Intention Progression under Uncertainty

Yuan Yao[1] , **Natasha Alechina**[2] , **Brian Logan**[3] and **John Thangarajah**[4]
[1] College of Computer Science and Technology, Zhejiang University of Technology
[2] Department of Information and Computing Sciences, University of Utrecht
[3] School of Computer Science, University of Nottingham
[4] School of Science, RMIT University
yaoyuan@zjut.edu.cn, n.a.alechina@uu.nl, brian.logan@nottingham.ac.uk, john.thangarajah@rmit.edu.au

Abstract

A key problem in Belief-Desire-Intention agents is how an agent progresses its intentions, i.e., which plans should be selected and how the execution of these plans should be interleaved so as to achieve the agent's goals. Previous approaches to the intention progression problem assume the agent has perfect information about the state of the environment. However, in many real-world applications an agent may be uncertain about whether an environment condition holds, and hence whether a particular plan is applicable or an action is executable. In this paper, we propose SA_U, a Monte-Carlo Tree Search (MCTS)-based solver for intention progression problems where the agent's beliefs are uncertain. We evaluate the performance of our approach experimentally by varying the degree of uncertainty in the agent's beliefs. The results suggest that SA_U is able to successfully achieve the agent's goals even in settings where there is significant uncertainty in the agent's beliefs.

1 Introduction

The Belief-Desire-Intention (BDI) architecture [Rao and Georgeff, 1992] is the foundation for numerous deployed agent systems [Pěchouček and Mařík, 2008]. In these systems, *beliefs* represent the agent's information about the environment and itself, *goals* are states of the environment the agents want to bring about, and *plans* are the recipes by which the agents can achieve their goals. *Plans* consist of primitive actions that directly change the state of the environment, and subgoals which are in turn achieved by subplans. An agent typically pursues multiple goals in parallel. When the agent commits to a particular plan to achieve a goal, an *intention* is formed. At each deliberation cycle, the agent chooses which of its multiple intentions it should progress (i.e., intention selection) and for the subgoals within an intention choose the best plan to achieve it (i.e., plan selection). In this work, we consider these two choices together as the *intention progression* problem.

A number of approaches have been proposed in the literature that address various aspects of intention progression, including, *summary-information-based (SI)* [Thangarajah *et al.*, 2003; Thangarajah and Padgham, 2011], *coverage-based (CB)* [Waters *et al.*, 2014; 2015] and *Monte-Carlo Tree Search-based* [Yao *et al.*, 2014; Yao and Logan, 2016; Yao *et al.*, 2016] approaches. However all of these existing approaches assume the agent has perfect information about its environment, i.e., if the agent believes p is true, then p is in fact true. In reality though, the agent may be only certain of some of its environment, and be unsure about the rest. For example, suppose an automated taxi is travelling to a taxi rank. It may have a certain belief that there is a taxi rank at a particular location but may not know for certain if there are passengers waiting at that rank but have some degree of certainty (for example, 75% based on prior experience).

There has been considerable work on modelling uncertain beliefs in agent systems. Casali et al. [2011] proposed a graded BDI agent model which allows the agent to reason about uncertain beliefs; Kwisthout et al. [2005] used Dempster-Shafer theory to model uncertainty in an agent's beliefs; Bauters et al. [2014] extended the operational semantics for the agent programming language CAN to deal with uncertain information; and Schut et al. [2001] implemented an intention reconsideration policy as a POMDP. However, these approaches do not address intention progression under uncertainty. The only work on intention progression for BDI agents with uncertain beliefs are the plan selection principles proposed by Ma et al. [2014] and the S_P scheduler proposed by Yao et al. [2016]. However, Ma et al. only considered plan selection, while S_P focussed on scheduling non-deterministic actions rather than uncertain beliefs.

In this paper we introduce $\boldsymbol{SA}_U$, a novel approach to progressing the intentions for BDI agents with uncertain beliefs. SA_U extends the MCTS-based scheduler *SA* [Yao and Logan, 2016]. We choose to extend *SA*, as it is a domain-independent approach compatible with most existing BDI languages, and was shown to outperform other approaches [Yao and Logan, 2016]. In order to extend *SA* to support uncertain beliefs, we revise the structure of the MCTS search tree, including the definition of MCTS nodes and edges, and modify all of the phases of the process in building a search tree. We evaluate the performance of our approach and compare it with first-in-first-out, round-robin and *SA* in synthetic domains with varying levels uncertainty and in both static and dynamic environments. The experimental results suggests SA_U is able to achieve the agent's goals even in settings where there is sig-

nificant uncertainty in the agent's beliefs.

In Section 2 we present preliminaries, including the definitions of beliefs, goals, plans and actions and the intention progression problem. We describe our approach, SA_U, in Section 3. In Section 4 we present an experimental evaluation of SA_U. We discuss related work in Section 5 and conclude with some future directions in Section 6.

