Risk-Averse Planning and Plan Assessment for Marine Robots *

Mahya Mohammadi Kashani[1], Tobias John[2], Jeremy P. Coffelt[3], Einar Broch Johnsen[2] and Andrzej Wąsowski[1]

***Abstract*— Autonomous Underwater Vehicles (AUVs) need to operate for days without human intervention and thus must be able to do efficient and reliable task planning. Unfortunately, efficient task planning requires deliberately abstract domain models (for scalability reasons), which in practice leads to plans that might be unreliable or under performing in practice. An optimal abstract plan may turn out suboptimal or unreliable during physical execution. To overcome this, we introduce a method that first generates a selection of diverse high-level plans and then assesses them in a low-level simulation to select the optimal and most reliable candidate. We evaluate the method using a realistic underwater robot simulation, estimating the risk metrics for different scenarios, demonstrating feasibility and effectiveness of the approach.**

I. Introduction

Autonomous Underwater Vehicles (AUVs) support unmanned operations, such as gathering data for scientific and commercial purposes. The promise of autonomy makes AUVs ideal for inspection tasks in harsh environments, as operating in these conditions poses a high risk to human workers. However the high cost of vehicles, operation in the vicinity of high-value assets, and the risk of causing natural disasters introduce a requirement of reliable and safe autonomy [1].

Traditionally, adaptive planning has been a key technology to achieve autonomy. Mainstream work on planning optimizes for expected rewards. However, ignoring the probability of achieving the expected reward (the *uncertainty*) seems overly optimistic for high risk operations. Although the expected reward might be high, an extremely different outcome may have similar or higher probabilities than the expectation.

Example 1: Consider a marine inspection scenario where the task is to safely inspect multiple sub-sea installations, e.g. the vertical small tanks in Fig. 1. During the mission, an AUV needs to choose between five paths $P_1, \ldots, P_5$. A typical planner here finds that plan P_1 is the *shortest* and selects it. If we simulate the different plans, we see that P_1 leads in general to a *smaller mean execution time* than the other plans, but the *variance of the execution time* is high, i.e., there is always the risk that the execution time is way higher than expected beforehand and possibly higher than the execution time of P_4. Arguably, P_4 might be the safest plan, with a decent expected execution time and a very small variance.

Fig. 1: An underwater scene visualized in Gazebo

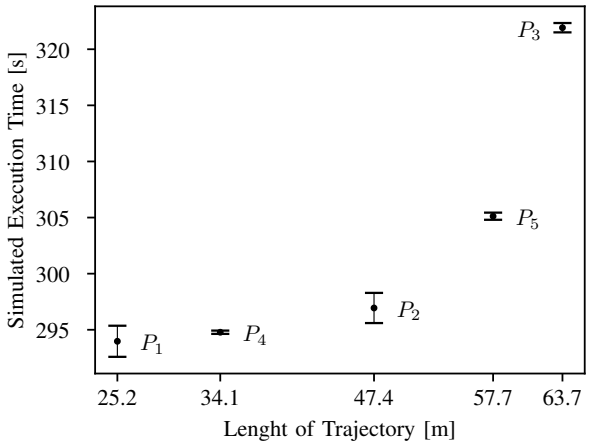


Fig. 2: Comparison of simulated execution times for different plans. Plan IDs refer to those in Table I.

Two key problems with high-level planning lead to situations like in Fig. 2. First, high-level planning deliberately operates under reduced information. Second, the uncertainty of obtaining the reward is typically ignored. Although handling uncertainty in planning systems seems necessary in dynamic environments, most existing research on underwater mission planning (e.g., on re-planning [2], [3] or temporal planning [4]) defines planning problems as optimization against the mean reward objective such as time or energy [5].

To address these problems, we propose a method that enables marine robots to select reliable plans by assessing high-level plan candidates using a low-level simulation model and statistical assessment of uncertainty. The goal of this work is to increase the robustness of planning for autonomous underwater robots in dilemma situations. We contribute a framework with the following distinct features:

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 956200 REMARO.

