Intelligent Battery Management for aquatic drones based on Task Difficulty driven POMDPs

Alberto Castellini*, Jason J. Blum, Domenico D. Bloisi, and Alessandro Farinelli

Verona University, Department of Computer Science,
Strada Le Grazie 15, 37134 Verona, Italy,
{name.surname}@univr.it

Abstract. Autonomous drones typically have limited battery capacity, which presents a major challenge for persistent, long-term deployments. In the context of water quality monitoring, aquatic drones must navigate rivers and lakes to collect real-time data, where the battery consumption is heavily influenced by dynamic environmental factors such as flowing current and wind. Intelligent battery management is a requirement for the success of these missions. We propose a formalization of the battery management problem in terms of Partially Observable Markov Decision Processes (POMDPs). We model the problem as a POMDP where the agent modulates the energy provided to its propellers. The drone can estimate the "difficulty" to traverse an interval by observing the environment. An important aspect of this formalization is the prediction of future intervals' difficulty values based on current observations by exploiting a priori geometric information, which we call Task Difficulty Propagation (TDP). We investigate variations of this approach and analyze related performance.

1 Introduction

Autonomous aquatic drones are increasingly used for water monitoring. A key factor for the success of data acquisition campaigns is *mission awareness*, which is composed of three main elements [2]: knowledge of mission objectives, internal self-situational awareness, and external self-situational awareness. Due to the limited energy on-board and the requirements for long-term autonomy and reliability, the ability to forecast the *mission-specific* energy consumption is necessary to complete missions reliably, smoothly, and efficiently. In this work we focus on battery management, which is an aspect of self-situational awareness.

Existing approaches for estimating the battery consumption in autonomous systems can be divided into two main categories, namely decision [3,8,2] and predictive models [4,11,9]. All these approaches deal with different aspects of a common problem: making optimal decisions for battery management in uncertain environments. In this paper, we propose a methodology based on Partially Observable Markov Decision Processes (POMDPs) [6], a decisional model that combines the strength of Hidden Markov Models with that of Markov Decision Process by capturing both dynamics that depend on unobserved states and effects of decisions over time.

We apply our method to the following case study: given *i)* a path made of four perpendicular segments (Figure 1.c), in which each segment has a specific degree of

difficulty (i.e., a measure of the forward speed achieved for a given amount of energy consumption) and *ii)* an initial amount of battery energy, the agent has to make decisions about the amount of engine power needed at each time instant to minimize the time spent to reach the end of the path and maximize the probability to complete the mission before battery depletion. To this end POMDPs are extended with a novel simulation strategy called *Task Difficulty Propagation* (TDP) which is based on Markov Random Fields (MRF), and allows to represent the dependencies between segment difficulties and to propagate this information for improving the belief over the difficulties in the entire path. First experiments show promising enhancements to mission performance. Future developments aim at using the method on the INTCATCH [5] drones (Fig. 1.a).

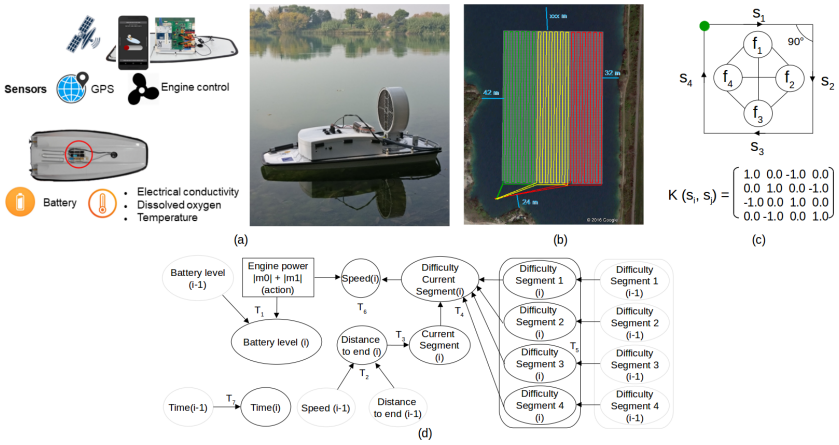

Fig. 1: a) Intcatch system for autonomous monitoring of catchments; b) Real path from a water monitoring campaign; c) Shape and cosine similarity matrices for a square path; d) POMDP model for battery management in water monitoring drones.

