Hanna Gratch
August 2016

# Planning Autonomous Underwater Vehicles with Ensembles

Introduction:

Less than five percent of the ocean has been discovered (NOAA).[1] Considering the fact that around 70% of the earth is covered with water, there is much more to uncover about our world. The ocean plays a crucial part in regulating the climate, and is thus important to study as scientists learn more about sustaining the Earth. For example, the carbon cycle, which is largely controlled by the ocean, circulates carbon dioxide (a greenhouse gas). Greenhouse gases directly affect the climate. By sampling the water, scientists can learn more about the ocean. Autonomous underwater vehicles (AUV's) can be employed to obtain data such as temperature and salinity. While these vehicles allow scientists to collect more data than manually collecting samples, they are expensive to use. Additionally, due to the vastness of the ocean, it is nearly impossible to cover every area. Therefore, these vehicles must perform efficiently. This efficiency relates to how accurately the vehicles follow an intended path in the ocean.

AUV's and gliders (Seagliders) are used to carry out experiments. There are various differences between these vehicles. AUV's need to be sent out and retrieved often (they can stay out for up to one week), whereas Seagliders have more endurance and can last for months in the ocean. However, AUV's have both horizontal and vertical control and Seagliders have only horizontal control. While AUV's have thrust power, Seagliders rely on their wings and shifting buoyancy to move in the water. Additionally, AUV's are more expensive than gliders. These are a few factors that scientists must consider before employing a certain instrument.

The gliders are affected by currents due to their slow speed (they lack thrust power). Therefore, planned paths are not perfectly executed because of the changing currents. By using simulations, we can determine the most efficient path without having to worry about the cost of physically testing an AUV or glider. Employing these vehicles takes a considerable amount of effort with tasks ranging from sending out/retrieving the asset, to tracking the asset and ensuring it is functioning properly. It is important to plan the vehicles with simulations to make the real testing process more effective.

The Regional Ocean Modeling System (ROMS) is a model that predicts the ocean properties including currents, temperature and salinity (1). From the ROMS  model, two types of models are generated for simulations: nature model and planning model.

The nature model is the most accurate model, and planning models are less accurate models. Two models are used because the currents cannot be accounted for exactly, meaning that there will always be a margin of error between the most accurate simulation and the real world model. The models are used to reflect the simulation errors between the most accurate model and the ocean model (real ocean).

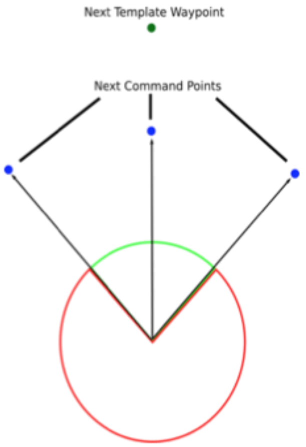
Rather than evaluating a single planning model, an ensemble of them can be used. An ensemble consists of a group of planning models with slightly varying initial conditions. With ensemble planning, these planning models are evaluated against each other and ranked based on the most accurate model. In this paper I examine if the use of ensembles can improve the

[1] http://oceanservice.noaa.gov/facts/exploration.html

robustness of model performance using Seagliders and if different scoring rules and planning methods can further enhance the performance.

**Planning Algorithms:**

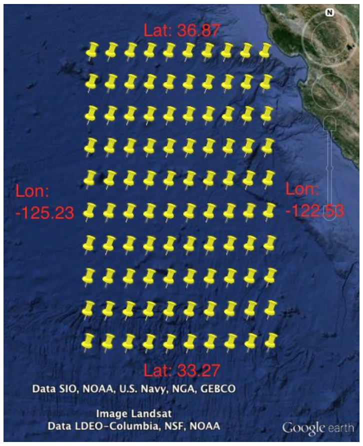
When generating runs for planning paths, a variety of algorithms were used: GreedyBeamSearch, BeamSearch, and BestFirstBeamSearch. These search algorithms attempt to stay on the intended path by trying to move to the next waypoint. When the asset resurfaces, it sets out a number of beams within a specified search angle (e.g., see the image to the right, from (2)), and chooses the point that will lead to the next waypoint. GreedyBeamSearch only looks ahead one dive and chooses the point closest to the next waypoint. BeamSearch and BestFirstBeamSearch both look ahead two dives, but BestFirstBeamSearch also has a give-up condition. This allows the asset to estimate an arrival time to the next waypoint and give up if it doesn't reach the waypoint in that amount of time. A variable named EpsilonTime allows a user to specify any additional time to the ETA if needed.



