Developing An Optimized Bayesian Search Algorithm For Scavenger Hunt

Santi Dasari
Student Researcher

Richa Gadre
Student Researcher

Esther Yoon
Student Researcher

***Abstract*—Object retrieval is a necessary function for many service robots. For tasks such as tidying up a house or packing a suitcase, the robot must be able to track down objects within a dynamic search space in an efficient manner. The scavenger hunt problem in robotics has the goal of optimizing the search for objects with uncertain positions. Another research group at The University of Texas at Austin studied how several different algorithms performed given a model of this problem [1]. They found the Exhaustive Bayesian Search algorithm traverses the shortest expected distance, but that its computation time for the best path to take scales poorly in spaces with over eight locations.**

In this paper we extend the work done on the scavenger hunt problem by developing an algorithm that provides accurate predictions similar to those of the Exhaustive Bayesian Search Algorithm without the time delays associated with larger map sizes. Our algorithm, the Timed Bayesian Search, completes its path cost calculations in a breadth-first, step-wise manner, and execution is halted after it exceeds a preset time limit. We tested the Timed Bayesian against the second-best search algorithm, the Probability-Proximity search algorithm, to see if performance improves in maps with over eight locations. Our findings indicate that the Timed Bayesian does not perform significantly better than the Probability-Proximity search in this scenario due to some limiting factors. Further optimizations or redesigns are needed to develop this algorithm to accomplish our original goal.

I. Introduction

The Scavenger Hunt problem in robotics concerns object location detection within a frequently changing environment. It motivates developing the search tactics of a robot so it chooses the most efficient retrieval path, updating its stored distribution models of where objects are located as a hunt progresses. In the Scavenger Hunt paper that was submitted to IROS, various search algorithms were formulated and tested to optimize the robot's retrieval: the Proximity-based search, the Probability-based search, the Probability-Proximity search, and the Exhaustive Bayesian search algorithms [1]. These searches were evaluated based on their average distance performance. The Exhaustive Bayesian search was found to perform optimally in smaller environments, from three to eight locations. The problem arises when the environment size increases to contain over eight locations. The Exhaustive Bayesian calculates the expected cost for every possible path permutation. This causes its path cost computation time to grow exponentially with each location that is added, making searches in large environments difficult to conduct. Although it provides accurate path predictions, this time overhead limits its practical use.

In this paper we focus on developing an improved version of the Exhaustive Bayesian search algorithm. The results outlined in the previous work deduced that the Exhaustive Bayesian search algorithm would be the optimal solution if its calculation time delays were reduced. In an attempt remedy this issue, we developed a new algorithm which we call the Timed Bayesian search. Instead of traversing every path exhaustively, the Time Bayesian seeks to reduce path cost calculation time by completing a breadth-first search down each path in a step-wise manner, taking single steps down every path until a preset time limit is reached or every path is fully calculated. Our aim is for the Timed Bayesian to perform at least as effectively as the current Bayesian search for small environments and travel the shortest average distance in environments with over eight locations when compared to the alternate search patterns.

The Timed Bayesian algorithm was implemented and tested by the same abstract simulator that was developed in previous work [1]. By using the same testing script, we can ensure our data communicates comparable results to the original experiment. The simulator tests multiple agents at once on the same randomly generated world, returning the average distance travelled and average run time for each algorithm at the conclusion of a number of specific trials.

The rest of the paper will contain the following. In Section II, we provide background on the algorithm implementations and the results established in the original scavenger hunt experiment. We will then discuss how we implemented our algorithm and how it compares to the Exhaustive Bayesian in Section III. Next, we will detail in Section IV how we tested the Timed Bayesian's performance for small and large environments and interpret our results in Section V. Finally, in Sections VI and VII we will discuss the purpose of our results, posit some possible future extensions to our work, and conclude.

II. Background

Scavenger Hunt is a variation of the NP-stochastic traveling purchaser problem and used in AI conferences to see the progression of autonomous robot performance [2]. To solve this problem, the original scavenger hunt research group developed multiple algorithms and evaluated their performance. The Proximity-based algorithm visits the closest location that may contain a not-yet-found object. If the object is found, then the object's distribution is disregarded. The Probability-

based algorithm visits the location with the highest probability of finding an object. The Probability-Proximity search, which was the second-best algorithm, combines the previous two search algorithms and goes to the closest location with a high probability [1]. The Bayesian search is based on the Bayesian search theory, which begins to search in areas with high probability, and then recalculates each time to find the next highest probability.

The Exhaustive Bayesian search computes the cost of every possible path and after calculating all travel paths, it chooses the path that has the lowest expected distance and repeats this calculation when an object location is changed. The following equation shows what the average distance travelled by the Bayesian agent might be before completing the scavenger hunt where p is the path being travelled and E(p) is expected cost.

$$E(p) = \sum_{A \epsilon P} P(A)K(A, p)$$

For the purpose of our experiment, the pertinent search agents the original group used are the Probability-Proximity agent and the Exhaustive Bayesian agent.

