Noname manuscript No.
(will be inserted by the editor)

Learning Strategies for Task Delegation in Norm-Governed Environments

Chukwuemeka David Emele · Timothy J. Norman · Murat Şensoy · Simon Parsons

Received: date / Accepted: date

Abstract How do I choose whom to delegate a task to? This is an important question for an autonomous agent collaborating with others to solve a problem. Were similar proposals accepted from similar agents in similar circumstances? What arguments were most convincing? What are the costs incurred in putting certain arguments forward? Can I exploit domain knowledge to improve the outcome of delegation decisions? In this paper, we present an agent decision-making mechanism where models of other agents are refined through evidence from past dialogues and domain knowledge, and where these models are used to guide future delegation decisions. Our approach combines ontological reasoning, argumentation and machine learning in a novel way, which exploits decision theory for guiding argumentation strategies. Using our approach, intelligent agents can autonomously reason about the restrictions (e.g., policies/norms) that others are operating with, and make informed decisions about whom to delegate a task to. In a set of experiments, we demonstrate the utility of this novel combination of techniques. Our empirical evaluation shows that decision-theory, machine learning and ontology reasoning techniques can significantly improve dialogical outcomes.

1 Introduction

In many scenarios, agents (whether human or artificial) depend on others to act on their behalf. Such dependence is common when agents engage in collaborative activities. However, agents acting individually can also depend on others (e.g., for the delivery of some service). Agents engaging in collaborative problem solving activities often form *alliances* with other agents. Agreements to collaborate are often *ad-hoc* and temporary in nature but can develop into more permanent *alliances*. The formation of agreements may, however, be subject to policy (or norm) restrictions. Such policies might regulate what resources may be released to an agent from some other organisation, under what conditions they may be used, and what information regarding their use is necessary

Chukwuemeka David Emele
Department of Computing Science
University of Aberdeen
AB24 3UE, Aberdeen, UK
E-mail: c.emele@abdn.ac.uk

to make a decision. Similarly, policies may govern actions that can be done either to pursue personal goals or on behalf of another [29].

One important aspect of collaborative activities is resource sharing and task delegation [9]. If a plan is not properly resourced and tasks delegated to appropriately competent agents then collaboration may fail to achieve shared goals. We explore in this paper strategies for plan resourcing where agents operate under policy constraints. This is important not only for autonomous agents operating on behalf of individuals or organisations, but also if these agents support human decision makers in team contexts [6,27,32]. To guide strategies regarding who to approach for a resource and what arguments to put forward to secure an agreement, agents require accurate models of other decision makers that may be able to provide such a resource. The first question addressed in this research is how we may utilise evidence from past encounters to develop accurate models of the policies of others. This is importantly different from the question of whether or not another agent is trustworthy [26,34,14]. Trust models are concerned with building models of others to determine the likelihood of them doing what they promise. We, however, focus on building models of others to determine the likelihood of them promising to do what we ask (e.g., provision of a resource).

Given that agents are operating under policies, and some policies may prohibit an agent from providing a resource to another under certain circumstances, how can we utilise the model of others' policies that have been learned to devise a strategy for selecting an appropriate provider from a pool of potential providers? Secondly, given that 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 these, we propose an agent decision-making mechanism, which utilises a model of the policies and resource availabilities of others to aid in deciding who to talk to and what information needs to be revealed if some other agent is to provide a resource (or perform an action). Our approach utilises a combination of ontological reasoning, argumentation and machine learning to aid in making effective predictions about whom to approach if some other agent is to be delegated to perform a task on the behalf of another.

The rest of this paper is organised as follows. In Section 2, we present our approach to task delegation in norm-governed environments. In Section 3, we describe how agents can learn others' policies from evidence derived from argumentation-based dialogue, and how reasoning over domain knowledge can be exploited to refine models of others' policies with fewer training examples. Section 4 discusses argumentation strategies for task delegation in norm-gorverned environments. Section 5 evaluates our approach through simulations. Section 6 summarises our findings, discusses related work and outlines future directions. We conclude in Section 7.

2 Task delegation in norm-governed environments

Task delegation, in general, is concerned with identifying a suitable candidate (or candidates) to transfer authority to act on one's behalf [20]. In norm-governed systems agents are regulated by sets of behavioural expectations, referred to as norms (or *policies*), which determine the actions an agent is (or is not) allowed to perform in various situations. Delegation in such environments is not just a matter of finding agents that possess the appropriate expertise (or resource); if an agent has the required expertise (or resource) but is operating under a policy that prohibits the performance of that action (or provision of the required resource), it may not take on the delegated task.

In such settings, the aim is to identify agents who possess the required expertise (or resource), and whose policies permit the performance of the required action (or provision of the required resource). It may also be useful to distinguish between agents that are permitted to accede to a request and those that are obliged to do so, but we do not make this distinction here.

Suppose that an agent has been assigned a task t, and that the performance of this task depends on the provision of some resource r. To fulfill its task, agent x must delegate the provision of r to another agent y. Similarly, task t, may depend upon the performance of subtasks $t_{1,1}$ and $t_{1,2}$, and to fulfill its task agent x must delegate $t_{1,2}$ to some other agent y. Provided agent y has resource r (or the expertise to perform task $t_{1,2}$), and has no policy that forbids the provision of r or the performance of $t_{1,2}$ then we assume y will make r available to x or do $t_{1,2}$ for x. In other words, 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 propose a framework that will allow agents to interact, argue and mine data from past encounters and use evidence from these encounters to build a model of the policies that others may be operating with. The purpose of the framework is to enable agents to find effective ways of delegating the provision of resources (or, similarly, the performance of tasks). To achieve this, we need to identify agents whose policy constraints will most likely enable the execution of the delegated task. In our framework, whenever there is a task to be delegated, policy predictions are generated alongside the confidence of those predictions from the policy models that have been learned over time. Confidence values of favourable policy predictions are compared to determine which candidate to approach for a resource, or to delegate a task. The confidence of these predictions are derived from evidence from past encounters.

In this section we present a simple model of conditional policies (or norms), specify the structure of the dialogue that agents use and the forms of evidence that can be exploited in such dialogues, and discuss the issue of the costs that may be incurred in revealing information to others. In the remainder of this paper we focus our discussion on resource negotiation.

2.1 Policies

In order to model our argumentation-based framework, we begin by formulating a mechanism to capture policies. Policies (aka. norms) govern how resources are normally provided to others. In our model, policies are conditional; they are relevant to an agent's decision under specific circumstances. These circumstances are characterised by a set of features. Some examples of features may include: the height of a tower, the weather condition, the temperature of a room, or the manufacturer of a vehicle.

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); that is, the task context. In this paper, we shall use the term *set of features* and *feature vector* interchangeably. Our concept of policy maps a set of features into an appropriate policy decision. We assume that an agent can make one of two policy decisions, namely: *grant*, which means that the policy allows the agent to provide the resource when requested, and *decline*, which means that the policy prohibits the agent from providing the resource.

Table 1 An example of an agent's policy profile.

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

We define a policy as a function with signature $\Pi : 2^{\mathcal{F}} \rightarrow \{grant, decline\}$, which maps feature vectors of tasks, $2^{\mathcal{F}}$, to appropriate policy decisions. In order to illustrate the way policies (or norms) are captured in this model, we present the following examples (see Table 1). Assuming, f_1 is the resource requested, f_2 is the purpose, f_3 is a weather report (with respect to a location), f_4 is the affiliation of the agent in receipt of the resource, and f_5 is the day the resource is required then policies $\mathbb{P}_1$, $\mathbb{P}_2$, and $\mathbb{P}_3$ (see Table 1) will be interpreted as follows:

$\mathbb{P}_1$**:** You are **permitted** to release a *helicopter* (h), to any agent if the *helicopter* is required for the purpose of transporting relief materials (trm) and the weather is good (g).

$\mathbb{P}_2$**:** You are **prohibited** from releasing a *helicopter* to any agent if the weather report says there are volcanic clouds (vc) in the location the agent intends to deploy the *helicopter*.

$\mathbb{P}_3$**:** You are **permitted** to release a jeep (j) to any agent for any purpose, irrespective of the day and the weather report.

If a *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 consumer if the resource is available.

Policies are important factors that regulate agents' behaviour in a society. Given that policies are often private, and agents are required to work together as they collaborate to solve a problem then *how can agents identify what policies others are working within*? Our claim is that there is useful evidence that one can extract from interactions with other agents. Such additional evidence can help to build more accurate models of others' policies. In the next section, we discuss how argumentation-based negotiation allows us to gather such useful evidence.

2.2 Argumentation-based negotiation

Negotiation may take many forms, but here, we focus on argumentation-based negotiation. First, we explore the evidence that argumentation-based dialogue provides in revealing underlying policy constraints, and then we present the protocol employed in this research.

