Risk-aware Planning in BDI Agents

Ronan Killough, Kim Bauters, Kevin McAreavey, Weiru Liu and Jun Hong
Queen's University Belfast, Belfast, U.K.

Keywords: Online Planning, BDI Agent, Risk Awareness, Decision Making.

Abstract: The ability of an autonomous agent to select rational actions is vital in enabling it to achieve its goals. To do so effectively in a high-stakes setting, the agent must be capable of considering the risk and potential reward of both immediate and future actions. In this paper we provide a novel method for calculating risk alongside utility in online planning algorithms. We integrate such a risk-aware planner with a BDI agent, allowing us to build agents that can set their risk aversion levels dynamically based on their changing beliefs about the environment. To guide the design of a risk-aware agent we propose a number of principles which such an agent should adhere to and show how our proposed framework satisfies these principles. Finally, we evaluate our approach and demonstrate that a dynamically risk-averse agent is capable of achieving a higher success rate than an agent that ignores risk, while obtaining a higher utility than an agent with a static risk attitude.

1 INTRODUCTION

The Belief-Desire-Intention (BDI) model (Bratman, 1987) is a framework for designing rational agents in which an agent is defined by its set of beliefs (*what it knows about the world*), desires (*what changes it wants to bring about*), and intentions (*desires it has comitted to bring about*) (Rao and Georgeff, 1991). Many implementations based on the BDI model have been proposed in the literature, including AgentSpeak (Rao, 1996) and Conceptual Agent Notation (CAN) (Sardina et al., 2006) and 2APL (Dastani, 2008). However, these implementations use predefined plans instead of lookahead planning to describe the execution of intentions and they ignore the uncertainty/risk involved with undertaking actions. In real-world scenarios, actions are often uncertain (*e.g.* a door may not open as expected) or risky (*e.g.* line jumping may get you to checkout sooner, or end you up at the back of the line). Risk can be interpreted in many different ways. Here, however, it is defined as *the possibility of obtaining a utility/reward lower than the expected utility due to an undesired outcome of taking an action*.

Recently, a number of papers have considered integrating first principles planners (FPPs) into BDI implementations (Sardina et al., 2006; Chen et al., 2014; Bauters et al., 2014). This allows an agent to generate custom plans when an unexpected situation is encountered.

However, none of these contributions use the potential of a planning algorithm to assess risk. By ignoring risk, the capabilities of such agents are limited in high-stakes scenarios where pursuing high utility often entails high potential costs. Instead, we want to give an agent the ability to balance the trade-off between maximising expected utility (increasing utility) and minimising potential costs (lowering risk).

Different agents may also have different risk aversion attitudes. The approach proposed in this paper allows an agent to assess risk alongside utility, and to adapt to changed situations with varying levels of risk. The contributions of this paper are threefold. *First*, we propose a novel approach to calculate the risk of an action in an online fashion by modifying existing state-of-the-art online FPPs. *Second*, we define how an agent can interact with a planner and obtain a set of actions with associated utility and risk assessments. *Third*, we define a way to assess the expected effect of a risk reduction, and a way to induce a level of risk aversion based on the agent's current beliefs.

The following principles guide and motivate our work in specifiying how a rational agent should react:

Principle 1. A rational agent will only consider an action with a lower utility (*resp.* higher risk) when this also involves a lower risk (*resp.* higher reward).

Principle 2. An action with a lower utility will only be adopted when it *sufficiently* reduces the risk according to the desired level of risk aversion.

Principle 3. The level of risk aversion increases as the number of resources decreases.

Killough, R., Bauters, K., McAreavey, K., Liu, W. and Hong, J.
Risk-aware Planning in BDI Agents.
DOI: 10.5220/0005703103220329
In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART 2016) - Volume 2*, pages 322-329
ISBN: 978-989-758-172-4

We later show that our work satisfies these principles.

We also consider the following running example. A number of robots are trying to reach a destination (*e.g.* nuclear reactor) in succession to avoid a disaster (*e.g.* a meltdown). The behaviour of the agents, for simplicity, is fully determined by the underlying risk-aware FPP. Each robot is aware of the success or failure of the others. To reach the destination, a robot has to cross a number of bridges. Some bridges are narrower, implying a higher risk. In this scenario the agents clearly need to balance risk and utility; narrower bridges should only be taken when it leads to a higher utility, and more risk-averse behaviour is required to improve the chance of mission success.

The remainder of the paper is organised as follows. Preliminaries are given in Section 2. In Section 3, we propose a modified version of a UCT-based online planner capable of assessing risk alongside utility. Section 4 outlines how such a planner can be integrated with a BDI agent to produce a set of utility-ranked actions. The applicability of our approach is validated in Section 5 and related work and conclusions are discussed in Section 6.