[1]Mahya Mohammadi Kashani and Andrzej Wąsowski are with Computer Science Department, IT University of Copenhagen, 2300 Copenhagen, Danmark mahmo@itu.dk, wasowski@itu.dk

[2]Tobias John and Einar Broch Johnsen are with the Department of Informatics, University of Oslo, Oslo, Norway tobiajoh@ifi.uio.no, einarj@ifi.uio.no

[3]Jeremy Coffelt is with the research and technology group, ROSENXT, Bremen, Germany jcoffelt@rosen-nxt.com

(i) Generation of probabilistic planning problems from sonar sensor data,
(ii) Transformation of planning problems to generate a selection of candidate plans that involve different degrees of risk-averseness,
(iii) Risk evaluation for candidate plans using realistic underwater simulation, and
(iv) Use of several risk metrics for selecting plans.

II. Related Work

We identified several lines of research on planning for AUVs that are related to our work: (A) modelling complex environments, (B) handling of uncertainties of outcomes of actions, (C) explaining generated plans and (D) risk-averse planning.

(A) The planning time depends significantly on how the planning domain is formulated. Steenstra shows how to compensate for environmental challenges, such as location uncertainty and communication limitations [6], using the Problem Domain Definition Language (PDDL) to model an abstract-level planning problem for Lightweight Autonomous Unmanned Vehicles [7]. In his work, a temporal deterministic planner OPTIC [4] consistently produces high-quality plans with the lowest cost, however at a prohibitively high computation cost. To overcome this, Steenstra translates the planning domain into Event-B [8], reducing the planning time. We address this issue differently, by using a simpler abstract planning model and statistical tests to select between several best plans.

(B) According to Cashmore et al., the efficiency of long-horizon mission planning is better and more reliable if the uncertainty is neglected in favor of re-planning [9], [10]. We argue that neglecting uncertainty is not generally useful: re-planning happens too late, after the robot has already committed to a (possibly) unreliable plan. Replanning is a very good plan adjustment strategy, orthogonal to what we propose—choosing a reliable plan to begin with.

(C) Carreno et al. support investigation of multiple planning choices at the planning and execution time, although finding appropriate explainable metric(s) still needs to be investigated [11]. We are not concerned with explanations of plans here, but with the robot itself assessing the plans using more faithful models than the planner, and selecting between several options.

(D) There were prior attempts to assess generated policies using variance of cumulative reward [12]. Unfortunately, the risk-sensitivity is not compatible with classical planning algorithms. Our approach of risk-averse planning is inspired by the work of Koening and Simmons [13] that explore the spectrum of risk-sensitivity from risk-seeking to risk-averse. The MDP emphasizing risk sensitivity to maximize expected exponential utility aligns with a robust MDP aimed at maximizing the worst-case criterion [14]. We bound the risk factor to a small interval to make the problem more compatible with Moldovan's result [14], and integrate risk-averse planning into a proof-of-concept implementation for a marine robot.

III. A Method for Risk-Averse Planning

A. Framework

Both high-level task planning and low-level path planning are crucial for success of marine missions. A mission comprises a high-level plan and sub-plans generated by low-level (path) planners. Such low-level plans can be used for docking (from or to the docking station), for generating inspection trajectories around objects of interest (such as the fuel tanks shown in Fig. 1), and for following waypoints selected by a robot operator. The goal of high-level planning is to synthesize a plan to reach a state where the desired goals are achieved, given abstract descriptions of the initial state of the world and a set of possible actions. Symbolic AI planning is advantageous if the problem can be described declaratively and non-trivial domain knowledge exists and is relevant. However, symbolic planning necessarily, by-design, operates on highly approximate models of reality (abstractions), which introduces a significant performance gap with reality. We address this by generating multiple abstract plans and evaluating their performance statistically at runtime, using a simulator, which has a much less abstract view of the world than the high-level planner.

Fig. 3 summarizes the method. We assume that a probabilistic model of the current situation of the robot is available, extracted from sensor data, e.g. sonar data. This includes the possible actions of the robot and the goal of the mission, e.g. inspect the underwater infrastructure of interest. We generate different plans (with different sensitivity to risk) and use a (low-level) path planner to refine the high-level actions of the generated plans. Afterwards, we evaluate the performance of the refined plans using simulation. The generated evaluation for the different plans allows us to choose the plan that balances the involved uncertainty and the expected performance.