2.2.1 Forms of evidence

Here, we present a number of forms of evidence that can be identified in argumentation-based negotiation. Three important types of evidence are considered in this paper,

namely: (i) seeking information about the issue under negotiation; (ii) providing explanations or justifications; and (iii) suggesting alternatives. This is not intended as an exhaustive list, but do represent three of the most common sources of evidence in argumentation-based dialogue in general.

Seeking further information. When an agent receives a request to provide a resource, it checks whether or not it is permitted to honour the request. To do this, it must compare the details provided by the consumer with the policies it must operate within to make a decision. An example of an agent's policy profile is captured in Table 1. If the details of the task context provided by the consumer is insufficient for the provider to make a decision, it will need to seek further information. For example, if an agent requests for the provision of a *helicopter* (denoted as h in Table 1), then the provider needs to know the weather report of the location where the *helicopter* is to be deployed; therefore, the provider will ask the consumer to forward details about the location and (possibly) to provide a weather report (although this may be acquired from elsewhere).

A request for further information about the features of a task context could indicate some of the characteristics of the constraints a provider agent is operating within. For instance, let us assume that a provider agent has a policy that forbids it from providing a *helicopter*, h, to any consumer agent that intends it to fly in an area affected by *volcanic ash*, vc (see Table 1, Policy Id = $\mathbb{P}_2$). Then, given that a consumer has requested for the provision of a *helicopter* and the provider agent has requested information to ascertain that the consumer does not intend to deploy the *helicopter* in an area with *volcanic clouds*, the consumer could use that information as input to try to model what policies the provider agent may be operating with. Such a request for further information could mean that there are specific values of certain features that may lead to different policy-governed decisions.

Suggesting alternatives. When a provider agent is unable to grant a request because there is either a policy restriction or a resource availability constraint, or both, it may wish to suggest alternatives. For example, a consumer may request the use of a *helicopter* to transport relief materials in bad weather conditions. If the provider is prohibited from providing a *helicopter* in such conditions but permitted to provide a *jeep* then it may offer a *jeep* as an alternative for transporting those materials (provided a jeep is available in its resource pool). If we assume that an agent will only suggest an alternative if that alternative is available and there is no policy that forbids its provision, the suggestion provides evidence regarding the policies of the provider with respect to the suggested resource. Specifically, it is positive evidence that the provider agent does not have a policy that forbids the provision of that resource to the consumer and that it is available. Furthermore, it serves as evidence regarding the model that the other agent has with respect to the equivalence of resources, but we do not consider this here. It is worth noting that we assume agents communicate truthfully. While the issue of deception remains an open problem, some techniques for addressing this assumption have been investigated [34,30].

Justifications. Following a request for a resource, ultimately the provider agent will either agree to provide it or decline the request (though further information may be sought in the interim and suggestions made). In the case where the provider agent agrees to grant the request, the consumer agent obtains a positive example of a task

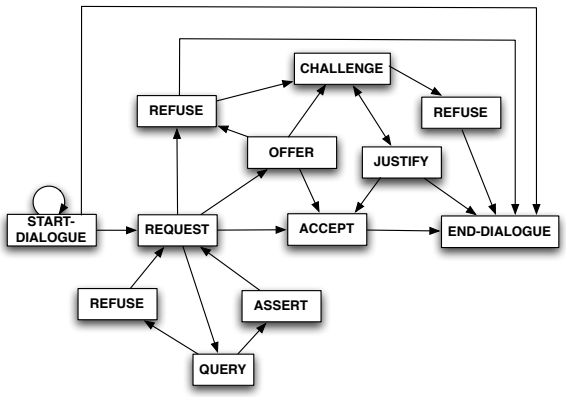


Fig. 1 The negotiation protocol.

context that the provider agent's policies permit the provision of the resource. On the other hand, if the request is refused then the consumer may seek further explanation for the refusal. The refusal could be due to policy restrictions or resource availability constraints or both. The justification provided in response to the challenge may offer further evidence that may help to identify the underlying constraints.

2.2.2 Interaction protocol

Having described three important sources of evidence for learning policy ad resource constraints, here we illustrate how these pieces of evidence can be extracted from a typical model of argumentation-based negotiation. The dialogue structure that we consider is shown in Figure 1. This protocol is modelled as the dialogue for resource negotiation proposed by McBurney and Parsons [18]. The key differences are: (1) agents may ask for more information if the information they have is insufficient to make a decision; and (2) agents can argue about alternatives that have been offered.

To illustrate the sorts of interaction between agents, consider the example dialogue in Table 2. Let x and y be consumer and provider agents respectively. Suppose we have an argumentation framework that allows agents to ask for and receive explanations (as in Table 2, *lines* 11 *to* 14), offer alternatives (*line* 10 in Table 2), or ask and receive more information about the attributes of requests (*lines* 4 *to* 9 in Table 2), then x can gather additional information regarding the policy rules guiding y concerning the provision of resources.

Negotiation for resources takes place in a turn-taking fashion. The dialogue starts, and then agent x sends a request to agent y (e.g., *line* 3, Table 2). The provider, y, may respond by conceding to the request (accept), refusing, offering an alternative resource, or asking for more information (query) such as in *line* 4 in Table 2. If the provider agrees to provide the resource then the negotiation ends. If, however, the provider declines the request then the consumer may challenge that decision, and so on. If the provider suggests an alternative (*line* 10 in Table 2) then the consumer evaluates it to see whether it is acceptable or not. Furthermore, if the provider needs more information from the consumer in order to decide, the provider would ask questions that will reveal the features it requires to make a decision (query, assert/refuse in Figure 1). The negotiation ends when agreement is reached or all possibilities explored have been

Table 2 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: I can provide you with a *jeep* for \$5,000.	OFFER
11	x: But I prefer a *helicopter*, why offer me a *jeep*?	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 then, I accept your offer of a *jeep*.	ACCEPT
16	y: That's alright. Good-bye.	END-DIALOGUE

refused. A complete specification of an argumentation-based dialogue game involves commitment rules, commencement rules, termination rules, etc. (see [18, 25]). We do not provide this additional detail here, however, because this will not add to the key contribution of the paper. A specification of the permitted moves in the dialogue and the evidence that may be revealed by certain moves is sufficient.

Before we discuss in detail how the evidence revealed during argumentation-based negotiation can be exploited to learn models of other agents, it is important to highlight the fact that in many situations there is a cost associated with revealing information to others.

2.3 Information revelation

In situations where private information must be revealed before a service can be provided such as in response to a query, the agent providing this information must take into account the costs of doing so. There are many reasons why there should be cost associated with revealing information. In general, the more information you reveal, the more likely it is that others may become aware of your private plans and goals, and hence be able to exploit that information. For example, if a consumer reveals to an insurance provider that there have been recent cases of (unreported) burglary in the neighbourhood, then the insurance provider might exploit that information and raise the premium. Similarly, if the consumer reveals the same piece of information to a friend then the friend may refuse to lend him a valuable item because she may fear that it could be stolen from his apartment, or, if the friend works for an insurance provider, then she may be obliged to pass on such information to her employers.

These two examples illustrate the possibility that an agent may be exploited on the basis of the information he reveals to others, whether in a hostile or cooperative setting. However, there are situations in which the revelation of an extra piece of information could be beneficial, but that does not alleviate the risk of exploitation by others. This risk is what we refer to as *information cost*, and will be defined in Section 4.

3 Learning policies from dialogue

One of the core goals of this research is to learn models of the policies of others. When an agent has a collection of experiences with other agents described by feature vectors (see Section 2.1), we can make use of existing machine learning techniques to learn associations between these sets of features (i.e. elements of $\mathcal{F}$) and policy decisions.

The aim of building these policy models is to produce predictions about whether or not other agents are likely to be forbidden (or permitted) to perform a delegated task such as providing a resource. One of the main problems regarding learning policies is that agents may not know in advance which features are good indicators of a specific policy decision. In reality, agents may start with some *a priori* model of others' policies, which may be incomplete or inaccurate. Here, we randomise the initial policy model to isolate the effect of learning policies through evidence derived from argumentation. In this research, we have explored a number of means through which policy models may be constructed, such as concept similarity measures, rules and decision trees.

Specifically, we investigate three classes of machine learning algorithms[1] [19], namely instance-based learning (using k-nearest neighbours), rule-based learning (using sequential covering), and decision tree learning (using C4.5). In addition, we present a novel semantic-enriched decision tree learning, STree.