Baselines were also used to ensure that the planning methods performed better than limited planning. Baselines ignore the current, so in theory they would have a higher score. Angle Baseline simply finds the angle between the current point and the end point and continues on that same path the entire time. Endpoint Baseline finds the angle, but every time it resurfaces, it corrects the angle.

**Planning to Nature Model Comparison:**

The following experiment was conducted to evaluate the performance of ensemble methods and various ranking methods. In the ROMS model, I defined a 10x10 grid and divided all latitude and longitude between these 100 points (see image to right, from (3)). Each planning/nature model evaluation ran through each of these points, so each evaluation had 100 runs. The score for these were all averaged to create an overall score for each planning/nature model evaluation. Total scores for each model were calculated by then averaging these scores for each planning model evaluation. This was done through standard numpy arrays and libraries. The process is visualized in Figure 2.



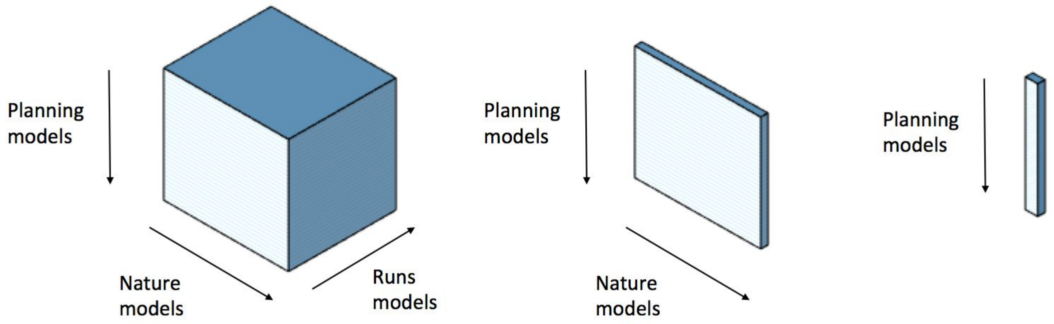


Figure 2: illustrates the steps in converting all planning/nature/run data into planning model scores

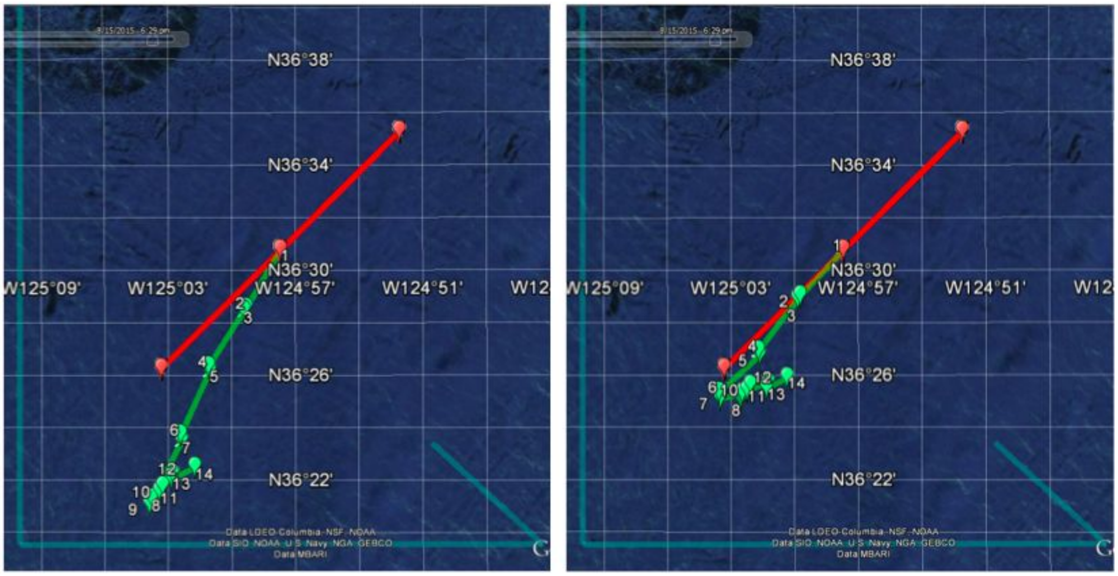


Figure 3: Illustrates a planning runs for the AngleBaseline (left) and BestFirstBeamSearch (right). The read lines indicate the intended path and the green lines indicate the actual path taking in the simulated execution