B. Plan Generation

We model high-level planning problems in *Probabilistic Programming Domain Definition Language* (PPDDL), a representation of *Markov Decision Processes* (MDPs) [15]. An MDP is a 5-tuple $\mathcal{M} = (S, A, \mathbf{P}, s_0, \mathrm{R})$ where S is a finite set of states, A a finite set of actions, $\mathbf{P} : S \times A \times S \to [0, 1]$ the transition function, $s_0 \in S$ the initial state, and $\mathrm{R} : S \mapsto \mathbb{R}_{\geq 0}$ the reward. A PPDDL problem specifies a set of goal states $G \subseteq S$ that should be reached.

Example 2: Fig. 4 shows an example of a planning model MDP. A robot can navigate between three waypoints and its mission is to inspect all tanks to make sure that there is no leakage. A non-deterministic choice (an action) determines which waypoint the robot inspects next. Some transitions involve risks, i.e. the AUV may collide with infrastructure with some probability. The goal is to reach the waypoint "final" after having performed an inspection at the waypoint "tank." The cost is the length of the generated plan. Note that this cost is deliberately abstract, not only for simplicity of the example, but to emphasize that the high-level planner always operates on an abstraction.

Given a plan $\pi : S \mapsto A$, non-determinism in the MDP can be resolved to obtain the induced *Markov Chain (MC)*

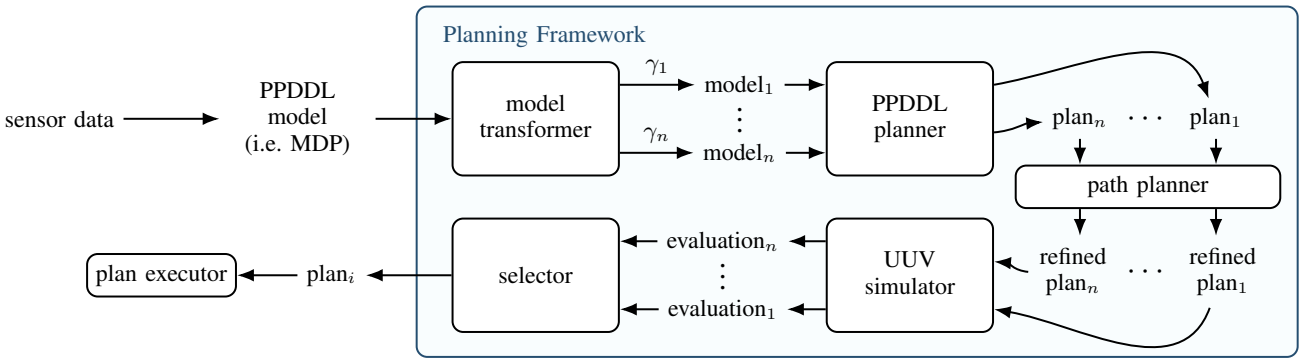


Fig. 3: Flow diagram of proposed method.

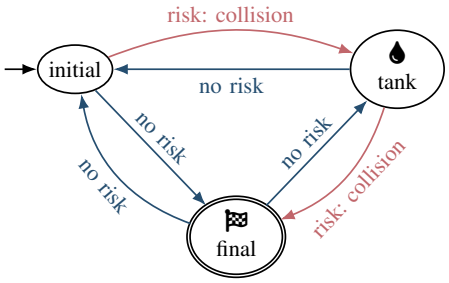


Fig. 4: Example scenario described by an MDP. It includes three waypoints, which are the critical ones for the mission.