2 Preliminaries

In this section, we introduce and define the basic elements of our approach to intention progression under uncertainty, including the model of the agent's beliefs, goals, actions and plans. We also formally define the intention progression problem.

Beliefs. As noted in the Introduction, a wide range of approaches to representing uncertain beliefs in BDI agents have been proposed in the literature [Schut *et al.*, 2001; Kwisthout and Dastani, 2005; Fagundes *et al.*, 2009; Silva and Gluz, 2011; Casali *et al.*, 2011; Ma *et al.*, 2014; Bauters *et al.*, 2014; Coelho and Nogueira, 2015]. In the interests of generality, we assume here a very simple model in which the agent's belief base is a finite set of pairs of the form (l, c) where l is a ground literal (proposition p or its negation $\neg p$) and $c \in [0, 1]$ is a degree of belief or certainty (subjective probability). We assume that at least all the preconditions of actions the agent can perform appear in the belief base with some degree of belief (if the agent is totally uncertain about l, it appears as $(l, 0.5)$). The agent's belief base B is given by:

$$B = \{(l_1, c_1), \ldots, (l_n, c_n)\}$$

If (l, c) is in the agent's state, then the certainty of l is c, and the certainty of its opposite signed literal $\sim l$ is $1 - c$ (we represent beliefs in literals rather than only in atoms for convenience). For example, a belief $(Sunny, 0.9)$ means the agent has certainty 0.9 that it is sunny and certainty 0.1 that it is not sunny. A belief $(l, 1)$ indicates that the agent is certain that l is true, and $(l, 0)$ that l is false. For convenience, we overload notation and denote by $c(l)$ the agent's degree of belief in l, i.e., $c(l) = c$ if $(l, c) \in B$.

The degree of belief in a literal may be updated when the agent receives percepts. We assume the agent's belief update function is of the form $B' = buf(B, percepts)$, where $precepts$ are the percepts at this cycle. Again, in the interests of generality, we make no assumptions about how the agent's beliefs are updated. For example, it may be determined by the readings and reliability of the agent's sensors using Bayes rule: $Pr(l \mid percepts) = \frac{Pr(l) \cdot Pr(precepts|l)}{Pr(percepts)}$. This would require the belief update function to have access to a Bayesian network [Pearl, 1988] with conditional probabilities of percepts given literals in the agent's belief state. However we stress that the availability of conditional or prior probabilities is not necessary for our approach; all we require is an ability to update probabilities in response to receiving percepts.

Goals. The agent's *top-level goals* represent states of affairs that the agent wants to bring about. The agent's goal base G is a finite set of literals:

$$G = \{g_1, \ldots, g_m\}$$

G does not need to be consistent, e.g., conflicting goals may be achieved at different times. A goal g_i is considered *achieved* (and any intention with g_i as top-level goal is dropped) if the agent's degree of belief in g_i exceeds an *achievement threshold* γ, i.e., $(g_i, \gamma') \in B$, where $\gamma' \geq \gamma$. For simplicity, we assume that γ is the same for all goals, but nothing hangs on this.

Actions and Plans. An agent can perform a set of basic actions in the environment. The *preconditions* of an action a are a set of literals ϕ which must be true before the execution of the action, and the *postconditions* of the action are a set of literals ψ that are true after the execution of the action. We assume that actions are deterministic: if the preconditions of an action hold, then the postconditions of the action hold after executing the action and the agent knows this (i.e., the postconditions of the action have certainty 1 after the action has been successfully executed). An action is (believed to be) *executable* given the agent's beliefs B if all literals in ϕ have degree of belief > 0 in B. If the agent attempts to execute an action whose preconditions do not hold, the action *fails* (cannot be executed). We assume that the agent receives a distinguished 'percept', $(\neg)\ done(a)$, at the cycle following execution of the action representing whether the action succeeded or failed. If the action failed, in addition to perceiving $\neg\ done(a)$, the agent may also perceive some subset of the false preconditions as percepts, i.e., as a result of attempting to perform an action, the agent may discover the true value of some preconditions of the action. In the special case when the failed action has a single precondition, we assume its degree of belief is revised to 0.

To achieve the agent's goals, actions are organised into plans. Each goal g is associated with a set of plans $\pi_1, \ldots, \pi_n$ that achieve g. Each plan π_i is of the form $g : \chi \leftarrow a_1; \ldots; a_m$, where χ is a set of literals specifying the *context condition* (or applicability condition) which must be true for π_i to begin execution, and $a_1; \ldots; a_m$ is a sequence of *steps* which are either primitive actions or subgoals.

