What is Going on: Utility-based Plan Selection in BDI Agents

Ameneh Deljoo
Institute of Informatics
University of Amsterdam
Amsterdam, Netherlands
a.deljoo@uva.nl

Tom van Engers
Leibniz Center for Law
University of Amsterdam
Amsterdam, Netherlands
t.m.vanEngers@uva.nl

Leon Gommans
Air France-KLM
Amsterdam,The Netherlands
leon.gommans@KLM.com

Cees de Laat
Institute of Informatics
University of Amsterdam
Amsterdam, Netherlands
delaat@uva.nl

Abstract

This work addresses the problem of choosing an appropriate plan for achieving a goal in any realistic complex situation where an agent has to respond and act upon uncertain and/or an unknown information. We use the belief-desire-intention (BDI) model, a popular model for developing agents. The flexibility of choosing among different plans to achieve a desired goal is one of the benefits of this model. This paper describes a particular algorithm for selecting the most appropriate plan. Since the agent may have to reason with incomplete or uncertain information, we explore how to integrate probabilities in the agent model for taking an appropriate action and keeping the system behavior within acceptable boundaries and compliance to acceptable norms. Considering the uncertainty of the current state of the environment, this process relies on probability and utility theory. The plan selection algorithm has been implemented with Jadex.

1 Introduction

In this paper, we propose to integrate the probability and the utility into the BDI agent model of Rao and Georgeff (Rao and Georgeff 1995). In order to perform this task, we designed an algorithm that given possible plans selects the most appropriate plan to keep the system within the boundaries that is determined by the applicable norms. This research focuses on a domain of collaborating service providers (*SPs*) (Gommans et al. 2015). *SPs* in our case are part of an organizational network (in this research a distributed network) they are also bound by the rules that define this collaborative network. This distributed network highlights some prerequisites, such as 1- needs for acting in an open, dynamic and unpredictable environment, 2- necessities of behaving according to the rules ($SP's$ rules) and 3- demands for providing the appropriate plan, which responds to the current state of the environment (Abdelkader 2003; Giorgini, Mylopoulos, and Sebastiani 2005).
Previously, (Deljoo et al. 2016) presented the elements of an application framework based on normative reasoning.
Agent Based Modeling (ABM) has been exploited to model open systems where agents are self-governed autonomous entities and pursue their individual goals based only on their beliefs and capabilities (Abdelkader 2003).

On the basis of our experience with open systems, we have identified an essential requirement for an agent model, which is a well-developed model of decision making under uncertainty, which is regarded as the basis for dealing with the complexities of open systems. In open systems decisions may involve high levels of uncertainty about the true state of the environment and agents may not be able to fully understand the consequences of actions on the environment.
Moreover, in a fully distributed system, there is no central control mechanism that can observe every agent's behavior regarding enforced conventions and norms into the agents. Therefore agents have to be able to cope with the other agent's behaviors. Due to the number of observations and received information, taking an appropriate action in the open system is much more complex compared to closed systems.
In this paper, we will introduce a plan selection algorithm that can cope with such uncertain situations. Choosing an appropriate plan based on received information about the current environment's state is the main contribution of the work presented in this paper. We developed an agent plan selection that can reason with probability and utility. Both probability and utility are used to choose a preferred plan from the set of possible plans, by weighing them towards achieving a particular goal. Depending on circumstances, the probability may not always result in an achievable plan for realizing the goal, but it is a prerequisite for selection the most applicable plans. Therefore, we have extended the architecture of the agent to address this requirement. First we will answer the following question, how should the plan selection of agent select the plans that fit the agent's context?
The contributions of this work are as follows:

- An extension of the BDI agent model to allow the representation of the concepts needed for agents, which is able to choose plans based on the utility and the probability.
- A simple but effective control loop algorithm to choose proper plans by applying the concepts of our extended BDI agent model.

In the next section, we review some essential literature about game theory and utility in ABM. In the section 3, we introduce an example to illustrate our scenario. The central part of the contribution is in sections 4 and 5, where the implementation of integrated probabilities in our BDI agent model with a set of concepts is discussed. The paper ends with dis-

cussion and further developments.

