Safe Multi-objective Planning With A Posteriori Preferences

Ralph Eastwood, Rob Alexander and Tim Kelly
University of York
York, United Kingdom
Email: ralph@cs.york.ac.uk,{rob.alexander,tim.kelly}@york.ac.uk

Abstract*—Autonomous planning in safety critical systems is a difficult task where decisions must carefully balance optimisation for performance goals of the system while also keeping the system away from safety hazards. These tasks often conflict, and hence present a challenging multi-objective planning problem where at least one of the objectives relates to safety risk. Recasting safety risk into an objective introduces additional requirements on planning algorithms: safety risk cannot be "averaged out" nor can it be combined with other objectives without loss of information and losing its intended purpose as a tool in risk reduction. Thus, existing algorithms for multi-objective planning cannot be used directly as they do not provide any facility to accurately track and update safety risk. A common workaround is to restrict available decisions to those guaranteed safe a priori, but this can be overly conservative and hamper performance significantly. In this paper, we propose a planning algorithm based on multi-objective Monte-Carlo Tree Search to resolve these problems by recognising safety risk as a first class objective. Our algorithm explicitly models the safety of the system separately from the performance of the system, uses safety risk to both optimise and provide constraints for safety in the planning process, and uses an ALARP-based preference selection method to choose an appropriate safe plan from its output. The preference selection method chooses from the set of multiple safe plans to weigh risk against performance. We demonstrate the behaviour of the algorithm using an example representative of safety critical decision-making.

***Keywords*-Safety critical systems, Planning, Safety, POMDP, Monte-Carlo Tree Search**

I. Introduction

Safe decision making will always concern trade-offs between safety and other performance characteristics of the system. This trade-off, naturally done by human decision-makers, is challenging for autonomous systems of unmanned vehicles and process industries.

The safety-performance trade-off has two goals: optimising the preformance-related attributes of the system when they are within acceptably safe limits, and, to prioritise safety over these performance-related attributes when safety and performance conflict with each other. Additionally, it is desirable that these safe limits still allow decisions to be made safer if there is no prohibitively large cost associated with the decision.

Traditionally, techniques based on Partially Observable Markov Decision Processes (POMDP) have both theoretical and practical limitations [1] in solving planning problems with multiple objectives. In particular, these limitations prevent any sensible method of combining safety attributes along with performance attributes and hence it is difficult to apply the safety-performance trade-off in single-objective POMDP planning based on dynamic programming.

To tackle this problem, we apply a recently introduced algorithm by Wang and Sebag [2] that enables using the safety-performance trade-off within a practical POMDP-based multi-objective planning algorithm. As shown in this paper, the algorithm provides us with the necessary tools to both treat safety as a first-class concept and objective alongside other performance-related objectives as well as select a final preference after gathering all possible solution plans (i.e. *a posteori*).

Preference selection in safety critical systems involves some concept of risk as it relates in safety – i.e. risk to human life. We use the conceptual framework of the As Low As Reasonably Practicable (ALARP) principle in this paper to be used as a preference selection mechanism for planning with objectives of cost and risk. The ALARP principle [3] is used outside normal decision-making to make decisions in safety-related systems in the UK and is typically defined as the act of reducing risk such that further reduction of risk is grossly disproportionate to the cost benefit gained.

Our paper builds upon the concept of decision-making that is currently used in POMDP-literature and adapts it for safety critical applications through the use of multi-objective planning. We propose a variant of the Monte-Carlo Tree Search algorithm that integrates methods to ensure that the decisions made are safe.

II. The Safety Planning Problem

The safety planning problem is one where individual decisions can be safety critical. Hence, the aim of a safety planner is to generate a policy (as a solution for the safety planning problem) that includes contingencies that take into account the safety critical behaviour of the system. We characterise safety as an objective separate from other system performance objectives.

A. Problem Structure

As safety critical systems have to deal with real world environments, we frame the safe decision-making problem as a variant of a Partially Observable Markov Decision

Process (POMDP) which has the ability to model stochastic real world processes. The role of the decision-maker is to make a sequence of decisions which influences the safety of the system: at runtime, risk may be increased or decreased through a combination of decisions.

As POMDPs are only a mathematical abstraction of the system, this section considers a number of characteristics that are specific to the problem of safe decision-making. In particular, we need to consider how to structure the state space of the problem, how the planner is guided to search for a good policy, and, what the practical and safety-related requirements of the POMDP representation of the decision problem and the planning algorithm are.

