Rollout Algorithms for Data Storage- and Energy-Aware Data Retrieval Using Autonomous Underwater Vehicles

Pedro A. Forero†, Stephan K. Lapic†, Cherry Wakayama†, Michele Zorzi‡
†Space and Naval Warfare Systems Center Pacific, 53560 Hull St., San Diego, CA 92152, United States
‡Department of Information Engineering, University of Padova, via Gradenigo 6/B, 35131 Padova, Italy
{pedro.a.forero;stephan.lapic;cherry.wakayama}@navy.mil; zorzi@ing.unife.it

ABSTRACT

Increasingly effective underwater networks will be required to meet the growing demand for undersea data. The impending exploitation of non-acoustic underwater communication modes and the proliferation of autonomous underwater vehicles (AUVs) will enable the development of underwater networks able to use multiple modes of wireless communications and AUVs to transport data. In this paradigm, planning the routes for AUVs to collect data from underwater sensors becomes critical due to the dynamic nature of the undersea environment and the data collection process. This work proposes a dynamic path planning framework that enables judicious decisions on which network nodes the AUVs should visit next, based on the most recent network-status information. Routing decisions are aware of the AUVs own data-storage and energy constraints. Motivated by the intractability of optimal AUV routing, we propose a rollout algorithm as an enabler for dynamic AUV routing. Numerical tests illustrate the performance of the proposed algorithm.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Dynamic programming*

1. INTRODUCTION

The advent of enhanced sensing and battery technologies has led to the development of underwater systems with extended operational lifetimes that are able to collect unprecedented volumes of data. Prompt access to these data is necessary for developing improved monitoring and surveillance systems tailored to commercial, scientific and military applications [8]. Current methods for retrieving data from underwater nodes include using acoustic communications with a surface ship, tethering the underwater node to a buoy that can employ radio communications, and physical recovery of the nodes [2]. These methods suffer from various shortcomings since data throughput and energy usage are critical considerations for these battery-powered systems. Acoustic communications are bandwidth-limited, and together with radio communications can have a high energy-per-bit cost. Retrieving data by physical recovery of the nodes is maximally efficient in terms of battery usage but can have a high financial cost and fail to provide timely access to data [11].

Improved autonomous underwater vehicle (AUV) technologies have motivated a data-retrieval paradigm in which low-latency, high-bandwidth underwater wireless communication technologies, such as optical communications, are coupled with controlled node mobility. The energy-per-bit cost associated with optical communications is orders of magnitude lower than that of acoustic communications [13, 7]. Thus, optical communications can reduce the overall energy requirements related to underwater data retrieval. In this paradigm, AUVs approach underwater nodes to retrieve their data using optical communications. AUVs can visit multiple data-collecting nodes before returning to a *depot*. Depots are special nodes from which data can be exfiltrated over a high-bandwidth data link and where AUVs can recharge their battery. Acoustic communications will continue to play a significant role in this paradigm as enablers of low-bandwidth, long-range *multihop* connectivity. While not necessarily viable for moving large amounts of data underwater, acoustic links are ideal for the transmission of control data. Thus, a connected acoustic backbone can be a critical enabler for higher-bandwidth underwater communications.

A fundamental problem that arises in this paradigm is that of *planning* the paths to be followed by each AUV for collecting data while satisfying their own data-storage and energy constraints. Path planning for multiple vehicles with capacity constraints has a long history in the area of operational research [12]. In the context of underwater data retrieval, Vasilescu et al. developed an architecture using an AUV that followed a predefined data collection route and used optical communications to collect data [13]. Route planning and acoustic communication protocols for an AUV collecting data acoustically from underwater nodes were considered by Hollinger et al. [9]. Their work used packet loss due to the acoustic channel as a metric for choosing the locations from where the AUV should collect data. The *value of information* was used by Basagni et al. to develop a greedy approach for routing a data collecting AUV [3]. An acoustic backbone network was used to send a request for data collection to the AUV, which then decided on which node to collect data from. Although the resulting greedy algorithm is dynamic in nature, it cannot guarantee that all nodes in the network are visited within

WUWNET '14, November 12 - 14 2014, Rome, Italy
Copyright 2014 ACM 978-1-4503-3277-4/14/11 $15.00
http://dx.doi.org/10.1145/2671490.2674583.

the desired period of operation. Node cooperation whereby neighboring nodes move data to judiciously chosen nodes acting as throw-boxes was proposed by Lucani et al. [10]. In this collaborative paradigm, an AUV only collects data from throw-boxes and, thus, has fewer and more productive data collection visits while being able to reduce its route length. Other works have considered different AUV-aided data retrieval strategies via acoustic communications where AUVs communicate with more than one underwater node at a time, see [6] and references therein. Traits common to most of these works are: (i) only one AUV following a predefined path is employed for data collection; (ii) the energy consumption, battery recharge rates, and data-storage capacity of the AUVs and network nodes are not considered in the analysis; (iii) the per-node data amount to be collected is fixed and known; and, (iv) there exists a backbone network that allows nodes to move their data and relay messages to the AUVs at an affordable energy-per-bit cost. Although using fixed routes for AUVs leads to reduced control-data communication requirements for the overall network, they may lead to poor AUV performance due to their inability to adapt to the dynamic undersea environment and data collection process [11].

The main contribution of this paper is to develop a dynamic planning algorithm for routing multiple AUVs to retrieve data from underwater nodes that are continuously collecting data. Our model captures the limited energy capacity of the AUV's battery, the AUV-battery recharge time, and the data-storage capacity limits of the AUVs and the network nodes. We refrain from defining fixed routes for the AUVs. Instead, AUVs are routed dynamically thereby allowing them to adapt to the current system conditions, including the volume of data waiting to be collected at the nodes, the amount of energy remaining in the AUVs, and the AUV-battery recharge times. Dynamically updating AUV routes in accordance with the AUV's battery levels is paramount to avoid costly route failures that may leave an AUV adrift. Moreover, the resulting algorithm can accommodate unexpected failures and changes in the network without recalculation of AUV routes. Our framework presumes that the underwater nodes form a connected acoustic network through which network control data can be collected at a fusion center (FC), where routing decisions are made. In sparse node deployments the sporadic frequency with which AUVs visit nodes and depots, and the long AUV travel times between nodes allow enough time to the FC for collecting all control data required, making a routing decision and delivering that decision back to the AUVs. Note that our framework can also be used to outline performance baselines for future *decentralized* dynamic AUV routing.

The dynamics of the AUV routing problem are modeled as a Markov Decision Process (MDP) [4]. Our goal is to find an optimal routing policy that maximizes the aggregate reward given to the AUVs as they retrieve data from the underwater nodes over a possibly infinite mission duration. It is known that even for single-vehicle routing problems, finding an optimal routing policy is computationally intractable [12]. Thus, we propose to approximate the optimal policy using a one-step lookahead with rollout algorithm [4]. Our framework incorporates a prediction component that enables us to *deterministically* represent future states and actions over a finite time horizon.

The rest of the paper is organized as follows. In Section 2 we present modeling preliminaries. In Section 3 the data collection problem is cast in a sequential decision-making framework defined by an MDP. In Section 4 the rollout-based AUV routing algorithm is presented. In Section 5 the performance of the proposed algorithm is illustrated via simulations. The paper concludes in Section 6.

