Stochastic Optimization for Autonomous Vehicles with Limited Control Authority

Dylan Jones[1], Michael J. Kuhlman[2,*], Donald A. Sofge[2], Satyandra K. Gupta[3], Geoffrey A. Hollinger[1]

***Abstract*— In this work, we present a Stochastic Gradient Ascent (SGA) algorithm for multi-vehicle information gathering that accounts for limitations on a vehicle's control authority caused by external forces. By representing vehicle paths using a novel action space representation, rather than a state space representation, we remove the need to perform feasibility calculations on the vehicle's path. Our algorithm uses a stochastic optimization scheme by sampling perturbed action sequences around the current best known sequence to estimate the gradient of a state space information function with respect to the action sequence. Additionally, we use sequential greedy allocation to plan for multiple vehicles. Results are shown using a Navy Coastal Ocean Model (NCOM) for the Gulf of Mexico (GoM). SGA shows improvement in the amount of information gained over a greedy baseline. Additionally, we compare to Monte Carlo Tree Search (MCTS) Method, which is able to gather competitive amounts of information but is more computationally intensive than our approach.**

I. INTRODUCTION

In order to better measure and understand oceanographic processes, oceanographers need to be able to monitor large areas of the ocean for long periods of time. Traditionally, researchers collect data by chartering a research vessel which can cost upwards of $30,000 per day as noted in [1]. As such, there is a significant interest in using autonomous vehicles, such as underwater gliders (see [2]) or ocean profiling Lagrangian drifters (see [3]), which are more cost efficient and allow for the collection of data for months (for gliders) or years (for drifters). These vehicles achieve extended endurance at the cost of strong actuation. Due to this trade off, vehicle motion is heavily dictated by external forces, such as ocean currents, as shown in Figure 1.

Previous work (such as [3], [4], [5]) has looked at path and motion planning for these classes of vehicles. However, these approaches have been limited by a number of assumptions, especially in dealing with limits to control authority introduced by ocean currents. Rather, the techniques usually assume that any desired navigation action can be achieved. This assumption introduces problems when the vehicle is commanded to travel in a direction against the ocean currents. When these ocean currents are stronger than the actuation the vehicle can provide, the vehicle is unable to achieve the desired control and so does not execute the planned path. When previous work has considered limits to control authority, it has not considered complex goals, such as information gathering, and instead focused on reaching a desired destination. The information gathering problem has rewards which are no longer path independent, as the vehicle will gather information as it moves through the world. An example of such an information function can be seen in Figure 1, which shows a simulated information field over the Gulf of Mexico.

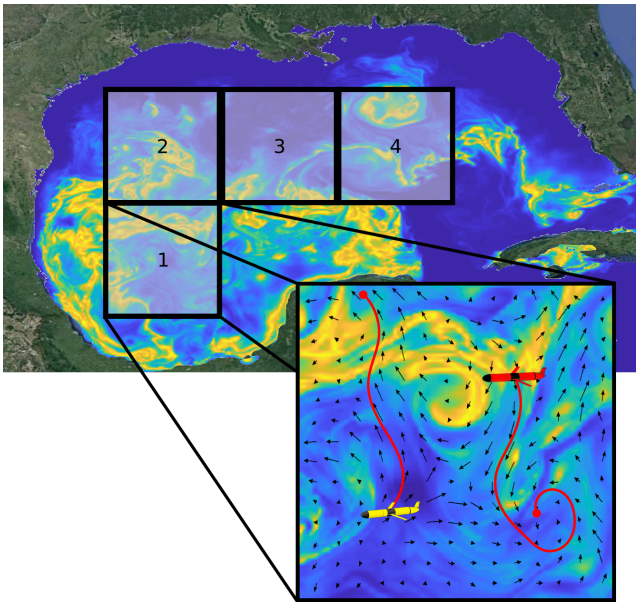

Fig. 1: Representative information field over the Gulf of Mexico where brighter colors indicate more information. The inset shows a zoomed in view of one of the four designated test regions used in the results. Two vehicles' paths are illustrated in the inset. The vehicle on the left (yellow) is able to utilize the ocean currents to move from a location of low information to one of high information. In contrast, the vehicle on the right (red) starts in a region of high information but is unable to stay there due to the limited control authority introduced by the ocean currents.

In this work we attempt to bridge the gap between previous work considering information gathering and previous work considering limits to control authority. We present a stochastic optimization algorithm for information gathering that:

- Allows for different levels of control authority by using a novel action sequence path representation
- Approximates the gradient of the path dependent state

†This work was supported by NRL 6.2 funding.