Intention Progression Problem. The relationships between goals, plans and plan steps naturally forms a tree structure, which is termed a *goal-plan tree* (GPT) [Thangarajah *et al.*, 2003; Thangarajah and Padgham, 2011]. The root of a GPT is a goal node g, its children are plan nodes that can be used to achieve g. Each plan node contains a sequence of action nodes and (sub)goals nodes. The subgoals have their associated plans as their children, giving rise to a tree structure representing all possible ways an agent can achieve the top-level goal g.

The *intentions* of an agent at each deliberation cycle are represented by a pair (T, S) where $T = \{t_1, \ldots, t_n\}$ is a set of goal-plan trees and $S = \{s_1, \ldots, s_n\}$ is a set of *pointers to the current step* of each t_i. The root goal g_i of each $t_i \in T$ corresponds to a top-level goal of the agent. The current step s_i of each t_i is either a primitive action or a (sub-)goal, and is initially set to the root goal of t_i, g_i. We define $next(s_i)$ as the action step of t_i following the current step s_i. If s_i is a primitive action, then $next(s_i)$ is the primitive action or following s_i in the same plan, or, if s_i is the last action in a plan, $next(s_i)$ is the next primitive action in the parent plan

of the current plan. If s_i is a (sub-)goal, determining the next step involves choosing a plan π for the (sub-)goal and returning the first action step in π (if the first step in π is again a subgoal, then we also need to choose a plan to achieve it and so on). That is, $next(s_i)$ is the first action that appears in π or its subplans. A current step s_i is *progressable* if $next(s_i)$ is an action whose precondition holds.

The *intention progression problem* (IPP) [Logan *et al.*, 2017] is that of choosing a current step $s_i \in S$ to progress (i.e., advance to $next(s_i)$) at each deliberation cycle so as to maximise the agent's utility. There are many ways in which utility may be defined, e.g., taking into account the importance or priority of goals, the deadlines by which goals should be achieved, the 'fairness' or order in which goals are achieved, the costs or preferences of actions, etc.. For concreteness, in what follows we assume that the agent's utility is maximised by achieving the largest number of goals. In this setting, the IPP is the problem of choosing which current step to progress so as to maximise the total number of goals achieved by the agent.

3 *SA*$_U$

In this section, we present *SA*$_U$, a solver for intention progression problems where the agent's beliefs are uncertain. *SA*$_U$ extends the *SA* Monte-Carlo Tree Search (MCTS)-based solver of [Yao and Logan, 2016] in explicitly taking account of the agent's uncertainty about the preconditions of an action in the search.

3.1 *SA*$_U$ Search Tree

At each deliberation cycle, *SA*$_U$ advances the current step pointer of one of the agent's intentions to the next step. To choose the action to execute at the current cycle, *SA*$_U$ iteratively builds a search tree. Edges in the tree represent an attempt to execute the next action in one of the agent's intentions. Nodes represent the possible states of the agent following the execution of the action. More precisely, a (non-root) node in the *SA*$_U$ search tree is a pair:

$$(q_t, q_f)$$

where q_t represents the state of the agent when the action executes successfully, and q_f represents the state when the action fails. Each state q_i in a node is a five tuple:

$$(B_i, S_i, P_i, \kappa_i, \nu_i)$$

where B_i is the beliefs of the agent following the execution of the action, S_i are the current steps of each intention (since the agent's goals do not change within a deliberation cycle, we omit T), P_i is the agent's subjective probability of reaching this state from its parent state,[1] κ_i is the number of times this state has been visited in the search, and ν_i is the total simulation value of the state (explained below). As each node contains two different states, an edge connects one state in a node (rather than the node itself) to one of its child nodes as shown in Figure 1. For example, action a_m is an edge between a parent state w_t and a child node n_x, meaning that executing a_m in state w_t may result in either of the two states in node n_x.

[1] An action either succeeds or fails, thus we have $P_t + P_f = 1$.

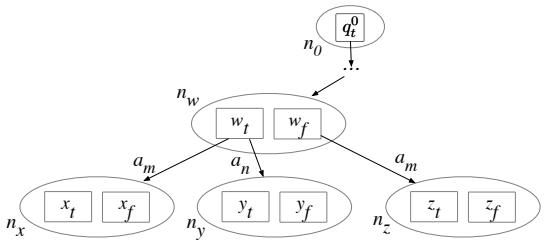

Figure 1: An example *SA*$_U$ search tree.

The root of the search tree is a special node, n_0, which contains only a q_t state representing the beliefs of the agent at the current deliberation cycle.

3.2 *SA*$_U$ Algorithm

As with MCTS and *SA*, the *SA*$_U$ algorithm consists of four main phases: selection, expansion, simulation and backpropagation. However the introduction of uncertain beliefs requires significant modifications to each phase which are described in detail below. *SA*$_U$ is shown in Algorithm 1.

