Exploiting Domain Knowledge in Collaborative Missions

Chukwuemeka David Emele*, Murat Şensoy*, Timothy J. Norman* and Simon Parsons†
* Department of Computing Science, University of Aberdeen, Aberdeen, Scotland, UK
† Department of Computer and Information Science, Brooklyn College, City University of New York, NY, USA

Abstract—**In collaborative missions, coalition members depend on one another to deliver on different aspects of shared goals. In such settings, task delegation decisions are complicated because activities of coalition members may be regulated by policies. Especially when such policies are private, learning these policies become crucial to estimate the outcome of delegation decisions. In this paper, we present an approach that utilises domain knowledge in aiding the learning of policies. Our approach combines ontological reasoning, machine learning and argumentation in a novel way to accomplish this. Using our approach, decision makers can reason about the policies that others are operating with, and make informed decisions about to whom to delegate a task. In a series of experiments, we demonstrate the utility of this novel combination of techniques. Our empirical evaluation shows that more accurate models of others' policies can be developed more rapidly using various forms of domain knowledge.**

I. Introduction

In collaborative missions, members (whether human or artificial) engage in joint activities, which often require them to share resources, act on each others' behalf, communicate and coordinate individual acts, and so on. Such collaborations may fail to achieve desired goals if joint plans are not properly resourced and tasks delegated to appropriately competent agents. Irrespective of the field of endeavour, coalition members depend on one another to deliver on different aspects of shared goals. However, they may operate under policies, and some policies may prohibit a member from providing a resource to another under certain circumstances. Such policies might regulate what resources may be released to a partner from some other organisation, under what conditions they may be used, and what information regarding their use is necessary to make a decision. In addition, policies may govern actions that can be performed either to pursue individual goals or on behalf of another.

Coalition members cannot assume that such policies are public knowledge nor can they presume that others will be willing to reveal their policies (at least, not directly). There are many reasons why coalition members may be reluctant to share (or reveal) their policies. Some of these include: (1) they may not want others to know their internal motives; (2) in some situations, they may have a policy that forbids them from revealing their policies (or certain aspects of their policies) directly to others; (3) fear of exploitation; and so on.

In such settings, it is very important to ascertain whether or not the other party possesses constraints that will hinder him from acting as requested or desired. In fact, coalition members cannot presuppose that the policies of their peers will always permit them to carry out tasks delegated to them. In addition, agents must accept the fact that a number of constraints can result in similar observable behaviours, or actions. So the question is, "how can agents identify what (social) constraints others are working with?" We envisage that one way to do this is to learn the policy of others by observing their behaviours. This approach is suitable for environments where there is a one-to-one mapping between policy and behaviour. Another way to do this is to use evidence derived from argumentation-based dialogue. Here, behaviour is correlated with both policy and other constraints, such as resource availability. Evidence acquired during dialogue is then used to refine models of others' policies.

In Emele *et al.* [2], we show that intelligent agents can determine what policies others are operating within by mining the data gathered from past encounters with that agent (or similar agents) as they collaborate to solve problems. This prior research uses a novel combination of evidence derived from argumentation-based dialogue (which we refer to as ADE) and machine learning to build stable models of others' policies. In this paper, we explore the question that given agents may have access to some background (or ontological) domain knowledge, how can we exploit such knowledge to improve models of others' policies? To do this, we propose the use of ontological reasoning, argumentation and machine learning to aid in making effective predictions about who to approach if some other collaborator is to be delegated to provide a resource or perform a task on the behalf of another.

The rest of this paper is organised as follows. Section II discusses delegation in collaborative missions. Section III presents our approach for learning others' policies. Section IV reports the results of our evaluations. Section V summarises our findings, discusses related work and outlines future directions. Section VI concludes.

II. Delegating in Collaborative Missions

Task delegation, in general, is concerned with identifying a suitable candidate (or candidates) to transfer authority to act on one's behalf [3]. In a collaborative settings, it entails finding

This paper is a submission to the internal ITA project conference. The research reported here appears in the peer-reviewed AAMAS 2011 workshop on Agents and Data Mining Interaction held in Taipei, Taiwan [1]. Please cite as C. D. Emele, T. J. Norman, M. Sensoy, and S. Parsons. Exploiting domain knowledge in making delegation decisions. In Proceedings of The Seventh International Workshop on Agents and Data Mining Interaction, 2011.

agents who possess the required expertise (or resources), and whose policies permit the performance of the required action (or provision of the required resource). If an agent is allowed to perform an action (according to its policy) then we assume it will be willing to perform it when requested, provided it has the necessary resources and/or expertise, and that doing so does not yield a negative utility.

We assume that there is a set $\mathcal{T} = \{t_1, \cdots, t_n\}$ of tasks which agents are assigned to fulfill. The fulfillment of each task requires a number of resources to be used. We focus on the resource acquisition (or allocation) necessary for the successful completion of assigned tasks. Resources generally refers to physical equipment, capabilities, expertise or information that are required to carry out a task. For example, $helicopters$, $jeeps$, etc. We define the set of resources as follows:

Resources The set of resources, denoted as $\mathcal{R}$, is a finite set such that $r_1, r_2, \ldots r_n \in \mathcal{R}$.

