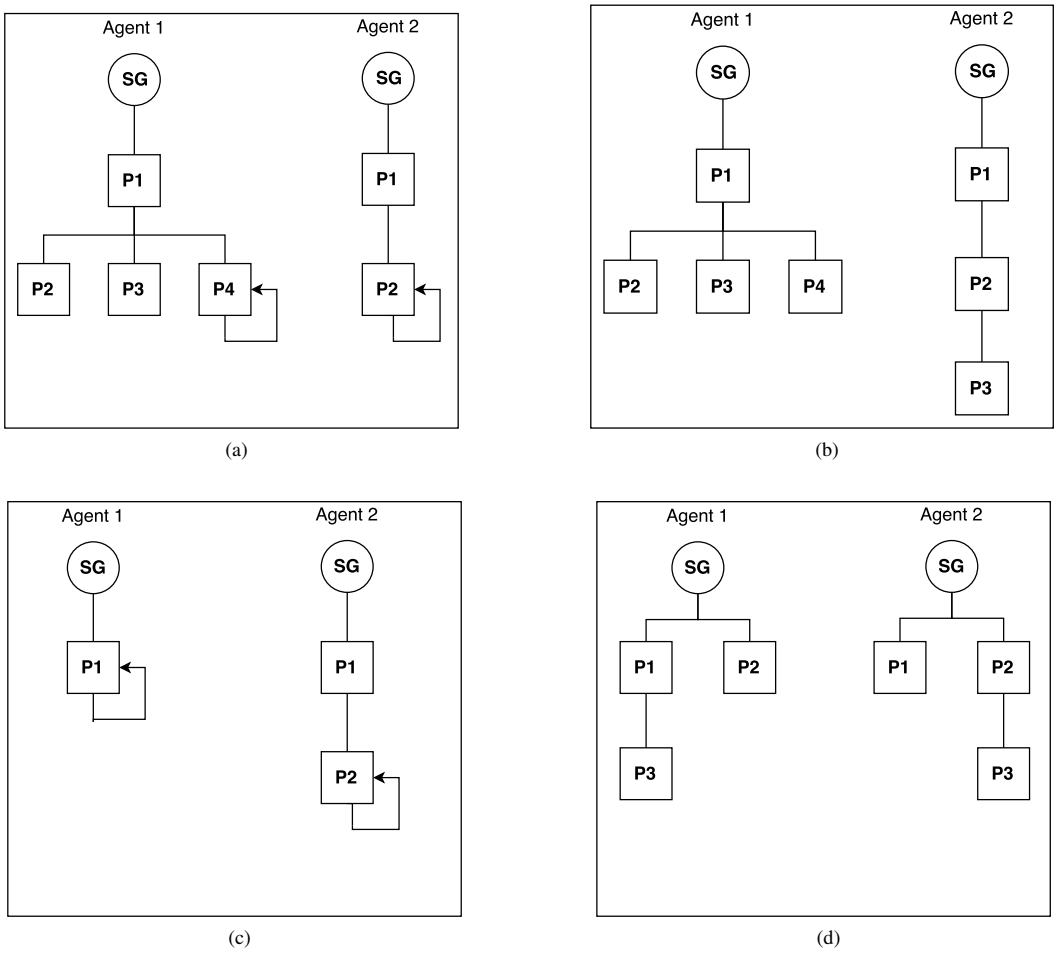

Fig. 1. Possible plan trees comparisons: (a) recursive distinct plan tree; (b) non-recursive distinct plan tree; (c) recursive similar plan tree; (d) non-recursive similar plan tree.

similar plan tree, where plan trees do not contain any recursive plans and the trees are again very similar.

We do not show specific types for the variations when some plan trees contain recursive plans and some do not, because if there is at least one agent that has a plan tree with recursive plans, then it can be considered to belong in the *recursive distinct plan tree* type.

C. Bid Evaluation

Agents place a multi-valued bid, a 4-tuple containing the following criteria: *total number of expanded plans*, the number of plans that the agent was able to expand; *total number of actions*, the number of actions found in the bodies of expanded plans and subplans; *maximum depth of the tree*, the maximum depth found while expanding the tree; and *maximum width of the tree*, the maximum width found while expanding the tree. These criteria are used as heuristics by the organisation manager (initiator) to allocate social goals to agents with the best (according to the heuristic chosen for that application) bids.

Agents with higher total number of expanded plans represent agents with the best computing power available (assuming that they expanded plans up until the deadline). A higher number of actions represent agents that may require longer steps in order to solve a social goal. When compared to other agents, agents with a lower maximum depth often represent that there were no expansion of recursive plans. While agents with a higher maximum depth can represent either very linear plan trees (few options), or the presence of recursive plans. Higher maximum width often indicates that the agent has more options available to accomplish the social goal.