B. Guiding the Planner

The planner is guided through the search space by two broad characteristics that need to be defined within the POMDP representation:

- Using rewards until the algorithm converges and finds a sequence of decisions maximise cumulative reward.
- Using costs (negative rewards) until the algorithm finds a sequence of decisions that can reach a goal state with the minimum costs.

Following the advice and reasoning stated by Hansen [4], we take the approach of using both costs and goal states. This approach is more suitable for safety critial real world problems as the goals can be explicitly defined.

C. Structuring the Model

Constructing a model for a particular application is domain-specific. Our particular modelling scheme explicitly captures both the process and safety artifacts of the safety critical system by structuring the model around safety-related concepts of *hazards* and *modes* (the mode of operation of the system). We propose modelling the system using three types of model to provide the necessary information for the planning algorithm to determine safe decisions, namely:

- the *process model*: the model of the entire decision process which defines the search space that the planner needs to make performance-related decisions but not necessarily be able to calculate safety-related ones accurately.
- the *mode model*: derived from the process model, it models the current mode of the system which dictates the safety criticality of the system – this is necessary to determine what risk models need to used to measure the risks of hazards within the system, and
- the *risk model(s)*: derived from the process model, it models potential hazards of individual decisions and provides the a measure of risk for failures of the system.

We recognise that there can be alternative formulations of modelling. The process model and risk models could be combined together to form one very complex model. The merit of this idea is that the process model can be easily supplemented with safety information. However, we recognise there are two issues. Firstly, mixing both the complexities of the process and risk models can hide the important detail about safety. Secondly, a complex process-risk model can only be updated approximately – keeping them both separate allows an approximate update of the non-safety related process model and an exact update of the risk model that provides safety critical information.

Furthermore, this separation allows computational benefits as it is effectively a factorisation (a minimisation of the POMDP state search space) of the model representation – we can update the process model approximately when running expensive simulations in our planning algorithm, whereas the safety and mode models are crucial to making *safe* decisions, and can be updated accurately when needed. As described later in Section III, this also avoids the problem where a limited number of Monte-Carlo samples may never capture low-probability events which are safety critical.

III. Multiobjective Partially Observable Monte Carlo Planning

A. Modelling the Process

One of the difficulties with using POMDPs to model a real decision process is providing a way to terminate the planning process. The standard representations of POMDPs and Goal-oriented POMDPs do have the ability to terminate, but are very restrictive for the following reasons [4]:

1) Standard POMDPs rely on a discount factor and tolerance parameters which make the assumption that as decisions in the future are less likely to impact the decision now; however, there is no guidance on how to define these parameters. Furthermore, information about future objectives is lost through repeated multiplying with the discount factor.
2) Decision processes in Goal-oriented POMDPs only terminates when the probability that the decision process is in goal states sum to one. Due to noisy sensors, this may not always be the case. Using tolerance parameters could solve the problem but they would also need to be tuning to the problem.

We address this by using a slightly stricter variation of Hansen's "Partially Observable Markov Decision Process with Action Termination" (POMDP-ACT) [4] to model the process. The chief differences from the standard POMDP representation are as follows:

1) All costs must be positive. Rewards (costs that are negative) are often used traditionally as it matches the concept of positive reinforcement and is sufficient for many domains – in safety-critical systems, risk is the predominant concept which requires costs.
2) POMDP-ACT has terminal states that explicitly model states where the process being modelled stops. These

are not present in the standard POMDP explicitly, but match the concept of goal states in Goal-POMDPs.

3) POMDP-ACT introduces termination actions that will always transition to a terminal state from any state. This differs from both standard and Goal-POMDPs, and helps design decision processes which can always terminate and return some policies.

More precisely, we can define a POMDP-ACT as a tuple $M_p : \langle S, A, Z, T, O, C \rangle$ where:

- S, A, Z - are the sets of: discrete states, actions, and observations, respectively.
- $T(s', s, a)$ - is the probability function that a state s transitions to a state s' given that an action a is performed.
- $O(z, s', a)$ - is the probability function that an observation z is observed given that the state transitioned to is s' when the action a is performed.
- $C(s, a)$ - is the cost of performing an action in state s. This is also known as negative reward. It is a vector corresponding to each objective.