2 Related Work

The BDI agent (Rao and Georgeff 1995) is a popular model for developing agents and is supported by different methods and techniquess(Meneguzzi and De Silva 2015; Giorgini, Mylopoulos, and Sebastiani 2005) and platforms (Bordini, Hübner, and Wooldridge 2007; Busetta et al. 1999; Nunes, Lucena, and Luck 2011; Pokahr, Braubach, and Lamersdorf 2005).

Researchers have proposed plan selection techniques (e.g., (Dasgupta and Ghose 2010; Visser, Thangarajah, and Harland 2011)), and other methods to customize different parts of BDI agents model, such as belief revision and goal generation (e.g., (Van Benthem 2007; Dietrich, List, and Bradley 2016)). (Singh et al. 2010) proposed a novel BDI programming framework that, by suitably modeling context conditions as decision trees, allows agents to learn the probability of success for plans based on previous execution experiences. Nevertheless, their direct use in open systems is still problematic since in such open system each agent aims to use a particular strategy th at maximizes its own welfare, which is not necessarily maximize the overall network's performance.

In the last few years, there has been an increasing interest in the use of techniques from decision theory and game theory for analyzing and implementing agents. In applications of game theory in multi-agent systems (MASs), agents are usually taken to maximize their expected utilities. This approach, however, is not always practical, as there are often boundaries on computational resources, which prevent the optimal solution being computed. As a result, there has been much interest in computing solutions under bounded rationality. This means that we aim to find a solution, maximizing the expected utility but limiting the required resources.

One sub-area of decision theory is the field of Markov decision processes (MDP), discussed in detail in (Boutilier, Dean, and Hanks 1999). In essence, an MDP is an iterative set of classical decision problems. Consider a state of the world as a node in a graph. Carrying out an action in that state will result in a transition to one of a number of states, each connected to the first state by an arc, with some probabilities, and some cost. After a series of transitions a goal state may be reached, and the sequence of actions that should be executed to reach that goal is known as a policy. MDPs aim to find a minimal cost policy for moving from some initial state to a goal state and capture many of the facets of real-world problems, but unrealistically assume that whatever system is solving the MDP knows at every point what state it is in. MDPs are not applicable in open domain systems because the network changes dynamically, and therefore the current state of the environment may be (partly) unknown to the agents. Indeed, in open systems the network changes constantly (an agent may join or leave the network) therefore, agents behavoir may change the network. In that sence, we need to apply an intraction framing algorithm (Rovatsos 2001) to frame the agent's behavior, which is not addressed in the MDP. Partially observable Markov decision process (POMDP) is a sub-branch of MPDs, which applied by a Robotic navigation and Intelligent Control (Liu et al. 2016). Although POMDPs have been known for many decades, they are scarcely used in practice. POMDPs are computationally very expensive and thus applicable in practice only to very simple problems (Hauskrecht 2011). This is mainly due to two major difficulties 1) obtaining the environment dynamics and 2) solving the resulting model (Shani 2007). For these reasons, POMDP is not the appropriate method for our domain.

In contrast, Lang, van der Torre and Weydert (Lang, Van Der Torre, and Weydert 2002) combined a logic-based method and decision theory to consider how an agent might reason about its goals and use them to define its utility. Recently, some approaches have been introduced based on whether the plan and preference formalisms are ordinal, qualitative or quantitative. They model temporal preferences or solely static preferences; and the formalism is propositional or first-order. Indeed, it induces a total order and the degree of incomparability in the ordering (Meneguzzi and De Silva 2015; Visser, Thangarajah, and Harland 2011). These approaches extended a language for preference based planning, without considering the expected utility for each plan.

3 Illustration

In this section, we present an example that is used subsequently to explicate our approach. In a distributed, open and dynamic environment with multiple administrative domains, it is a challenge for agents to take an appropriate action due to the number of observations.

The scenario is as follows: Bob is a security manager at company A. For the sake of his company, he is looking for a way to collaborate with Alice, who is a security manager at company B. Alice and Bob are not part of a collaborative group (i.e., *SPs*). To establish this collaboration, each agent needs to plan its actions based on the estimated risks and benefits, which means maximizing the benefits while minimizing the risks.

Bob has three options. The options are:

Plan A Give complete access over the company's data to Alice;

Plan B Request certification from her company;

Plan C Deny Alice's request.