Selection. In the *selection* phase, a leaf node, n_e is selected for expansion (line 4). A node may be expanded if it represents a non-terminal state (a state in which the agent believes it is possible to execute the next step of an intention). n_e is selected using a modified version of Upper Confidence Bounds applied to Trees (UCT) [Kocsis and Szepesvári, 2006], UCT^*, which takes the value of both states q_t and q_f in a node into account when determining the utility or "urgency" of expanding the node. The UCT^* value is given by:

$$UCT^* = \overline{\nu}_t \cdot P_t + \overline{\nu}_f \cdot P_f + C \cdot \sqrt{\frac{\ln \kappa_p}{\kappa_t + \kappa_f}}$$

where $\overline{\nu}_t$ and $\overline{\nu}_f$ are the average simulation value for q_t and q_f respectively, C is a constant that controls the balance between exploitation and exploration, and κ_p is the total number of times the parent of this node has been visited. Together, $\overline{\nu}_t \cdot P_t + \overline{\nu}_f \cdot P_f$ gives the weighted average performance of

Algorithm 1 Return the best action at this cycle

```
1:  function SA_U(B, S, buf, α, β)
2:      n_0 ← ((B, S, 1, 0, 0))
3:      for i from 1 to α do
4:          n_e ← MAX-UCT*-NODE(n_0)
5:          for each state q_i in n_e with P_i > 0 do
6:              children(q_i) ← EXPAND(q_i, buf)
7:              n_s ← RANDOM-CHILD(children(q_i))
8:              for j from 1 to β do
9:                  for each state q_i in n_s do
10:                     value(q_i) ← SIMULATE(q_i, buf)
11:                     BACKUP(value(q_i), q_i)
12:     return BEST-CHILD(n_0)
```

node, which favours the exploitation of the action currently believed to be optimal, and the square-root term represents exploration of a suboptimal action. In the case where $P_t = 1$ and $P_f = 0$ (i.e., the agent is certain that the preconditions of the action represented by the edge to this node are true), the UCT^* value is the same as the classical UCT value.

Starting from the root state (i.e., q_t in the root node), we calculate the UCT^* values for the children of the root state, and select the node which has the largest UCT^* value. If this is not a leaf node, MAX-UCT*-NODE randomly selects a state q_i in the node based on its probability of being reached from the parent node, and continue from the child node of q_i with the highest UCT^* value. This process continues until a leaf node (i.e., its states do not have child nodes) is reached.

Expansion. In the *expansion* phase, all states in the selected node n_e with $P_i > 0$ are expanded by adding child nodes representing the agent's beliefs and the current step of each intention resulting from executing all the next action steps in each intention the agent believes are executable (lines 5-6). Expanding states with $P_i > 0$ ensures that the simulation values of a visited node contains at least one simulation for the state in which the action succeeds and one for the state in which the action fails.

EXPAND generates the child nodes as follows. If the next step of an intention in a state q_i of n_e is an executable action a (i.e., $s_j = a$ for some $s_j \in S_i$) with preconditions ϕ and postconditions ψ, then the execution of a in state q_i of n_e is represented by a child node n' of q_i containing two states, q'_t and q'_f, corresponding to the success and failure of a. In the success state q'_t, both the preconditions ϕ and postconditions ψ of a are believed to be true, and the belief base B'_t in state q'_t of n' is given by:

$$B'_t = \{(l_1, u(c_1)), \ldots, (l_n, u(c_n))\}$$

where

$$u(c_i) = \begin{cases} 1 & \text{if } l_i \in \phi \cup \psi \\ 0 & \text{if } \neg l_i \in \phi \text{ or } \psi \\ c(l_i) & \text{in } B_i \text{ otherwise} \end{cases}$$

the current step in the intention containing a is updated

$$S'_t = S_t \setminus \{s_i\} \cup \{next(a)\}$$

and the probability P'_t of state q'_t being reached is given by:

$$P'_t = \prod_{l_i \in \phi} c(l_i)$$

Note that, in the case where all the preconditions of a are certain, i.e., $P'_t = 1$, $P'_f = 0$, the failure state q'_f cannot be expanded further. In contrast, in the q'_f state representing the failure of a, at least one literal in ϕ does not hold in the current environment, and the belief base is given by $B'_f = buf(B_i, \{\neg done(a)\})$. (We assume that SA_U has no access to the 'false precondition' percepts the agent receives if the attempt to perform a fails, so B_i is updated only with $\neg done(a)$.) In the special case where there is only one uncertain literal l in ϕ, the degree of belief $c(l)$ is updated to 0, and the probability of its negation $c(\neg l)$ is set to 1 in B'_f. If an action a fails, the plan containing the action is also deemed to have failed, and the current step s_j is set to the (sub)goal the plan was selected to achieve. The probability of q'_f being reached is given by $P'_f = 1 - P'_t$.