[1]D. Jones and G. Hollinger are with the Collaborative Robotics and Intelligent Systems (CoRIS) Institute, Corvallis, OR, 97330 USA {jonesdy, geoff.hollinger}@oregonstate.edu

[2]M. Kuhlman and D. Sofge are with the Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington, DC 20375 USA {michael.kuhlman, donald.sofge}@nrl.navy.mil

[3] S. Gupta is with the Aerospace and Mechanical Engineering Department and the Center for Advanced Manufacturing at the University of Southern California, Los Angeles, CA 90089 USA guptask@usc.edu

*M. Kuhlman is a Ph.D candidate in the Department of Mechanical Engineering, University of Maryland, College Park, MD 20742 USA

space reward function with respect to the action sequence path using random roll-outs
- Uses a Sequential Greedy Allocation scheme that allows the algorithm to be scalable for multi-vehicle implementations

We validate this algorithm on data from a Navy Coastal Ocean Model (NCOM) of the Gulf of Mexico, where the dataset is constructed similarly to [6]. This data set includes ocean currents, which affect the control authority of the vehicles, and ocean temperatures, that we use to construct an information function. We assume that there is some desired temperature that could correlate with an oceanographic process of interest, such as algae blooms, and assign each location an amount of information corresponding to how close it is to that desired temperature such as in [7].

The remainder of the paper is organized as follows. We begin with a discussion of the related work on trajectory optimization, marine robotics, and information gathering (Section II). We next introduce our problem formulation (Section III). Following this, we present our proposed Stochastic Gradient Ascent (SGA) algorithm and multi-robot coordination scheme (Section IV). We then introduce our comparison methods (Section V) and include a brief introduction to the Monte Carlo Tree Search (MCTS) Method that we use for comparison (Section V-A). Next, results for SGA and MCTS methods are shown using data from a NCOM model (Section VI). Lastly, we conclude and discuss future directions for this research (Section VII).

II. Related Work

This work is inspired by previous work in three major areas of research: path optimization, marine robotics, and information gathering. Much previous work has addressed trajectory optimization for motion planning. In [8], the authors present CHOMP, a trajectory optimization scheme based upon covariant gradient descent. In [9], the authors present an algorithm called TrajOpt, which uses sequential convex optimization and a novel collision checking formulation to reduce the number of iterations the algorithm requires to converge. In [10], trajectories are represented as a Gaussian Process, and a gradient-based optimization technique is used to optimize the path. However, all these works require calculation of the gradient of the cost function, which requires the method to approximate the cost function when an analytical gradient cannot be calculated. To remove this requirement, the authors in [11] present STOMP, which uses noisy trajectory rollouts to iteratively improve the path. Building on this work, [12] presents EESTO, which expands this stochastic optimization to allow the algorithm to consider cost functions where the time spacing between trajectory waypoints can change. Our current work is complementary to these previous works by considering information gathering cost functions and using an action sequence trajectory representation.

Previous work in path planning and information gathering for marine robotics has mostly either assumed fully actuated systems or dealt with planning to reach a goal region. Authors in [4] use a branch and bound technique to plan informative paths. However, this method scales exponentially with the length of the path and would not extend well to a multi-robot implementation. In [13], the authors present an implementation of a sparse Gaussian Process, which allows for a compact representation of a discovered information function. In [5], the authors propose a Markov-based path planner for information gathering. However, both these works assume that the agent can achieve any desired path, which may not be true for the class of limited control authority vehicles considered here.

To address the problem of planning achievable paths, [14] creates a controllability map over the ocean. However, they use this controllability map to plan trajectories for patrolling tasks and do not account for the situation where the agent may want to enter a region of low controllability if the direction of controllability is favorable. Additionally, the authors in [14] utilize an A* based path planner, which is ill-suited to our problem as it is quite difficult to design an informative heuristic for information gathering as it is difficult to compute a tight bound on the future information. In [15], the authors describe a high-level controller design for spreading and attracting Lagrangian drifters. However, they only consider final destinations and do not account for path dependent rewards such as information.

In [16], the authors introduce an information theoretic solution method for model predictive path integral (MPPI) control. To calculate updates to the control, the authors use a large number of samples to approximate the KL-Divergence between the current control distribution and an optimal control distribution, which is used to derive an iterative control update law. In this work, we use sampling to approximate a gradient, which allows our method to use a comparatively smaller number of samples. Additionally, it is unknown how MPPI will handle the large control space which our vehicles can choose from.

In order to bridge the gap in previous work, we present a scalable algorithm which allows for an efficient multi-robot implementation and uses stochastic optimization to approximate the gradient of the path dependent reward given by an information field.

