SafeMAP: Safe Multi-Agent Planning framework based on Dynamic Probabilistic Risk Assessment

Mohammad Hejase[1] and Portia Banerjee[2]

[1] *NASA Ames Research Center, Moffett Field, CA 94035.*
mohammad.hejase@nasa.gov

[2] *KBR, NASA Ames Research Center, Moffett Field, CA 94035.*
portia.banerjee@nasa.gov

ABSTRACT

This paper proposes a risk-aware framework for Safe Multi-Agent Planning (SafeMAP) that unifies disparate models for multi-agent systems in a Markovian process that allows for simultaneous system health monitoring, decision making under uncertainty, and multi-agent system collaboration. As operations beyond low earth orbit mature, there is an increased need for autonomous cyber-physical systems with onboard decision making capabilities. Multi-agent cyber-physical systems in particular offer the potential of increased efficiency, resiliency, and mission capabilities for future applications such as multi-rover terrain operations, distributed satellite operations, and management of smart lunar habitats. SafeMAP utilizes physics-based models of each agent and the relevant components, probability models of the environment and component operational states, and reward models for mission-specific objectives such as scientific task completion or resource consumption. The output of SafeMAP is a set of mission plans that satisfy the mission objective under specified risk/reward constraints. A readable interpretation of each of these generated mission plans is provided as an additional output. SafeMAP has been demonstrated on a simulated case study involving a four-rover system performing surface mapping operations and science tasks. Results of this paper demonstrate SafeMAP's ability to generate explainable mission plans that satisfy the mission objective while minimizing risk under nominal and off-nominal conditions.

1. INTRODUCTION

Multi-agent planning is a necessary component for efficient implementation of distributed systems, essentially a group of agents communicating over a network to achieve a common goal. Distributed systems have been an integral part of several NASA missions such as distributed space system of multiple cubesats in Starling (Cramer et al., 2021), distributed systems with human astronaut interactions (Clancey et al., 2004), fractionated spacecraft systems (Faber et al., 2014), and multi-UAV cooperative missions (Casbeer, Beard, McLain, Li, & Mehra, 2005).

Similarly, distributed systems can be employed for planetary surface exploration by rovers. Planetary surface exploration has traditionally been performed by large, complex, single vehicle systems with high functionality and high science return potential. Most single rover systems such as the Mars Science Laboratory's Curiosity and the Mars 2020 Perseverance rovers are equipped with multiple science instruments on one platform and rely primarily on ground station support to perform their traverses. They are mechanically limited to a set of terrains suitable to their configuration. Additionally, areas of higher risk need to be completely avoided in order to maintain mission success due to no rover redundancies (Kobayashi, Fujiwara, Yamakawa, Yasufuku, & Omine, 2010).

With recent advances in robotic systems and associated software technology in autonomy, there has been an emerging interest towards replacing larger vehicles with smaller low-cost rovers developed as a mobile utility service to support lunar payloads (Kalita & Thangavelautham, 2020) . Successful multi-agent rover missions would allow extension of mission and communication range. This is of particular importance in missions involving exploration of permanently shadowed regions or lava tubes/caves (Kalita & Thangavelautham, 2021). Such missions are very challenging for single rover systems. Moreover utilizing multi-agent systems offers improvements in mission resiliency as agents may be redistributed based on mission status. One of the obvious consequences of a paradigm shift from larger vehicles to smaller rover systems is the added need for intelligent task management and coordination required in order to maintain the same level of system functionality or even improve the science return as a single larger system.

Path-planning tools allow the mission designers to understand how multiple assets can be used on the surface of a planet and how they can work together to achieve the mission goals, even in the presence of faults in one of the agents. Moreover, the availability of environment information, such as the illumination properties, communication visibility and slope profiles can help the designer to best utilize the vehicles' resources. The primary focus of this paper is to propose a multi-agent planning framework that contains the necessary building blocks to allow for intelligent and risk aware planning in multi-agent systems.

Section 2 of this paper provides a literature review of planning tools for multi-agent systems with a focus on tools utilizing Markov Decision Processes (MDPs). Section 3 proposed the the Safe Multi-Agent Planning Framework based on an inductive implementation of the Markov Cell-to-Cell Mapping technique. Section 4 contains a 4 rover surface area exploration case study to demonstrate SafeMAP. Section 5 contains the case study results and discussion. The last section of this paper is the conclusion.

2. LITERATURE REVIEW: MULTI-ROVER PLANNING TOOLS

A multi-agent system can significantly expand the capabilities of a planetary prospecting and exploration mission, yet this paradigm comes with challenges. Agents in this setting lack full-autonomy and are remotely controlled by human operators on Earth. Increasing the number of agents from one to many without any additional autonomy will increase human operator burden. In order to achieve practical implementation of such systems, several studies have been ongoing to improve autonomy in multi-agent planning systems. Particularly three solvers for multi-agent planning, (1) MARMOT, (2) SEXTANT and (3) PDRA, will be described in this Section.

- MARMOT: Multi-Agent Resource Mission Operations Tools, is an extensible, open-source tool that allows operators to expand and improve current functionalities and automation capabilities of mission planning systems for extra-terrestrial operation by aiming to reduce the exhaustive mental load accumulated by mission operators (Kakish et al., 2019). In the case of Lunar prospecting, opportunistic path planning for multiple agents consists of selecting points-of-interest that will yield the greatest amount of scientific return while maintaining constant communication between the agents. Attempting to solve this requires an extensive state-action space due to discretizing states into number of agents, number of points-of-interest, and path proposals.MARMOT attempts to reduce this space by splitting the problem into two parts. The first selects and manages the paths of the multiple agents relative to the points-of-interest by using a brute-force traveling salesman solver. An A* algorithm is then used to establish an approximated best traversable route between 2 or n points-of-interest. The second part takes the approximated paths for each agent and uses a version of the Distributed Path Consensus (DPC) algorithm to select the traversal paths that maintains continued communication between the agents (Bhattacharya, Likhachev, & Kumar, 2010).
- SEXTANT: A path-planning tool aimed for planetary multi-rover systems known as the Surface Exploration Traverse Analysis and Navigation Tool (SEXTANT) was developed at the Massachusetts Institute of Technology (MIT), beginning in 2001 (Norheim, 2018). With this tool, a user selects a set of waypoints over a terrain digital elevation model, and the Geologic Traverse Planner draws a path between these waypoints. Further capabilities include calculating the illumination properties during the traverse. This enables the computation of the thermal load on suited astronauts and the power properties of solar powered rovers over the entire traverse. SEXTANT was also enhanced to allow its use as a simulation tool for fractionated and spatially distributed heterogeneous planetary surface vehicles. The primary algorithm behind SEXTANT is the A* optimization algorithm that is used to calculate the optimal route of the rovers from point to point in terms of energy expended, distance, or speed. While planning for more than one vehicle using SEXTANT, paths are optimized sequentially i.e. the first rover plans its path without considering the other rovers. Each successive rover then plans its path such that there are no conflicts with its predecessors. This approach under-utilized a multi-agent system in terms of functional and spatial distribution of assets.
- PDRA : Developed by Rossi et al. at NASA JPL, the Pluggable Distributed Resource Allocator (PDRA) serves as a middleware for distributed computing in heterogeneous mobile robotic networks (Rossi, Vaquero, Sanchez-Net, da Silva, & Vander Hook, 2020). PDRA enables autonomous robotic agents to share computational resources for computationally expensive tasks such as localization and path planning. Task allocation decisions are performed by a mixed-integer programming algorithm, solved in a shared-world fashion, that models CPU resources, latency requirements, and multi-hop, periodic, bandwidth-limited network communications. The algorithm can minimize overall energy usage or maximize the reward for completing optional tasks. Simulation results show that PDRA can reduce energy and CPU usage by over 50% in representative multi-robot scenarios compared to a naive scheduler; runs on embedded platforms; and performs well in delay- and disruption-tolerant networks (DTNs).

