2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

BDI-Agent Plan Selection Based on Prediction of Plan Outcomes

João Faccin
Prosoft Research Group
UFRGS
Porto Alegre, Brazil
jgfaccin@inf.ufrgs.br

Ingrid Nunes
Prosoft Research Group
UFRGS
Porto Alegre, Brazil
ingridnunes@inf.ufrgs.br

***Abstract*—The agent technology arises as a solution that provides flexibility and robustness to address dynamic and complex domains. Such flexibility can be achieved by the adoption of existing agent-based approaches, such as the BDI architecture, which provides agents with the mental attitudes of beliefs, desires and intentions. This architecture is highly customisable, leaving gaps to be fulfilled in particular applications. One of these gaps is the plan selection algorithm that is responsible for selecting a plan to be executed by an agent to achieve a goal, having an important influence on the overall agent performance. Thus, in this paper, we propose a plan selection approach, which is able to learn plans that provide possibly best outcomes, based on a current context and agent's preferences. Our approach is composed of a meta-model, which must be instantiated to specify plan metadata, and a technique that uses such metadata to learn and predict plan outcomes. We evaluated our approach experimentally, and results indicate it is effective.**

I. Introduction

Software agents are a promising approach to deal with domains of increasing complexity and dynamic problems. Such agents can be defined as computational entities with autonomous, proactive, reactive and social characteristics. This means that an agent is capable of perceiving, through sensors, the environment in which it is located, and acting on this environment through actuators, or reacting to changes on it, aiming to achieve their own goals, without the need for human intervention.

Several approaches, including methodologies, models and techniques, were proposed to support the design and implementation of software agents. The belief-desire-intention (BDI) architecture [1] is such an approach, which is the foundation of much research work. A key advantage of this architecture is that it provides robustness to deal with different application areas in which flexible and intelligent behaviour is needed. The BDI architecture structures agents in terms of the mental attitudes of beliefs, desires and intentions, which collectively determine agent behaviour. Such mental attitudes are used within a reasoning cycle, which leaves many gaps to be fulfilled in particular applications. One of these gaps is the *plan selection process*, which is the process in which a plan is selected, from a set of predefined suitable plans, to achieve a goal. Agents can select a plan randomly and, in case this plan fails, an alternative plan is selected to achieve the goal. If this plan selection process is able to choose the most appropriate plan to achieve a goal, given the current context and agent particularities such as its preferences, agent's performance will likely improve.

To achieve the full potential of the BDI architecture, it is thus necessary to specify an adequate plan selection algorithm. Thereby, several approaches have been proposed in this context, e.g. [2], [3]. Some of them focus on learning the context in which a plan possibly fails, thus increasing the chance that only plans that will succeed are executed. Others aim to identify the plan that best matches agent's preferences. A recent approach [4] not only considers agent's preferences, but also the uncertainty of possible plan outcomes regarding secondary goals (or *softgoals* [5]), such as minimisation of execution time, that are important to an agent. Nunes and Luck [4] claimed that their approach requires little information about the domain to develop specific applications; however, it requires the specification of probabilities of each possible plan outcome, which are hard to elicit, may evolve over time, and may be context-dependent. Consequently, their required input may be considered *unrealistic* for real applications because: (i) probabilities and values are often unknown at design time; (ii) if it is possible to obtain them, the effort required to obtain all probabilities and all outcomes for all plans is too high, and likely it will not be made for real applications; (iii) probabilities and values are context-dependent; (vi) probabilities evolve over time. These shortcomings thus call for new approaches that learn these probabilities and outcomes to support the plan selection process, and evolve them over time.

We, in this paper, thus propose an approach that eliminates the need for the specification of such detailed information. In our approach, agents learn which are the expected plan outcomes according to particular contexts. Based on this, agents are able to *predict plan outcomes* based on the current context, information which is used to select a plan to achieve a goal, taking into account preferences that are expressed over agent's *softgoals*. Our approach is composed of a meta-model, which must be instantiated to specify plan metadata, and a technique that uses such metadata to learn and predict plan outcomes, in order to select plans. We evaluated our approach with a simulation that indicates that our approach is effective. Our approach thus extends existing work mainly by providing means of learning the required input of the plan selection algorithm, and simplifies agent design by requiring simple information as part of an agent design built based on our meta-model, thus hiding technique details usually unknown by mainstream software developers. This is a concern that drives

978-1-4673-9618-9/15 $31.00 © 2015 IEEE
DOI 10.1109/WI-IAT.2015.58

our research and is essential to make the industrial adoption of the agent technology practically feasible, but is often left unaddressed by agent community.

In Section II, we briefly introduce concept definitions proposed by Nunes and Luck, which are adopted in our work. We also describe an example used throughout this paper. We next detail our approach in Section III, and then present the experiment conducted to evaluate it in Section IV. Finally, Section V discusses related work, and Section VI concludes.

II. Background and Running Example

The problem addressed by Nunes and Luck [4] consists of choosing the best plan to achieve a goal, from a set of candidate plans. All these candidates are able to achieve the goal, but they have different side effects, which are associated with agent softgoals. Their ultimate goal is to maximise agent satisfaction, considering agent's preferences over softgoals. Our goal is similar but, as said above, we do not require the specification of information that is difficult, if not impossible, to be provided. Given that our work is based on this existing work, we present next three definitions of these authors that we use.

