A Hybrid POMDP-BDI Agent Architecture with Online Stochastic Planning and Desires with Changing Intensity Levels

Gavin Rens[1,2(✉)] and Thomas Meyer[2,3]

[1] School of Mathematics, Statistics and Computer Science,
University of KwaZulu-Natal, Durban, South Africa
grens@csir.co.za
[2] Centre for Artificial Intelligence Research, CSIR Meraka, Pretoria, South Africa
[3] Department of Computer Science, University of Cape Town,
Cape Town, South Africa
tmeyer@cs.uct.ac.za

Abstract. We propose an agent architecture which combines Partially observable Markov decision processes (POMDPs) and the belief-desire-intention (BDI) framework to capitalize on their complimentary strengths. Our architecture introduces the notion of intensity of the desire for a goal's achievement. We also define an update rule for goals' desire levels. When to select a new goal to focus on is also defined. To verify that the proposed architecture works, experiments were run with an agent based on the architecture, in a domain where multiple goals must continually be achieved. The results show that (i) while the agent is pursuing goals, it can concurrently perform rewarding actions not directly related to its goals, (ii) the trade-off between goals and preferences can be set effectively and (iii) goals and preferences can be satisfied even while dealing with stochastic actions and perceptions. We believe that the proposed architecture furthers the theory of high-level autonomous agent reasoning.

Keywords: POMDP · BDI · Online planning · Desire intensity · Preference

1 Introduction

Imagine a scenario where a planetary rover has four main tasks and one task it can do when it does not interfere with performing the main tasks. The main tasks could be, for instance, collecting gas (for industrial use) from a natural vent at the base of a hill, taking a temperature measurement at the top of the hill, performing self-diagnostics and repairs, and reloading its batteries at the solar charging station. The less important task is to collect soil samples wherever the rover is. The rover is programmed to know the relative importance of collecting soil samples. The rover also has a model of the probabilities with which its

B. Duval et al. (Eds.): ICAART 2015, LNAI 9494, pp. 3–19, 2015.
DOI: 10.1007/978-3-319-27947-3_1

various actuators fail and the probabilistic noise-profile of its various sensors. The rover must be able to reason (plan) in real-time to pursue the right task at the right time while considering its resources and dealing with unforeseen events, all while considering the uncertainties about its actions (actuators) and perceptions (sensors).

We propose an architecture for the proper control of an agent in a complex environment such as the scenario described above. The architecture combines belief-desire-intention (BDI) theory [1,2] and partially observable Markov decision processes (POMDPs) [3,4]. Traditional BDI architectures (BDIAs) cannot deal with probabilistic uncertainties and they do not generate plans in real-time. A traditional POMDP cannot manage goals (major and minor tasks) as well as BDIAs can. Next, we analyse the POMDPs and BDIAs in a little more detail.

One of the benefits of agents based on BDI theory, is that they need not generate plans from scratch; their plans are already (partially) compiled, and they can act quickly once a goal is focused on. Furthermore, the BDI framework can deal with multiple goals. However, their plans are usually not optimal, and it may be difficult to find a plan which is applicable to the current situation. That is, the agent may not have a plan in its library which exactly 'matches' what it ideally wants to achieve. On the other hand, POMDPs can generate optimal policies on the spot to be highly applicable to the current situation. Moreover, policies account for stochastic actions and partially observable environments. Unfortunately, generating optimal POMDP policies is usually intractable. One solution to the intractability of POMDP policy generation is to employ a *continuous planning* strategy, or *agent-centred search* [5]. Aligned with agent-centred search is the *forward-search* approach or *online* planning approach in POMDPs [6].

The traditional BDIA maintains goals as *desires*; there is no reward for performing some action in some state. The reward function provided by POMDP theory is useful for modeling certain kinds of behavior or preferences. For instance, an agent based on a POMDP may want to avoid moist areas to prevent its parts becoming rusty. Moreover, a POMDP agent can generate plans which can optimally avoid moist areas. But one would not say that avoiding moist areas is the agent's goal. And POMDP theory maintains a single reward function; there is no possibility of weighing alternative reward functions and pursuing one at a time for a fixed period—all objectives must be considered simultaneously, in one reward function. Reasoning about objectives in POMDP theory is not as sophisticated as in BDI theory. A BDI agent cannot, however, simultaneously avoid moist areas *and* collect gold; it has to switch between the two or combine the desire to avoid moist areas with every other goal.

We argue that maintenance goals like avoiding moist areas (or collecting soil samples) should rather be viewed as a *preference* and modeled as a POMDP reward function. And specific tasks to complete (like collecting gas or keeping its battery charged) should be modeled as BDI desires.

Given the advantages of POMDP theoretic reasoning and the potentially sophisticated means-ends reasoning of BDI theory, we propose to combine the best features of these two theories in a coherent agent architecture. We call it the Hybrid POMDP-BDI agent architecture (or HPB architecture, for short).

In BDI theory, one of the big challenges is to know *when* the agent should switch its current goal and *what* its new goal should be [7]. To address this challenge with an intuitive explanation, we propose that an agent should maintain intensity levels of desire for every goal. (This intensity of desire could be interpreted as a kind of emotion.) The goal most intensely desired should be the current goal sought (the intention). We also define the notion of how much an intention is satisfied in the agent's current belief-state.

Typically, BDI agents do not deal with stochastic uncertainty. Integrating POMDP notions into a BDIA addresses this. For instance, an HPB agent will maintain a (subjective) belief-state representing its probabilistic (uncertain) belief about its current state. Planning with models of stochastic actions and perceptions is thus possible in the proposed architecture. The tight integration of POMDPs and BDIAs is novel, especially in combination with desires with changing intensity levels.

Section 2 briefly reviews the necessary theory. The proposed agent architecture is presented in Sect. 3 and formally defined. Section 4 shows an implementation of the architecture on an example domain and evaluates the performance on various dimensions, confirming that the approach may be useful in some domains. In Sect. 5, we propose one approach to making the specification of goals and preferences more general or flexible. The last section discusses some related work and points out some future directions for research in this area.