2 Methods

A POMDP is a tuple $(S, A, O, T, Z, R, \gamma)$, where S is a finite set of *states*, A is a finite set of *actions*, Z is a finite set of *observations*, T: $S \times A \rightarrow \Pi(S)$ is the *state-transition model*, O: $S \times A \rightarrow \Pi(Z)$ is the *observation model*, R: $S \times A \rightarrow \mathbb{R}$ is the *reward function* and $\gamma \in [0, 1)$ is the *discount factor*.

The goal of an agent based on POMDPs is to maximize its expected total reward $E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$, by choosing the best action a_t in each state s_t at time t.

Figure 1.d shows the proposed POMDP for battery management as a graph, where states are represented by circles, actions by rectangles and conditional dependencies between states by arrows. The states are: *i)* battery voltage interval $B \in \{b_0, b_1, b_2, b_3\}$, where b_0 represents empty battery, *ii)* speed interval $V \in \{v_1, v_2, v_3\}$, *iii)* distance to path end $D \in \{d_0, d_1, ..., d_8\}$, where d_8 is the starting interval of the path (i.e., the maximal distance) and d_0 is the end point of the path (i.e., the goal), *iv)* time interval $T \in$

$\{t_1, t_2, t_3, t_4, t_5\}$, where t_1 is the time interval in which the path starts, *v)* current segment $S \in \{s_1, s_2, s_3, s_4\}$, *vi)* difficulty of segments $F_1, F_2, F_3, F_4 \mid F_j \in \{L, M, H\}$, where L=low, M=medium, H=high, and difficulties may depend on water flow or other environmental factors, *vii)* difficulty of current segment $F_c \in \{L, M, H\}$, *viii)* engine power $P \in \{L, M, H\}$, which corresponds to a uniform discretization of the combined effort of the drone's propellers.

States B, V, D, T and S are observable, while states $F_1, \ldots, F_4$ and F_c are hidden, hence the model has a mixed observability [10]. Rewards depend on engine power (i.e., -0.01 for *low* power, -0.02 for *medium* power and -0.05 for *high* power), time of arrival (i.e., +500 for arriving at time t_5, +1000 for arriving at time t_4, +1500 for arriving at time t_3, +2000 for arriving at time t_2, +2500 for arriving at time t_1) and battery depletion before arrival (i.e., -5000). The discount factor is $\gamma = 0.95$. The five hidden variables can assume three possible values each (i.e., L, M or H), for a total of $3^5 = 243$ configurations of the hidden state (e.g., $\mathbf{f} = (L, M, H, L, L)$). The belief b_t provides the probability over all possible configurations at time t. It is a vector $(b_t^1, \ldots, b_t^{243})$, such that $b_t^j \geq 0$ and $\sum_{j=1}^{243} b_t^j = 1$. We set the initial belief b_0 to the uniform distribution $(\frac{1}{243}, \ldots, \frac{1}{243})$.