3.1 Policies modelled as concepts

Since policies map features of agents' task contexts into categories of decisions (i.e. *grant* or *decline*), it makes sense to capture them as concepts. Concepts are simply categories that help in classifying objects (in this case feature vectors), which have a set of common features that are relevant. With concept learning, a set of policy features (captured as a feature vector) can be classified based on policy decisions. By capturing an agent's policy decision function in this way, we can make use of standard machine learning techniques (e.g., case-based reasoning [1]) to classify future cases by computing how similar (or close) they are to labelled examples [5]. When a policy decision is made for a given input, one can reason about the concept that has been observed and apply it to future cases. Intuition is that similar task contexts will lead to similar policy decisions. Instance-based learners, such as k-nearest neighbours (kNN) are popular for classifying tasks in this kind of settings.

In this regard, concept learning attempts to abstract the central tendency of policy training examples, and use this for categorising future instances. This means that the classification of a new task context is based on the similarity between prior examples and the current task context. Each example represents a point in the instance space, and new task configurations are classified on the basis of how close they are to others.

In order to identify nearest neighbours, the kNN algorithm adopts a number of distance metrics, which measure the dissimilarities between feature vectors. One such metric is Euclidean distance, which is calculated as follows:

$$d(\mathbf{F}_a, \mathbf{F}_b) = \sqrt{\sum_{i=1}^{n} w_i (f_i(\mathbf{F}_a) - f_i(\mathbf{F}_b))^2}, \quad (1)$$

[1] We use the Weka [35] implementation of these algorithms.

where a policy example is defined as a vector $\mathbf{F} \subseteq \mathcal{F}$, f_i is the ith feature of the policy example, w_i is the weight of the ith feature, and i ranges from 1 to $|\mathbf{F}|$. Generally, the smaller the value of $d(\mathbf{F}_a, \mathbf{F}_b)$ the more similar the two examples are.

Using the above distance metric, the kNN algorithm identifies k nearest neighbours of a given instance. Suppose we define kNN as the set of k nearest neighbours of a given test instance (d_x), we proceed to classify the instance as follows.

$$\textit{classify}(d_x) = \arg\max_{k} \sum_{\mathbf{F}_b \in kNN} z(\mathbf{F}_b, c_k), \tag{2}$$

where d_x is a test example, $\mathbf{F}_b$ is one of its k nearest neighbours in the training set. $z(\mathbf{F}_b, c_k)$ indicates whether or not $\mathbf{F}_b$ is classified to be of class c_k . From Equation 2, the test instance will be classified to belong to the class that has most members in the k nearest neighbours. In other words, the test instance will be assigned a class label based on the majority vote of its k nearest neighbours [28].

3.2 Policies modelled as rules

Rules provide an appropriate means for specifying policies because they allow an agent to find what the decision (or action) for a given set of pre-conditions will be. In other words, given these sets of conditions (derived from features in the task context), what is likely to be the policy decision of the other party. One advantage of representing polices in this way is the fact that they are amenable to scrutiny by a human decision maker. In particular, rules (or trees) are relatively simple to understand, as compared with models produced by other machine learning techniques.

Using rules, it is easier to explain why a particular policy decision is likely to be made by summarising the pre-conditions of the learned rule that is the closest match to the current context. This is especially important in scenarios where agents are acting on behalf of (or supporting) human decision makers. In practical terms, a human decision maker may want to know why an agent is advising him/her to follow a certain course of action [33].[2]

By representing policies as rules, we can make use of standard rule learning techniques (e.g., sequential covering [19,11]) to classify rule sets from training examples. When a particular decision is chosen for a given input, you can trace the reason for that decision by looking at the conditions attached to the rule. A rule is not triggered unless the conditions for the rule are met, and that is exactly how a policy operates. To this end, it is possible to exploit one of the well-known rule-based induction techniques to classify an agent's policies in terms of features in $\mathcal{F}$ [22].

3.3 Policies modelled as decision trees

Decision trees [4] provide an appropriate representational abstraction for policies because they allow a decision (or label) to be found for a given input by selecting paths from the root of the tree on the basis of values (or conditions) specified at nodes to

[2] The rule learning algorithm used here (sequential covering) does not necessarily find the best or smallest set of rules, but other, more sophisticated rule induction methods may equally be employed [15].

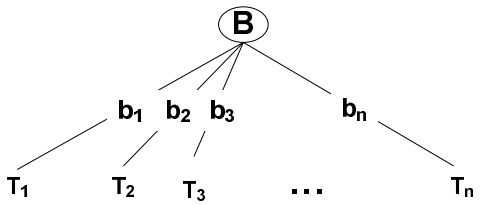


Fig. 2 Tree rooted at the test B and its breaches based on its outcomes.

leaf nodes (decisions). By capturing an agent's policy function as a decision tree, we can make use of standard machine learning techniques to induce trees from labelled examples [8,16,23]. Nodes of the decision tree capture features of an agent's policy, edges denote feature values, while the leaves are policy decisions.

Although decision trees can be used to generate explanations of policy predictions, such explanations are not necessarily formulated in an accessible logical structure, and standard algorithms in the ID3 family cannot exploit background (or domain) knowledge [21]. In this section we briefly outline a decision tree algorithm (C4.5) and then present an extension that employs ontological reasoning to exploit background knowledge during tree induction.

3.3.1 C4.5 decision tree algorithm

The C4.5 decision tree algorithm [24] 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 2, 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 by information theory as the attribute with the highest *information gain*.

The 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 3. The *information gain* for A is computed using Equation 4.

$$I(S) = -\sum_{i=1}^{n} RF(C_i, S) \times log(RF(C_i, S)) \tag{3}$$

$$G(S, A) = I(S) - \sum_{i=1}^{t} \frac{|S_i|}{|S|} \times I(S_i) \tag{4}$$

Once the attribute representing the root node is selected based on its information gain, each value of the attribute leads to a branch from the node. These branches divide the training set 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 class C_i, we create a leaf node labeled with 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 classifies all training examples, or until no unused attribute remains.

Table 3 Training Examples.

#	Type	Age	Price	Class
1	Van	10	10,000	*grant*
2	Van	5	20,000	*grant*
3	Car	8	5,000	*grant*
4	Car	15	1,000	*grant*
5	Coach	2	200,000	*grant*
6	Yacht	20	300,000	*decline*
7	Yacht	2	500,000	*decline*
8	Speedboat	4	8,000	*decline*
9	Speedboat	15	2,000	*decline*
10	Cruiser	10	100,000	*decline*

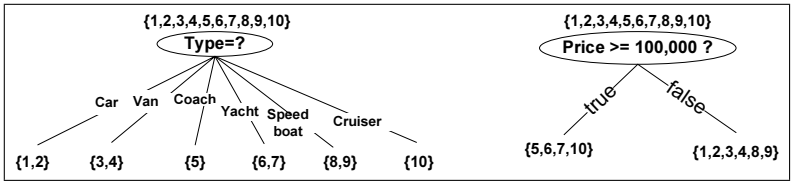


Fig. 3 Decision nodes created using tests on *Type* (on the left) and *Price* (on the right).

Table 3 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. It cannot generalise over the nominal attribute values (i.e., terms). For instance, a decision node based on the *Price* attribute in Figure 3 (on the right) can be used to classify a new case with price equal to $\$250,000$, 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* in Figure 3 (on the left). This limitation can have significant impact on the accuracy of the models of others built by an agent.

3.3.2 Semantic-enriched decision trees

In this section, we describe one way of addressing the problem identified in the preceding section. Semantic Web technologies allow software agents to use ontologies to capture domain knowledge [3,13,10], and employ ontological reasoning to reason about it [31]. Figure 4 shows a part of a simple ontology about vehicles and weather conditions. The hierarchical relationships between terms in an ontology can be used to generalise over the values of features while learning policies as demonstrated in Example 1. Specifically, we describe how we can exploit ontological reasoning to improve the performance of agents in learning the policies of others using C4.5 decision trees.

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 ontological reasoning over relevant domain knowledge, the agent may require a large training set containing examples with these nominal values. However, ontological reasoning allows agents to reason about terms unseen in the training set and learn more general policies with fewer training examples.

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*

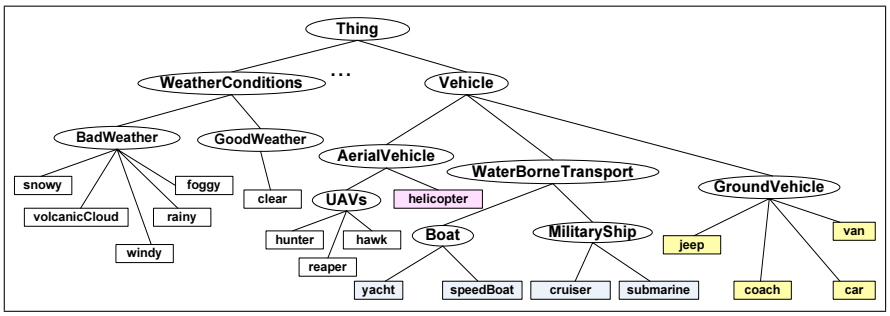


Fig. 4 A simple ontology for vehicles and weather conditions. Ellipsis and rectangles represent concepts and their instances respectively.