2 Preliminaries

The basic components of a BDI architecture [8,9] are

- a set or knowledge-base B of beliefs;
- an option generation function 'wish', generating the objectives the agent would ideally like to pursue (its desires);
- a set of desires D (goals to be achieved);
- a 'focus' function which selects intentions from the set of desires;
- a structure of intentions I of the most desirable options/desires returned by the focus function;
- a library of plans and subplans;
- a 'reconsideration' function which decides whether to call the focus function;
- an execution procedure, which affects the world according to the plan associated with the intention;
- a sensing or perception procedure, which gathers information about the state of the environment; and
- a belief update function, which updates the agent's beliefs according to its latest observations and actions.

Exactly how these components are implemented result in a particular BDI architecture.

Algorithm 1 (adapted from [10, Fig. 2.3]) is a basic BDI agent control loop. π is the current plan to be executed. $getPercept(\cdot)$ senses the environment and

Algorithm 1. Basic BDI agent control loop.

Input: B_0: initial beliefs
Input: I_0: initial intentions
$B \leftarrow B_0$;
$I \leftarrow I_0$;
$\pi \leftarrow null$;
while *alive* **do**
  $p \leftarrow getPercept()$;
  $B \leftarrow update(B, p)$;
  $D \leftarrow wish(B, I)$;
  $I \leftarrow focus(B, D, I)$;
  $\pi \leftarrow plan(B, I)$;
  $execute(\pi)$;

Algorithm 2. Control loop for an agent with reconsideration.

Input: B_0: initial beliefs
Input: I_0: initial intentions
$B \leftarrow B_0$;
$I \leftarrow I_0$;
$\pi \leftarrow null$;
while *alive* **do**
  $p \leftarrow getPercept()$;
  $B \leftarrow update(B, p)$;
  if $reconsider(B, I)$ **then**
    $D \leftarrow wish(B, I)$;
    $I \leftarrow focus(B, D, I)$;
    if *not* $sound(\pi, I, B)$ **then** $\pi \leftarrow plan(B, I)$
  if *not* $empty(\pi)$ **then**
    $\alpha \leftarrow head(\pi)$;
    $execute(\alpha)$;
    $\pi \leftarrow tail(\pi)$;
  $I \leftarrow succeeded(I, B)$;
  $I \leftarrow impossible(I, B)$;

returns a percept (processed sensor data) which is an input to $update(\cdot)$. $plan(\cdot)$ returns a plan from the plan library to achieve the agent's current intentions. $wish : B \times I \rightarrow D$ generates a set of desires, given the agent's beliefs, current intentions and possibly its innate motives. It is usually impractical for an agent to pursue the achievement of all its desires. It must thus filter out the most valuable and achievable desires. This is the function of $focus : B \times D \times I \rightarrow I$, taking beliefs, desires and current intentions as parameters. Together, the processes performed by *wish* and *focus* may be called deliberation, formally encapsulated by the *deliberate* procedure.

Algorithm 2 (adapted from [11]) has some more sophisticated controls. It controls when the agent would consider *whether* to re-deliberate, with the *reconsider* function (line 7) placed just before deliberation would take place. *reconsider*$(\cdot)$ is a Boolean function which tells the agent *whether* to reconsider its intentions (every time line 7 is reached).

The agent tests at every iteration through the main loop whether the currently pursued intention is still possibly achievable, using *impossible*$(\cdot)$. In the algorithm, serendipity is also taken advantage of by periodically testing—using *succeeded*$(\cdot)$—whether the intention has been achieved, without the plan being fully executed. This agent is considered 'reactive' because it executes one action per loop iteration; this allows for deliberation between executions. The soundness (or applicability) of the plan to achieve the current intention is checked at every iteration of the loop.

There are various mechanisms which an agent might use to decide when to reconsider its intentions. See, for instance, [1,7,12–16].

In a partially observable Markov decision process (POMDP), the actions the agent performs have non-deterministic effects in the sense that the agent can only predict with a likelihood in which state it will end up after performing an action. Furthermore, its perception is noisy. That is, when the agent uses its sensors to determine in which state it is, it will have a probability distribution over a set of possible states to reflect its conviction for being in each state.

Formally [17], a POMDP is a tuple $\langle S, A, T, R, Z, P, b^0 \rangle$ with

- S, a finite set of states of the world (that the agent can be in),
- A a finite set of actions (that the agent can choose to execute),
- a transition function $T(s, a, s')$, the probability of being in s' after performing action a in state s,
- $R(a, s)$, the immediate reward gained for executing action a while in state s,
- Z, a finite set of observations the agent can perceive in its world,
- a perception function $P(s', a, z)$, the probability of observing z in state s' resulting from performing action a in some other state, and
- b^0 the initial probability distribution over all states in S.

A belief-state b is a set of pairs $\langle s, p \rangle$ where each state s in b is associated with a probability p. All probabilities must sum up to one, hence, b forms a probability distribution over the set S of all states. To update the agent's beliefs about the world, a special function $SE(z, a, b) = b_n$ is defined as

$$b_n(s') = \frac{P(s', a, z) \sum_{s \in S} T(s, a, s') b(s)}{Pr(z|a, b)}, \tag{1}$$

where a is an action performed in 'current' belief-state b, z is the resultant observation and $b_n(s')$ denotes the probability of the agent being in state s' in 'new' belief-state b_n. Note that $Pr(z \mid a, b)$ is a normalizing constant.

Let the *planning horizon* h (also called the *look-ahead depth*) be the number of future steps the agent plans ahead each time it plans. $V^*(b, h)$ is the *optimal* value of future courses of actions the agent can take with respect to a finite

horizon h starting in belief-state b. This function assumes that at each step the action that will maximize the state's value will be selected.