$\mathcal{M}_\pi = (S, \mathbf{P}_\pi, s_0, \mathrm{R})$ where for each pair of states s, s' the transition probability is defined by $\mathbf{P}_\pi(s, s') = \mathbf{P}(s, \pi(s), s')$. A *history* h is a finite sequence of states $h = (s_0, s_1, \ldots, s_n)$. Define a history's *probability* by $\mathbf{P}(h) = \prod_{i=0}^{n-1} \mathbf{P}_\pi(s_i, s_{i+1})$ and its *reward* by $\mathrm{R}(h) = \sum_{i=0}^{n-1} \mathrm{R}(s_i)$. For goal states $G \subseteq S$, the corresponding reward of the Markov Chain, $\mathrm{R}_\mathcal{M}(G)$, is a discrete random variable with:

$$\Pr\left(\mathrm{R}_\mathcal{M}(G) = x\right) = \sum_{h=(s_0,\ldots,s_n)} \mathbf{P}(h) \tag{1}$$

where the summation ranges over all paths h such that $s_n \in G$, $s_0, \ldots, s_{n-1} \notin G$, and $\mathrm{R}(h) = x$

For a given MDP model, we generate a set of different *candidate high-level plans*. Our method is agnostic to the algorithm to compute the different candidates, but in this paper, we use Koenig and Simmons' approach [13] to generate risk-averse plans with a non-linear utility function for cumulative rewards. This function values a difference between high rewards less than the same difference between smaller rewards; i.e., the cumulative reward of a few "best-case" paths has smaller impact on plan selection than many paths with a decent cumulative reward. The utility function depends on a parameter γ, which balances expected reward and involved risk. The main advantage of this construction is that we obtain a utility function for the cumulative reward for risk-sensitive planning by only making local changes; i.e., the probabilities of the MDP's transitions are replaced by values that take the probability, immediate reward and the risk sensitivity into account.

Definition 1 (Transformed Transition System): Given an MDP $\mathcal{M} = (S, A, \mathbf{P}, s_0, R)$ and a parameter $0 < \gamma < 1$, we define the transformed transition system $\mathcal{M}^\gamma = (S, A, \mathbf{P}^\gamma, s_0)$ where $\mathbf{P}^\gamma : S \times A \times S \to [-1, 0]$ and:

$$\mathbf{P}^\gamma(s, a, s') = \mathbf{P}(s, a, s') \cdot \left(-\gamma^{R(s)}\right) \tag{2}$$

Although the resulting transition system is not an MDP, standard algorithms can be used to find the plan with maximal utility by maximizing the "pseudo-probability" $\mathbf{P}^\gamma$ to reach a goal state [13]. The transformation described in Def. 1 can be done at the level of the actions in PPDDL domains.

We call a PPDDL planner for different random values of γ from the interval $(0, 1)$ to generate the optimal plan for each value. Each optimal plan is simply the plan that maximizes the probability of reaching one of the goal states in the transformed model. For different values of γ, we get different plans, with different levels of risk-aversion. A value of $\gamma = 1$ means that the planner only tries to optimize expected cost, while a value of $\gamma = 0$ means that the planner tries to minimize the cost that can be guaranteed. These plans form our set of candidate plans.

Crucially, plans are always generated with respect to a limited number of risk metrics, as planners can only handle a limited number of objectives at a time. Most often, as in our case, only one parameter is considered (here γ). This risk measurement might—but need not—correlate to other risk metrics, e.g., variance or entropy of the cumulative reward. Therefore, we propose to separate the assessment of candidate plans against risk measurements of interest from the plan generation.

C. Plan Evaluation

To select the best plan, we use metrics based on the probability distribution $D = \mathrm{R}_{\mathcal{M}_\pi}(G)$ of the reward to evaluate each of the candidate plans in the Bayesian spirit. One possibility is to use Monte Carlo simulation in the MDP of the planning problem. This, however, is typically insufficient, and definitely too unreliable for underwater robots. An MDP model for behavior abstracts away too much information about the physical environment. For example, our model neglects critical considerations such as unpredictable water currents, GPS-denied vehicle localization, or noisy acoustic sensors. In order to overcome this limitation, we perform the plan assessment