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). Suppose agent x 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 4, 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 the behaviour of y_1 for cases that have not yet been encountered. That is, with the help of the domain knowledge, agent x can deduce that agent y_1 may be prohibited from providing a helicopter if there is an evidence of volcanic clouds in the region.

Here, we propose 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 such as *Pellet* [31]. 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 a decrease in the classification performance.

Algorithm 1 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 $Yacht123$ (not shown in the ontology in Figure 4). If the term is a concept, its generalisation is its parent concept, e.g., $Boat$ is generalisation of $Yacht$, while $WaterBorneTransport$ is generalisation of $Boat$. For instance, let S be the data in Table 3, 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 $GroundVehicle$.

Algorithm 1 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: terms = getAttributeValues(S, A)
6: while true do
7:    if ∃t such that t ∈ T ∧ t ∉ 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 ∪ {t}
15:      end if
16:   else
17:      break
18:   end if
19: end while
```

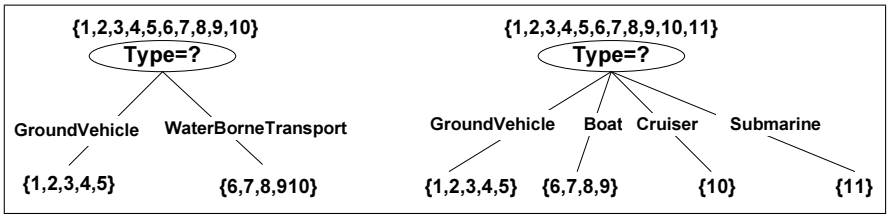


Fig. 5 Decision nodes using the generalisation of cases in Figure 3 (left hand side) and after the addition of a new case (11, *Submarine*, 40, 800,000, *grant*) (right hand side).

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 a decrease in the information gain.

For the examples cited in Table 3, the output of the semantic-enriched decision tree algorithm would be $\{GroundVehicle, WaterBorneTransport\}$, 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 5 (left hand side). A new test case (11, *Submarine*, $40years$, \$800,000) would be classified as *decline* using this decision node, because a submarine is a WaterBorneTransport and all known WaterBorneTransport types are labeled as *decline*. If the actual classification of the case is *grant* instead of *decline*, the decision node would be updated as seen in Figure 5 (right hand side), because generalisation of *Submarine* or *Cruiser* now results in a decrease in the information gain.

The machine learning algorithms investigated in this paper have very different properties. Instance-based learning is useful in this context because it can adapt to and exploit evidence from dialogical episodes as they accrue. In contrast, decision trees and rule learning are not incremental; the tree or the set of rules must be reassessed periodically as new evidence is acquired. We define a *learning interval*, ϕ, which de-

termines the number of interactions an agent must engage in before building (or rebuilding) its policy model. Once an agent has had ϕ interactions, the policy learning process proceeds as follows. For each interaction, which involves resourcing a task t using a given provider, we add the example $(\mathbf{F}, grant)$ or $(\mathbf{F}, decline)$ to the training set (where $\mathbf{F} \subseteq \mathcal{F}$), depending on the evidence obtained from the interaction. The model is then constructed. 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 a particular provider has a policy that permits the provision of the resource.

4 Argumentation strategies

Having described how the policies of others can be learned with the help of evidence derived from argumentation and further refined by reasoning with relevant domain knowledge, we demonstrate the use of such structures in developing argumentation strategies for deciding which agent to negotiate with and what arguments to put forward. Our model takes into account communication cost and the benefit to be derived from fulfilling a task. Consumer agents attempt to complete tasks by approaching the most promising provider. Here, we formalise the decision model developed for this aim; a model that we empirically evaluate in Section 5.

Let $\mathcal{A}$ be a society of agents. In any encounter, agents play one of two roles: consumer or provider. Let $\mathcal{R}$ be the set of resources such that $r_1, r_2, \ldots \in \mathcal{R}$ and $\mathcal{T}$ be the set of tasks such that $t_1, t_2, \ldots \in \mathcal{T}$, and, as noted above, $\mathcal{F}$ is the set of features of possible task contexts. Each consumer agent $x \in \mathcal{A}$ maintains a list of tasks $t_1, t_2, \ldots t_n \in \mathcal{T}$ and the rewards $\Omega_x^{t_1}, \Omega_x^{t_2}, \ldots \Omega_x^{t_n}$ to be received for fulfilling each corresponding task. We assume here that tasks are independent; in other words, x will receive $\Omega_x^{t_1}$ if t_1 is fulfilled irrespective of the fulfilment of any other task. Further, we assume that tasks require single resources that can each be provided by a single agent; i.e. we do not address problems related to the logical or temporal relationships among tasks or resources. Providers operate according to a set of policies that regulate their actions, and (normally) agents act according to their policies. For example, a car rental company may be prohibited from renting out a car if the customer intends to travel across a country border.

Each consumer agent $x \in \mathcal{A}$ has a function μ_x^r with signature $\mathcal{A} \times \mathcal{R} \times \mathcal{T} \times 2^{\mathcal{F}} \rightarrow \mathbb{R}$ that computes the utility gained if x acquires resource $r \in \mathcal{R}$ from provider $y \in \mathcal{A}$ in order to fulfil task $t \in \mathcal{T}$, assuming that the information revealed to y regarding the use of r is $F \subseteq \mathcal{F}$. This F will typically consist of the information features revealed to persuade y to provide r within a specific task context. (Although we focus here on resource provision, the model is equally applicable to task delegation, where we may define a function $\mu_x^t : \mathcal{A} \times \mathcal{T} \times 2^{\mathcal{F}} \rightarrow \mathbb{R}$ that computes the utility gained if y agrees to complete task t for x, assuming that the information revealed to y to persuade it to do t is $F \subseteq \mathcal{F}$.)

Generally, agents receive some utility for resourcing a task. Revealing certain private information to others about a task context, however, may incur costs. In our model, the price that an agent pays for resources they acquire from others may be influenced by the information revealed. Such influence could be positive (in which case the consumer pays less) or negative (the consumer pays more). In the first case, the

consumer gets a discount in the price of the resource. For example, a shop may give discounts on some items to consumers provided they possesses the shop's loyalty card, and to obtain a loyalty card the consumer is required to reveal certain private information such as email address, telephone, age to the shopkeeper. These details are then used to send such things as adverts, promotions or clearance sales to the consumer at a later date. In the second case, the consumer might actually pay more if, for instance, he is perceived as a high risk customer for an insurance quote. The utility for the consumer is, therefore, the reward obtained for resourcing a task minus the cost of the resource and the cost of revealing information regarding the task context.

Definition 1 ***(Resource Acquisition Utility)*** The utility gained by x in acquiring resource r from y through the revelation of information F is:

$$\mu_x(y, r, t, F) = \Omega_x^t - (\Phi_y^r + Cost_x(F, y)) \tag{5}$$

where Ω_x^t is the reward received by x for resourcing task t, Φ_y^r is the cost of acquiring r from y (given by $\Phi_y^r = \mathbf{\Phi'}_y^r \pm |\delta_{F_y}^r|$, where $\mathbf{\Phi'}_y^r$ is the original/published price of the resource, and $\delta_{F_y}^r$ is the influence on the original/published price of the resource), and $Cost_x(F, y)$ is the cost of revealing the information features contained in F to y (which we define below).

The cost of revealing information to some agent captures the idea that there is some risk in informing others of, for example, details of private plans.

Definition 2 ***(Information Cost)*** We model the cost of agent x revealing a single item of information, $f \in \mathcal{F}$, to a specific agent, $y \in \mathcal{A}$, through a function: $cost_x : \mathcal{F} \times \mathcal{A} \rightarrow \mathbb{R}$. On the basis of this function, we define the cost of revealing a set of information $F \in 2^{\mathcal{F}}$ to agent y, as the sum of the cost of each $f \in F$.

$$Cost_x(F, y) = \sum_{f \in F} cost_x(f, y) \tag{6}$$

Cost, therefore, depends on y, but not on the task/resource. This definition captures a further assumption of the model; i.e. that information costs are additive. In general, we may define a cost function $Cost'_x : 2^{\mathcal{F}} \times \mathcal{A} \rightarrow \mathbb{R}$. Such a cost function, however, will have some impact upon the strategies employed (e.g., if the cost of revealing f_j is significantly higher if f_k has already been revealed), but the fundamental ideas presented in this paper do not depend on this additive information cost assumption.