Some assumptions were made for the evaluation of those bidding criteria. First, agents can only place bids for one contract at a time; there is no concurrent or parallel bidding. This is required to avoid cases where one agent could be awarded with most or all social goals. This assumption could be easily ignored if there was a mechanism in place to monitor which contracts have been awarded and to whom. Second, we assume that agents will not place bids on contracts that can

cause conflicts with their previously awarded contract. For example, if a social goal *sg1* causes conflict with *sg2*, and an agent is awarded *sg1*, then it cannot place any bids for *sg2*. Cases where conflicting social goals are necessary are not very common, but a possible solution is to allow agents to subcontract any conflicting social goals that they were awarded with.

IV. Experiments

To evaluate our algorithm and determine which heuristics are better for a certain plan tree configuration, we used the Distributed Online Multi-Agent Planning System (DOMAPS) [8] to run our experiments. The DOMAPS framework consists of four main components: *planning formalism* – a formal representation of the information from the planning domain and problem that will be used during planning; *goal allocation* – the mechanism 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.

Modularity is one of the advantages of DOMAPS, allowing these four components to be separated and replaced with different approaches. We added our CNP approach, bid formulation algorithm, and heuristics for bid evaluation criteria, as the goal allocation mechanism of DOMAPS.

We use the Floods domain [8] for our experiments. It is a multi-agent domain inspired in classical planning domains such as rover and logistics. In Floods, a team of autonomous robots are dispatched to monitor flood activity in a region, a sequence of interconnected areas. Movement through the region occurs from traversing these areas. The Centre for Disaster Management (CDM) establishes a base of operation in the region that is being monitored. The base is used to receive and interpret data, provide some assistance to the robots, and create new social goals.

There are two types of autonomous robots in this domain. Naval units are composed of Unmanned Surface Vehicles (*USVs*) that can move through areas connected by water paths, collect water samples, and take pictures of flood events. Meanwhile, Unmanned Ground Vehicles (*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.

We made four versions of the Floods domain, one for each of the possible topology types of plan trees (described in Section III-B). We used two agents (one USV and one UGV), with both agents starting in the same area. The social goal that the agents had to expand was to take a picture of a nearby flood disaster. In order to compare which bid evaluation criterion was more appropriate for each version, we used the individual planner in DOMAPS to find the solutions, and then analysed the solutions found between the agents.

Table I shows the results of those comparisons. In the first version of the domain, the maximum width of the tree indicated agents with more options. These agents were the ones that found more solutions for the problem, because they had different ways of achieving the social goal, which is very useful in dynamic and non-deterministic environments. In the second version of the domain, both maximum depth and maximum width proved to be important. Agents with a higher width still indicated agents with more options, but lower depth indicated agents that could potentially accomplish the goal using fewer plans than others.

TABLE I
INDEX OF THE BEST EVALUATION CRITERION FOR EACH PLAN TREE COMPARISON: 0 – NOT RELEVANT, 1 – RELEVANT, 2 – VERY RELEVANT.

Floods domain	**n_plans**	**n_actions**	**m_depth**	**m_width**
recursive distinct plan tree	1	1	0	2
non-recursive distinct plan tree	1	1	2	2
recursive similar plan tree	2	1	0	0
non-recursive similar plan tree	2	1	1	0

In versions three and four of the domain, the number of expanded plans was the most important criterion, as it was indicative of computing power, and because agents had a similar plan tree, the maximum depth and width were almost equal. Maximum depth is not relevant when agents have expanded recursive plans because it will be equal, or close to equal, to the number of expanded plans. The number of actions can be relevant in domains where actions have a high cost of execution.

V. Related Work

There are many options that can be used to allocate goals, many of these being well-established coordination protocols found in the literature. A Vickrey [9] auction is an example of an auction protocol that is used quite often in MAS. In this protocol, each agent can make one closed bid (i.e., other agents in the system do not have access to this value) for a particular auction that contains a goal. The task of achieving that goal is assigned to the highest bidder for the price of the second-highest bidder. This means that each bidding agent should simply bid their true private values (i.e., exactly what they think it's worth to them), since this will provide them with the best chance of winning without overspending. Our approach does not consider selfish agents, they will not try to interfere with other agents' bids.

In [10], de-commitment penalties and a Vickrey auction mechanism are proposed to solve a MAP problem in the context of an airport, to solve the problem of deicing and anti-icing aircrafts during winter. The agents are self-interested and can have conflicting interests. Performance on both of these mechanisms were positive, and could be applied as goal allocation or coordination mechanisms in problems involving self-interested agents. Our approach differs from theirs in that it targets domain-independent problems.

Market simulations and economics can also be used to allocate large quantities of resources to agents. For example, in [11], a decentralised market protocol for allocating tasks and scarce resources among agents is presented. We currently do not consider resource-based problems in our approach.

Regarding argumentation, in [12], an argumentation mechanism is utilised for the coordination of agents during MAP. The authors use a well-known defeasible logic programming formalism (DeLP) to implement a defeasible argumentation mechanism. Although it is used as the coordination mechanism in this instance, we argue that it could also be used as a goal allocation mechanism, and that it would not severely suffer from performance issues like it did as a coordination mechanism, since the goal allocation phase tends to be fast paced. This approach could be more useful when dealing with entirely cooperative agents.