2. MODELING PRELIMINARIES

Consider an underwater data-collection system comprising the sets $\mathcal{V}_N := \{v_1, \ldots, v_N\}$ with N data-collecting nodes and $\mathcal{V}_D := \{v_{N+1}, \ldots, v_{N+D}\}$ with D depots, and U AUVs indexed by $\mathcal{U} := \{1, \ldots, U\}$. Nodes in $\mathcal{V}_N$ collect data until their respective data-storage capacity $\{\Phi_n\}_{n=1}^N$ is exhausted. Any new datum arriving to a node without available data-storage space is immediately discarded. Depots are special nodes with access to high-bandwidth data links and battery recharging capabilities for the AUVs. Each depot has $\breve{U} \leq U$ docking stations that AUVs can use to recharge their batteries and download their data payload. It is assumed that depots do not collect data, are always able to satisfy AUV energy demands, and are connected to an FC where AUV routing decisions are made. All nodes are endowed with acoustic and optical communications capabilities. The graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ models the underlying acoustic network, with vertices $\mathcal{V} := \mathcal{V}_N \cup \mathcal{V}_D$ and edges $\mathcal{E}$ defined by all node-pairs that can communicate acoustically. It is assumed that $\mathcal{G}$ is *connected*, i.e., for any $v, v' \in \mathcal{V}$ there exists a path in $\mathcal{G}$ (comprising acoustic links only) that connects v and v'.

The flotilla of AUVs collects data from the nodes via optical communication and delivers them to *any* one of several depots to be exfiltrated. AUVs have limited operational lifetimes dictated by their battery capacities $\{B_u\}_{u=1}^U$, and finite data-storage capacities $\{Q_u\}_{u=1}^U$. When an AUV visits a depot, the AUV downloads all its data payload and recharges its battery to full capacity. The charging rates for the AUV's batteries are $\{c_u\}_{u=1}^U$. If an AUV arrives at a depot whose docking stations are busy, then it has to wait until a docking station becomes available. For simplicity, it is assumed that the main energy cost associated with the operation of the AUVs is due to propulsion, which is modeled to be proportional to the distance travelled. The distance between any $v, v' \in \mathcal{V}$ is denoted $w_{v,v'} \geq 0$. Energy consumption and transmission delays associated with collecting data from the nodes and delivering data to the depots via optical communication links are assumed negligible.

The goal of the FC is to dynamically decide where to route AUVs so that the amount of data they collect is maximized and the amount of data lost at the nodes due to data-storage overflows is minimized. Routing decisions should be mindful of the AUVs' data-storage and energy constraints, so that the AUVs' storage capacity is not exceeded and their battery is not depleted. In the following section, the AUV routing problem is cast within a dynamic programming framework.

3. DYNAMIC AUV ROUTING

In this section, we model the data-collection system as an MDP and describe how an optimal routing policy can be obtained. An MDP is comprised by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R)$ where $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ the set of actions, $\mathcal{P}$ a set of transition probabilities, $\Pr(S'|S, \mathbf{a})$, each representing the probability of transitioning from $S \in \mathcal{S}$ to $S' \in \mathcal{S}$ after taking action $\mathbf{a} \in \mathcal{A}$, and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a one-step reward function obtained when transitioning from S to S' after using $\mathbf{a}$. MDPs satisfy the Markov property, thus the next state and reward of the MDP can be predicted based on the current state S and action $\mathbf{a}$ alone.

Next, we describe the components of the MDP characterizing our data-collection system under the assumption that state transitions are deterministic. Let the discrete-time index $k \in \mathbb{N}$ define decision-making epochs at which AUVs are to be routed, and $\mathbb{R}_+$ the set of non-negative real numbers. At the k-th decision-making epoch, $x_u^{(k)} \in \mathcal{V}$ represents the destination of the u-th AUV; $\tau_u^{(k)} \in \mathbb{R}_+$ the u-th AUV's time of arrival at its intended destination $x_u^{(k)}$, or the time at which the AUV will finish recharging its battery if $x_u^{(k)} \in \mathcal{V}_D$; $q_u^{(k)} \in [0, Q_u]$ the amount of data in the u-th AUV's storage buffer upon arrival at its destination; $b_u^{(k)} \in [0, B_u]$ the battery remaining at the u-th AUV after arriving at its destination; $\phi_n^{(k)} \in [0, \Phi_n]$ the amount of data waiting to be collected at the n-th node at the beginning of the next decision epoch; $\delta_n^{(k)} \in \mathbb{R}_+$ the last time that node n was visited by an AUV; and, $t_{d,\breve{u}}^{(k)} \in \mathbb{R}_+$ the remaining waiting time at the $\breve{u}$-th docking station of $v_d \in \mathcal{V}_D$. The state of the data-collection system is defined by the tuple

$$S^{(k)} := (\mathbf{x}^{(k)}, \boldsymbol{\tau}^{(k)}, \mathbf{q}^{(k)}, \mathbf{b}^{(k)}, \boldsymbol{\phi}^{(k)}, \boldsymbol{\delta}^{(k)}, \mathbf{T}^{(k)}) \in \mathcal{S} \tag{1}$$

where $\mathbf{x}^{(k)} := [x_1^{(k)}, \ldots, x_U^{(k)}]' \in \mathcal{V}^U$, $\boldsymbol{\tau}^{(k)} := [\tau_1^{(k)}, \ldots, \tau_U^{(k)}]' \in \mathbb{R}_+^U$, $\mathbf{q}^{(k)} := [q_1^{(k)}, \ldots, q_U^{(k)}]' \in [0, Q_1] \times \ldots \times [0, Q_U]$, $\mathbf{b}^{(k)} := [b_1^{(k)}, \ldots, b_U^{(k)}]' \in [0, B_1] \times \ldots \times [0, B_U]$, $\boldsymbol{\phi}^{(k)} := [\phi_1^{(k)}, \ldots, \phi_N^{(k)}]' \in [0, \Phi_1] \times \ldots \times [0, \Phi_N]$, $\boldsymbol{\delta}^{(k)} := [\delta_1^{(k)}, \ldots, \delta_N^{(k)}]' \in \mathbb{R}_+^N$, $\mathbf{T}^{(k)} := [\mathbf{t}_1^{(k)}, \ldots, \mathbf{t}_D^{(k)}] \in \mathbb{R}_+^{\breve{U} \times D}$ with $\mathbf{t}_d^{(k)} := [t_{1,d}^{(k)}, \ldots, t_{\breve{U},d}^{(k)}]' \in \mathbb{R}_+^{\breve{U}}$ $d = 1, \ldots, D$, and $(\cdot)'$ denotes transposition.

At k, the routing decision $\mathbf{a}^{(k)}$ is made using information from $S^{(k)}$ only. The set of AUVs ready to be routed, denoted as $\mathcal{U}^{(k)} \subseteq \mathcal{U}$, comprises all AUVs that have arrived to their destination $v_n \in \mathcal{V}_N$, and those that have finished recharging their batteries at a depot. The k-th decision-making epoch occurs at time $\tau^{(k)} := \min_{u \in \mathcal{U}} \tau_u^{(k)}$. Since $\tau_u^{(k)}$ defines the time at which the u-th AUV is ready to be routed, $\mathcal{U}^{(k)} = \{u \in \mathcal{U} : \tau^{(k)} = \tau_u^{(k)}\}$. Our definition of $\mathcal{U}^{(k)}$ tacitly assumes that the times required to upload and download data from nodes and depots are negligible and, thus, an AUV is ready to depart as soon as it arrives to a node. In contrast, battery charging times are not negligible and are explicitly considered.