If the current step of an intention in a state q_i of n_e is a (sub-)goal g, then for each applicable plan π_i for g, a child node representing the selection of π_i is generated. The subjective probability P'_t of π_i being applicable in the current environment is given by $P'_t = \prod_{l_j \in \chi_i} c(l_j)$ where χ_i is the context condition of π_i, and S'_t is set to the first step in π_i.

One of the newly created child nodes, $n_s = (q_t, q_f)$, is then selected at random for simulation (line 7).

Simulation. In the *simulation* phase, the *value* of both states q_t and q_f are estimated (lines 9–11). For each state q_i in n_s with $P_i > 0$ β simulations are performed. Starting in q_i, an executable next step of an intention is randomly selected and executed, and the agent's beliefs and the current step of the selected intention updated. In each simulation, the result of executing an action is randomly selected based on the agent's subjective probability of reaching the success and failure states. As in the expansion phase, if an action succeeds, all its preconditions and postconditions are believed to be true, and the current step is updated accordingly. In the case of execution failure, the agent's belief base is updated by the belief update function buf and the current step of the corresponding intention is set to the (sub)goal the plan was selected to achieve. This process is repeated until a terminal state is reached in which no next steps can be executed or all top-level goals are achieved. The value of the simulation is taken to be the number of top-level goals achieved in the terminal state.

Back-propagation. Finally, all the simulation values for each state generated in the simulation are back-propagated from the simulated state on the path to the initial state in the root node (line 11).

After α iterations, the action that leads to the child of the root node with the greatest κ_i value is returned, and the current step pointer of the selected intention is updated for use at the next deliberation cycle.

4 Evaluating SA_U

In this section, we evaluate the performance of SA_U under varying degrees of uncertainty in the agent's beliefs in both static and dynamic environments. We compare the the number of goals achieved by SA_U, *SA* [Yao and Logan, 2016] and two approaches to intention progression commonly used in practical agent programming languages: first-in-first-out (FIFO) and round-robin (RR). First-in-first-out executes each of the agent's intentions to completion (the goal is achieved or the next step in the intention cannot be executed) before starting to execute the next intention. FIFO minimises interactions between intentions, however it has the disadvantage that the achievement of some goals may be significantly delayed compared to other goals. RR attempts to ensure 'fairness' between intentions by executing a fixed number of steps (typically one) of each intention in turn. However RR increases the number of possible conflicts between intentions: the interleaving of steps in different intentions may destroy a

precondition established by a step in another intention before the action that requires the precondition is executed.

4.1 Experimental Setup

In the interests of generality, we evaluate SA_U using sets of randomly-generated, synthetic goal-plan trees representing the current intentions of an agent in a simple environment. The synthetic trees are similar to those used in the Intention Progression Competition[2] [Logan *et al.*, 2017] and in [Yao and Logan, 2016], except that each action has two literals in its pre-condition. Each goal-plan tree is of depth 5, each goal has two relevant plans, and each plan contains a subgoal and 3 actions.[3]

The agent's environment is built from 60 propositions (corresponding to 120 literals). For each goal-plan tree, we select 30 propositions at random and choose from the corresponding literals pre- and postconditions of the actions in the tree. The values of the literals can change when an action has been successfully executed and its postcondition is applied, or the environment itself changes. Each proposition is modelled as a Poisson process with a specified mean value, allowing the frequency of environmental changes to be controlled. For simplicity, we assume environmental changes always occur after the execution of actions. As a result, the postcondition of an action might be undone by the changes of the environment.

We assume the FIFO, RR and *SA* agents have perfect knowledge of the environment, i.e., their beliefs correspond to the true values of all the literals in the environment and all percepts. In contrast, the SA_U agent has varying degrees of uncertainty about the environment and its percepts. The SA_U agent maintains a set of beliefs which may be more or less at variance with the actual value of the corresponding literals in the environment. We define the 'error' in a belief (l_i, c_i) as:

$$\epsilon(l_i, c_i) = \begin{cases} 1 - c_i & \text{if } l_i \text{ holds,} \\ c_i & \text{otherwise} \end{cases}$$

For example, if a literal l is true in the current environment, but the agent's degree of belief in l is 0.6, i.e., $(l, 0.6)$, then the error in this belief is $1 - 0.6 = 0.4$. For the experiments reported below, the error in each belief in the SA_U agent's initial belief base and percept received from the environment during a trial is sampled from a normal distribution with mean μ and a standard deviation σ. For each trial, we vary the mean 'error' in the agent's degree of belief, holding σ constant at 0.2. The SA_U agent uses a goal achievement threshold $\gamma = 1$, i.e., a goal g is only considered believed when the corresponding belief in g is certain. The only percepts the agent receives are action success and failure. If an action succeeds, *buf* sets the certainty of its pre and postconditions to 1, if it fails and has a single precondition, its certainty is set to 0, if it has two preconditions, their certainty is multiplied by 0.5 (imitating Bayesian reasoning with $\frac{P(\neg done(a)|l_i)}{P(\neg done(a))} = 0.5$).

