Alternatives to Threshold-Based Desire Selection in Bayesian BDI Agents

Bernardo Luz[1], Felipe Meneguzzi[2], and Rosa Vicari[1]

[1] Informatics Institute, Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500 – Porto Alegre, Brazil
{bernardo.luz,rosa}@inf.ufrgs.br,
[2] School of Computer Science, Pontifical Catholic University of Rio Grande do Sul
Av. Ipiranga, 6681 – Porto Alegre, Brazil
felipe.meneguzzi@pucrs.br

Abstract. Bayesian BDI agents employ bayesian networks to represent uncertain knowledge within an agent's beliefs. Although such models allow a richer belief representation, current models of bayesian BDI agents employ a rather limited strategy for desire selection, namely one based on threshold values on belief probability. Consequently, such an approach precludes an agent from selecting desires conditioned on beliefs with probabilities below a certain threshold, even if those desires could be achieved if they had been selected. To address this limitation, we develop three alternative approaches to desire selection under uncertainty. We show how these approaches allow an agent to sometimes select desires whose belief conditions have very low probabilities and discuss experimental scenarios.

1 Introduction

Due to its computable representation of practical reasoning and its folk psychological abstraction to autonomous reasoning, the beliefs, desires and intentions (BDI) model has been extensively studied within the autonomous agents community. Most traditional implementations of BDI agents include a logic-based belief base representing the knowledge an agent has about the world, and plan library that can be selected by an agent when it adopts certain desires. Once a course of action is selected by an agent for execution, it becomes part of an agent's intention to which an agent commits to execute.

Beliefs are traditionally represented by a closed set of ground atomic literals, each of which is associated with a truth value, and consequently does not normally represent uncertainty. Nevertheless, there exist logical formalisms to represent uncertainty regarding an agent's beliefs. Bayesian networks [9] are a popular way of representing uncertain information probabilistically, where parts of it are conditioned on others (e.g., cause and consequence relationships, diseases and symptoms). They are directed acyclic graphs (DAGs), whose nodes represent event variables associated with two or more possible states, and each state has an explicit occurrence probability.

Given the lack of support for uncertainty in the BDI agent model and the representational power of bayesian networks, there has been work focused on extending the BDI agent model to reason with uncertainty using bayesian networks [4]. This type of agent model no longer relies solely on ground literals to represent an agent's belief base, but rather on a bayesian network.

Traditional BDI agents select desires (or plans with implicit desires) based on a binary condition on the literals of the belief base, under the assumption that this condition is minimal for the desire's viability. The underlying idea is that, if the context condition is not true, then a desire and its associated plans have no chance of being successful. However, even if the context condition is true, a desire might be impossible and an intention associated with it might fail. Similarly, bayesian BDI agents are susceptible to selecting desires that cannot be satisfied in the current world state. Previous approaches to bayesian BDI reasoning [4] have relied on performing desire selection validation by applying a threshold on the probability being evaluted (e.g., that of the desire itself), so that if a certain logical query is less probable than the threshold, then the desire does not meet the minimal requirement to being successful. However, in a probabilistic world, context conditions are less crisply defined. In response, we have developed three alternative desire selection strategies that relax the requirement on the probability threshold for the context condition, and analyze situations where these strategies might be advantageous.

This paper is organized as follows: Section 2.1 presents BDI agents, Section 2.2 presents bayesian networks, Section 2.3 presents bayesian BDI agents, Section 3 presents bayesian BDI reasoning, Section 3.1 presents a threshold-based desire selection process, Section 4 presents alternative approaches to bayesian BDI desire selection, Section 4.1 presents Probability Ranking desire selection, Section 4.2 presents Biased Lottery desire selection, Section 4.3 presents Multi-Desire Biased Random Selection, Section 5 presents an example, and Section 6 presents our final considerations.

2 Background

In this section, we review previous efforts upon which our work is based. We start by briefly explaining the BDI model, then proceed to introducing the basics of bayesian networks and finally we enumerate existing work on bayesian BDI agents.

2.1 BDI Agents

Autonomous agents are often defined as encapsulated computer systems *situated* in an environment and capable of *flexible* autonomous action in this environment in order to achieve certain goals [5]. The agent must adapt itself to a dynamic environment, while seeking to fulfill its goals. In order to provide a stronger computational grounding for this notion of agent, many architectures have been proposed, among which, one of the most widely studied is the one centered around the mental attitudes of beliefs, desires and intentions (or BDI) [2]. This architecture was originally proposed as a philosophical model of human *practical reasoning*, that is, reasoning aimed at deciding how to act in the world towards achieving one's goals.