2 PRELIMINARIES

We start with some necessary preliminaries on the CAN language, Markov Decision Processes (MDPs), and Monte-Carlo algorithms for online planning.

BDI Agents. In CAN (Sardina et al., 2006), an agent configuration, or agent, $\mathcal{C}$ is a tuple $\mathcal{C} = \langle \mathcal{B}, \Pi, \Lambda, \Gamma, \mathcal{H} \rangle$. The belief base $\mathcal{B}$ is a set of atoms describing the agent's current beliefs. The plan library Π consists of a set of pre-defined plans of the form $e : \psi \leftarrow P$ with e an event-goal, ψ the *context*, or preconditions, and P the plan body. The plan body may consist of operations to add/delete beliefs $+b, -b$, trigger events $!e$, tests for conditions $?\phi$, or execute primitive actions a. Primitive actions are defined in the action description library Λ. The intention base Γ consists of a set of (partially) executed plans P. Finally, $\mathcal{H}$ is the sequence of primitive actions executed by the agent so far. A *basic configuration* $\langle \mathcal{B}, \mathcal{H}, P \rangle$ is also often used in notation.

The operational semantics of CAN are described in terms of configurations C and transitions $C \rightarrow C'$. A derivation rule describes in which cases the agent can transition to a new configuration. Such a rule consists of a (possibly empty) set of premises p_i, and a single transition relation c as conclusion:

$$\frac{p_1 \quad p_2 \quad \cdots \quad p_n}{c}\ l$$

We refer the reader to (Sardina et al., 2006) for an overview of the full semantics.

Online Planning in MDPs. An MDP consists of a set of states $\mathcal{S}$ and actions A, a transition function T, and a reward function R. The transition function is defined as $T : \mathcal{S} \times A \times \mathcal{S} \rightarrow [0,1]$, *i.e.* given a state s and an action a, we transition to a new state s' with probability $T(s,a,s')$. Rewards are defined as $R : \mathcal{S} \times A \times \mathcal{S} \rightarrow \mathbb{R}$ with $R(s,a,s')$ the reward for taking action a in state s and arriving in state s'. A discount factor γ prioritises immediate rewards. Our aim is to find the best action at each state. Such a mapping between states and actions is referred to as a *policy*. Since the objective is to maximise the overall expected reward, an optimal policy will consistently select actions which will maximise both immediate and potential future rewards. We can define a non-stationary policy[1] as $\pi_t : \mathcal{S} \rightarrow A$ with t as the current time step. The quantitative *value* of a policy π with an initial state s_0 is given by:

$$V(\pi(s_0)) = E\left(\sum_{t=0}^{h-1} \gamma^t \cdot R(s_t, \pi_t(s_t), s_{t+1})\right)$$

Here, $\pi_t(s_t)$ produces an action according to policy π_t. The optimal policy π^* maximises $V(\pi^*(s_0))$.

Finding optimal policies in this way is infeasible as the size of the problem grows. In recent years focus has therefore shifted to approximate methods such as Monte Carlo Tree Search (MCTS). Planners which employ such methods are often referred to as *online* or *agent-centric* planners because they do not produce a policy, but simply return the single next best action to execute. These methods use sampling to quickly build up the most promising part of the search tree and allow a *"good enough"* action to be returned at any time. In each cycle, (1) a leaf node is selected, (2) that node is expanded with a new child node, (3) a simulation of a random playout from this new child to a terminal state (or the horizon) is performed, and (4) the rewards from this simulation are backpropagated towards the root node to guide future searches. Improvements to the selection phase, *e.g.* using UCT (Kocsis and Szepesvári, 2006), have led to very competitive online planning algorithms. A comprehensive overview of MCTS approaches can be found in (Browne et al., 2012).

3 ASSESSING RISK ONLINE

In this section, we discuss how an online planner can be modified to assess risk alongside utility. We

[1] The behaviour of a nonstationary (*resp.* stationary) policy is dependent (*resp.* independent) on the current timestep.

start with the popular UCT algorithm (Kocsis and Szepesvári, 2006), which combines MCTS with a Multi-Bandit selection procedure to balance exploration and exploitation (Auer et al., 2002). We modify UCT so that it provides a set of utility-ranked actions, each with an associated risk assessment, instead of a single "best" action. UCT constructs a biased layered search tree where higher utility nodes are explored more thoroughly while also biasing exploration toward rarely visited branches (Keller and Eyerich, 2012). As shown in Figure 1, layers in the tree alternate between decision nodes and chance nodes (*resp.* states and actions). The children of a decision node reflect the actions available at this state. The children of a chance node reflect the states obtained as a stochastic outcome of applying the action.