4.2 Static Environment

Our first set of experiments evaluate the performance of FIFO, RR, SA_U and *SA* in a static environment by setting the mean arrival rate of Poisson process for each environment literal to 0. We generated 50 sets of 10 goal-plan trees, and report the number of goals achieved on average for each approach. *SA* and SA_U were configured to perform 100 iterations ($\alpha = 100$) and 10 simulations per iteration ($\beta = 10$) (the values of α and β used in [Yao and Logan, 2016]).

As a 'baseline', we first evaluated the average number of goals achieved by FIFO, RR, *SA* and SA_U with perfect information about the state of the environment and percepts. This indicates the intrinsic difficulty of the intention progression problem posed by the synthetic trees, and allows comparison with the results in previous work, e.g., [Yao and Logan, 2016].

	RR	FIFO	*SA*	SA_U
#	1.16	6.16	9.58	9.58

Table 1: Av. # goals with perfect information

Table 1 shows the average number of goals achieved for each approach. As can be seen, SA_U and *SA* outperform both FIFO and RR. The results for *SA* are slightly lower than those reported in [Yao and Logan, 2016] due to the increased number of preconditions for each action in the synthetic trees.

μ	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7
#	9.30	9.08	8.66	8.24	7.74	7.30	6.78	6.46

Table 2: Av. # goals achieved by SA_U with increasing error

We then evaluated how the performance of SA_U varies with the degree of error in the agent's beliefs, by varying the mean error μ of the distribution from 0 to 0.7 ($\sigma = 0.2$ in all cases). The results are shown in Table 2. As might be expected, the performance of SA_U declines as the mean error in the agent's beliefs increases. With a mean of 0.7, i.e., most beliefs are incorrect, the performance of SA_U is close to the performance of FIFO. However, as the mean error decreases, the performance of SA_U improves. When the mean is 0, i.e., most beliefs are fairly certain and correct (recall that $\sigma = 0.2$ so even with $\mu = 0$ there is some error in the agent's degree of belief), the performance of SA_U is close to that of *SA*. However, even with a very high error rate, SA_U can still achieve more goals than RR and FIFO.

4.3 Dynamic Environment

Our second set of experiments evaluated the performance of FIFO, RR, *SA* and SA_U in a dynamic environment. In the dynamic case, the mean value for all Poisson processes is set to 0.01, which gives approximately 0.5 literal changes per cycle. For SA_U we again set the standard deviation σ to 0.2 and varied the mean error μ of the distribution from 0 to 0.7. We used the same 50 sets of 10 goal-plan trees as in the static case, and report the number of goals achieved on average for each approach.

As in the static case, we first performed a 'baseline' evaluation of FIFO, RR, *SA* and SA_U with perfect information. The results are shown in Table 3. As can be seen, SA_U and *SA* still outperform both FIFO and RR in the dynamic case, though the dynamic case is significantly harder for all approaches.

[2] https://www.intentionprogression.org/

[3] The trees are available at: https://bit.ly/35jxkt2.

	RR	FIFO	*SA*	SA_U
#	0.58	4.94	7.98	7.98

Table 3: Av. # goals with perfect information

μ	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7
#	7.52	7.22	6.96	6.66	6.38	5.94	5.50	5.10

Table 4: Av. # goals achieved by SA_U with increasing error

The performance of SA_U with varying mean error is shown in Table 4. As in the static case, the performance of SA_U declines as the mean error in the agent's beliefs increases. However the relative number of goals achieved with $\mu = 0.7$ is similar in both cases (approximately 70% of the goals achieved with $\mu = 0$). When the mean error is 0, the performance of SA_U is within 5% of *SA*. In the worst case, when the mean error is 0.7, SA_U still outperforms FIFO and RR.

Overall, SA_U has the same performance as *SA* if perfect information is available, but its performance degrades gracefully as the error in the agent's beliefs increases, and even with significant error, SA_U still outperforms RR and FIFO in terms of the number of goals achieved.

4.4 Computational Overhead

As with *SA*, the computational overhead of SA_U depends on the search configuration α and β, i.e., how many iterations are performed and how many simulations are run from each state. With the search configuration used for the experiments, SA_U requires approximately 300 milliseconds to return the first action to be performed, i.e., to compute a complete interleaving of actions in the 10 goal-plan trees used in the experiments. (As the agent's intentions are progressed, the time required to select an action decreases, as there are fewer actions in the interleaving.)