In this scenario, the goal for agent Bob is "Sharing with Alice" and sub-goals are "calculate risks" and "estimate Benefits". Although all these plans are a way of achieving the goal of "Sharing with Alice" or sub-goals, each plan has different characteristics. For example, suppose that Bob and Alice have not yet collaborated before, then Bob would take a risk if he would select plan (A), as he can not be sure about the trustworthiness of Alice. Whereas if he chooses Plan (C) Bob knows right away that he will not be able to gain benefits of this collaboration. One may think that selecting plan (B) is the most appropriate plan for this scenario. However, each plan is associated with a particular response time and requires a different amount of work; e.g., requesting a certification from the company implies completing many processes.

This scenario exposes the following problem: how can agent Bob select the most appropriate plan to achieve its goal based on its current state, the utility of the selected action and act upon the unknown information about the trustworthiness of Alice?
In the following, we provide a brief introduction to the area of decision theory, game theory, and we formalize our contribution in this area.

Decision Theory in Agent systems

A classical decision theory is a set of mathematical techniques for making decisions about what action to take when the outcomes of the various actions are not known. This theory can usefully apply in agent based theory (i.e., such agents are canonical examples of the decision makers). An agent operating in a complex environment is inherently uncertain about that environment; it simply does not have enough information about the environment to know either the precise current state of its environment nor how that environment will evolve. Thus, for every variable S, which captures some aspect of the current state of the environment, all the agent typically knows is that each possible value has some probabilities $Pr(S)$. Writing S for the set of all S reads:

$$Pr : S \in [0,1] \tag{1}$$

and

$$Pr(S_1) + Pr(S_2) + Pr(S_3) + ... + Pr(S_n) = 1 \tag{2}$$

Expected Utility

As we presented in the earlier described scenario, agent Bob has a set of possible plans to select. To choose an appropriate one, Bob has three alternatives:

1. Select the first plan on the list and execute that plan, which is sharing everything with Alice. And, taking this plan will put the organization at risk.
2. Select the cheapest plan (i.e., the most efficient plan) regarding resource consumption. In this case, Bob takes plan (C) to reduce the resource and will not gain benefits from this collaboration.
3. Select the best outcome (i.e., the most effective plan).

To solve this problem and avoid the risk, we use a utility function to combine all the alternatives and select the best alternative regarding this case study. By applying the utility function, we aim to propose an approach to formulate the appropriate plan selection algorithm in a way that is applicable in practice and theory (Parsons and Wooldridge 2002). A utility represents the value that the agent places on that state of the world (or environment). It also provides a convenient means of encoding the preferences of the agent (Von Neumann and Morgenstern 2007).

$$EU(P) = \sum_{S_i \in S} Pr(S_i \mid P) U(S_i) \tag{3}$$

where S is the set of all states. Then, the agent selects the plan with:

$$P^* = arg\max_{p \in P} \sum_{S_i \in S} Pr(S_i \mid P) U(S_i). \tag{4}$$

As we mentioned before, our aim is to extend an existing BDI agent model by integrating the utility and the probability in the BDI control loop algorithm and redesign the BDI planner component. To do so, first, we illustrate a current BDI agent model and its decision loop algorithm and then, look at the agent planner in detail and apply the probability and the utility in the extension of our BDI agent model.

4 Processing Agent Models

In this section, we introduce the agent definition and the classic BDI agent model.
Definition 1 (Agent). Agent Bob is a tuple

$$\{O, B, G, P, A_p\}, \tag{5}$$

where:

- *O is a set of the observations made by the agent.*
- *B is the agent beliefs (including beliefs about other agents and beliefs about the current state of the environment);*
- *G is the set of goals of the agent (e.g., the agent Bob's goal is to share information with Alice);*
- *P is a set of current plans (a new plan or the remaining part of the plan that Bob is currently executing). P is selected from the set of candidate plans produced by the planner;*
- *A_p is the set of actions performed when the plan is executed.*

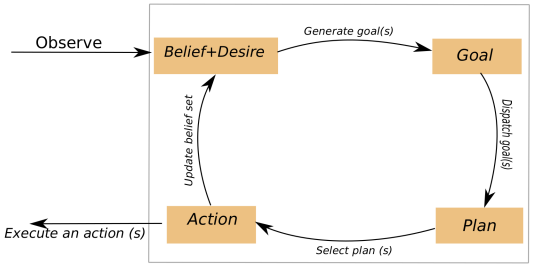

Figure 1: A classical BDI agent model.