III. Problem Formulation

We seek to find the set of feasible paths for a team of vehicles which maximizes a given reward function. This is formulated as the following optimization problem:

$$\mathbb{X}^* = \max_{\mathbb{X}\in\Omega} \; R(\mathbb{X}), \text{ s.t. } \forall \; \mathbf{x}_i \in \mathbb{X}, \mathbf{x}_i \in \Psi, \tag{1}$$

$$\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\},$$

where $\mathbf{x}_i$ is the path of an individual agent, $\mathbb{X}$ is a set of paths, $R(\mathbb{X})$ is a user defined reward function, Ω is the set of all paths, Ψ is the set of all feasible paths, and n is the number of agents being considered. In previous work, such as [14] and [15], these paths were specified

through a set of state-space coordinates defined by latitude-longitude-depth coordinates. However, optimizing the state-space coordinates directly requires nontrivial calculations to ensure that the trajectory is feasible due to control limits compared to environmental disturbances. We choose to plan over sequences of actions available to the vehicle, which removes the need to perform these feasibility calculations. Instead, we assume that we have a function $\Phi : \mathbb{A}, x_0 \mapsto \mathbb{X}$ where $\mathbb{A}$ is a set of action sequences and x_0 is a defined state-space starting location. This redefines our optimization problem as:

$$\mathbb{A}^* = \max_{\mathbb{A} \in \Lambda} R(\Phi(\mathbb{A}, x_0)), \tag{2}$$

where we are trying to find the optimal set of action sequences in Λ, the set of all possible action sequences.

We define an individual action sequence as:

$$\mathbf{a} = \{(d_1, t_1, v_1), (d_2, t_2, v_2), \cdots, (d_m, t_m, v_m)\},$$

where (d_i, t_i, v_i) defines a single action of diving to depth d_i, thrusting at velocity v_i, and maintaining that depth and thrust for an amount of time t_i. We assume that the agents have the ability to achieve and maintain a range of depths and provide a limited amount thrust. Using this definition, $\mathbb{A} = \{\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n\}$.

We use a generalized autonomous underwater vehicle (AUV) model to demonstrate our algorithm, but our formulation is general and a number of different sets of actions, a, can be used in the action sequence, $\mathbf{a}$. We do make two assumptions about the actions that can be included in the actions sequence. First, we assume that the action can be represented by a function $f : x_0, a \mapsto x_1$, which maps actions to states with a defined on a range:

$$a \in \lambda = [\beta, \alpha],$$

where α defines an upper bound and β defines a lower bound. This means that the action can be represented as a continuous real number which can be used to map from one location to another. Second, we assume that it is simple to check both that $a \in \lambda$ and that x_1 is a valid state for the vehicle.

In this work, we are interested in information gathering tasks defined by an information function, $R(\mathbb{X}) = I(\mathbb{X})$, which maps a set of state-space paths to an amount of information gathered by the team of agents. Again, we can transform this into our action sequence representation as $I(\Phi(\mathbb{A}, x_0))$.

IV. Method

We use the Stochastic Gradient Ascent (SGA) algorithm, shown in Algorithm I, to approximate $\mathbb{A}^*$ by iteratively updating an initial guess. The first step is to initialize the set $\mathbb{A}$ using a simple default policy. Next, we optimize the action sequences using a sequential greedy allocation method by optimizing one vehicle while holding all others fixed. This was shown in [17] to produce high-quality results in similar domains. For each vehicle, we iteratively improve the action sequence by calculating a number of perturbed sequences, scoring them, and then doing a weighted recombination of the perturbations to estimate the gradient which is used to update the sequence for the next iteration.

Algorithm 1 Stochastic Gradient Ascent (SGA)

```
Initialize 𝔸
for all a_i in 𝔸 do                          ▷ Loop through vehicles
    c ← 0
    for c < num_its do
        Θ, E ← get_perturbed_paths(a_i)
        S ← get_scores(Θ, 𝔸)
        Δ ← estimate_grad(S, E)
        a_i ← a_i + Δ * η(c)
        c ← c + 1
    end for
end for
```

We will now go through each of the subroutines:

- **get_perturbed_paths**: We calculate the set of perturbations, E, in the following way:

$$E = \epsilon_1, \epsilon_2, \cdots, \epsilon_K,$$

$$\epsilon_k = \{(\hat{d_1}, \hat{t_1}, \hat{v_1}), (\hat{d_2}, \hat{t_2}, \hat{v_2}), \cdots, (\hat{d_m}, \hat{t_m}, \hat{v_m})\},$$

$$\hat{d_i} = \mathcal{N}(0, \sigma_d),\ \hat{t_i} = \mathcal{N}(0, \sigma_t),\ \hat{v_i} = \mathcal{N}(0, \sigma_v),$$