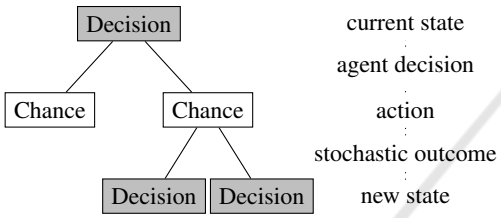

Figure 1: Chance and decision nodes.

3.1 Establishing a Risk Metric

There are a number of ways to produce a measure of risk from knowledge of rewards and probabilities of occurrence (Johansen and Rausand, 2014). The most appropriate metric for this framework is variance. We therefore treat the *immediate* risk of taking an action in a given state as the probability-weighted variance of rewards from that action's outcomes.

Definition 1. The *immediate risk* of taking action a in state s, denoted $IR(s,a)$, is defined as:

$$IR(s,a) = \sum_{s' \in \mathcal{S}} T(s,a,s') \cdot (R(s,a,s') - E(s,a))^2$$

where $E(s,a) = \sum_{s' \in \mathcal{S}} T(s,a,s') R(s,a,s')$ is the expected utility of taking action a in state s. $IR(s,a)$ is simply the probability weighted variance of the outcome rewards of an action a. □

To assess the risk of each immediate action, we compute the variance of this chance node, where the expected utility functions as the mean. However, a number of issues prevent us from doing so directly. One problem is that the classical way of computing variance as shown above is an offline method which assumes all samples are known. Crucially, an online planning algorithm has to be able to return a solution that is "good enough" at any moment. Therefore, this approach for calculating variance is unsuitable. Instead, we must rely on approximations of variance that work in an online setting such as the method proposed in (Welford, 1962). Another problem is that we must treat decision and chance nodes differently. Indeed, a chance node imposes a risk due to the stochastic outcome. In a decision node, an agent has a choice of which action to take. As a result, the risk in a decision node should adhere to rational decision constraints where it can be viewed as *e.g.* the risk of the least risky action available.

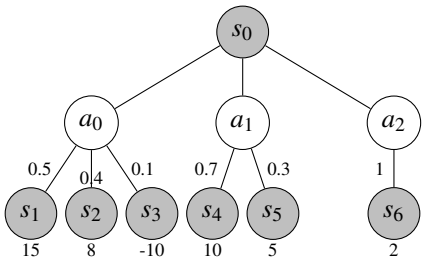

Figure 2.

We now formalise these ideas. For simplicity, we first consider a full search tree to explain how we compute risk, and how the notion of *risk exposure* is used to associate a level of risk with decision nodes. The search tree is depicted in Figure 2, with alternating decision and chance nodes, probabilities of transitions on the edges, and rewards below the leaf nodes.

Example 1. Clearly, from Figure 2, we can see that taking action a_2 does not involve risk. Indeed, we are guaranteed of the outcome, so there is no chance of obtaining any other utility than the expected one. Similarly, $IR(s_0,a_2) = 0$ based on 100 samples. This is not the case for actions a_0 and a_1. For action a_0 it is easy to verify that the expected utility is $E(s_0,a_0) = 0.5 \cdot 15 + 0.4 \cdot 8 + 0.1 \cdot (-10) = 9.7$. However, we find that $IR(s_0,a_0) = 54.0$ based on 100 samples. For a_1 we have $E(s_0,a_1) = 8.5$ and $IR(s_0,a_1) = 5.3$. Intuitively, a_0 is indeed the riskiest action since we have the potential for a wider range of rewards. □

While it is possible to simply use immediate variance as the risk indicator for an action, this would tend toward very short-sighted behaviour with regards to risk. Consider the example of the choice between joining the army and becoming an accountant. Both immediate actions (apply for, and accept, the job) have the same, very low, risks. The level of risk one would be *exposed to* having taken these initial actions is, however, very different. In order to make informed decisions, our risk metric must consider both immediate risk and the risk of future actions.

We only calculate the risk measure for chance nodes, which have stochastic outcomes, as only these

nodes can convey an unexpected outcome. Risk as we define it does not apply to decision nodes since generally there is a choice of which action to take. To consider the risk of future actions however, we introduce the notion of *risk exposure* to quantify the risk of being in a state with a choice of actions.

Definition 2. Let $A_s \subseteq A$ be the set of available actions in state s. We have:

$$IRE(s) = \min_{a \in A_s} IR(s,a).$$

With $IRE(s)$ as the *immediate risk exposure* of s. □

We define it as such since this is the minimum amount of risk which the agent *must* take. Note that this minimum risk action is not guaranteed to be the action the agent *will* take in this state if it is encountered. However, when weighing cumulative expected utility against cumulative risk it is more useful to have a risk measure indicating the risk the agent *must* take by following a certain path, rather than some distribution of the available risks at each step.