Resource Requirements The resource requirements of tasks is a function which maps a task to a set of resources required to carry out that task, and is defined as follows:

$$requirements : \mathcal{T} \rightarrow 2^{\mathcal{R}}$$

In our framework, an agent that has been assigned a task is solely responsible for its fulfillment. However, an agent can delegate an aspect of a task to another. For example, if agent x is responsible for performing task t_x, which requires the use of some resource r_x (i.e. $r_x \in requirements(t_x)$) then x may choose to delegate the provision of r_x to another agent y_1. Provided agent y_1 has resource r_x, and does not have any policy that forbids the provision of r_x then we assume y_1 will make r_x available to x.

Since policies are private, agent x needs to find effective ways of delegating the provision of resource r_x if she is to successfully resource task t_x. To do so, our approach allows agents to find out partners whose policy constraints will most likely, according to a chosen metric, permit to execute the delegated task — in this case, resource provision. In our framework, whenever there is a task to be delegated, policy predictions are generated alongside their confidence values from the policy models that have been learned over time. Confidence values of favourable policy predictions are easily compared to determine which candidate to delegate the task to. In our case, confidence values range from 0 to 1, with 0 being *no confidence* in the prediction, and 1 being *total confidence*.

The delegating agent explores the candidate space to identify suitable candidates to whom it can delegate a task. In these terms, our delegation problem is concerned with finding potential candidates whose policies permit to perform the delegated task, and thereafter, selecting the most promising candidate from the pool of eligible candidates. Borrowing ideas from economics, we assume that some payment will be made to an agent for performing a delegated task (e.g. payment for the provision of a service).

Example 1: Consider a situation where an agent x is collaborating with a number of agents, y_1, y_2, y_3, and y_4, to solve an emergency response problem. Let us assume that agent x does not have a *helicopter* in its resource pool, and that each of agents y_1, y_2, y_3, and y_4 can provide *helicopters*, *jeeps*, *vans*, *bikes*, *fire extinguishers*, and *unmanned aerial vehicles (UAVs)*.

Agent x in Example 1 has to decide which of the potential providers, y_1, y_2, y_3, and y_4 to approach to provide the *helicopter*. Let us assume that the four providers advertise similar services. Agent x, at this point, attempts to predict the policy of the providers with respect to task delegation (or resource provision). This prediction is based on policy models built from past experience with these providers (or similar agents). Assuming the predictions are as follows: (i) y_1 *will accept to provide the helicopter with 0.6 confidence*; (ii) y_2 *will accept with 0.9 confidence*; (iii) y_3 *will accept with 0.7 confidence*; and (iv) y_4 *will decline with 0.8 confidence*. If the decision to choose a provider is based on policy predictions alone, then y_2 is the best candidate.

III. Learning Others' Policies

The framework we propose here enables agents to negotiate and argue about task delegation, and use evidence derived from argumentation to build more accurate and stable models of others' policies.

A. Policies

Agents have policies that govern how resources are deployed to others. In our model, policies are conditional entities (or rules) and so are relevant to an agent under specific circumstances only. These circumstances are characterised by a set of features, e.g., vehicle type, weather conditions, etc.

Features Let $\mathcal{F}$ be the set of all features such that $f_1, f_2, \ldots \in \mathcal{F}$. We define a feature as a characteristic of the prevailing circumstance under which an agent is operating (or carrying out an activity).

Our concept of policy maps a set of features into an appropriate policy decision. In our framework, an agent can make one of two policy decisions at a time, namely (1) *grant*, which means that the policy allows the agent to provide the resource when requested; and (2) *deny*, which means that the policy prohibits the agent from providing the resource.

Policies A policy is defined as a function $\Pi : \vec{\mathcal{F}} \rightarrow \{grant, deny\}$, which maps feature vectors of agents, $\vec{\mathcal{F}}$, to appropriate policy decisions.

In order to illustrate the way policies may be captured in this model, we present an example. Let us assume that f_1 is resource, f_2 is purpose, f_3 is weather report (with respect to a location), f_4 is the affiliation of the agent, and f_5 is the day the resource is required, then $\mathbb{P}_1$, $\mathbb{P}_2$, and $\mathbb{P}_3$ in Table I will be interpreted as follows:

> $\mathbb{P}_1$**:** You are **permitted** to release a *helicopter* (h), to an agent if the $helicopter$ is required for the purpose of transporting relief materials (trm);

TABLE I
AN AGENT'S POLICY PROFILE.

Policy Id	f_1	f_2	f_3	f_4	f_5	Decision
$\mathbb{P}_1$	h	trm				grant
$\mathbb{P}_2$	av		vc			deny
$\mathbb{P}_3$	j					grant
$\mathbb{P}_4$	c		vc	xx		grant
...	...	...	...	...	...	...
$\mathbb{P}_n$	q	yy	w	xx	z	deny

$\mathbb{P}_2$**:** You are **prohibited** from releasing an *aerial vehicle* (av) to an agent in bad weather conditions - e.g. volcanic clouds (vc);

$\mathbb{P}_3$**:** You are **permitted** to release a *jeep* (j) to an agent.

