Planning Interactions for Agents in Argumentation-Based Negotiation

Alison R. Panisson, Giovani Farias, Artur Freitas,
Felipe Meneguzzi, Renata Vieira, and Rafael H. Bordini

Pontifical Catholic University of Rio Grande do Sul – PUCRS
Postgraduate Programme in Computer Science – School of Informatics (FACIN)
Porto Alegre – RS – Brazil
{alison.panisson,giovani.farias,artur.freitas}@acad.pucrs.br,
{felipe.meneguzzi,renata.vieira,rafael.bordini}@pucrs.br

Abstract. In this paper, we present the modelling of a domain of argumentation-based negotiation with scarce resources so that Hierarchical Task Network planning can be used for finding appropriate strategies for the negotiating agents. In our modelling, the agents justify their positions on interactions throughout the course of a negotiation episode, and such justifications are taken into account in future interactions. This approach allows the agents to plan the outcome of a negotiation and use this information in their favour. Our method, further, selects the best plan among those found by the HTN planner, based on the plan costs. This allows the agents to choose the best reachable plan in order to reach the desired outcome.

Keywords: HTN Planning, Argumentation-based Negotiation, Multi-Agent Systems

1 Introduction

Argumentation can be divided into two main lines of research in the multi-agent community [1]: (i) argumentation focused on reasoning (nonmonotonic reasoning) over incomplete, conflicting, or uncertain information, where arguments for and against certain conclusions (beliefs, goals, etc.) are constructed and compared; and (ii) argumentation focused on communication/interaction between agents allowing the exchange of arguments that justify a stance and provide reasons to defend previous claims.

Negotiation is one of the key techniques for multi-agent systems where there is interaction between self-interested agents [2]. The second line of research cited above includes argumentation-based negotiation which is an approach for negotiation that has been much investigated in the last few years because of its potential for making the negotiation process more efficient than other approaches, such as game-theoretic and heuristic-based approaches [1]. Argumentation-based negotiation allows us to include additional information (that are typically unavailable in other approaches [2]) within the negotiation exchanges, for example,

allowing participants to *justify* their negotiation stance or to *influence* another agent's negotiation stance [3, 4]. Such additional information makes the negotiation richer and potentially leading to outcomes that the others approaches could not achieve.

Despite argumentation-based negotiation being claimed in the literature as the most efficient approach to negotiation, practical work is scarce. The first practical approaches to appear (e.g., [5]) are based in the interaction of arguments from a single theory, identifying which arguments are *acceptable* given certain semantics, most of them based on Dung's abstract argumentation system [6].

Due to the scarcity of practical work, we propose an approach to argumentation-based negotiation/interaction planning where the agents can plan the outcome of the negotiation/interaction and use this information to obtain advantages over other agents which do not possess such ability. In this work, we formalise methods in Hierarchical Task Network (HTN) [7] using JShop2[1] [8] to specify problems of argumentation-based negotiation/interaction, where HTN operators represent the performatives exchanged between the agents. Given some assumptions, we can simulate the possible outcomes of such interactions.

In this work, we select the best plan among those found by the HTN planner; Each interaction with the planner can return the following possibilities: (i) the number of plans to obtain is passed as parameter (i.e., the planner stops the search for plans after a given number of attempts, which is passed as parameter); or (ii) any number of plans found (i.e., all possible plans are returned). This is an important characteristic of this work which is different from other approaches because agents can request a number of different possible interactions (i.e., plans), and use the best among the returned plans. Further, they can request a new round of planning with a larger number of interactions and possibly obtain an event better plan. Yet, if the time to obtain the better plan is too long, the agent can use a previously returned plan.

The remainder of this paper is organised as follows. In Section 2, we present a background of concepts and technologies used. Section 3 describes some related work in argumentation-based negotiation/interaction and planning. Section 4 describes the development of the work, including the modelling of the domain and problems, the adaptation of the planner to return the best plan, and the problems we modelled to validate the implementation. In Section 5, we make some final remarks and discuss possible directions for future work.

2 Background

In this section we describe the main concepts and technologies involved in the development of this work, explaining briefly Hierarchical Task Network, which we use as planning formalism, and argumentation systems, which is the basis for argumentation-based negotiation used as domain in this work.