The decision vector $\mathbf{a}^{(k)}(S^{(k)}) := [a_1^{(k)}(S^{(k)}), \ldots, a_U^{(k)}(S^{(k)})]' \in \mathcal{A}(S^{(k)})$ comprises the per-AUV routing decisions $a_u^{(k)}(S^{(k)}) \in \mathcal{V}$ made at k. The action space $\mathcal{A}(S^{(k)}) \subseteq \mathcal{V}^U$ defines the feasible AUV routing decisions as a function of $S^{(k)}$ as

$$\mathcal{A}(S^{(k)}) = \Big\{\mathbf{a} \in \mathcal{V}^U : a_u = x_u^{(k)} \ \forall u \in \bar{\mathcal{U}}^{(k)}, \tag{2a}$$

$$a_u = \underset{v_d \in \mathcal{V}_D}{\arg\min} \ \Big(\zeta_u w_{x_u^{(k)}, v_d} + \min_{\breve{u}} t_{d,\breve{u}}^{(k)}\Big) \ \forall u \in \mathcal{D}^{(k)}, \tag{2b}$$

$$a_u = x_u^{(k)} \ \ \forall \Big\{u \in \mathcal{U}^{(k)} : x_u^{(k)} \in \mathcal{V}_D, b_u^{(k)} < B_u\Big\}, \tag{2c}$$

$$a_u \neq a_{u'} \ \forall \{u, u' \in \mathcal{U} : u \neq u', a_u, a_{u'} \notin \mathcal{V}_D\}, \tag{2d}$$

$$a_u \notin \mathcal{V}_D \ \ \forall \{u \in \mathcal{U} : x_u^{(k)} \in \mathcal{V}_D, b_u^{(k)} = B_u\}, \tag{2e}$$

$$a_u \neq x_u^{(k)} \ \forall u \in \mathcal{U}^{(k)}, x_u^{(k)} \notin \mathcal{V}_D, \tag{2f}$$

$$b_u^{(k)} - \gamma_u \check{w}_{x_u^{(k)}, a_u} \geq 0 \ \ \forall u \in \mathcal{U}\Big\} \tag{2g}$$

where $\zeta_u > 0$ is the inverse of the velocity of the u-th AUV, $\bar{\mathcal{U}}^{(k)} := \mathcal{U} \backslash \mathcal{U}^{(k)}$ with $\backslash$ denoting set difference, $\check{w}_{v,v'} := w_{v,v'} + \min_{v_d \in \mathcal{V}_D} w_{v',v_d}$ the length of the shortest path from v to v' and from v' to its closest depot,

$$\mathcal{D}^{(k)} := \left\{u \in \mathcal{U}^{(k)} : (b_u^{(k)} - \min_{v_n \in \tilde{\mathcal{V}}_N^u} \gamma_u \check{w}_{x_u^{(k)}, v_n} < 0) \vee (q_u^{(k)} = Q_u)\right\} \tag{3}$$

with $\tilde{\mathcal{V}}_N^u := \mathcal{V}_N \backslash \{x_u^{(k)}, x_{u'}^{(k)} \in \mathcal{V}_N : u' \in \bar{\mathcal{U}}^{(k)}\}$, and $\gamma_u > 0$ is a scaling factor that maps distance traversed to energy consumption per AUV. Condition (2a) guarantees that AUVs that have not reached their destination are not rerouted and that those AUVs at a depot that have not fully recharged remain at that depot; (2b) that AUVs that do not have enough energy to visit any $v \in \mathcal{V}_N$ and then return to a depot, and those whose data-storage buffer is full are routed directly to a depot; (2c) that an AUV that just arrived at a depot remains at that depot until its battery is replenished; (2d) that no more than one AUV is assigned to any $v \in \mathcal{V}_N$ (only depots can host multiple AUVs); (2e) that a fully recharged AUV at a depot is routed to collect data; (2f) that AUVs ready to be routed are not routed to the same node where they currently are except for those at the depot that need to recharge; and, (2g) that all AUVs have enough energy to reach a depot after they arrive to their new destination as defined by $\mathbf{a}^{(k)}$. Note that (2b) assigns AUVs to the depot offering the lowest travel plus wait-to-recharge time.

It is assumed that the system transitions deterministically from $S^{(k)}$ to $S^{(k+1)}$ once $\mathbf{a}^{(k)}$ is chosen. Hence, for a given k , the corresponding entries of $\mathcal{P}$ are $\Pr(S'|S = S^{(k)}, \mathbf{a} = \mathbf{a}^{(k)}) = 1$ if $S' = S^{(k+1)}$ and $\Pr(S'|S = S^{(k)}, \mathbf{a} = \mathbf{a}^{(k)}) = 0$, otherwise. This model is relevant for a situation where the data collection rates at the nodes are fixed and known. The next section introduces our proposal for the reward R.

3.1 The reward function

By choosing $\mathbf{a}^{(k)}$, the system earns a reward $R(S^{(k)}, \mathbf{a}^{(k)})$. Our choice for R rewards the system for the amount of data collected by the AUVs, and penalizes it for the amount of data lost to overflows at the nodes and for violations to the desired revisit period $T > 0$. The reward $R(S^{(k)}, \mathbf{a}^{(k)})$ is defined as

$$R(S^{(k)}, \mathbf{a}^{(k)}) = \sum_{(u,n) \in \mathcal{H}_1^{(k)}} \Big[(q_u^{(k+1)} - q_u^{(k)}) - \lambda \beta_n^{(k),u}\Big] \tag{4}$$
$$- \lambda \sum_{n \in \mathcal{H}_2^{(k)}} \beta_n^{(k)} - \theta \left[|\mathcal{H}_3^{(k)}|(\tau^{(k+1)} - \tau^{(k)}) + \sum_{n \in \mathcal{H}_4^{(k)}} \varphi_n^{(k+1)}\right]$$

where $\lambda > 0$ scales data loses due to overflows, $\theta > 0$ scales violations to T, $\varphi_n^{(k+1)} := \max\{0, \tau^{(k+1)} - \delta_n^{(k+1)} - T\}$ denotes the length of the violation, $\mathcal{H}_1^{(k)} := \{(u, n) : u \in \mathcal{U}^{(k)}, a_u^{(k)} = v_n \in \mathcal{V}_N\}$ the set of node-AUV pairs for AUVs currently at nodes, $\mathcal{H}_2^{(k)} := \{n : v_n \in \mathcal{V}_N\} \backslash \{a_u^{(k)} : u \in \mathcal{U}\}$ the set of nodes not visited by AUVs, $\mathcal{H}_3^{(k)} := \{n : \delta_n^{(k+1)} + T < \tau^{(k)}, v_n \in \mathcal{V}_N\}$, $\mathcal{H}_4^{(k)} := \{n : \tau^{(k)} \leq \delta_n^{(k+1)} + T \leq \tau^{(k+1)}, v_n \in \mathcal{V}_N\}$, $\beta_n^{(k),u} := \max[\phi_n^{(k)} + z_n^{(k),u} - \Phi_n, 0]$ the data lost at $v_n \in \mathcal{V}_N$ between $\tau^{(k)}$ and $\tau_u^{(k+1)}$, $\beta_n^{(k)} := \max[\phi_n^{(k)} + z_n^{(k)} - \Phi_n, 0]$ the data lost at $v_n \in \mathcal{V}_N$ between $\tau^{(k)}$ and $\tau^{(k+1)}$, and $z_n^{(k)}$ ($z_n^{(k),u}$) the amount of data collected between $\tau^{(k)}$ and $\tau^{(k+1)}$ ($\tau_u^{(k+1)}$) at $v_n \in \mathcal{V}_N$ if the storage capacity of v_n were unlimited.

In (4), the amount of data collected (lost) by AUVs (at nodes) visiting (visited by) nodes (AUVs) is aggregated between $\tau^{(k)}$ and the arrival times of the corresponding AUVs. Data lost at all other nodes and revisit violations are aggregated between $\tau^{(k)}$ and $\tau^{(k+1)}$. The next sections describe the dynamics of the system and an optimal AUV routing approach based on dynamic programming.