B. Mode and Risk Models

Constructing mode and risk models separately from the process model can easily lead to having them inconsistent from each other. As the process model should be adequately defined to contain the information required to distinguish modes of the system as well as the risk present within the system, the mode and risk models can be defined through a transformation from the process model (we recognise that this is a separate and difficult problem within itself). The chief difference is the ability to update the mode and risk models accurately and efficiently to detect the safety-criticality of the system and allow the planner to perform accordingly. As these models do not have any decision-making ability of their own, they are defined as Hidden Markov Models (HMM) rather than POMDPs.

The mode model is defined as a tuple: $M_m : \langle S_m, Z_m, O_m \rangle$ where:

- S_m - is the set of discrete states that are sufficient to represent the different modes of the system's process.
- Z_m - is the set of discrete observations that can be observed that influence the belief of the mode.
- $O_m(z_m, s'_m, s_m)$ - is the probability function that an observation $z_m \in Z_m$ is observed given that the state transitioned to from $s_m \in S_m$ is $s'_m \in S_m$.

The risk model(s) is defined as a tuple: $M_x : \langle S_x, Z_x, O_x, S_{m'}, V, H \rangle$ where:

- S_x - is the set of discrete states representing the safety aspects i.e. hazardous states in system's process.
- Z_x - is the set of discrete observations that can be observed that influence the belief of the hazard.
- $O_x(z_x, s'_x, s_x)$ - is the probability function that an observation $z_x \in Z_x$ is observed given that the state transitioned to from $s_x \in S_x$ is $s'_x \in S_x$.
- V - is a set of hazard severities.
- $S_{m'}$ - is a set of such that $S_{m'} \subseteq S_m$, i.e. it is subset of the mode state set where the risk model is active.
- $H : S_x \rightarrow V$ - is function of state to hazard severity.

To complete the definition, a group of functions are needed to transform the process model to the mode and risk models. We call this mapping tuple ϕ_m and define it as $\langle \alpha_m, \beta_m, \gamma_m \rangle$, where:

- $\alpha_m : S \rightarrow S_m$ is a surjective (many-to-one) function from the process state space to the mode state space.
- $\beta_m : A \times Z \rightarrow Z_m$ is a surjective (many-to-one) function from the action and observation state space to the observations required to detect the mode.
- $\gamma_m : T \times O \rightarrow O_m$ which is a function that transforms the conditional probability tables (CPTs) of the transitions and observations in the process model to a CPT of the observations in the mode model.

Likewise, for each risk model M_x, there is a mapping tuple ϕ_x which defines the mapping of the process model to the risk model. This tuple is $\langle \alpha_x, \beta_x, \gamma_x \rangle$, where α_x, β_x, and γ_x are defined similarly to α_m, β_m, and γ_m respectively.

C. Adequacy Properties of Models

We define an adequate model in a safe decision-making context to be one that leads to decision-making behaviour that has an acceptable level of safety risk. To aid discussion, we use the concept of *ground truth*, which is the true or real process which we are attempting to model. This is used in the context of comparing our models, which are approximations, with it.

Using this definition, the models must have these high level properties to be adequate:

1) The mode model is acceptably accurate relative to the ground truth: it has a significant impact on safety because it activates and deactivates risk models.
2) The risk model calculates an exact or pessimistic risk values (relative to the ground truth) for the system to be safe.
3) The mode model and risk model is consistent with the process model.

As a result of these properties, (3) implies that the process model maintains a degree of independence from the safety-related properties of the system and can be used solely for the purpose of optimising performance-related system behaviour.

These properties give a 'separation of concerns' with regards to verification and validation of models separating out what models need to be consistent with the ground truth. As, by definition, the ground truth relates to the reality of some domain-specific problem, the adequacy of the mode and risk models involves showing that they are accurate, well tested, validated using known theory of the domain and that the risk of the runtime plan execution to detect the mode

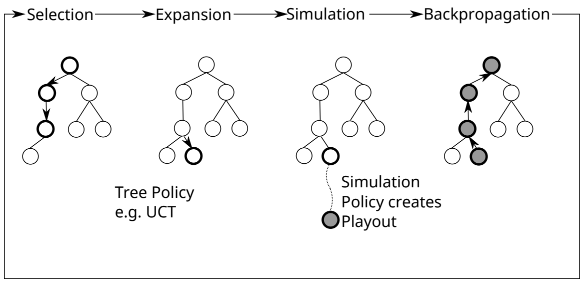

Figure 1: Standard Monte-Carlo Tree Search Stages (based on diagrams in [5] and [6])

wrongly is acceptably low. We do not go into verification and validation further in this paper – this is an avenue for future work.