Definition 1 (Softgoal): An agent softgoal $sg \in SG$ is a broad agent objective, which is not achieved by a plan but is more or less satisfied due to the effect produced by agent's actions that are part of plans.

Definition 2 (Preferences): $Pref \rightarrow [0, 1]$ is a function that maps a softgoal $sg \in SG$ to a value indicating the preferences for softgoals. A preference is a value $pref_{val} \in [0, 1]$ that indicates the importance of a softgoal $sg \in SG$, 0 and 1 being the lowest and highest preference, respectively. Moreover $\sum_{sg \in SG} Pref(sg) = 1$.

Definition 3 (Agent): An agent is a tuple $\langle B, G, SG, P, Pref \rangle$ where B is a set of beliefs, G is a set of goals, SG is a set of softgoals, P is a set of plans, and $Pref$ is a preference function.

We exemplify the concepts defined above in the scenario described next, which is in the context of logistic companies. This scenario also serves as a motivation for our work, and will be used throughout the paper. Logistic companies have, in the core of their business model, the action of *delivering* customer's *loads* from one place to another. Aiming to maximise its profits, a company with several *modes of transportation* must choose one, which better fits a given scenario.

A logistic company C has four options (or plans) to transport loads, which are delivering loads (i) by airplane (*AirplanePlan*); (ii) by ship (*ShipPlan*); (iii) by train (*TrainPlan*); and (iv) by truck (*TruckPlan*). An agent A is able to coordinate C, and must choose one of these available plans for each delivery request. Though all these plans can achieve the goal of *transporting a load from place x to place y* (for short, $transport(x, y)$), the choice must take into account other factors that influence the decision. First, the rapidness to accomplish the goal may differ, e.g., transporting a load by airplane tends to be faster than by ship, if x is far from y. However, an airport can be closed due to bad weather condition, turning the other options more suitable in this context. Second, the cost of transportation varies across the different transportation modes, being influenced by a number of factors, such as the fuel consumption, vehicle's maintenance condition, among others. Third, it is possible that the load integrity is compromised, when vehicles are exposed to particular contexts. For example, when an airplane goes through turbulence, or when a truck rides in a road with poor conditions.

In this scenario, all plans achieve the goal $transport(x, y)$. But this goal can be achieved in different amounts of time, with different costs, and be associated with different levels of reliability (related to load losses and damages) — these are plan outcomes. These are associated with different agent concerns, more specifically: maximise performance (*maxPerformance*), minimise costs (*minCosts*), and maximise reliability (*maxReliability*). These are softgoals, which are defined in Definition 1. Note that these softgoals are often conflicting, i.e. they cannot be maximised at the same time. Preferences (Definition 2), in turn, are the trade-off relationship between these softgoals. They express how much important each softgoal is. For instance, our agent needs to deliver a fragile load with extreme urgency. It may prefer to attempt a plan that offers a faster transport with highest reliability. Preferences that represent this scenario could be $maxPerformance = 0.4$, $maxReliability = 0.4$, and, $minCost = 0.2$. That is, the first two softgoals are equality important, and both are more important than the latter.

In summary, agent A is thus an entity able to execute plans in order to achieve its goal. It has the goal $transport(x, y)$; the plans, preferences and softgoals described above; in addition to beliefs, which comprise its knowledge. These are the five elements that comprise the definition of an agent, given in Definition 3. Considering this scenario, a question remains: how can agent A choose a plan, without requiring us to explicitly provide probabilities of plan outcomes?

III. Learning-based Plan Selection

We presented the problem that is the focus of this paper, and we now describe our approach to address it. We first provide an overview of our approach in Section III-A. This approach has two main components, which are a meta-model that is described in Section III-B, and a technique to learn and predict plan outcomes that is detailed in Section III-C.

A. Approach Overview

In order to learn plan outcomes, our approach takes as input the factors that influence plan outcomes, e.g. the weather conditions may influence the time taken to deliver a load. Therefore, the key information that must be provided while modelling specific applications is this relationship between these factors and plan outcomes. Once this information is learned, it can be used to predict plan outcomes for a given context, and then the plan with the best outcomes is selected.

Figure 1 shows the steps of our approach. Plans must be specified with additional metadata, which must specify the influence factors and outcomes for each plan. Moreover, such outcomes must be associated with softgoals; for example, load integrity can be associated with the *maxReliability* softgoal. In the BDI reasoning cycle, after agents generate goals to achieve, select those they are committed to achieve, and select a plan to achieve them; they execute a plan. When this happens, our approach records the current state of influence factors of the