  where K is the desired number of perturbed sequences and individual waypoint perturbations $\epsilon_k^j = (\hat{d_j}, \hat{t_j}, \hat{v_j})$ are sampled from zero-mean normal distributions with standard deviations σ_d, σ_t and σ_v, respectively.
  We then calculate the set of perturbed sequences, Θ, as:

$$\Theta = \theta_1, \theta_2, \cdots, \theta_K,$$

$$\theta_k = \mathbf{a}_i + \epsilon_k,$$

  where θ_k is perturbed action sequence k produced by adding perturbation vector ϵ_k to $\mathbf{a}_i$.
- **get_scores**: We calculate the score of each individual action of all perturbed action sequences in Θ using a method inspired from difference learning, presented in [18]. This calculation approximates the contribution of action θ_k^j by estimating how much it improves over the existing solution. The score, $s_{k,j}$, is the score of action j of θ_k and is calculated as:

$$s_{k,j} = \mathcal{I}(\bar{\mathbb{A}}) - \mathcal{I}(\tilde{\mathbb{A}}),$$

  where $\bar{\mathbb{A}}$ is $\mathbb{A}$ except $\mathbf{a}_i$ is replaced with θ_k and $\tilde{\mathbb{A}}$ is $\mathbb{A}$ with $\mathbf{a}_i$ replaced with θ_k but action θ_k^j replaced by a_i^j. By doing this, we also enable our gradient calculation to be more efficient by containing information specifically about the improvement offered by the waypoint, rather than containing a large amount of noise about the environment and other waypoints.
- **estimate_grad**: To calculate the gradient we need to convert the scores into costs by taking the inverse, which

gives a cost matrix $C = \frac{1}{S}$. We then use the following calculation to approximate the gradient:

$$\Delta_j = \frac{1}{K} \sum_{k=1}^{K} w(\theta_k^j) * \epsilon_k^j,$$

$$w(\theta_k^j) = e^{-h\left(\frac{C_k^j - \min C_k^j}{\max C_k^j - \min C_k^j}\right)},$$

where the function $w(\cdot)$ is a weighting function and Δ_j is gradient at action j in the action sequence. The variable h serves as a weighting term and is set equal to 10 in this work. The $\max$ and $\min$ functions are taken over all K perturbed sequences at action j. This compares the cost of the action under consideration to that of all other actions at the same point in the path. By scaling the weighting by the maximum and minimum seen in that iteration, the algorithm is able to more accurately account for large improvements offered by a small number of samples. Additionally, during implementation we include the original path as one of the samples which helps stabilize the algorithm and allows for the gradient calculation to account for the existing solution and scale the contribution of each individual waypoint based upon how much the waypoint improves over the existing solution.

Using the calculated Δ_j values, we can compute Δ as $\{\Delta_1, \Delta_2, \cdots, \Delta_m\}$. Using this Δ, we can then update the sequence using $\eta(c)$ which serves as a learning rate, which can be a function of the iteration number and in this work is calculated as $\eta(c) = 0.99^c$.

V. Benchmark Algorithms

We now discuss the various algorithms we benchmark against our SGA algorithm. We assume an action sequence consists of seven actions, each of which lasts for a duration of 24 hours, which, unless otherwise noted, randomly selects a velocity which is executed for the duration of the mission. Below we list our five benchmark algorithms.

- **Const Depth**: The first default policy is to have all vehicles maintain the same constant depth for the duration of the mission. In this work, we have the vehicle maintain a constant depth of zero (i.e. floating on the surface).
- **Diff Depth**: The second default policy is to equally distribute the vehicles throughout the water column. By doing this, the vehicles hope to find different ocean currents which will cause them to spread out. In this work, we have the vehicles spread out every 20 meters in the depth column from zero to 180 meters.
- **Greedy Depth**: The third default policy is iteratively choosing a constant depth and thrust for a new vehicle as it is added to the team that maximizes the amount of information gathered by the team. To do this, we discretize the possible depths and thrusts and then forward simulate the team with the new vehicle at all these possible depths and thrusts. We then select the depth and thrust that maximizes the information gathered.
- **Rand Policy**: The final default policy is a uniform random policy. Each action is chosen randomly, with the vehicle choosing from a discrete set of depths and thrusting in one of the four cardinal directions.
- **Monte Carlo Tree Search (MCTS)**: MCTS is a natural extension of the Rand Policy, iteratively improving the policy over time. This state-of-the-art approach is outlined below.

A. Monte Carlo Tree Search

Instead of approximately solving the optimization problem by solving for the best action sequences, consider instead the task of finding a stochastic policy, which maps states in the ocean to distributions over control actions to maximize the total reward. This is similar to finding policies for Markov Decision Processes but has a different reward structure. Techniques for solving MDPs that rely upon value functions, such as dynamic programming [19], or reinforcement learning approaches [20] are not suitable. We instead focus on techniques based on Monte Carlo rollouts to evaluate the efficacy of a given policy.