Predictions regarding the information that an agent, x, will need to reveal to y for a resource r to persuade it to make that resource available is captured in the model that x has developed of the policies of y. For example, if, through prior experience, it is predicted that a car rental company will not rent a car for a trip outside the country, revealing the fact that the destination of the trip is within the country will be necessary. Revealing the actual destination may not be necessary, but the costs incurred in each case may differ. Let $Pr(Permitted|y, r, F)$ be the probability that, according to the policies of y (as learned by x), y is permitted to provide resource r to x given the information revealed is F.[3]

[3] We adopt a simple probabilistic approach to compute the probability of a resource being available based on past experience, but there are far more sophisticated approaches to model resource availability; e.g. [7].

Predictions about the availability of resources also form part of the model of other agents; for example, the probability that there are cars for rent. Let $Pr(Avail|y, r)$ be the probability of resource r being available from agent y. These probabilities are captured in the models learned about other agents from previous encounters.

Definition 3 ***(Resource Acquisition Probability)*** A prediction of the likelihood of a resource being acquired from an agent y can be computed on the basis of predictions of the policy constraints of y and the availability of r from y:

$$Pr(Yes|y, r, F) = Pr(Permitted|y, r, F) \times Pr(Avail|y, r) \tag{7}$$

With these definitions in place, we may now model the utility that an agent may expect to acquire in approaching some other agent to resource a task.

Definition 4 ***(Expected Utility)*** The utility that an agent, x, can expect by revealing F to agent y to persuade y to provide resource r for a task t is computed as follows:

$$E(x, y, r, t, F) = \mu_x(y, r, t, F) \times Pr(Yes|y, r, F) \tag{8}$$

At this stage we again utilise the model of resource provider agents that have been learned from experience. The models learned also provide the minimal set of information that needs to be revealed to some agent y about the task context in which some resource r is to be used that maximises the likelihood of there being no policy constraint that restricts the provision of the resource in that context. This set of information depends upon the potential provider, y, the resource being requested, r, and the task context, t. If, according to our model, there is no way to convince y to provide the r in context t, then this is the empty set.

Definition 5 ***(Information Function)*** The information required for y to make available resource r in task context t according to x's model of the policies of y is a function $\lambda_x : \mathcal{A} \times \mathcal{R} \times \mathcal{T} \rightarrow 2^{\mathcal{F}}$

Now, we can characterise the optimal agent to approach for resource r, given an information function λ_x as the agent that maximises the expected utility of the encounter:

$$y_{opt} = \arg\max_{y \in \mathcal{A}} E(x, y, r, t, F) \text{ s.t. } F = \lambda_x(y, r, t) \tag{9}$$

Our aim here is to support decisions regarding which agent to approach regarding task resourcing (or equivalently task performance); an aim that is met through the identification of y_{opt}. The question remains, however, how the agent seeking a resource presents arguments to the potential provider, and what arguments to put forward. To this aim, we present argumentation strategies that focus on minimising communication overhead (i.e. reducing the number of messages exchanged between agents), minimising the information communicated (i.e. reducing the costs incurred in revealing information, thereby maximising profits), and a trade-off between minimising the messages exchanged and maximising profits. To illustrate these strategies, consider a situation in which, according to the evaluation made by x (the consumer) of y_{opt}'s (the provider's) policies, $\lambda_x(y_{opt}, r, t) = \{f_1, f_2, f_3, f_4\}$ for resource r used for task t. The costs for revealing each feature is, as described above, $cost_x(f_1, y_{opt})$, etc. Using this situation, in the following sections we discuss 3 strategies: message minimisation; profit maximisation; and combined.

4.1 Message minimisation

The rationale for the use of this first strategy is for the consumer agent, x, to resource task, t, as soon as possible. There are many situations in the real-world in which it is important for dialogue to be a brief (and to the point) as possible, particularly in time-critical situations such as emergency response. To this aim, x seeks to minimise the number of messages exchanged with potential providers required to release the required resource, r. The consumer, therefore, reveals all the information that, according to λ_x, the provider will require to release the resource in a single proposal. Since cost is incurred when information is revealed, however, this strategy will, at best, get the *baseline* utility; i.e. the utility expected if the provider indeed requires all information predicted to release the resource.

In the example above, consumer x will send $\lambda_x(y, r, t) = \{f_1, f_2, f_3, f_4\}$ to provider y in one message, and, if the request is successful, the utility gained by x will be:

$$\mu_x(y, r, t, \lambda_x(y, r, t)) = \Omega_x^t - (\Phi_y^r + Cost_x(\lambda_x(y, r, t), y)) \tag{10}$$

This strategy ensures minimal messaging overhead if the consumer has accurate models of the policy and resource availability of providers. Such strategy is certainly useful in scenarios where the agent is interested in getting a plan resourced (or task performed) with as few messages as possible with little or no consideration to the cost of revealing necessary information. Performing a task with as few messages as possible would save time and so could connote performing the task as quickly as possible (assuming that it does not take longer to pull the features into fewer messages).

4.2 Profit maximisation

The rationale for this strategy is to attempt to maximise the profit acquired in resourcing a task by attempting to reduce the information revelation costs in acquiring a resource. Using this strategy, the agent uses the models of other agents developed from past encounters to compute confidence values for each diagnostic information feature (i.e. their persuasive power). Suppose that the relative impact on a positive response from the provider in revealing features from $\lambda_x(y, r, t)$ are $f_3 > f_1$, $f_3 > f_2$, $f_1 > f_4$ and $f_2 > f_4$. Using this information, the agent will inform the potential provider of these features of the task context in successive messages according to this order when asked for justification of its request until agreement is reached (or the request fails).

In the above example, if the most persuasive justification (feature of the task context) succeeds, it will achieve an outcome of $\Omega_x^t - (\Phi_x^r + cost_x(f_3, y))$, if further justification is required either f_1 or f_2 is used, and so on.

Other strategies are, of course, possible; for example, ordering features according to the ratio of the confidence value for each diagnostic feature in $\lambda_x(y, r, t)$ and the cost of revealing that feature to y. Rather than discussing such alternatives, in the following we discuss how simple strategies can be combined.

4.3 Combined strategies

The rationale for these combined strategies is to capture the trade-off between presenting all the features of the task context in a single message, thereby, minimising

communication, and attempting to extract as much utility as possible from the encounter (in this case by utilising information regarding relative persuasive power and cost). This may be modelled as a multiple criteria decision problem with the goals of maximising persuasive power versus cost of each message and of minimising the number of messages (with appropriate weighting between these two goals). Here, we outline a couple of heuristic methods to solve this general problem. One heuristic is to set a message threshold (a limit to the number of messages sent to a potential provider), σ_m. Then the agent will try to cluster features in $\lambda_x(y, r, t)$ in σ_m messages and use these in order of reducing persuasive power/cost ratio. It is easy to see that when σ_m is set to 1 then the agent adopts the *message minimisation* strategy, and if σ_m is set to $|\lambda_x(y, r, t)|$ this is equivalent to *profit maximisation*.

Another way, is to exploit the fact that the confidence values for each feature represent the likelihood that the resource provider will need this information to make a policy-governed decision. If the confidence value of a given feature exceeds some threshold, σ_c, then that feature is included in the set of information that will be revealed first; this is the set of features most likely to persuade the provider to release the resource. If this does not succeed, the remaining features are revealed according to the profit maximisation strategy. For example, if f_3, f_2 and f_1 all exceed σ_c, these are sent in the first message, providing an outcome of $\Omega_x^t - (\Phi_y^r + Cost_x(\{f_1, f_2, f_3\}, y))$ if successful, and, if not, f_4 is used in a follow-up message.