When self-interested agents are involved, game theory can be used. The use of game theory in MAP has not been explored very much, at least not when compared to the other approaches. In [13], a mechanism is used to design a set of rules for a game, with the objective of ensuring that the agents behave honestly. This mechanism can be used to generate optimal multi-agent plans, when planning is performed by and for a group of self-interested agents.

VI. Conclusions and Future Work

We have discovered from the experiments shown in this paper that the best evaluation criteria can depend on the similarity between plan trees of agents. If all agents have the same plans (same body, context can be different because we are ignoring it) for a specific social goal (i.e., same or very similar plan tree for that goal), then the best criterion is the total number of expanded plans, since this is a possible indication that the agent has more computing power than the others. Otherwise, if agents have very distinct plan trees (i.e., plan bodies with different actions and subgoals), then the total number of actions, maximum width, and maximum depth can provide more useful information.

Agents with higher maximum width often represented agents with multiple paths in the goal-plan tree, an important information for non-deterministic and dynamic environments. Agents with a lower number of actions often represented agents that would require less action steps to achieve the social goal, especially useful in domains with high cost actions. More experiments on domains with distinct plan trees are needed in order to provide a more in-depth analysis of bid evaluation criteria. A measurement that we are currently investigating is plan-library coverage, which would indicate the percentage of the plan library that the agent was able to expand.

An interesting property of CNP is that it allows bidders to partition the task that was awarded to them into subtasks. As future work, it could be interesting to allow agents to become initiators of tasks that they believe they can only do a part of, subcontracting the rest. This would be especially useful for solving goals that require tightly-coupled agents, that is, where actions that are required to achieve social goals have a lot of dependencies.

Furthermore, the authors of [14] propose some interesting extensions to CNP that could provide several benefits if integrated into our CNP mechanism. Two main ideas are discussed. First, the authors suggest adding a threshold to limit the number of concurrent contracts that each bidder can participate, which could be useful in domains with many social goals where agents can only realistically pursue a limited number of such goals. Second, a degree of availability is suggested to allow the initiator to consider the availability of the bidders along with their respective bid, which could avoid cases where agents are already too busy to be taking on more goals.

Acknowledgment

We are grateful for the support given by CNPq and CAPES.

References

[1] V. Dignum and J. Padget, "Multiagent organizations," in *Multiagent Systems 2nd Edition*, G. Weiss, Ed. MIT Press, 2013, ch. 2, pp. 51–98.

[2] E. H. Durfee, "Distributed problem solving and planning," in *Mutli-agents Systems and Applications*, J. G. Carbonell and J. Siekmann, Eds. New York, NY, USA: Springer-Verlag New York, Inc., 2001, pp. 118–149.

[3] A. S. Rao, "AgentSpeak(L): BDI agents speak out in a logical computable language," in *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world*, ser. MAAMAW '96, Secaucus, NJ, USA, 1996, pp. 42–55.

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

[5] M. de Weerdt and B. Clement, "Introduction to Planning in Multiagent Systems," *Multiagent Grid Syst.*, vol. 5, no. 4, pp. 345–355, 2009.

[6] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Trans. Comput.*, vol. 29, no. 12, pp. 1104–1113, Dec. 1980.

[7] J. Thangarajah, L. Padgham, and M. Winikoff, "Detecting and avoiding interference between goals in intelligent agents," in *International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers, 2003.

[8] R. C. Cardoso and R. H. Bordini, "A distributed online multi-agent planning system," in *Proceedings of the 4th Workshop on Distributed and Multi-agent Planning (DMAP), held with ICAPS'16*, 2016.

[9] W. Vickrey, "Counterspeculation, Auctions and Competitive Sealed Tenders," *Journal of Finance*, pp. 8–37, 1961.

[10] X. Mao, A. Mors, N. Roos, and C. Witteveen, "Coordinating Competitive Agents in Dynamic Airport Resource Scheduling," in *Proceedings of the 5th German conference on Multiagent System Technologies*, Berlin, Heidelberg, 2007, pp. 133–144.

[11] W. E. Walsh and M. P. Wellman, "A market protocol for decentralized task allocation," in *Proceedings of the Third International Conference on Multiagent Systems, ICMAS 1998, Paris, France, July 3-7, 1998*, 1998, pp. 325–332.

[12] E. O. Sergio Pajares Ferrando, "Defeasible argumentation for multi-agent planning in ambient intelligence applications," in *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, vol. 1, 2012, pp. 509–516.

[13] R. P. van der Krogt, M. M. de Weerdt, and Y. Zhang, "Of mechanism design and multiagent planning," in *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08)*, M. Ghallab, C. D. Spyropoulos, N. Fakotakis, and N. Avouris, Eds. IOS Press, 2008, pp. 423–427.

[14] C. Xueguang and S. Haigang, "Further extensions of fipa contract net protocol: Threshold plus doa," in *Proceedings of the 2004 ACM Symposium on Applied Computing*, ser. SAC '04. New York, NY, USA: ACM, 2004, pp. 45–51.