3.2 Characterization of the system evolution

The dynamics of the data-collection system describe how $S^{(k)}$ evolves as decisions are made and new information arrives. Given $(S^{(k)}, \mathbf{a}^{(k)})$, the components of $S^{(k+1)}$ are defined as follows. Vehicle destinations are set to $x_u^{(k+1)} = a_u^{(k)}$.

Let the auxiliary vector $\boldsymbol{v}^{(k)} = \mathbf{0}_U$, where $\mathbf{0}_U$ is a $U \times 1$ vector of zeros. Per depot, we define a queue management policy that assigns AUVs to docking-station queues according to their waiting times. Let $\mathcal{U}_d^{(k)} := \{u \in \mathcal{U}^{(k)} : a_u^{(k)} = v_d \in \mathcal{V}_D\}$, with $|\mathcal{U}_d^{(k)}| =: L_d^{(k)} \leq U$, comprise the indices of AUVs assigned to $v_d \in \mathcal{V}_D$, and $\{b_{u_\ell}^{(k)} : u_\ell \in \mathcal{U}_d^{(k)}\}_{\ell=1}^{L_d^{(k)}}$ the remaining battery of the AUVs in $\mathcal{U}_d^{(k)}$. AUVs are assigned to queues via a sequential procedure as follows. First, we construct the order statistics $b_{(u_1)}^{(k)} \leq \ldots \leq b_{(u_{L_d^{(k)}})}^{(k)}$, and define auxiliary docking station waiting-time variables as $t_{\check{u},d}^{(k,1)} := t_{\check{u},d}^{(k)}$ $\forall \check{u}$. During the ℓ-th iteration, the AUV corresponding to $b_{(u_\ell)}^{(k)}$, denoted by $i_\ell^{(k)} \in \mathcal{U}_d^{(k)}$, is assigned to $\check{u}_d^{(k,\ell)} := \arg\min_{\check{u}} t_{\check{u},d}^{(k,\ell)}$. Then, the $i_\ell^{(k)}$ entry of $\boldsymbol{v}^{(k)}$ is set to $[\boldsymbol{v}^{(k)}]_{i_\ell^{(k)}} = t_{\check{u}_d^{(k,\ell)},d}^{(k,\ell)}$ and the auxiliary waiting times $t_{\check{u},d}^{(k,\ell)}$ are updated as

$$t_{\check{u},d}^{(k,\ell+1)} = \begin{cases} t_{\check{u},d}^{(k,\ell)} + (B_{i_\ell^{(k)}} - b_{i_\ell^{(k)}}^{(k)})/c_{i_\ell^{(k)}} & \check{u} = \check{u}_d^{(k,\ell)} \\ t_{\check{u},d}^{(k,\ell)} & \check{u} \neq \check{u}_d^{(k,\ell)} \end{cases}. \quad (5)$$

This procedure is repeated for $\ell = 1, \ldots, L_d^{(k)}$ until all elements of $\mathcal{U}_d^{(k)}$ are assigned to a docking station. Finally, at $\ell = L_d^{(k)}$, the entries of $\mathbf{t}_d^{(k)}$ are *partially* updated to $t_{\check{u},d}^{(k)} := t_{\check{u},d}^{(k,L_d^{(k)})}$. Now $\boldsymbol{v}^{(k)}$ contains waiting times for all $u \in \mathcal{U}_d^{(k)}$ at their assigned docking station. Note that $[\boldsymbol{v}^{(k)}]_u = 0$ $\forall u \notin \mathcal{U}_d^{(k)}$. AUV arrival times are updated to

$$\tau_u^{(k+1)} = \begin{cases} \tau_u^{(k)} & u \in \bar{\mathcal{U}}^{(k)} \quad (6a) \\ \tau_u^{(k)} + \zeta_u w_{x_u^{(k)}, a_u^{(k)}} & u \in \mathcal{B}_1^{(k)} \quad (6b) \\ \tau_u^{(k)} + [\boldsymbol{v}^{(k)}]_u + (B_u - b_u^{(k)})/c_u & u \in \mathcal{B}_2^{(k)} \quad (6c) \end{cases}$$

where $\mathcal{B}_1^{(k)} := \{u \in \mathcal{U}^{(k)} : (x_u^{(k)} \in \mathcal{V}_N) \vee (x_u^{(k)} \in \mathcal{V}_D, b_u^{(k)} = B_u)\}$, and $\mathcal{B}_2^{(k)} := \{u \in \mathcal{U}^{(k)} : x_u^{(k)} \in \mathcal{V}_D, b_u^{(k)} < B_u\}$. Row (6a) corresponds to arrival times for AUVs that remained in route, (6b) to those that were rerouted to some $v \in \mathcal{V}_N$ or those that are at a depot being routed elsewhere, and (6c) to those that just arrived to a depot and need to recharge their batteries. Then, each entry of $\mathbf{T}^{(k)}$ is updated as $t_{\check{u},d}^{(k+1)} = \max\left[t_{\check{u},d}^{(k)} - (\tau^{(k+1)} - \tau^{(k)}), 0\right]$ to account for the time elapsed between decision epochs.

The amount of data stored at each AUV is updated to

$$q_u^{(k+1)} = \begin{cases} q_u^{(k)} & u \in \bar{\mathcal{U}}^{(k)} \quad (7a) \\ 0 & u \in \mathcal{U}^{(k)}, a_u^{(k)} \in \mathcal{V}_D \quad (7b) \\ \min\left[q_u^{(k)} + \tilde{\phi}_{a_u^{(k)}}^{(k),u}, Q_u\right] & u \in \mathcal{U}^{(k)}, a_u^{(k)} \in \mathcal{V}_N \quad (7c) \end{cases}$$

where $\tilde{\phi}_n^{(k),u} := \min[\phi_n^{(k)} + z_n^{(k),u}, \Phi_n]$ denotes the data stored in $v_n \in \mathcal{V}_N$ at $\tau_u^{(k+1)}$. Row (7a) corresponds to AUVs that remained in route, (7b) to those that were routed to a depot, and (7c) to those that were routed to collected data.

The battery remaining at each AUV is updated to

$$b_u^{(k+1)} = \begin{cases} b_u^{(k)} & u \in \bar{\mathcal{U}}^{(k)} \quad (8a) \\ \max\left[b_u^{(k)} - \gamma_u w_{x_u^{(k)}, a_u^{(k)}}, 0\right] & u \in \mathcal{B}_1^{(k)} \quad (8b) \\ B_u & u \in \mathcal{B}_2^{(k)} \quad (8c) \end{cases}$$

where (8a) corresponds to the battery of the AUVs that remained in route, (8b) to the battery of AUVs not located at the depot or to AUVs at the depot that have recharged their batteries, and (8c) to AUVs at the depot that have not recharged their batteries.

The amount of data stored at each node is updated to

$$\phi_n^{(k+1)} = \max\left[0, \tilde{\phi}_n^{(k)} - \sum_{u \in \tilde{\mathcal{U}}_n^{(k)}} (Q_u - q_u^{(k)})\right] \quad (9)$$

where $\tilde{\mathcal{U}}_n^{(k)} := \{u \in \mathcal{U}^{(k+1)} : a_u^{(k)} = v_n \in \mathcal{V}_N\}$, $\tilde{\phi}_n^{(k)} := \min[\phi_n^{(k)} + z_n^{(k)}, \Phi_n]$ denotes the amount of data collected by $v_n \in \mathcal{V}_N$ up to $\tau^{(k+1)}$, and the sum accounts for data retrieved by an AUV from $v_n \in \mathcal{V}_N$ at $\tau^{(k+1)}$. Note that $\tilde{\mathcal{U}}_n^{(k)}$ has at most one element due to (2d).