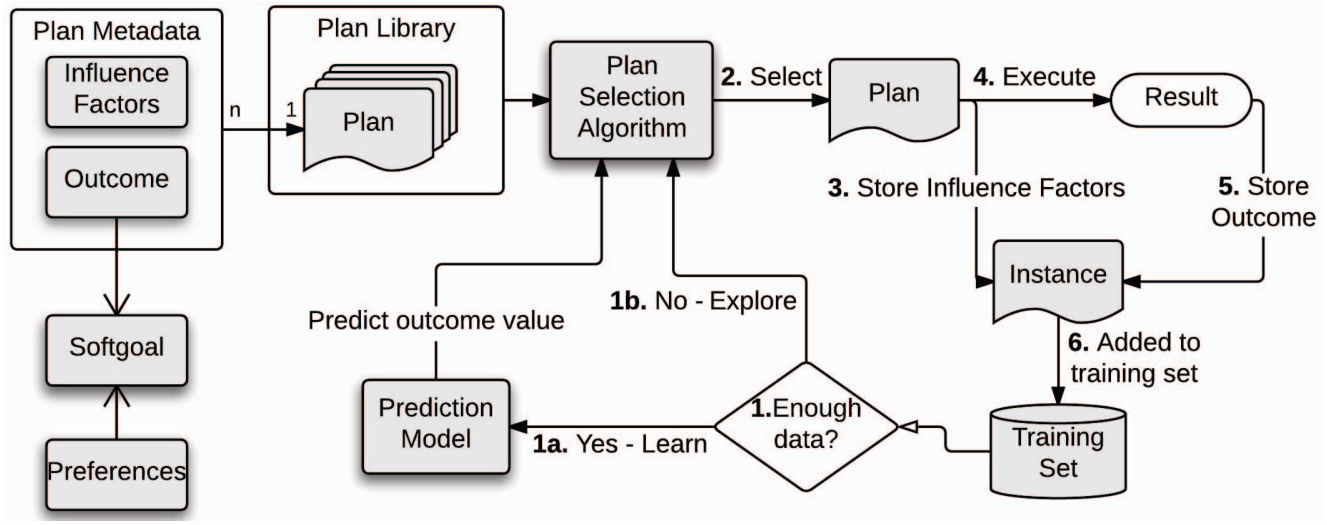

Fig. 1. Approach Overview.

selected plan and, after the plan execution, it records plan outcomes. This allows us to collect the data needed to predict outcomes in the future.

When there is enough data collected, agents use them to build a prediction model. When a plan must be selected, plan outcomes are predicted using these data and current context. These expected outcomes are then transformed into utilities, associated with the satisfaction of softgoals. Next, utilities are combined using agent's preferences and, finally, the plan with the best utility (i.e. which maximises preference satisfaction) is selected. The execution of this selected plan is recorded as well, adding new data to improve the prediction model.

The different concepts associated with our approach are specified in a meta-model described next, and then we describe how we build and use our prediction model.

B. Meta-model

We briefly described above the main concepts of our meta-model. These concepts are those that need to be instantiated in specific applications that use our approach. Our meta-model must specify concepts to allow representation of: (i) plan outcomes, and factors that influence them; (ii) a function that transforms plan outcomes into utilities; (iii) association of such utilities with softgoals; and (iv) preferences over softgoals. We represent the last with the function introduced in Section II. We begin by defining what we want to observe from plan executions, i.e. outcomes, as shown next.

Definition 4 (Outcome): An outcome $o \in O$ is a measurement that can be taken during and/or after a plan execution. It is associated with a domain. Measuring an outcome from a plan execution gives us a *value* within this domain.

As previously introduced, such outcomes are directly or indirectly affected by factors, that is, outcomes can be seen as an (unknown) function of these factors that influence them. Factors give the context information that a plan executes, which is a state that, in the BDI architecture, is given by beliefs. Consequently, in our meta-model, factors are mapped to agent beliefs. Influence factors are thus defined as follows.

Definition 5 (Influence Factor): An influence factor $if \in IF$ is context variable, which is mapped to belief $b \in B$, and the value of which influences one or more plan outcomes $o \in O$.

For instance, the traffic condition can influence the time that the *TruckPlan* takes to achieve its goal and, also, a vehicle in maintenance condition can affect the integrity of the load. In these cases, traffic condition and maintenance condition are influence factors (whose values are given by agent beliefs), and time and integrity are outcomes, respectively. An outcome is thus just value; therefore, we must provide means of specifying how such values satisfy agent softgoals. Currently, we assume that outcomes are numeric values, and agents have one of two objectives: either maximise or minimise an outcome value, i.e. the higher (in case of maximise) or the lower (in case of minimise) the outcome value, the more satisfied a softgoal. This is given by the definition below.

Definition 6 (Optimisation Function): An optimisation function $of \in OF$, where $OF = \{min, max\}$, specifies whether the value of an outcome $o \in O$ must be minimised or maximised to better satisfy a softgoal $sg \in SG$.

For example, the optimisation function should be *min* for the outcome *time* and softgoal *maxPerformance*, meaning that the lower the time, the more satisfied the *maxPerformance* softgoal. Now, we put all these defined concepts together in the definitions of a plan and plan metadata.

Definition 7 (Plan): A plan $p \in P$ is a tuple

$$\langle Ctx, G', Body, MD \rangle$$

where Ctx is the context conditions that specify when plan is applicable, G' is a set of goals that can be achieved by this plan, $Body$ is a set of actions performed when the plan is executed, and MD is the plan metadata.

Definition 8 (Plan Metadata): A plan metadata is a partial function

$$MD : SG \nrightarrow O \times \mathbb{P}(IF) \times OF$$

TABLE I. EXAMPLE OF RECORDED PLAN EXECUTIONS.

Execution	Truck Conditions	Traffic Conditions	Road Conditions	Time Taken
1	0.591	0.903	0.096	2.35
2	0.858	0.987	0.419	2.5
3	0.495	0.430	0.677	1.725
...	...	...	...	...

which specifies an outcome, a set of influence factors and an optimisation function, for a particular softgoal.