As SA_U is an anytime algorithm, the computational overhead can be reduced by reducing the value of α and β. In addition, in a static environment, the existing search tree can be reused to improve the efficiency of SA_U, however this is future work.

5 Related Work

There is a body of work on modelling uncertainty in BDI languages, for example [Schut *et al.*, 2001; Kwisthout and Dastani, 2005; Fagundes *et al.*, 2009; Silva and Gluz, 2011; Casali *et al.*, 2011; Ma *et al.*, 2014; Bauters *et al.*, 2014; Coelho and Nogueira, 2015]. In some approaches, Bayesian or Dempster-Shafer probabilities are included in the agent's belief base, e.g., [Kwisthout and Dastani, 2005; Silva and Gluz, 2011], while other approaches separate the symbolic BDI cycle from the Bayesian update of percepts (where the update function uses HMM internally but returns the most probable percept as a symbolic element of belief base). However none of this work has addressed the problem of intention progression under uncertainty.

MCTS with imperfect information has been mostly explored in the context of games where imperfect information arises from uncertainty about the opponent's actions, e.g., [Bitan and Kraus, 2018; Browne *et al.*, 2012]. This differs from our setting, where there is no adversarial opponent and the uncertainty arises from incomplete information about the current state of the environment. MCTS has also been applied in domains with non-deterministic actions. In such settings, determinisation and sampling possible outcomes of determinised actions is often used. As in our approach, nodes may have several children corresponding to the same move [Bjarnason *et al.*, 2009], however selection of child nodes is random and does not depend on probability values. Another approach to dealing with uncertainty in MCTS is based on information sets [Whitehouse *et al.*, 2011]; however such approaches fail to outperform approaches using determinisation [Browne *et al.*, 2012]. Bayesian approaches to MCTS [Tesauro *et al.*, 2010] have also been proposed.

6 Conclusion and Future Work

In this paper we presented SA_U, an MCTS-based solver for intention progression problems where the agent's beliefs are uncertain. We evaluated the performance of SA_U in both static and dynamic environments using sets of randomly-generated, synthetic goal-plan trees. Our preliminary results suggest that the performance of SA_U is close to that of perfect information *SA* if the error in beliefs is small, and performance degrades gracefully as the error in the agent's beliefs increases. However, even where there is significant uncertainty in the agent's beliefs, SA_U still outperforms RR and FIFO in terms of the number of goals achieved.

One limitation of SA_U is that it doesn't consider nondeterministic actions which have several possible outcomes as in [Yao *et al.*, 2016]. In addition to an action failing when its preconditions do not hold, an action may fail if it achieves an undesired postcondition. However, we believe it is straightforward to extend SA_U to deal with nondeterministic actions by increasing the number of states in an MCTS node. Another limitation of SA_U is that it assumes the environment is static, i.e., it does not consider the evolution of the environment when building the search tree. Another potential direction for future work is to incorporate a simple environment model into the current solver to allow the prediction of likely environment changes in the expansion and simulation phases.

Acknowledgements

This work was supported by Zhejiang Provincial Natural Science Foundation of China (Q19F030028) and National Natural Science Foundation of China (61906169).

References

[Bauters *et al.*, 2014] Kim Bauters, Weiru Liu, Jun Hong, Carles Sierra, and Lluis Godo. CAN(PLAN)+: extending the operational semantics of the BDI architecture to deal with uncertain information. In *30th Conference on Uncertainty in Artificial Intelligence*, pages 52–61. AUAI Press, 2014.

[Bitan and Kraus, 2018] Moshe Bitan and Sarit Kraus. Combining prediction of human decisions with ISMCTS in imperfect information games. In *17th International Con-*

ference on Autonomous Agents and MultiAgent Systems, pages 1874–1876. IFAAMAS/ACM, 2018.

[Bjarnason *et al.*, 2009] Ronald Bjarnason, Alan Fern, and Prasad Tadepalli. Lower bounding Klondike Solitaire with Monte-Carlo Planning. In *19th International Conference on Automated Planning and Scheduling*. AAAI, 2009.

[Browne *et al.*, 2012] Cameron Browne, Edward Jack Powley, Daniel Whitehouse, and et al. A survey of Monte Carlo Tree Search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[Casali *et al.*, 2011] Ana Casali, Lluis Godo, and Carles Sierra. A graded BDI agent model to represent and reason about preferences. *Artificial Intelligence*, 175(7-8):1468–1478, 2011.

[Coelho and Nogueira, 2015] Francisco Coelho and Vítor Nogueira. Probabilistic perception revision in agentspeak(l). In *PRIMA 2015: Principles and Practice of Multi-Agent Systems*, volume 9387 of *LNCS*, pages 613–621. Springer, 2015.