In the classic BDI agent model, observations trigger a belief set revision stage. Candidate goal sets are generated based on primitive mental attitudes such as beliefs and desires. This results in many different sets of mutually consistent goals. The goal selection stage imposes an ordering on the sets of generated candidate goal sets, such that the best one will select for planning. This goal set is then passed on to the planning stage to construct a plan set. Finally, the selected plans trigger the action stage. The agent executes the selected actions to achieve the goal set.
Figure 1 depicts the classic BDI framework (where goals are equated with desires and plans with intentions). The deliberation cycle of the BDI agent model is presented in Algorithm 1.

Algorithm 1: Control loop for the classic BDI agent.

Given an agent $\{O,B,G,P,A_p\}$
repeat
  $O := Observe(O)$;
  $B := Revise(B,O)$;
  $G := Generate\ G_g\ (B)$;
  $P := \forall g \in G \rightarrow generate\ P(B,G)$;
  $take\ (A_p)$;
  $revise(B)$;
until *forever*;

5 Extended BDI Agent Models

We presented the BDI extension architecture in this section with two determination cycles. As said before, our goal is to build agents that are able to: (i) use the probability and utility in their planner components; (ii) represent plan utility over sub-plans, and (iii) use this information to choose plans. For the purposes of this paper, we shall mostly focus on the plan selection.
Our goal consists of the following parts:

- describes an agent architecture based on the extension of the classical BDI model by (Rao and Georgeff 1995) in a certain state *(current state = i)*;
- an algorithm to process a model based on our extended BDI agent model and to select plans;
- a modified control loop for the extended BDI agent model.

Definition 2 (Agent at the current state *i*). The agent Bob's tuple in the current state *i* is

$$\{O_i,B_i,G_i,P_i,A_{p_i}\} \in \{O,B,G,P,A_p\} \tag{6}$$

where:

- O_i *is a set of the observations made by the agent in the current state i .*
- B_i *is the agent beliefs in the current state i;*
- G_i *is the set of goals in the current state i;*
- P_i *is a set of plans in the current state i ;*
- A_{p_i} *is the set of actions in the current state i when the plan is executed.*

Figure 2 shows the extended BDI model. It builds around the three core entities of the original BDI model, which can itself be seen as an extension of the BDI agent model. In our terminology, beliefs encode the agents knowledge about the world or its mental states, which the agent holds to be true (that is, the agent will act upon them while they continue to hold). Goals are equated with "desires", and plans with "intentions". We view intentions as commitments to new beliefs or to carrying out certain plans or pursuing new goals and actions in the future.
As stated above, the agent has a set of plans, where each is primarily characterized by the goals and a set of possible actions. In other words, each plan consists of an invocation which is the event that the plan responds to and may contribute to a (sub)goal, and is characterized by a set of contributions to (sub)goal. We present a contribution value as a

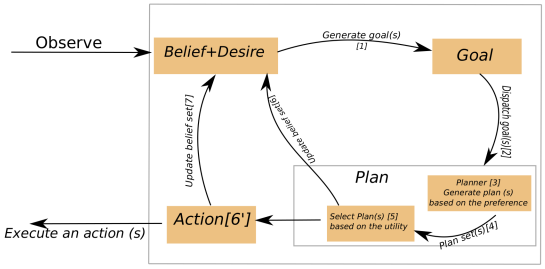

Figure 2: The extended BDI agent: considering utility in the planner component. *6 and 6'* are executed simultaneously.

number between 0 and 1 (the worst and the best contribution)[1].
The set P_i is produced by the planner starting from the initial state S_i, and inspecting A_{p_i} to find all the action recipes which have among their effects a goal in G_i. Then, by the utility function, the possible alternatives are examined, and the best P_i will be chosen, which becomes the current intention of the agent. The best plan is the one that maximizes the utility (i.e., equation 4).
Definition 3 (Plan). A plan $p_i \in P_i$

$$\{p_i,A_{p_i},Contributionvalue\} \tag{7}$$

where p_i *are a set of (sub)plans, Contribution value of* p_i *to a goal* G_i *and* $\{A_{p_i} \in Ap\}$ *is a set of actions performed when the plan is executed.*