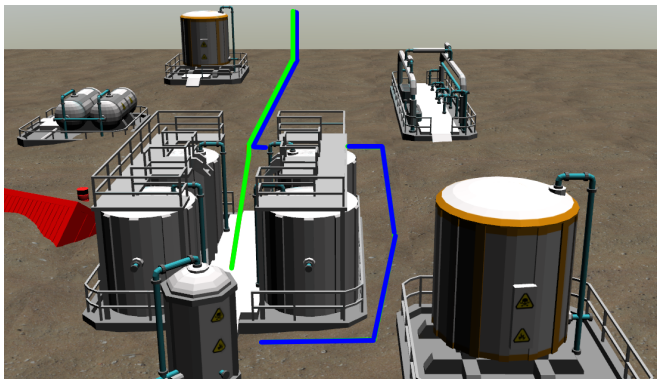


Fig. 5: Blue line: a plan fragment following the safest path as returned by our method. Green line: a risky path suggested by a high-level risk-neutral planner

not in the MDP, but in a more detailed model, using an *underwater physics simulator* that provides increased realism. This involves using low-level planners to refine the actions in our generated high-level plan. In order to get a good estimation of the risk associated with the plans, each generated plan needs to be simulated multiple times. The simulator uses random variables for environment conditions, e.g. underwater currents or water visibility, to obtain a representative sample of runs.

We use different metrics to assess the current and anticipated risks for the above mathematical model.

1) The *expected reward* $\mathbb{E}[D]$ is typically a primary concern when selecting a plan. Even in risk-averse settings, a decent expected reward is required.
2) The *variance* of the reward $\mathbb{E}[D - \mathbb{E}[D]^2]$ is an often used measurement for risk [16]. The higher the variance, the more risk is involved.
3) The *entropy* of the reward $-\sum_{x \in R} \Pr(D = x) \log_2(\Pr(D = x))$ is another common metric used to measure the uncertainty or "surprise" of a reward.

After simulating the candidate plans and estimating the risk metrics of interest, we select the plan that balances the different metrics best. Afterwards, we hand over the selected plan to the plan executor, which runs the plan on the actual robot.

IV. Experimental Results

The goal of the experiment is to demonstrate that our framework can generate a reliable plan for an AUV given an inspection mission, even if this plan is not optimal with respect to the expected cost in the high-level planning model. We used a scenario where the underwater robot needs to inspect an area of interest while avoiding risky places. Our scenarios always allow for several different strategies or plans to accomplish the mission.

A. Evaluation Scenario

The robot starts somewhere in the distance on the top of Fig. 1. The goal is to navigate and inspect the tank on the bottom of the illustration. Fig. 5 shows two plans that demonstrate the trade-off between expected cost and risk. We show the safest plan, as proposed by our method, with a blue line, and a shorter plan with a green line, which requires the robot to navigate through the narrow path between the quad tanks. The longer plan, where the marine robot navigates around the large vertical tank, is safer and thus preferred. To avoid collisions, the robot needs to moves slowly close to the tanks and it accelerates, when there is enough space to do so safely. Therefore, the safer path would not result in a large delay.

We use plan length and the risk of collision with the infrastructure as cost metrics (negative reward). Navigating close to objects, such as navigating through a narrow gap between tanks as in Fig. 1, can be part of a plan with low cost, but with high risk.

We created a PPDDL model of the scenario, fixing the goal of the mission and the actions that the AUV can perform. The actions not only involve moving from one waypoint to another but also inspecting objects at 50 waypoints using a helical trajectory around the object. The details of the actions are refined by low-level path planners after the high-level plan is synthesized. To keep the state space as small as possible, we do not use all waypoints for the high-level planner, as the low-level path planner does, but we only consider critical waypoints. Such points are the ones close to one of the fuel tanks or between two narrow infrastructure objects.

We use six different inspection points including in front of the *large vertical tank*, inside of the *platform*, in front of the *tank pairs*, in front of and behind the *quad tank*, next to the *large vertical tank clone* and in front of the *small vertical tank*. We select four waypoints as critical: (1) right of the small vertical tank, (2) in front of the narrow path in the quad tank, (3) between the quad tank and the largest tank and (4) left of the small vertical tank. The velocity of robot is decreased in vicinity of critical waypoints to avoid collisions.

To create risk-aware variants of the model, we sample risk factors γ with a random uniform distribution over $[0.4, 1)$ and transform the mode as in Def. 1. This interval provides a broad spectrum of plans, while still ensuring that the plans optimize the expected reward sufficiently. We use an off-the-shelf PPDDL planner to generate plans for the generated transformed models. Afterwards, we refine the plans by instantiating the actions using low-level planners for path planning. After running the refined plans in Unmanned Underwater Vehicle (UUV) simulations, we estimate the metrics expected value, variance and entropy for the execution time. We select the plan based on these metrics.