This completes the description of our meta-model, which is implemented as an extension of the BDI4JADE[1] platform [6] — Figure 2 shows its main classes. This implementation was used to run the simulation presented in the evaluation section. Some classes, namely `SoftGoal`, `NamedSoftGoal`, and `SoftGoalPreferences`, were reused from the extension available in the BDI4JADE website.

C. Learning and Plan Selection

Our meta-model must be instantiated in particular applications to provide the input needed by our plan selection technique. This section shows how such inputs are used to build a prediction model, which is used to select plans. We assume that influence factors and outcomes are provided at development time, and are fixed at runtime; while softgoals, preferences over them, and optimisation functions can be provided at runtime, and evolve over time, but can be initialised at development time.

Given that our aim is to build a prediction model based on recorded plan executions, we first need an initial learning stage, where minimal amount of data is collected to extract useful information. During this stage, plans may be selected randomly, for example. A threshold must be specified by developers to indicate when a prediction model must be built to select plans. For example, a minimum number of plan executions to achieve a particular goal may be provided.

As result of recording plan executions, we store collected information in the form of a table, for each plan, so as to form a *dataset*. An example of this table is shown in Table I, which is associated with the *TruckPlan*. The first column of this table (*Execution*) is an identifier of the recorded plan execution, the second, third and fourth columns are influence factors, and the last column is an outcome (in hours). In this example, there is only one outcome, but it is possible to have others, associated with different softgoals. If there were other outcomes, influence factors of all of them would be recorded and be columns in this table. As previously mentioned, we are currently considering only numeric outcomes, so time is a continuous real number in this case. Influence factors, in this example, are given as a value between 0 and 1, which are the worst and best conditions, respectively. However, influence factors may be associated with any domain.

As influence factors are associated with beliefs, and collecting them means consulting the current value associated with beliefs in the agent belief base, which is easy. Outcomes, in turn, are expected to be collected from external sources through sensors, such as speedometers, timers, and so on. Therefore, in our approach, outcome is an abstract concept, and specific applications must provide means of measuring outcomes during or after plan execution. Some outcomes just need to be measured in the end of the plan execution (e.g. load integrity), but others require also to do some preprocessing (e.g. measure the start and end times, and calculate the difference). This is the reason why the outcome class of our implementation provides methods to start and end the measurement of an outcome (see Figure 2), which are invoked by our plan selection strategy.

Given this collected data, we are now able to use existing machine learning algorithms to predict outcome values. The nature of our problem consists of understanding the relationship between a dependent variable (an outcome) and one or more independent variables (influence factors). Thus, any machine learning algorithm able to perform regression can be used in our approach, e.g. linear regression [7], support vector machines [8] and multilayer perceptron [9]. Our approach is currently using linear regression, but better performance may be achieved when a learning algorithm is chosen taking into account the specificities of what is being learned. The predicted outcome value v_o using linear regression is thus given by $pred_{lr}(dataset(p), o)$, where $dataset(p)$ is a collected dataset for plan p, and o is an outcome.

When enough data is available, a prediction model is thus built for each outcome, because each has its particular influence factors. Then, in order to select a plan, we use such models to predict outcomes according to the current context. After learning plan outcomes based on the current context, there are two challenges left that need to be addressed: (i) how to convert predicted outcomes to an agent utility; and (ii) how to combine these utilities and select a plan.

Predicted plan outcomes are values of the outcome domain. The conversion of this outcome domain to a utility value, $\mathbb{T}(v_o, o)$, ranging from 0 (worst) to 1 (best), is made using the optimisation function and domain boundaries (i.e. maximum and minimum values). Such domain boundaries are used to normalise the value to a scale ranging from 0 to 1, which are the minimum and maximum domain values, respectively, if the optimisation function is *max*. If the optimisation function is *min*, domain boundaries are used in the opposite way. For example, the outcome value 2.35 hours for the time outcome is converted to 0.530, because the domain boundaries are 0 and 5, and time is associated with the *min* optimisation function. If no domain boundaries are available, we use the minimum and maximum values across datasets.

The selected plan is that with the *highest utility*. In order to combine utilities of individual outcomes, we use a *weighted sum* model, given that agent's preferences can be used as weights that establish the trade-off relationship among softgoals, and each of which is associated with an outcome. This sum is also possible as all utilities of individual outcomes are in the same scale. Consequently, the plan selected to achieve a goal g is the plan that satisfies the following equation from all candidate plans to achieve g, where o_{sg} is the outcome associated with the softgoal sg.

$$\arg\max \sum_{sg \in SG} Pref(sg) \cdot \mathbb{T}(pred_{lr}(dataset(p), o_{sg}), o_{sg})$$