[1] Available at `https://sourceforge.net/projects/shop/files/JSHOP2/`.

2.1 Hierarchical Task Network Planning

Classical planning refers generically to planning for restricted state-transition systems. This class of planning problems is also referred to in the literature as STRIPS planning, in reference to STRIPS (Stanford Research Institute Problem Solver), an early planner for restricted state-transition systems [9]. Hierarchical Task Network (HTN) [7] planning is like classical planning in that each state of the world is represented by a set of atoms, and each action corresponds to a deterministic state transition. However, HTN planners differ from classical planners in what they plan for and how they plan for it.

In an HTN planner, the objective is not to achieve a set of goals but instead to perform some set of tasks. The input to the planning system includes a set of operators and also a set of methods, each of which is a prescription for how to decompose some task into some set of subtasks (i.e., smaller tasks). Planning proceeds by decomposing nonprimitive tasks recursively into smaller and smaller subtasks, until primitive tasks that can be performed directly using the planning operators are reached.

HTN planning can be described best by contrasting it with its predecessor, STRIPS-style planning. The representations of the world and the actions in HTN planning are very similar to those of STRIPS-style planning. Each state of the world is represented by the set of atoms which are true in that state. Actions correspond to state transitions, that is, each action is a partial mapping from a set of states to other set of states. However, actions in HTN planning are usually called *primitive tasks* or *operators*.

STRIPS-style planners search for a sequence of actions that brings the world to a state that satisfies certain conditions (achieving goals). Planning proceeds by finding operators that have the desired effects, and by asserting the preconditions of those operators as subgoals. HTN planners search for plans that accomplish task networks, and they plan via task decomposition and conflict resolution. A task network is a collection of tasks that need to be carried out, together with constraints on the order in which the tasks are to be carried out, the way variables are instantiated, and what literals must be true before or after each task is performed [10].

An HTN [7] planner generates a plan by successive refinements of tasks. Tasks are either primitive (equivalent to actions in STRIPS planning) or compound (abstract highlevel tasks). A HTN planner recursively decomposes compound tasks by applying a set of methods until only primitive tasks remain. Methods are elements of the domain knowledge that describe how a higher-level task can be decomposed into more concrete tasks; they constrain the search space, helping to improve the runtime efficiency of the planning algorithm.

2.2 Argumentation

"Argumentation can be seen as the principled interaction of different, potentially conflicting arguments, for the sake of arriving at a consistent conclusion" [1]. Maudet et al. [1] state that argumentation in multi-agent systems has two main lines of research:

i. *Autonomous agent reasoning*, such as, belief revision and decision-making under uncertainty;
ii. As a means for facilitating *multi-agent interaction*, because argumentation naturally provides tools to design, implement, and analyse sophisticated forms of interaction among rational agents.

In the communication/interaction strand, an inherent characteristic of multi-agent systems is that agents need to communicate in order to achieve their individual or collective goals. Agent communication with argumentation techniques allows agents to exchange arguments to justify their stance and to provide reasons that defend their claims. The improvement on expressivity has many potential benefits, but it is often claimed that it should, in particular [1]:

- Make communication more efficient by allowing agents to reveal relevant pieces of information when it is required during a conversation;
- Allow for a verifiable semantics based on the agents' ability to justify their claims (and not on private mental states); and
- Make protocols more flexible, by replacing traditional protocol-base regulation by more sophisticated mechanics based on commitments.

Generally, argumentation is treated abstractly, where the content of individual arguments is not relevant and an overall structure of the relations between arguments is used instead. Abstract argumentation frameworks have their origins in [6], a seminal work that studied the acceptability of arguments. In [6], the focus is on the attack relation between arguments, and the sets of arguments that defend its members, representing the ones that, given a set of arguments, are *acceptable*. The set of arguments that are mutually defensive, and thus cannot be attacked is referred to as being *admissible*.

In argumentation theory, an argument is a pair (S, C), where S denotes the *Support* and C denotes the *Conclusion*, meaning that C is supported by S. Arguments can be *defeated* (a.k.a. *attacked*) by other arguments. Defeat between arguments can be of two types: *rebut* and *undercut*, as defined below.