Finally, the last time that a node was visited by an AUV is updated to

$$\delta_n^{(k+1)} = \begin{cases} \tau^{(k)} & n \in \mathcal{H}_5^{(k)} \quad (10a) \\ \delta_n^{(k)} & n \notin \mathcal{H}_5^{(k)} \quad (10b) \end{cases}$$

where $\mathcal{H}_5^{(k)} := \{n : a_u^{(k)} = v_n \in \mathcal{V}_N \text{ for some } u \in \mathcal{U}^{(k)}\}$. Row (10a) corresponds to nodes visited by an AUV at $\tau^{(k)}$, and (10b) to those nodes not visited at $\tau^{(k)}$.

3.3 Routing based on dynamic programming

Let Π denote the space of feasible AUV routing policies. A policy is a sequence of decision rules $\pi := (\mu_\pi^{(0)}, \mu_\pi^{(1)}, \ldots)$, where each decision rule $\mu_\pi^{(k)} : \mathcal{S} \to \mathcal{A}(S^{(k)})$ specifies how to route the AUVs when in $S^{(k)}$, i.e., $\mathbf{a}^{(k)}(S^{(k)}) = \mu_\pi^{(k)}(S^{(k)})$. Our goal is to obtain the best policy π^* dynamically, i.e., as the system evolves and new information is revealed, such that $\pi^* = \arg\max_{\pi \in \Pi} \sum_{k=0}^{\infty} \gamma^k R(S^{(k)}, \mu_\pi^{(k)}(S^{(k)}))$ where $\gamma \in (0, 1)$ is a discount factor and R the one-step reward.

Dynamic programming provides an algorithmic framework that lends itself naturally to finding π^* [4]. Bellman's principle of optimality states that if the optimal policy $\{\mu_{\pi^*}^{(i)}\}_{i \geq k+1}$ for $\{S^{(i)}\}_{i \geq k+1}$ were known, then at $S^{(k)}$ one would chose $\mathbf{a}^{(k)}$ as

$$\mathbf{a}^{(k)} = \arg\max_{\mathbf{a} \in \mathcal{A}(S^{(k)})} \left[R(S^{(k)}, \mathbf{a}) + \gamma V_{k+1}(S^{(k+1)}(S^{(k)}, \mathbf{a}))\right] \quad (11)$$

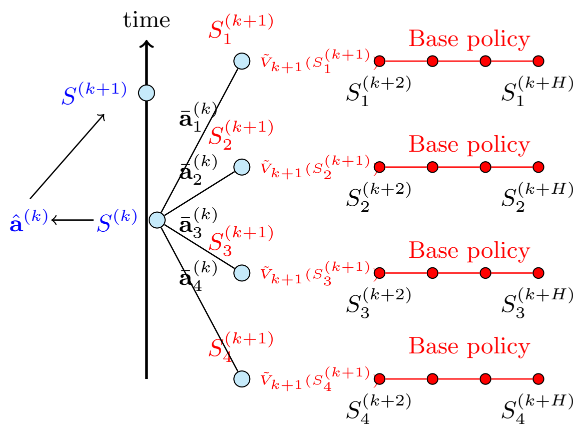


Figure 1: Schematic of one-step lookahead with rollout algorithm at $S^{(k)}$ for $|\mathcal{A}(S^{(k)})| = 4$.

where the notation $S^{(k+1)}(S^{(k)}, \mathbf{a})$ emphasizes the dependence of $S^{(k+1)}$ on the pair $(S^{(k)}, \mathbf{a})$, and

$$V_k(S^{(k)}) := \sum_{k'=k}^{\infty} \gamma^{k'-k} R(S^{(k')}, \mu_{\pi^*}^{(k')}(S^{(k')})) \qquad (12)$$

denotes the optimal cost-to-go. Unfortunately, using (11) to solve the AUV routing problem turns out to be computationally intractable even if the operational horizon were finite [4]. The following section develops a tractable approach to approximate the policy π^* that yields $\mathbf{a}^{(k)}$.

4. AUV ROUTING VIA ROLLOUT

This section develops a one-step lookahead with rollout algorithm for approximating π^*. Lookahead policies rely on explicit representations of future states and actions of the system over a finite horizon. Rollout algorithms rely on a suboptimal policy, known as *base-policy* that can be used to route AUVs. At the end of the lookahead horizon, the rollout component uses the base-policy to approximate the optimal cost-to-go V from that state onward.

When in $S^{(k)}$, the one-step lookahead with rollout algorithm yields an AUV routing decision

$$\hat{\mathbf{a}}^{(k)} = \underset{\mathbf{a}\in\mathcal{A}(S^{(k)})}{\arg\max} \Big[R(S^{(k)}, \mathbf{a}) + \gamma \tilde{V}_{k+1}(S^{(k+1)}(S^{(k)}, \mathbf{a})) \Big] \qquad (13)$$

where $\tilde{V}_{k+1}(S^{(k+1)}(S^{(k)}, \mathbf{a}))$ is an approximation of the cost-to-go function V_{k+1} in (11), and $\hat{\mathbf{a}}^{(k)}$ is the AUV routing decision made in lieu of the optimal $\mu_{\pi^*}^{(k)}$. The one-step lookahead approach constructs a rooted tree $\mathcal{T}^{(k)}$ with unit tree-depth, root node $S^{(k)}$, and $|\mathcal{A}(S^{(k)})|$ end-nodes. Each end-node corresponds to a possible new state $S_i^{(k+1)}$ and has an associated reward $R(S_i^{(k)}, \bar{\mathbf{a}}_i^{(k)})$, where $\bar{\mathbf{a}}_i^{(k)}$ denotes the action taken to evolve from $S^{(k)}$ to $S_i^{(k+1)}$. In general, a rollout algorithm with an l-step lookahead policy constructs an l-level tree rooted at $S^{(k)}$ summarizing all possible states and actions that follow from $S^{(k)}$. Unfortunately, the order of the tree that must be explored to find $\hat{\mathbf{a}}^{(k)}$ grows as $O((N+D)^l)$, thus we constrain ourselves to the case $l = 1$.

For each end-node of $\mathcal{T}^{(k)}$, we construct an approximation for the cost-to-go $\tilde{V}_{k+1}$ by using a base policy $\bar{\pi} :=$

Algorithm 1 Dynamic AUV routing

```
Require: Select $H, \gamma, \lambda, \theta > 0$ and
    $S^{(0)} = (\mathbf{0}_U, \mathbf{0}_U, \mathbf{0}_U, [B_1, \ldots, B_U]', \mathbf{0}_N, \mathbf{0}_N, \mathbf{0}_{\check{U}}\mathbf{0}'_D)$.
 1: for $k = 0, 1, 2, \ldots$ do
 2:     Acquire network-wide control-data to construct $S^{(k)}$.
 3:     Construct one-step lookahead tree $\mathcal{T}^{(k)}$ rooted at $S^{(k)}$.
 4:     for each end-node $S_i^{(k+1)}$ of $\mathcal{T}^{(k)}$ do
 5:         Generate $\{(S_i^{(k+h)}, \bar{\mathbf{a}}_i^{(k+h)})\}_{h=1}^{H}$ via (15).
 6:         Compute $\tilde{V}_{k+1}(S_i^{(k+1)})$ via (14).
 7:     end for
 8:     Compute $\hat{\mathbf{a}}^{(k)}$ via (13).
 9:     Disseminate routing decision to appropriate AUVs.
10:     Deterministic evolution to $S^{(k+1)}$ due to $\hat{\mathbf{a}}^{(k)}$.
11: end for
```

$\{\bar{\mu}_{\bar{\pi}}^{(k+h)}\}_{h=1}^{H}$ to *simulate* the evolution of the system over H decision epochs. Following $\bar{\pi}$ yields a sequence of H state-action pairs $\{(S_i^{(k+h)}, \bar{\mathbf{a}}_i^{(k+h)})\}_{h=1}^{H}$, where $\bar{\mathbf{a}}_i^{(k+h)} := \bar{\mu}_{\bar{\pi}}^{(k+h)}(S^{(k+h)})$. Then, $\tilde{V}_{k+1}(S_i^{(k+1)})$ is approximated as

$$\tilde{V}_{k+1}(S_i^{(k+1)}) = \sum_{h=1}^{H} \gamma^{h-1} R(S_i^{(k+h)}, \bar{\mathbf{a}}_i^{(k+h)}). \qquad (14)$$

This procedure is illustrated in Fig. 1.