In order to represent dependencies between segment difficulties we define two elements: *i)* a cosine similarity matrix which provides a measure of geometric similarity between pairs of segments in the path (see matrix K in Figure 1.c), *ii)* a pairwise MRF to compute a joint probability function over configurations of segment difficulties (see the graph in Figure 1.c). The MRF is used to factorize the joint probability of segment difficulties as a product of potential functions over the maximal cliques of the graph. The joint probability is then written as $p(\boldsymbol{f}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{q \in Q} \psi_q(\boldsymbol{f}_q|\boldsymbol{\theta}_q)$, where $\boldsymbol{f}$ is a vector of difficulties (e.g., (H, M, L, L)), $\boldsymbol{\theta}$ is a parametrization of the MRF, Q is the set of (maximal) cliques of the graph, $\psi_q(\boldsymbol{f}_q|\boldsymbol{\theta}_q)$ is a potential function for clique q and $Z(\boldsymbol{\theta})$ is a normalization factor called *partition function*. We use a simplified version of MRF called *pairwise MRF* in which the parametrization is restricted to the edges of the graph rather than to the maximal cliques and we conveniently express potentials as exponentials with *Boltzmann distribution*. In this way the product of potentials in $p(\boldsymbol{f}|\boldsymbol{\theta})$ can be computed by adding the energies of each of the maximal cliques. Since our difficulty variables are discrete, we represent the energy functions as tables of (non-negative) numbers representing the relative "compatibility" between the different assignments of segment difficulties. In particular, given a pair of segments $(s_i, s_j) \mid i, j \in \{1, 2, 3, 4\}$ having cosine similarity equal to 1, 0 or -1, the energy functions of the corresponding pair of difficulties $(f_i, f_j) \mid f_i, f_j \in \{L, M, H\}$ are defined as $E_1(v, w)$, $E_0(v, w)$ and $E_{-1}(v, w)$ where $v = \mathbb{I}(f_i)$, $w = \mathbb{I}(f_j)$ and $\mathbb{I}(\cdot)$ is the index function. To compute the energy function of couples of segments having generic cosine similarity $k \in [-1, 1]$ we interpolate the parameters in $E_1(v, w)$, $E_0(v, w)$ and $E_{-1}(v, w)$ by quadratic polynomials.

The joint probability computed by the MRF is finally used to update the beliefs generated step-by-step by the TDP simulator according to the formula $\bar{b}_t^j = \alpha \cdot b_t^j + (1 - \alpha) \cdot b_t^j \cdot p(\boldsymbol{f}^j)$, where $t \in \mathbb{N}$ is the time step, $j \in \{1, \ldots, 243\}$ is the index of the difficulty configuration (i.e., hidden state), $\alpha \in [0, 1]$ is a weighting factor, b_t^j is the probability of the j-th difficulty configuration provided by standard simulator, $p(\boldsymbol{f}^j))$

is the joint probability of the j-th difficulty configuration provided by the MRF, and $\bar{b}_t^j$ is the probability of the j-th difficulty configuration provided by TDP simulator.

3 Results

We implemented the POMDP model in a POMDPX XML file [1] and computed a policy for this model using function *pomdpsol* of the SARSOP solver [7] in the APPL Toolkit [1]. Then we simulated the model 100 times using the function *pomdpsim* of the APPL toolkit, thus obtaining 100 time-evolutions of the system. In all these simulations the true hidden state was manually fixed to $\bar{\boldsymbol{f}}^* = (H, H, L, L)$. We call these time evolutions *standard* (STD) simulations since they were computed by the standard simulator and do not consider the path geometry to improve the belief over segment difficulties. Each simulation has 300 time steps and stores information about the evolution of observable state variables, belief over hidden state, actions and rewards.

Subsequently, we modified the *pomdpsim* simulator in two ways: *i)* by forcing the belief to the true hidden state (i.e., the probability distribution has all the mass into the real state and zero probability for other states), we call this simulator *oracle* (ORC) since it has complete information about the true hidden state, *ii)* by implementing the TDP approach presented above, we call this simulator TDP since it considers path geometry to improve the belief update over segment (i.e., task) difficulties. We performed 100 simulations by ORC and TDP simulators, respectively, and compared results with those achieved by STD simulator, in terms of belief evolution, average number of steps to reach the goal, and average number of battery failures.