Because the reward function $R(a, s)$ provides feedback about the utility of a particular state s (due to a executed in it), an agent who does not know in which state it is in cannot use this reward function directly. The agent must consider, for each state s, the probability $b(s)$ of being in s, according to its current belief-state b. Hence, a *belief* reward function $\rho(a, b)$ is defined, which takes a belief-state as argument. Let $\rho(a, b) \stackrel{def}{=} \sum_{s \in S} R(a, s)b(s)$.

The optimal *state-value* function is define by

$$V^*(b, h) \stackrel{def}{=} \max_{a \in \mathcal{A}} \Big[\rho(a, b) + \gamma \sum_{z \in Z} Pr(z \,|\, a, b)V^*(SE(z, a, b), h - 1)\Big],$$

where $0 \leq \gamma < 1$ is a factor to discount the value of future rewards and $Pr(z \,|\, a, b)$ denotes the probability of reaching belief-state $b_n = SE(z, a, b)$. While V^* denotes the optimal value of a belief-state, function Q^* denotes the optimal *action-value*:

$$Q^*(a, b, h) \stackrel{def}{=} \rho(a, b) + \gamma \sum_{z \in Z} Pr(z \,|\, a, b)V^*(SE(z, a, b), h - 1)$$

is the value of executing a in the current belief-state, plus the total expected value of belief-states reached thereafter.

3 The HPB Architecture

A hybrid POMDP-BDI (HPB) agent maintains (i) a belief-state which is periodically updated, (ii) a mapping from goals to numbers representing the level of desire to achieve the goals, and (iii) the current intention, the goal with the highest desire level. As the agent acts, its desire levels are updated and it may consider choosing a new intention based on new desire levels.

The *state* of an HPB agent is defined by the tuple $\langle B, D, I\rangle$, where B is the agent's current belief-state (i.e., a probability distribution over the states S, defined below), D is the agent's current desire function and I is the agent's current intention. We'll have more to say about D and I a little later.

An HPB agent could be defined by the tuple $\langle Atrb, G, A, Z, T, P, Util\rangle$, where

- *Atrb* is a set of attribute-sort pairs (for short, the *attribute set*). For every $(atrb : sort) \in Atrb$, *atrb* is the name or identifier of an attribute of interest in the domain of interest, like *BattryLevel* or *Direction*, and *sort* is the set from which *atrb* can take a value, for instance, real numbers in the range $[0, 55]$ or a list of values like $\{North, East, West, South\}$. So $\{(BattryLevel : [0, 55]), (Direction : \{North, East, West, South\})\}$ could be an attribute set.

  Let $\mathcal{N} = \{atrb \mid (atrb : sort) \in Atrb\}$ be the set of all attribute names. We define a state s induced from $Atrb$ as one possible way of assigning values to attributes: $s = \{(atrb : v) \mid atrb \in \mathcal{N}, (atrb : sort) \in Atrb, v \in sort\}$ such that if $(atrb : v), (atrb' : v') \in s$ and $atrb = atrb'$, then $v = v'$. The set of all possible states is denoted S.

- G is a set of goals. A goal $g \in G$ is a subset of some state $s \in S$. For instance, $\{(BattryLevel : 13), (Direction : South)\}$ is a goal, and so are $\{(BattryLevel : 33)\}$ and $\{(Direction : West)\}$. It is even possible to have one goal overlap or be a subset of another goal. For instance, one is allowed to have $\{(BattryLevel : 13), (Direction : South)\} \in G$ and simultaneously $\{(BattryLevel : 13)\}$, $\{(BattryLevel : 14), (Direction : South)\} \in G$. In this architecture, it is assumed that the set of goals is given.
- A is a finite set of actions.
- Z is a finite set of observations.
- T is the transition function of POMDPs.
- P is the perception function of POMDPs.
- *Util* consists of two functions *Pref* and *Satf* which allow an agent to determine the utilities of alternative sequences of actions. $Util = \langle Pref, Satf \rangle$.

Pref is the preference function with a range in $\mathbb{R} \cap [0, 1]$. It takes an action a and a state s, and returns a value reflecting the preference for performing a in s. That is, $Pref(a, s) \in [0, 1]$. Numbers closer to 1 imply greater preference and numbers closer to 0 imply less preference. Except for the range restriction of $[0, 1]$, it has the same definition as a POMDP reward function, but its name indicates that it models the agent's preferences and not what is typically thought of as rewards. An HPB agent gets 'rewarded' by achieving its goals. The preference function is especially important to model action costs; the agent should prefer 'inexpensive' actions. *Pref* has a local flavor. Designing the preference function to have a value lying in [0,1] may sometimes be challenging, but we believe it is always possible.

Satf is the satisfaction function with a range in $\mathbb{R} \cap [0, 1]$. It takes a state s and an intention I, and returns a value representing the degree to which the state satisfies the intention. That is, $Satf(I, s) \in [0, 1]$. It is completely up to the agent designer to decide how the satisfaction function is defined, as long as numbers closer to 1 mean more satisfaction and numbers closer to 0 mean less satisfaction. *Satf* has a global flavor.

The desire function D is a total function from goals in G into the positive real numbers $\mathbb{R}^+$. The real number represents the intensity or level of desire of the goal. For instance, $(\{(BattryLevel : 13), (Direction : South)\}, 2.2)$ could be in D, meaning that the goal of having the battery level at 13 and moving in a southerly direction is desired with a level of 2.2. $(\{(BattryLevel : 33)\}, 56)$ and $(\{(Direction : West)\}, 444)$ are also examples of desires in D.

I is the agent's current intention; an element of G; the goal with the highest desire level. This goal will be actively pursued by the agent, shifting the importance of the other goals to the background. The fact that only one intention is maintained makes the HPB agent architecture quite different to standard BDIAs.

Figure 1 shows a flow diagram representing the operational semantics of the HPB architecture.