Firstly, we consider the mode model. The mode model is constructed using a transformation mapping ϕ_m. α_m transforms the state space from the process model to the mode model. This effectively partitions the process model state space into a number of modes. For this to be consistent, the transitions between states in the mode model must be consistent with the transitions between the corresponding subsets of the process model state space: $\forall s, s' \in S, a \in A, z \in Z, O_m(\alpha_m(s), \beta_m(a, z)) = T(s', s, a) \times O(z, s', a)$.

Secondly, we consider the risk model. The risk model must be consistent with the process model when the risk model is 'active' according to the mode model. Recall that for a given risk model M_x, it is active when there is a state $s \in S_{m'}$ that is also $s \in S_m$ in the process model M_p. Hence, for an active risk model M_x, $\forall s, s' \in S, a \in A, z \in Z, O_x(\alpha_x(s), \beta_x(a, z)) = T(s', s, a) \times O(z, s', a) \wedge s'' \in S_{m'} \wedge s'' \in S_m$, provided the process model's transition function is consistent with the mode model.

D. Algorithm

The Monte-Carlo Tree Search is a search technique that builds a search tree of results by making random decisions and evaluating their outcomes through stochastic simulations [5]. Here, we consider this algorithm in terms of a planning technique. A set of four typical stages (Fig. 1) is run each search iteration to build this search tree incrementally [6]: Selection, Expansion, Simulation and Backpropagation. After many simulations, the set of best policies (i.e. the pareto-front over all policies searched) can be inferred from this search tree [6] that can be used for execution of the plan at runtime.

We propose an adaptation of the Multi-objective Monte Carlo Tree Search algorithm [2] for safe decision-making in partially observable domains called Multi-Objective Monte-Carlo Tree Search for Safety Critical Applications (MOMCTS-SC). Our algorithm is specifically adapted to use the POMDP-ACT representation. We define two types of node rather than one: an action node and an observation node because the state is no longer fully observable after each action and depends on the observation following the action. Like Silver and Veness [7], we maintain beliefs as a particle filter (a sampled representation of the belief) but, as opposed to storing with each node, we store the particle filter with just each observation node.

The general operation of this algorithm is described in this section and, for reference, the pseudo code of MOMCTS-SC is shown in Algorithm 1, 2 and 3. The notation used is defined as follows:

- N (node), N_a (action node), N_o (observation node).
- n_X (visit count of a node X), B_X (belief of node X).
- M_m (mode model), M_x (risk model).
- Heuristic $PW(n)$ is used to widen the search tree [8].
- r is a reward vector (i.e. costs in our case).

To simplify discussion, we consider the behaviour of the algorithm from two conditions: firstly, we consider the behaviour without the mode and risk models, and secondly, we consider the behaviour when the mode and risk models actively constrain the actions that are admissible for planning. Finally, we provide a brief description on how the safety objective is optimised.

In the first condition, the behaviour of the algorithm is much the same as Wang's MOMCTS algorithm[8]. It chooses between whether to apply exploration and exploitation in the expansion stage of the algorithm by using the hypervolume-based UCT heuristic (HV-UCT). Exploitation within the algorithm involves walking down the search tree to *exploit* existing nodes that have been previously visited if they seem promising. Exploitation involves *exploring* less visited nodes by performing new playout (a Monte-Carlo run) simulations to expand the search tree. The Algorithm 2 illustrates this. The chief difference from the original algorithm in this condition is that the algorithm is POMDP-ACT aware: it terminates on termination actions A_T and can handle partially observable states by keeping track of observations and the associated beliefs.

Under this condition, low probability events may never be sampled. Take, for example, events regarding failures of components: the IEC 61580 standard [9] defines Safety Integrity Levels which define the probabilities that a failure can occur for a component in continuous operation to be in the range of 10^{-5} and $10^{-9}h^{-1}$. For "failures" in decisions made hourly, this would only be optimistically sampled once every 100,000 and 1,000,000,000 samples respectively (only if every event can be accounted for and tracked). Having a satisfactory magnitude of samples to be confident that the algorithm has found any of these low probability events would not be possible with current hardware. Instead, the behaviour of the algorithm under the second condition tackles this problem.