In the foregoing example, if *helicopter* is intended to be deployed in an area with volcanic clouds then the provider is forbidden from providing the resource but might offer a ground vehicle (e.g. *jeep*) to the seeker if there is no policy prohibiting this and the resource is available. Furthermore, whenever a seeker's request is refused, the seeker may challenge the decision, and seek justifications for the refusal. This additional evidence is beneficial, and could be used to improve the model, hence, the quality of decisions made in future episodes.

B. Argumentation-based Dialogue

Figure 1 illustrates the protocol employed in this framework, which guides dialogical moves. Our approach in this regard is similar to the dialogue for resource negotiation proposed by McBurney & Parsons [4].

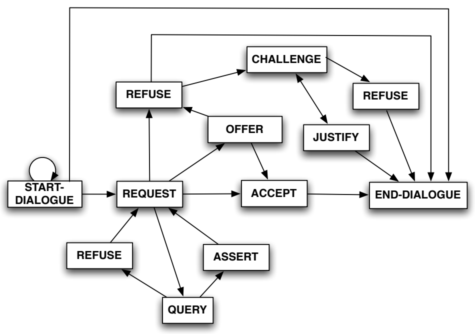


Fig. 1. The negotiation protocol.

To illustrate the sorts of interaction between agents, consider the example dialogue in Table II. Let x and y be seeker and provider agents respectively. Suppose we have an argumentation framework that allows agents to ask for and receive explanations (as in Table II, *lines* 11 *and* 12), offer alternatives (counter-propose in Figure 1), or ask and receive more information about the attributes of requests (*lines* 4 *to* 9 in Table II), then x can gather additional information regarding the policy rules guiding y concerning provision of resources.

Negotiation for resources takes place in a turn-taking fashion. The dialogue starts, and then agent x sends a request (line 3, Table II) to agent y. The provider, y, may respond by conceding to the request (accept), rejecting it, offering an alternative, or asking for more information (query) such as in line 4 in Table II. If the provider agrees to provide the resource then the negotiation ends. If, however, the provider rejects the proposal (line 10, Table II) then the seeker may challenge that decision (line 11), and so on. If the provider suggests an alternative then the seeker evaluates it to see whether it is acceptable or not. Furthermore, if the provider agent needs more information from the seeker in order to make a decision, the provider agent would ask questions that will reveal the features it requires to make a decision (query, inform/refuse). The negotiation ends when agreement is reached or all possibilities explored have been rejected.

TABLE II
DIALOGUE EXAMPLE.

#	Dialogue Sequence	Locution Type
1	x: Start dialogue.	START-DIALOGUE
2	y: Start dialogue.	START-DIALOGUE
3	x: Can I have a *helicopter* for \$0.1M reward?	REQUEST
4	y: What do you need it for?	QUERY
5	x: To transport relief materials.	ASSERT
6	y: To where?	QUERY
7	x: A refugee camp near Indonesia.	ASSERT
8	y: Which date?	QUERY
9	x: On Friday 16/4/2010.	ASSERT
10	y: No, I can't provide you with a *helicopter*.	REFUSE
11	x: Why?	CHALLENGE
12	y: I am not permitted to release a *helicopter* in volcanic eruption.	JUSTIFY
13	x: There is no volcanic eruption near Indonesia.	CHALLENGE
14	y: I agree, but the ash cloud is spreading, and weather report advises that it is not safe to fly on that day.	JUSTIFY
15	x: Ok, thanks.	END-DIALOGUE

C. Learning from past experience through dialogue

When an agent has a collection of experiences with other agents described by feature vectors (see Section III-A), we can make use of existing machine learning techniques for learning associations between sets of discrete attributes (e.g. $f_1, f_2, \ldots f_n \in \mathcal{F}$) and policy decisions (i.e., *grant* and *deny*). In previous research [5], we investigated three classes of machine learning algorithms: (i) instance-based learning (using k-nearest neighbours); (ii) rule-based learning (using sequential covering); and (iii) decision tree learning (using C4.5). Figure 2 shows an example decision tree representing a model of the policies of some other agent learned from interactions with that agent. Nodes of the decision tree capture features of an agent's policy, edges denote feature values, while the leaves are policy decisions.

The machine learning algorithms were chosen to explore the utility of different classes of learning techniques. Instance-based learning is useful in this context because it can adapt to and exploit evidence from dialogical episodes incrementally as they accrue. In contrast, decision trees, and rule learning are not incremental; the tree or the set of rules must be

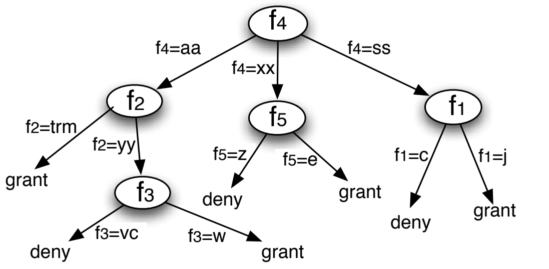


Fig. 2. Example decision tree.

reassessed periodically as new evidence is acquired[1]. Learning mechanisms such as sequential covering, decision trees do have a number of advantages over instance-based approaches; in particular, the rules (or trees) learned are more amenable to scrutiny by a human decision maker.