The satisfaction an agent gets for an intention in its current belief-state is defined as

$$Satf_{\beta}(I,B) \stackrel{def}{=} \sum_{s \in S} Satf(I,s)B(s),$$

where $Satf(I,s)$ is defined above and $B(s)$ is the probability of being in state s. The definition of $Pref_{\beta}$ has the same form as the reward function ρ over belief-states in POMDP theory:

$$Pref_{\beta}(a,B) \stackrel{def}{=} \sum_{s \in S} Pref(a,s)B(s),$$

where $Pref(a,s)$ was discussed above.

We propose the following desire update rule.

$$D(g) \leftarrow D(g) + 1 - Satf_{\beta}(g,B) \tag{2}$$

Rule 2 is defined so that as $Satf_{\beta}(g,B)$ tends to one (total satisfaction), the intensity with which the incumbent goal is desired does not increase. On the other hand, as $Satf_{\beta}(g,B)$ becomes smaller (more dissatisfaction), the goal's intensity is incremented. The rule transforms D with respect to B and g. A goal's intensity should drop the more it is being satisfied. The update rule thus defines how a goal's intensity changes over time with respect to satisfaction.

Note that desire levels never decrease. This does not reflect reality. It is however convenient to represent the intensity of desires like this: only *relative* differences in desire levels matter in our approach and we want to avoid unnecessarily complicating the architecture.

An HPB agent controls its behaviour according to the policies it generates. *Plan* is a procedure which generates a POMDP policy π of depth h. Essentially, we want to consider all action sequences of length h and the belief-states in which the agent would find itself if it followed the sequences. Then we want to choose the sequence (or at least its first action) which yields the highest preference and which ends in the belief-state most satisfying with respect to the intention.

During planning, preferences and intention satisfaction must be maximized. The main function used in the *Plan* procedure is the HPB action-value function Q^*_{HPB}, giving the value of some action a, conditioned on the current belief-state B, intention I and look-ahead depth h:

$$\begin{aligned} Q^*_{HPB}(a,B,I,h) &\stackrel{def}{=} \alpha Satf_{\beta}(I,B) + (1-\alpha)Pref_{\beta}(a,B) \\ &\quad + \gamma \sum_{z \in Z} Pr(z \mid a,B) \max_{a' \in A} Q^*_{HPB}(a',B',I,h-1), \\ Q^*_{HPB}(a,B,I,1) &\stackrel{def}{=} \alpha Satf_{\beta}(I,B) + (1-\alpha)Pref_{\beta}(a,B), \end{aligned}$$

where $B' = SE(a,z,B)$, $0 \leq \alpha \leq 1$ is the goal/preference 'trade-off' factor, γ is the normal POMDP discount factor and SE is the normal POMDP state estimation function. To keep things simple for this introductory paper, we define