Definition 1 (Rebut). *Let (S_1, C_1) and (S_2, C_2) be two arguments. Argument (S_1, C_1) rebuts argument (S_2, C_2) iff C_1 is equivalent to the negation of C_2 (i.e., $C_1 \equiv \neg C_2$).*

Definition 2 (Undercut). *Let (S_1, C_1) and (S_2, C_2) be two arguments. Argument (S_1, C_1) undercuts argument (S_2, C_2) iff C_1 is equivalent to the negation of a formula contained in S_2 (i.e., $\exists \varphi . \varphi \in S_2$ and $C_1 \equiv \neg \varphi$).*

An abstract argumentation framework is defined as a tuple $\langle A, R \rangle$ with a set of arguments (A) and a binary relation (R) that defines an attack relation between the arguments.

The status of an argument depends on its *justification state*, whereby an argument is *justified* if it survives the attacks (or has no argument attacking it, or the set of argument defends it from the attacks) and it is *rejected* otherwise. The formal methods that describe this evolution (whether an argument turns out as justified or not) are called *argumentation semantics*.

3 Related Work

In this section we describe some related work. The focus of this section is to describe the main performatives used in argumentative dialogues and some of the work on argumentation-based negotiation, planning, and resource reallocation.

3.1 Argumentative Dialogues

Argumentation-based negotiation is based on the exchange of proposals which the agents believe are acceptable. The exchange of proposals takes place as a dialogue between two or more agents. In an argumentative dialogue, the agents trade propositions for which they have acceptable arguments, and accept proposition put forward by other agents if they find that the arguments are acceptable. The locutions presented at each round and the way that they are exchanged define a formal dialogue game (a dialogue governed by a protocol) in which agents engage [11, 12].

How that dialogue unfolds depends on the message exchanges (what messages agents actually send and how they respond to the messages they receive). This aspect of the dialogue is specified by a protocol (stating the allowed message exchanges), and by some decision-making apparatus within the agent (the agent strategy depends on the choices made by the agent through reasoning).

Some of the locutions/performatives commonly used in argumentative dialogues are found in work such as [11, 13, 12]. Also, the performatives listed below were suggested by McBurney and Parsons [14] to be added to FIPA ACL [15] as the performatives necessary to enable argumentation-based communication between agents. The informal meaning of those performatives are as follows:

- `assert`: an agent that performs an `assert` utterance declares, to all participants of the dialogue, that it is committed to defend this claim. The receivers of the message become aware of this commitment.
- `accept`: an agent that performs an `accept` utterance declares, to all participants of the dialogue, that it accepts the previous claim of another agent. The receivers of the message become aware of this acceptance.
- `retract`: an agent that performs a `retract` utterance declares, to all participants of the dialogue, that it is no longer committed to defend its previous claim. The receivers of the message become aware of this fact.
- `question`: an agent that performs a `question` utterance desires to know the reasons for a previous claim from another agent. The receiver of the message is committed to defend its claim, and presumably will provide the support set for its previous claim.
- `challenge`: the `challenge` performative is similar to the `question` performative, except that the sender of the message is committed to defend a claim contrary to the previous claim of another agent.

We use some of these performatives in our work, where their informal meaning is used in the definitions of operators for the planning process.

3.2 Planning and Argumentation-Based Negotiation

It has been shown that agents able to plan their action have better performance than agents without this capability. An example in the literature is found in [16], where argumentation-based negotiation is planned by agents before the beginning of the interactions with the other agents. There are two key aspects of the approach used in [16]. First, the agents create their plans based on probabilities using the information that they have about the environment and of other agents. Second, the agents interact using appeals [17] (or, as defined in [18], explanatory arguments), threats, and promises of rewards [19–21]. *Appeals* are used to justify a proposal; *threats* are used to warn about negative consequences in case the counterpart does not accept a proposal; and *rewards* are used to promise future rewards if the counterpart accepts the proposal. Our work differs in two aspects. First, we do not use probabilities, rather the planner uses the information present in the agent belief base and the expected reaction of the other agents. Second, we use more general interactions as defined in Section 3.1. In our approach, agents can check which is the best plan to be used to achieve their goals. We assume that agents are cooperative, differently from [16], and we use the domain of argumentation-based negotiation over scarce system resources.

