Planning with Formal Reachability Guarantees in Goal-Oriented MDPs with Dead-Ends

Matisse Roche[1,2,3] **Caroline P.C. Chanel**[1,3] **Yoko Watanabe**[1,2]

[1]Fédération ONERA ISAE-SUPAERO ENAC, Université de Toulouse, Toulouse, France
[2]ONERA, Toulouse, France, {matisse.roche,yoko.watanabe}@onera.fr
[3]ISAE-SUPAERO, Toulouse, France, caroline.chanel@isae-supaero.fr

Abstract

Goal-oriented Markov Decision Processes with unavoidable dead-ends pose a significant challenge to standard planning algorithms, as the goal cannot always be reached with probability 1. In certain applications, it can be desirable to optimize efficiency only over successful trajectories while enforcing a lower bound on goal reachability, yet no existing method directly addresses this conditional formulation, whose non-linear structure makes direct optimization difficult. To overcome this, we introduce a tractable approximation. Working within an unconstrained utility-maximization framework, we derive an analytical formula that determines a value for the goal-reward parameter sufficient to guarantee that the optimal policy respects the requested reachability threshold. The proposed approach is evaluated on two common benchmark domains, Aircraft Routing and Exploding Blocksworld, and compared against state-of-the-art constrained optimization methods, specifically i-dual and Chance-Constrained. Results show that our method offers a compelling trade-off: it significantly reduces the expected cost compared to conservative lexicographic approaches (i.e., i-dual), while avoiding the failure-seeking bias inherent to methods introducing an artificial zero-cost give-up action (i.e., Chance-Constrained).

1 INTRODUCTION

Goal-oriented Markov Decision Processes (Goal-Oriented MDPs) serve as the fundamental framework for sequential decision-making in stochastic domains. This formalism powers critical applications ranging from autonomous navigation to industrial logistics. Classically, these problems are modeled as Stochastic Shortest Path (SSP) problems [Bertsekas and Tsitsiklis, 1991]. Standard SSP algorithms rely on a foundational assumption: the existence of at least one proper policy—a policy guaranteed to reach the goal state with probability 1 from any starting configuration. However, this assumption is often unrealistic in practical applications [Kolobov et al., 2012]. Indeed, many environments feature dead-end states, from which the goal becomes strictly unreachable. Consider a fleet of firefighting drones dispatched to contain a wildfire: while each drone must navigate through turbulent winds and dynamic obstacles, a collision results in a catastrophic crash from which recovery is impossible. Similarly, in high-stakes manufacturing, if a robotic arm breaks a unique component, the production goal becomes unattainable. In such environments, dead-ends may be unavoidable: any policy can carry a non-zero probability of failure. Consequently, the standard SSP objective of minimizing the expected total cost becomes ill-posed, as the expected cost of any policy diverges to infinity.

Existing approaches handle this by modifying the MDP to render the expected cost finite. A common device is to introduce an artificial "give-up" action from dead-end states to the goal, effectively terminating failed trajectories. This makes the unconditional expected cost well-defined and amenable to standard optimization.

However, within this modified model, managing the safety-efficiency trade-off remains problematic. Two paradigms coexist. The first assigns a finite penalty cost to the give-up action and minimizes the resulting unconditional expected cost, either directly [Kolobov et al., 2012] or subject to a failure probability constraint as in Chance-Constrained SSPs [Hong et al., 2021]. But the choice of this penalty is inherently arbitrary, and as illustrated in Figure 1, it can create a structural failure-seeking bias: the solver exploits cheap failures to deflate the global expectation, even when the cost of successful trajectories is strictly higher. The safety constraint limits this effect but does not eliminate it, since the objective itself still rewards failure. The second paradigm, represented by S3P [Teichteil-Königsbuch, 2012] or i-dual with high penalties [Trevizan et al., 2017], implicitly as-

Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence (UAI 2026), PMLR 337:5731–5739.

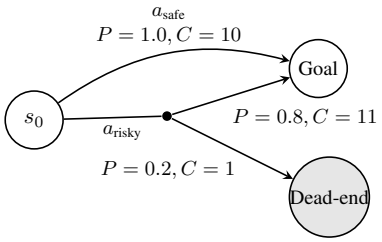

Figure 1: Counter-example. Policy a_{safe} is safe and efficient with goal-reaching conditional cost 10. However, minimizing the unconditional expected cost paradoxically prefers the unsafe, less efficient a_{risky} action. This is because the cheap failure outcome lowers its global expectation: $0.8 \times 11 + 0.2 \times 1 = 9 < 10$ that is less than its goal-reaching conditional cost 11.

signs an infinite cost to failure, thereby strictly prioritizing safety. While theoretically robust, this lexicographic ordering prevents any trade-off: as shown by Freire and Delgado [2017], such solvers often discard highly efficient policies for negligible safety gains.

A different perspective arises from noting that, in many safety-critical applications, the cost accumulated before a failure carries no operational value. If a firefighting drone crashes after 10 seconds of flight, those 10 seconds contribute nothing to extinguishing the fire: the operator only cares how fast successful drones arrive and how often they fail. This suggests treating safety and efficiency as separate concerns: enforcing a lower bound on the success probability, while measuring efficiency only over successful trajectories. This conditional formulation is moreover inherently well-posed, as every goal-reaching trajectory has finite cost. As a first attempt, instead of solving the constrained optimization problem, we operate within the Goal-Directed Utility-Based Semantic (GUBS) framework [Freire and Delgado, 2017], which evaluates policies through a scalar utility combining accumulated costs and a parametrizable bonus K_g awarded for reaching the goal.