Plan Selection

As we discussed in the introduction, we aim to support the development of agents with the ability to choose plans under uncertain states when they have to keep the system within boundaries. We now explore the integration of the probability and the expected utility of the deliberation process of our BDI agent. Each plan has different probabilities of successfully achieving the intended effect of the plan (i.e., the goal associated with plan), which are known by the agent. And, each goal and state are associated with the plan's utility, which is a weighted average of all possible (sub)plans according to their probabilities. Therefore, the plan's utility of a plan p_i with respect to the current state of the agent (S_i) is given by:
Definition 4 (Plan Utility).

$$PU(P_i) = \sum_{p_i \in P_i} Pr(S_i|p_i) \times U(S_i) \tag{8}$$

and

$$U(S_i) = Pr(S_i) \times Contributionvalue(G_i,S_i) \tag{9}$$

where $Pr \in [0,1]$ *is the probability of a plan promote the contribution value* $Contributionvalue \in [0,1]$ *with respect to the (sub)goal, 0 and 1 being the lowest and highest possible contributions, respectively.*

[1]The data for the contribution value adopted from (Nunes and Luck 2014) for each goal.

Definition 5 (Plan Expected Utility Preference).

$$Pref(P_i, S_i) = argmax \sum_{p_i \in P_i} PU(P_i, S_i) \quad (10)$$

Agent Bob takes the plan, which maximizes Bob's utility and in this work we present it as the preferred plan based upon expected utility. Such preferences express the trade-off between different plans. Suppose Bob needs to share information with Alice for an important project, and he wants to receive a quick response, secure collaboration, with minimum risk and high benefits. Preferences for plans that reflect this situation may be "ask for the certification" leads to a secure collaboration. The definition of preferences above completes the description of our model.

Based on the plan's utility, we select the plan with the highest utility of a set of possible plans to achieve a goal. This approach is implemented by algorithm 2; the algorithm has linear complexity and is simple as well as effective.

Note that more than one plan may have the same plan utility, and in this case, a plan is selected randomly from those with the maximum utility. In algorithm 2, we select the first processed plan. Thus, the plan options in the modified interpreter loop are possible sets of plans to be intended simultaneously. The deliberation step then selects between these sets of plans on the basis of the utility. In this work, we assume that each plan has a unique utility number.

We summarize our method in the following steps:

1. A planner receives the current state S_i where $S_i \in S$ and produces the states $S_1, S_2, ..., S_i$; i is the number of different recipes for G_i.
2. For each state we generate the probability value $Pr \in [0,1]$, which is assigned to $S_1, S_2, ..., S_i$.
3. The utility function applies to these states and the preferred plan $Pref_P$ regarding that states is chosen.

Using the rationale described above, each of the plans has different contributions (probability and value) to (sub-)goal, as detailed in Table1. For example, the probability of a given over access is 0.35. Therefore, if the Overall access plan is chosen, the contribution of this plan with respect to estimate benefits is 0.06 and start to share plan probability 0.65, and 0.0. Suppose, we have three plans $Plan(A)$, $Plan(B)$ and $Plan(C)$ having the respective probabilities of $Pr_{pA} = 1.0$, $Pr_{pB} = 0.2$ and $Pr_{pC} = 0.4$, and with the respective contribution's values $val_A = 0.06$, $val_B = 0.8$ and $val_C = 0.05$. This will result in utility $U(S_1, P(A)) = 0.021$, $U(S_1, P(B)) = 0.12$ and $U(S_1, P(C)) = 0.02$. The agent prefers to choose the plan with the highest utility. Thus, the expected utility of the plan p_i for the agent is given by equation (10). The agent will choose the plan that has the highest rank. In our example that is Plan (B).