Definition 3. Let $A_s \subseteq A$ be the set of available actions in state s and γ^t be a discount factor for time t. Then the *cumulative minimum risk* up to horizon h of taking action a in state s, denoted *CMR*, is defined as:

$$CRE(s,h) = \min_{a \in A_s} CMR(s,a,h) \tag{1}$$

$$CMR(s,a,h) = \sum_{t=1}^{h-1} \left(\gamma^{t-1} \sum_{s' \in \mathcal{S}} T(s,a,s') \cdot CRE(s',h-1) \right) \tag{2}$$

where $CRE(s,h)$ is the *cumulative risk exposure* up to horizon h of state s such that $\min(\emptyset) = 0$. □

The *CRE* is identical to $IRE(s')$ except that it considers the cumulative minimum risk of an action a instead of the immediate risk. *CMR* is used as our risk metric when considering actions. Note that it is based on the *probability adjusted* risk exposure of relevant states $s' \in \mathcal{S}$ (*i.e.* $s' \in \mathcal{S}$ such that $T(s,a,s') > 0$).

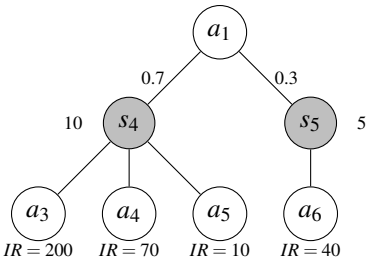

Figure 3.

Example 2. We now extend Figure 2 to show outcomes and subsequent available actions from action a_1. From s_4 we can take actions a_3, a_4 or a_5 and from state s_5 we have a single action choice a_6. The subtree with a_1 as its root is shown in Figure 3. For simplicity, a set of terminal state outcomes are implied for each leaf action $a_3 \ldots a_6$. We know the *immediate* risk (*IR*) of a_1 is 5.3 and the *IR* of a_3, a_4, a_5 and a_6 is *resp.* 200, 70, 10 and 40. In order to determine the *CMR* of taking a_1, we must first calculate the risk exposure of its child decision nodes s_4 and s_5. This is given by $CRE(n) = \min_{c \in C_n} \big(IR(n,c)\big)$ where n is a decision node and each $c \in C_n$ a child chance node. These risk exposure values will then be modulated by the decision node's likelihood of occurrence. We thus assess *CMR* of a_1 as $IR(s_0,a_1) + CRE(s_4) \cdot 0.7 + CRE(s_5) \cdot 0.3 = 5.3 + 10 + 40 = 55.3$. □

3.2 Approximate Algorithms

In calculating *CMR*, we have thus far assumed that the tree is fully known. However, in order to harness the speed and tractability afforded by agent-centric planning algorithms, we provide a method of calculating our proposed risk metric in an online fashion. This is carried out *alongside* the usual accumulation of expected reward. When not all outcome rewards are known and only a single value is sampled at a time, an algorithm is required which is capable of maintaining a running variance estimate based on the rewards of the nodes which have been sampled so far.

During the backpropagation step, the reward associated with the most recently sampled decision node and its parent (chance node) is used to update the variance estimate of the chance node. This value is then compared to the current variance estimates of all sibling nodes and, if a sibling exists with lower variance, then the variance of that sibling will be backpropagated to the parent (decision) node instead.

An issue which arises when calculating variance online is that accuracy is severely reduced when sample counts are very low. This is exacerbated when we backpropagate the lowest variance node, since this will often be a node which has not been sampled and thus has a variance of zero. To tackle this problem, we further modify the UCT procedure to include multiple "variance" rollouts. This provides a preliminary estimate of the variance of a newly generated chance node with an accuracy corresponding to the number of rollouts performed, denoted as ρ. Unlike traditional rollouts carried out in UCT, which simulate a playout of the domain until a terminal state is reached, variance rollouts repeatedly sample the outcomes of a single leaf decision node to build up a preliminary estimate of immediate variance for this node.

We can now obtain a reasonably accurate variance estimate for each action by the time it is initially sam-

pled. When the sibling nodes are compared during backpropagation, nodes with no samples are ignored. Nodes with at least one sample can now be guaranteed to already have a reasonably accurate variance estimate. Both the variance rollout and backpropagation steps of the algorithm are shown *resp.* in Algorithm 1 and Algorithm 2. The selection, expansion and rollout procedures are omitted for brevity as they remain largely unchanged from a plain UCT implementation.