In this context, our contribution is twofold. First, we establish a theoretical sufficiency condition: we prove that setting K_g above a specific, closed-form analytical bound guarantees that the optimal policy respects the user-specified safety level P_{th}. This formula leverages both structural bounds of the MDP (via the deterministic shortest path) and reference values derived from the risk-sensitive lexicographic policy. Second, we propose the S-GUBS (Safety-Guaranteed GUBS) algorithm to operationalize this result, returning a policy formally guaranteed to satisfy the threshold P_{th}.

We evaluate S-GUBS on two common stochastic benchmark domains with dead-ends: Aircraft Routing and Exploding Blocksworld. We compare it against state-of-the-art constrained optimization methods, specifically i-dual [Trevizan et al., 2017] and Chance-Constrained SSPs [Hong et al., 2021]. Results show that our method offers a compelling trade-off: it significantly reduces the expected cost compared to conservative lexicographic approaches (i.e. i-dual), while avoiding the unnecessary risks inherent to give-up modelling tricks with arbitrary penalty setting (i.e. Chance-Constrained).

2 BACKGROUND AND RELATED WORK

2.1 DEFINITIONS

Markov Decision Processes (MDPs) [Bertsekas and Tsitsiklis, 1991, Mausam and Kolobov, 2012] provide a fundamental framework for sequential decision-making. A goal-oriented MDP is defined as a tuple $\langle S, A, T, c, \mathcal{G} \rangle$ where S is a finite set of states, A is a finite set of actions, and $T : S \times A \times S \to [0, 1]$ is the transition function. The cost function $c : S \times A \times S \to \mathbb{R}$ assigns a strictly positive cost to every transition originating from non-goal states. $\mathcal{G} \subseteq S$ is the set of goal states, which are cost-free and absorbing ($T(g, a, g) = 1$ and $c(g, a, g) = 0$ for all $g \in \mathcal{G}, a \in A$).

A *policy* π is a decision rule mapping states to a probability distribution over actions. A *trajectory* $\theta = (s_0, a_0, s_1, a_1, \ldots)$ is a sequence of states and actions. We define the cumulative cost of a trajectory as $C(\theta) = \sum_{t=0}^{\infty} c(s_t, a_t, s_{t+1})$. We denote by $\Theta_{\mathcal{G}}^{\pi,s}$ the set of all trajectories generated by policy π starting from s that eventually reach a goal state $g \in \mathcal{G}$.

Writing $\mathbb{P}^{\pi,s}$ (resp. $\mathbb{E}^{\pi,s}$) for the probability measure (resp. expectation) over trajectories induced by executing π from s, the probability of success of a policy π from state s is:

$$P_{\mathcal{G}}^{\pi}(s) = \mathbb{P}^{\pi,s}(\theta \in \Theta_{\mathcal{G}}^{\pi,s}). \tag{1}$$

This allows us to formally define a *dead-end* state. A state $s \in S$ is a dead-end if the goal cannot be reached from s under any policy, i.e., $\max_{\pi} P_{\mathcal{G}}^{\pi}(s) = 0$.

Finally, the conditional expected cost is the expectation of $C(\theta)$ over the set of successful trajectories:

$$E_c^{\pi}(s) = \mathbb{E}^{\pi,s}[C(\theta) \mid \theta \in \Theta_{\mathcal{G}}^{\pi,s}]. \tag{2}$$

2.2 RELATED WORK

Safety-Maximizing Approaches. To handle unavoidable dead-ends without incurring infinite costs, a prominent family of approaches adopts a strict prioritization of goal reachability. **iSSPUDE** [Kolobov et al., 2012] assigns an infinite penalty to dead-end states, forcing the agent to avoid them whenever possible. In the standard context where action costs are strictly positive, the optimal policies yielded by iSSPUDE are equivalent to those of the **S3P** (Stochastic Safest and Shortest Path) framework [Teichteil-Königsbuch, 2012]. Using our notations, it seeks a policy π^* that lexicographically optimizes the probability of success $P_{\mathcal{G}}^{\pi}(s_0)$ first, and

the conditional expected cost $E_c^\pi(s_0)$ second. Formally:

$$\pi^* \in \arg\min_{\pi \in \Pi_{safe}} E_c^\pi(s_0), \text{ where } \Pi_{safe} = \arg\max_{\pi \in \Pi} P_{\mathcal{G}}^\pi(s_0). \tag{3}$$

MCMP. Min-Cost given Max-Prob (MCMP) [Trevizan and Teichteil-Königsbuch, 2017] proposes a variation of this lexicographic scheme. While it shares the same primary objective (searching within Π_{safe}), MCMP optimizes a different secondary criterion: it minimizes the expected cost over all trajectories, explicitly accounting for the costs incurred *before entering a dead-end*, whereas S3P typically focuses solely on successful trajectories. By sharing this rigid lexicographic prioritization, these approaches ensure maximum safety but frequently yield inefficient policies, as they are inherently unable to balance risk and cost.

Threshold-Based Approaches. Instead of absolute maximization, other approaches seek to satisfy a specific safety requirement defined by a user. The **AtLeastProb** framework [Steinmetz et al., 2016] focuses on the feasibility problem. It efficiently verifies whether there exists a policy π such that the probability of reaching the goal satisfies $P_{\mathcal{G}}^\pi(s_0) \geq P_{th}$. However, AtLeastProb terminates as soon as the threshold is met; it does not explicitly optimize a cost metric and provides no guidance on which policy to choose among the set of safe policies.

The **Chance-Constrained SSP (CC-SSP)** framework [Hong et al., 2021] explicitly allows users to define a set of undesirable states $\mathcal{S}_U$ and constrains the probability of entering them. Formally, the problem minimizes the expected cost over a finite horizon H:

$$\min_\pi \mathbb{E}^\pi \left[\sum_{t=0}^{H} c(s_t, a_t) \right] \quad \text{s.t.} \quad \mathbb{P}^\pi(\exists t \leq H, s_t \in \mathcal{S}_U) \leq \Delta. \tag{4}$$

Unlike approaches that isolate successful missions, standard CC-SSP formulations minimize the total expected cost over all trajectories. In this setting, when an agent enters an undesirable state, cost accumulation ceases. This framework allows users to specify arbitrary sets of undesirable states $\mathcal{S}_U$. In our study, we naturally designate dead-end states as undesirable. To solve this problem, Hong et al. [2021] propose an anytime algorithm based on heuristic search. This method quickly generates an initial, conservative solution that satisfies the safety constraint, and then iteratively improves the policy's efficiency over time.