One technique to find the best rollout is to start with a random stochastic policy, and, over many crude Monte Carlo rollouts, store the rollout with the highest reward. However, the optimal reward is a low probability event, and it will take prohibitively many rollouts to find a decent solution. Monte Carlo Tree Search (MCTS), on the other hand, combines multi-armed bandit techniques with graph based search to efficiently guide Monte Carlo rollouts to maximize expected reward, and is often used in playing adversarial games such as Go [21]. MCTS is readily extensible to MDP-like problems [22], [23]. When implementing MCTS, we use UCB1 for the selection procedure of MCTS [22]. We also use a graph based structure to store search nodes to save on memory requirements [23]. Use of UCB1 as a selection algorithm requires that rewards be upper bounded by a constant, which is unknown. The algorithm stores the maximum reward discovered and scales all rewards by this constant at each selection phase. However, the selection algorithm still requires experimental tuning of the parameter C which establishes tradeoffs between exploration and exploitation.

One can generate a discrete stochastic ocean motion model by selecting a discrete set of actions (depth and thrust velocities) and dividing the ocean into rectangular cells. Placing particles evenly spaced in each cell, one can track the motion of the particles according to the control action and ocean dynamics approximating the discrete transition probabilities of the system. While discrete rollouts may not be feasible trajectories, they approximate the continuous system's performance well enough, and the discrete stochastic policies can be used to control the continuous system, generating feasible paths.

VI. Results

To evaluate the performance of SGA, we used a NCOM model of the Gulf of Mexico. This model gives an ocean current forecast at a two kilometer resolution, every three hours for a week (168 hours) as a vector field of ocean currents. In this work, we assume that the vehicle can maintain its depth and so assume that the vertical current is zero. Additionally, we assume that the vehicle can provide a maximum thrust of 0.1 meters per second. This NCOM model also contains ocean temperature predictions at the same 2 km resolution. We use this temperature to generate a simulated information field over the ocean (see Figure 1) which can correlate with ocean phenomena such as algae blooms. To do this we assume that there is some desired temperature T_0 and use the equation:

$$I(x) = \begin{cases} e^{-a_p(T-T_0)} & \text{if } T \geq T_0 \\ e^{-a_n(T_0-T)} & \text{if } T < T_0 \end{cases}$$

where a_p is a scale factor for the positive case and a_n is a scale factor for the negative case. We calculate T_0, a_p, and a_n so that 20 percent of the information lies above a threshold b, which was selected to be 0.5.

To generate starting regions for the vehicles we split the gulf into seven different regions, three degrees of latitude and longitude on a side. Three of these regions were used for parameter tuning, and four were used for testing our algorithms. The four regions used for testing are shown in Figure 1. In each of these regions we selected a square roughly 20 kilometers a side in the center as the possible starting location area.

From the 3 regions used for parameter tuning, we selected the maximum number of iterations as 50, a $\sigma_d = 100$ (represented in meters), $\sigma_t = 5$ (represented in hours), and a $\sigma_v = 0.02$ (represented in meters per second). Additionally, we found the number of noisy paths $K = 20$ to provide good results. A relatively small number of noisy paths gives better results by allowing the information from more informative paths to more significantly influence the gradient. Due to the randomness inherent in the gradient estimations of the information function, we optimized each action sequence five times and selected the highest scoring action sequence from those.

For tuning MCTS exploration/exploitation parameter C we tested various values of $C \in [10^{-3}, 10^{1}]$ over 21 logarithmically spaced points in the training dataset. Note that MCTS's performance is sensitive to selection of C and the best vs. worst performance varied by 20% over the parameter space. Setting $C = 0.025$ yielded the best performance and was used in testing.

To test the multi-vehicle coordination aspect of the algorithm, we used teams of size 10. Inside the ∼20km square starting regions we generate four different locations with two locations containing four vehicles each and two locations containing one vehicle each. This was done to ensure that there was ample opportunity for the vehicles to interact while allowing the vehicles to start slightly dispersed. When all the vehicles started from the same location, there was not enough time over the week deployment to see diversity in paths.