One of the key components missing in all the above multi-agent planning tools is the notion of "system health". While it is critical to maintain communication with Earth, maintaining 'health' of the system autonomously is equally important to ensure mission success. Health in a multi-agent paradigm can be at two levels. Level 1: The individual agent's health (each of its components should be in their nominal operational range such as motor temperature, battery state of charge, etc.); Level 2: Overall system's health dependent on operation environment (obstacles in the path of a rover, uncertain terrain or slope causing slips or roll-overs, degraded navigation links etc.). Accurate estimation of state-of-health of the system can enable quick decision making by human operators to modify roles of each agents thereby maintaining mission success. Hence, if multi-agent systems are expected to increase mission resiliency compared to their single-agent counterparts, assessing system health and integrating that information into the planning tool becomes necessary. In this paper, previous health assessment techniques implemented on UAV systems (Corbetta, Kulkarni, Banerjee, & Robinson, 2021) will be explored and tailored towards multi-agent planning.

3. SAFE MULTI-AGENT PLANNING (SAFEMAP)

3.1. Overview of the SafeMAP Framework

The objective of the implemented prototype of SafeMAP in this paper is to provide a generalized framework that can be generically used for multi-agent systems and can capture risk information (i.e. health) from various and disparate sources. The solver used in the version of SafeMAP prsented in this paper is based on the Markov/CCMT solver. This can be represented as a Multi-Agent Markov Decision Process (MDP) and is implemented in a modular fashion that allows future improvement and substitution of the solver with ones that are more suited to the computational challenges that multi-agent systems face. Examples of such solvers are Decentralized Partially Observable MDPs (Dec-POMDPS) and Macro Action POMDPs (Mac-POMDPs) (Amato & Oliehoek, 2015; Amato, Konidaris, Kaelbling, & How, 2019). Additionally, an objective of this paper is to generate output in an interpretable fashion that allows for translation of search results into user-friendly mission plans that clearly outline and detail system dynamics evolution, system risk evolution, and system reward or objective evolution. The main innovation in the implementation of SafeMAP stems from the adaptation of tools and technologies utilized for Dynamic Probabilistic Risk Assessment tools into a tool used for risk-aware planning for multi-agent systems. The modular nature of SafeMAP is also innovative as it enables adaption to various use cases with different constraints and requirements by allowing for the use of a wide array of inputs in relation to system and risk models, solvers, and outputs.

The SafeMAP framework for mission planning works in a sequence of several stages as illustrated in Figure 1. The first stage is State Space Definition, followed by Cell-Space creation, Construction of a cell-to-cell map, and finally implementation of the solver that will compute the optimal mission plan with respect to risk and reward metrics. Within State-space definition, the objective is to define the continuous variables that govern system behavior, and the discrete state variables that dictate system component state (e.g. nominal, degraded). Continuous state variables (position, velocity, state-of-charge, etc) are ones that can generally be modeled via differential equations. Whereas discrete state variables (weather, terrain conditions, etc) can be thought of as variables whose behaviors are governed by state transition models such as truth tables, finite state machine, Bayesian belief networks, etc. Cell-Space creation involves discretizing the continuous state variables into a set of discrete states and creating a system cell space that is composed of all discretized continuous variables augmented to the discrete state variables. Cell-Space evolution information is then obtained using a system simulator. A cell-to-cell map is created from all combinations of system states and component conditions and their transitions within a user-defined time-step. Such transitions result in a probabilistic mapping of the system state-space onto itself which allows for simultaneously capturing system trajectories and quantifying risks associated with such trajectories.

Markovian solvers such as MDPs or POMDPs utilize the cell-to-cell map representation of the system along with rewards models and search algorithms in order to generate a set of plans and prune them to a plan that satisfies the mission risk/reward criteria. An inductive version of a Centralized MDP algorithm is utilized in this framework for the exploration of paths that originate from a user-specified set of initial conditions and propagate for a user-specified time-horizon. A generic reward functionality is also added and used in conjunction with the risk models to guide plan selection and pruning. The developed exploration algorithm can be thought of as a search tree that uses a probabilistic map of the system state space on itself along with generated rewards information. This search tree structure is achieved by recursive enumeration of subtrees from a set of initial conditions and the traversal of possible paths through a branching process. In order to alleviate the inevitable problem of computational explosions during the search process, only reward-significant scenarios with acceptable levels of risks as specified by user cutoff values are retained.

3.2. Markov Cell-to-Cell Mapping Technique

The Cell-to-Cell Mapping Technique (CCMT) has been proposed and developed in the 1980s to facilitate an efficient and practical way of determining global behavior of chaotic systems. CCMT regards the state space of a system as a collec-