A greedy algorithm with respect to a one-step reward function $R_G : \mathcal{S} \times \mathcal{A}(S^{(k)}) \to \mathbb{R}$ is used to define the base policy $\bar{\pi}$. Note that it is not necessary to set R_G equal to R. Given $S_i^{(k+h)}$, the routing decision for the AUVs based on $\bar{\pi}$ is

$$\mathbf{a}_i^{(k+h)} := \underset{\mathbf{a}\in\mathcal{A}(S_i^{(k+h)})}{\arg\max} \; R_G(S_i^{(k+h)}, \mathbf{a}). \qquad (15)$$

The resulting AUV routing algorithm is summarized as Algorithm 1. Its computational complexity grows linearly with the order of $\mathcal{T}^{(k)}$ which is upper bounded by $(N+D)^{|\mathcal{U}^{(k)}|}$. Note that in practical scenarios it is unlikely that $|\mathcal{U}^{(k)}| > 1$, except for the case when multiple AUVs must be concurrently routed from, e.g., a depot. Often, $|\mathcal{A}(S^{(k)})| < N + D$ due to the data-storage and energy constraints of the AUVs, especially as AUVs are closer to depleting their batteries. Solving (13) and (15) is done by first evaluating their corresponding costs for all feasible actions, as dictated by $|\mathcal{A}(S^{(k)})|$ and $|\mathcal{A}(S_i^{(k+h)})|$, respectively, and then choosing the actions that yield the largest reward. The computational complexity of these enumeration procedures grows linearly with the cardinality of the corresponding action space $\mathcal{A}$.

4.1 Practical implementation challenges

Algorithm 1 has an acquisition and a dissemination stage (lines 2 and 9, respectively) where network state information is sent from the nodes and AUVs towards the FC and routing decisions are sent back from the FC to the AUVs, respectively. The acquisition stage is added to cope with the uncertainty associated with the state transitions that would occur in a real environment. The true state $\check{S}^{(k)}$ may differ from $S^{(k)}$. Thus, making routing decisions based on $S^{(k)}$ can degrade the performance of the data-collection system. Instead, one can collect relevant state variables from the network at the beginning of the decision epoch and use them to construct an *updated* view of the system state.

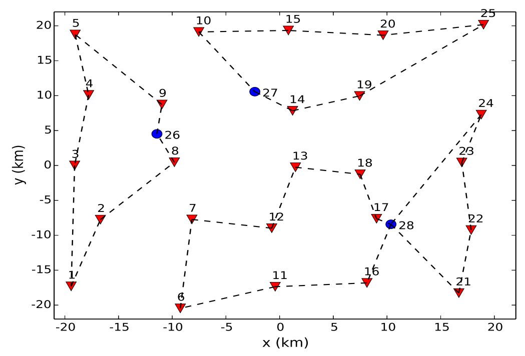


Figure 2: Node and depot deployment. Triangles (circles) represent data-collecting nodes (depots). Dashed lines show fixed AUV routes obtained with CW. Each route can be completed by an AUV with a single battery charge.

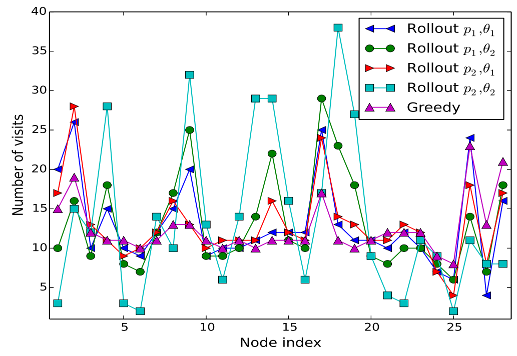
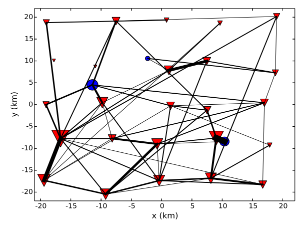
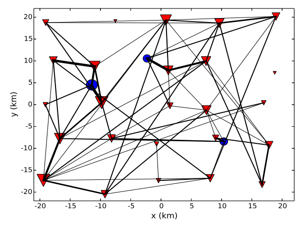


Figure 4: Frequency of AUV visits to nodes and depots after 400 decision epochs. Parameter values for Algorithm 1 were $p_1 := (H_1 = 50, \gamma_1 = 0.545, \lambda_1 = 0.575)$, $p_2 := (H_2 = 5, \gamma_2 = 2.0, \lambda_2 = 0.3225)$, and, $\theta_1 = 1,000$, and $\theta_2 = 100,000$. The greedy algorithm used $(\gamma_3 = 0.99, \lambda_1, \theta_1)$.

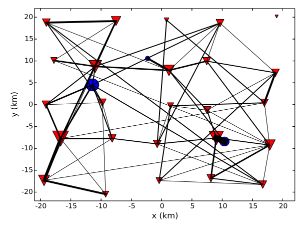
A decision epoch, k, is triggered when any of the AUVs arrive at a data-collecting node, or are at a depot ready to be routed. AUVs use the backbone acoustic network defined by $\mathcal{G}$ to send the tuple $(\breve{x}_u^{(k)}, \breve{\tau}_u^{(k)}, \breve{q}_u^{(k)}, \breve{b}_u^{(k)})$ of updated AUV state information to the FC, and a broadcast message alerting all $v \in \mathcal{V}_N$ that a decision epoch was triggered and requesting their state information. All nodes will send their updated state variables $\{(\breve{\phi}_n^{(k)})\}_{n=1}^N$ to the FC as soon as they receive the broadcast message from the AUVs. Note that $\mathbf{T}^{(k)}$ is immediately available to the FC since depots were endowed with high-bandwidth communication links connecting them directly to it, and the true $\breve{\boldsymbol{\delta}}^{(k)}$ is obtained by setting $\delta_n^{(k)} = \breve{\tau}_u^{(k)}$, $\forall u \in \mathcal{U}^{(k)}$. At this point, the FC can use $\breve{S}^{(k)}$ in lieu of $S^{(k)}$ and make a routing decision for the AUVs according to Algorithm 1.



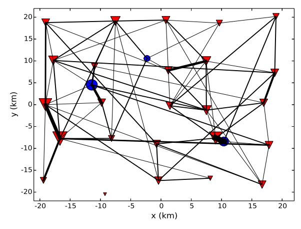
(a) AUV 1.



(b) AUV 2.



(c) AUV 3.



(d) AUV 4.

Figure 3: Illustration of AUV routes yielded by Algorithm 1 for $H = 50$, $\gamma = 0.7675$, $\lambda = 17$, and $\theta = 1,000$. Thickness of edges connecting node pairs is proportional to the frequency with which the edge was traversed and size of nodes is proportional to the frequency with which it was visited.

Once a decision is made at the FC, the new destinations for AUVs in $\mathcal{U}^{(k)}$ are sent through $\mathcal{G}$. Note that updated state variables from AUVs $u \in \bar{\mathcal{U}}^{(k)}$ were not collected since those AUVs may not be reachable through $\mathcal{G}$. If communicating with them were possible, updated data-storage and battery levels, and the $\breve{\phi}_n^{(k)}$'s of their corresponding destinations could be used to update their local state variables.