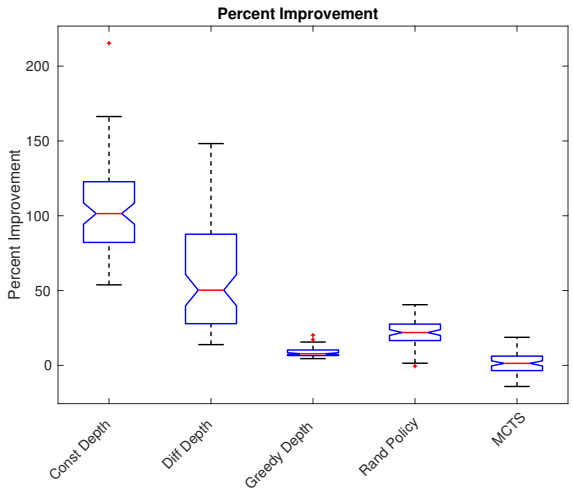

Fig. 2: Percent improvement offered by SGA with respect to all comparison algorithms. We are able to offer a large improvement over the simple baselines. MCTS is able to perform comparably to our algorithm, however it requires approximately 5.2x the computation. See Figure 3 for a more detailed look at the results for the final three methods.

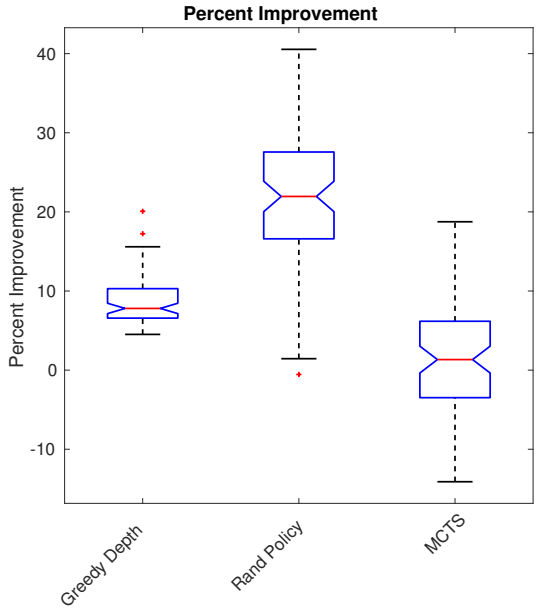

Fig. 3: Percent improvement over the three best performing comparison algorithms from Figure 2. SGA is able to improve over greedy depth selection (8.63%), Random Policy (21.58%) and MCTS (1.50%).

The results from our tests can be seen in Figures 2 and 3 and Table I. These show the percent improvement offered by SGA with respect to the other methods. Unsurprisingly, a default policy of having all vehicles stay at the same depth performs poorly, especially when some vehicles are started at the same location. Our optimized solution is able to offer a 104% improvement on average and can offer much larger improvements in environments with larger differences in ocean current magnitudes and directions.

A default policy of equally distributing the vehicles through the water column performs reasonably well. During

testing, it was observed that this policy allows the vehicles to spread out spatially reasonably well but did not allow the team to prioritize information rich sections of the ocean. Our optimized solution offered an average of 61.00% improvement over equally distributing the vehicles.

A greedy depth selection method is effective at gathering information in these environments. However, our approach is still able to offer an average improvement of 8.63% over this greedy selection method. SGA is able to intelligently consider interactions between different actions within the sequence. While greedy depth selection is able to compute these plans very quickly (on the order of a second), there is a significant financial trade-off for any performance improvement. Daily deployment costs from a crewed surface ship can be upward of $30,000 dollars (see [1]) while a minute of cloud computing time costs less than $1. Even if there is only one day of deployment for a 30 day mission using an autonomous vehicle, the amortized cost is still $1000 per day for the mission. Put another way, to gather the same amount of information required for a mission, 10 vehicles using the proposed approach gather the same amount of information on average that about 10.9 vehicles gather using the greedy method. The effective deployment costs of each mission therefore increases on average by 9% ($90 per day) which dramatically outweighs the additional 1-2 minutes of compute time ($1-$2).

Using a random policy results in surprisingly good performance. The random policy has an advantage over some of the other simple policies by being able to choose a different depth at each action. However, this selection is not made in an intelligent way. Thus, the greedy depth selection, despite being only able to select a single depth for a vehicle, outperforms the random policy. Our optimized solution is able to offer an average of 21.58% improvement.

The comparison method most competitive with SGA is MCTS, where we are only able to offer a 1.5% improvement. However, this is done with an average computational time of 68.09 seconds in comparison with the 354.21 seconds average computational time required by MCTS, which can be seen in Figure 4. This represents a 5.2-fold increase in computational time required by MCTS in comparison with SGA. Future work will look at the effects of parallelizing the sampling process has on computational time.

An example of the paths found by MCTS (green, x's) and SGA (magenta, o's) is shown in Figure 5. SGA is able to plan trajectories which collect information from the small field seen at the end of the path at approximately 21.85 latitude and 265.85 longitude while MCTS has trouble finding this extra information through its random rollouts. The vehicles are able to intelligently balance between spreading out to find information and grouping up to exploit high information regions.