The training examples used in each learning mechanism are derived from plan resourcing episodes (or interactions), which involves resourcing a task t using provider y and may result in $(\vec{F}_y, grant)$ or $(\vec{F}_y, deny)$. In this way, an agent may build a model of the relationship between observable features of agents' and the policies they are operating under. Subsequently, when faced with resourcing a new task, the policy model can be used to obtain a prediction of whether or not a particular provider has a policy that permits the provision of the resource. In this paper, we take this aspect of the research further by investigating *semantic-enriched decision trees* ($STree$), which extend C4.5 decision trees using ontological reasoning to explore how much domain knowledge can improve learning.

D. Learning from domain knowledge

We argue that domain knowledge can be used to improve the performance of machine learning approaches. Specifically, in this section, we will describe how we can exploit domain knowledge to improve C4.5 decision trees.

1) C4.5 Decision Tree Algorithm: In this section, we shortly describe the induction of C4.5 decision trees. Then, in the following section, we describe how domain knowledge can be exploited during tree induction.

The well-known C4.5 decision tree algorithm [6] uses a method known as divide and conquer to construct a suitable tree from a training set S of cases. If all the cases in S belong to the same class C_i, the decision tree is a leaf labeled with C_i. Otherwise, let B be some test with outcomes $\{b_1, b_2, \ldots, b_n\}$ that produces a partition of S, and denote by S_i the set of cases in S that has outcome b_i of B. The decision tree rooted at B is shown in Figure 4, where T_i is the result of growing a sub-tree for the cases in S_i. The root node B is based on an attribute that best classifies S. This attribute is determined using information theory. That is, the attribute having the highest *information gain* is selected.

[1]For these algorithms we define a *learning interval*, ϕ, which determines the number of plan resourcing episodes (or interactions) an agent must engage in before building (or re-building) models of others' policies.

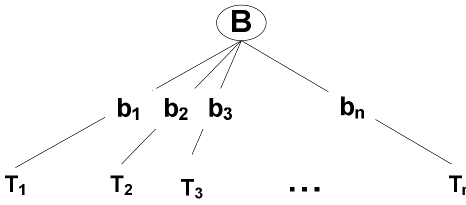


Fig. 4. Tree rooted at the test B and its branches based on its outcomes.

Information gain of an attribute is computed based on *information content*. Assume that testing an attribute A in the root of the tree will partition S into disjoint subsets $\{S_1, S_2, \ldots, S_t\}$. Let $RF(C_i, S)$ denote the relative frequency of cases in S that belong to class C_i. The information content of S is then computed using Equation 1. The *information gain* for A is computed using Equation 2.

$$I(S) = -\sum_{i=1}^{n} RF(C_i, S) \times log(RF(C_i, S)) \quad (1)$$

$$G(S, A) = I(S) - \sum_{i=1}^{t} \frac{|S_i|}{|S|} \times I(S_i) \quad (2)$$

Once the attribute representing the root node is selected based on its information gain, each value of the attribute leads to a branch of the node. These branches divide the training set used to create the node into disjoint sets $\{S_1, S_2, \ldots, S_t\}$. Then, we recursively create new nodes of the tree using these subsets. If S_i contains training examples only from the class C_i, we create a leaf node labeled with the class C_i; otherwise, we recursively build a child node by selecting another attribute based on S_i. This recursive process stops either when the tree perfectly classifies all training examples, or until no unused attribute remains (see Algorithm 1).

Algorithm 1 The C4.5 Decision Tree Algorithm.

```
1: Input : the set of non-categorical attributes $R$
2: Input : the categorical attribute $C$
3: Input : a training set $S$
4: Output: a decision tree $T$
5: $T = \{\}$
6: if $S = \{\}$ then
7:     $T = SingleNodeWithValue(``Failure")$
8:     Return $T$
9: end if
10: if $\forall s \in S$ such that $c = getValueOfClass(s)$ then
11:     $T = SingleNodeWithValue(c)$
12:     Return $T$
13: end if
14: if $R = \{\}$ then
15:     mostFrequent = getMostFrequentValueOfClass(S)
16:     $T = SingleNodeWithValue(mostFrequent)$
17:     Return $T$
18: end if
19: $D = AttributeWithLargestNormalisedGain(S, R)$
20: Let $d_j$ be the values of $D$ such that $j = 1 \cdots m$
21: Let $S_j \subset S$ consist of instances with value $d_j$ for attribute $D$ such that $j = 1 \cdots m$
22: Create a tree, $T$, with root $D$, having edges labeled as $d_1, d_2, \cdots, d_m$
23: Recurse on the sublists obtained by splitting on $D$
24: Add those nodes as children of the node that was splitted
25: Return $T$
```

Table III lists 10 training examples, where $Type$, Age, and $Price$ are the only features. C4.5 decision tree algorithm makes induction only over numerical attribute values. However, it could not make induction or generalisation over the