[1] http://inf.ufrgs.br/prosoft/bdi4jade

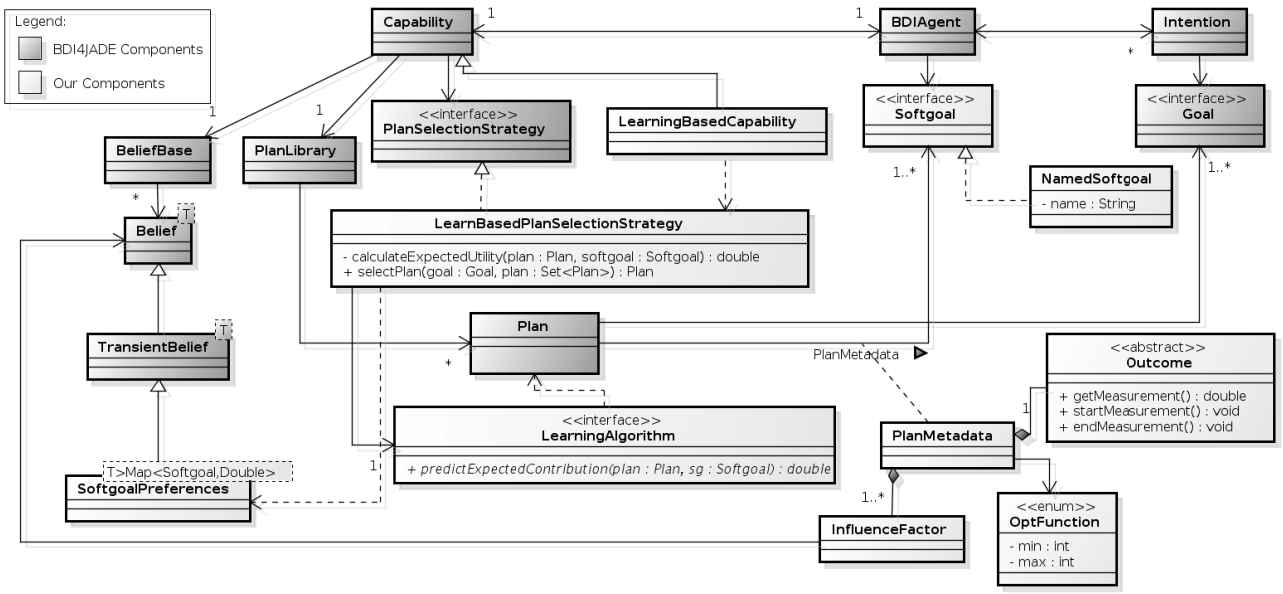

Fig. 2. Meta-model as a BDI4JADE Extension.

IV. Evaluation

Now that we presented our meta-model with its concepts and how they are used to endow an agent with a learning capability to perform an improved plan selection, we evaluate our approach with an empirical experiment. Although much work considers the use of learning techniques in the plan selection process, they all differ from our approach in the subject of learning. Furthermore, some of them focus on identifying applicable plans, i.e. learning when plans fail [10], [11], [12], while in our approach we select the best of a set of plans, which are applicable to the current context (which is given). In our approach, other plans are still candidates in case this purported best plan fails. Therefore, because the goal of these approaches differs from ours, it is not possible to compare it with them. The most similar approach to ours is that from Nunes and Luck [4], but their approach requires a different and more complicated input — actually, our approach learns the input that they require to be explicitly provided, and be fixed at runtime. We then compare our approach to a random plan selection, which is the default option in existing BDI platforms. The procedure of our evaluation is presented in Section IV-A, and its results and discussion are presented in Sections IV-B and IV-C, respectively.

A. Procedure

To evaluate our approach we implemented the scenario presented in Section II. This scenario involves an agent A with the goal $transport(x, y)$, which means transporting a load from place x to y. As introduced before, to achieve this goal our agent has four possible plans: *AirplanePlan*, *ShipPlan*, *TrainPlan*, and *TruckPlan*. Moreover, the agent has three softgoals to consider, namely *maxPerformance*, *minCosts*, and *maxReliability*. In addition, each plan is associated with its metadata, specifying influence factors, an outcome, and an optimisation function, for each softgoal. For example, the *TruckPlan* has the influence factors and outcome shown in Table I and is associated with the *min* optimisation function, with respect to the *maxPerformance* softgoal. Similar metadata are specified for each softgoal in each plan. Table II presents these details.

TABLE II. Plans and Respective Metadata

Plan	Softgoal	Influence Factor	Outcome
AirplanePlan	minCosts	AirplaneConditions	FuelConsumption
		WeatherConditions	
		Distance	
	maxPerformance	AirplaneConditions	TimeTaken
		WeatherConditions	
		AirportConditions	
	maxReliability	AirplaneConditions	LoadIntegrity
		AccidentProbability	
		ChanceOfTheft	
ShipPlan	minCosts	ShipConditions	FuelConsumption
		WeatherConditions	
		Distance	
	maxPerformance	ShipConditions	TimeTaken
		WeatherConditions	
		SeaConditions	
		HarborConditions	
	maxReliability	ShipConditions	LoadIntegrity
		AccidentProbability	
		ChanceOfTheft	
TrainPlan	minCosts	TrainConditions	FuelConsumption
		WeatherConditions	
		Distance	
	maxPerformance	TrainConditions	TimeTaken
		TrafficConditions	
		RailroadConditions	
	maxReliability	TrainConditions	LoadIntegrity
		AccidentProbability	
		ChanceOfTheft	
TruckPlan	minCosts	TruckConditions	FuelConsumption
		TrafficConditions	
		Distance	
	maxPerformance	TruckConditions	TimeTaken
		TrafficConditions	
		RoadConditions	
	maxReliability	TruckConditions	LoadIntegrity
		AccidentProbability	
		ChanceOfTheft	

TABLE III. SATISFACTION BY PLAN SELECTOR (n = 5000).