In the second condition, under the mode and risk models, the algorithm also updates the admissible actions according to these dynamically updated risk models. When the algorithm is in the expansion stage, an admissible action

```
Input: root node $N$, sample size $L$
Output: pareto front $P$
for $i \leftarrow 1$ to $L$ do
    Sample state $s$ from $B_N$
    $r, P \leftarrow$ TREEWALK$(N, \emptyset, s)$
end
return $P$
```

Algorithm 1: MOMCTS-SC

is chosen. The admissible actions are actions that can be performed from the current playout's state because they are valid for the process model (e.g. physically possible) and constrained by the active risk model(s) (i.e. constrained otherwise it is too dangerous). Before the simulation of the playout occurs, the least visited action is chosen. The mode and active risk models are updated with this choice of action and this is used to validate whether the action is indeed admissible (and removing it from the set of admissible actions if necessary). If it is admissible, the simulation playout proceeds. Otherwise this process of checking least visited actions, removing if inadmissible, and updating the risk models will proceed recursively.

The update of the mode model also activates or deactivates new risk models based on whether they are appropriate for the new belief in the mode (i.e. whether the model should be activated for the belief of the system). Thus, for low probability events such as failures, activated risk models can give an evaluation of risk without invoking many simulation playouts as would the first condition.

Finally, optimising a safety-related objective is achieved by using the risk models to provide a risk value that is passed as one of the multiple objectives. The algorithm will then proceed to plan using this additional risk objective and provide solutions that do optimise for this objective.

E. Updating Mode and Risk Models

To prevent recalculation of the beliefs at each traversal of the search tree, a particle filter of each belief can be maintained in each observation node. Thus, the beliefs in the search tree only need to be updated when a new node is created – and this can be done using a *single-step belief update* [10].

When the current belief is updated, both the mode and risk model may be updated. The mode model is updated first as it may activate and deactivate risk models depending whether they are still relevant to the current belief. The mode model is updated before the risk models because the mode belief state may change when the action and observation pair update the current belief. Hence, if the state of the system moves into a new mode, other risk models may be activated whereas others would be deactivated as they are no longer relevant.

However, this leads to the question: what is the initial belief of the risk model. Several approaches may be taken:

```
Input: node $N$, pareto front $P$
Output: reward vector $r$, pareto front $P$
// if $N$ has children and no PW needed
if CHILDREN$(N) \neq \emptyset \wedge \neg PW(n_N)$ then
    // Selection stage
    $c^* \leftarrow \arg\max_c \{\forall c \in$ CHILDREN$(N)$, HV-UCT$(r_c, P)\}$;
    $r, P \leftarrow$ TREEWALK$(N, P, c)$;
else
    // Expansion and Simulation stage
    Recursively try and select the least visited action $a$ from $A_N$ which is admissible subject to the models $M_p$, $M_m$ and $M_x$ whilst updating $M_m$ and $M_x$ before each of these selections.
    $s', o, r, P \leftarrow$ SIMULATE$(N, P, a)$;
    if $\neg\exists N_a, N_a \in$ CHILDREN$(N)$ then
        Create new node $N_a$ as child of $N$;
        CHILDREN$(N) \leftarrow$ CHILDREN$(N) \cup N_a$;
    end
    if $\neg\exists N_o, N_o \in$ CHILDREN$(N_a)$ then
        Create new node $N_o$ as child of $N_a$;
        Update belief $B_{N_o}$ with $s'$;
        CHILDREN$(N_a) \leftarrow$ CHILDREN$(N_a) \cup N_o$;
        Store updated $M_m$ and $M_x$ in $N_o$;
    end
    Increment visit count $n_{N_a}$ and $n_{N_o}$;
    // Backpropagate rewards
    Update $r_{N_o}$ with reward $r$ and risk model(s) $M_x$;
end
return $P$
```

Algorithm 2: TREEWALK

```
Input: node $T$, pareto front $P$, action $a$
Output: first state $s$, first obs. $o$, total reward vector $r$,
        pareto front $P$
$trace \leftarrow$ run simulation with action $a$;
$s, o, r \leftarrow$ first step of $trace$;
// update pareto front
if $\forall r_p \in P, r \succeq r_p$ then
    $P \leftarrow (P \setminus \{\forall r_p \in P, r \succ r_p\}) \cup \{r\}$;
end
return $s, o, r, P$
```

Algorithm 3: SIMULATE

1) the initial state is predefined by the mode change,
2) the previous history of observations before the risk model is used to initialise the state, and
3) the risk model begins uninitialised with a special 'uncertainty' state linked with an 'uncertainty' objective which drives the tree search to update sensor values.

The first two approaches are fairly trivial to implement and are domain specific, whereas the latter is more general but incurs the penalty of another search objective. We opted for the first approach in our example.