Constrained SSPs (C-SSPs) and i-dual. The C-SSP framework [Altman, 1999] generalizes standard SSPs to multi-objective settings. It seeks a policy π that minimizes a primary expected cost C_0 subject to upper bounds $\hat{c}_j$ on K secondary expected costs $C_j, j \in \{1...K\}$. Formally, the optimization problem is:

$$\min_\pi \mathbb{E}^\pi[C_0(\theta)] \quad \text{s.t.} \quad \mathbb{E}^\pi[C_j(\theta)] \leq \hat{c}_j, \quad \forall j \in \{1...K\}.$$

This optimization problem is typically solved by transforming it into a Linear Program (LP) operating on the convex set of occupation measures (the expected number of times state-action pairs are visited). In contrast to standard SSPs, for which a deterministic optimal policy always exists, theoretical results establish that optimal solutions to this constrained formulation are generally stochastic [Chatterjee et al., 2006]. To efficiently solve the optimization problem above Trevizan et al. [2017] introduced **i-dual**, an algorithm that uses heuristic search within the dual space. i-dual avoids enumerating all states, offering significant computational speedups over standard LP solvers. However, to remain applicable in the presence of dead-ends, i-dual relies on a penalty-based modification: an artificial action is introduced to allow transitions from dead-ends to the goal at the cost of a high penalty. This trick thereby numerically ensures that the primary expected cost remains finite.

3 CONDITIONAL SSP PROBLEM

In this section, we formally introduce the Safety-constrained Conditional Stochastic Shortest Path problem, show that it generalizes existing frameworks, and demonstrate why standard constrained MDP algorithms cannot solve it directly.

3.1 THE CONDITIONAL OBJECTIVE

For a given probability threshold P_{th}, we define the Conditional Stochastic Shortest Path problem as:

$$\min_\pi \; E_c^\pi(s_0) \quad \text{subject to} \quad P_{\mathcal{G}}^\pi(s_0) \geq P_{th}. \tag{5}$$

This formulation isolates operational efficiency from failure scenarios and offers a flexible safety-efficiency trade-off controlled by P_{th}. Furthermore, it strictly generalizes two well-known frameworks:

- **Generalization of S3P:** Setting $P_{th} = P^*(s_0)$ (the maximum achievable success probability) recovers the secondary objective of the S3P framework [Teichteil-Königsbuch, 2012], which minimizes the conditional cost within the set of safest policies.
- **Generalization of SSP:** In environments without dead-ends, $P^*(s_0) = 1$. Setting $P_{th} = 1$ makes the conditional cost equal to the unconditional cost, reducing the problem to a standard SSP.

3.2 WHY STANDARD C-MDP ALGORITHMS FALL SHORT

At first glance, problem (5) might appear solvable within the Constrained MDP (C-MDP) framework [Altman, 1999]: one would minimize an expected cost subject to the linear constraint $P_{\mathcal{G}}^\pi(s_0) \geq P_{th}$. However, there are two fundamental obstacles.

The objective is non-linear. Standard C-MDP algorithms require the objective to be linear in the state-action occupation measures. In our formulation, the conditional expected cost can be rewritten as:

$$E_c^\pi(s_0) = \frac{\mathbb{E}^{\pi,s_0}[C(\theta)\,\mathbf{1}_{\theta\in\Theta_\mathcal{G}^{\pi,s_0}}]}{P_\mathcal{G}^\pi(s_0)}. \tag{6}$$

This fractional structure places the problem outside the scope of standard linear programming methods used to solve C-MDPs. To the best of our knowledge, no efficient algorithm exists in the literature to solve problem (5) exactly for goal-oriented MDPs with dead-ends.

Enforcing the constraint does not prevent the penalty bias. To work within the C-MDP framework, one must replace the conditional objective by an unconditional expected cost, which is linear. However, as discussed in the introduction, the unconditional cost is ill-defined when dead-ends are unavoidable. The standard workaround is to introduce an artificial action from dead-end states to the goal at a finite penalty cost D, yielding a valid C-MDP solvable by algorithms such as i-dual [Trevizan et al., 2017]. The safety constraint $P_G^\pi(s_0) \geq P_{th}$ remains useful in this reformulation: it guarantees that the returned policy achieves the requested success probability. However, it does not prevent the failure-seeking bias. If D is zero or small, the solver can exploit cheap failures to deflate the global expectation, even when the constraint is satisfied, as we demonstrate in the experiments (Section 6) and illustrate in Figure 1. A natural remedy is to increase D to penalize failures more heavily. But if D is too large, the penalty dominates the unconditional cost and the solver maximizes safety regardless of P_{th}, making the constraint redundant. In summary, the C-MDP formulation with a penalty can enforce the constraint, but cannot optimize the conditional cost: the trade-off nature remains governed by the arbitrary choice of D.

These observations motivate a different strategy. Rather than solving a constrained optimization problem that additionally requires an auxiliary penalty parameter to handle dead-ends – a parameter that implicitly influences the safety level without offering explicit control over it – we operate within the unconstrained utility-maximization framework of GUBS [Freire and Delgado, 2017]. In this setting, the safety-efficiency trade-off is governed by a single parameter K_g, like fSSPUDE [Kolobov et al., 2012]. Our first contribution is an analytical formula that calibrates K_g directly from the user-specified threshold P_{th}, providing a formal guarantee on the safety level without requiring any constrained optimization method.