Figure 2.a shows the evolution of beliefs in the three simulation cases: columns represent different simulation strategies, namely, STD, TDP and ORC, while rows represent segments 1, 2, 3 and 4 of the square path. Each chart shows the average belief (over 100 simulations) for a particular simulation strategy in a specific segment. A consequence of the improved belief update strategy in TDP is the improvement of performance in terms of (reduced) number of steps to reach the end of the path. This is observable in Figure 2.b which shows the average number of steps required by each approach to reach the end of segments s_1, s_2, s_3 and s_4. We observe that using TDP (blue line) the agent is able to reach the end of the path with less steps than using STD (red line) and with a similar failure rate (i.e., the percentage of times in which the battery ends before the end of the path). ORC has the best performance with 96.4 steps, on average, to reach the end of the square and 24% of failures. The second best performance is that of TDP with 102.1 steps and a failure rate of 22%. Both approaches outperform STD which needs 111.6 steps and has a failure rate of 22%. A key extension of the proposed approach concerns scaling on more complex scenarios. To this end we are currently exploring the usage of Partially Observable Monte Carlo Planning (POMCP) methods [12].

Acknowledgments

This work is partially funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 689341.

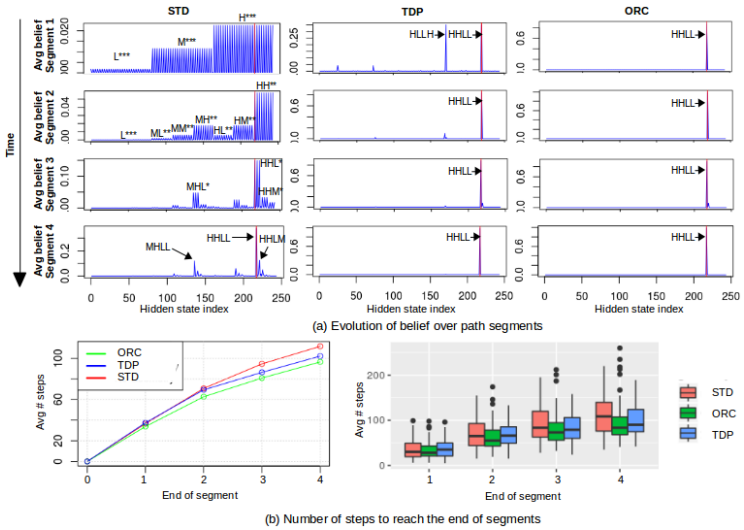

Fig. 2: a) Time evolution of the belief state. Red lines identify the true hidden state (H, H, L, L). Asteriscs in notation $H * **$ represents any difficulty. The belief of all simulators evolve towards the true hidden state but ORC overperformes TDP, and TDP overperforms STD. b) Number of steps needed to reach the end of each segment. Averages are computed over 100 simulations. ORC (green) reaches the end first, then TDP and finally STD.

References

1. APPL Toolkit. `http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/`.
2. V. Berenz, F. Tanaka, and K. Suzuki. Autonomous battery management for mobile robots based on risk and gain assessment. *Artificial Intelligence Review*, 37(3):217–237, 2012.
3. F. Dressler and G. Fuchs. Energy-aware operation and task allocation of autonomous robots. In *Proc. of the 5th Int. Workshop on Robot Motion and Control*, pages 163–168, 2005.
4. A. Hamza and N. Ayanian. Forecasting battery state of charge for robot missions. In *SAC*, pages 249–255, 2017.
5. INTCATCH H2020 EU project website. `http://www.intcatch.eu/`.
6. L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998.
7. H. Kurniawati, D. Hsu, and W. S. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.
8. Malrey Lee, Mahmoud Tarokh, and Matthew Cross. Fuzzy logic decision making for multi-robot security systems. *Artificial Intelligence Review*, 34(2):177–194, 2010.
9. J. LeSage and R. Longoria. Characterization of load uncertainty in unstructured terrains and applications to battery remaining run-time prediction. *J. Field Rob.*, 30(3):472–487, 2013.
10. S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee. Planning under uncertainty for robotic tasks with mixed observability. *Int. J. of Robotics Research*, 29(8):1053–1068, 2010.
11. A. Sadrpour, J. Jin, and A. G. Ulsoy. Mission energy prediction for unmanned ground vehicles. In *ICRA*, pages 2229–2234, 2012.
12. D. Silver and J. Veness. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems 23*, pages 2164–2172, 2010.