Beliefs contain a representation, internal to the agent, of environment elements considered relevant for the agent's reasoning. The state of an agent's beliefs may contain either less information than the current state of the environment (e.g., because of limited sensing ability), or more (e.g., if the agent does additional information processing on its sensing). Desires represent objectives that the agent would like to achieve (i.e., they can be considered an agent's *motivation* [10]). Intentions are those desires that the agent has committed itself to achieving, as well as the steps towards achieving these desires. Agents resist abandoning their intentions, and, should a plan fail, it is often the case that they choose to re-plan.

A BDI agent selects desires through a process that considers the current viability and the absence of conflict with existing intentions. Desires often have preconditioning beliefs that indicate whether or not they should be selected by the agent, as a matter of logical evaluation [8].

2.2 Bayesian Networks

Traditional first-order logic approaches to knowledge representation are insufficient to represent certain domains where there is uncertainty in the validity of statements over time [6, 11]. Examples of reasons for this limitation are the high cost of exhaustively representing all possible combinations of truth values using logic rules (laziness), the lack of a complete theory of the domain in question (theoretical ignorance), and the potential impossibility or inviability of performing all necessary tests to ascertain complete truth for certain statements (practical ignorance).

The fact remains that people commonly reason with incomplete knowledge and make decisions based on assumptions over unknown facts. This knowledge comprises what is *known* to be true, what is *not known* and *estimates* based on relationships between elements of the world. Pearl [9] devised a formalism to represent partial knowledge based on the causal relationships between elements in the world, using probability theory to represent how knowledge about one element in the world influences the certainty about others related to it. Here, relationships between elements are represented in a network, and probabilities between related elements are calculated using *Bayes' Rule*, with the resulting formalism being called a *Bayesian Network.* A bayesian network is a type of *causal network* that allows the specification of knowledge where parts of it are conditioned on others, supporting the update of probabilities when new information (i.e., *evidence*) is obtained.

Given two events A and B, if we know the probability of A given B and the probability of B, we can calculate the probability of seeing both A and B, as shown in Equation 1, which represents the *fundamental rule* for probability calculus. It can also be conditioned on another event C, as shown in Equation 2.[3]

$$P(A|B)P(B) = P(A \cap B). \tag{1}$$

$$P(A|B \cap C)P(B|C) = P(A \cap B|C). \tag{2}$$

Equation 3 is the key equation behind bayesian networks: *Bayes' Rule*. Bayes' Rule makes it possible to update beliefs about an event A, provided that we get information about another event B. Thus, $P(A)$ is usually called the *prior probability* of A, whereas $P(A|B)$ is called the *posterior probability* of A given B. There is also a *general version* of Bayes' Rule, in a context C – exhibited as Equation 4.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{3}$$

$$P(A|B,C) = \frac{P(B|A,C)P(A|C)}{P(B|C)}. \tag{4}$$

There may be evidence that a given variable is in a certain state. When this happens, it is said that such a variable is *instantiated*. This kind of evidence is called *hard evidence*. Conversely, if a statement about a variable state is made based on dependencies rather than explicit knowledge, it is said that there is *soft evidence* about that variable.

The *d-separation* property tells us if two variables are independent of each other in the current state of the bayesian network. There are three types of connection in the topology of a bayesian network: serial, diverging and converging. Each connection type accounts for a specific reasoning as to whether variables are *d-separated* or *d-connected* (what we call variables that are not d-separated)[4]. In a serial connection, if we have no hard evidence about a variable, evidence about its parent/child passes through it, affecting our beliefs about it and about its uninstantiated child/parent. In a diverging connection, if we have no hard evidence concerning the parent, evidence about one of its children affects our beliefs about the other – uninstantiated – children. In a converging connection, if we have no hard evidence about the child or one of its descendants, evidence about a parent does not influence our beliefs about the other(s).

2.3 Bayesian BDI Agents

Although traditional implementations of BDI agents use a logic-based approach to model the world, these approaches fail to account for the uncertainty inherently associated with the real world. In order to address this shortcoming, work has been carried out to switch from a purely logical view of the agent's beliefs

[3] The equations in Section 2.2 have been extracted from [6].
[4] "d" is for "directed graph".

based on traditional logic to a bayesian network, integrating it into the reasoning process of BDI agents. Fagundes et al. [4] have created an ontology-based BDI agent in which the belief base is replaced by a bayesian network and the desire selection process relies on probability thresholds to adopt new desires. Kieling and Vicari [7] integrate a bayesian network into an implementation of the Jason [1] AgentSpeak(L) [10] interpreter. Finally, Carrera and Iglesias [3] focus on the process of updating beliefs within a bayesian BDI agent.

3 Bayesian BDI reasoning

In this section, we develop a reasoning cycle that should be general enough that it could be used to describe the reasoning performed by previous work on bayesian BDI agents [3, 4, 7]. Later, we describe a desire selection process that is built around a threshold evaluation, as in [4], within such a reasoning cycle.