4 THE S-GUBS APPROACH

4.1 UTILITY-BASED FORMULATION

Lexicographic planning methods inherently preclude trade-offs by strictly prioritizing safety over efficiency. To overcome this rigidity, the GUBS framework addresses the problem by evaluating a trajectory θ based on a unified utility function that accounts for both the cumulative cost and the final outcome (success or failure). Specifically, the utility of a trajectory is defined as:

$$U(\theta) = u(C(\theta)) + K_g \cdot \mathbf{1}_{\theta\in\Theta_\mathcal{G}}, \tag{7}$$

where:

- $u : \mathbb{R}^+ \cup \{\infty\} \to [U_{min}, U_{max}]$ is a strictly decreasing utility function over costs;
- $C(\theta)$ is the cumulative cost of trajectory θ;
- $K_g > 0$ is a scalar parameter representing the utility bonus awarded for reaching a goal state;
- $\mathbf{1}_{\theta\in\Theta_\mathcal{G}}$ is the indicator function for goal achievement.

A policy π is evaluated by its value function, defined as the expected utility over all possible trajectories starting from state s:

$$V_{GUBS}^\pi(s) = \mathbb{E}^{\pi,s}[U(\theta)]. \tag{8}$$

By distinguishing between successful and failed trajectories, this formulation can be decomposed as follows:

$$V_{GUBS}^\pi(s) = P_\mathcal{G}^\pi(s)\left(U_\mathcal{G}^\pi(s) + K_g\right) + (1 - P_\mathcal{G}^\pi(s))U_{min}, \tag{9}$$

where $P_\mathcal{G}^\pi(s)$ is the probability of reaching the goal, and $U_\mathcal{G}^\pi(s)$ represents the expected utility of costs conditional on successful trajectories. The term $U_{min} = \lim_{C\to+\infty} u(C)$ corresponds to the utility of trajectories that fail to reach the goal. Indeed, consistent with SSP assumptions, trajectories not reaching the goal are assumed to accumulate infinite cost, thus yielding the minimum asymptotic utility.

A consequence of the utility depending on the final outcome is that the optimal policy is generally non-stationary, operating in an augmented state space $S \times \mathbb{R}^+$ with cost-to-come. However, when employing the exponential utility $u(C) = e^{\lambda C}$ (with $\lambda < 0$, implying $U_{min} = 0, U_{max} = 1$), Crispino et al. [2023] demonstrated that the problem becomes tractable. They proved the existence of a computable cost threshold C_{max} such that for any accumulated cost $c \geq C_{max}$, the optimal policy becomes stationary. Furthermore, this tail policy corresponds exactly to the optimal *risk-sensitive lexicographic policy*, denoted π_L^*.

Formally, π_L^* optimizes a preference relation $\succ$ defined as $\pi \succ \pi'$ if:

- $P_\mathcal{G}^\pi(s) > P_\mathcal{G}^{\pi'}(s)$, or
- $P_\mathcal{G}^\pi(s) = P_\mathcal{G}^{\pi'}(s)$ and $U_G^\pi(s) > U_G^{\pi'}(s)$.

This policy π_L^* plays a crucial role in our framework, serving as the reference for the maximum achievable reachability.

4.2 FORMAL REACHABILITY GUARANTEES

While GUBS provides the framework to model the trade-off, it does not allow users to enforce a specific safety constraint. The relationship between the bonus K_g and the resulting success probability of the optimal policy is implicit. Typically, using GUBS to satisfy a constraint P_{th} would require an iterative trial-and-error process to find the correct K_g. We resolve this ambiguity by deriving a closed-form analytical lower bound for K_g, naming this S-GUBS (Safety-Guaranteed GUBS).

Our approach relies on the topological structure of the MDP to bound the maximum possible efficiency of any policy. We define $d_{\mathcal{G}}(s_0)$ as the cost of the shortest path from the initial state s_0 to the goal set $\mathcal{G}$ in the deterministic graph underlying the MDP. Since the utility function u is strictly decreasing, $u(d_{\mathcal{G}}(s_0))$ represents a strict upper bound on the conditional utility of any trajectory.

Theorem 1 (S-GUBS Sufficient Condition)**.** *Let $\mathcal{M}$ be a goal-oriented MDP and $P^*(s_0)$ be the maximum probability of reaching $\mathcal{G}$ from s_0. Let π^*_L be the optimal risk-sensitive lexicographic policy. For any safety threshold $P_{th} \in [0, P^*(s_0))$, we define the critical utility bonus $K^+_{P_{th}}$ as:*

$$K^+_{P_{th}} = \frac{P_{th}[u(d_{\mathcal{G}}(s_0)) - U_{min}] - P^*(s_0)[U^{\pi^*_L}_{\mathcal{G}}(s_0) - U_{min}]}{P^*(s_0) - P_{th}}, \tag{10}$$

where $U^{\pi^*_L}_{\mathcal{G}}(s_0)$ is the expected conditional utility of π^*_L.

If $K_g \geq K^+_{P_{th}}$, then any optimal policy π^* with respect to the GUBS criterion satisfies $P^{\pi^*}_{\mathcal{G}}(s_0) \geq P_{th}$.

Proof. Let π^- be any policy that violates the reachability constraint, i.e., $P^{\pi^-}_{\mathcal{G}}(s_0) < P_{th}$. We aim to show that if $K_g \geq K^+_{P_{th}}$, then the expected utility of the lexicographic policy π^*_L is strictly greater than that of π^-.