During the acquisition stage, a node, n, can send information to the FC to enable better estimates for $z_n^{(k+h)}$ and $z_n^{(k+h),u}$, $h = 0, \ldots, H$. Per node, historical data can be used to develop predictors for the amount of data that will be collected. These predictors will enable AUV routing even when the data-collection behavior of the nodes is unknown. This particular feature is not explored further in this work, rather it is left as a future research direction.

5. NUMERICAL EXPERIMENTS

In this section, the performance of Algorithm 1, implemented in Python 2.7.8, is illustrated via numerical tests. A data-collection system with $N = 25$ nodes, $D = 3$ depots, and $U = 4$ AUVs deployed over a 40 km $\times$ 40 km area is considered as a base case. A squared-grid topology is used to define the locations of the nodes, whose true locations are chosen uniformly at random within a 2-km radius of the corresponding grid point as shown in Fig. 2. Each depot has $\breve{U} = 2$ docking stations for recharging AUVs. AUVs are assumed to belong to the REMUS 100 family [1]. This family of AUVs have a 5.2 kWh battery that, when traveling at a nominal speed of 1.53 m/s (roughly 3 knots), yields an operational endurance of 22 hours. At this speed the shortest travel time between the closest pair of nodes is roughly 1.8 hours. Typical recharge rate for these batteries is 8 hours. In terms of data-storage space, each node has 162 kB of memory and each AUV has 1.62 MB of memory. Thus, if nodes collect data at a rate of 10 bits-per-second (bps) it would take 36 hours for them to overflow.

The one-step reward R_G for the base policy was set to $R_G(S^{(k)}, \mathbf{a}^{(k)}) := \sum_{(u,n)\in\mathcal{H}_1^{(k)}}[(q_u^{(k+1)} - q_u^{(k)}) - \lambda\beta_n^{(k),u}] - \lambda\sum_{n\in\mathcal{H}_2^{(k)}}\beta_n^{(k)} + \theta\sum_{n\in\mathcal{H}_5^{(k)}}\max\{0, \varphi_n^{(k+1)}\}$. Note that the last term of R_G has a positive sign (cf. (4)) and captures

Table 1: Sample performance statistics for Algorithm 1 after 400 decision epochs. Average (Ave.) data statistics correspond to system-wide averages over $\tau^{(400)}$ hours. The median in-between decision epoch time is denoted $\overline{\Delta\tau^{(k)}}$. Average and standard deviation (SD) for the per-decision-epoch execution times for experiments ran on a MacBook Pro with a 2.6 GHz Intel Core i5 processor and 8 GB of RAM are included. Statistics for greedy AUV routing correspond to the best choice of parameters found for R_G. Boldfaced numbers feature highest and lowest values per column for Algorithm 1.

Parameters				Duration (hours)		Total data (MB)		Ave. data (kB/hour)		Exec. time (sec.)
H	γ	λ	θ	$\tau^{(400)}$	$\overline{\Delta\tau^{(k)}}$	Lost	Collected	Lost	Collected	Ave. (SD)
5	0.3225	2.000	1,000	298.05	**0.62**	2.13	34.14	7.15	114.54	0.3565 (0.3026)
5	0.3225	2.000	10,000	253.00	0.40	3.73	26.82	14.76	106.00	0.3778 (0.2856)
5	0.3225	2.000	100,000	**185.45**	**0.31**	**6.20**	**15.96**	**33.46**	**86.04**	0.4048 (0.2830)
5	0.7675	2.000	1,000	317.71	0.60	2.64	36.08	8.32	113.57	**0.3001** (0.2893)
5	0.7675	2.000	10,000	280.41	0.47	4.13	30.30	14.73	108.06	0.3309 (0.2823)
5	0.7675	2.000	100,000	282.39	0.52	3.83	30.68	13.55	108.63	0.3131 (0.2708)
10	0.3225	1.050	1,000	308.64	**0.62**	2.76	35.10	8.94	113.74	0.6087 (0.5385)
10	0.3225	1.050	10,000	215.32	0.37	3.68	21.75	17.08	101.03	0.6935 (0.5019)
10	0.3225	1.050	100,000	192.93	**0.31**	5.52	16.97	28.01	87.96	0.6724 (0.5150)
10	0.5450	0.575	1,000	315.00	0.55	2.45	35.77	7.76	113.57	0.5722 (0.5331)
10	0.5450	0.575	10,000	296.28	0.46	3.94	31.77	13.30	107.21	0.6108 (0.5492)
10	0.5450	0.575	100,000	285.30	0.48	4.14	30.44	14.53	106.69	0.5826 (0.5322)
10	0.9900	6.000	1,000	311.70	0.49	2.15	36.20	6.90	**116.13**	0.5432 (0.5345)
50	0.5450	0.575	1,000	311.89	0.52	2.52	35.78	8.09	114.73	2.5617 (1.8525)
50	0.5450	0.575	10,000	283.52	0.51	3.75	30.41	13.21	107.25	2.4552 (1.7681)
50	0.5450	0.575	100,000	261.65	0.40	4.02	27.60	15.35	105.46	2.5381 (1.7483)
50	0.7675	1.525	1,000	327.58	0.56	2.22	**37.49**	6.78	114.45	2.3190 (1.8005)
50	0.7675	1.525	10,000	**329.49**	0.59	3.69	36.40	11.21	110.47	2.3834 (1.8334)
50	0.7675	1.525	100,000	320.26	0.57	4.37	34.29	13.65	107.07	2.3212 (1.7903)
50	0.7675	17.000	1,000	306.35	0.52	**1.76**	35.36	**5.74**	115.44	**2.6194** (1.9480)
Greedy	0.9900	0.575	1,000	404.34	0.69	9.07	40.91	22.42	101.18	0.0039 (0.0028)
CW	–	–	–	196.72	0.43	2.35	21.86	11.97	111.13	–

the entire length of the violation so as to encourage AUVs to visit nodes with longer revisit violations first. At $\tau^{(0)} = 0$, all AUVs were launched from depot 26 (see Fig. 2). To reduce the computational cost of constructing $\mathcal{T}^{(k)}$ for cases when $|\mathcal{U}^{(k)}| > 1$, our implementation of Algorithm 1 incrementally delays AUVs in $\mathcal{U}^{(k)}$ by 0.01 s. Thus, at every decision epoch AUVs are effectively routed one at a time, even when multiple AUVs trigger the same decision epoch. In the following tests, a deterministic scenario encompassing 400 decision epochs was considered. Data generation rates r_n were set to $r_1 = r_2 = 30$ bps, $r_{24} = r_{25} = 3$ bps, and $r_n = 10$ bps for all other $v_n \in \mathcal{V}_N$. The revisit period was set to $T = 36$ hours.

Table 1 displays various performance metrics for the data-collection system in Fig. 2. The values yielded by the best performers with respect to average data collected for $H = 5, 10, 50$ were tabulated. For the range of γ and λ parameters considered, the best performance was consistently achieved for $\theta = 1,000$. As θ increased it was also observed that the median of the in-between decision epoch durations decreased. This behavior was more apparent for small values of H. For comparison, a variation of the Clarke-Wright (CW) algorithm was employed to find fixed routes for the AUVs [5], see Fig. 2. The operating region was first divided into three sectors, each containing one of the depots and having areas in the ratio 1:1:2. The CW algorithm was then used per sector to find routes between the depot and the other nodes in that sector. The smaller sectors were assigned one AUV each and the larger region was assigned two. Greedy AUV routing, using R_G as defined earlier in this section, was also tested. Their performance is summarized at the bottom of Table 1. Note that the best average data collection performance of Algorithm 1 was 14.78% (4.50%) better than that of the greedy (CW) one. Also, the average data loss of Algorithm 1 for the same case was 69.22% (42.36%) less than that of the greedy (CW) one. Finally, note that the average execution time per decision epoch increases with H. However, their values are much smaller than those of $\overline{\Delta\tau^{(k)}}$. Thus, for the deployment in Fig. 2, Algorithm 1 does not preclude a computationally constrained FC from making decisions in *nearly* real time.