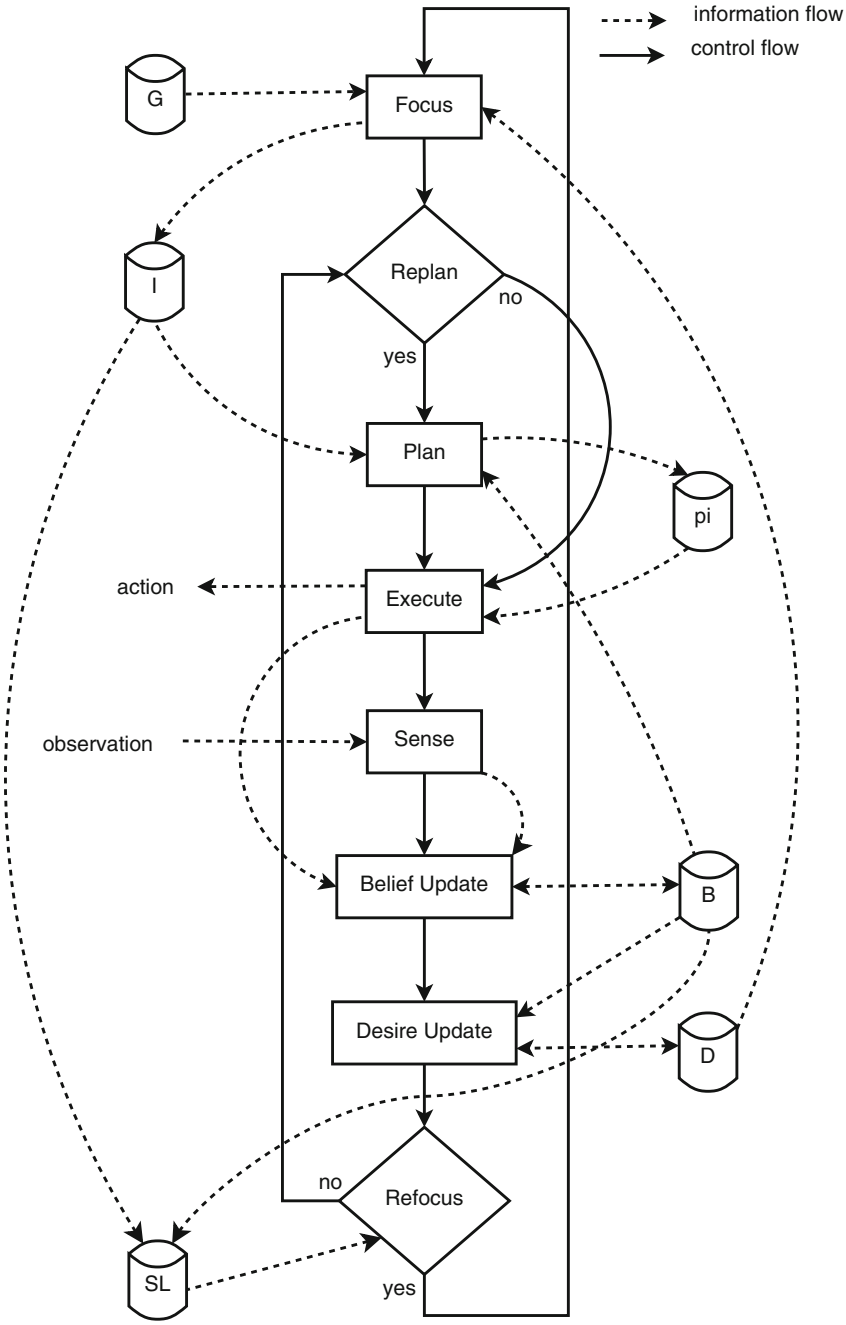

Fig. 1. Operational semantics of the HPB architecture. SL stands for *Satf_levels*. Note that *Satf_levels* depends on the current belief-state and intention, but not on desire levels. Planning is also independent of desire levels. The focus function depends on desire levels, but not on satisfaction. Whether to refocus depends on satisfaction levels, but not on desire levels.

Plan to return arg $\max_{a \in A} Q^*_{HPB}(a, B, I, h)$, the trivial policy of a single action. In general, *Plan* could return a policy of depth h, that is, a sequence of h actions, where the choice of exactly which action to take at each step depends on the observation received just prior.

Focus is a function which returns one member of G called the (current) intention I. Presently, we define it simply as selecting the goal with the highest desire level. After every execution of an action in the real-world, *Refocus* is called to decide whether to call *Focus* to select a new intention. *Refocus* is a meta-reasoning function analogous to the *reconsider* function discussed in Sect. 2. It is important to keep the agent focused on one goal long enough to give it a reasonable chance of achieving it. It is the job of *Refocus* to recognize when the current intention seems impossible or too expensive to achieve.

Let *Satf_levels* be the sequence of satisfaction levels of the current intention since it became active and let *MEMORY* be a designer-specified number representing the length of a sub-sequence of *Satf_levels*—the *MEMORY* last satisfaction levels. One possible definition of *Refocus* is

$$Refocus(c, \theta) \stackrel{def}{=} \begin{cases} \text{'no' if } |Satf\_levels| < MEMORY \\ \text{'yes' if } c < \theta \\ \text{'no' otherwise,} \end{cases}$$

where c is the average change from one satisfaction level to the next in the agent's 'MEMORY', and θ is some threshold chosen by the agent designer. If the agent is expected to increase its satisfaction by at least, say, 0.1 on average for the current intention, then θ should be set to 0.1. With this approach, if the agent 'gets stuck' trying to achieve its current intention, it will not blindly keep on trying to achieve it, but will start pursuing another goal (with the highest desire level). Note that if an intention was not well satisfied, its desire level still increases at a relatively high rate. So whenever the agent focuses again, a goal not well satisfied in the past will be a top contender to become the intention (again).

4 Evaluation

We performed some tests on an HPB agent in a six-by-six grid-world. In this world, the agent's task is to visit each of the four corners, while collecting items on the way. That is, the agent's goals are the states representing the four corners, but the collecting of items is regarded as a preferred behavior, not a goal to be pursued.

States are quadruples $\langle x, y, d, t \rangle$, with $x, y \in \{1, \cdots, 6\}$ being the coordinates of the agent's position in the world, $d \in \{North, East, West, South\}$ the direction it is facing, and $t \in \{0, 1\}$, $t = 1$ if an item is present in the cell with the agent, else $t = 0$. The agent can perform five actions $\{left, right, forward, see, collect\}$, meaning, turn left, turn right, move one cell forward, see whether an item is present and collect an item. The only observation possible when executing one

of the physical actions is *obsNil*, the null observation, and *see* has possible observations from the set $\{0, 1\}$ for whether the agent sees the presence of an item (1) or not (0).

Next, we define the possible outcomes for each action: When the agent turns left or right, it can get stuck in the same direction, turn 90° or overshoots by 90°. When the agent moves forward, it moves one cell in the direction it is facing or it gets stuck and does not move. The agent can see an item or see nothing (no item in the cell), and collecting is deterministic (if there is an item present, it will be collected with certainty, if the agent executes *collect*). All actions except *collect* are designed so that the correct outcome is achieved 95 % of the time and incorrect outcomes are achieved 5 % of the time.

So that the agent does not get lost too quickly, we have included an automatic localization action, that is, a sensing action returns information about the agent's approximate location. The action is automatic because the agent cannot choose whether to perform it; the agent localizes itself after every regular/chosen action is executed. However, just as with regular actions, the localization sensor is noisy, and it correctly reports the agent's location with probability 0.95, else the sensor reports a location adjacent to the agent with probability uniformly distributed over 0.05.

Errors in the agent's actions and perceptions are thus modeled, not ignored.

In the experiments which follow, the threshold θ is set to 0.05, *MEMORY* is set to 5 and $h = 4$. Desire levels are initially set to zero for all goals. Four experiments were performed. First, collecting items but not intentionally visiting corners, second and third, visiting corners while collecting items (with different values for the goal/preference 'trade-off' factor), and fourth, visiting corners but not collecting items. For each experiment, 10 trials were run with the agent starting in random locations and performing 100 actions per trial. We let $Satf(I, s) = 1 - dist/10$ where 10 is the maximum Manhattan distance between two cells in the world and *dist* is the Manhattan distance between the cells represented by I and s, and we let

$$Pref(a, s) = (1 - dist/10 + collUtil + sensUtil)/100,$$

where *dist* is the Manhattan distance between the cell representing s and the closest cell containing an item, *collUtil* is 98 if a is *collect* and there is actually an item in the cell represented by s, else 0, and *sensUtil* is 1 if the agent tries to *see*, else 0.[1] The division by 100 is to bring the value of $Pref(\cdot)$ within the limits of 0 and 1.