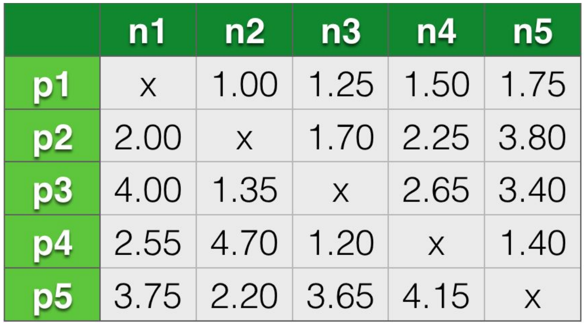
In order to evaluate the planning models, a score calculator and a root mean square error (RMSE)  calculator were needed. The score calculator generates a score based on the distance away from the target point at each node. The underwater vehicle gives a new data point every time it resurfaces. The farther away the planning model is at each node, the less accurate it is. Therefore, the lower the score, the better the model.

The RMSE calculator determines the variance of the ensemble. If there is a larger variance, we are less confident that the data was sufficiently generalized over all of the models. Thus, a greater RMSE entails more runs. The RMSE calculator was not used for the actual ranking of the different planning models. When writing the code for the RMSE calculator, I implemented two different ways. The first way took the RMSE of ‘u,’or zonal current, and the RMSE of ‘v’, or meridional current, using the standard RMSE formula:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

After the code calculates the RMSE for u and v, it averages the two and returns the value. The calculator also computes the RMSE by calculating the magnitude of the vector between the u and v vectors, and then using the standard RMSE formula on the summation of the magnitudes (there are multiple because the calculator gets the values from each resurfacing).

The first ensemble I used consisted of five planning models. After experimenting with five planning models, I evaluated each planning model against the a second ensemble of 8 models. These models were the same planning models but were treated as nature models to evaluate the performance of the first ensemble. For example, the first planning model was evaluated against the remaining seven planning models. The score calculator was used to evaluate these comparisons (see Table 1).



| | n1 | n2 | n3 | n4 | n5 |
|---|---|---|---|---|---|
| p1 | x | 1.00 | 1.25 | 1.50 | 1.75 |
| p2 | 2.00 | x | 1.70 | 2.25 | 3.80 |
| p3 | 4.00 | 1.35 | x | 2.65 | 3.40 |
| p4 | 2.55 | 4.70 | 1.20 | x | 1.40 |
| p5 | 3.75 | 2.20 | 3.65 | 4.15 | x |

Table 1: Example score matrix for an ensemble of five planning models

After every model had been evaluated against each other, I wrote rank the planning models based on their scores in this table. I ranked the models using three different ranking rules that ranked models based on their score across the row of scores in Table 1:

- Mean: sorted in ascending order by average score across each row
- Min min: sorted in ascending order by minimum score in each row
- Max min: sorted in ascending order by the maximum score in each row

For "mean," I simply averaged all of the scores from each planning model and ranked them from lowest to highest score. For "min min," I looked at each of the planning model scores and picked the lowest scores. I then ranked the planning models based on which had the lowest scores. For "max min," I looked at the highest scores for each planning model and ranked them based on the best (lowest) of those scores.

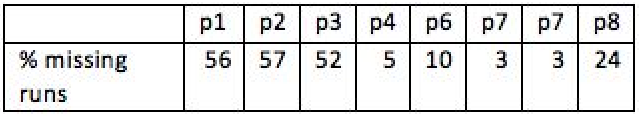
Because each ranking method was different, different planning models were ranked first. Taking these three models, I then evaluated them against a new ensemble to determine the best model. After evaluating these models, I averaged each model's scores and ranked these.

A "random," "ROMS mean," and "worst" ranking were also created as a control group to give more insight into the potential advantages of the scoring rules (i.e., are they better than picking the worst or simply picking a random strategy?). In random, all of the scores of the evaluations on the second ensemble were averaged.

In ROMS mean, a mean of all of the planning models for the ensembles (16) was evaluated against the second ensemble (9-16). In other words, in the score calculator, the ROMS mean was assigned to the planning model variable, and the models in the second ensemble were set as the nature model. In worst, the highest (or worst) scores of each model were ranked.