Algorithm 1: Variance rollout.

```
1:  function VarRollout(n)
2:      if visits(n) = 0 then
3:          n ← parent(n)
4:          visits(n) ← visits(n) + 1
5:          for i ← 0, ρ do
6:              outcome ← SimulateAction(node)
7:              rwd ← GetRwd(action(n), outcome)
8:              risk(n) ← UpdateRisk(n, rwd)
9:          risk(n) ← risk(n)/ρ
10:         M(n) ← 0
11:         mean(n) ← 0
12:     n ← SimulateAction(n)
13:     return n
```

Both algorithms assume that the parameter (n) is a decision node containing a state. In Algorithm 1, the $UpdateRisk(\cdot)$ function implements an online variance calculation (Welford, 1962), taking a new reward and producing a running variance estimate, which is then saved in $risk(n)$.

Algorithm 2: Backpropagation.

```
1:  function Backpropagation(n)
2:      while n ≠ nil do
3:          cReward ← 0
4:          cRisk ← 0
5:          visits(n) ← visits(n) + 1
6:          outcome ← state(n)
7:          n ← parent(n)
8:          visits(n) ← visits(n) + 1
9:          rwd ← GetRwd(action(n), outcome)
10:         risk(n) ← UpdateRisk(n, rwd)
11:         cReward ← cReward + rwd · γ^depth
12:         cRisk ← cRisk + risk(n) · γ^depth
13:         lwstRisk ← lwstRSib.risk/lwstRSib.visits
14:         if cRisk > lwstRisk then
15:             cRisk ← lwstRisk
16:         UpdateReward(n, cReward)
17:         n ← parent(n)
```

The backpropagation algorithm shown in Algorithm 2 steps back through the constructed tree until the root node is reached ($parent(n) = nil$). Lines 8 and 9 show the reward and risk being accumulated and discounted by γ^{depth}. Lines 10 to 12 state that if a sibling with a lower risk exists (*lwstRSib*), then the value to be backpropagated (*cRisk*) should accumulate the risk level of that sibling, instead of the risk level of the node sampled. This has the effect of building *CMR* estimates for each of the root node's children, which correspond to the agent's available actions. The function $UpdateReward(\cdot)$ adds the accumulated reward from this backpropagation to the node n's total accumulated reward.

4 A RISK-AWARE BDI AGENT

Agents with the ability to perform lookahead planning can respond to an unforeseen event by generating custom plans for these events. In this section, we improve on this idea by describing how an agent can decide between actions given both utility and risk assessments from an online planner. We consider scenarios in which we may encounter dilemmas between risk and utility and must therefore be able to make risk-aware decisions. To do this, the agent must have a choice of actions from which it can select based on some decision strategy. Such a strategy must be capable of appropriately balancing the utility and risk associated with an action. The latter part of this section addresses how a BDI agent can adjust its level of risk aversion based on beliefs about its environment.

4.1 Integrating an Online Planner

Returning multiple actions from an online planner requires a trivial modification to existing algorithms. This is because online planners rank actions by utility and simply return the action with highest utility. Instead of returning a single action, we return a set of *assessed actions* $\mathcal{A}$, where each $\alpha \in \mathcal{A}$ is a tuple $\langle a, u, r \rangle$ with a as a primitive action, u the cumulative utility approximation and r the *CMR* approximation for that action, as produced by the planner. We denote the cumulative utility and *CMR* of an assessed action α as *resp.* $u(\alpha)$ and $r(\alpha)$, and the primitive action as $a(\alpha)$. An action with a lower utility is not guaranteed to also have a lower risk. It is therefore possible to have an assessed action with both a lower utility and a higher risk than that of alternative actions. Such actions generally pose no benefit to the agent and are referred to as *irrational*.

Definition 4. Let $\mathcal{A}$ be a set of assessed actions. The set of *rational assessed actions* $\mathcal{A}_R$ in $\mathcal{A}$ is defined as:

$$\mathcal{A}_R = \{\langle a,u,r\rangle \in \mathcal{A} \mid \nexists \langle a',u',r'\rangle \in \mathcal{A}, u \leq u' \wedge r \geq r'\}.$$

The set of *irrational assessed actions* in $\mathcal{A}$, denoted $\mathcal{A}_I$, is defined as $\mathcal{A}_I = \mathcal{A} \setminus \mathcal{A}_R$. □

Consider the running example. A robot is one decision away from its goal of reaching the reactor (also see Figure 4). We have the assessed actions $\alpha_0 =$

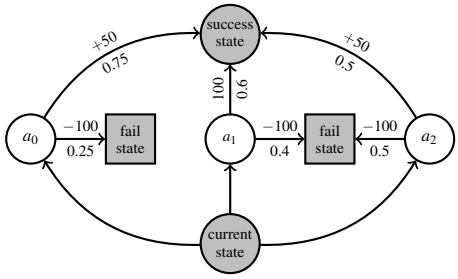

Figure 4: A simple risk/utility trade-off.