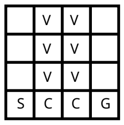

Figure 2: Cliff World

IV. Example

We have applied our algorithm to a variant of "Cliff World" [11][12]. The purpose of this example is to demonstrate the behaviour of the algorithm under normal circumstances and a circumstance that is particularly concerning with respect to safety critical planning: low probability events with catastrophic consequences. As mentioned in section III, existing Monte-Carlo planning algorithms cannot cope with these types of events due to sampling, whereas exact algorithms, such as dynamic programming, lose information about the risk when it is encoded into the reward value. In the first scenario, we consider "usual" behaviour where the bad outcomes can be easily determined. The second and third scenarios consider low probability events that have catastropic consequences. In latter scenario, it is shown that the risk model can be used to avoid this catastrophic consequence – the risk model approach naturally also handle the "usual" behaviour and is not included due to space constraints. We have made the algorithm and example available at: https://github.com/raedwulf/libysp.

A. Cliff World Definition

Consider a simple agent (robot) that travels from a starting square to a goal square in a two dimensional X by Y grid which has a slope towards the cliff edge. The journey has an element of risk such that the agent may slip off the edge of the cliff squares in grid if it is too close. This slipping occurs under some probability when the agent attempts to leave the square – i.e. in this example, if the robot travels right and slips at the same time, it is a diagonal movement that only falls if the target square is a cliff square. The goal is to minimise the cost of travelling to the goal and reduce the risk of falling off the edge of the cliff.

To simplify the discussion, the agent cannot leave the X by Y grid; actions that would take the agent outside the grid are inadmissable. The intelligent agent has five actions: *up*, *down*, *left*, *right* and *terminate*. It has observations which indicate whether the actions successfully occurred, whether the robot slipped, whether it falls of a cliff (die) or reaches a goal. (right & slip) is an observation that the robot successfully moved right and slipped down one square. A four by four Cliff World is illustrated in the Fig. 2 where **S** is the starting square, **G** is the goal square, **V** are the squares where the robot can slip down one square and **C** are cliff squares.

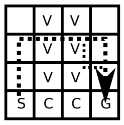

Figure 3: A Cliff World Policy that reaches goal safely

B. With highly probable slip and no risk model

We demonstrate behaviour of the algorithm without the need for a risk model; the probability of slipping is high (0.5) such that the MOMCTS sampling will be able to plan for the contingency of slipping towards the goal. This was run with the number of Monte-Carlo samples set to 10^5.

Here the algorithm makes a good decision. Fig. 3 (graphical interpretation) and Fig. 4 (policy tree) show one of the policies from the solution set which optimised both objectives.

C. With low probability slip and no risk model

We now consider the problem where the risk of falling that is acceptable is 10^{-8}. The Cliff World structure remains the same but the probability of slipping is changed to 10^{-6}.

In this scenario, the MOMCTS-SC without a risk model would produce an optimistic policy as it unlikely to encounter a simulation that shows the catastrophic falling off a cliff. Since the number of samples set was to 10^5, the probability of simulating a single catastrophic event is 0.1 in the best case if all simulations are (and can be) tracked, or, 5×10^-6 in the worst case if the simulations were not tracked at all. In either case, expecting an event to be simulated at these low probabilities is not feasible.

The action sequence in the produced policy that reached the goal was: $up \rightarrow right \rightarrow right \rightarrow right \rightarrow down$. This ends up being the shortest path to the goal using squares on the cliff's edge – not very safe if the risk of falling with 10^{-6} is not acceptable.

D. With low probability slip and risk model

Now, we use a risk model M_x to constrain the search algorithm to solve the problem in Section IV-C.

The risk model would need to behave such that it constrains the actions of the POMDP planner by calculating the probability that an action enters a hazardous state. In this example, it is easy to see that since the only approach into the cliff are the squares above the cliff square, the probability of slipping into the cliff will only exceed the acceptable threshold if these squares above the cliff are entered.

To create a HMM-based model we define two approaches: one with additional information-sharing through adding observations, and one without.

In the case without additional information, a HMM-encoding of the risk model would match the process model where the action and observation pairs of the process model are combined to be observations in the risk model. This follows from the representation defined in Section III-B.