In this paper, we consider the belief base to correspond to an entire bayesian network whereby the causal relations between beliefs are explicitly represented. Moreover, given current evidence, we also explicitly represent the probability that a certain variable is in a particular state. Each event variable has n possible states, each with an associated probability, that either is readily available from a *conditional probability table* if the state of all the parent variables is known (i.e., there is hard evidence on each of them) or has to be calculated.

Desires in the bayesian BDI agent model refer to specific event variable states in the bayesian network, and each desire has a preconditioning belief, indicating when that desire can be adopted by the agent. Our choice of belief-preconditioned desire representation follows the tradition of many implemented BDI systems (e.g., [8, 10]). We present two types of desire for bayesian BDI agents: *strong* and *weak*. Strong desires are desires on which there must be *hard evidence* so that they can be considered fulfilled. There may not be any doubt, however small, on whether or not a strong desire has been satisfied. Weak desires are those that are not necessarily expected to be confirmed via hard evidence, but are expected to be believed to be sufficiently likely to be true, i.e., to reach a certain minimum probability value. These may be viewed as desires that accept *soft evidence* as sufficient in order to be considered satisfied. Strong desires may be viewed as a special case of what would otherwise be weak desires, where, given each desire d, $P(d) = 1$.

Similarly to traditional BDI agents, intentions are desires to the fulfillment of which the agent has committed itself. The agent will seek a plan – a sequence of actions – that is applicable to the current situation: any plan that is aimed at satisfying at least one of the intentions and whose preconditions are not conclusively denied (i.e., preconditions that are not contradictory to hard evidence) is valid.

Algorithm 1 outlines a generic reasoning cycle for a bayesian BDI agent reasoning within an uncertain environment. First, the agent updates its belief base (i.e., the probabilities in the bayesian network), according to the latest perceptions from the environment (Line 2). The agent then proceeds to evaluate

Algorithm 1 Reasoning Cycle for Bayesian BDI Agents

```
1: procedure Reasoning Cycle for Bayesian BDI Agents
2:     update beliefs based on percept
3:     evaluate and possibly choose desires
4:     seek plans that might satisfy the chosen desires
5:     for each chosen desire, if an applicable plan has been found, create an intention
       and associate it with the desire and the plan, which is therefore adopted
6:     if a plan was not found, mark the desire as unsatisfiable at this time
7:     if adopted plan failed, either seek another plan or remove the intention (subject
       to commitment policy)
8:     if adopted plan succeeded, desire is considered fulfilled and this is reflected on
       beliefs
9: end procedure
```

its possible desires (Line 3), if a desire was selected, the agent must commit to satisfying it by adopting an intention. Once it has committed it seeks plans capable of satisfying it (Line 4) and then executes the plans in an attempt to achieve the goal (Line 5). For each chosen desire for which no way of attempting to fulfill it has been found, mark it as "unsatisfiable at this time" and refrain from creating an intention for it in the current cycle (Line 6). If there is an adopted plan and it fails, then the agent may either seek an alternate plan or give up on the corresponding intention altogether (Line 7). This is subject to a *commitment policy* that may take into account whether this happened before to the desire associated with this intention, to intentions in general (there could conceivably be some kind of overall environment issue behind the failures), the rate at which alternate plans have proven effective, the computational cost for obtaining such plans – perhaps compared to the cost of desire selection, etc.. Successful plan executions cause their associated intentions and, in turn, desires to be marked as satisfied (Line 8). This fact will be implicitly reflected in the agent's beliefs in the next perception cycle, unless there is a contradicting change before then, e.g., by sensing an environment change caused by another agent.

Since Bayesian BDI agents' beliefs are extended with probabilistic data, it is no longer sufficient to perform the logical evaluation for each desire's preconditions to determine those that are eligible for intention creation. Just as there are degrees of probability in the beliefs, selecting a desire is now a decision made with varying degrees of confidence, which implies that preconditions are no longer strictly about *validity* of selection, but also about *confidence* in a selection that is made under uncertainty. The only case where it is a matter of validity is when there is hard evidence *against* the desire's precondition (i.e., evidence of a different state of the event variable).

Moser et al. [4] performs reasoning using a threshold-based evaluation: if the probability being evaluated is equal to or greater than the threshold value, the associated event variable state is considered valid; in that work, the existence of a belief is dependent on this validation. Such reasoning involves checking if the probability associated with the applicability of the desire satisfies the threshold;

if so, the desire may be selected; otherwise, the agent simulates hard evidence on all combinations of the preconditioning event variable states – one state per variable – to determine which such state combinations would, if supported by hard evidence, allow for a threshold-satisfying probability of the event variable state corresponding to the desire itself, if any. New desires are then created from these states, connected to the original desire through causality.