$\langle a_0, 12.5, 4218.75\rangle$, $\alpha_1 = \langle a_1, 20, 9600\rangle$, and $\alpha_2 = \langle a_2, -25, 5625\rangle$. We can see that $\mathcal{A}_R = \{a_0, a_1\}$. The expected utility of a_0 is $0.75 \cdot 50 + 0.25 \cdot -100 = 12.5$, and its *IR* value is 4218.75. These same calculations give a_1 a utility of 20 and an *IR* value of 9600. The agent must decide whether to take a riskier action for a possibility of higher gain (a_1), or take the safer option (a_0) and gain a lower reward if successful. This decision must be based upon the agent's determined level of risk aversion, which in turn is based on its beliefs about its current situation.

4.2 Risk-aware Decision Making

We now discuss a decision rule capable of balancing the risk and utility of actions produced by the planner in order to decide between rational assessed actions. The BDI agent has no control over how the FPP produces its results, but chooses from the actions returned. The agent's influence is therefore purely in the post-planning phase. This rule requires a defined risk aversion level R in order to produce a single next best action from a set of assessed actions.

Unlike in the BDI agent framework, "goals" in MDPs are expressed implicitly in the rewards assigned to action outcomes. Therefore in order to have the planner be goal-oriented, we must assign some relatively high reward to the goal state(s) and relatively low rewards to any fail state(s)[2]. By consistently selecting the best action in terms of utility, as in plain UCT, the agent should accumulate a higher total reward over a sequence of actions *if* it is successful in reaching the goal state. However, when considering scenarios where there are trade-offs between risk and utility, we can observe that taking lower risk actions increases the likelihood of success while obtaining a lower overall reward if successful.

In order to appropriately balance utility and risk given the information from the planner, we treat utility and risk as *resp.* the mean and variance. The interval $u(\alpha) \pm R\sqrt{r(\alpha)}$ provides the interval in which we can reasonably expect to find the expected utility given a risk aversion level R. Since we are risk-averse, we only consider the lower bound of each interval, and select the action with the maximum lower bound to minimise our potential worst-case. For actions with higher risk values, raising the R will widen the interval more significantly than for actions with lower risk. Thus, a risk aversion level can be defined in terms of R that will ensure our model satisfies Principle 2.

Definition 5. Let $\mathcal{A}$ be a set of assessed actions and $R \in [0, +\infty]$ be a risk aversion degree. Then the *optimal assessed action* in $\mathcal{A}$, denoted α^*, is defined as:

$$\alpha^* = \arg\max_{\alpha \in \mathcal{A}} \left(u(\alpha) - R\left(\sqrt{r(\alpha)}\right)\right). \quad \square$$

This formula will select a "best" action α^* from the set of assessed actions according to a risk aversion setting R. $R = 0$ corresponds to absolute risk *tolerance*, as the lower bound of the interval will be equivalent to the utility. Therefore at $R = 0$, maximising the lower bound will give us the action with the highest utility and also, if only considering rational actions, the highest risk. As R is increased, when considering only $\mathcal{A}_R$, the lower interval bound of high utility actions will begin to fall below that of lower utility actions, due to their lower associated risk values. This means as we raise R, we will increasingly accept lower utility ($u(\alpha)$) actions provided they have a proportional reduction in associated risk ($r(\alpha)$).

Proposition 1. Let $\mathcal{A}$ be a set of assessed actions, given Definition 5, then $\alpha^* \in \mathcal{A}_R$.

Proof. Since $\forall \alpha_I \in \mathcal{A}_I, \exists \alpha \in \mathcal{A}\ u(\alpha) > u(\alpha_I) \wedge r(\alpha) < r(\alpha_I)$, at $R = 0$, the rule defined in *Definition* 5 will consistently select the action with the highest $u(\alpha)$. This eliminates the possibility of an action from $\mathcal{A}_I$ being selected since otherwise $u(\alpha) > u(\alpha_I)$ would not hold. For any given R where $R > 0$, the lower bound $R\sqrt{r(\alpha)}$ of the interval will never be greater than other actions' lower bounds since otherwise $r(\alpha) < r(\alpha_I)$ would not hold. $\square$

At $R = \infty$, the lowest risk action would always be chosen, regardless of any loss of utility. We define *maxR* as the value at which, for a given domain, the probability of successfully reaching the goal is highest. Values above this will be more risk-averse, but will not necessarily produce goal-oriented behaviour in the agent. Since goals in MDPs are implicitly encoded in the reward values of actions; a completely risk-averse agent will not be sufficiently utility-driven to ensure it progresses towards its goal.