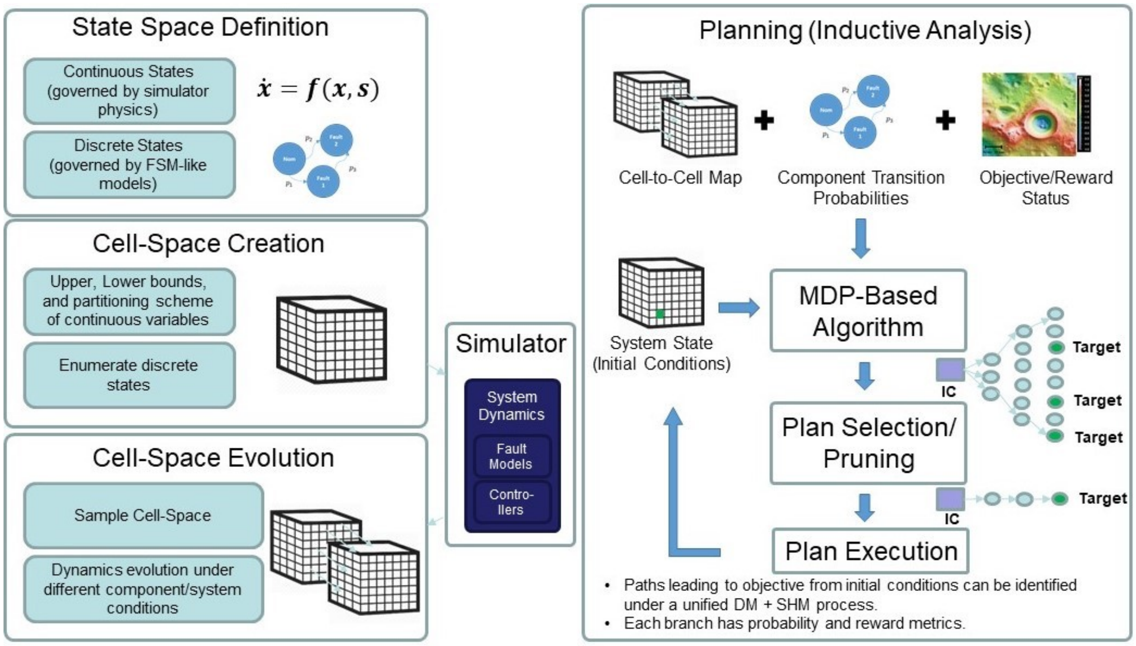

State Space Definition
Continuous States (governed by simulator physics)
$\dot{x} = f(x, s)$
Discrete States (governed by FSM-like models)
Cell-Space Creation
Upper, Lower bounds, and partitioning scheme of continuous variables
Enumerate discrete states
Cell-Space Evolution
Sample Cell-Space
Dynamics evolution under different component/system conditions
Simulator
System Dynamics
Fault Models
Contro-llers
Planning (Inductive Analysis)
Cell-to-Cell Map
Component Transition Probabilities
Objective/Reward Status
System State (Initial Conditions)
MDP-Based Algorithm
Plan Selection/ Pruning
Plan Execution
IC
Target
Target
Target
IC
Target
• Paths leading to objective from initial conditions can be identified under a unified DM + SHM process.
• Each branch has probability and reward metrics.

Figure 1. The SafeMAP Framework

tion of a number of cells rather than a continuous space (Yang & Aldemir, 2016).

Markov/CCMT is a Markovian interpretation of generalized CCMT and can be used as a Dynamic Probabilistic Risk Assessment (DPRA) tool to provide metrics for system reliability and safety (Hejase, Kurt, Aldemir, Özgüner, et al., 2018; Hejase, Kurt, Aldemir, Ozguner, et al., 2018). It extends the concepts from the generalized cell-to-cell mapping and allows capturing interactions between the system dynamics and time-varying system configurations. As a result, cell-to-cell transitions can be used to produce a probabilistic mapping of the entire system state space onto itself, including system configuration state transitions such as hardware normal or faulted states over a user-defined time-step Δt.

It is assumed that planning is performed over a sequence of time-steps with interval length Δt. Two assumptions are placed on the system of interest in order to employ Markov/CCMT in planning or risk assessment algorithms.

Assumption 1: The system component states are fixed over an arbitrary interval $[t, t + \Delta t)$, but are prone to change at the boundary $t + \Delta t$.

Assumption 2: Transitions among cells do not depend on system history.

The first assumption allows factoring faults or system component changes in the planning algorithm once every planning interval. Through proper selection of the time-step Δt, the system component changes and the probabilities of these state changes can be realistically modeled and captured. The second assumption is the Markov property which is necessary in Markovian planning frameworks. When necessary, the second assumption can be relaxed through the use of auxiliary states.

3.3. System Cell Representation

The continuous L dimensional state space is represented by $\mathcal{X}\triangleq\mathbb{R}^L$. The M dimensional discrete state space of the system components is represented by $\mathcal{N}\triangleq\mathbb{Z}^M$. The space $\mathcal{X}\triangleq\mathbb{R}^L$is discretized by partitioning each continuous variable x_l $(l = 1, \ldots, L)$ into intervals of J_l partitions and considering combinations of those partitions to form the cells. Knowledge of the state-space upper bounds $\overline{x}$, and lower bounds $\underline{x}$ is required for the partitioning. The cells can be regarded as means to accommodate epistemic uncertainties (such as model uncertainties) or aleatory uncertainties (such as process noise and minor environmental disturbances).

The possible states of each hardware component M of interest are then defined (e.g. operational, degraded, failed), with each component m, having N_m possible states, each denoted by n_m $(m = 1, \ldots, M)$.

The unique combinations of the partitioned $\mathcal{X}\triangleq\mathbb{R}^L$ along with the discrete system component configurations forms the complete state-space of the system, denoted by $\mathcal{V}$. Each cell in the cell space is represented by an $(L+M)$ dimensional vector $[\mathbf{j}\ \mathbf{n}] \equiv [j_1, \ldots, j_l, \cdots, j_L, n_1, \ldots, n_m, \cdots, n_M]$, where $(j_l = 1, 2, \ldots, J_l; l = 1, \ldots, L)$ enumerate the partitioning of the interval $\underline{x_l} \leq x_l < \overline{x_l}$, and n_m represents the state of component m

$(n_m = 1, \ldots, N_m;\ m = 1\ , \ \ldots,\ M)$. The cell space $\mathcal{V}$ is composed of $J \times N$ unique cells with $J = J_1 \times \cdots \times J_L$ and $N = N_1 \times \cdots \times N_M$.

Let $\mathcal{V}_{\mathcal{X}} \triangleq \mathbb{Z}^L$ be a subspace of $\mathcal{V}$ containing the vectors $\mathbf{j}$. Let $\mathcal{V}_{\mathcal{N}} \triangleq \mathbb{Z}^M$ be a subspace of $\mathcal{V}$ containing the vectors $\mathbf{n}$. Note that $\mathcal{V}_{\mathcal{X}} \cup \mathcal{V}_{\mathcal{N}} = \mathcal{V}$.

Using the Markov property, and as derived in (Aldemir, 1987), the cell-to-cell probabilities over a single time-step transition Δt can be calculated from

$$q\left(\boldsymbol{j}, \mathbf{n} |\, \mathbf{j'}, \mathbf{n'}, \Delta t\right) = h\left(\mathbf{n} | \mathbf{n'}, \mathbf{j'} \rightarrow \mathbf{j},\ \Delta t\right) \times g\left(\mathbf{j} | \mathbf{j'}, \mathbf{n'}, \Delta t\right) \tag{1}$$

where $g\left(\mathbf{j}|\mathbf{j'}, \mathbf{n'}, \Delta t\right)$represents the transition probability from cell $\mathbf{j'}$ to $\mathbf{j}$ over Δt under configuration $\mathbf{n'}$, and $h\left(\mathbf{n}|\mathbf{n'}, \mathbf{j'} \rightarrow \mathbf{j},\ \Delta t\right)$ quantifies the system configuration transition probabilities over Δt.

3.4. Cell-Space Evolution

For each component of interest m, a component state transition probability matrix H_{n_m} is constructed. Contents of this matrix represent the probability of component state transitions over Δt. These probabilities can be based on hardware component data, such as failure rates, or expert opinion in the absence of reliable data. An example of such a matrix can be seen in Table 1 where $\lambda_{n'_m, n_m}$ denotes the transition rate from n'_m to n_m.