In the generation of plans in the modified interpreter loop, there are two places where choices are made: the plan generator (planner) and the plan selector. The "planner" selects some subset of "high ranked" plans based on the utility and updates the belief set. In the following, the "plan selector" selects the best plan to fulfill the goal and determines the alternative courses of action which should be used to respond to these events. Bob's preference is plan B, and so he asks for Alice's certification. So the course of actions for this plan would be to check the certification, calculate risk and benefits and allow access for Alice. In the extended BDI model, we also consider agents preferences.

Algorithm 2: Select Plan

input : (sub)Goal, Set of plans ($p_i \in P_i$), the Probability of each plan
output: Selected P_i, Plan that has the best utility.

$SelectedPlan(P_i) := null$;
for $p_i \in P_i$ **do**
  $U(p_i) := Pr(p_i) \times U(s_i)$;
  $PU(P_i) := setofPU(p_i)$;
end
$PrefP_i := argmaxPU(P_i)$;
$SelectedPlan(P_i) := PrefP_i$;
return $SelectedPlan(P_i)$

Algorithm 3: Modified control loop for the extended BDI agent, (1-6) are referring to Figure 2. In the extended BDI model, *6 and 6'* are executed simultaneously. *i* is the current state of the agent.

Given an agent $\{O_i, B_i, G_i, P_i, A_{p_i}\}$
repeat
  $O_i := Observe(O_i)$;
  $B := Revise(B, O)$;
  $(1) G_i := Generate\ G_g\ (B)$;
  $(2) P_i := \forall g \in G \rightarrow generate\ P_i(B_i, G_i)$;
  $(3,4) P_i := Calculte\ U_P \forall\ p_i \in P(B_i, G_i, P_i)$;
  $(4,5) PrefP_i := Update\ P\ to\ PrefP_i(B_i, G_i, A_{p_i}, P_i)$;
  $(6,6') B_i := revise(B_{i-1}, PrefP_i)$;
  $(6') take\ (A_{p_i})$;
  $i := i + 1$;
until *forever*;

6 Experiment Settings

Our experiment consists of a simulation to compare the accumulated satisfaction of an agent after executing a plan to achieve a goal, when using our approach to select the plan and when selecting it randomly from a set of possible plans. The satisfaction of an agent is calculated based on how goals are satisfied, i.e., the contribution of each plan (probability and value). Our experiment consists of running a number of iterations in which we perform the following steps. $Step1$: Probability for each event randomly generated number in the interval $[0,1]$. $Step2$: Instantiate ascribed scenario for each plan, according to the given probability of events. $Step3$: Compute the utility for each plan. And, select a plan in three different situations:

Table 1: Plans and sub-plans Contributions value and probabilities.

	Plans and sub-plans	Probabilities $Pr \in [0,1]$	Contribution value $val \in [0,1]$
Plan A	Give overall access	0.35	0.06
	Start to share data	0.65	0.0
Plan B	Request a certification	0.95	0.08
	Check the certification	0.05	1.0
Plan C	Deny Alice's request	0.40	0.05
	Use the resources for own purpose	0.60	0.0

- When the agent selects our algorithm to assign utility for each plan.
- The agent selects a plan randomly from the list of plans.
- The agent always selects the same plan over and over again (constant plan selector).

And, we store the satisfaction of the scenario associated with the selected plan. In our experiment, we ran 1000 iterations of the steps described above, each of which takes less than 1 second to run. As result, we compared the average satisfaction and the accumulated satisfaction of all iterations for each plan selector (random and utility-based). Moreover, we also analyzed constant plan selectors: those that always choose the same plan. The average satisfaction obtained, the standard deviation and minimum and maximum values, and accumulated satisfaction are detailed in Table 2. In Table 2, the highest values are in bold and the lowest values are in italics. As can be seen in Table 2 the plan selector with

Table 2: Satisfaction by Plan Selector (n = 1000). Ask for a Certification (AskCTA) and Share everything are based on the utility plan selection algorithm. Deny plan is the constant plan that agent chooses as a current plan without considering the utility.