Fig. 3 illustrates the routes followed by each of the AUVs as they collect data from the network nodes. For this scenario, $H = 50$, $\gamma = 0.7675$, $\lambda = 17$, and $\theta = 1,000$ were used since they yielded the a good tradeoff between average data collected and data lost (cf. Table 1). Note that AUVs were routed throughout the entire deployment (cf. Fig. 2). Increasing θ led to decisions with shorter AUV travel times. In these cases, Algorithm 1 failed to make decisions that led to more frequent visits to nodes with higher r_n so as to avoid the large cost attached to long node-revisit-period violations (cf. Fig. 4). For this test, fewer than 10% of the in-between decision epoch time lengths were 6 minutes or less, and their median duration was 31.2 minutes.

Fig. 4 shows the frequency with which each node was visited by an AUV for different configurations of Algorithm 1. Not surprisingly, v_{24} and v_{25} were the least visited ones. The number of AUV visits to v_8, v_9, v_{14} and v_{17} was consistently higher than that to other nodes. Their close proximity to the depots allowed AUVs to visit these nodes when in the vicinity of a depot. The number of visits to each of these nodes is correlated to the number of visits to their closest depot. Also, note that both Algorithm 1 and the greedy algorithm visited all nodes in every case considered (cf. the

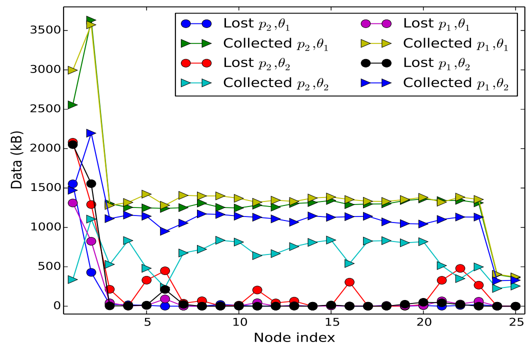
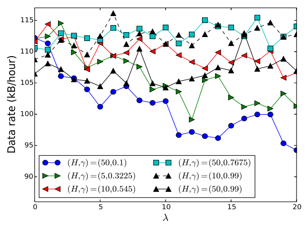


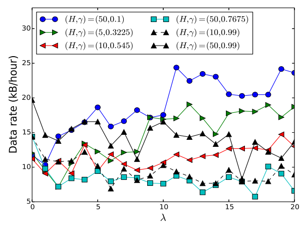
Figure 5: Aggregate data collected and lost up to $\tau^{(400)}$. Parameter values for Algorithm 1 are as defined in Fig. 4.

greedy approach in [3]).

Fig. 5 illustrates the aggregate amounts of data collected by AUVs per node and the aggregate amounts of data lost per node. AUVs collect the most (least) data out of v_1 and v_2 (v_{24} and v_{25}) which correlates with the fact that these were the nodes with the largest (lowest) r_n. Larger values of θ led to smaller amounts of data collected across all nodes, with the most noticeable decrease occurring at v_1 and v_2. Fig. 6 illustrates average data collected and lost throughout the network as a function of λ. Since data losses become more costly as λ increases, AUVs tend to travel more often to nodes located nearby thereby decreasing the in-between decision epoch lengths, and thus decreasing $\tau^{(400)}$. By varying γ, one can indirectly control the length of the planning horizon. As shown in Fig. 6, the data collection system benefits from medium-length planning horizons, with $(H,\gamma) = (50, 0.7675)$ and $(H,\gamma) = (10, 0.99)$ yielding the best performances. Interestingly, long planning horizons yielded inferior performance in part due to the approximation error inherent to $\tilde{V}$.



(a) Average data collected.



(b) Average data lost.

Figure 6: Average statistics of data collected and lost obtained for a wide range of λ's with $\theta = 1,000$.

6. CONCLUSIONS AND FUTURE WORK

This work proposed a dynamic path-planning framework for AUVs used to retrieve data from underwater nodes. Routing decisions are based on the most recent network-status information and cognizant of data-storage and energy constraints. Our routing policy was obtained via a one-step lookahead rollout algorithm.

In future work, we plan to include stochastic models for AUV travel times and perturbations on the observable state variables, and develop predictors for data volumes collected at the nodes. These predictors will reduce the computational complexity associated with making a judicious AUV routing decision when facing stochastic state transitions, and when the backbone acoustic network is not available. Lastly, we plan to explore the impact of the characteristics of the backbone acoustic network on the effectiveness of the dynamic routing decisions made.

7. ACKNOWLEDGMENTS

This work was funded by the Office of Naval Research under award N0001413WX00411 AA.

8. REFERENCES

[1] REMUS 100 - Autonomous underwater vehicle. `http://www.km.kongsberg.com`.

[2] I. F. Akyildiz, D. Pompili, and T. Melodia. State of the art in protocol research for underwater acoustic sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 11(4):11–22, Oct. 2007.

[3] S. Basagni, L. Bölöni, P. Gjanci, C. Petrioli, C. A. Phillips, and D. Turgut. Maximizing the value of sensed information in underwater wireless sensor networks via an autonomous underwater vehicle. In *Proc. of IEEE INFOCOM*, pages 988–996, Apr. 27 - May 2, Toronto, Canada, 2014.

[4] D. P. Bertsekas. *Dynamic Programming and Optimal Control: Approximate Dynamic Programming.* Athena Scientific, 4nd edition, 2012.

[5] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[6] F. Favaro, L. Brolo, G. Toso, P. Casari, and M. Zorzi. A study on remote data retrieval strategies in underwater acoustic networks. In *Proc. of MTS/IEEE OCEANS Conference*, pages 1–8, Sep. 23-26, San Diego, CA, USA, 2013.

[7] F. Hanson and S. Radic. High bandwidth underwater optical communication. *Appl. Opt.*, 47(2):277–283, Jan. 2008.

[8] J. Heidemann, M. Stojanovic, and M. Zorzi. Underwater sensor networks: applications, advances and challenges. *Phil. Trans. R. Soc. A*, 370(1958):158–175, 2012.

[9] G. A. Hollinger, S. Choudhary, P. Qarabaqi, C. Murphy, U. Mitra, G. Sukhatme, M. Stojanovic, H. Singh, and F. Hover. Underwater data collection using robotic sensor networks. *IEEE J. Sel. Areas Commun.*, 30(5):899–911, 2012.

[10] D. E. Lucani, P. B. Sujit, and J. B. Sousa. Spare the mule, help your neighbors: Robot route planning for data retrieval on large scale sensor networks. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, pages 2534–2541, May 6 - 10, Karlsruhe, Germany, 2013.

[11] J. Partan, J. Kurose, and B. N. Levine. A survey of practical issues in underwater networks. In *Proc. of the 1st ACM Intl. Workshop on Underwater Networks*, WUWNet '06, pages 17–24, New York, NY, USA, 2006. ACM.

[12] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia. A review of dynamic vehicle routing problems. *Eur. J. Oper. Res.*, 225(1):1 –11, 2013.

[13] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke. Data collection, storage, and retrieval with an underwater sensor network. In *Proc. of the 3rd Intl. Conf. on Embedded Networked Sensor Systems*, SenSys '05, pages 154–165, New York, NY, USA, 2005. ACM.