Plan Selector	M	SD	Min	Max	Cum Sat
LB-1	0.6812	0.091	0.219	0.961	3406.04
LB-100	0.6815	0.091	**0.262**	0.968	3407.99
LB-500	0.680	0.092	0.214	**0.982**	3402.68
LB-1000	**0.6816**	*0.090*	0.138	0.969	**3408.00**
RAN	*0.597*	**0.114**	*0.102*	*0.941*	*2988.98*

Our experiment consists of measuring the agent satisfaction (i.e. obtained utility) produced using different plan selection strategies. For this purpose we ran a simulation, in which we performed the following steps in each iteration: (i) randomly generate preferences for each softgoal; (ii) randomly instantiate a current context, and agent beliefs are adapted accordingly; (iii) predict outcomes for each plan; (iv) select a plan using a plan selector (our algorithm or randomly); and (v) measure and store the satisfaction of agent's preferences of the plan execution.

Satisfaction of agent's preferences is measured by transforming the results of a plan execution (outcome values) to utility. The result of a plan execution (e.g. actual time taken) was generated randomly, using a normal distribution parameterised with arbitrary average and standard deviation. In order to ensure unbiased results, we ran our simulation with different parameters, and they all performed similarly. Furthermore, the threshold for building the prediction model and using it was set to 50 plan executions. This prediction model was updated with different update-rates (i.e. the prediction model is rebuilt after a specified number of plan executions), and results for each are reported next.

B. Results and Analysis

After running 5000 iterations of steps listed above, we compared the accumulated satisfaction for each plan selector. Besides a random plan selector, we used plan selectors using our approach, updating the prediction model after each plan execution (*LB-1*), 100 plan executions (*LB-100*), 500 plan executions (*LB-500*), and 1000 plan executions (*LB-1000*). The average satisfaction, standard deviation, minimum, maximum and accumulated satisfaction for each plan selector are presented in Table III, with highest values highlighted in bold and lowest ones in italics; while Figure 3 shows the box plot of the agent satisfaction obtained for each plan selector. We also compared the accumulated satisfaction of each plan selector, which is depicted in Figure 4.

We can observe that obtained satisfaction may seen not extremely high (the best average satisfaction is 0.6816); however, we need to highlight that the *uncertainty* associated with plan executions is still playing its role, preventing results for being always accurately predicted. For example, bad weather means that there is a high probability of delay, but it is not guaranteed that there will actually be delay. Moreover, the plan selection process involves a choice where *trade-off* must be resolved, that is, some plans are better with respect to certain softgoals and others are better with respect to other softgoals — e.g. resolve a trade-off between transportation time and cost. Consequently, no plan leads to the best satisfaction value (i.e. 1.0), and 0.6816 can be considered a high value.

Analysing our results, it can be observed that our learning-based plan selector always performs better than the random

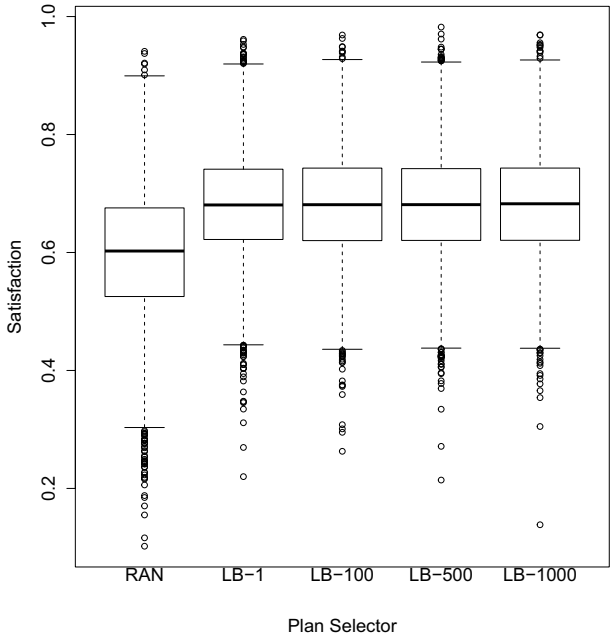

Fig. 3. Analysis of Satisfaction by Plan Selector.

plan selector, independently from the selected update-rate. We highlight that, although it is not possible to see in the presented charts, the results for all plan selectors are similar in the first iterations, given that there is a need for obtaining an initial dataset for building prediction models. However, the lower the update-rate, the faster the plan selector learns. LB-1000 has slightly better results than those obtained with LB-1, LB-100 and LB-500. However, these results do not differ significantly among them (see Figure 3). The *small* differences between the plan selectors using our approach with different update-rates are expected, because after building a model with enough data to learn an adequate model, the behaviour of all plan selectors tend to be the same. However, if there were an event in the system that would impact in the relationship between influence factors and outcomes, plan selectors with lower update-rates would faster react to these changes.

A one-way ANOVA was used to test for preference differences among satisfaction of each plan selector. Satisfaction of plan selectors differed *significantly* across the five selectors, $F(4, 24995) = 747.8$, $p \ll .05$. Post-hoc Tukey's HSD [13] tests showed that comparisons between all learning-based plan selectors and the random plan selector were significantly different at .05 level of significance. With respect to performance, all simulations using plan selectors based on our approach execute in around 10 seconds, except LB-1, which executes in approximately 6 minutes.