At runtime, during plan assessment in simulation, a collision detector monitors the distance between the robot and the infrastructure objects and logs incidents.

B. Design and Implementation

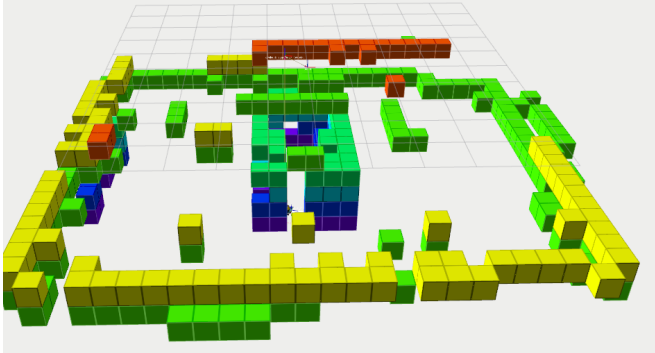
We execute the experiments on a laptop with 2.60GHz Intel Core CPU (i7-10750H) and 32GB of RAM, running Ubuntu 20.04. The implementation is based on ROS1 Noetic, Gazebo 11, and a modified version of UUV simulator.[1] The source code to reproduce our experiments is available online.[2]

[1]https://github.com/mahyamkashani/uuv_simulator
[2]https://github.com/remaro-network/risk-averse_planning

ID	Plan schema	planning time [s]	high-level length	low-level length	assessment (exec. time) mean [s]	variance	entropy
P_1	lg tank → quad tank → sm tank	0.09	3	25.23	293	1.92	1.6
P_2	lg tank → tank pairs → platform → quad tank → sm tank	0.11	5	47.48	297	1.8	1.6
P_3	lg tank → tank pairs → lg tank back → platform → quad tank → sm tank	0.15	6	63.73	322	0.17	1.6
P_4	lg tank → quad tank → lg tank clone → sm tank	0.07	4	34.19	294	0.02	0.2
P_5	lg tank → tank pairs → lg tank back → quad tank → sm tank	0.11	5	57.73	305	0.1	1.6

TABLE I: Generated plans using risk-averse planning. We have 5 types of infrastructure in the simulation. The names "lg tank" and "sm tank" stand for "large vertical tank" and "small vertical tank." The high-level length is measured in the number of steps (plan depth), and the low-level length is the total distance between all waypoints in meters. The vertical bar separates the high-level planning from plan execution and assessment. The IDs are the same as in Fig. 2

Fig. 6: A seabed OctoMap [17] for a confined place, like a basin, using a low resolution (1000 samples with 0.5 resolution) sufficient for high-level planning. Each Voxel represents a probability of occupancy.



We created the scenarios for gas and oil infrastructure inspection by using public assets from the DAVE project.[3] The 3D models can be obtained from the TurboSquid website.[4]

We estimate the occupancy probability for the discretized (high-level) map using OctoMap [17], calculating the probability that a section of the environment is occupied using simulated forward multibeam p900 sonar data.[5] We tried this procedure with different resolutions starting from 0.05 up to 0.5 resolution. We settled on the latter (low-resolution), judging that this is suitable for high-level task abstraction. Fig. 6 shows an example octomap. Each color represent a probability between 0 and 1. The values of the cubes (voxels) are used to create the state space of our planning problem.

We use a modified version of safe-planner to generate different risk-averse plans [18]. The modifications comprise incorporating the risk parameter γ. The output for each plan generation is a JSON file containing the plan and a statistics file that documents the performance of the produced plan. LIPBInterpolator (Linear Parabolic Interpolator) is used as a low-level path planner,[6] to refine the actions of the high-level plans with intermediate waypoints.

We execute each resulting plan 10 times to evaluate its performance under slight randomization. We perturb positions of the vertical small tank in the simulation environment with random noise to mimic uncertainty in a physical environment, and use it to estimate the risk metrics of the plan (expected reward, variance, and entropy of the execution time).