Desires preconditioned on beliefs holding a probability that is exactly equal to zero, i.e., as a result of hard evidence on a state other than the one referred to by the belief in question, but associated with the same event variable, must not be selected. It is important to point out that once an intention has been dropped (i.e., not fulfilled), the desire is added back to the list of desires; without this, failed desires would be lost. Although this is not shown in any of the selection algorithms in this paper, as it is not a part of desire selection itself, it is an underlying assumption of the reasoning cycle.

3.1 Threshold-Based Desire Selection

In this section we describe a desire selection process that is threshold-based, which is a key characteristic in previous work [4], in terms of our reasoning cycle. This process is summarized in the pseudocode of Algorithm 2, representing a threshold-based desire selection algorithm in the context of a Bayesian BDI agent. It takes as parameters the numeric threshold value and the list of available *desires* (Line 1). The algorithm traverses the list of desires (Line 2) and, for each one, evaluates whether the probability of its associated precondition is greater than, or equal to the threshold (Line 3). If so, the desire in question is removed from the list of desires (Line 4) and returned (Line 5), thereby refraining from continuing to traverse the list, the implication being that this algorithm only selects one desire. If the entire list of desires is traversed and no desire has been selected (Line 7), the algorithm returns *null* (Line 8), denoting that no new desire is pursued by the agent.

Algorithm 2 threshold-based selection

```
1: function ThresholdBasedSelection(threshold, desires)
2:     for each desire such that $desire \in desires$ do
3:         if $desire.preCondition.probability \geq threshold$ then
4:             $desires.remove(desire)$
5:             return desire
6:         end if
7:     end for
8:     return null
9: end function
```

4 Alternatives for Bayesian BDI Desire Selection

The threshold-based desire selection algorithm shown in Section 3 avoids selecting desires whose belief preconditions do not meet a minimal degree of probabilistic support. As such, it constitutes a relatively simplistic mechanism for desire selection in a probabilistic section, and suffers from two key limitations. On the one hand, as the selection threshold approaches one, the agent becomes extremely conservative, and may not select any desire and remain idle for long periods of time.

On the other hand, if the threshold approaches zero, the agent becomes less strict in ensuring the viability of the desires it chooses to pursue. Importantly, depending on the order in which the desires are checked, the agent might select desires that are less likely than others.

Moser et al. [4] work with the notion of incompatible desires in Bayesian BDI agents, which is beyond the scope of this work. These incompatible desires are sorted by probability and the one with the highest probability is selected. The selection process for multiple desires that are not considered incompatible is not a concern in their work; there, competition is not assumed to be a part of the desire selection process. In this paper, we assume that desires do not conflict with each other, or that there is a process that filters conflicting desires. Moreover, we assume competition among desires during selection, unless otherwise specified.

In order to address the limitations of threshold-based selection, we propose a number of alternative desire selection mechanisms that ensure a finer control over an agent's choice of desires while taking into consideration the probabilistic nature of an environment. These approaches eliminate idleness and ensure that more likely desires are selected more often. In the algorithms developed in this section, similarly to the desire selection algorithm shown in Section 3, we assume that once an intention is dropped the desire is added back to the list of desires.

4.1 Probability Ranking

This approach involves sorting the desire list in decreasing order of precondition probability, resulting in a ranking from highest to lowest probability precondition, and picking up the desire backed by the belief most likely to be true. The pseudocode in Algorithm 3 illustrates the Probability Ranking desire selection algorithm. Its only parameter is a list of *desires* (Line 1). If there are any desires (Line 2) the algorithm sorts them by precondition probability ($rankedDesires$, Line 3), selects the first desire (Line 4), removes it from the list (Line 5) and, if the probability of that desire's precondition is greater than 0 (Line 6) – to prevent a desire associated with a contradicted precondition from being selected – returns that desire (Line 7). Otherwise, the algorithm returns *null* (Line 10).

4.2 Biased Lottery

Selecting desires by ranking them over their precondition probability as we show in Section 4.1 helps ensure that an agent is never idle. However, it is still possible that certain desires will never be selected, even if they were possible but

Algorithm 3 Probability Ranking Selection

```
 1: function ProbabilityRankingSelection(desires)
 2:     if $desires.length > 0$ then
 3:         $rankedDesires := desires$ ordered by precondition probability
 4:         $desire := rankedDesires.first()$
 5:         $desires.remove(desire)$
 6:         if $desire.preCondition.probability > 0$ then
 7:             return desire
 8:         end if
 9:     end if
10:     return null
11: end function
```

were weakly supported by the agent's beliefs. Situations where this is detrimental to the agent occur when the agent has not obtained enough evidence about the environment, or has obtained the wrong evidence. In order to address that limitation, we now develop a technique that randomly picks desires using their precondition probability to weight this selection. The idea is to randomly generate a number and use it to determine which desire to choose, according to a probability distribution reflecting the probabilities of the desires' preconditions.