Some may argue the random plan selector is not a reliable baseline for evaluating our approach. However, as argued before, we cannot make a direct comparison with other approaches since they focus on learning in which context plans fail. Nunes and Luck's approach is the most similar to ours, but

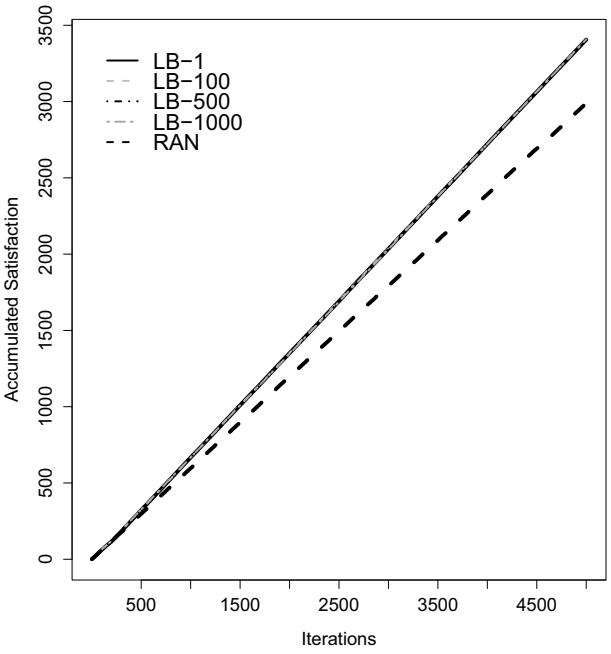

Fig. 4. Accumulated Satisfaction by Plan Selector.

requires an input often unavailable for real applications (their baseline is also a random plan selector). Consequently, given the novelty of our approach, it was impossible to evaluate it against existing approaches.

C. Discussion

Our simulations indicate that our approach can significantly increase agent satisfaction by selecting plans that will likely satisfy more agent's preferences. Based on our evaluation, we discuss in this section important issues. Although we do not constrain the value types of influence factors, we are currently considering only numeric continuous values. We may notice that, in order to use discrete values for an influence factor, we should know in advance all possible values it can have. In addition, we give the same importance to all records in our dataset, but it may be interesting to give more weight to recently collected records, mainly in dynamic environments.

One may say that online learning, such as Q-learning, is a more adequate solution to our problem. Note, however, that here we make the reward independent from the plan outcome, and there are different plan outcomes for each plan. In this way, using the same prediction model, we are able to deal with agent's preferences and optimisation functions that evolve over time. We are aware that our present form of transforming plan outcomes to agent utility is simple (i.e. minimise or maximise values), but it is already effective and applicable to many scenarios. It is part of our future work to explore other forms of transformation functions, leveraging preference elicitation approaches [14].

V. Related Work

In this section, we discuss work related to our approach. Approaches discussed here focus on the use of learning techniques and agent's preferences for improving the plan selection process in BDI agents.

One of the first attempts of using learning methods to support plan selection was proposed by Guerra-Hernández *et al.* [2], who aimed to learn in which context a plan is applicable. This is the same concern of Phung *et al.* [15]. In complex environments, writing correct preconditions may be a difficult task. So, these approaches can save effort reducing the burden of encoding adequate plan preconditions for a plan be applicable. With these approaches, preconditions can be learned or refined. Other approaches in this context focused on learning, from a set of applicable plans, which is the most suitable plan in a given context [16], [10], [17], [11], [18], [12]. However, all these approaches consider a most suitable plan that with the lowest probability of failing, and do not cope with other important aspects (represented here as softgoals) to be considered when selecting a plan. Even though our focus is not on learning failure possibility, our approach can deal with it if a failed execution is recorded with the worst possible outcomes values. In addition, none of these approaches explore different learning algorithms, they all use decision trees to support plan selection. Furthermore, only Singh *et al.* [12] deal with dynamic environments, which provides the ability to re-learn its behaviour when perceiving loss of effectiveness while executing plans. Our approach is also able to cope with dynamic environments by updating prediction models using a specified update rate.

Considering the use of preferences in the plan selection process, there are only three main proposed approaches. The work of Visser *et al.* [3] considers preference specification over attributes of a goal. Padgham and Singh [19] use logical expressions to specify preferences. Both of these approaches require the expression of complex preferences over all properties or attributes of a plan or goal, which may be unrealistic in real-world scenarios. Moreover, plan results must be given. By adopting the notion of preference of *softgoals* proposed by Nunes and Luck [4], we are able to capture preferences in a simple way. Nevertheless, as discussed in the introduction, their work requires the specification of information (probabilities for each plan outcome) that is difficult (if not impossible) to be provided for complex applications. Furthermore, their approach cannot cope with dynamic domains, where probabilities and outcome values are context-dependent and evolve over time. Our approach thus addresses these limitations, and extends their work by proposing a meta-model which is easier to instantiate to build applications, with the concepts of outcomes, influence factors, optimisation function and so on, and leverage learning techniques to select plans.

VI. Conclusion

The BDI architecture is a robust solution proposed to deal with dynamic and complex domains, addressed by applications that need to provide flexible and intelligent behaviour. One of the main reasons for the flexibility and robustness of this approach is the plan selection process, part of the BDI reasoning cycle. This plan selection process is highly customisable and many techniques have been developed to improve it.