The domain of argumentation-based negotiation for reallocation of scarce resources is well known in the literature and can be found in [22] and [23]. Both papers assume scarce resources (the resources are unique) and that agents need the resources to achieve their goals. In our work, we followed this line, i.e., we are interested in systems where the agents need scarce resources to achieve their goals. Further in [23], the authors make clear the benefit of exchanging additional information during the negotiation, where the agents justify both the request for a resource as well as when they refuse to provide the resource giving the reasons for this. The work in [23] also assumes that agents are cooperatives, i.e., that agents provide information which helps other agents to find the resources needed or to find alternative courses of action, as in our work. Both [22] and [23] do not use planning, but they demonstrate the expressivity of the resource reallocation domain, which we also use in this paper.

4 HTN Formalisation

In this section, we describe the development of our work. We describe the domain modelling, problem modelling, and the predicates used in both models. We also describe the selection of the best plan and present some problems we modelled to validate our implementation.

The domain chosen for modelling and the development of this work was the reallocation of scarce resources, where the resources are unique and the agents negotiate resources which they need to achieve their goals. Further, we assume that agents are cooperative, so:

i. the agents in the system will always provide resources that they do not need in order to achieve their own goals;

ii. if an agent α requests a resource to an agent β who also needs this resource, agent β will suggest a plan that can be used to achieve the goal with resources that it does not need or that uses resources the agent α, who requested the resource, already has (if it knows one such plan).

We represent the agent knowledge (beliefs) as a HTN problem, using predicates such as:

```
(hasPlan <agent> <plan>)
```

to represent an agent plan, where `<agent>` is the agent and `<plan>` is the plan name (we use *p1, p2* etc. to represent plan names).

4.1 Domain Modelling

The operations presented in Table 1 (which describes their meanings), were used to model the domain. The cost of these operations may be amended using the syntax for operators with costs shown below:

```
(:operator (!<operator_name> <parameters...>)
        (<precondition>)
        (<delete list>)
        (<add list>)
        <cost>
)
```

In the operator model with cost, as usual, the operator has a name, parameters, preconditions, deleted predicates, added predicates, and it is possible to assign a cost which is added to the total cost of the plan that uses this operator.

Table 1. Operators used in domain modelling.

Operator	**Meaning**
!*achieve_goal*	achieves the goal
!*obtain_resource*	obtains a resource
!*assert*	makes an assertion
!*justify*	justifies an assertion
!*question*	questions a previous assertion
!*inform_plan*	inform a plan

In addition to operators, an HTN planning domain has methods that decompose non-primitive tasks into more refined tasks.

- **Method *Achieve***: the *achieve* method has two parameters, being the root method in the plan decomposition. The parameters are the *agent* and the *goal* which the agent wishes to achieve. The achieve method checks the plans of the agent that can be use to achieve the goal, and decomposes itself into the execute_plan method recursively applied to all suitable plans.

```
(:method (achieve ?agent ?goal)
  ((agent ?agent) (goal ?goal) (hasPlan ?agent ?plan)
   (achievedBy ?goal ?plan) (plan ?plan))
  ((execute_plan ?agent ?plan ?goal))
)
```

- **Method *execute_plan***: the method *execute_plan* has three options for decomposition (i.e., contexts), according with its preconditions:
  i. First is when the agent has all necessary resources to execute the plan to achieve its goal; in this case, the method is decomposed into the operator !*achieve_goal* because the agent is able to achieve its goal.

```
(:method (execute_plan ?agent ?plan ?goal)
  (forall (?r) ((resource ?r) (need ?plan ?r))
   (has ?agent ?r))
  ((!achieve_goal ?agent ?plan ?goal))
)
```

  ii. Second is when the agent does not have the resources to execute its plan, but it knows who has such resources.

```
(:method (execute_plan ?agent ?plan ?goal)
 ((need ?plan ?r) (resource ?r) (not(has ?agent ?r))
  (believe_has ?agent ?agent1 ?r))
 ((!assert ?agent ?agent1 ?r)
  (!justify ?agent ?agent1 ?r ?goal ?plan)
  (request_resource ?agent ?agent1 ?r ?goal)
  (achieve ?agent ?goal))
)
```

  this method is decomposed into:
  - !*assert*: the agent α informs agent β (who has the resource) that it needs the resource;
  - !*justify*: the agent justifies the need for a resource, stating that the resource is to achieve a certain goal through the execution of a specific plan;
  - *request_resource*: the agent requests the resource to an agent that it believes to have the resource;
  - *achieve*: a recursive call to the root method.