In order to generate this probability distribution over the desires, we generate a series of numeric intervals within the $[0, 1]$ range assigning, for each belief, an interval proportional to the probability of their belief precondition. The probabilities, thus, serve as weights that create bias in what would otherwise constitute a purely random selection; it is a nondeterministic desire selection that is subject to bias from the precondition probability. This desire selection method neither disregards desires backed by beliefs holding very low probabilities, nor is designed to embrace them more often than common sense would permit – than such probabilities would suggest. We formalize this selection mechanism in the pseudocode of Algorithm 4, which uses the function described in Algorithm 5 to generate the selection probability intervals. Algorithm 4 takes as input the list of desires (Line 1) and generates a random numeric value (Line 2) and a list of numeric values (Line 3) that correspond to the upper limits (boundaries) for the numeric intervals used in desire selection; Function *GenerateIntervals* (Line 3) is detailed in Algorithm 5. The algorithm proceeds to traverse the list of upper interval limits (Lines 4–10); it uses the randomly generated number to select a desire (Lines 5 and 6), which is then removed from the list of desires and returned (Lines 7 and 8). If the entire list of upper interval limits is traversed and the random value has not been found to belong to any of the intervals (Line 10), the algorithm returns *null* (Line 11).

Algorithm 5 takes as input a list of *desires* (Line 1) and starts by creating a list to store the upper numeric interval limits that will be calculated (*intervals*, Line 2). Provided that there are elements in the input list, the algorithm proceeds to create a list that will contain the *probabilities* of the desires' preconditions (Lines 3 and 4). It also defines a variable *sum* that will be used to store the

Algorithm 4 Biased Lottery

```
1: function BiasedLottery($desires$)
2:     $randomValue := random\ number \in [0, 1]$
3:     $intervals :=$ GenerateIntervals($desires$)
4:     for $i = 0$ to $intervals.length$ do
5:         if $randomValue < intervals[i]$ then
6:             $desire := desires[i]$
7:             $desires.remove(desire)$
8:             return $desire$
9:         end if
10:    end for
11:    return $null$
12: end function
```

sum of all such probabilities, initializing it to 0 (Line 5). It then traverses the desire list (Lines 6–9) storing the probabilities of the desires' preconditions in the corresponding positions of *probabilities* and accumulating the probability of all desire preconditions in the *sum* variable (Lines 7 and 8). If the sum is greater than 1 (Line 10), it normalizes the *probabilities* and uses these values as interval sizes while generating numeric intervals (Lines 12–14). If not (Line 15), it generates numeric intervals using the *probabilities* as interval sizes (Lines 17–19). Lines 11 and 13 are the normalized equivalents of Lines 16 and 18, calculating and ultimately assigning upper interval limits to the positions in *intervals*.

We do not perform normalization when the sum of the precondition probabilities is less than 1.0, as this would inflate selection probabilities for desires preconditioned on insignificant events. For example, a single desire preconditioned on a belief with 0.0001 probability would be treated as though its probability were 1.0. Note that the numeric intervals for the desires are forced not to intersect with one another, since the one randomly generated number (per selection cycle) is expected to select, at most, one desire-associated numeric interval. Although this algorithm now allows an agent to sometimes pick desires that would not normally be selected, it is still limited to the choice of a single desire.

4.3 Multi-Desire Biased Random Selection

This approach to desire selection removes inter-desire competition, by considering desires independently of each other (e.g., full parallelism is possible), allowing multiple desires to be selected simultaneously. Given a desire D_i preconditioned on a belief holding a probability P_i, we say that D_i is assigned a numeric interval $I_i = [0, P_i]$. This is done for every pending (i.e., unfulfilled) desire. For every such desire D_i, if a randomly generated numeric value N_i in interval $[0, 1]$ belongs to interval I_i, the desire is added to the set of desires to be selected at the end of this selection cycle.

The pseudocode of Algorithm 6 formalizes our proposed approach for Multi-Desire Biased Random Selection. It takes as input the list of *desires* (Line 1), and

Algorithm 5 Biased Lottery – Desire Intervals

```
function BiasedLottery:GenerateIntervals($desires$)
    $intervals[desires.length]$
    if $desires.length > 0$ then
        $probabilities[desires.length]$
        $sum := 0$
        for $i := 0$ to $desires.length$ do
            $probabilities[i] := desires[i].preCondition.probability$
            $sum := sum + probabilities[i]$
        end for
        if $sum > 1$ then
            $intervals[0] := \frac{probabilities[0]}{sum}$
            for $i := 1$ to $intervals.length$ do
                $intervals[i] := intervals[i-1] + \frac{probabilities[i]}{sum}$
            end for
        else
            $intervals[0] := probabilities[0]$
            for $i := 1$ to $intervals.length$ do
                $intervals[i] := intervals[i-1] + probabilities[i]$
            end for
        end if
    end if
    return $intervals$
end function
```

creates a list that will be used to store any number of desires that may be selected (*selectedDesires*, Line 2). This selection takes place by traversing the list of desires (Lines 3–8), and randomly picking desires based on their precondition probability (Lines 4–6).