In this paper, we proposed an approach that improves the plan selection process by the use of machine learning techniques. Our approach allows agents to learn the plans that possibly will perform best considering the current context and agent's preferences over softgoals. Our approach is twofold. First, we introduced a meta-model that defines concepts that allow the representation of the information needed for the plan selection, which must be provided by specific applications. Second, we proposed a technique that selects plans based on agent's preferences and the information provided by the meta-model instances. The latter is used to predict plan outcomes according to the current context. Therefore, an agent built with our approach is capable of learning to select a plan that best satisfies agent's preferences using previous plan executions. Moreover, learning allows agents to cope with dynamic environments that evolve over time. Our empirical evaluation showed that our approach is effective even when the prediction model is not frequently updated.

This work leaves many possibilities to be explored that we aim to investigate. First, our approach will be extended to support different forms of transforming plan outcome values to utilities. Second, we will take into consideration when plan executions are performed to give weights to recorded observations according to their recency. Third, we will explore learning algorithms other than that adopted in this paper. In addition, our goal is to integrate this approach to a model-driven approach for BDI-agent development, so that sophisticated agents can be generated with the instantiation of a simple meta-model. Our main objective is to make this approach easier to be adopted by mainstream software developers, thus promoting the large-scale adoption of agent technology in industry.

Acknowledgements

This work receives financial support of CNPq, project #442582/2014-5: "Desenvolvendo Agentes BDI Efetivamente com uma Abordagem Dirigida a Modelos."

References

[1] A. S. Rao and M. P. Georgeff, "Bdi agents: From theory to practice," in *IN PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS (ICMAS-95)*, 1995, pp. 312–319.

[2] A. Guerra-Hernández, A. E. Fallah-Seghrouchni, and H. Soldano, "Learning in bdi multi-agent systems." in *CLIMA*, ser. Lecture Notes in Computer Science, J. Dix and J. A. Leite, Eds., vol. 3259. Springer, 2004, pp. 218–233.

[3] S. Visser, J. Thangarajah, and J. Harland, "Reasoning about preferences in intelligent agent systems." in *IJCAI*, T. Walsh, Ed. IJCAI/AAAI, 2011, pp. 426–431. [Online]. Available: http://dblp.uni-trier.de/db/conf/ijcai/ijcai2011.html#VisserTH11

[4] I. Nunes and M. Luck, "Softgoal-based plan selection in model-driven bdi agents," in *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, ser. AAMAS '14. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 749–756.

[5] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, pp. 203–236, 2004. [Online]. Available: http://dx.doi.org/10.1023/B:AGNT.0000018806.20944.ef

[6] I. Nunes, C. Lucena, and M. Luck, "Bdi4jade: a bdi layer on top of jade," in *ProMAS 2011*, Taiwan, 2011, pp. 88–103.

[7] F. Galton, *Natural inheritance*. Macmillan and Company, 1894.

[8] A. Smola and B. Schoelkopf, "A tutorial on support vector regression," Tech. Rep., 1998, neuroCOLT2 Technical Report NC2-TR-1998-030.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds. Cambridge, MA, USA: MIT Press, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362. [Online]. Available: http://dl.acm.org/citation.cfm?id=104279.104293

[10] S. Airiau, L. Padgham, S. Sardina, and S. Sen, "Incorporating learning in BDI agents," in *Proceedings of the Adaptive Learning Agents and Multi-Agent Systems Workshop (ALAMAS+ALAg-08)*, Esteroil, Protugal, May 2008.

[11] D. Singh, S. Sardina, L. Padgham, and S. Airiau, "Learning context conditions for bdi plan selection." in *AAMAS*, W. van der Hoek, G. A. Kaminka, Y. Lesprance, M. Luck, and S. Sen, Eds. IFAAMAS, 2010, pp. 325–332.

[12] D. Singh, S. Sardina, L. Padgham, and G. James, "Integrating learning into a BDI agent for environments with changing dynamics," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, C. K. Toby Walsh and C. Sierra, Eds. Barcelona, Spain: AAAI Press, Aug. 2011, pp. 2525–2530.

[13] J. W. Tukey, "Comparing individual means in the analysis of variance," *Biometrics*, vol. 5, no. 2, pp. pp. 99–114, 1949. [Online]. Available: http://www.jstor.org/stable/3001913

[14] R. L. Keeney and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. New York: John Wiley & Sons, Inc, 1976.

[15] T. Phung, M. Winikoff, and L. Padgham, "Learning within the bdi framework: An empirical analysis," in *Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems - Volume Part III*, ser. KES'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 282–288. [Online]. Available: http://dx.doi.org/10.1007/11553939_41

[16] A. Nguyen and W. Wobcke, "An adaptive plan-based dialogue agent: integrating learning into a bdi architecture." in *AAMAS*, H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, Eds. ACM, 2006, pp. 786–788. [Online]. Available: http://dblp.uni-trier.de/db/conf/atal/aamas2006.html#NguyenW06

[17] S. Airiau, L. Padham, S. Sardina, and S. Sen, "Enhancing adaptation in BDI agents using learning techniques," *International Journal of Agent Technologies and Systems (IJATS)*, vol. 1, no. 2, pp. 1–18, Jan. 2009.

[18] D. Singh, S. Sardina, and L. Padgham, "Extending BDI plan selection to incorporate learning from experience," *Journal of Robotics and Autonomous Systems*, vol. 58, pp. 1067–1075, 2010.

[19] L. Padgham and D. Singh, "Situational preferences for bdi plans," in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, ser. AAMAS '13. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 1013–1020.