**Results and Discussion:**

Two different data sets were used to produce the following results. The first used all of the available data from the two 8-model ensembles. There was missing data from run crashes of the planning algorithm, however, which caused some models to be tested on fewer runs than other models (for example, some models might be tested on 100 runs, whereas other models might only be tested on 50 runs). Table 2 summarizes what data was missing. The results of the full data set are shown in Figure 4.



| | p1 | p2 | p3 | p4 | p6 | p7 | p7 | p8 |
|---|---|---|---|---|---|---|---|---|
| % missing runs | 56 | 57 | 52 | 5 | 10 | 3 | 3 | 24 |

Table 2: shows the number of runs missing for each model

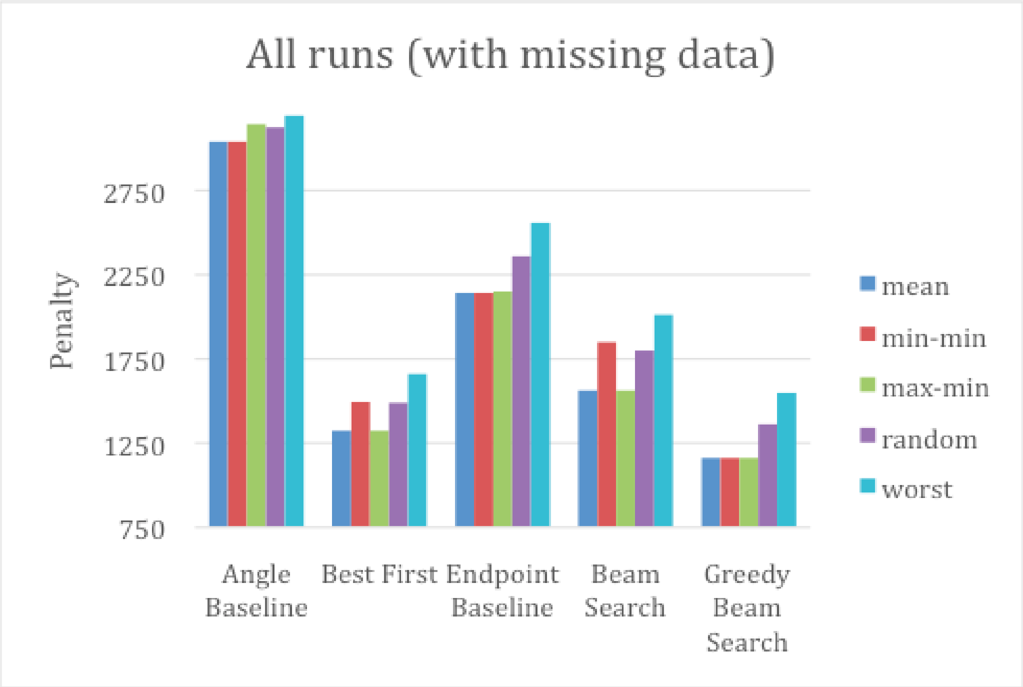
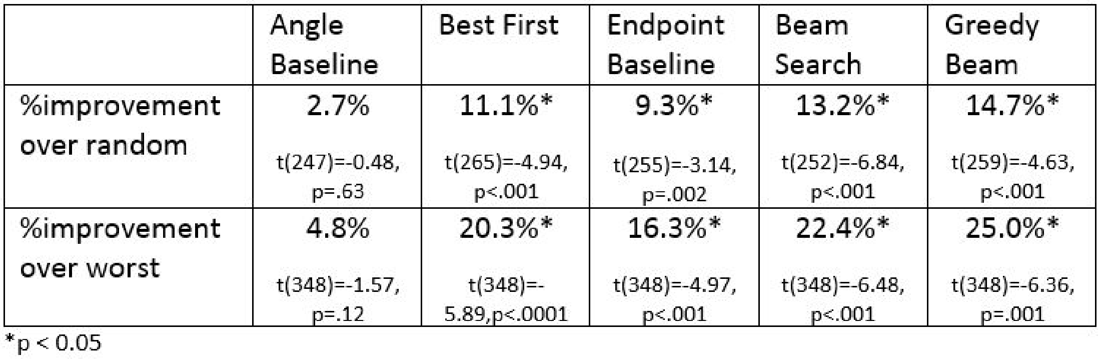


Figure 4: Results using all data. Figure shows the average penalty-score on the second ensemble for all <ranking-rule x planning algorithm> combinations.