[3]https://field-robotics-lab.github.io/dave.doc/

[4]https://www.turbosquid.com/3d-models/3d-fuel-tank-1443266

[5]https://github.com/Bluerov2

[6]https://uuvsimulator.github.io/packages/uuv_simulator/docs/python_api/uuv_trajectory_generator/lipbinterpolator

C. Results

Table I summarizes the generated plans and sub-plans for one of our evaluation scenarios (scenario ID 3 with five critical states in Table II). Interestingly, the variation in length at the low-level is lower than at the high-level. While the former only differ by about 10%, the latter can be 100% longer, see for example P_1 and P_3. This illustrates how typically, and deliberately, abstraction hides information from high-level models; underlining that a high-level planner always operates under imperfect information. This is an inherent problem. In practice it is unrealistic to expect that the high-level model will be a sound abstraction from the perspective of all metrics of interest. This also indicates that it might be useful to consider several reasonable plans and evaluate them at lower level of abstraction. (Note that optimal planning at the low-level of abstraction is too expensive.)

While the length of the low-level plans does not differ much, the simulation-based assessment shows that the actual execution times still can differ significantly (cf. Fig. 2 and the mean time in Table I). Plan P_1, the shortest in the abstract domain, turns out to be relatively slow in a more detailed simulation with the real robot controller. The high variance also indicates unreliable execution. It seems more desirable to execute plan P_4 in the physical environment, since this plan has both low and predictable running time. In extreme cases, not observed in this experiment, the best plan according to the mean reward might be unusable, because its high variance makes it less desirable than another plan with worse mean but smaller variance. In our scenario, the entropy is the same for all the candidate plans. This is however not always the case. Entropy is a more useful assessment metric, in scenarios when distribution of rewards is multimodal.

Table II summarizes properties of optimal plans generated for different evaluation scenarios. The scenarios differ in the number of critical states in the high-level planning problem; and increase in complexity with the number of critical states.

We find that the method scales—all scenarios are solvable; i.e. we can generate a plan to reach the goal of the mission. In all scenarios, the path planner generates between 109 and 129 waypoints, depending on the plan length. The planning time

and the length of the found plan increase with the number of critical states as expected—the planner needs to search through a larger state space. The crucial aspect of this method is that when the planner stops scaling, one can abstract the domain model more aggressively (to make it scale), and use simulation-based assessment to select reasonable plans. Notably, the risk parameter is often smaller than one. This indicates, that the found plan, which is the safest for the scenario, is not the one that minimizes cost, i.e. plan length, but one finds a trade-off between cost and risk aversion. This shows that the method is making non-trivial choices, comparing to a planner that only optimizes mean cost.

We have experimented with increasing branching in the planning model. Branching is typical for ordered tasks like manipulation. Since our plan generation is based on Breadth-First Search, the planning time does increase exponentially. This can be improved by switching to a different plan generation method, while keeping the same simulation-based plan assessment. In pipeline inspection tasks, which is the main use case in our project, we do not experience high branching.

V. Conclusion and Outlook

We have presented a risk-averse planning method based on generating multiple plans and separately, performing their simulation-based assessment. We implemented and evaluated the method, showing that it scales for simple inspection tasks in the underwater domain. The evaluation has demonstrated the effectiveness of generating diverse plans, the effectiveness of simulation-based assessment in reducing the information-gap between high-level and low-level dynamics, and the effectiveness of the method overall to select the safest plan from a small list of candidates.

Notably, the separation of planning from assessment is applicable to different plan generation methods, not necessary ours. It can be used with a diversity of risk and performance metrics, such as (i) *reward-bounded probability*, the probability that the reward falls below a given bound, designed to estimate how often bad runs occur, (ii) *value at risk*, the maximum reward expected given a time horizon and a confidence level, capturing tail risk, and (iii) *expected shortfall*, the average of the worst percentile of losses. It is not required that the high-level planning and the low-level assessment use the same performance objective. One can, for instance, plan for minimal distance traveled, but assess against the simulated power consumption. In the future, we intend to exploit these flexibility in planning underwater inspection scenarios.