Again, other strategies are possible such as clustering by topic (if such background information is available). Our aim here is not to exhaustively list possible strategies, but to empirically evaluate the impact of utilising information from the models of others learned from past encounters to guide decisions regarding whom to engage in dialogue and what arguments to put forward to secure the provision of a resource (or, equivalently, a commitment to act). We turn to the evaluation of our model in the following section.

5 Evaluation

As a way of evaluating the contributions of our approach in making informed decisions about who to collaborate with, based on the policies of others, we test the following hypotheses:

- *Hypothesis 1:* In relatively small domains or in situations where sufficiently large training data is available (which we refer to as a *closed world* scenario), agents that reason over domain knowledge will perform no worse than those without such reasoning.
- *Hypothesis 2:* In complex domains or in situations where limited training data is available (which we refer to as an *open world* scenario), reasoning over appropriate domain knowledge will mean that more accurate and stable models of others' policies can be derived more rapidly than without exploiting such reasoning.
- *Hypothesis 3:* Agents that build more accurate models of others and use this to drive argumentation strategy will perform better than those that do not. In other words, a combination of decision-theoretic and machine learning techniques can both significantly improve the cumulative utility of dialogical outcomes, and help to reduce communication overhead more than those that do not.

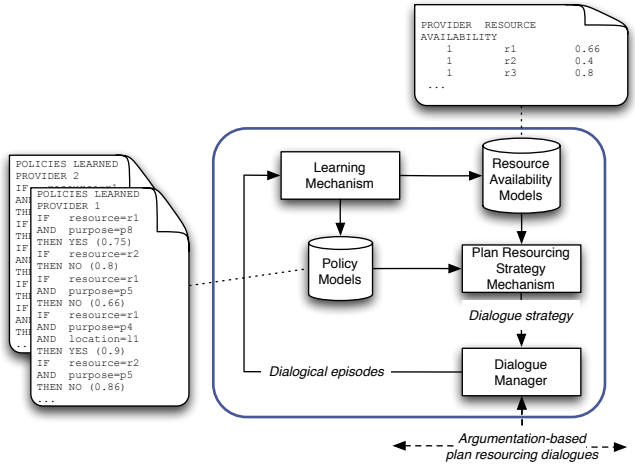


Fig. 6 Agent reasoning architecture.

5.1 Architecture

The framework we propose here (illustrated in Figure 6) enables agents to negotiate regarding resource provision, and use evidence derived from argumentation to build more accurate and stable models of others' policies. These policy models, along with models of resource availability also derived from previous encounters, are used to guide dialogical strategies for resourcing plans.

The dialogue manager handles all communication with other agents. In learning policies from previous encounters, various machine learning techniques can be employed; Figure 6 illustrates a model derived using rule learning. The arguments exchanged during dialogue constitute the evidence used to learn policies and resource availability. Arguments refer to features of the task context in which a resource is to be used, and decisions regarding whether or not a resource is made available to another agent may depend on such features. The plan resourcing strategy mechanism reasons over policy and resource availability models, and selects the potential provider with the highest expected utility (see Section 4).

5.2 Experimental setup

In evaluating our approach, we implemented an artificial society where agents acting as consumers interact with provider agents with regard to resourcing their plans over a number of runs. Each provider is assigned a set of resources, and resources are associated with some charge, Φ_r. Providers also operate under a set of policy constraints that determine under what circumstances they are permitted to provide a resource to a consumer. A task involves the consumer agent collaborating with provider agents to see how assigned tasks can be resourced. Experiments were conducted with consumer agents initialised with random models of the policies of provider agents. 100 runs were conducted in 8 rounds for each case, and tasks were randomly created during each run from the possible configurations. In the control condition (simple memorisation, SM),

Table 4 Experimental Conditions

Configuration	Description
C4.5	Decision tree algorithm using C4.5
SM	Simple memorisation of outcomes
kNN	Instance-based algorithm using k-nearest neighbour
SC	Rule learning algorithm using sequential covering
STree	Semantic-enriched decision tree learning algorithm

the consumer 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. The evaluation reported in this section is in two parts. In the first part, we demonstrate that it is possible to use evidence derived from argumentation to learn models of others' policies. Furthermore, we demonstrate that it is possible to exploit ontological reasoning to improve models of others' policies, hence increase their predictive accuracy, and performance.

To do this, we consider two experimental scenarios (i.e. *closed world* and *open world*). The *closed world* scenario refers to simple domains with relatively small numbers of terms (each feature may have one of 5 different values), while the *open world* scenario refers to more complex domains (each feature may have one of 20 different values). There are five features that are used to capture agents' policies, namely *resource type*, *affiliation*, *purpose*, *location*, and *day*. These features provide the possible task context for each agent in the system. Thus, in the *closed world* scenario, the consumer is faced with a problem domain in which there are (potentially) 3,125 individual policies for different task configurations while the *open world* scenario has 3.2M possible task configurations. We investigate the performance of agents in a number of experimental conditions. These conditions are summarised in Table 4.

In the second part of this evaluation, we utilise the two scenarios described earlier in the first part of this evaluation (that is, *closed world* and *open world* scenarios). We demonstrate that a combination of machine learning and decision theory can be used to aid agents in choosing who to collaborate with, and what information needs to be revealed in order to persuade that agent to release a resource. We consider nine experimental configurations in total. These configurations are outlined in Table 5.

5.3 Results

In this section, we present the results of the experiments carried out to validate this work. 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 $p < 0.05$.

Hypothesis 1
Figure 7 shows the performance of STree with respect to performances of SC, kNN, C4.5 and SM in terms of policy prediction accuracy, in the *closed world* scenario. For clarity, error bars are omitted from Figure 7. The figure clearly demonstrates that STree performs at least as good as SC while C4.5 always performs significantly worse than STree. Furthermore, STree outperforms SC when the number of tasks are relatively small, which leads to relatively smaller training sets. This is because STree takes advantage of reasoning about background domain knowledge and so can make informed

Table 5 Experimental Configurations

Condition	Description
SM	Simple memorisation of outcomes
SMMMS	SM + message minimising strategy
SMPMS	SM + profit maximising strategy
SC	Sequential covering- rule learning algorithm
SCMMS	SC + message minimising strategy
SCPMS	SC + profit maximising strategy
STree	Semantic-enriched decision tree learning algorithm
STreeMMS	STree + message minimising strategy
STreePMS	STree + profit maximising strategy

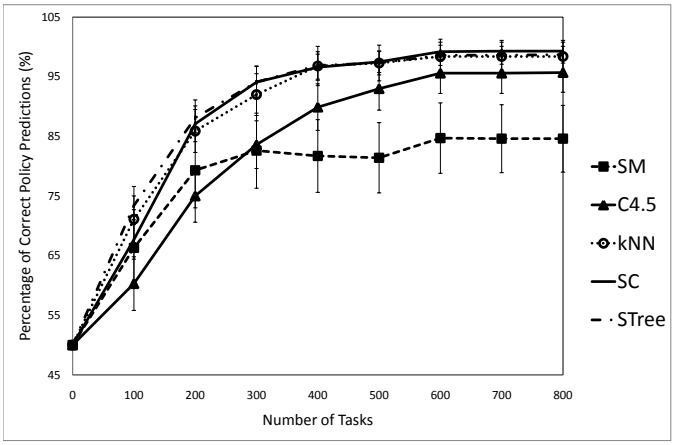


Fig. 7 Policy prediction accuracy (*closed world* scenario).

inference (or guess) with respect to feature values that do not exist in the training set. After 400 tasks the accuracy of SC reached 96% to tie with STree. We believe, at this point, a significant majority of the test instances have been encountered and so have been learned (and now exist in the training set for future episodes). Since STree is just C4.5 enhanced with ontological reasoning, we note (from the results) that agents that exploit ontological reasoning over appropriate domain knowledge can indeed perform significantly better than those that do not incorporate such reasoning (even in relatively small domains).

In order to test the statistical significance of the results of our evaluation, we carried out an analysis of variance (ANOVA), which shows whether or not several means come from the same population. More specifically, we conducted a one-way repeated measures ANOVA to compare the performance of agents in various configurations. The results were found to be statistically significant using ANOVA testing at the $p < 0.001$ level $[\mathrm{F}(6.757, 6689.446) = 1486.791, p < 0.001]$. Therefore, we conclude that there is significant change in performance of at least two configurations tested. Post-hoc comparison using a Bonferroni test indicated that all mean differences between config-

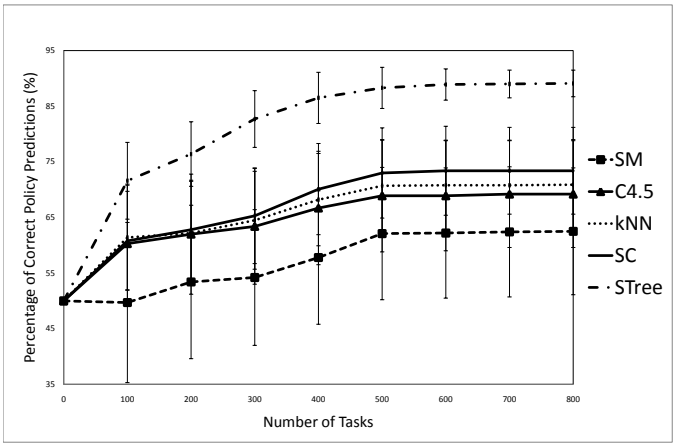
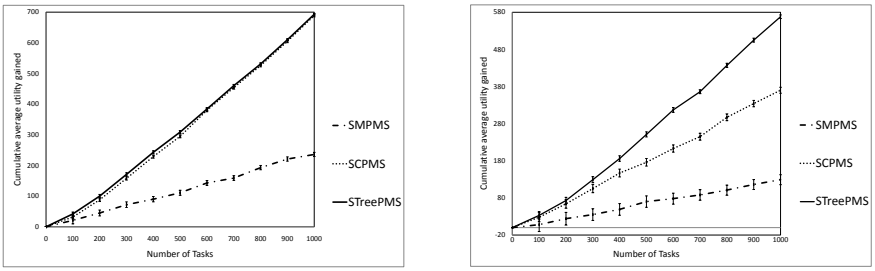


Fig. 8 Policy prediction accuracy (*open world* scenario).

urations are significant except for the following cases: STree vs SC, STree vs kNN, SC vs kNN that each records $p > 0.05$. Thus, the performance of STree is similar to SC as well as kNN in relatively small domains or in situations where sufficiently large training data is available. This supports our hypothesis that agents that exploit ontological reasoning in such situations will perform no worse than those without such reasoning.

Hypothesis 2
Figure 8 illustrates the effectiveness of four learning techniques (C4.5, kNN, SC, and STree) and SM in learning policies in the *open world* scenario. For clarity, error bars are omitted from Figure 8. The results show that the technique that exploits ontological reasoning over domain knowledge (STree) significantly outperforms the other techniques that did not. The decision trees (i.e. STree and C4.5) were pruned after each set of 100 tasks and after 300 tasks the accuracy of the STree model had exceeded 82% while those of SM, C4.5, kNN and SC were just over 54%, 63%, 64%, and 65% respectively. Furthermore, throughout the experiment, the performance of STree was significantly higher than all the other configurations.