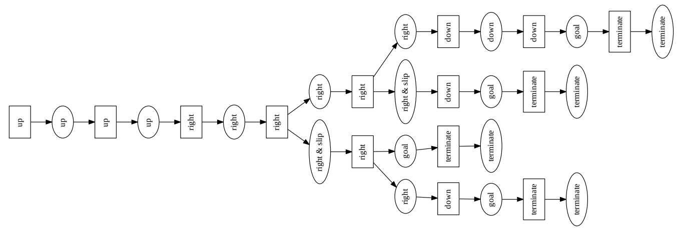

Figure 4: Policy generated by MOMCTS-SC with contingencies to avoid the cliff.

```
Input: policy set $P = \{p_0, p_1, \ldots, p_N\}$
Output: policy $p_o$
Sort $P$ according to lowest risk first;
for $i \leftarrow 1$ to $N$ do
    $risk_{diff} \leftarrow$ risk difference between $p_{i-1}$ and $p_i$;
    $reward_{diff} \leftarrow$ reward diff. between $p_{i-1}$ and $p_i$;
    if $g(risk_{diff}, reward_{diff})$ then
        return $p_{i-1}$
    end
end
return $p_N$
```
Algorithm 4: POLICY-PREFERENCE

In the case of adding additional observations to the problem definition (i.e. add sensors, which may or may not be physical), a much simpler risk model can be created. For example, we let the risk model 'see' the cliff edge by maintaining the value of the manhattan distance to the cliff. This would allow the risk model to define a state space based on the distance to the cliff edge: $S_x = \{$*normal*,*nearcliffedge*,*cliffedge*$\}$. The transition functions based on this additional observation would give the risk model the equivalent power as our previous one.

The policy generated by the algorithm using either risk model is $up \rightarrow up \rightarrow right \rightarrow right \rightarrow right \rightarrow down \rightarrow down$. Thus, the algorithm generates a plan that has a 'safety margin' of one square from the edge of the cliff as dictated by the risk model.

V. Preference Selection

Preference selection of objectives *a priori*, i.e. before the planning process, typically involves some mathematical combination of the components in the vector of objectives to produce a scalar value, for example, a linear weighted sum of objective values. *A posteori* preference selection involves selecting the preferences after the planning process has taken place and yielded a set of viable policies.

As our planning algorithm gives an output a set of safe policies, it enables *a posteori* preference selection. For safe decision-making, a preference selection mechanism is required that selects the policies that do not seek reward at the expense of excessive risk. Seeking reward with some acceptable amount of risk is fine, as long as it is limited and not minimised or maximised unnecessarily.

We give one possible two-objective preference selection mechanism based on As Low As Reasonably Practicable (ALARP) principle. This is shown in Algorithm 4. The ALARP principle is often used to make safety-related decisions in the UK and utilises the concept of *grossly disproportionate* to refer to a factor (function g in Algorithm 4) whereby the cost of risk reduction is far greater than the benefit gained. Such a factor is normally expressed as a limit of the cost per life saved; for example, it may be considered disproportionate to spend more than £5M per life saved.

VI. Further Work

There are two main directions for further work.

Firstly, our algorithm is only part of a planner; other concerns not addressed here include real-time concerns – it is difficult for pure Monte-Carlo Tree Search algorithms to guarantee when it has found a safe policy as this is highly dependent on the search space. This may require deterministic fallbacks and further work must consider how this can be integrated with our solution.

Secondly, our work lays some foundations for separate process and risk models. However, a method to construct these risk models such that they are both valid for the domain and sufficient to represent the risk for a particular decision-making problem is still an open avenue for research.

VII. Related Work

The planner we introduced in this paper operates using a searching algorithm to generate the policy. It builds upon seminal work in POMDP-solving by Kaelbling, Cassandra and Littman [10], concepts of state factorisation [13], goal-oriented (and state-based termination) POMDPs [14][4] and modern Monte-Carlo Tree Search-based planners [7][8]. This latter choice in technology is important as it avoids the explosion of computational complexity when using dynamic programming [10] as opposed to MCTS-based planners and is able to plan with respect to multiple objectives [8].

We are not alone in considering that single objective utilities are not suitable for some applications [15]; a survey by Roijers et.al. [1] provides a thorough treatment on classifying the underlying problems of these applications. The majority of existing POMDP algorithms rely on dynamic programming using the Bellman equation [10] which implicitly assumes that the objectives are represented by scalar values that can be combined linearly. Fortunately, Wang and Sebag [2] have overcome this difficulty by utilising a measure called the hypervolume indicator [16]: they developed a new multi-objective planning algorithm that uses the Monte-Carlo Tree Search to optimise the sets of the best policies. This provides the ability for a planner based on this multi-objective search to select policies based on their relative quality after the search has been performed. For our work, this is crucial for an *a posteori* preference selection: in an autonomous system context, this can be used to select policies based on the ALARP principle (Section V), and, in a decision-support context, the operator can make his decision based on a number of different choices.