Using the Chapman-Kolmogorov equation under the assumptions stated earlier, the system cell-to-cell state transition probabilities $g\left(\mathbf{j}|\mathbf{j'}, \mathbf{n'}, \Delta t\right)$ over a single time-step can be found from (Yang & Aldemir, 2016)

$$g\left(\mathbf{j}|\mathbf{j'}, \mathbf{n'}, \Delta t\right) = \frac{1}{v_{j'}} \int_{v_{j'}} u_{\mathbf{j}} \left[\mathbf{x}(\mathbf{x'}, \mathbf{n'}, \Delta t)\right] dx' \tag{2}$$

$$u_{\mathbf{j}} \left[\mathbf{x}(\mathbf{x'}, \mathbf{n'}, \Delta t)\right] = \begin{cases} 1 & if\ \mathrm{x} \in v_{\mathbf{j}} \\ 0 & otherwise \end{cases} \tag{3}$$

where $v_{\mathbf{j}}$ is the volume of the cell $\mathbf{j}$,

$$\mathbf{x}(\mathbf{x'}, \mathbf{n'}, \Delta t) = \int_t^{t+\Delta t} f\left(\mathbf{x}\left(t'\right),\ \mathbf{n'}\right) \boldsymbol{dt'} + \mathbf{x'} \tag{4}$$

and $f\left(\mathbf{x}\left(t'\right),\ \mathbf{n'}\right)$ represents the equations describing system dynamics.

It is generally not practical or possible to evaluate (4) for complex systems. These equations can be approximated by utilizing the simulator via the use of an equal weight quadrature approximation scheme. Rather than integrating over a cell, multiple points are sampled from each cell and are passed to a simulator to compute transitions over Δt. Then (2) can be approximated as

$$g\left(\mathbf{j}|\mathbf{j'}, \mathbf{n'}, \Delta t\right) = \frac{\#\ samples\ in\ \mathbf{j}'\ arriving\ in\ \mathbf{J}\ over\ \Delta t}{\#\ points\ sampled\ from\ \text{cell}\ \mathbf{j}'} \tag{5}$$

3.5. Inductive Search Algorithm

An inductive search algorithm is implemented to construct the search tree. The search algorithm is Breadth First in nature but utilizes truncation criteria to keep the search tractable. A probability truncation criterion ϵ_P is used to truncate all paths that fall below a user-specified risk threshold. A reward truncation criterion ϵ_R is used to truncate all paths that fall below a user-specified rewards threshold. A branching criterion ϵ_B is used to limit the number of cells retained from branching in the search tree. Note that all truncation criteria can be defined as vectors where each entry represents the criterion value at a different search depth.

The algorithm described in Algorithm 1 is employed to construct an inductive search tree of depth K originating from the cell c^0_{IC}. Within the algorithm, the user first specifies the search depth K and truncation criteria ϵ_P, ϵ_R, and ϵ_B. The user also specifies the rewards function f_R and a sorting algorithm $\mathcal{S}$ used to rank and retain selected exploration paths when cells exceed the branching truncation criterion ϵ_B. Such a sorting algorithm can also be utilized to diversify exploration paths and eliminate bias in the search results. Algorithm 1 then branches out and expands the search tree up to a search depth K by utilizing the system simulator, probabilistic models, and reward models all while ensuring retained paths do not violate the defined truncation criteria. The result of the search tree is a set of paths that originate from the system initial conditions and aim to maximize rewards while minimizing risks. Each node of the search tree is represented by the cell tuple $[\mathbf{j}, \mathbf{n}]$ described in Section 3.3 where $\mathbf{j}$ is the tuple representing system discretized location in the state space (e.g. position velocity cell locations) and $\mathbf{n}$ is the tuple representing system component states (e.g. component states or actions). Each of the nodes also has a cumulative reward metric and path probability metric from the initial conditions. The use of the tuples to describe nodes of the search tree allows for explainability of all paths obtained from the search algorithm in terms of system states, risk probability, and rewards evolution in Δt intervals over the search horizon.

A visualization of the search tree with the relevant notations used in Algorithm 1 can be seen in Figure 2.

The top node of the search tree is the System Initial Conditions (defined by a cell), and each subsequent search depth represents an inductive evolution of system dynamics and configurations over Δt. Each depth $k \in 0, \ldots, K$ has M_k cells $c^k_{m_k}$ such that $m_k \in 1, \ldots, M_k$. For each cell m_k,

Table 1. Sample System Configuration Transition Matrix H_{n_m}

		Final System Component State			
		Normal State (N)	Failure State 1 (F_1)	... ...	FailureState N (F_N)
Initial System Component State	Normal State (N)	$\lambda_{N,N}\Delta t$	$\lambda_{N,F_1}\Delta t$	...	$\lambda_{N,F_N}\Delta t$
	Fail 1 State (F_1)	$\lambda_{F_1,N}\Delta t$	$\lambda_{F_1,F_1}\Delta t$	...	$\lambda_{F_1,F_N}\Delta t$
	$\vdots$	$\vdots$	$\vdots$	-	$\vdots$
	Fail N State (F_N)	$\lambda_{F_N,N}\Delta t$	$\lambda_{F_N,F_1}\Delta t$	...	$\lambda_{F_N,F_N}\Delta t$

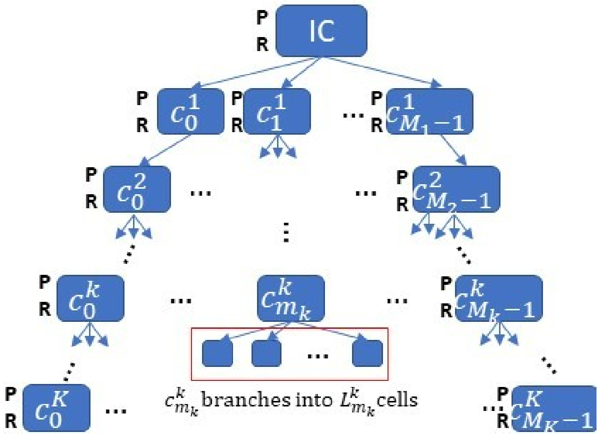


Figure 2. Search Tree Visualization

the variable $L_{m_k}^k$ represents the number of cells that cell m_k branches into inductively over a single time-step. A path probability $P_{m_0|m_1|\ldots|m_K}$ and reward value $R_{m_0|m_1|\ldots|m_K}$ from the initial condition to up to cells in the bottom node $c_{m_K}^K$ is defined at each node. Note that a cell $c_{m_k}^k$ here represents the m_k-th tuple [j,n] at a search depth k.

4. CASE STUDY

4.1. Case Study Description

A multi-agent simulator of 4 rovers exploring a 50m x 50m area of interest was constructed in MATLAB/Simulink environment. The states of the rovers used for analysis were the x and y positions of the rovers in the exploration area reference frame. The rovers are assumed to traverse at a velocity set-point of $0.1m/s$. The motor state of a rover i, denoted as M_{ri}, was modeled as the component of interest with two operational modes: Nominal ($M_{ri} = 1$) and degraded ($M_{ri} = 2$).

The area of exploration was divided into a uniform 5×5 grid map and a reward function was created to represent the number of map cells that were explored such that $R \in \{1, \ldots, 25\}$. Rovers are assumed to be able to take one of three actions: Exploring cell in the forward direction $A_{ri} = 1$, exploring cells to the right $A_{ri} = 2$, or exploring cells to the left $A_{ri} = 3$. Based on the actions that the rovers may take, a probabilistic model of the motor state was constructed as seen in Tables 2 and 3. Note that the assigned probability values are arbitrary for demonstrative purposes in this case study.