Results from the first data set suggest that the mean and max-min ranking rules always improved performance, regardless of the planning algorithm, and the GreedyBeamSearch algorithm performed best. An independent sample t-test was run to verify the significance of the improvements from using an ensemble to select the best planning model. The performance of the model chosen by the mean ranking rule on the first ensemble was tested on the second ensemble. It was also contrasted with the worst model and a random planning model. The improvements were significant for all planning algorithms except AngleBaseline (see Table 3).



| | Angle Baseline | Best First | Endpoint Baseline | Beam Search | Greedy Beam |
|---|---|---|---|---|---|
| %improvement over random | 2.7%<br>t(247)=-0.48, p=.63 | 11.1%*<br>t(265)=-4.94, p<.001 | 9.3%*<br>t(255)=-3.14, p=.002 | 13.2%*<br>t(252)=-6.84, p<.001 | 14.7%*<br>t(259)=-4.63, p<.001 |
| %improvement over worst | 4.8%<br>t(348)=-1.57, p=.12 | 20.3%*<br>t(348)=-5.89,p<.0001 | 16.3%*<br>t(348)=-4.97, p<.001 | 22.4%*<br>t(348)=-6.48, p<.001 | 25.0%*<br>t(348)=-6.36, p=.001 |

*p < 0.05

Table 3: Improvement to each planning algorithm when using an ensemble to select the planning model (compared with random or worst model; all data)

One issue with these results raises a concern about their validity. Since the two baseline planning algorithms (AngleBaseline and EndpointBaseline) use limited planning, the scores for the different ranking methods should all average out to be the same. Thus, the first data set should reflect the fact that not all planning models were tested on the same set of runs.

The second data set was created to correct for the missing data and the resulting bias of the scores. The data is the same as the first set, but includes only runs 7-9, 60-81, 91 and 98. These were the runs that were available for all of the models. While the first data set had more data, the second set avoids any possible bias that could occur (using all data means that some models are tested on different lat/lon points in the ocean than other models). Results using only complete runs are shown in Figure 5.

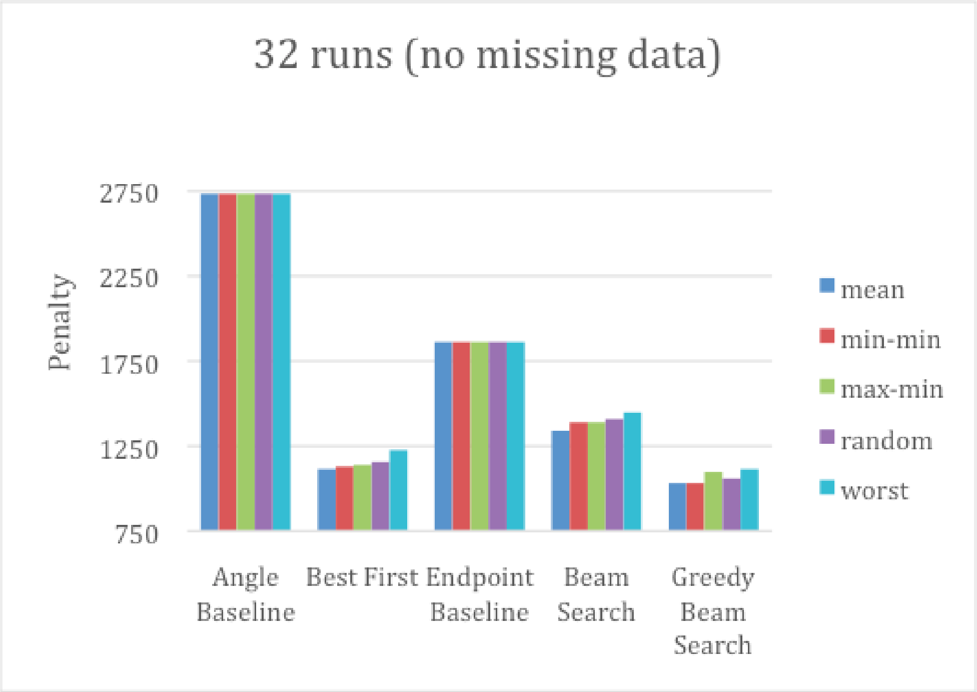


Figure 5: Results using only complete runs. Figure shows the average penalty-score on the second ensemble for all <ranking-rule x planning algorithm> combinations.