As R is increased, the agent should take paths towards a goal which have higher probabilities of success. Of course, these paths may not exist in a given

[2]For our purposes, goal and fail states must also be terminal states.

$$\mathbf{p_i}$$
$$+!robot\_failed : robots\_left(3) \leftarrow risk(a);$$
$$-robots\_left(3); +robots\_left(2).$$
$$+!robot\_failed : robots\_left(2) \leftarrow risk(b);$$
$$-robots\_left(2); +robots\_left(1).$$

Figure 5: Plan sets which alter the agent's R value when an event is triggered. Where for $\mathbf{p_1} : a = -3.5, b = -1$ and for $\mathbf{p_2} : a = -2.5, b = -2$.

domain. It is perfectly possible that the path providing the highest payoff is also the path with the lowest risk. In such domains, awareness of the risk provides no benefit. We argue however that this risk vs. reward dilemma is common, as committing less resources to achieving a goal tends to entail a higher risk of failure.

As outlined in Principle 3, an agent's risk aversion level R should be dynamic with respect to its situation. To achieve this, we utilise the agent's plan library Π and instantiate a set of pre-defined plans which alter the R value when a set of preconditions are met. We then define a CAN derivation rule describing this behaviour. The plan's preconditions represent states considered to signal a change in the agent's risk state. This enables the agent to adopt an appropriate risk aversion level based on its current situation.

We now extend the standard agent configuration to include our risk aversion value R, producing a tuple $\mathcal{C} = \langle \mathcal{B}, \Pi, \Lambda, \Gamma, \mathcal{H}, R \rangle$. We illustrate this idea in the context of the running scenario by considering three robots simultaneously trying to reach the goal. The number of robots left is treated as an indicator of the risk aversion level remaining robots should adopt. This assumes awareness of the failure/success status of the other agents. Given three robots operating simultaneously, the chances of reaching the goal successfully with any given strategy is increased. Each robot should therefore be more risk tolerant (lower R). If a robot fails, however, the remaining robots should become more risk-averse (higher R), to ensure a similar chance of success as before. The plans outlined in Figure 5 increase an agent's risk aversion level when another agent has failed.

We now define a CAN derivation rule describing the semantics of the $risk()$ plan shown in Figure 5.

$$\frac{k \in \mathbb{R} \qquad R' = R + k}{\langle \mathcal{B}, \mathcal{H}, risk(+k), R \rangle \longrightarrow \langle \mathcal{B}, \mathcal{H}, nil, R' \rangle} \; risk$$

The event $robot\_failed$ indicates to surviving agents that another robot has failed. Depending on the number of robots left (comprising the plan's *context*), the risk level is adjusted accordingly and the beliefs about the number of robots remaining is updated.

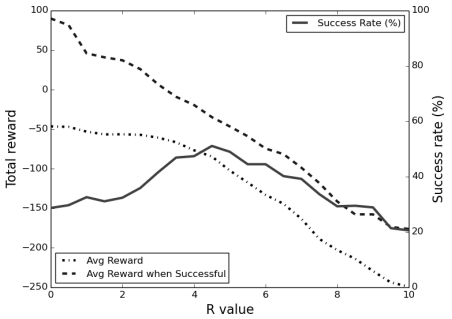

Figure 6: Different levels of risk aversion for a single agent.

5 EVALUATION

We first demonstrate that raising an agent's risk aversion level will indeed cause it to take less risky actions and, by extension, have a greater probability of successfully reaching its goal. We illustrate this using the running example. We assign a reward of 100 and -100 for *resp.* reaching the goal or falling in the pits. Negative rewards (costs) are assigned to actions based on how much time they will take to carry out, with higher costs for actions reflecting longer paths.

The graph in Figure 6 shows the effects of increasing R values on the scenario outcome. For each setting of R, the scenario is run through to completion (either succeeding or failing) 1000 times. As can be seen, The probability of success initially increases (at the cost of diminishing overall reward) as R is increased, showing that increasing risk awareness causes the agent to select less risky paths over those with greater expected utility. However, probability of success only increases up to a point. Further bias toward risk aversion causes the agent to be overly cautious and not progress toward the goal, leading to both diminishing total reward and likelihood of success.