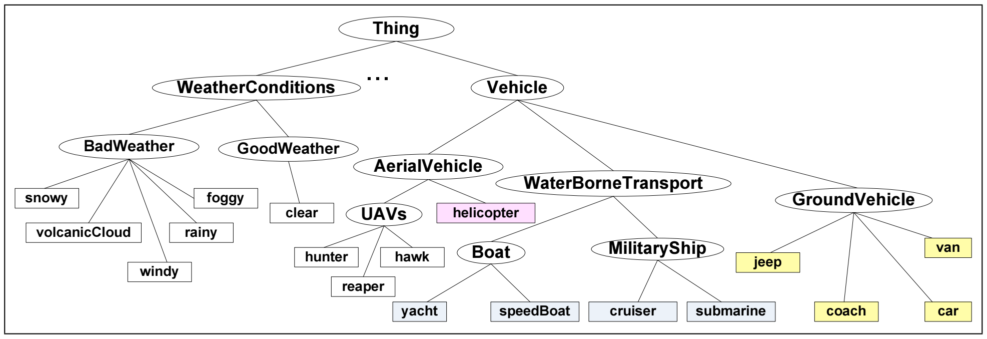


Fig. 3. A simple ontology for vehicles and weather conditions. Ellipsis and rectangles represent concepts and their instances respectively.

TABLE III
TRAINING EXAMPLES.

#	Vehicle ID	Type	Age	Price	Class
1	Van789	Van	10	10,000	*grant*
2	Van999	Van	5	20,000	*grant*
3	Car767	Car	8	5,000	*grant*
4	Car777	Car	15	1,000	*grant*
5	Coach09	Coach	2	200,000	*grant*
6	Yacht121	Yacht	20	300,000	*decline*
7	Yacht123	Yacht	2	500,000	*decline*
8	Speedboat1	Speedboat	4	8,000	*decline*
9	Speedboat2	Speedboat	15	2,000	*decline*
10	Cruiser31	Cruiser	10	100,000	*decline*

nominal attribute values (i.e., terms). For instance, a decision node based on the *price* test in Figure 5 can be used to classify a new case with price $\$250,000$, even though there is no case in the training examples with this price value. However, a new case with an unseen type, for instance a *submarine*, cannot be classified using the decision node based on the attribute $Type$.

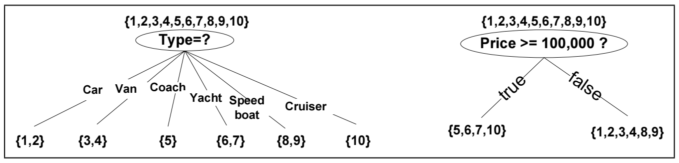


Fig. 5. Decision nodes created using the tests on $Type$ (on the left hand side) and $Price$ (on the right hand side).

2) Domain Knowledge: Domain knowledge consists of such background information that an expert (in a field) would deploy in reasoning about a specific situation. Semantic Web technologies allow software agents to use ontologies to capture domain knowledge, and employ ontological reasoning to reason about it [7]. Figure 3 shows a part of a simple ontology about vehicles and weather conditions. The hierarchical relationships between terms in an ontology can be used to make generalisation over the values of features while learning policies as demonstrated in Example 2. Policies are often specified using numerical features (e.g., vehicle price) and nominal features (e.g., vehicle type). Each nominal feature may have a large set of possible values. Without domain knowledge, the agent may require a large training set containing examples with these nominal values. However, domain knowledge allows agents to reason about to the terms unseen in the training set and learn more general policies with fewer number of training examples.

Example 2: Suppose agent x in Example 1 has learned from previous interactions with agent y_1 that there is a policy that forbids y_1 from providing a *helicopter* when the weather is *rainy*, *foggy*, *snowy*, or *windy*. In addition, suppose agent x has learned from previous experience that agent y_1 is permitted to provide a *jeep* in these conditions. This information has little value for x if it needs a helicopter when the weather is not *rainy*, *foggy*, *snowy*, or *windy* but volcanic clouds are reported. On the other hand, with the help of the ontology in Figure 3, agent x can generalise over the already experienced weather conditions and expect that *"agent y_1 is prohibited from providing helicopters in bad weather conditions"*. Such a generalisation allows x to reason about y_1's behavior for the cases that are not experienced yet. That is, with the help of the domain knowledge, agent x can deduce (even without having training examples involving *volcanic clouds* directly) that agent y_1 may be prohibited from providing a *helicopter* if there is an evidence of volcanic clouds in the region.

3) Semantic-enriched decision trees: Here, we present semantic-enriched decision trees ($STree$) built upon the subsumptions relationships between terms in the ontology. These relationships can be derived automatically using an off-the-shelf ontology reasoner [7]. The main idea of $STree$ is to replace the values of nominal attributes with more general terms iteratively during tree induction, unless this replacement results in any decrease in the classification performance.