5 Example

In order to illustrate the effects of each desire selection strategy described in Section 4, we now introduce a working example to show how an agent would react to situations using our proposed algorithms. Our example scenario consists of a watchman agent that is tasked with guarding an installation. The presence of suspicious people nearby increases its estimate of a security breach. There is an alarm in the installation, that is effective under normal circumstances. However, there are reports of occasional electrical malfunctions in the installation, which may cause the alarm to ring for no reason or not to ring when it is expected to. Moreover, the watchman becomes interested in seeking evidence that there is not an electrical malfunction if it knows that there are suspicious people nearby. The surrounding area is known for intense traffic, and accidents are more common than in most other areas, resulting in noise that is almost always perceived by the agent. However, noise might be caused by trespassers, though that is not very likely. In order to patrol the installation, the watchman periodically chooses

Algorithm 6 Multi-Desire Biased Random Selection

```
 1: procedure MultiDesireBiasedRandomSelection($desires$)
 2:     $selectedDesires := \{\}$
 3:     for each $desire \in desires$ do
 4:         $randomValue := random\ number \in [0, 1]$
 5:         if $randomValue \leq desire.preCondition.probability$ then
 6:             $selectedDesires.add(desire)$
 7:         end if
 8:     end for
 9:     return $selectedDesires$
10: end procedure
```

between the default and an alternate route, and it becomes more inclined to patrol the alternate route as its belief that a security breach is either imminent or already taking place increases, and conversely, the watchman is more inclined to patrol the default route when everything looks calm.

Regarding the relationships among the event variables in the network, we note that: *i)* Evidence of the presence of suspicious people nearby increases the probability of a security breach; *ii)* Evidence of the alarm activating increases the probability of a security breach occurring, as well as the probability of there being suspicious people nearby. This is still true if there is also evidence of an electrical malfunction, but the probability increase for both event variable states is smaller. If there is evidence that there is no electrical malfunction (e.g., a notification about maintenance very recently performed), the probability increase is the greatest of the three cases; *iii)* evidence of noise increases the probability of a security breach. However, this increase is almost nullified upon evidence of an accident, as this network tells us that an accident is a much more probable cause of noise than a security breach, and the impact of a security breach on the probability of noise if we already know of an accident is small; and *iv)* An increase on the probability of a security breach (e.g., through evidence of suspicious people and noise) increases the probability of the alarm activating, even if there is an electrical malfunction, though then the probability increase is smaller.

The bayesian belief base of the watchman encoding the domain knowledge described in the scenario is represented in Figure 1. Do note that we do not associate the *Route* variable with a belief about the environment state, but rather we associate it with an internal belief associated with the agent's currently chosen route. It is not a part of the reasoning surrounding the probability of a security breach or any of the other event variables, and this is the reason we left it disconnected from all the other nodes in the network.

This watchman agent has two mutually exclusive strong desires that are periodically renewed:[5] $Route.default(SecurityBreach.false)$ and $Route.alternate($

[5] We denote desires in the form `<desire>(<preconditioning belief>)`, where both elements are described as `<event variable>.<state>`.

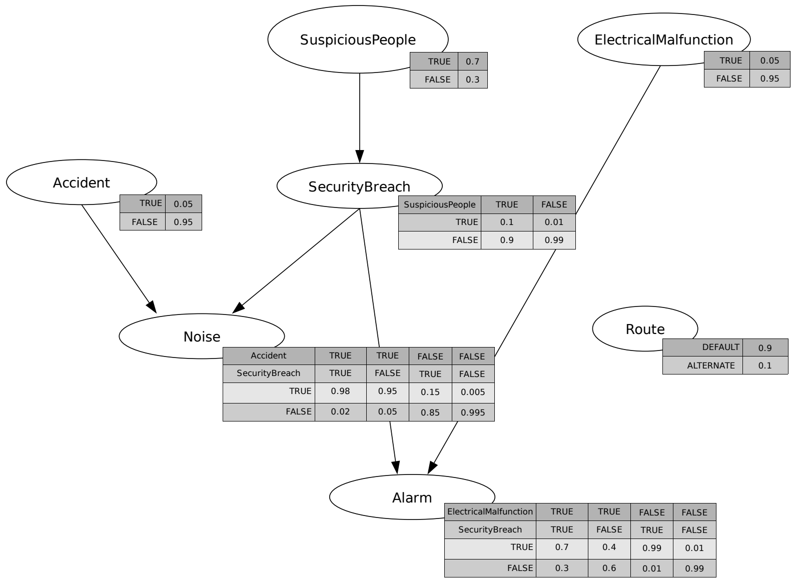

Fig. 1. Bayesian network corresponding to the *Watchman* agent's beliefs

$SecurityBreach.true$). That is, the agent desires to patrol the default route if there has not been a security breach and the alternate one otherwise. It also has the weak desire $ElectricalMalfunction.false(SuspiciousPeople.true)$, i.e., the desire to believe that there is no electrical malfunction at the moment, conditioned on the presence of suspicious people nearby, as well as the strong desire $Accident.true(Noise.true)$, i.e., the desire to discover that there has been an accident, if noise has been heard.