First, we establish an upper bound on the expected utility $V^{\pi^-}(s_0)$. Recall that:

$$V^{\pi^-}(s_0) = P^{\pi^-}_{\mathcal{G}}(s_0)[U^{\pi^-}_{\mathcal{G}}(s_0) + K_g - U_{min}] + U_{min}.$$

Since u is strictly decreasing with cost, for any policy, the conditional utility is bounded by the utility of the shortest path: $U^{\pi^-}_{G}(s_0) \leq u(d_{\mathcal{G}}(s_0))$. Furthermore, since $P^{\pi^-}_{\mathcal{G}}(s_0) < P_{th}$, we have the strict inequality:

$$V^{\pi^-}(s_0) < P_{th}[u(d_{\mathcal{G}}(s_0)) + K_g - U_{min}] + U_{min}. \tag{11}$$

Now, assume for the sake of contraposition that the unsafe policy π^- is at least as good as the lexicographic policy π^*_L, i.e., $V^{\pi^*_L}(s_0) \leq V^{\pi^-}(s_0)$. Substituting the value of π^*_L and using the bound from (11), we get:

$$\begin{aligned} P^*(s_0)[U^{\pi^*_L}_{\mathcal{G}}(s_0) + K_g - U_{min}] + U_{min} \leq V^{\pi^-}(s_0) \\ < P_{th}[u(d_{\mathcal{G}}(s_0)) + K_g - U_{min}] + U_{min}. \end{aligned}$$

Simplifying by U_{min} and rearranging the terms to isolate K_g, we have:

$$\begin{aligned} K_g[P^*(s_0) - P_{th}] < P_{th}[u(d_{\mathcal{G}}(s_0)) - U_{min}] \\ - P^*(s_0)[U^{\pi^*_L}_{\mathcal{G}}(s_0) - U_{min}]. \end{aligned}$$

Since $P^*(s_0) > P_{th}$, we can divide by $[P^*(s_0) - P_{th}]$ without reversing the inequality:

$$K_g < K^+_{P_{th}}.$$

We have shown that $V^{\pi^*_L}(s_0) \leq V^{\pi^-}(s_0) \implies K_g < K^+_{P_{th}}$. By contraposition, if $K_g \geq K^+_{P_{th}}$, then $V^{\pi^*_L}(s_0) > V^{\pi^-}(s_0)$.

Finally, since any optimal policy π^* satisfies $V^{\pi^*}(s_0) \geq V^{\pi^*_L}(s_0)$, we obtain $V^{\pi^*}(s_0) > V^{\pi^-}(s_0)$. Thus, π^* strictly dominates any policy violating the safety constraint, ensuring $P^{\pi^*}_{\mathcal{G}}(s_0) \geq P_{th}$. □

4.3 S-GUBS POLICY

For a given safety threshold P_{th}, an S-GUBS policy $\pi^*_{S-GUBS}(P_{th})$ is defined as the optimal policy under GUBS when using parameter $K_g = K^+_{P_{th}}$, where $K^+_{P_{th}}$ is calculated using Eq. (10).

5 RESOLUTION METHOD

To go beyond the presented theoretical results, we aim to produce S-GUBS policies. To that end, we propose a practical algorithm that combines existing solvers with our parameter calculation. In particular, our approach relies on two specific subroutines derived from the GUBS literature [Crispino et al., 2023] and standard graph theory. This section details the steps required to compute the S-GUBS policy.

Risk-Sensitive Evaluation. To compute the formula for $K^+_{P_{th}}$ (Eq. (10)), we first require the characteristics of the safest policy: the maximum probability $P^*(s_0)$ and its associated conditional utility $U^{\pi^*_L}_{\mathcal{G}}(s_0)$. We utilize the RISKSENSITIVE-VI algorithm [Crispino et al., 2023]. This variation of Value Iteration (VI) performs a lexicographic optimization: it first computes the maximum reachability probabilities, and then optimizes the expected utility strictly within the set of probability-maximizing actions.

Topological Upper Bound. The final component required for our analytical formula is the utility of the "most efficient" trajectory, which we approximate using the shortest path in the underlying graph. Specifically, let $G_{det} = (S, E_{det}, w)$ be the deterministic weighted graph associated with the MDP, where an edge $(s, s') \in E_{det}$ exists if there is an action a such that $T(s, a, s') > 0$. The weight is defined as the minimum cost to transition between these states:

$$w(s, s') = \min\{c(s, a, s') \mid a \in A, T(s, a, s') > 0\}.$$

Algorithm 1: S-GUBS (Safety-Guaranteed GUBS)

Input: MDP $\mathcal{M} = \langle S, A, T, c, \mathcal{G} \rangle$, Safety Threshold P_{th}, Utility function $u(\cdot)$

Output: Optimal Policy π^* satisfying $P_{\mathcal{G}}^{\pi^*}(s_0) \geq P_{th}$

```
/* Topological analysis */
```
1. Construct the deterministic graph G_{det} from $\mathcal{M}$
2. Compute $d_{\mathcal{G}}(s_0) \leftarrow$ DIJKSTRA$(G_{det}, s_0, \mathcal{G})$

```
/* Max-Prob analysis */
```
3. **for all** $s \in S$ **do**
4. Compute $P^*(s)$ and $U_{\mathcal{G}}^{\pi_L^*}(s)$ with RISKSENSITIVE-VI
5. **end for**

```
/* Feasibility check */
```
6. **if** $P^*(s_0) < P_{th}$ **then**
7. **return** "Target safety threshold cannot be met."
8. **end if**

```
/* K_g parameter calibration */
```
9. Compute the critical utility bonus $K_{P_{th}}^{+}$ using Eq. (10)
10. $K_g \leftarrow K_{P_{th}}^{+}$

```
/* Policy optimization: */
```
11. $\pi^* \leftarrow$ EGUBS-VI$(\mathcal{M}, u, K_g)$
12. **return** π^*

We compute $d_{\mathcal{G}}(s_0)$, the cost of the shortest path from s_0 to any goal state in G_{det}, using Dijkstra's algorithm.

eGUBS Optimization. Once the parameter K_g is determined, we solve the unconstrained utility maximization problem using the eGUBS-VI algorithm [Crispino et al., 2023]. This algorithm handles the non-stationary nature of the optimal policy by operating in the augmented state space $(s, c) \in S \times \mathbb{R}$. It leverages the properties of the exponential utility function to define a cost threshold C_{max}, beyond which the policy becomes stationary, ensuring termination and computability.