The objective of the multi-rover system is exploration of the entirety of the grid area while minimizing the risk of failure.

Table 2. Probability Matrix for Forward Exploration Action $A_{ri} = 1$

$P(M_{ri}(t+\Delta t) = a$ $\lvert M_{ri}(t) = b, A_{ri} = 1)$	Nominal M$_{ri}$(t) $= 1$	Degraded M$_{ri}$(t) $= 1$
Nominal M$_{ri}$(t+Δt) = 1	1-1e-3	0
Degraded M$_{ri}$(t+Δt) = 2	1e-3	1

Table 3. Probability Matrix for Left or Right Direction Exploration Action $A_{ri} \in \{2, 3\}$

$P(M_{ri}(t+\Delta t) = a$ $\lvert M_{ri}(t) = b, A_{ri} \in \{2,3\})$	Nominal M$_{ri}$(t) $= 1$	Degraded M$_{ri}$(t) $= 1$
Nominal M$_{ri}$(t+Δt) = 1	1-5e-3	0
Degraded M$_{ri}$(t+Δt) = 2 or 3	5e-3	1

4.2. Cell Space Definition

In order to create the cell-space, the system continuous variables are first discretized and then augmented to the system configuration states as described in Table 4.

The $x_i, y_i | i \in \{1, \ldots, 4\}$ position states are segmented uniformly into 5 partitions with an upperbound of $50m$ and a lower bound of $0m$. The system component is defined to be the 4 rovers "Control Mode" and has three modes of operation defined for each rover: "Forward Area Exploration", "Right Area Exploration", and "Left Area Exploration". The result of this discretization is the system state $[\mathbf{j}, \mathbf{n}]$ where $\mathbf{j} = [j_1, \ldots, j_8]$ is the tuple representing system location in the discretized continuous states and $\mathbf{n} = [n_1, \ldots, n_4]$ is the tuple representing the multi-agent system actions. Each of the 4 rovers are initially assumed to be at the starting location of $(0, 0)$ with nominal Motor States. The time-step Δt is selected to be $100s$ (enough time to arrive at next grid location).

Algorithm 1 Inductive Search Algorithm

```
 1: Input $K, \epsilon_P, \epsilon_R, \epsilon_B, f_R,$ and $\mathcal{S}$
 2: Initialize the current Search Depth ($k = 0$).
 3: Initialize $P_0 = 1$, $R_0 = 0$, $c_0^0 = c_{IC}^0, i = 0$, $M_0 = 1$, $M_1 = 0, \dots, M_K = 0$
 4: Determine all cells with valid transitions from $c_i^k$. There $L_{m_k}^k$ such cells $l_j : j \in \{0, \dots, L_{m_k}^k - 1\}$
 5: while $j < L_{m_k}^k$ do                                  ▷ loop through cells originating from cell $c_i^k$
 6:     Compute path probability $\mathbf{P}_{l_j} = P_{l_j}|P_{c_i^k}| \dots |P_{c_0^0}$
 7:     Compute path reward $\mathbf{R}_{l_j} = f_R(l_j|c_i^k, \dots, c_0^0)$
 8:     if $\mathbf{P}_{l_j} <= \epsilon_P$ OR $\mathbf{R}_{l_j} <= \epsilon_R$ then   ▷ truncate risk-significant or rewards-insignificant paths
 9:         delete $l_j$
10:         $L_{m_k}^k = L_{m_k}^k - 1$
11:     else
12:         keep $l_j$
13:         $j = j + 1$
14: $M_{k+1} = M_{k+1} + L_{m_k}^k$                              ▷ Update Number of cells in depth $k + 1$
15: if $M_{k+1} > \epsilon_B$ then
16:     $\{l_0, \dots, l_{\epsilon_B}\} = \mathcal{S}(l_0, \dots, l_{M_{k+1}-1})$   ▷ Bound to Branching Limit $\epsilon_B$ based on selection algorithm $\mathcal{S}$
17:     $M_{k+1} = \epsilon_B$
18: $i = i + 1$
19: if $i <= M_k$ then
20:     go to Step 4                                     ▷ if there are more cells in depth k, go to next cell
21: else
22:     $k = k + 1$
23:     if $k \leq K$ then                                  ▷ if not at max depth, go to next depth
24:         $i = 0$
25:         go to Step 4
26:     else
27:         END                                         ▷ End when search up to depth limit
```

Table 4. User Input to SafeMAP: System Definition

Variable Name	Notation	Value
numContinuousVariables	L	8
continuousVariablesNames		[“x-pos rov1.”, “y-pos rov1”, “x-pos rov2”, “y-pos rov2” , “x-pos rov3”, “y-pos rov3” , “x-pos rov4”, “y-pos rov4”]
variableUpperBounds	$\overline{x}$	[50, 50, 50, 50, 50, 50, 50, 50]
variableLowerBounds	$\underline{\text{x}}$	[0,0,0,0,0,0,0,0]
numberOfCells	J_i	[5,5,5,5,5,5,5,5]
numSystemComponents	M	1
systemComponentNames		[“Controller Action”]
systemComponentStates	N_i	[3]
systemComponentStateNames		[“Forward Grid Exp”, “Right Grid Exp”, “Left Grid Exp”]
sysConfTransProb	H_{n_m}	$\begin{bmatrix} 1-1e-3 & 1-1e-3 & 1-1e-3 \\ 1-5e-3 & 1-5e-3 & 1-5e-3 \\ 1-5e-3 & 1-5e-3 & 1-5e-3 \end{bmatrix}$
icLocation	IC	[0,0,0,0,0,0,0,0]
timeStep	Δt	100

Table 5. User Input to SafeMAP: The Search Tree

searchDepth	K	9
truncProb	ϵ_P	0.87
truncReward	ϵ_R	3 @ k=1 6 @ k=2 10 @ k=3 13 @ k=4 16 @ k=5 19 @ k=6 22 @ k=7 23 @ k=8 24 @ k=9
truncBranching	ϵ_B	50

4.3. Search Tree Setup

The user input to SafeMAP for the baseline case “Case 1” is then utilized by the search tree along with the system simulator to construct a system cell-to-cell map. Such a map contains the system transitions over the time-step Δt from all possible permutations of the system cell space values. In this case study this is the product of the number of partitions of each state of the system, the system component states (motor state and control actions) $5 \times 5 \times 5 \times 5 \times 5 \times 5 \times 5 \times 5 \times 3 \times 3 \times 3 \times 3 = 31640625$ unique system cells. Note that in this case study risk was captured by modeling risk of motor failure based on control action performed. Should the user wish to do so, it is possible to also augment motor states to generate plans under failed system conditions.

Once the cell-to-cell map has been constructed the user defined the search tree criteria as seen in Table 5. A search tree depth of $K = 9$ means that mission plans will be generated from $t = t_0 = 0s$ up to $t = 9 \times \Delta t = 900s$. The truncation probability ϵ_P was arbitrarily selected to be 0.9 for all search depths. This would typically be selected based on acceptable risk levels. The reward criteria was selected in a manner that restricts the search algorithm to find paths that yield new rewards in each time-step. The branching truncation criterion was arbitrary selected to be 50 for all search levels. This means that in each search depth a maximum of 50 paths are to be retained.