First, we see how an HPB agent behaves when it has no goal state ($\alpha = 0$), but continually only 'prefers' to collect items. That is, we let

$$Q^*_{HPB}(a, B, I, h) \stackrel{def}{=} Pref_\beta(a, B) + \gamma \sum_{z \in Z} Pr(z \mid a, B) \max_{a' \in A} Q^*_{HPB}(a', B', I, h - 1).$$

[1] $Pref(\cdot)$ is designed such that the agent collects a maximum number of items (ignoring goals). The agent collects more when it is encouraged to sense where items are, hence *sensUtil* is 1 if the agent tries to *see*.

On average, it collects 7.4 of 12 possible items. The left-most results column of Table 1 shows how often corners are (unintentionally) visited.

Next, if the HPB agent prefers to collect items *while* equally trying to reach corners ($\alpha = 0.5$), it collects 4.3 of 12 possible items and the corners it visits is summarized in the second-from-left results column of Table 1.

Table 1. The average number of times each corner was visited (on separate occasions), percentage of times all corners were visited, and percentage of items (out of 12) collected.

Corner	Times visited			
	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0.75$	$\alpha = 1$
(1,1)	2.2	2.8	2.7	2.9
(1,6)	2.1	2.6	2.7	3.2
(6,1)	2.0	2.7	2.6	3.0
(6,6)	1.7	2.6	2.9	3.0
All	8.0 %	10.7 %	10.9 %	12.1 %
Items coll'ed	62 %	36 %	29 %	0 %

Then, we observe the agent's behavior if we set $\alpha = 0.75$. In this case, the agent collects 3.5 items on average, and its corner-visiting behavior—as given in the second-from-right column of Table 1—is proportional to the value of α, as expected.

Finally, we ignore the collection of items by setting $\alpha = 1$. That is, we let

$$Q^*_{HPB}(a, B, I, h) \stackrel{def}{=} Satf_\beta(I, B) + \gamma \sum_{z \in Z} Pr(z \mid a, B) \max_{a' \in A} Q^*_{HPB}(a', B', I, h-1).$$

The right-most results column of Table 1 shows the average number of times each corner was visited when collecting items is not a preference. No items were collected.

These experiments highlight five important features of an HPB agent:
(1) While the agent is pursuing goals, it can concurrently perform rewarding actions not directly related to its goals. (2) Each of several goals can be pursued individually until satisfactorily achieved. (3) Goals must periodically be reachieved. (4) The trade-off between goals and preferences can be set effectively. (5) Goals and preferences can be satisfied even while dealing with stochastic actions and perceptions.

5 Towards Generalizing Goals

Considering exactly one preference, and pursuing exactly one goal at a time does not leave the agent designer with much flexibility. Moreover, are we justified in making such an absolute distinction between *preferences* and *goals*?

In an attempt to generalize the specification of goals and preferences, one might define $I \subseteq G$ to be the agent's current *set* of intentions. So here, there is not necessarily a single goal assigned the status of *intention*, but a set of goals are intentions; every goal (intention) in I is simultaneously pursued.

Instead of the agent having a particular preference, the design process could be made more flexible if the agent may be designed to exhibit preferential behavior—as loosely defined earlier via $Pref(\cdot)$—with respect to one or more goals.

And we let $Util = \langle \kappa, Satf \rangle$, where κ is a cost function in $\mathbb{R} \cap [0, 1]$ and $Satf$ is a *set* of satisfaction functions $\{Satf^g \mid g \in G\}$. κ has the same definition as a POMDP reward function, but models the agent's action costs and not what is typically thought of as rewards. Rewards are gained to the degree the agent's goals are satisfied: Every $Satf^g$ is a satisfaction function with domain in S and range in $\mathbb{R} \cap [0, 1]$, that is, $Satf^g(s) \in [0, 1]$. $Satf^g$ measures the degree to which g is satisfied.

Every goal $g_i \in G$ will be weighted by α_{g_i} according to the importance of g_i to the agent. Let $\{\alpha_{g_1}, \alpha_{g_2}, \ldots, \alpha_{g_n}\}$ be the weights of the goals in $G = \{g_1, g_2, \ldots, g_n\}$ such that α_{g_i} is the weight of g_i, $\alpha_{g_i} > 0$ for all i, and $\sum_{i=1}^{n} \alpha_{g_i} = 1$. Then the generalized action-value function can be defined as

$$Q^*_{HPB}(a, B, I, h) \stackrel{def}{=} i(I, 1)\alpha_{g_1} Satf^{g_1}_{\beta}(B) + \cdots + i(I, n)\alpha_{g_n} Satf^{g_n}_{\beta}(B) - \kappa_{\beta}(a, B) + \gamma \sum_{z \in Z} Pr(z \mid a, B) \max_{a' \in A} Q^*_{HPB}(a', B', I, h - 1),$$

$$Q^*_{HPB}(a, B, I, 1) \stackrel{def}{=} i(I, 1)\alpha_{g_1} Satf^{g_1}_{\beta}(B) + \cdots + i(I, 1)\alpha_{g_n} Satf^{g_n}_{\beta}(B) - \kappa_{\beta}(a, B),$$

where

- $Satf^g_{\beta}(\cdot)$ and $\kappa_{\beta}(\cdot)$ are the expected (w.r.t. a belief-state) values of $Satf^g(\cdot)$, respectively, $\kappa(\cdot)$,
- $i(I, j) = 1$ if $j \in I$, else $i(I, j) = 0$ if $j \notin I$,
- $B' = SE(a, z, B)$,
- γ is the normal POMDP discount factor and
- SE is the normal POMDP state estimation function.

Focus could now be defined as follows. If $g \notin I$ and for all $g' \in I$, $D(g) > D(g')$, then add g to I. And for every $g \in I$, if $Remove(g, I)$ returns 'yes', then remove g from I. It is the job of $Remove(g, I)$ to recognize when g seems impossible or too expensive to achieve, and thus needs to be removed from I.