With this data set, it is clear that consistent data has more accurate results. All of the baseline algorithms showed the same score, which indicates more accuracy. In this graph, GreedyBeamSearch performed the best. This reveals that the mean scoring rule is consistently the best. A paired-sample t-test was run to verify the significance of the improvements from the ensemble-ranking procedure. The performance of model chosen by the mean ranking rule on the first ensemble was tested on the second ensemble, and contrasted with the worst model and a random planning model. The improvements were significant for all planning algorithms except the two Baseline algorithms (see Table 4).

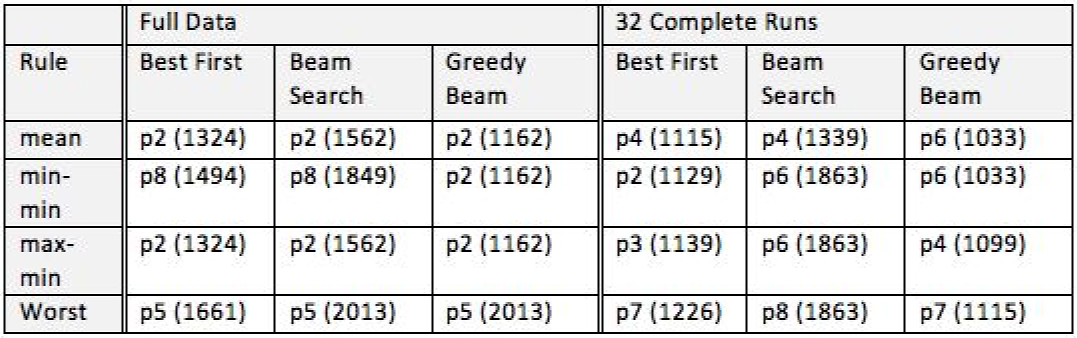
| | Angle Baseline | Best First | Endpoint Baseline | Beam Search | Greedy Beam |
|---|---|---|---|---|---|
| %improvement over random | 0.0%<br>t(255)=0,<br>p=1.0 | 3.5%*<br>t(255)=-3.10,<br>p=.002 | 0.0%<br>t(255)=0,<br>p=1.0 | 5.0%*<br>t(255)=-3.29,<br>p=.001 | 2.5%*<br>t(255)=-3.09,<br>p=.002 |
| %improvement over worst | 0.0%<br>t(255)=0,<br>p=1.0 | 9.1%*<br>t(255)=-4.04,<br>p<.0001 | 0.0%<br>t(255)=0,<br>p=1.0 | 7.6%*<br>t(255)=-5.47,<br>p<.001 | 7.4%*<br>t(255)=-2.38,<br>p=.02 |

*p < 0.05

Table 4:Improvement to each planning algorithm when using an ensemble to select the planning model (compared with random or worst model; no missing data)

For all planning algorithms (excluding the baselines), the use of ensembles improved their performance. As seen in Table 4, the percentage of improvement against random and worst for every planning algorithm was statistically significant. The difference between ranking methods themselves was more modest. However, using these ranking methods help improve even the best planning algorithm, GreedyBeamSearch. Therefore, planning with ensembles have significant value.

To give more insight into the reasons behind the differences in performance, I did further analysis. Table 5 shows which planning model was selected for each planning algorithm by each ranking rule. Planning model 2 was always selected by the mean and maxMin rules when full data was used, but different models were selected when only the 32 complete runs were used for selection. Planning model 5 always performed worst when all runs were included, but different models performed worse when only the portion of complete runs were used.



| | Full Data | | | 32 Complete Runs | | |
|---|---|---|---|---|---|---|
| Rule | Best First | Beam Search | Greedy Beam | Best First | Beam Search | Greedy Beam |
| mean | p2 (1324) | p2 (1562) | p2 (1162) | p4 (1115) | p4 (1339) | p6 (1033) |
| min-min | p8 (1494) | p8 (1849) | p2 (1162) | p2 (1129) | p6 (1863) | p6 (1033) |
| max-min | p2 (1324) | p2 (1562) | p2 (1162) | p3 (1139) | p6 (1863) | p4 (1099) |
| Worst | p5 (1661) | p5 (2013) | p5 (2013) | p7 (1226) | p8 (1863) | p7 (1115) |

.