TABLE II: Different scenarios for risk-averse planning, between 109 and 122 way points. All scenarios are solvable and the method produces the safest plan.

	Problem properties		**Properties of the safest plan**		
ID	**depth [state]**	**#crit. states**	**length**	**risk (γ)**	**planning time [s]**
1	3	3	5	0.95	0.05
2	4	4	5	0.74	0.06
3	5	5	6	0.40	0.11
4	6	6	17	0.57	0.10
5	10	5	13	0.67	0.11
6	15	5	13	0.49	0.14
7	20	6	17	0.84	0.08
8	30	10	52	0.58	0.13
9	40	15	66	0.84	0.16
10	50	15	86	0.96	0.21
11	60	15	106	0.97	0.29
12	70	15	126	0.95	0.40
13	80	15	146	0.59	0.54
14	90	15	166	0.57	0.93
15	90	25	166	0.42	0.98
16	90	35	166	0.75	1.04

References

[1] X. Chen, N. Bose, M. Brito, F. Khan, B. Thanyamanta, and T. Zou, "A review of risk analysis research for the operations of autonomous underwater vehicles," *Reliability Engineering & System Safety*, vol. 216, p. 108011, 2021.

[2] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302, 2001.

[3] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[4] J. Benton, A. Coles, and A. Coles, "Temporal planning with preferences and time-dependent continuous costs," in *Twenty-Second International Conference on Automated Planning and Scheduling*, 2012.

[5] M. Cashmore, M. Fox, T. Larkworthy, D. Long, and D. Magazzeni, "AUV mission control via temporal planning," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014.

[6] L. Steenstra, "PDDL-based task planning of survey missions for autonomous underwater vehicles: A generic planning system, taking into account location uncertainty and environmental properties," Master's thesis, TU Delft, 2019.

[7] A. Sousa, L. Madureira, J. Coelho, J. Pinto, J. Pereira, J. B. Sousa, and P. Dias, "LAUV: The man-portable autonomous underwater vehicle," *IFAC Proceedings Volumes*, vol. 45, no. 5, pp. 268–274, 2012.

[8] F. Fourati, M. T. Bhiri, and R. Robbana, "Verification and validation of PDDL descriptions using Event-B formal method," in *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)*. IEEE, 2016, pp. 770–776.

[9] M. Cashmore, M. Fox, D. Long, B. C. Ridder, and D. Magazzeni, "Opportunistic planning for increased plan utility," in *The 4th ICAPS Workshop on Planning and Robotics (PlanRob 2016)*, 2016.

[10] M. Cashmore, M. Fox, D. Long, D. Magazzeni, and B. Ridder, "Opportunistic planning in autonomous underwater missions," *IEEE Trans. on Automation Science and Eng.*, vol. 15, no. 2, 2017.

[11] Y. Carreno, A. Lindsay, and R. Petrick, "Explaining temporal plans with incomplete knowledge and sensing information," in *ICAPS 2021 Workshop on Explainable AI Planning*, 2021.

[12] M. Sato, H. Kimura, and S. Kobayashi, "TD algorithm for the variance of return and mean-variance reinforcement learning," *Trans. Jap. Soc. for Artificial Intelligence*, vol. 16, no. 3, pp. 353–362, 2001.

[13] S. Koenig and R. G. Simmons, "How to make probabilistic planners risk-sensitive (without altering anything)," in *Proc. 2nd Intl. Conf. on Artificial Intelligence Planning Systems*. AAAI, 1994.

[14] T. Moldovan and P. Abbeel, "Risk aversion in Markov decision processes via near optimal Chernoff bounds," *Advances in neural information processing systems*, vol. 25, 2012.

[15] H. L. S. Younes and M. L. Littman, "PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects," Carnegie Mellon University, Tech. Rep. CMU-CS-04-167, 2004.

[16] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *Journal of Artificial Intelligence Research*, vol. 24, pp. 81–108, 2005.

[17] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[18] V. Mokhtari, A. S. Sathya, N. Tsiogkas, and W. Decré, "Safe-Planner: A single-outcome replanner for computing strong cyclic policies in fully observable non-deterministic domains," in *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE, 2021, pp. 974–981.