Let $Satf\_levels(g)$ be the sequence of satisfaction levels of some goal $g \in I$ since g became active (i.e., was added to I). For every goal, its satisfaction levels are maintained if and only if the goal is currently an intention.

From preliminary simulations, it seems that the definition of *Focus*, just given, is inadequate for the proposed generalization. It does, however, provide a stepping-stone in the ongoing research.

6 Related Work and Conclusion

Our work focuses on providing high-level decision-making capabilities for robots and agents who live in dynamic stochastic environments, where multiple goals and goal types must be pursued. We introduced a hybrid POMDP-BDI agent architecture, which may display emergent behavior, driven by the intensities of their desires. In the past decade, several BDIAs have been augmented with capabilities to deal with uncertainty. The HPB architecture is novel in that, while the agent is pursuing goals, it can concurrently perform rewarding actions not directly related to its goals, and goals must periodically be re-achieved, depending on the goals' desire levels, which change over time and in proportion to how close the goals are to being satisfied.

The ideas presented in Sect. 5 and the associated preliminary simulations indicate that generalizing our agent architecture will be an interesting and challenging endeavour.

[18,19] have incorporated online plan generation into BDI systems, however the planners deal only with deterministic actions and observations.

[20] use POMDP theory to coordinate teams of agents. However, their framework is very different to our architecture. They use POMDP theory to determine good role assignments of team members, not for generating policies online.

[21] provide a rather sophisticated architecture for controlling the behavior of an emotional agent. Their agents reason with several classes of emotion and their agents are supposed to portray emotional behavior, not simply to solve problems, but to look believable to humans. Their architecture has a "continuous planner [...] that is capable of partial order planning and includes emotion-focused coping [...]" Their work has a different application to ours, however, we could take inspiration from them to improve the HPB architecture.

[22] take a different approach to use POMDPs to improve BDI agents. By leveraging the relationship between POMDP and BDI models, as discussed by [23], they devised an algorithm to extract BDI plans from optimal POMDP policies. The main difference to our work is that their policies are pre-generated and BDI-style rules are extracted for all contingencies. The advantage is that no (time-consuming) online plan/policy generation is necessary. The disadvantage of their approach is that all the BDI plans must be stores and every time the domain model changes, a new POMDP must be solved and the policy-to-BDI-plan algorithm must be run. It is not exactly clear from their paper [22] how or when intentions are chosen. Although it is interesting to know the relationship between POMDPs and BDI models [23,24], we did not use any of these insights in developing our architecture. However, the fact that the HPB architecture does integrate the two frameworks, is probably due to the existence of the relationship.

[25] also introduced a hybrid POMDP-BDI architecture, but without a notion of desire levels or satisfaction levels. Although their basic approaches to combine the POMDP and BDI frameworks is the same as ours, there are at least two major differences: Firstly, they define their architecture in terms of the GOLOG agent language [26]. Secondly, their approach uses a computationally intensive method for deciding whether to refocus; performing short policy look-aheads to

ascertain the most valuable goal to pursue.[2] Our approach seems much more efficient.

[27] incorporate probabilistic graphical models into the BDI framework for plan selection in stochastic environments. An agent maintains epistemic states (with random variables) to model the uncertainty about the stochastic environment, and corresponding belief sets of the epistemic state are defined. The possible states of the environment, according to sensory observations, and their relationships are modeled using probabilistic graphical models: The uncertainty propagation is carried out by Bayesian Networks, and belief sets derived from the epistemic states trigger the selection of relevant plans from a plan library. For cases when more than one plan is applicable due to uncertainty in an agent's beliefs, they propose a utility-driven approach for plan selection, where utilities of actions are modeled in influence diagrams. Our architecture is different in that it does not have a library of pre-supplied plans; in our architecture, policies (plans) are generated online.

None of the approaches mentioned maintain desire levels for selecting intentions. The benefit of maintaining desire levels is that intentions are not selected only according what they offer with respect to their *current* expected reward, but also according to when last they were achieved.

Although [20,27] call their approaches hybrid, our architecture can arguably more confidently be called hybrid because of its more intimate integration of POMDP and BDI concepts.

We could take some advice from [28]. They provide a systematic methodology to incorporate emotion into a decision-theoretic framework, and also provide "a principled, domain-independent methodology for generating heuristics in novel situations".

Policies returned by *Plan* as defined in this paper are optimal. A major benefit of a POMDP-based architecture is that the literature on POMDP planning optimization [6,29–35] (for instance) can be drawn upon to improve the speed with which policies can be generated.

Our architecture cannot yet control how often one goal is sought relative to other goals. It would be advantageous to be able to do this.

Evaluating the proposed architecture in richer domains would highlight problems in the architecture and indicate new directions for research and development in the area of hybrid POMDP-BDI architectures.

References

1. Bratman, M.: Intention, Plans, and Practical Reason. Harvard University Press, Massachusetts (1987)
2. Rao, A., Georgeff, M.: BDI agents: From theory to practice. In: Proceedings of the ICMAS 1995, pp. 312–319. AAAI Press (1995)

[2] Essentially, the goals in G are stacked in descending order of the value of $V^*_{HPB}(B, g, h^-)$, where $h^- < h$ and B is the current belief-state. The goal on top of the stack becomes the intention.