We combine these components into the **S-GUBS algorithm**, depicted in Algorithm 1. The procedure takes a goal-oriented MDP and a safety threshold as input. It computes the topological-based and probabilistic bounds to calibrate K_g, then it verifies feasibility, and finally executes the utility maximization with K_g.

Computational Complexity. The overhead introduced by S-GUBS over the base GUBS framework is negligible. Constructing G_{det} and running Dijkstra cost $O(|A||S|^2)$ and $O(|E_{det}| + |S| \log |S|)$ respectively, and evaluating Eq. (10) is $O(1)$. The dominant computational cost is that of eGUBS-VI, which operates in the augmented space $S \times \{0, \ldots, C_{max}\}$ with overall complexity $O(|S|^2|A|\, n_{\text{iter}} + C_{max}^2|S||A| + C_{max}|S|^2|A|)$, where C_{max} is a computable stationarity threshold [Crispino et al., 2023, Section 7].

In the following section, we evaluate S-GUBS against state-of-the-art approaches in terms of policy quality.

6 EXPERIMENTS

6.1 BENCHMARK ENVIRONMENTS DESCRIPTION

We evaluate our method on two distinct domains selected for their complementary challenges: a standard planning benchmark with unavoidable dead-ends and a realistic safety-critical navigation task.

Exploding Blocksworld. First, we select the Exploding Blocksworld domain, introduced in the probabilistic track of the first International Probabilistic Planning Competition (IPPC 2004) [Younes et al., 2005]. This domain extends the classic Blocksworld by introducing a non-zero probability that a block is destroyed (or "explodes") when moved, potentially leading to an unavoidable dead-end state from which the goal configuration becomes unreachable. Due to its combinatorial state space and the presence of dead-ends, it serves as a canonical benchmark in the literature for goal-oriented MDPs. It has been extensively used to evaluate foundational frameworks including AtLeastProb [Steinmetz et al., 2016], SSPUDE [Kolobov et al., 2012], and *i-dual* [Trevizan and Teichteil-Königsbuch, 2017], making it an ideal candidate for comparative analysis.

Aircraft Routing. Second, we employ the Aircraft Routing domain, sourced directly from the Chance-Constrained SSP literature [Hong et al., 2021]. By using the exact environment definition proposed by the authors of the CC-SSP framework, we ensure a fair comparison against their method on their own benchmark. This environment models a mission planning task where an aircraft must navigate a topological map to reach a destination while avoiding dynamic hazardous zones (e.g., storm cells). We treat the undesirable states as dead-ends, thereby framing this problem within the class of goal-oriented MDPs with unavoidable dead-ends that our framework addresses.

6.2 EXPERIMENTAL SETUP AND PROTOCOL

Our experiments aim to compare the properties of the optimal policies generated by S-GUBS against those produced by state-of-the-art constrained solvers. We focus on qualitatively analysing policies, and we propose a close look at the conditional expected cost achieved by the different approaches.

S-GUBS. We implemented the S-GUBS algorithm as described in the previous section (see Algorithm 1). To isolate the theoretical contribution of the analytical mapping, our current implementation relies on standard dynamic programming (VI in the augmented space) and does not yet exploit heuristic search techniques available for the GUBS framework, such as those proposed by Crispino et al. [2023]. Consequently, resolution times for S-GUBS ranged between 5 and 15 minutes depending on the instance.

i-dual (Lexicographic Baseline). We compare against *i-dual* [Trevizan and Teichteil-Königsbuch, 2017], a state-of-the-art heuristic search algorithm for Constrained SSPs. As recommended by the authors for domains with unavoidable dead-ends, we assign a large finite penalty ($D = 1000$) to failure states. While Trevizan and Teichteil-Königsbuch [2017] discuss alternative paradigms for deploying *i-dual* in such contexts, these alternatives refer to methods like S3P or iSSPUDE, which structurally prioritize safety maximization. Thus, we utilize i-dual as the representative baseline for the lexicographic paradigm, as the penalty forces it to converge to the safest possible policies. Regarding computational performance, we note that the algorithm is very efficient, with resolution times typically around 1 second.

Chance-Constrained SSP (CC-SSP). We use the anytime algorithm proposed by Hong et al. [2021]. To ensure a fair comparison, we granted CC-SSP a runtime budget of 15 minutes (matching the upper bound of S-GUBS). To handle dead-ends within this framework, we adopted the modeling approach used by Hong et al. [2021], where the agent can execute a specific "give-up" action upon entering a dead-end state. This action transitions directly to the goal state with probability 1 and zero cost. We applied this mechanism to both Aircraft Routing and Exploding Blocksworld benchmarks, effectively treating dead-end entry as a zero-cost termination in the calculation of the expected cost.

For each configuration (Domain-Instance-P_{th}), we compute the optimal policy using each method and report: **Success Probability** ($P_{\mathcal{G}}^{\pi}$) as the probability of reaching the goal state; and, the **Conditional Expected Cost** (E_c^{π}), as the average cumulative cost of successful trajectories estimated empirically via Monte Carlo simulation over 10^5 trajectories. We highlight that the state-of-the-art approaches do not optimize for the conditional constrained criteria. Our analysis goal is to point out the modelling limitations and the solving tricks of current baselines (high penalty for i-dual, and give-up action with zero-cost for CC-SSP).