One interesting observation from our testing is that this percent improvement is highly environment dependent. In two environments (1 and 4) SGA offers around the expected 1.5% improvement. However, in environment 2, SGA offers about a 10% improvement over MCTS, while in environment 3 MCTS offers a 3% improvement over the optimizer. Future work will look more at identifying what features of these environments lead to these disparities and what situations each method is better suited for.

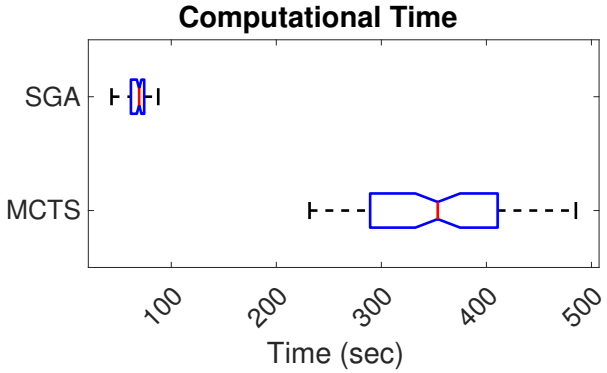

Fig. 4: Computational time taken by the MCTS and SGA. Note that SGA takes 68.09 second on average while MCTS takes 354.21 seconds which represents a 5.2 fold increase.

TABLE I: Percent Improvement by SGA versus competing algorithms

Const Depth	Diff Depth	Greedy Depth	Rand Policy	MCTS
104.10	61.00	8.63	21.58	1.50

VII. CONCLUSIONS

In this work, we showed that by using a novel action sequence representation and our SGA algorithm we are able to improve upon a greedy baseline. We also compared to a computationally intensive Monte Carlo Tree Search Method, which performs comparably with SGA, but requires 5.2x the amount of computation on average. By approximating the gradient with random roll-outs we were able to efficiently search the space and quickly improve the path. We presented results for a NCOM model, which closely approximates what the vehicle would experience in the ocean.

Future work should look in a number of directions. First, the current computational bottleneck of the algorithm is the numerous calculations of the information gathered by the team of vehicles and to improve the efficiency of the algorithm, methods for speeding up this calculation should be investigated. Second, the algorithm is sensitive to the starting conditions passed to the optimizer. To help ensure robust performance, methods for helping remove this dependency should be investigated, such as using random restarts or other methods for guiding the search. Third, while there are high quality ocean prediction models, they are subject to some amount of uncertainty. Future work should look at methods for reducing the effects that this uncertainty could have on the team's performance. Lastly, methods for allowing the optimizer to find narrow solutions should be investigated. One method that shows promise here is utilizing an adaptive sampling technique when constructing the perturbations. Rather than treating σ_d, σ_t, and σ_v as constant, they can vary in response to the environment.