  iii. The third option is when the agent does not have the resources that it needs to execute the plan, and does not know who has them. The plan is decomposed into the !*question* operation that will be used to seek information about who has the resource (it is important to note that *question* refers to question who has the resource needed, and not the question performative presented in Section 3.1), and then the method *achieve* is called recursively.

```
(:method (execute_plan ?agent ?plan ?goal)
  ((need ?plan ?r) (resource ?r)
   (not(has ?agent ?r))
```

```
      (not(believe_has ?agent ?agent1 ?r)))
     ((!question ?agent ?agent1 ?r)
      (achieve ?agent ?goal))
  )
```

- **Method *request_resource***: the method *request_resource* can be decomposed into two different flows:

```
(:method (request_resource ?agent ?agent1 ?r ?goal)
     ((not ((hasGoal ?agent1 ?goal2)
      (achievedBy ?goal2 ?plan2)
      (need ?plan2 ?r))))
     ((!obtain_resource ?agent ?agent1 ?r ?goal))
)

(:method (request_resource ?agent ?agent1 ?r ?goal)
   ((hasGoal ?agent1 ?goal2) (achievedBy ?goal2 ?plan2)
    (hasPlan ?agent1 ?plan2) (need ?plan2 ?r)
    (hasPlan ?agent1 ?plan3) (achievedBy ?goal ?plan3)
    (not(hasGoal ?agent1 ?goal))
    (not(hasPlan ?agent ?plan3)))
   ((sugest_plan ?agent1 ?agent ?plan3 ?goal))
)
```

  i. The first flow is when the agent for which the resource was requested does not need that resource, then the method is decomposed into the operator !*obtain_resource*, which is the method to actually obtain the resource.
  ii. The second option is when the resource is necessary for the agent that is being requested: then the agent will not provide the resource, but it can suggest another plan it knows which achieves the same goal and needs resources it can supply or the requesting agent itself has. This latter method is decomposed into the *sugest_plan* method (shown below) which, in turn, is decomposed into the !*inform_plan* operator, which informs the agent a new plan to achieve its goal.

```
(:method (sugest_plan ?agent2 ?agent1 ?plan ?goal)
   (forall (?r) ((resource ?r) (need ?plan ?r))
    (or (has ?agent1 ?r)
    ((has ?agent2 ?r) (not ((hasGoal ?agent2 ?goal2)
    (achievedBy ?goal2 ?plan) (need ?plan ?r))))))
   ((!inform_plan ?agent2 ?agent1 ?plan ?goal))
)
```

The possible planning flows created by the planner is shown in graph presented in Figure 1, where oval shapes represent methods and quadrangular shapes represent operators. The !*achieve* operator is represented in green because it is the planning goal.