6.3 RESULTS AND ANALYSIS

Table 1 presents the performance of S-GUBS compared to CC-SSP and i-dual across the different domains. First, we observe that i-dual consistently behaves as a lexicographic solver, maximizing the probability of success in all cases, regardless of the requested threshold P_{th}. As previously pointed out, this rigidity often results in extreme inefficiency. A striking example is Exploding Blocksworld (Instance 3). To secure a marginal safety gain (probability 0.96 vs 0.94), i-dual accepts a policy with an expected cost of 53.9, whereas S-GUBS and CC-SSP find a policy with a cost of only 4. In this case, the lexicographic requirement induces a $13\times$ increase in cost for a 2% increase in safety, illustrating that maximizing safety at all costs is rarely desirable in resource-constrained problems.

Regarding the comparison between S-GUBS and CC-SSP, we must acknowledge that CC-SSP achieves better conditional efficiency in a majority of instances. This is a direct consequence of the conservative nature of our analytical bound : S-GUBS tends to "overshoot" the safety requirement, incurring higher costs to secure this extra margin. In contrast, CC-SSP typically converges to the most efficient policy that lies closest to the safety boundary, effectively exploiting the trade-off. However, despite this general trend, there are critical configurations where S-GUBS finds policies that are both safer and more efficient than CC-SSP.

Table 1: Experimental results comparing S-GUBS against CC-SSP and i-dual. Columns show the Conditional Expected Cost and Success Probability for varying safety thresholds P_{th} (P_{succ} denotes the probability of reaching the goal from s_0).

			S-GUBS		**CC-SSP**		**iDUAL**	
Domain	**Instance**	P_{th}	E_c^{π}	P_{succ}	E_c^{π}	P_{succ}	E_c^{π}	P_{succ}
Aircraft Routing	1	0.7	**5.1**	0.72	**5.1**	0.72	7.6	0.98
		0.8	7.4	0.95	**5.2**	0.82	7.6	0.98
		0.9	7.6	0.98	**6.1**	0.9	7.6	0.98
	2	0.7	**7.3**	0.93	7.5	0.85	8.6	1
		0.8	8.6	1	**7.5**	0.85	8.6	1
		0.9	8.6	1	**7.3**	0.93	8.6	1
	3	0.7	**13**	1	15	0.7	**13**	1
		0.8	**13**	1	**13**	1	**13**	1
		0.9	**13**	1	**13**	1	**13**	1
Exploding Blocksworld	1	0.4	**2**	0.72	**2**	0.72	14.3	1
		0.7	11.9	0.92	**2**	0.72	14.3	1
		0.95	**14.3**	1	**14.3**	1	**14.3**	1
	2	0.4	**3**	1	4	0.6	**3**	1
		0.7	**3**	1	**3**	1	**3**	1
		0.95	**3**	1	**3**	1	**3**	1
	3	0.4	**4**	0.94	**4**	0.92	53.9	0.96
		0.7	**4**	0.94	**4**	0.92	53.9	0.96
		0.96	**53.9**	0.96	**53.9**	0.96	**53.9**	0.96

6.4 DISCUSSION

One might hypothesize that CC-SSP failed to find the policies found by S-GUBS on configurations Routing Instance 2 ($P_{th} = 0.7$), Routing Instance 3 ($P_{th} = 0.7$), and Exploding Blocksworld Instance 2 ($P_{th} = 0.4$) (Table 2) due to insufficient runtime. However, an analysis of the objective function of CC-SSP reveals that this behavior is structural, not computational.

To deepen this analysis, Table 2 compares the unconditional expected cost – the metric actually minimized by CC-SSP (Eq. (4)) – for both the policy it selected (π_{CC}) and the policy found by our method (π_{S-GUBS}) in the three critical configurations identified (Routing Instance 2 ($P_{th} = 0.7$), Routing Instance 3 ($P_{th} = 0.7$), and Exploding Blocksworld Instance 2 ($P_{th} = 0.4$)). In all three cases, the policy π_{CC} has a lower unconditional cost than the policy π_{S-GUBS}. This confirms that the CC-SSP solver would never converge to the policy found by S-GUBS, regardless of the runtime. Since the S-GUBS policy yields a higher unconditional cost, it is strictly dominated according to the objective function optimized by CC-SSP.

This illustrates a severe manifestation of "failure-seeking bias". By treating entry into an undesirable state as a termi-

Table 2: Analysis of Objective Misalignment. For instances where S-GUBS strictly dominates CC-SSP (safer and more efficient), we compare the two resulting policies. The last column represents the actual metric minimized by CC-SSP, i.e. the unconditional (or total) expected cost.

Instance	Policy	Safety (P_{succ})	Cond. Cost (E_c^π)	Uncond. Cost
Routing 2	π_{S-GUBS}	**0.93**	**7.3**	6.79
($P_{th} = 0.7$)	π_{CC-SSP}	0.85	7.5	6.37
Routing 3	π_{S-GUBS}	**1.00**	**13.0**	13.0
($P_{th} = 0.7$)	π_{CC-SSP}	0.70	15.0	10.8
Exp. Blocks 2	π_{S-GUBS}	**1.00**	**3.0**	3.00
($P_{th} = 0.4$)	π_{CC-SSP}	0.60	4.0	2.80

nal event where cost accumulation ceases, the formulation creates a perverse incentive: the agent minimizes global cost by choosing risky paths where early failures artificially deflate the expectation. Crucially, the safety constraint $P_G^\pi(s_0) \geq P_{th}$ is satisfied: the solver does not violate the requested safety level. However, it satisfies the mathematical objective while violating the operational intent: within the set of safe policies, it selects one that takes unnecessary risks bringing no efficiency gain.

Qualitative analysis. To provide a concrete understanding of the policy differences hidden behind the aggregate metrics, we analyze a specific scenario from Table 1. We focus on Instance 3 of the Exploding Blocksworld domain ($P_{th} = 0.4$), represented in Figure 2. Specifically, we investigate why CC-SSP converges to a lower success probability than S-GUBS despite incurring an identical conditional expected cost.