[Fagundes *et al.*, 2009] Moser Silva Fagundes, Rosa Maria Vicari, and Helder Coelho. Deliberation process in a BDI model with bayesian networks. In *PRIMA 2009: Principles and Practice of Multi-Agent Systems*, volume 5044 of *LNCS*, pages 207–218. Springer, 2009.

[Kocsis and Szepesvári, 2006] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *17th European Conference on Machine Learning*, pages 282–293. Springer, 2006.

[Kwisthout and Dastani, 2005] Johan Kwisthout and Mehdi Dastani. Modelling uncertainty in agent programming. In *3rd International Workshop on Declarative Agent Languages and Technologies*, volume 3904 of *LNCS*, pages 17–32. Springer, 2005.

[Logan *et al.*, 2017] Brian Logan, John Thangarajah, and Neil Yorke-Smith. Progressing intention progression: A call for a goal-plan tree contest. In *16th Conference on Autonomous Agents and MultiAgent Systems*, pages 768–772. IFAAMAS, 2017.

[Ma *et al.*, 2014] Jianbing Ma, Weiru Liu, Jun Hong, Lluis Godo, and Carles Sierra. Plan selection for probabilistic BDI agents. In *26th IEEE International Conference on Tools with Artificial Intelligence*, pages 83–90. IEEE, 2014.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[Pěchouček and Mařík, 2008] Michal Pěchouček and Vladimír Mařík. Industrial deployment of multi-agent technologies: Review and selected case studies. *Autonomous Agents and Multi-Agent Systems*, 17:397–431, 2008.

[Rao and Georgeff, 1992] Anand S. Rao and Michael P. Georgeff. An abstract architecture for rational agents. In *3rd International Conference on Principles of Knowledge Representation and Reasoning*, pages 439–449. Morgan Kaufmann, 1992.

[Schut *et al.*, 2001] Martijn C. Schut, Michael J. Wooldridge, and Simon Parsons. Reasoning about intentions in uncertain domains. In *6th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 2143 of *LNCS*, pages 84–95. Springer, 2001.

[Silva and Gluz, 2011] Diego Goncalves Silva and Joao Carlos Gluz. AgentSpeak(PL): A new programming language for BDI agents with integrated bayesian network model. In *International Conference on Information Science and Applications*. IEEE, 2011.

[Tesauro *et al.*, 2010] Gerald Tesauro, V. T. Rajan, and Richard Segal. Bayesian inference in Monte-Carlo Tree Search. In *26th Conference on Uncertainty in Artificial Intelligence*, pages 580–588. AUAI Press, 2010.

[Thangarajah and Padgham, 2011] John Thangarajah and Lin Padgham. Computationally Effective Reasoning About Goal Interactions. *Journal of Automated Reasoning*, 47(1):17–56, 2011.

[Thangarajah *et al.*, 2003] John Thangarajah, Lin Padgham, and Michael Winikoff. Detecting & Avoiding Interference Between Goals in Intelligent Agents. In *18th International Joint Conference on Artificial Intelligence*, pages 721–726. Morgan Kaufmann, 2003.

[Waters *et al.*, 2014] Max Waters, Lin Padgham, and Sebastian Sardina. Evaluating coverage based intention selection. In *13th International Conference on Autonomous Agents and Multi-agent Systems*, pages 957–964. IFAAMAS, 2014.

[Waters *et al.*, 2015] Max Waters, Lin Padgham, and Sebastian Sardiña. Improving domain-independent intention selection in BDI systems. *Autonomous Agents and Multi-Agent Systems*, 29(4):683–717, 2015.

[Whitehouse *et al.*, 2011] Daniel Whitehouse, Edward Jack Powley, and Peter I. Cowling. Determinization and information set Monte Carlo Tree Search for the card game Dou Di Zhu. In *IEEE Conference on Computational Intelligence and Games*, pages 87–94. IEEE, 2011.

[Yao and Logan, 2016] Yuan Yao and Brian Logan. Action-level intention selection for BDI agents. In *15th International Conference on Autonomous Agents and Multiagent Systems*, pages 1227–1236. IFAAMAS, 2016.

[Yao *et al.*, 2014] Yuan Yao, Brian Logan, and John Thangarajah. SP-MCTS-based intention scheduling for BDI agents. In *21st European Conference on Artificial Intelligence*, pages 1133–1134. IOS Press, 2014.

[Yao *et al.*, 2016] Yuan Yao, Brian Logan, and John Thangarajah. Robust execution of BDI agent programs by exploiting synergies between intentions. In *30th AAAI Conference on Artificial Intelligence*, pages 2558–2565. AAAI Press, 2016.