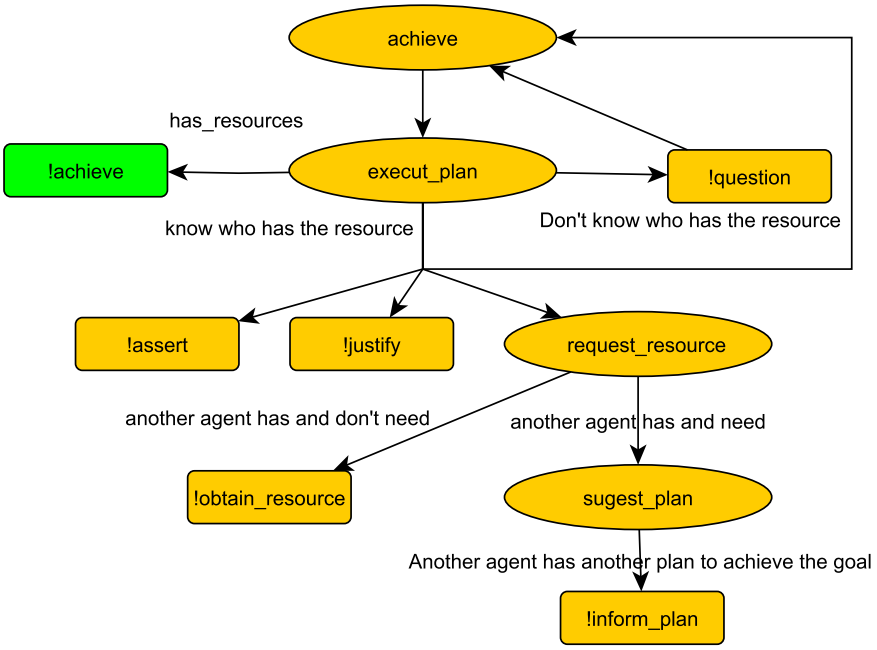

Fig. 1. Graph representing the possible planning flows.

4.2 Problem Modelling

The modelling of the problems was based on the description of beliefs, goals and resources that each agent has in the system, as well as information (beliefs) that agents have about the others agents.

To representation of beliefs and goals of the agents in modelling of planning problems we use predicates, for example, assuming that an agent ag has a certain goal *goal1*, the predicate that represent this idea is defined as ($hasGoal\ ag\ goal1$).

The predicates used in problem modelling are presented in Table 2 along with their meaning.

Other predicates that represent agent knowledge are used during planning, and are not described here in the problem modelling because they are related to information acquired by agents during the interaction. An example of such predicates is ($believe\_has\ ag_1\ ag_2\ r$), where ag_1 is the agent which believes that agent ag_2 has the resource r (this information is acquired through the *question* operator).

In all problems we modelled to be used as experiments, the predicates presented in Table 2 were used in the definition of the initial state to assign tasks to be performed by agents; in our experiments, the task is always to *achieve* particular goals.

Table 2. Predicates used in modelling the planning problems.

Predicates	**Meaning**
$(agent\ ag)$	ag is an agent
$(plan\ p)$	p is a plan
$(resource\ r)$	r is a resource
$(goal\ g)$	g is a goal
$(has\ ag\ r)$	agent ag has resource r
$(hasPlan\ ag\ p)$	agent ag has plan p
$(hasGoal\ ag\ g)$	agent ag has goal g
$(achievedBy\ g\ p)$	goal g is achieved by plan p
$(need\ p\ r)$	plan p needs resource r to be executed

4.3 Selecting the Best Plan

As part of this work, we select the best plan from the list returned by the planner. This list has a number of plans (equal to the number passed as parameter at runtime or as many plans as could be found), each with a cost associated with it.

The cost of plans regards the operators that were executed to perform the plan; by default the planner assigns the value 1 ("one") for each operator that is required to complete the plan, for example, if five operations with the default value are part of the plan's course of action, then the cost of that plan is five. The operator values can be changed, which is done by altering the last parameter in the description of operators in the domain specification.

In our approach, it is possible to request the planner to try to generate thousands of plans, then only the lowest-cost plan will be selected.

4.4 Modeled Problems

Among the problems we modelled in HTN for testing our domain, we present two examples which are interesting because they cover most of the modelled domain.

The first model relates to a Multi-Agent System with four agents (ag_1, ag_2, ag_3, and ag_4). The resources are distributed as follows:

- Agent ag_1 has no resource;
- Agent ag_2 has resource r_2;
- Agent ag_3 has resources r_1 and r_3;
- Agent ag_4 has resource r_4.

Agent ag_1 needs to achieve goal g_1, which can be achieved by plans p_1 or p_2. Plan p_1 needs resources r_1, r_2 and r_3 to be used in order to achieve the goal. Performing plan p_2 requires only resource r_4 for the goal to be achieved. In this first example, only agent ag_1 has a goal, so the other agents provide the resources that will be requested (in accordance with the assumed cooperative

behaviour). As the planner takes into account the order in which the predicates are described in the modelling of the problem, it can provide a plan that is not the best for the agent if the number of plans to be returned is not enough to find it.