Tests of statistical significance were applied to the results of our evaluation using a one-way ANOVA to compare the performance of various agent configurations in the *open world* scenario. The results were found to be statistically significant using ANOVA testing at the $p < 0.001$ level [$\mathrm{F}(6.884, 6895.734) = 1958.685, p < 0.001$]. Therefore, we conclude that there is significant change in performance of at least two configurations tested. Post-hoc comparison using a Bonferroni test indicated that all mean differences between configurations are significant. From the profile plots in Figure 8), we can see that predictive accuracy increases as the number of tasks increase for all configurations, with STree recording the highest. This confirms our hypothesis that exploiting ontological reasoning in complex domains or in situations where limited training data is available will mean that more accurate and stable models of others' policies can be



(i) *closed world* scenario (ii) *open world* scenario

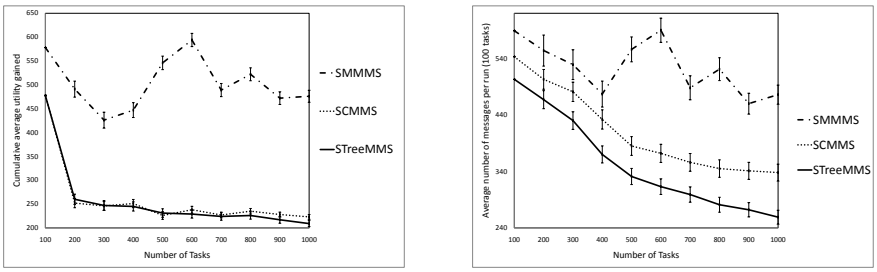
Fig. 9 Comparing the cumulative average utility gained by SMPMS, SCPMS and STreePMS in both *open world* and *closed world* scenarios.

derived more rapidly than without exploiting such reasoning.

Hypothesis 3
Here, we evaluate the effectiveness of configurations that allow agents to utilise a combination of decision-theory and machine learning in comparison with those that do not utilise such combination. In other words, we evaluate configurations in which agents build more accurate models of others (using machine learning) and develop argumentation strategies (using decision theory) on this basis and compare their performance with those in which agents utilise decision theory alone. We consider the following experimental configurations — SMPMS, SCPMS, STreePMS, SMMMS, SCMMS, and STreeMMS.

Firstly, we show the cumulative average utility gained by agents that exploit incremental revelation of information in different configurations. In the SMPMS configuration, agents utilise simple memorisation of outcomes (which involves little or no learning) and develop argumentation strategies on that basis. On the other hand, in the SCPMS and STreePMS configurations agents employ a combination of decision-theoretic model and standard machine learning techniques. More specifically, agents that use SCPMS and STreePMS configurations build models of others using rule learning and semantic-enriched decision trees respectively, and thereafter develop strategies to argue based on the models learned. Here, we compare the performance of a combination of decision theory and machine learning against decision theory alone.

In Figures 9(i) and 9(ii), results clearly show that there is a significant difference between the performance of agents that employ argumentation strategies based on a combination of machine learning and decision-theoretic models (i.e., SCPMS and STreePMS) and those that employ argumentation strategies based on simple memorisation of outcomes (SMPMS). In particular, results show that both SCPMS and STreePMS outperform SMPMS in both *closed* and *open world* scenarios. The reason for this, after detailed analysis of the data, is because the agent (in SCPMS and STreePMS) is (1) able to build more accurate models of others' policies; (2) able to make an informed decision concerning which provider to approach for a given resource; and (3) able to utilise those models to reduce cost of information revelation by incrementally revealing information based on the persuasive power. For example, after 900



(i) *closed world* scenario (ii) *open world* scenario

Fig. 10 Comparing the number of messages exchanged by SMMMS, SCMMS and STreeMMS in both *open world* and *closed world* scenarios.

tasks in the *closed world* scenario, the cumulative average utility gained by agents in the SCPMS and STreePMS configurations had each risen above $600 while the cumulative average utility gained by agents in the SMPMS configuration is below $222. We believe, the reason for the poor performance of SMPMS stems from the fact that the agent is unable to generalise from a number of examples; it only uses exact matches. This inability to build an accurate model of the policy of others reduces the effectiveness of the decision-theoretic model whereas a careful combination of decision-theoretic models and standard machine learning techniques leads to better performance. This confirms our hypothesis that a combination of machine learning and decision theory will enable agents to perform better than when there is no such combination.

Tests of statistical significance were applied to the results of our evaluation using a one-way ANOVA to compare the performance of various agent configurations that utilise incremental revelation of information in both *closed world* and *open world* scenarios. The results were found to be statistically significant using ANOVA testing at the $p < 0.001$ level $[\mathrm{F}(8.768, 5697.825) = 1987.968, p < 0.001]$. Therefore, we conclude that there is significant change in performance of at least two configurations tested. Post-hoc comparison using a Bonferroni test indicated that all mean differences between configurations are significant except for the following cases: STree vs SC in the *closed world* scenario. From the profile plots (see Figure 9), we can see that cumulative utility increases as the number of tasks increase for all configurations, with STreePMS and SCPMS (in the *closed world* scenario) recording the highest. In the *open world* scenario, STreePMS clearly outperformed all the other cobnfigurations. This supports our hypothesis that agents that build more accurate models of others and develop their argumentation strategies on this basis will perform better than those that do not.

Secondly, we show the number of messages exchanged by agents that anticipate the information needs of others. In SCMMS and STreeMMS configurations, agents build models of others using rule learning and semantic-enriched decision trees respectively, and thereafter develop strategies to argue based on the models learned whereas in SMMMS configuration agents rely on simple memorisation of outcomes in forming argumentation strategies. Here, we analyse the performance of agents that employ a combination of decision theory and machine learning against those that utilise decision theory alone. In Figures 10(i) and 10(ii), results clearly show that there is a significant difference between the performance of agents in SCMMS and STreeMMS configura-

tions and those that employ argumentation strategies based on simple memorisation of outcomes (SMMMS). The reason for this, after detailed analysis of the data, is similar to the reasons discussed earlier that account for the difference between SCPMS, STreePMS and SMPMS in the first part of this evaluation. That is, the agent that utilises machine learning and decision theory (e.g., SCMMS, STreeMMS) is able to build more accurate models of others' policies, preempt the information requirements of the provider and thereby present it without having to be asked. For example, after 200 tasks in the *closed world* scenario, the number of messages exchanged per 100 tasks by agents in the SMMMS configuration is more than twice the number of messages exchanged by agents using SCMMS and STreeMMS configurations respectively. This, we believe, is because the agent is unable to generalise from a number of examples since it only uses exact matches from previous encounters, which reduces the performance of the decision-theoretic model built from that. This confirms our hypothesis that a combination of machine learning and decision theory will enable agents perform better than when there is no such combination.

Tests of statistical analysis showed that the results were statistically significant at the $p < 0.001$ level [$F(8.743, 5778.384) = 1979.663, p < 0.001$]. Therefore, we conclude that there is significant change in performance of at least two configurations tested. Post-hoc comparison using a Bonferroni test indicated that all pairwise comparison of mean differences between configurations are significant except for the following cases: STreeMMS vs SCMMS in the *closed world* scenario. From the profile plots (see Figure 10), we can see that communication overhead reduces as the number of tasks increase for all configurations, with STreeMMS and SCMMS (in the *closed world*) recording the lowest, followed by STreeMMS (in the *open world*). This supports our hypothesis that agents that build more accurate models of others and develop their argumentation strategies on this basis will perform better than those that do not.

6 Discussion

We started with the question "What do I need to say to convince you to do something?", and have presented and evaluated a model that starts to address this multi-faceted question. The approach combines argumentation, machine learning and decision theory to learn underlying social characteristics (e.g., policies or norms) of others and exploit these models to reduce communication overhead and improve strategic outcomes. Furthermore, we describe how reasoning about domain knowledge can be leveraged while learning policy models of others. We empirically show that employing ontological reasoning over domain knowledge significantly improves policy learning performance especially in complex domains with large number of concepts and instances.