To observe the benefits of situation-dependant risk awareness, we run the scenario with three agents, one of which must reach the destination for the mission to succeed. For comparison, we first run the scenario with various static R values. For these agents, the risk aversion level remains the same regardless of the number of robots remaining. We then consider two sets of *dynamic* R values with R changing depending on the number of robots remaining. Their risk aversion level is therefore reactive to changes in the environment. The results are shown in Figure 7. The dynamic values correspond to plans p_1 and p_2 respectively, as per Figure 5. The dynamic R values show an increased likelihood of success with only a slightly reduced total reward compared to a low, static risk aversion level such as $R = 1$.

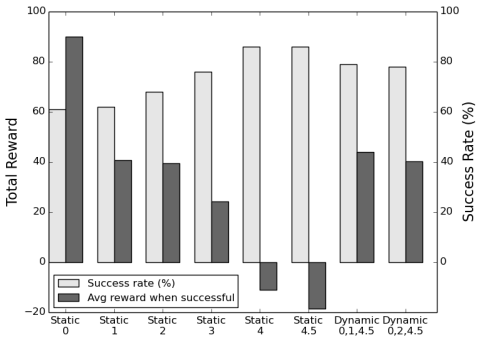

Figure 7: Static/dynamic R values in agent collaboration.

6 CONCLUSIONS

The MARAB algorithm (Galichet et al., 2014), is a risk-aware multi-armed bandit algorithm. It provides an alternative to the UCB1 selection formula and treats the conditional value at risk as the measure of branch quality. This approach can deal with risk, but does not offer any decision making aspect. Other approaches, such as (Liu and Koenig, 2008), solve MDPs in a risk sensitive manner, while taking resource levels into consideration. This is done by making use of a type of non-linear utility functions known as *"one-switch"* utility functions. However, this approach computes offline policies, and thus cannot readily be integrated with the highly dynamic framework of BDI.

In this paper we presented a novel approach to calculate risk alongside utility in popular MCTS algorithms. We showed how such an online planner can be integrated with a BDI agent. This integration allows an autonomous agent to reason about its risk tolerance level based on its current beliefs (*e.g.* the availability of resources) and dynamically adjust them. It also allows such an agent to react to unforeseen events, an approach impossible in BDI agents that only use predefined plans. Furthermore, our proposed framework agrees with the principles as outlined in Section 1, enabling an agent to act appropriately in high-stakes environments. Experimental results underpin our theoretical contributions and show that taking risk dynamically into account can lead to higher success rates with only minimal reductions in utility compared to agents with static risk aversion levels.

ACKNOWLEDGEMENTS

We would like to thank Carles Sierra, Lluís Godo, and Jianbing Ma for their inspiring discussions and comments. This work is partially funded by EPSRC PACES project (Ref: EP/J012149/1).

REFERENCES

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *ML*, 47(2-3):235–256.

Bauters, K., Liu, W., Hong, J., Sierra, C., and Godo, L. (2014). CAN(PLAN)+: Extending the operational semantics of the BDI architecture to deal with uncertain information. In *Proc. of UAI'14*, pages 52–61.

Bratman, M. (1987). *Intention, Plans, and Practical Reason*. Harvard University Press.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of Monte Carlo tree search methods. *IEEE Transactions on comp. intell. and AI in games*, 4(1):1–43.

Chen, Y., Bauters, K., Liu, W., Hong, J., McAreavey, K., Sierra, C., and Godo, L. (2014). AgentSpeak+: AgentSpeak with probabilistic planning. In *Proc. of CIMA'14*, pages 15–20.

Dastani, M. (2008). 2APL: a practical agent programming language. *JAAMAS*, 16(3).

Galichet, N., Sebag, M., and Teytaud, O. (2014). Exploration vs exploitation vs safety: Risk-averse multi-armed bandits. *JMLR: Workshop and Conference Proceedings*, 29:245–260.

Johansen, I. L. and Rausand, M. (2014). Foundations and choice of risk metrics. *Safety science*, 62:386–399.

Keller, T. and Eyerich, P. (2012). PROST: Probabilistic planning based on UCT. In *Proc. of ICAPS*.

Kocsis, L. and Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In *Proc. of ECML'06*, pages 282–293.

Liu, Y. and Koenig, S. (2008). An exact algorithm for solving MDPs under risk-sensitive planning objectives with one-switch utility functions. In *Proc. of AAMAS'08*, pages 453–460.

Rao, A. S. (1996). AgentSpeak (L): BDI agents speak out in a logical computable language. In *Agents Breaking Away*, pages 42–55. Springer.

Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. *KR*, 91:473–484.

Sardina, S., de Silva, L., and Padgham, L. (2006). Hierarchical planning in BDI agent programming languages: A formal approach. In *Proc. of AAMAS'06*, pages 1001–1008.

Welford, B. (1962). Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420.