Algorithm 2 summarises how the values of A are generalised for S. First, we compute the original gain $G(S, A)$ (line 3). Second, we create a set called *banned*, which contains the terms that cannot be generalised further (line 4). Initially, this set contains only the top concept $Thing$. Third, we create the set T that contains A's values in S (line 5). While there is a generalisable term $t \in T$ (lines 6-18), we compute its generalisation t' using ontological reasoning (line 8) and create the set T' by replacing more specific terms in T with t' (line 9). If this term is an instance of a concept, then the generalisation of the term is the concept, e.g., $Yacht$ is generalisation of $Yacht121$. If the term is a concept, its generalisation is its parent concept, e.g., $WaterBorneTransport$ is generalisation of $Yacht$. For instance, let S be the data in Table III, then T would contain $Yacht$, $Speedboat$, $Cruiser$, Van, Car, $Coach$, and

$Cruiser$. If Car is selected as t, t' would be $GroundVehicle$. In this case, T' would contain $Yacht$, $Speedboat$, $Cruiser$, and $GroundVehicle$. Next, we check if the generalisation leads to any decrease in the information gain. This is done by creating a temporary training set s from S by replacing A's values in S with the more general terms in T' (line 10) and then comparing $G(s, A)$ with the original gain g (line 11). If there is no decrease in the information gain, S and T are replaced with s and T' respectively; otherwise t is added to $banned$. We iterate through until we cannot find any term in T to generalise without any decrease in the information gain.

Algorithm 2 Generalising values of nominal attribute A in training set S.

```
1: Input : $S$, $A$
2: Output: $T$
3: $g = G(S, A)$
4: $banned = \{Thing\}$
5: $T = getAttributeValues(S, A)$
6: while $true$ do
7:     if $\exists t$ such that $t \in T \wedge t \notin banned$ then
8:         $t' = generalise(t)$
9:         $T' = replaceWithMoreSpecificTerms(T, t')$
10:        $s = replaceAttributeValues(S, A, T')$
11:        if $G(s, A) = g$ then
12:            $S = s$ and $T = T'$
13:        else
14:            $banned = banned \cup \{t\}$
15:        end if
16:    else
17:        break
18:    end if
19: end while
```

The output of the algorithm would be $\{SeaVessel, GroundVehicle\}$ for the examples in Table III, because any further generalisation results in a decrease in information gain. Hence, a decision node based on $Type$ attribute would be as shown in Figure 6 (left hand side). A new test case (11, $Submarine$, $40years$, $\$800,000$) would be classified as $deny$ using this decision node, because a submarine is a sea vessel and all known sea vessels are labeled as $deny$. If the actual classification of the case is $grant$ instead of $deny$, the decision node would be updated as seen in Figure 6 (right hand side), because generalisation of $Submarine$ or $Cruiser$ now results in a decrease in the information gain.

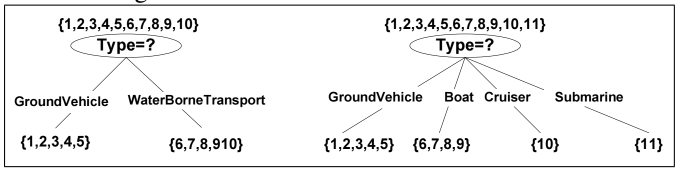


Fig. 6. Decision nodes using the generalisation of cases in Table III (left hand) and after the addition of a new case (11, $Submarine$, 40, 800, 000, $grant$) (right hand).

IV. Evaluation

In evaluating our approach, we employed a simulated agent society where a set of seeker agents interact with a set of provider agents with regard to resourcing their plans over a number of runs. Each provider is assigned a set of resources. Providers also operate under a set of policy constraints that determine under what circumstances they are permitted to provide resources to seekers. In the evaluation reported in this section, we demonstrate that it is possible to use domain knowledge to improve models of others' policies, hence increase their predictive accuracy, and performance. We hypothesize as follows:

- *Hypothesis 1:* In relatively small domains or in situations where sufficiently large training data is available, agents that incorporate domain knowledge will perform no worse than those without domain knowledge.
- *Hypothesis 2:* In complex domains or in situations where limited training data is available, exploiting appropriate domain knowledge in learning policies mean that more accurate and stable models of others' policies can be derived more rapidly than without exploiting such knowledge.

A. Architecture

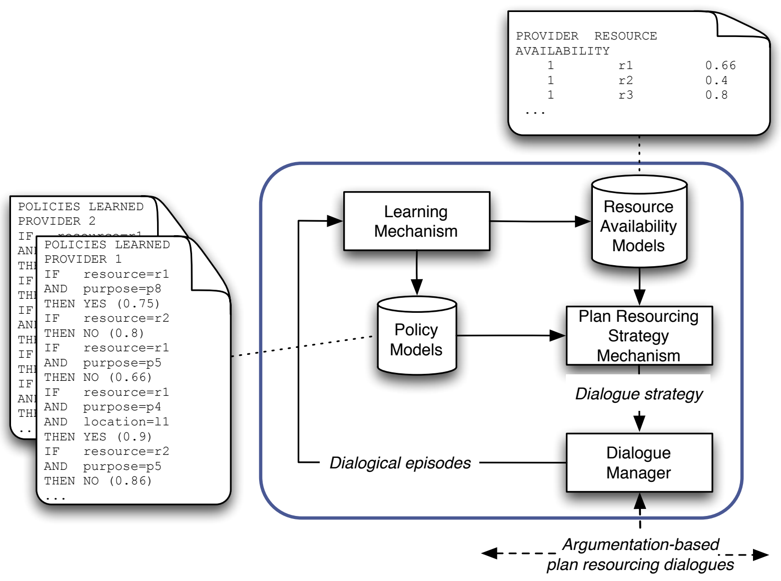
The architecture of our framework, sketched in Figure 7, enables agents to learn the policies and resource availabilities of others through evidence derived from argumentation, and improve those models by exploiting domain knowledge. The dialogue manager handles all communication with other agents. The learning mechanism uses machine learning techniques to reason over the dialogue and attempts to build models of other agents' policies and resource availabilities based on arguments exchanged during encounters. The arguments include the features that an agent requires in order to make a decision about accepting a task delegation or not. The agent attempts to predict the policies of others by reasoning over policy models (built from past experience). Such reasoning is further improved by exploiting background domain knowledge and concept hierarchies in an ontology (as described in Section III-D).