It is important to note that algorithm efficiency and ability of finding a solution is very strongly linked to intelligent selection of truncation criteria.

SafeMAP was then run for several cases to demonstrate the effect of search parametrization on the generated plans. Table 6 depicts the different parameters that SafeMAP was run with. Case 2 executes SafeMAP with relaxed reward criteria compared to Case 1. Case 3 executes SafeMAP with more restrictive branching $Epsilon_B$.

Table 6. Case Studies 2 and 3 Parameter Variations. (k: timestep)

Case Study 2	Parameter	Value
truncReward	ϵ_R	3 @ k=1 5 @ k=2 9 @ k=3 11 @ k=4 13 @ k=5 16 @ k=6 18 @ k=7 20 @ k=8 24 @ k=9
Case Study 3	Parameter	Value
Branching Limit	ϵ_B	20

5. RESULTS

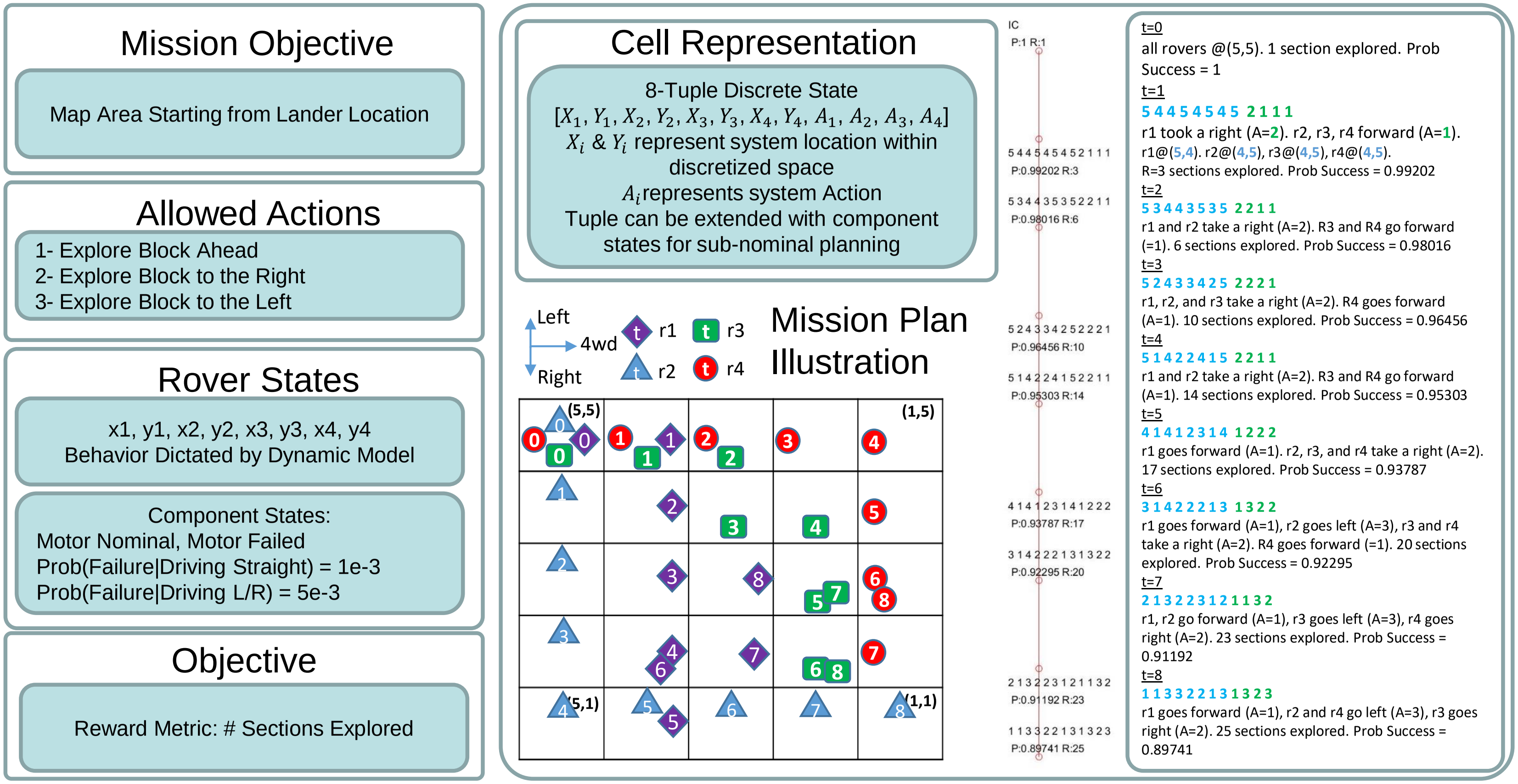
The case study details along with sample results and their interpretation are summarized in Figure 3. Generated plans are based on the system discretization, cell-to-cell map construction from the system simulator, the Markovian Process for incorporating actions and component states, and the search tree parameters. Each node of the search tree. The planner was able to successfully identify mission plans that satisfy the user-defined risk and reward criteria. Each node in the search tree is represented by a tuple that describes system location in the state space (4 rovers x and y positions) and the actions to be taken at that time-step (Forward, Right, Left). Additionally each node contains the evolving risk and reward metric associated with the plan on that branch. The sequence of tuples originating from the initial conditions can be interpreted into a verbose mission plan as seen in the Search Tree Interpretation Section of Figure 3.

5.1. Case Study: Parameter Set 1

The Risk vs. Reward results of case-study 1 are summarized in Table 7. Note that a reward value of 25 indicates full exploration of the 5x5 grid while a reward value of 24 indicates 96% mapping of the area (horizon needs to be expanded more for full exploration).

- 12 plans were generated with a mission time of 800s. 1 of these mission plans completes mapping with 89-90% probability of nominal mission completion. 3 mission plans and 8 mission plans map 24/25 blocks with 89-90% success and 90-91% respectively.
- 50 plans were generated with a mission time of 900s. 10 and 8 of these mission plans complete mapping with an 89-90% and 87-88% probability of nominal mission completion respectively.24 mission plans and 8 mission plans map 24/25 blocks with 88-89% success and 89-90% respectively. 12 mission

It is worth mentioning again that the probability of success stems from a combination of rover action taken and probability of motor degradation. Although not performed in this



Mission Objective

Map Area Starting from Lander Location

Allowed Actions

1- Explore Block Ahead
2- Explore Block to the Right
3- Explore Block to the Left

Rover States

x1, y1, x2, y2, x3, y3, x4, y4
Behavior Dictated by Dynamic Model

Component States:
Motor Nominal, Motor Failed
Prob(Failure|Driving Straight) = 1e-3
Prob(Failure|Driving L/R) = 5e-3

Objective

Reward Metric: # Sections Explored

Cell Representation

8-Tuple Discrete State
$[X_1, Y_1, X_2, Y_2, X_3, Y_3, X_4, Y_4, A_1, A_2, A_3, A_4]$
X_i & Y_i represent system location within discretized space
A_i represents system Action
Tuple can be extended with component states for sub-nominal planning