Outputs 1.1 and 1.2 show the results of two executions of the first proposed model. Output 1.1 presents the plan found by the planner with a parameter to generate only 5 plans, and Output 1.2 shows the plan found by the planner with parameter 5000; in the latter experiment, 2290 plans were found, which means that all plans were returned, so the best overall plan is guaranteed to be included. Both executions are for the same domain and the same problem, which makes it clear the importance of our online approach which both allows for an immediate response but also provides optimal solutions in the long run.

Output 1.1. First problem, execution 1.

```
5 plans found!
Best Plan -> Plan cost: 25.0

(!question ag1 ag3 r1)
(!assert ag1 ag3 r1 g1)
(!justify ag1 ag3 r1 g1)
(!obtain_resource ag1 ag3 r1 g1)
(!question ag1 ag2 r2)
(!assert ag1 ag2 r2 g1)
(!justify ag1 ag2 r2 g1)
(!obtain_resource ag1 ag2 r2 g1)
(!question ag1 ag3 r3)
(!assert ag1 ag3 r3 g1)
(!justify ag1 ag3 r3 g1)
(!obtain_resource ag1 ag3 r3 g1)
(!achieve_goal ag1 p1 g1)
```

Output 1.2. First problem, execution 2.

```
2290 plans found!
Best Plan -> Plan cost: 4.0

(!assert ag1 ag4 r4 g1)
(!justify ag1 ag4 r4 g1)
(!obtain_resource ag1 ag4 r4 g1)
(!achieve_goal ag1 p2 g1)
```

The second problem explores the situation where an agent is asked to provide a resource that it needs and, in this case, it offers an alternative plan to achieve the goal of the requesting agent. To model this problem, we used four agents and four resources, as in the first problem. Now agent ag_2 has also a goal that requires resource r_3 and agent ag_1 has a goal that needs resources r_1, r_2, and r_3. The agent ag_2 has the resource r_3, so it can achieve its goal. In this example, if another agent request the resource r_3, the agent ag_2 will deny the request because the resource is necessary to achieve its goal.

As agent ag_2 is cooperative, it tries to find an alternative plan for agent ag_1, who requested a resource that ag_2 needs to achieve its goal. In this case, there is the plan p_2 that needs resource r_4 that agent ag_2 has and does not need, therefore the agent informs to agent ag_1 about this plan and agent ag_1 can perform the process to obtain the resource and then achieve its goal.

Output 1.3. Execution of the second problem.

```
21 plans found!
Best Plan -> Plan cost: 17.0

(!question ag1 ag2 r2)
(!assert ag1 ag2 r2 g1)
(!justify ag1 ag2 r2 g1)
(!inform_plan ag2 ag1 p2 g1)
(!question ag1 ag2 r4)
(!assert ag1 ag2 r4 g1)
(!justify ag1 ag2 r4 g1)
(!obtain_resource ag1 ag2 r4 g1)
(!achieve_goal ag1 p2 g1)
```

The result of this implementation shows that the agent tries to obtain the resource and when faced with a new plan $p2$ to achieve that goal, ends up adopting plan $p2$ because it is the better option (lower cost and all resources are available).

5 Conclusion

In this work, we modelled negotiation over resources based on argumentation as planning domain, so that agents can plan the interactions needed to achieve their goals. In our approach, agents can use justifications for requesting resources, allowing agents to (cooperatively) offer alternative plans to achieve the same goal with different resources that can be supplied or are already in possession of the agent that wants to achieve that goal.

In addition, our method selects the best plan among those found by the HTN planner, based on the plan costs, and this is the plan that an agent will typically use. Furthermore, the agent might allow the planner to produce a larger quantity of plans, if more time becomes available for a particular negotiation. We also used different costs on operators; for example the plan with !*question* initially appeared good but ended up being more costly and thus not selected. We argue that this improves the capabilities of agents, which allows the agents to predict the outcome of argumentation-based negotiation/interaction in cooperative environments and with that select the best strategy for itself.

Acknowledgements

Part of the results presented in this paper were obtained through research on a project titled “Semantic and Multi-Agent Technologies for Group Interaction”,

sponsored by Samsung Eletrônica da Amazônia Ltda. under the terms of Brazilian federal law No. 8.248/91.