Plans	M	SDV	Min	Max
Randomly	*0.38*	1.59	0.0001	0.44
AskCTA	**0.93**	*0.54*	0.0001	0.98
Share everything	0.41	0.72	0.0002	0.21
Deny	0.53	**2.76**	0.0001	0.60

the best results is the plan (B = AskCAT) with utility-based plan selector, while the constant plan selector has the worst results (Plan C = Deny). Therefore, even with an uncertain outcome when selecting a plan, our approach manages to achieve the best average satisfaction for the agent. However, this is not the case for every individual iteration, since the utility-based plan selector chooses the plan with the best expected value, but an undesired event, such as a crash or being selfish, could cause other plans to be more successful. This uncertainty is clearly seen in the results of selecting the"Deny" plan, which is associated with high standard deviation that can also be observed in Table 2 As a consequence of choosing the (Plan C = Deny), the agent may get very satisfied (very good performance and good costs) or very unsatisfied (if the agent does not gain benefits). Even though the plan"AskCTA" achieves the highest average satisfaction, and the plan best fits the agent preferences – an agent may want to take the risk of sharing the resources if this increases its chances of gaining the more benefits. Therefore, our plan selector selects a different plan for a different set of preferences. In order to consider the impact of using a plan selector over time, we also show in Figure 3 the accumulated satisfaction obtained after running 1000 iterations. The difference increases over time, but in the very first iterations this difference is small, due to the uncertainty of the scenario that arises in the selection of a plan. A proof-of-concept plan selection algorithm was performed to demonstrate the efficacy of this algorithm, using the collaboration scenario. We used the Jadex (Braubach, Lamersdorf, and

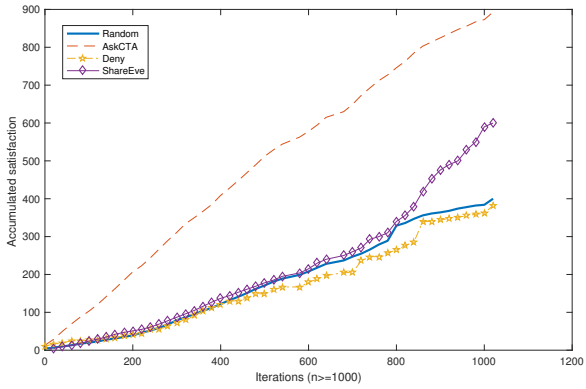

Figure 3: Accumulated Satisfaction.

Pokahr 2003) platform to implement the plan selection algorithm.

7 Discussion

As discussed above, our plan selector significantly increases an agent's satisfaction in comparison to other plan selectors. Our experiment allows us to identify a limitation of our approach: the representation of dependent probabilities. This dependency is not captured in our model yet. Including these dependencies is future work.

In our selection plan algorithm, we do not assume that each action that is executed will succeed. Observations of the agent (monitoring) will enable the agent to learn about the effectiveness of its actions. Responding to action failure, diagnosing the situation and evidential reasoning, which is necessary for reasoning about incomplete information, are future research and will be addressed in a next publication.

Considering the consequences of other agent's actions in a time-dependent environment is one of the fundamental problems in open systems.

This issue is also known as the "Interaction Framing Problem (IFP)" and will cover in the future work. The work presented here is just one step towards a model that is capable of capturing the knowledge that is necessary for agents to understand *"what is going on"* when they meet each other (Rovatsos 2001; Rovatsos, Weiss, and Wolf 2002).

8 Conclusion

Utility-based model development is a promising approach for taking the appropriate action when it is uncertain about the current state of observed information to maximize the expected utility of the decision-maker. It has also been investigated in the context of the result of taking an action (Parsons and Wooldridge 2002). In the current work, we proposed integrating utility and probabilities into our BDI planner to develop BDI agents which are able to select plans based on the highest expected utility.

We proposed a simple, but effective algorithm, that chooses a plan based on the plan utility, considering an uncertain outcome of the plan execution.

The effectiveness of our approach was shown with an em-

pirical evaluation. As future work, we will extend this approach to represent dependent probabilities in plan contributions, and also use languages and algorithms to represent and reason about qualitative preferences. Moreover, we aim to address other issues of BDI agents, such as revising an agent belief set applying evidential reasoning using expectations, confirming and disconfirming information.

9 Acknowledgments

We would like to thank the Netherlands COMMIT/program and NWO organization for making this research possible. We also like to thank KLM for providing guidance and the context for this research. We would like to thank the anonymous reviewers for their suggestions and comments.