B. Experimental Setup

We evaluate the performance of background domain knowledge in refining models of others' policies in situations where agents' policies are relatively small or large training data is available (which we refer to as the *closed world* scenario) and in situations where agents' policies are relatively large or limited training data is available (which we refer to as the *open world* scenario).

TABLE IV
Experimental Parameters

Parameter	Closed	Open	Description
$Seeker$	1	1	No. of consumer agents
$Provider$	4	4	No. of provider agents
ϕ	100	100	Learning interval
$Task$	800	800	No. of tasks per experiment
$Resources$	5	20	No. of resource types
$Locations$	5	20	No. of locations
$Purposes$	5	20	No. of purposes
$Days$	3	3	No. of simulated days



```
PROVIDER  RESOURCE
AVAILABILITY
    1         r1         0.66
    1         r2         0.4
    1         r3         0.8
 ...
```

```
POLICIES LEARNED
PROVIDER 2
```

```
POLICIES LEARNED
PROVIDER 1
IF   resource=r1
AND  purpose=p8
THEN YES (0.75)
IF   resource=r2
THEN NO (0.8)
IF   resource=r1
AND  purpose=p5
THEN NO (0.66)
IF   resource=r1
AND  purpose=p4
AND  location=l1
THEN YES (0.9)
IF   resource=r2
AND  purpose=p5
THEN NO (0.86)
...
```

Learning Mechanism
Resource Availability Models
Policy Models
Plan Resourcing Strategy Mechanism
Dialogue strategy
Dialogical episodes
Dialogue Manager
Argumentation-based plan resourcing dialogues

Fig. 7. Agent reasoning architecture.

There are five features that are used to capture agents' policies, namely *resource type* ($Resource$), *affiliation*, *purpose* ($Purpose$), *location* ($Location$), and *day* (Day). In each scenario, five agent configurations (SM, $C4.5$, kNN, SC, and $STree$) are investigated. In SM, simple memorisation of outcomes is used. In $C4.5$, C4.5 decision tree classifier [8] is used. In kNN, k-nearest neighbour algorithm [9] is used. In SC, sequential covering rule learning algorithm [10] is used. Lastly, in $STree$, agents use semantic-enriched decision trees to learn policies of others. The experimental parameters are summarised in Table IV.

These attributes provide the possible task context for each agent in the system. In the closed scenario, there are 375 (that is, $3 \times 5 \times 5 \times 5$) possible task configurations. Thus, given 4 providers, the consumer is faced with a problem domain in which there are (potentially) 1,500 individual policies for different task configurations (that is, 375×4). In the open scenario, however, there are $3 \times 20 \times 20 \times 20 = 24,000$ possible task configurations, and the problem domain can (potentially) have 96,000 individual policies for different task configurations (considering the 4 providers). Seeker agents were initialised with random models of the policies of providers. 100 runs were conducted in 10 rounds for each case, and tasks were randomly created during each run from the possible configurations. In the control condition (SM configuration), the seeker simply memorises outcomes from past interactions. Since there is no generalisation in SM, the *confidence* (or prediction accuracy) is 1.0 if there is an exact match in memory, else the probability is 0.5.

C. Results

Figure 8 illustrates the performance of the five configurations we considered in predicting agents' policies in the closed scenario. The results show that $STree$, SC, kNN and $C4.5$ consistently outperform SM. Furthermore, STree, SC and kNN consistently outperform C4.5. It is interesting to see that, with relatively small training set, SM performed better than C4.5. This is, we believe, because the model built by C4.5 overfit the data.

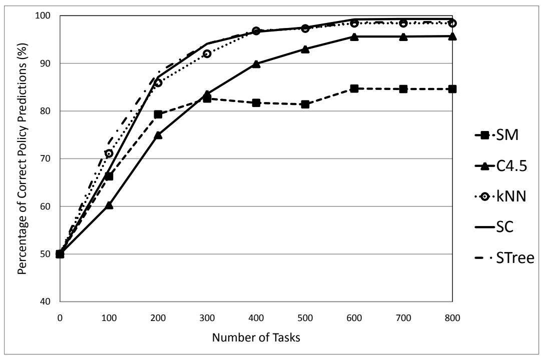


Fig. 8. The effectiveness of exploiting domain knowledge in learning policies (closed scenario).

The decision trees (i.e. STree and C4.5) was pruned after each set of 100 tasks and after 300 tasks the accuracy of the C4.5 model rose to about 83% to tie with SM and from then C4.5 performed better than SM. Similarly, STree performed much better than SC with relatively small training set. We believe, this is because STree takes advantage of domain knowledge and so can make informed inference (or guess) with respect to feature values that do not exist in the training set. After 400 tasks the accuracy of STree, SC, and kNN remained unchanged at about 97% from 400 to 800 tasks. We believe, at this point, almost all the test instances have been encountered and so have been learned (and now exist in the training set for future episodes). This confirms hypothesis 1.