We now briefly present the result of using each of the four algorithms while working with an initial scenario – where what happens during the execution of each algorithm is not carried over to the next – where there is no hard evidence of any event. Since, there is no hard evidence yet, $P(SuspiciousPeople) = (0.7, 0.3)$ (i.e., $P(SuspiciousPeople = true) = 0.7$ and $P(SuspiciousPeople = false) = 0.3$), and consequently $P(SecurityBreach) = (0.073, 0.927)$; also, $P(Noise) = (0.0624, 0.9376)$. Desire $Route.default$ is preconditioned on a belief that $SecurityBreach$ is false, which has a 0.927 probability; desire *Route.alternate* is preconditioned on a belief that $SecurityBreach$ is true, which holds a 0.073 probability; desire $ElectricalMalfunction.false$ is preconditioned on a belief that $SuspiciousPeople$ is true, which holds a 0.7 probability; and desire *Accident.true* is preconditioned on a belief that $Noise$ is true, which holds a 0.0624 probability.

First, let us consider threshold-based desire selection, with a threshold of 0.75. This means that only $Route.default(SecurityBreach.false)$ is an eligible desire for selection, since $P(SecurityBreach = false) = 0.927$. In a scenario where there is hard evidence of noise (i.e., $P(Noise = true) = 1$), the probability of suspicious people nearby is increased: $P(SuspiciousPeople = true|Noise = true) = 0.7422$. However, desire $ElectricalMalfunction.false(SuspiciousPeople.true)$ still fails to satisfy our threshold even so $(0.7422 < 0.75)$. A lower threshold would work in this case, but a precondition's probability might always be smaller than the threshold, if evidence able to increase its probability enough is never obtained – this exemplifies how there may be desires that *are never* selected using this criterion. One might suggest simply lowering the threshold to an extremely low value, but without other criteria we would then simply have a desire selection process that is indifferent to the various probabilities presented.

If we use Probability Ranking desire selection, we get the following ranking:

1. $Route.default(SecurityBreach.false)$: 0.927
2. $ElectricalMalfunction.false(SuspiciousPeople.true)$: 0.7
3. $Route.alternate(SecurityBreach.true)$: 0.073
4. $Accident.true(Noise.true)$: 0.0624

The agent will desire to patrol the default route, then to establish that there is no electrical malfunction, then to patrol the alternate route, and finally to verify if there has been an accident, in this order, unless a belief update (e.g., evidence that $SecurityBreach = true$) causes the ranking to be modified. Note that although the preconditioning probabilities serve as a criterion for sorting the desires, the probability values by themselves have no impact on how often the desires may be selected, so $Accident.true(Noise.true)$ will be promptly selected in the absence of higher-ranked desires regardless of the fact that its precondition holds a low probability.

If we use Biased Lottery, we get the following list of numeric intervals for the desires (the order is irrelevant):

– $Route.default(SecurityBreach.false)$: $[0.0, 0.526)$
– $Route.alternate(SecurityBreach.true)$: $[0.526, 0.5674)$
– $ElectricalMalfunction.false(SuspiciousPeople.true)$: $[0.5674, 0.9646)$
– $Accident.true(Noise.true)$: $[0.9646, 1.0]$

The sum of the desires' precondition probabilities is greater than 1, so these values are normalized in the $[0, 1]$ interval and used to generate the intervals. Following the algorithm, a numeric value in the $[0, 1]$ interval is generated, and whichever interval it belongs to determines which desire is selected – if there were not a normalization, it could also tell us that no desire should be selected, by not belonging to any of the intervals. In this example, desire $Route.default(SecurityBreach.false)$ has a 0.526 probability of being selected, desire $Route.alternate(SecurityBreach.true)$ has a 0.0414 probability of being selected, desire $ElectricalMalfunction.false(SuspiciousPeople.true)$ has a 0.3972 probability of being selected, and desire $Accident.true(Noise.true)$ has a 0.0354 probability of being selected, each one competing with the others. So, if the randomly generated number is 0.3 (and thus within the first interval), the agent performs a patrol through the default route, or if the random number is 0.55, the patrol is through the alternate route.