3. Monahan, G.: A survey of partially observable Markov decision processes: theory, models, and algorithms. Manage. Sci. **28**, 1–16 (1982)
4. Lovejoy, W.: A survey of algorithmic methods for partially observed Markov decision processes. Ann. Oper. Res. **28**, 47–66 (1991)
5. Koenig, S.: Agent-centered search. Artif. Intell. Mag. **22**, 109–131 (2001)
6. Ross, S., Pineau, J., Paquet, S., Chaib-draa, B.: Online planning algorithms for POMDPs. J. Artif. Intell. Res. (JAIR) **32**, 663–704 (2008)
7. Schut, M., Wooldridge, M., Parsons, S.: The theory and practice of intention reconsideration. Exp. Theor. Artif. Intell. **16**, 261–293 (2004)
8. Wooldridge, M.: Intelligent agents. In: Weiss, G. (ed.) Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Massachusetts (1999)
9. Wooldridge, M.: An Introduction to Multiagent Systems. Wiley, Chichester (2002)
10. Wooldridge, M.: Reasoning About Rational Agents. MIT Press, Massachusetts (2000)
11. Schut, M., Wooldridge, M.: Principles of intention reconsideration. In: Agents 2001: Proceedings of the 5th International Conference on Autonomous Agents, pp. 340–347. ACM Press, New York (2001)
12. Pollack, M., Ringuette, M.: Introducing the tileworld: experimentally evaluating agent architectures. In: Proceedings of the AAAI 1990, pp. 183–189. AAAI Press (1990)
13. Kinny, D., Georgeff, M.: Commitment and effectiveness of situated agents. In: Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91), pp. 82–88 (1991)
14. Kinny, D., Georgeff, M.: Experiments in optimal sensing for situated agents. In: Proceedings of the 2nd Pacific Rim Internatioanl Conference on Artificial Intelligence (PRICAI 1992) (1992)
15. Schut, M., Wooldridge, M.: Intention reconsideration in complex environments. In: Proceedings of the 4th International Conference on Autonomous Agents (AGENTS 2000). ACM, New York (2000)
16. Schut, M., Wooldridge, M.: The control of reasoning in resource-bounded agents. Knowl. Eng. Rev. **16**, 215–240 (2001)
17. Kaelbling, L., Littman, M., Cassandra, A.: Planning and acting in partially observable stochastic domains. Artif. Intell. **101**, 99–134 (1998)
18. Walczak, A., Braubach, L., Pokahr, A., Lamersdorf, W.: Augmenting BDI agents with deliberative planning techniques. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2006. LNCS (LNAI), vol. 4411, pp. 113–127. Springer, Heidelberg (2007)
19. Meneguzzi, F., Zorzo, A., Móra, M., Luck, M.: Incorporating planning into BDI systems. Scalable Comput. Pract. Experience **8**, 15–28 (2007)
20. Nair, R., Tambe, M.: Hybrid bdi-pomdp framework for multiagent teaming. J. Artif. Intell. Res. (JAIR) **23**, 367–420 (2005)
21. Lim, M.Y., Dias, J., Aylett, R.S., Paiva, A.C.R.: Improving adaptiveness in autonomous characters. In: Prendinger, H., Lester, J.C., Ishizuka, M. (eds.) IVA 2008. LNCS (LNAI), vol. 5208, pp. 348–355. Springer, Heidelberg (2008)
22. Pereira, D., Gonçalves, L., Dimuro, G., Costa, A.: Constructing bdi plans from optimal pomdp policies, with an application to agentspeak programming. In: Henning, G., Galli, M., Goneet, S. (eds.) XXXIV Conferência Latinoamericano de Informática, Santa Fe. Anales CLEI 2008, pp. 240–249 (2008)

23. Simari, G., Parsons, S.: On the relationship between mdps and the bdi architecture. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2006, pp. 1041–1048. ACM, New York (2006)
24. Simari, G., Parsons, S.: Markov Decision Processes and the Belief-Desire-Intention Model. Springer Briefs in Computer Science. Springer, Heidelberg (2011)
25. Rens, G., Ferrein, A., Van der Poel, E.: A BDI agent architecture for a POMDP planner. In: Lakemeyer, G., Morgenstern, L., Williams, M.A. (eds.) Proceedings of the 9th International Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense 2009), University of Technology, pp. 109–114. UTSe Press, Sydney (2009)
26. Boutilier, C., Reiter, R., Soutchanski, M., Thrun, S.: Decision-theoretic, high-level agent programming in the situation calculus. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000) and of the Twelfth Conference on Innovative Applications of Artificial Intelligence (IAAI 2000), pp. 355–362. AAAI Press, Menlo Park (2000)
27. Chen, Y., Hong, J., Liu, W., Godo, L., Sierra, C., Loughlin, M.: Incorporating PGMs into a BDI architecture. In: Boella, G., Elkind, E., Savarimuthu, B.T.R., Dignum, F., Purvis, M.K. (eds.) PRIMA 2013. LNCS, vol. 8291, pp. 54–69. Springer, Heidelberg (2013)
28. Antos, D., Pfeffer, A.: Using emotions to enhance decision-making. In: Walsh, T. (ed.) Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), pp. 24–30. AAAI Press, Menlo Park (2011)
29. Murphy, R.: Introduction to AI Robotics. MIT Press, Massachusetts (2000)
30. Roy, N., Gordon, G., Thrun, S.: Finding approximate POMDP solutions through belief compressions. J. Artif. Intell. Res. (JAIR) **23**, 1–40 (2005)
31. Paquet, S., Tobin, L., Chaib-draa, B.: Real-time decision making for large POMDPs. In: Kégl, B., Lee, H.-H. (eds.) Canadian AI 2005. LNCS (LNAI), vol. 3501, pp. 450–455. Springer, Heidelberg (2005)
32. Li, X., Cheung, W., Liu, J.: Towards solving large-scale POMDP problems via spatio-temporal belief state clustering. In: Proceedings of IJCAI-05 Workshop on Reasoning with Uncertainty in Robotics (RUR 2005) (2005)
33. Shani, G., Brafman, R., Shimony, S.: Forward search value iteration for POMDPs. In: de Mantaras, R.L. (ed.) Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 2619–2624. AAAI Press, Menlo Park (2007)
34. Cai, C., Liao, X., Carin, L.: Learning to explore and exploit in pomdps. In: NIPS, pp. 198–206 (2009)
35. Shani, G., Pineau, J., Kaplow, R.: A survey of point-based pomdp solvers. Auton. Agent. Multi-Agent Syst. **27**, 1–51 (2013)