Figure 9 illustrates the effectiveness of exploiting domain knowledge in learning policies in the open scenario. The results show that the technique that exploits domain knowledge

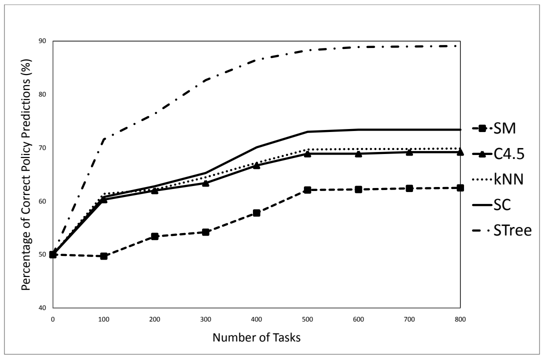


Fig. 9. The effectiveness of exploiting domain knowledge in learning policies (open scenario).

(STree) significantly outperforms the other techniques that did not. After 300 tasks the accuracy of the STree model had exceeded 82% while that of SM, C4.5, kNN, and SC were just approaching 53%, 64%, 65% and 67% respectively. For the sake of clarity, we omit the error bars from the graphs. However, tests of statistical significance were applied to all results presented in this evaluation and they have been found to be statistically significant by t-test with $p < 0.05$. This confirms hypothesis 2.

Overall, these results confirm that exploiting appropriate domain knowledge in learning policies mean that more accurate and stable models of others' policies can be derived more rapidly than without exploiting such knowledge.

V. Discussion

Our results show that background domain knowledge can be exploited to refine and further improve models of others' policies. Of all the approaches examined, the semantically-enriched decision tree learner (STree) built consistently more accurate models of others policies. This is, we believe, as a result of the use of background domain knowledge in making appropriate generalisations/specialisations. When an agent encounters a new term, without domain knowledge, there is no way for the agent to make any useful inference regarding that term/concept on the basis of other terms/concepts. However, had the agent been aware of certain background domain information that relates to that term/concept then it could utilise ontological reasoning to see if there is a semantic link between the new term and any of the other terms that it had encountered before.

In a related work, [11] attempts to learn from, and classify partially specified datasets. In their work, the authors extended C4.5 decision trees [6] with attribute-value taxonomies (AVT). In their approach, classification is done using taxonomies defined over attribute values, which may be specified at different levels of precision. However, our approach is different from theirs because we directly incorporate existing domain ontologies and exploit these ontologies during policy learning. Unlike their approach, which does not allow ontological reasoning during tree induction.

In future, we plan to extend other machine learning methods with domain knowledge and explore how much this extension improves policy learning and enhances agents' support for human decision-making.

VI. Conclusions

We have presented an agent decision-making mechanism where models of other agents are refined through argumentation-derived evidence from past dialogues, and these models are used to guide future task delegation. We have also empirically evaluated our approach and shown that accurate models of others' policies could be learned by exploiting domain knowledge. We believe that this research contributes both to the understanding of how policies may affect delegation in collaborative missions, and applications of evidence derived from argumentation and background domain knowledge to support human decision-making.

Acknowledgements

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] C. D. Emele, T. J. Norman, M. Sensoy, and S. Parsons, "Exploiting domain knowledge in making delegation decisions," in *Proc. of The Seventh International Workshop on Agents and Data Mining Interaction (ADMI 2011)*, Taipei, Taiwan, 2011.

[2] C. D. Emele, T. J. Norman, and S. Parsons, "Argumentation strategies for plan resourcing," in *Proc. of 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Yolum, Tumer, Stone, and Sonenberg, Eds., Taipei, Taiwan, 2011, pp. 913–920.

[3] T. J. Norman and C. Reed, "Delegation and responsibility," in *Intelligent Agents VII: Proceedings of the Seventh International Workshop on Agent Theories, Architectures and Languages*, ser. Lecture Notes in Artificial Intelligence, C. Castelfranchi and Y. Lesperance, Eds. Springer Verlag, 2001, vol. 1986, pp. 136–149.

[4] P. McBurney and S. Parsons, "Games that agents play: A formal framework for dialogues between autonomous agents," *Journal of Logic, Language and Information*, vol. 12, no. 2, pp. 315 – 334, 2002.

[5] C. D. Emele, T. J. Norman, F. Guerin, and S. Parsons, "On the benefit of argumentation-derived evidence in learning policies," in *Proc. of the 3rd Intl. Workshop on Argumentation in Multi-Agent Systems*, Toronto, Canada, 2010.

[6] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[7] P. Hitzler, M. Krötzsch, and S. Rudolph, *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[8] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.

[9] R. Duda and P. Hart, *Pattern classification and scene analysis*. New York: John Wiley & Sons, 1973.

[10] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*, 2nd ed. Morgan Kaufmann, 2006.

[11] J. Zhang and V. Honavar, "Learning decision tree classifiers from attribute value taxonomies and partially specified data," in *Proceedings of the International Conference on Machine Learning*, 2003.