If we use Multi-Desire Biased Random Selection, we get the following list of numeric intervals for the desires (the order is irrelevant):

– $Route.default(SecurityBreach.false)$: $[0.0, 0.927]$
– $Route.alternate(SecurityBreach.true)$: $[0.0, 0.073]$
– $ElectricalMalfunction.false(SuspiciousPeople.true)$: $[0.0, 0.7]$
– $Accident.true(Noise.true)$: $[0.0, 0.0624]$

For each of the four desires a numeric value in the $[0, 1]$ interval is generated, and if the numeric value belongs to the corresponding desire's numeric interval, the desire is selected. In this example, desires $Route.default(SecurityBreach.false)$, $Route.alternate(SecurityBreach.true)$, $ElectricalMalfunction.false(SuspiciousPeople.true)$ and $Accident.true(Noise.true)$ have, respectively, 0.927, 0.073, 0.7 and 0.0624 probabilities of being selected, whereas each possible selection is fully independent of the others, thus rendering the selection process passive to yield multiple selected desires.

6 Conclusions

From a conservative standpoint, one may argue that threshold-based selection is sensible as it is, as resources will not be used *without justification*. However, we believe that ignoring desires that are *probabilistically irrelevant* in desire selection is not necessarily a rational choice, since it precludes an agent from exploring an environment. In response, we have developed three desire selection strategies that try to overcome this limitation.

In Probability Ranking selection, desires that would be ignored by threshold-based selection do get a chance, though only after the ones that would be accepted by it. However, it might be undesirable to select a desire preconditioned on a belief holding a very low probability just because there is no better alternative.

In Biased Lottery, we rely on nondeterminism to consider all desires while ensuring that desires backed by beliefs holding high probabilities should be selected more often than those backed by beliefs holding low probabilities, in proportion to their probabilities. Ideally, the probability of selecting each desire would be the same as the one associated with its precondition. However, in the cases where the total sum of desire probabilities is greater than 1, the competition between the desires in question proportionally reduces the individual probabilities of selection for each desire.

In Multi-Desire Biased Random Selection, we also rely on nondeterminism for the same reason. A key difference is that the desires are considered independently of one another, so that there is no competition among the desires, thus the number of desires possibly selected is not limited to one. One limitation of our work is that we do not deal with the issue of desire incompatibility, as this would pose significant problems in a probabilistic setting.

The nondeterministic nature of Biased Lottery and Multi-Desire Biased Random Selection makes it so that the watchman agent's behavior may not be anticipated by a third party (e.g., another agent) intent on exploiting it. Such an exploitation could involve forging an accident to drive away suspicion arising from a noise heard by the watchman, for instance. Hiring an employee to plant false evidence of an electrical malfunction would also impact the watchman's beliefs, as a second, albeit more roundabout method of attempting to manipulate the watchman. This is an agent trait that we now describe as *unpredictable proactiveness*: agent behavior at a specific point in time cannot be *completely* determined by analyzing its beliefs, and is thus resistant to exploitation by a third party. Finally, our future work aims to evaluate the algorithms developed in different scenarios, and to investigate joint uses of Biased Lottery and Multi-Desire Biased Random Selection while considering desire incompatibilities.

References

[1] Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak Using Jason. John Wiley & Sons, Ltd (2007)
[2] Bratman, M., Israel, D.J., Pollack, M.E.: Plans and resource-bounded practical reasoning. Computational Intelligence 4, 349–355 (1988)
[3] Carrera, Á., Iglesias, C.A.: B2DI A Bayesian BDI Agent Model with Causal Belief Updating based on MSBN. In: Proceedings of the 4th International Conference on Agents and Artificial Intelligence (ICAART2012). pp. 343–346. SciTePress (2012)
[4] Fagundes, M.S., Vicari, R.M., Coelho, H.: Deliberation process in a BDI model with bayesian networks. In: Ghose, A.K., Governatori, G., Sadananda, R. (eds.) PRIMA. Lecture Notes in Computer Science, vol. 5044, pp. 207–218. Springer (2007)
[5] Jennings, N.R.: On agent-based software engineering. Artificial Intelligence 117, 277–296 (2000)
[6] Jensen, F.V., Nielsen, T.D.: Bayesian Networks and Decision Graphs. Springer, 2nd edn. (2007)
[7] Kieling, G., Vicari, R.M.: Insertion of probabilistic knowledge into BDI agents construction modeled in Bayesian Networks. In: International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS 2011). vol. 1, pp. 115–122. California: Conference Publishing Services (CPS), Seoul (2011)
[8] Móra, M.C., Lopes, J.G., Vicari, R.M., Coelho, H.: BDI models and systems: Reducing the gap. In: Intelligent Agents V: Agents Theories, Architectures, and Languages, 5th International Workshop, ATAL '98. pp. 11–27. Springer Berlin Heidelberg (1999)
[9] Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan-Kaufmann, San Mateo, CA (1988)
[10] Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: de Velde, W.V., Perram, J.W. (eds.) Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, LNCS, vol. 1038, pp. 42–55. Springer-Verlag (1996)
[11] Russel, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, New Jersey (1994)