Table 5: Shows the planning model selected by each scoring rule for each planning approach and its mean performance on ensemble 2 (models 9-16 used for testing data)

To give further explanation of why different rules were selected in different datasets, I analyzed the performance of the best planning algorithm in more detail. Figure 6 shows the performance of each model across the two datasets (all runs and only complete runs), just looking at the performance of the best-performing planning algorithm (i.e., greedy beam search). To generate these graphs, I found the minimum, the mean, and the maximum penalty across all runs for a given nature model, and then averaged these values across all 8 nature models in the second ensemble.

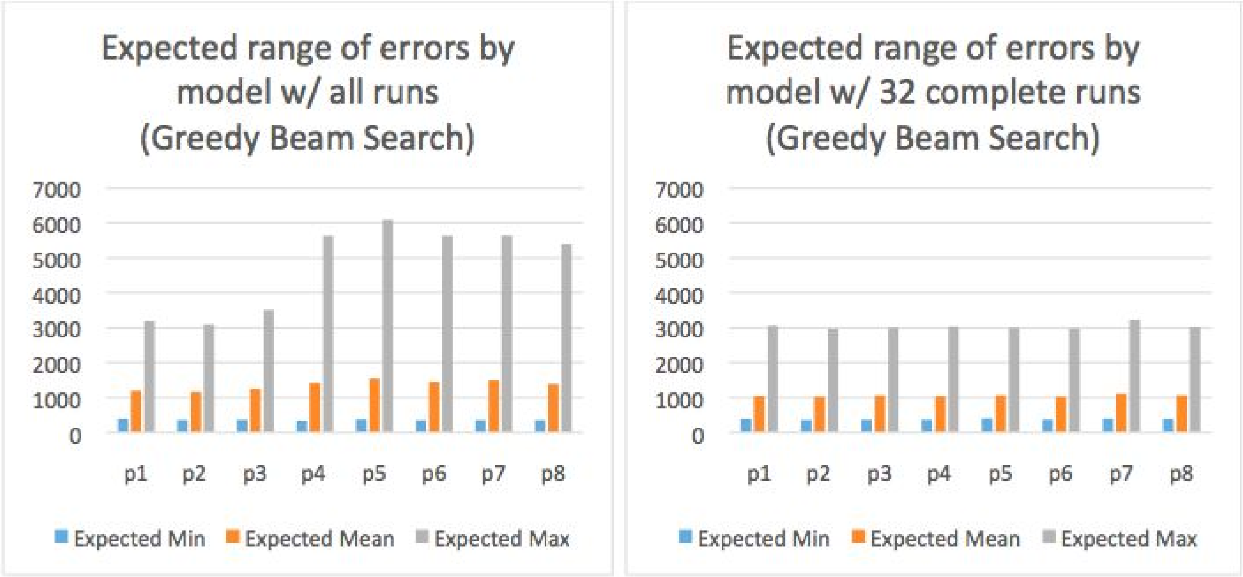


Figure 6: Shows the minimum, mean, and maximum expected penalties by model.

The maximum possible error appears to be greater for models p4 through p8 when we use all the data (the left-most graph in Figure 6), yet these models exactly correspond to the models that have the most missing runs (see Table 2).

This is most likely due to the fact that the data from the missing runs would have contributed to a bigger error. Thus, p1, p2, and p3 appear to perform better because they lack the missing data. When we look at the graph on the right in figure six, all of the models have similar error ranges. Due to these differences, it is important to understand the results that have missing data, and evaluate the credibility of those results.

Finally, it is important to look at the differences between planning models. When looking at the errors made by the algorithms at different points in the ocean (show by a heat-map in Figure 7), the planning models have similar patterns of errors. This could be an explanation of the ineffective nature of the ranking methods. If the planning models are too similar, the ranking methods will output similar scores.

The heat-map also reveals that GreedyBeamSearch results in the smallest error (the cooler the color, the less error there is). This reinforces the prior graph that GreedyBeamSearch is the best planning algorithm out of the five.