References

Abdelkader, G. 2003. Requirements for achieving software agents autonomy and defining their responsibility. In *Proc. Autonomy Workshop at AAMAS 2003*, volume 236.

Bordini, R. H.; Hübner, J. F.; and Wooldridge, M. 2007. *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. John Wiley & Sons.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11(1):94.

Braubach, L.; Lamersdorf, W.; and Pokahr, A. 2003. Jadex: Implementing a bdi-infrastructure for jade agents.

Busetta, P.; Rönnquist, R.; Hodgson, A.; and Lucas, A. 1999. Jack intelligent agents-components for intelligent agents in java. *AgentLink News Letter* 2(1):2–5.

Dasgupta, A., and Ghose, A. K. 2010. Implementing reactive bdi agents with user-given constraints and objectives. *International Journal of Agent-Oriented Software Engineering* 4(2):141–154.

Deljoo, A.; Gommans, L.; Van Engers, T.; and de Laat, C. 2016. An agent-based framework for multi-domain service networks:eduroam case study. In *Proc. of International Conference, ICAART 2016, Rome, Italy, February 24-26, 2016*.

Dietrich, F.; List, C.; and Bradley, R. 2016. Belief revision generalized: A joint characterization of bayes' and jeffrey's rules. *Journal of Economic Theory* 162:352–371.

Giorgini, P.; Mylopoulos, J.; and Sebastiani, R. 2005. Goal-oriented requirements analysis and reasoning in the tropos methodology. *Engineering Applications of Artificial Intelligence* 18(2):159–171.

Gommans, L.; Vollbrecht, J.; Gommans-de Bruijn, B.; and de Laat, C. 2015. The service provider group framework: A framework for arranging trust and power to facilitate authorization of network services. *Future Generation Computer Systems* 45:176–192.

Hauskrecht, M. 2011. Value-function approximations for partially observable markov decision processes. *Arxiv preprint*.

Lang, J.; Van Der Torre, L.; and Weydert, E. 2002. Utilitarian desires. *Autonomous agents and Multi-agent systems* 5(3):329–363.

Liu, M.; Amato, C.; Anesta, E. P.; Griffith, J. D.; and How, J. P. 2016. Learning for decentralized control of multiagent systems in large, partially-observable stochastic environments. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Meneguzzi, F., and De Silva, L. 2015. Planning in bdi agents: a survey of the integration of planning algorithms and agent reasoning. *The Knowledge Engineering Review* 30(01):1–44.

Nunes, I., and Luck, M. 2014. Softgoal-based plan selection in model-driven bdi agents. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, 749–756. International Foundation for Autonomous Agents and Multiagent Systems.

Nunes, I.; Lucena, C.; and Luck, M. 2011. Bdi4jade: a bdi layer on top of jade. In *Proc. of the Workshop on Programming Multiagent Systems*, 88–103.

Parsons, S., and Wooldridge, M. 2002. Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 5(3):243–254.

Pokahr, A.; Braubach, L.; and Lamersdorf, W. 2005. Jadex: A bdi reasoning engine. In *Multi-agent programming*. Springer. 149–174.

Rao, A. S., and Georgeff, Michael P, e. a. 1995. Bdi agents: From theory to practice. In *ICMAS*, volume 95, 312–319.

Rovatsos, M.; Weiss, G.; and Wolf, M. 2002. An approach to the analysis and design of multiagent systems based on interaction frames. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, 682–689. ACM.

Rovatsos, M. 2001. Interaction frames for artificial agents.

Shani, G. 2007. *Learning and solving partially observable markov decision processes*. Ph.D. Dissertation, Citeseer.

Singh, D.; Sardina, S.; Padgham, L.; and Airiau, S. 2010. Learning context conditions for bdi plan selection. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, 325–332. International Foundation for Autonomous Agents and Multiagent Systems.

Van Benthem, J. 2007. Dynamic logic for belief revision. *Journal of applied non-classical logics* 17(2):129–155.

Visser, S.; Thangarajah, J.; and Harland, J. 2011. Reasoning about preferences in intelligent agent systems. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, 426.

Von Neumann, J., and Morgenstern, O. 2007. *Theory of games and economic behavior*. Princeton university press.