Mission Plan Illustration

t=0
all rovers @(5,5). 1 section explored. Prob Success = 1
t=1
5 4 4 5 4 5 4 5 2 1 1 1
r1 took a right (A=2). r2, r3, r4 forward (A=1).
r1@(5,4). r2@(4,5), r3@(4,5), r4@(4,5).
R=3 sections explored. Prob Success = 0.99202
t=2
5 3 4 4 3 5 3 5 2 2 1 1
r1 and r2 take a right (A=2). R3 and R4 go forward (=1). 6 sections explored. Prob Success = 0.98016
t=3
5 2 4 3 3 4 2 5 2 2 2 1
r1, r2, and r3 take a right (A=2). R4 goes forward (A=1). 10 sections explored. Prob Success = 0.96456
t=4
5 1 4 2 2 4 1 5 2 2 1 1
r1 and r2 take a right (A=2). R3 and R4 go forward (A=1). 14 sections explored. Prob Success = 0.95303
t=5
4 1 4 1 2 3 1 4 1 2 2 2
r1 goes forward (A=1). r2, r3, and r4 take a right (A=2). 17 sections explored. Prob Success = 0.93787
t=6
3 1 4 2 2 2 1 3 1 3 2 2
r1 goes forward (A=1), r2 goes left (A=3), r3 and r4 take a right (A=2). R4 goes forward (=1). 20 sections explored. Prob Success = 0.92295
t=7
2 1 3 2 2 3 1 2 1 1 3 2
r1, r2 go forward (A=1), r3 goes left (A=3), r4 goes right (A=2). 23 sections explored. Prob Success = 0.91192
t=8
1 1 3 3 2 2 1 3 1 3 2 3
r1 goes forward (A=1), r2 and r4 go left (A=3), r3 goes right (A=2). 25 sections explored. Prob Success = 0.89741

Figure 3. SafeMAP Case Study Results

case study, SafeMAP allows linking motor states to simulated behavior and subsequent plan generation.

The best plan generated by SafeMAP was the one identified over 8 time-steps. This is the mission plan depicted in Figure 3. The search tree for case study 1 is depicted in Figure 4.

Table 7. Case Study 1 Summary of Mission Plan Metrics.

time: 8 Δt	24 cells exp.	25 cells exp.
89-90% Success	3 plans	1 plans
90-91% Success	8 plans	0 plans
time: 9 Δt	24 cells exp.	25 cells exp.
87-88% Success	0 plans	8 plans
88-89% Success	24 plans	10 plans
89-90% Success	8 plans	0 plans

5.2. Results of Parameter Variations

In case study 2 the reward truncation values ϵ_r were relaxed. The results of this case study can be seen in Table 8. The effect of having a relaxed truncation parameter is the search tree retaining non-optimal paths for a few more time steps. Even though valid plans were still generated based on desired planning horizon, the number of plans was less than case study 1.

In case study 3 the branching criterion was made to be more strict. This meant that selection and sorting of scenarios early on was a lot more aggressive. This causes the loss in paths that get more rewarding as the mission time passes. However, a major gain in restricting branching is faster computation time. In time-sensitive scenarios branching should be restricted with greater emphasis placed on smarter filtering and selection of branches to be pursued during the truncation process.

Table 8. Case Study 2 Summary of Mission Plan Metrics: More relaxed rewards.

time: 8 Δt	24 cells exp.	25 cells exp.
89-90% Success	1 plans	0 plans
90-91% Success	0 plans	0 plans
time: 9 Δt	24 cells exp.	25 cells exp.
87-88% Success	17 plans	11 plans
88-89% Success	9 plans	5 plans
89-90% Success	0 plans	0 plans

5.3. Insights and Lessons Learned

There are several key outcomes and insights from the work described in this paper. Notable among these lessons are that computational explosions are inevitable when planning for multi-agent systems. In order to keep the problem feasible and solvable it is important to control the size of the problem state space, and to perform search/planning in an intelligent and informed manner. This can be done by coarser partitioning of the state space, reducing the action space that the planner is responsible for, and reducing rover-rover interactions whenever possible. Under these limitations, increased system autonomy and intelligence is required under clearly defined roles, where the mission planner would be responsible of controlling rover roles rather than dictating a detailed plan for each rover. Note that this increased autonomy does increase the burden of validation and verification. Furthermore, the efficiency of the planner largely depends on the search criteria and problem formulation. An understanding of the system is required when selecting pruning criteria for risk and reward metrics as that can be detrimental to the size of the search space. Loose criteria can lead to a large number of paths to be explored which can cause computational issues. Whereas overly-tight criteria can bias the search algorithms to behaviors that yield high rewards early on but may not be optimal over mission duration.

Table 9. Case Study 3 Summary of Mission Plan Metrics: Stricter branching Criterion.

time: 8 Δt	24 cells exp.	25 cells exp.
89-90% Success	4 plans	1 plans
90-91% Success	8 plans	0 plans
time: 9 Δt	24 cells exp.	25 cells exp.
87-88% Success	0 plans	4 plans
88-89% Success	14 plans	9 plans
89-90% Success	3 plans	0 plans

6. FUTURE WORK

Future work involves a comprehensive integration of rover prognostics and battery models. The inductive search algorithm will also be modified to allow for dynamic re-planning based on feedback from prognostics. Another avenue of interest is the optimization of the MDP algorithm deployed in SafeMAP to obtain deployable results under desirable computational performance (e.g. parallelization, more intelligent search algorithms, etc.). The authors are also working on demonstrating SafeMAP on a higher fidelity case study that involves a wider variety of system environmental states such varying terrain characteristics and obstacles.

REFERENCES

Aldemir, T. (1987). Computer-assisted markov failure modeling of process control systems. *IEEE Transactions on reliability*, *36*(1), 133–144.

Amato, C., Konidaris, G., Kaelbling, L. P., & How, J. P. (2019). Modeling and planning with macro-actions in decentralized pomdps. *Journal of Artificial Intelligence Research*, *64*, 817–859.

Amato, C., & Oliehoek, F. (2015). Scalable planning and learning for multiagent pomdps. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 29).