In this specific problem, the three solvers yield distinct strategies that we detail here. The policy generated by i-dual seeks absolute safety: it repeatedly places C onto the irrelevant block D until an explosion occurs, effectively "disarming" the block before further manipulation. The S-GUBS policy first moves C onto D. Here, D acts as a buffer; if C explodes, the table remains intact. In contrast, the policy found by CC-SSP moves C directly to the table, accepting the risk of destroying it.

Following this initial step, both S-GUBS and CC-SSP policies employ the same sequence: placing B on the table, C on B, and finally A on C. However, due to its risky

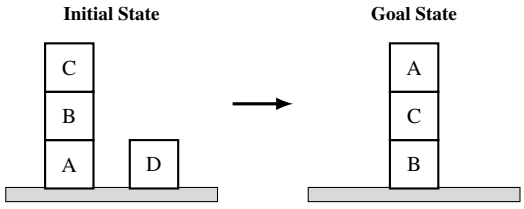

Figure 2: Configuration for Exploding Blocksworld (Instance 3). The agent must reorganize the blocks from the initial state (left) to form the target stack B-C-A (right). In this specific instance, the probability of a block exploding during a move is set to $P_{explod} = 0.02$.

first move, the CC-SSP solution results in a lower success probability (0.92 vs 0.94) despite having the exact same conditional cost (4). Exhibiting these specific choices illustrates the unnatural nature of this risk-seeking behavior: the CC-SSP policy incurs a 2% loss in safety that brings no efficiency gain, as the conditional cost remains identical. These observations reinforce the necessity of the conditional optimization criterion (Eq. (5)). Integrating failed trajectories into the efficiency metric creates an inherent conflict: assigning an infinite cost leads to the extreme conservatism of lexicographic approaches, while assigning a finite cost induces the failure-seeking bias risk exposed above. By removing failed trajectories from the efficiency metric entirely, the conditional optimization criterion resolves this dilemma.

Finally, S-GUBS exhibits desirable behavior regarding the user-defined threshold. As P_{th} increases, our analytical formula yields a higher K_g, naturally pushing the policy towards safer behaviors. Conversely, we observed cases where CC-SSP behaved erratically (finding a better policy when the constraint was tightened), highlighting that S-GUBS offers a more predictable control lever for system designers.

7 CONCLUSION AND PERSPECTIVES

In goal-oriented MDPs with unavoidable dead-ends, the standard criteria are often impractical for deployment: lexicographic safety maximization pays a prohibitive operational cost, while penalty-based methods offer no formal safety guarantee. We argued instead for minimizing the expected cost conditional on success, subject to a strict reachability constraint, and introduced S-GUBS, a tractable approximation built on utility maximization. Its core is a closed-form bound on the goal bonus K_g that suffices to guarantee the requested safety level P_{th}, together with the algorithm that turns this bound into a policy. Computing it adds essentially no overhead over the base solver; the flip side is that, relying only on aggregated structural information, the bound is conservative, so S-GUBS guarantees safety but may overshoot the target and incur a higher cost than strictly necessary.

Several directions remain open. Tightening the bound would reduce this conservatism while preserving the guarantee, and heuristic search should improve scalability. More fundamentally, S-GUBS does not solve the Conditional SSP of Eq. (5) exactly: we are currently investigating fractional programming techniques to optimize the conditional objective directly, without going through a utility-based reformulation.

References

Eitan Altman. *Constrained Markov Decision Processes*. CRC Press, London, 1999.

Dimitri P. Bertsekas and John N. Tsitsiklis. An analysis

of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.

Krishnendu Chatterjee, Rupak Majumdar, and Thomas A Henzinger. Markov decision processes with multiple objectives. In *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 3884 of *LNCS*, pages 325–336. Springer, 2006.

Gabriel Nunes Crispino, Valdinei Freire, and Karina Valdivia Delgado. GUBS criterion: Arbitrary trade-offs between cost and probability-to-goal in stochastic planning based on expected utility theory. *Artificial Intelligence*, 316:103848, 2023.

Valdinei Freire and Karina Valdivia Delgado. GUBS: A utility-based semantic for goal-directed markov decision processes. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 741–749, 2017.

Sungkweon Hong, Sang Uk Lee, Xin Huang, Majid Khonji, Rashid Alyassi, and Brian C Williams. An anytime algorithm for chance constrained stochastic shortest path problems and its application to aircraft routing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 475–481. IEEE, 2021.

Andrey Kolobov, Mausam, and Daniel S. Weld. A theory of goal-oriented MDPs with dead ends. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 438–447. AUAI Press, 2012.

Mausam and Andrey Kolobov. *Planning with Markov Decision Processes: An AI Perspective*. Morgan & Claypool Publishers, 2012.

Marcel Steinmetz, Jörg Hoffmann, and Olivier Buffet. Goal probability analysis in probabilistic planning: Exploring and enhancing the state of the art. *Journal of Artificial Intelligence Research (JAIR)*, 57:229–271, 2016.

Florent Teichteil-Königsbuch. Stochastic safest and shortest path problems. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1825–1831. AAAI Press, 2012.

Felipe Trevizan and Florent Teichteil-Königsbuch. Efficient solutions for stochastic shortest path problems with dead ends. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 1–10. AUAI Press, 2017.

Felipe Trevizan, Sylvie Thiébaux, Pedro Santana, and Brian Williams. I-dual: Solving constrained SSPs via heuristic search in the dual space. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4399–4405, 2017. doi: 10.24963/ijcai.2017/613.

Håkan L. S. Younes, Michael L. Littman, David Weissman, and John Asmuth. The first probabilistic track of the International Planning Competition. *Journal of Artificial Intelligence Research*, 24:851–887, 2005.