The use of domain knowledge to enhance the performance of algorithms to develop classifiers is not new [21], but the STree algorithm uses a novel approach to combining these reasoning techniques. We extended C4.5 decision trees with ontological reasoning to leverage domain knowledge during policy learning. Zhang and Honavar have also extended C4.5 decision trees with Attribute-value taxonomies [36]. Their approach is similar to STree, but it does not allow ontological reasoning during tree induction. In contrast, our approach can directly incorporate existing domain ontologies and exploit these ontologies during policy learning. In the biological domain, Hirsh and Noordewier [12] showed that DNA sequence learning is possible with significantly lower error rates by using background knowledge of molecular biology and re-expressing data

in terms of higher-level features. In their work, these high level features are used by C4.5 decision trees and neural networks. Ambrosino and Buchanan [2] showed that the addition of domain knowledge improves the learning of a rule induction program for predicting the risk of mortality in patients with community-acquired pneumonia. Kopanas *et al.* [17] examined the role of domain knowledge using a case study of customer insolvency in the telecommunications industry. They showed that domain knowledge plays a critical role mainly in all phases of data mining. As we have shown here, exploiting domain knowledge also provides significant benefits in building models of behavioural expectation (in our case norms or policies for resource sharing) from observed behaviour (in our case dialogue).

We believe that our research also contributes to both to the understanding of argumentation strategy, and to applications of these techniques in agent support for human decision-making. Sycara et al. [33] report on a study into how software agents can effectively support human teams in complex collaborative planning activities. One area of support that was identified as important in this context is guidance in making policy-compliant decisions. This prior research focuses on giving guidance to humans regarding their own policies. An important and open question, however, is how can agents support humans in developing models of others' policies and using these in decision making? Our work seeks to bridge (part of) this gap. Furthermore, as demonstrated by Núñez [21], the use of domain knowledge improves the logical structure of decision trees, and hence we expect, would improve explanations of policy predictions given to decision-makers derived from the classifier. This, however, is an open question for future research.

There a number of other avenues for future research that can be identified. Here, we assume that the agent seeking to resource its plan makes a single decision per task about which provider to negotiate with; i.e. it has one go at resourcing a task. In reality, such a decision process should be iterative; i.e. if the most promising candidate fails to provide the resource, the next most promising is approached and the sunk cost incurred is taken into consideration, and so on. In this paper, we have explored dialogue strategies for an individual agent modelling another. We have not explored the case in which both parties attempt to (competatively) model each other from dialogical encounters and exploit these in future interactions. Further, we do not address the question of deception, including strategic choices made by an agent to influence the formation of models of its policies/norms.

7 Conclusions

We have presented an agent decision-making mechanism where models of other agents are refined through evidence from past dialogues and domain knowledge, and where these models are used to guide future delegation decisions. Furthermore, we have empirically evaluated our approach and the results of our investigations show that decision-theoretic and machine learning techniques can individually and in combination significantly improve the cumulative utility of dialogical outcomes. In addition, we have shown that domain knowledge is useful in improving the utility of dialogical outcomes. The results also demonstrate that this combination of techniques can help in developing more robust and adaptive strategies for advising human decision makers on how a plan may be resourced (or a task delegated), who to talk to, and what arguments are most persuasive.

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. A. Aamodt and E. Plaza. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, 7:39–59, 1994.
2. R. Ambrosino and B. Buchanan. The use of physician domain knowledge to improve the learning of rule-based models for decision-support. In *Proceedings of the American Medical Informatics Association Symposium*, pages 192–196, 1999.
3. G. Antoniou and F. van Harmelen. *A Semantic Web Primer, 2nd Edition (Cooperative Information Systems)*. The MIT Press, 2008.
4. L. Breiman. *Classification and regression trees*. Chapman & Hall, 1984.
5. R. Duda and P. Hart. *Pattern classification and scene analysis*. John Wiley & Sons, 1973.
6. X. Fan and J. Yen. Modeling and simulating human teamwork behaviors using intelligent agents. *Physics of Life Reviews*, 1(3):173 – 201, 2004.
7. M. Finger, G. C. Bezerra, and D. R. Conde. Resource use pattern analysis for predicting resource availability in opportunistic grids. *Concurrency and Computation: Practice and Experience*, 22(3):295–313, 2010.
8. E. Frank, Y. Wang, S. Inglis, G. Holmes, and I. H. Witten. Using model trees for classification. *Machine Learning*, 32:63–76, 1998. 10.1023/A:1007421302149.
9. B. Grosz and S. Kraus. Collaborative plans for group activities. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 367–373, 1993.
10. W. O. W. Group. OWL 2 web ontology language: Document overview, October 2009. http://www.w3.org/TR/owl2-overview.
11. J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques*. Morgan Kaufmann, 2nd edition, 2006.
12. H. Hirsh and M. Noordewier. Using background knowledge to improve inductive learning: A case study in molecular biology. *IEEE Expert: Intelligent Systems and Their Applications*, 9:3–6, 1994.
13. P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall, 2009.
14. T. Huynh, N. R. Jennings, and N. Shadbolt. An integrated trust and reputation model for open multi-agent systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 13(2):119–154, 2006.
15. J. Huysmans, R. Setiono, B. Baesens, and J. Vanthienen. Minerva: Sequential covering for rule extraction. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 38(2):299 –309, 2008.
16. D. Kalles and T. Morris. Efficient incremental induction of decision trees. *Machine Learning*, 24:231–242, 1996.
17. I. Kopanas, N. Avouris, and S. Daskalaki. The role of domain knowledge in a large scale data mining project. *Methods and Applications of Artificial Intelligence*, pages 746–746, 2002.
18. P. McBurney and S. Parsons. Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information*, 12(2):315 – 334, 2002.
19. T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.
20. T. J. Norman and C. Reed. A logic of delegation. *Artificial Intelligence*, 174(1):51–71, 2010.

21. M. Núñez. The use of background knowledge in decision tree induction. *Machine Learning*, 6:231–250, 1991.
22. J. Pearl. *Causality: Modeling, Reasoning, and Inference*. Cambridge University Press, 2000.
23. J. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
24. J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, 1993.
25. I. Rahwan, L. Sonenberg, and F. Dignum. Towards interest-based negotiation. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 773–780, 2003.
26. S. D. Ramchurn. *Multi-agent negotiation using trust and persuasion*. PhD thesis, ECS, University of Southampton, 2004.
27. P. Rathod and M. des Jardins. Stable team formation among self-interested agents. In *AAAI Workshop on Forming and Maintaing Coalitions in Adaptive Multiagent Systems*, 2004.
28. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2ed edition, 2003.
29. M. Sensoy, T. J. Norman, W. W. Vasconcelos, and K. Sycara. OWL-POLAR: Semantic policies for agent reasoning. In *Proceedings of the Ninth International Semantic Web Conference*, volume 6414 of *Lecture Notes in Computer Science*, pages 679–695. Springer-Verlag, 2010.
30. M. Sensoy, J. Zhang, P. Yolum, and R. Cohen. Context-aware service selection under deception. *Computational Intelligence*, 25(4):335–364, 2009.
31. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Web Semant.*, 5(2):51–53, 2007.
32. G. Sukthankar and K. Sycara. Analyzing team decision-making in tactical scenarios. *The Computer Journal*, 53(5):503–512, 2010.
33. K. Sycara, T. J. Norman, J. A. Giampapa, M. J. Kollingbaum, C. Burnett, D. Masato, M. McCallum, and M. H. Strub. Agent support for policy-driven collaborative mission planning. *The Computer Journal*, 53(5):528–540, 2010.
34. W. Teacy, J. Patel, N. Jennings, and M. Luck. Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, 12(2):183–198, 2006.
35. I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2nd edition, 2005.
36. 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.