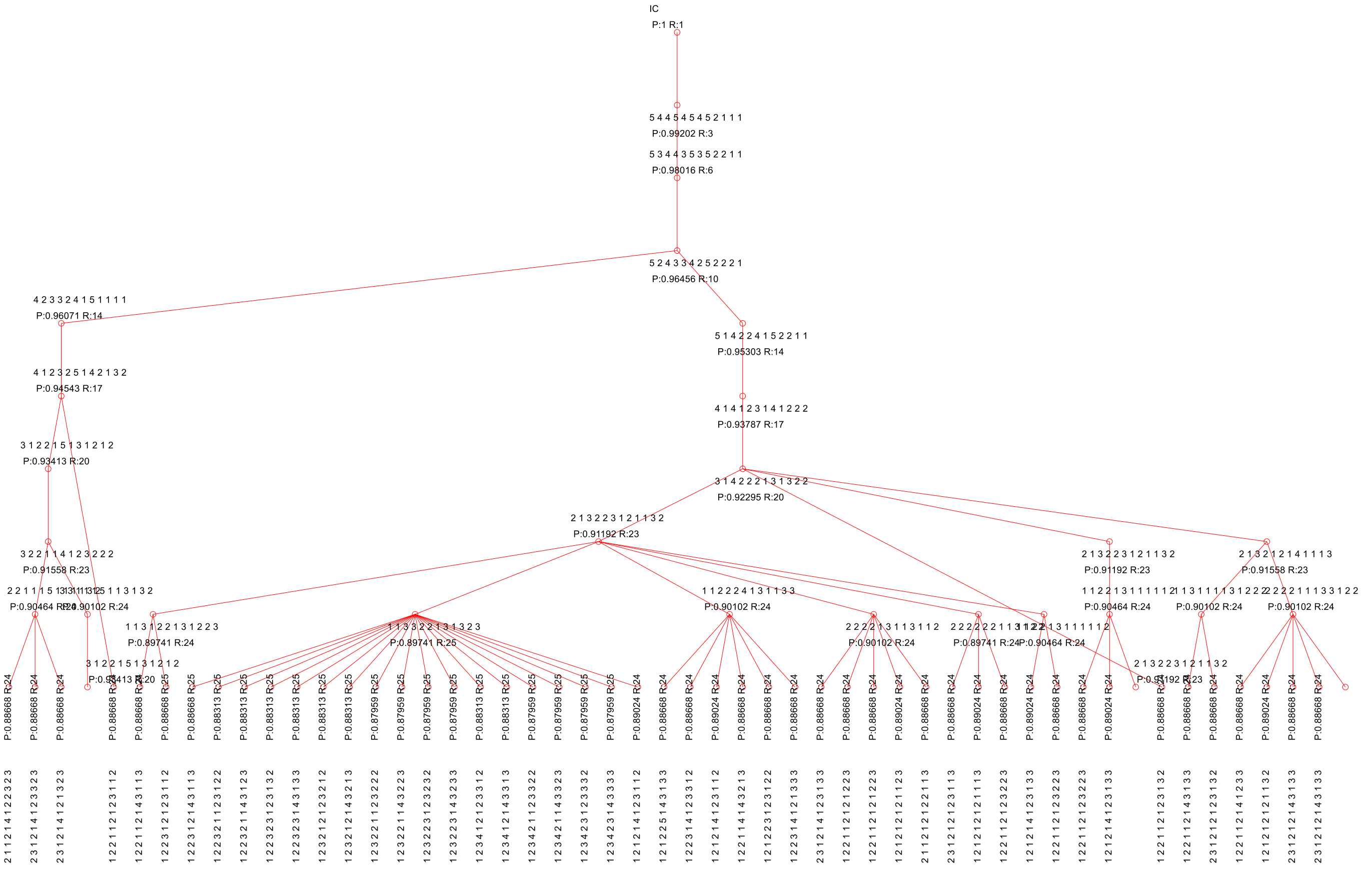


Figure 4. Case Study 1 Search Tree

Bhattacharya, S., Likhachev, M., & Kumar, V. (2010). Multi-agent path planning with multiple tasks and distance constraints. In *2010 ieee international conference on robotics and automation* (pp. 953–959).

Casbeer, D. W., Beard, R. W., McLain, T. W., Li, S.-M., & Mehra, R. K. (2005). Forest fire monitoring with multiple small uavs. In *Proceedings of the 2005, american control conference, 2005.* (pp. 3530–3535).

Clancey, W. J., Sierhuis, M., Alena, R., Crawford, S., Dowding, J., Graham, J., ... van Hoof, R. (2004). Mobile agents: A distributed voice-commanded sensory and robotic system for surface eva assistance. In *Engineering, construction, and operations in challenging environments: Earth and space 2004* (pp. 85–92).

Corbetta, M., Kulkarni, C., Banerjee, P., & Robinson, E. (2021). An uncertainty quantification framework for autonomous system tracking and health monitoring. *International Journal of Prognostics and Health Management*, *12*(3).

Cramer, N., Cellucci, D., Adams, C., Sweet, A., Hejase, M., Frank, J., ... Brown, L. (2021). Design and testing of autonomous distributed space systems.

Faber, N., Nakamura, Y., Alena, R., Mauro, D., Frost, C. R., Bhat, G., & McNair, J. (2014). Heterogeneous spacecraft networks: General concept and case study of a cost-effective, multi-institutional earth observation platform. In *2014 ieee aerospace conference* (pp. 1–16).

Hejase, M., Kurt, A., Aldemir, T., Ozguner, U., Guarro, S., Yau, M. K., & Knudson, M. (2018). Dynamic probabilistic risk assessment of unmanned aircraft adaptive flight control systems. In *2018 aiaa information systems-aiaa infotech@ aerospace* (p. 1982).

Hejase, M., Kurt, A., Aldemir, T., Özgüner, Ü., Guarro, S. B., Yau, M. K., & Knudson, M. D. (2018). Quantitative and risk-based framework for unmanned aircraft control system assurance. *Journal of Aerospace Information Systems*, *15*(2), 57–71.

Kakish, Z. M., RodrÃguez-Lera, F., Bischel, D., Mosquera, A., Boumghar, R., Kaczmarek, S., ... Galanche, J. (2019). Open-source ai assistant for cooperative multi-agent systems for lunar prospecting missions. In *8th european conference for aeronautics and space sciences (eucass).*

Kalita, H., & Thangavelautham, J. (2020). Lunar cubesat lander to explore mare tranquilitatis pit. In *Aiaa scitech 2020 forum* (p. 2163).

Kalita, H., & Thangavelautham, J. (2021). Strategies for deploying a sensor network to explore planetary lava tubes. *Sensors*, *21*(18), 6203.

Kobayashi, T., Fujiwara, Y., Yamakawa, J., Yasufuku, N., & Omine, K. (2010). Mobility performance of a rigid wheel in low gravity environments. *Journal of Terramechanics*, *47*(4), 261–274.

Norheim, J. J. (2018). *Path planning for human planetary surface exploration in rough terrain* (Unpublished doctoral dissertation). Massachusetts Institute of Technology.

Rossi, F., Vaquero, T. S., Sanchez-Net, M., da Silva, M. S., & Vander Hook, J. (2020). The pluggable distributed resource allocator (pdra): a middleware for distributed computing in mobile robotic networks. In *2020 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 4337–4344).

Yang, J., & Aldemir, T. (2016). An algorithm for the computationally efficient deductive implementation of the markov/cell-to-cell-mapping technique for risk significant scenario identification. *Reliability Engineering & System Safety*, *145*, 1–8.

Biographies

Mohammad Hejase is a research scientist with the Robust Software Engineering Group in the Intelligent Systems Division at the NASA Ames Research Center, CA. His research interests are dynamics probabilistic risk assessment of intelligent unmanned systems, and decision making for multi-agent systems under uncertainty.

Portia Banerjee works as a research engineer with KBR. at NASA Ames Research Center, CA. Her research interests include statistical signal processing, image processing, data mining, uncertainty management and reliability analysis focusing towards diagnostic and prognostic applications in structures and autonomous systems. She is a member of the AIAA, ASNT, IEEE, PHM and ASME societies and is a member of the Executive Committee of ASME NDE Diagnosis and Prognosis Division.