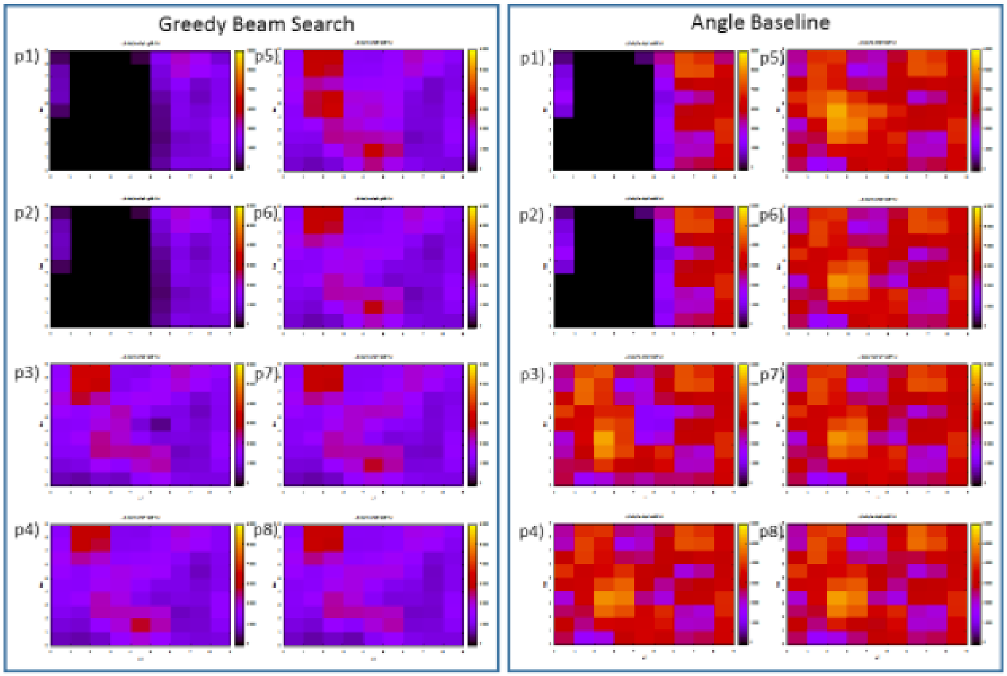


Figure 7: Illustrates two sets of heat-maps that represent the penalty score for planning algorithms across the 10x10 ocean grid. Each heat-map shows the error at each of the 100 target points. Different graphs illustrate different combinations of planning algorithms (GreedyBeamSearch on the left and AngleBaseline on the right) and different planning models (p1, p2, etc.). Red means more error. Black means missing data.

**Conclusion**

It can be concluded from the graphs that the planning algorithms were all improved with the use of ensembles. While the variance in ranking-method performance (the performance of ensembles with ranking methods) is small when compared to planning algorithm variance, the fact that they improved all of the planning algorithms reveals the benefit of using ensembles. If planning models were to have greater differences, the ranking methods in turn would have greater variability.

**Related Work:**

Another paper also looked into planning with ensembles, but focused on a different way of improving models. The project planned with different ensemble types and compared their performance to four individual models. Two different weighted average methods were used to place a weight for each individual model forecast. The equal weighting (EW) ensemble is a simple average of the four models, whereas the objective weighting (OBJ) ensemble is a weighted average of the models, and takes the individual model performance into account. The weight depends on the individual model performance from an earlier training period. Each model type improved the performance by eliminating or minimizing errors in planning. While the OBJ ensemble performed better than all of the individual models. These results suggest that we might improve ensemble performance further (in our experiment) with a similar weighting method.

**Future Work:**

In the future, different assets such as AUV's could be tested (this project focused on the Seaglider) to see how the ensembles perform. The parameters for the runs can also be tweaked to see how different factors affect the scores and which parameters lead to the best performance. For example, the search angle for the planning algorithms can be changed. The effect of ensemble size and different ranking methods (such as minimax) can be explored as well. Finally, ensemble planning should function for dynamic assets instead of trying to maintain a single position.

**References:**

1. Chao Y, Li Z, Farrara J, McWilliams JC, Bellingham J, Capet X, et al. Development, implementation and evaluation of a data-assimilative ocean forecasting system off the central California coast. Deep Sea Research Part II: Topical Studies in Oceanography. 2009;56(3):100-26.
2. Branch A, Troesch M, Chien S, Chao Y, Farrara J, Thompson A. Evaluating scientific coverage strategies for a heterogeneous fleet of marine assets using a predictive model of ocean currents. 2016.
3. Troesch M, Chien S, Chao Y, Farrara J. Evaluating the Impact of Model Accuracy in Batch and Continuous Planning for control of marine floats. 2016.
4. Wang X, Chao Y, Thompson DR, Chien SA, Farrara J, Li P, et al. Multi-model ensemble forecasting and glider path planning in the Mid-Atlantic Bight. Continental Shelf Research. 2013;63:S223-S34.