References

1. Maudet, N., Parsons, S., Rahwan, I.: Argumentation in multi-agent systems: Context and recent developments. In Maudet, N., Parsons, S., Rahwan, I., eds.: ArgMAS. Volume 4766 of Lecture Notes in Computer Science., Springer (2006) 1–16
2. Amgoud, L., Parsons, S., Maudet, N.: Arguments, dialogue, and negotiation. Journal of Artificial Intelligence Research **23** (2000) 2005
3. Jennings, N.R., Parsons, S., Noriega, P., Sierra, C.: On argumentation-based negotiation. In: Int. Workshop on Multi-Agent Systems. (1998)
4. Dimopoulos, Y., Moraitis, P.: Advances in argumentation based negotiation. Chapter 4, Negotiation and Argumentation in Multi-Agent Systems (2011)
5. Berariu, T.: An argumentation framework for bdi agents. In Zavoral, F., Jung, J.J., Badica, C., eds.: Intelligent Distributed Computing VII. Volume 511 of Studies in Computational Intelligence. Springer International Publishing (2014) 343–354
6. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence **77** (1995) 321–357
7. Nau, D., Ghallab, M., Traverso, P.: Automated Planning: Theory & Practice. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2004)
8. Ilghami, O.: Documentation for jshop2. Technical report, University of Maryland, Department of Computer Science, College Park, MD 20742, USA (May 2006)
9. Fikes, R.E., Nilsson, N.J.: Strips: A new approach to the application of theorem proving to problem solving. Artificial Intelligence **2**(3–4) (1971) 189 – 208
10. Erol, K., Hendler, J.A., Nau, D.S.: Complexity results for hierarchical task-network planning. Annals of Mathematics and Artificial Intelligence **18** (1996) 69–93
11. Amgoud, L., Maudet, N., Parsons, S.: Modeling dialogues using argumentation. In: ICMAS, IEEE Computer Society (2000) 31–38
12. Parsons, S., Wooldridge, M., Amgoud, L.: An analysis of formal inter-agent dialogues. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1. AAMAS '02, New York, NY, USA, ACM (2002) 394–401
13. Parsons, S., McBurney, P.: Argumentation-based dialogues for agent coordination. group decision and negotiation. Group Decision and Negotiation (2004)
14. McBurney, P., Parsons, S.: Locutions for argumentation in agent interaction protocols. In van Eijk, R.M., Huget, M.P., Dignum, F., eds.: AC. Volume 3396 of Lecture Notes in Computer Science., Springer (2004) 209–225
15. Foundation for Intelligent Physical Agents: Fipa communicative act library specification. http://www.fipa.org/specs/fipa00037 (2002)
16. Monteserin, A., Amandi, A.: Argumentation-based negotiation planning for autonomous agents. Decision Support Systems **51**(3) (jun 2011) 532–548
17. Sycara, K.P.: Persuasive argumentation in negotiation. Theory and Decision **28**(3) (may 1990) 203–242
18. Amgoud, L., Prade, H.: Generation and evaluation of different types of arguments in negotiation (2004)

19. Amgoud, L., Prade, H.: Formal handling of threats and rewards in a negotiation dialogue. In Parsons, S., Maudet, N., Moraitis, P., Rahwan, I., eds.: ArgMAS. Volume 4049 of Lecture Notes in Computer Science., Springer (2005) 88–103
20. Kraus, S., Sycara, K., Evenchik, A.: Reaching agreements through argumentation: A logical model and implementation. Artificial Intelligence **104** (1998) 1–69
21. Sierra, C., Jennings, N.R., Noriega, P., Parsons, S.: A framework for argumentation-based negotiation. In: Proceedings of the 4th International Workshop on Intelligent Agents IV, Agent Theories, Architectures, and Languages. ATAL '97, London, UK, UK, Springer-Verlag (1998) 177–192
22. Rahwan, I., Pasquier, P., Sonenberg, L., Dignum, F.: A formal analysis of interest-based negotiation. Annals of Mathematics and Artificial Intelligence **55**(3-4) (2009) 253–276
23. Hussain, A., Toni, F.: On the benefits of argumentation for negotiation-preliminary version. In: Proceedings of 6th European workshop on multi-agent systems (EUMAS-2008). (2008)