In systems safety literature, the direct implications of algorithms on safety have been less explored. Work by Seward et al. [17] uses a POMDP-based approach for autonomous vehicle navigation: we take inspiration from their approach in representing hazardous areas as states but they do not give a firm theoretical basis for their reward value which gives a high penalty for entering hazardous states to prevent policies from making decisions that may lead to hazards. We explicitly use multiple objectives including one of safety risk instead. Later work by Kolobov [14] give a more theoretical treatment of a similar concept called "dead end states" that is analogous of states representing accidents in safety-critical system processes. His work introduces a number of different MDP formulations and solving algorithms that help avoid "dead end states" through problem transformation which effectively constrains the actions available to the decision maker to reach "dead end states". As his work was based on problems defined with fully observable state spaces [14], there is no easy adaptation of his work for partially observable state spaces and assigning a range of severities that interpolate between 'no safety effect' to 'catastrophic' accident. However, our work does take inspiration from concepts of constraining actions, albeit during the planning process rather than transforming the problem beforehand.

VIII. Conclusion

In this paper, we have captured the non-trivial nature of ensuring that a planner is able to make safe decisions. We have made the first steps to bridge the gap between an autonomous planner that just simply optimises some criteria to one that is able to make safe decisions using a complex trade-off between multiple-objectives.

Specifically, we have promoted the concept of risk as a first-class concept in our algorithm. It is treated 'above' other objectives in the sense that it acts as both an optimisation criteria as well as a hard constraint on what counts as an "acceptably safe policy". In doing so, we also recognise that risk is not something that is to be forgotten at the end of the planning stage. We provide an example ALARP-inspired preference selection algorithm which chooses a policy amongst the other acceptably safe policies that will be the one that will be executed on the system.

Furthermore, we provide a method to define adequate models for the purpose of our planning algorithm. These models are structured to be both manageable from a safety point of view, and, updated efficiently by the planning algorithm without losing accuracy that may impact safety.

Acknowledgment

The authors would like to thank Agent Oriented Software Limited and Statoil ASA for support in this research.

References

[1] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[2] W. Wang, M. Sebag *et al.*, "Multi-objective monte-carlo tree search," in *Asian conference on machine learning*, 2012.

[3] Health and S. Executive, *Reducing Risks, Protecting People: HSE's Decision-Making Process*. HSE, 2001.

[4] E. A. Hansen, "Indefinite-horizon pomdps with action-based termination," in *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, ser. AAAI'07, vol. 2. AAAI Press, 2007, pp. 1237–1242.

[5] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 4, no. 1, pp. 1–43, 2012.

[6] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, "Monte-Carlo Tree Search: A New Framework for Game AI," in *AIIDEC-08*, 2008.

[7] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems*, 2010, pp. 2164–2172.

[8] W. Wang, "Multi-objective sequential decision making," Ph.D. dissertation, Universit Paris-Sud, 2014.

[9] International Electrotechnical Commission (IEC), *Functional safety of electrical/electronic/programmable electronic safety related systems (IEC 61508)*. IEC, 2000.

[10] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1, pp. 99–134, 1998.

[11] V. B. Zubek and T. Dietterich, "A pomdp approximation algorithm that anticipates the need to observe," in *Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence*, ser. PRICAI'00. Berlin, Heidelberg: Springer-Verlag, 2000, pp. 521–532.

[12] H. Ren, A. A. Bitaghsir, and M. Barley, "Safe stochastic planning: Planning to avoid fatal states," in *Safety and Security in Multiagent Systems*. Springer, 2009, pp. 101–115.

[13] E. A. Hansen and Z. Feng, "Dynamic programming for pomdps using a factored state representation," in *AIPS*, 2000, pp. 130–139.

[14] A. Kolobov, "Scalable methods and expressive models for planning under uncertainty," Ph.D. dissertation, University of Washington, 2013.

[15] D. Bryce, W. Cushing, and S. Kambhampati, "Probabilistic planning is multi-objective," Arizona State University, Tech. Rep. ASU-CSE-07-006, 2007.

[16] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Parallel Problem Solving from Nature-PPSN VIII*. Springer, 2004, pp. 832–842.

[17] D. Seward, C. Pace, and R. Agate, "Safe and effective navigation of autonomous robots in hazardous environments," *Autonomous Robots*, vol. 22, no. 3, pp. 223–242, 2007.