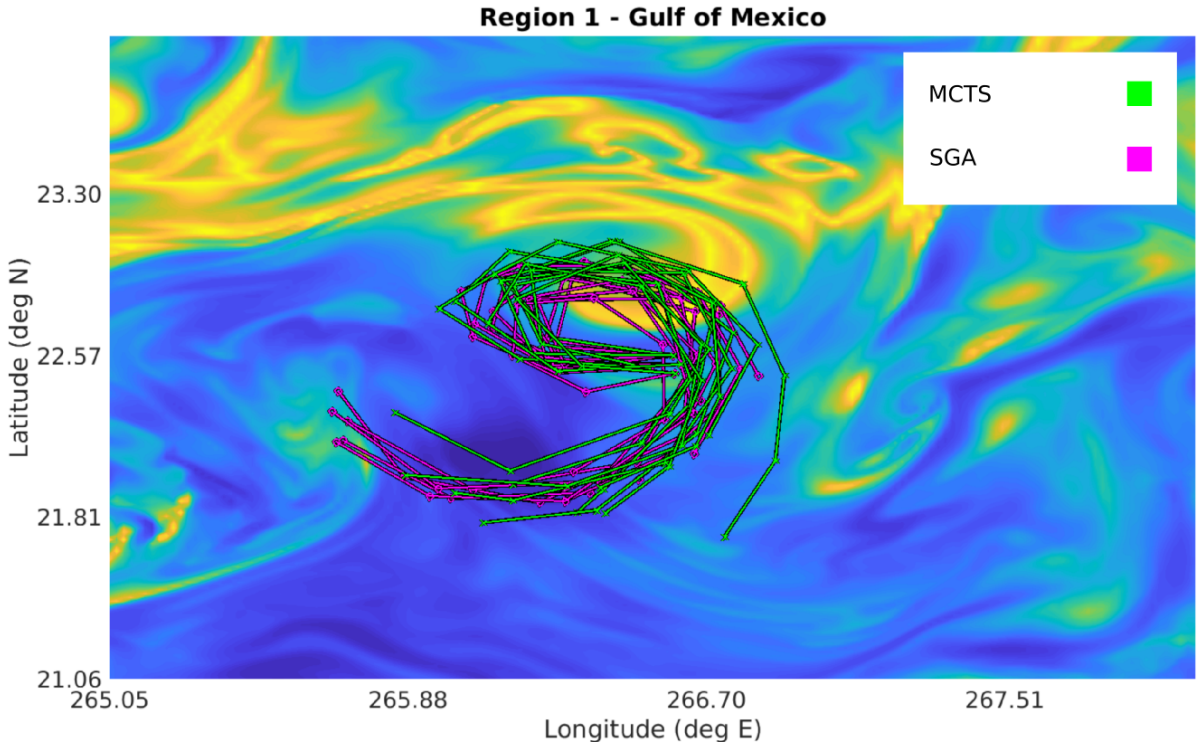

Fig. 5: Paths from MCTS and SGA for one deployment in Region 1 from Figure 1. MCTS is in green and x's and SGA is in magenta and o's. Lighter indicates more information and vehicles start close to the center of the figure. SGA is able to utilize the ocean currents to find the information at the end of the path (left side of the figure) while MCTS is unable to find this path using its random rollouts.

REFERENCES

[1] National Research Council, *Science at Sea: Meeting Future Oceanographic Goals with a Robust Academic Research Fleet*. Washington, DC: The National Academies Press, 2009.

[2] R. A. Smith, Y. Chao, P. P. Li, D. A. Caron, B. H. Jones, and G. S. Sukhatme, "Planning and implementing trajectories for autonomous underwater vehicles to track evolving ocean processes based on predictions from a regional ocean model," *The International Journal of Robotics Research*, vol. 29, no. 12, pp. 1475–1497, 2010.

[3] J. Gould, D. Roemmich, S. Wijffels, H. Freeland, M. Ignaszewsky, X. Jianping, S. Pouliquen, Y. Desaubies, U. Send, K. Radhakrishnan, *et al.*, "Argo profiling floats bring new era of in situ ocean observations," *Eos, Transactions American Geophysical Union*, vol. 85, no. 19, pp. 185–191, 2004.

[4] J. Binney and G. S. Sukhatme, "Branch and bound for informative path planning," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2012, pp. 2147–2154.

[5] K. H. Low, J. M. Dolan, and P. Khosla, "Active markov information-theoretic path planning for robotic environmental sensing," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 753–760.

[6] G. A. Jacobs, H. S. Huntley, A. Kirwan, B. L. Lipphardt, T. Campbell, T. Smith, K. Edwards, and B. Bartels, "Ocean processes underlying surface clustering," *Journal of Geophysical Research: Oceans*, vol. 121, no. 1, pp. 180–197, 2016.

[7] J. A. Caley and G. A. Hollinger, "Data-driven comparison of spatio-temporal monitoring techniques," in *OCEANS'15 MTS/IEEE Washington DC*, 2015.

[8] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," 2009, pp. 489–494.

[9] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[10] M. Mukadam, X. Yan, and B. Boots, "Gaussian process motion planning," 2016., pp. 9–15.

[11] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 4569–4574.

[12] D. Jones and G. A. Hollinger, "Planning energy-efficient trajectories in strong disturbances," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2080–2087, 2017.

[13] K.-C. Ma, L. Liu, and G. S. Sukhatme, "Informative planning and online learning with sparse gaussian processes," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2017, pp. 4292–4298.

[14] R. N. Smith and V. T. Huynh, "Controlling buoyancy-driven profiling floats for applications in ocean observation," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 3, pp. 571–586, 2014.

[15] A. Molchanov, A. Breitenmoser, and G. S. Sukhatme, "Active drifters: Towards a practical multi-robot system for ocean monitoring," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2015, pp. 545–552.

[16] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2017, pp. 1714–1721.

[17] G. A. Hollinger, S. Yerramalli, S. Singh, U. Mitra, and G. S. Sukhatme, "Distributed data fusion for multirobot search," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 55–66, 2015.

[18] A. K. Agogino and K. Tumer, "Analyzing and visualizing multiagent rewards in dynamic and stochastic domains," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 2, pp. 320–338, 2008.

[19] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[20] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[21] G. Chaslot, "Monte-carlo tree search," Ph.D. dissertation, Maastricht: Universiteit Maastricht, Maastricht, Netherlands, 2010.

[22] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European Conference on Machine Learning*, vol. 6. Springer, 2006, pp. 282–293.

[23] B. Bonet and H. Geffner, "Action selection for mdps: Anytime AO* versus UCT." in *AAAI Conference on Artifical Intelligence*, 2012.