The searches were tested both virtually, on an abstract simulator written in Python 3, and in a physical lab. These algorithms are tested on fully connected graphs, in which a fixed number of objects are randomly placed based on randomly generated probabilities for each object. Each object's probabilities were guaranteed to sum to one. The searches were compared to upper and lower bounds described as follows. The lower bound was the Travelling Salesman search, which goes to the shortest path between all nodes and does not consider a distribution model. The upper bound was the Offline Optimal search, in which the agent is informed of all of the locations of every object, providing it with the information needed to calculate the absolute shortest path. All of the searches described fell between the upper and lower bounds when tested in the software simulation.

The Probability-Proximity search outperformed the others in a large environment, including the Exhaustive Bayesian search as it cannot complete the hunt in a timely manner past eight nodes. However, the Bayesian search algorithm travels a significantly shorter average distance than the Probability-Proximity search in smaller environments [1].

III. Method

In this section, we will describe the design of the Timed Bayesian Search algorithm and how its implementation is meant to improve upon the Exhaustive Bayesian search.

The original scavenger hunt study tested search algorithms with both a physical robot and an abstract simulator. In our experiment, we focused solely on the abstract simulator portion of their work, conducting tests virtually. All code discussed is written in Python 3. The simulator generates worlds with object distributions for an agent to search in; more details can be found in the IROS paper [1]. Our agent extends the existing agent.py class.

The Timed Bayesian Search is a variation on the Exhaustive Bayesian search algorithm, and, if it never exceeds the time limit, will obtain an exhaustive view of the search space. For smaller worlds, this means the Timed Bayesian and Exhaustive Bayesian will always choose the same path, travelling identical distances in a given hunt.

The difference arises in the manner in which the Timed Bayesian progresses through its path cost calculations; it is breadth-first instead of depth-first. Instead of iterating over a path completely before moving to the next path, as the Exhaustive does, it iterates on a single "step" down a path, calculating a partial cost of that path before switching to the next path, and tracking the best path and its associated cost found so far. After completing the cost analysis for one step down every path for each object distribution arrangement, the algorithm verifies if the time limit (a predefined number of seconds) has been exceeded. If it has, it terminates its searching and moves to the currently saved best path, where, after it moves, the world is refactored based on any objects found at the new location, and the search process starts again. If the time limit has not been exceeded, the agent proceeds to take an additional step in each direction, adding to the cost it determined in the previous iterations for that path. In this way, the cost for each path is gradually accumulated, becoming increasingly more precise as more steps are taken.

An important change to note in the Timed Bayesian implementation is the inclusion of a "saved paths" data structure that was not part of the Exhaustive Bayesian. This structure is composed of a list of entries, with an entry for each possible path. Each entry contains the path itself, the accumulated cost of the path (all are initialized to zero), and the possible valid arrangement and hunts associated with that path (all are initially identical, but the hunts, which store what objects are left to be found, change as the agent virtually steps through different paths). This path saving component is required in order to save the state of each path after a step is taken on it, so when the next step is calculated, the correct information may be retrieved.

IV. Experimental Setup

We will now describe the experiments run to evaluate the effectiveness of our algorithm. The tests were conducted using the same Python testing script the original scavenger hunt group developed. The simulator defines a graph world, set of objects, and a randomized distribution model for those objects [1]. This allowed us to test multiple agents at once on the same randomly generated environment. We were able to set the number of locations, or nodes, in the generated environment, the number of hunts ran in a particular environment, and the number of trials ran for each hunt. The test script returned two data points: the average distance the agent travelled and the average run time until it completed a hunt, the average being over all the trials of hunts in the test run. There were a couple questions we wanted to answer through testing.

First, we wanted to verify whether the Timed Bayesian was able to perform to the same level of correctness as the

Exhaustive Bayesian in smaller environments. Since the Timed Bayesian is designed to perform identically to the Exhaustive given enough time, and the search time for environments with under eight locations never exceeds our set time limits, the two Bayesian algorithms should travel the exact same average distance in these cases. To test this, we ran the Timed Bayesian, the Exhaustive Bayesian, and the Probability-Proximity (for comparison) in environments with between three and eight locations. Our reasoning for the upper bound on locations is that eight was the highest number of nodes that the Exhaustive Bayesian search was tested in the original paper due to time constraints. We ran 30 trials, with 10 hunts per trial, for each number of locations.

Second, we wanted to see if the Timed Bayesian search performed more efficiently than the Exhaustive Bayesian search while maintaining accuracy for maps with 9 and 10 nodes. We determined that this would be true under two conditions. One, that the Timed Bayesian could merely run for a large amount of hunts without taking an unreasonable amount of time. Two, that the Timed Bayesian would travel, on average, a significantly shorter average distance than the Probability-Proximity search, which was the named best search for large environments. To test these conditions, we once again ran 30 trials, 10 hunts per trial each for environments with 9 and 10 locations. We ran both the Probability-Proximity agent and the Timed Bayesian agent. For the Timed Bayesian, we tested three different time limits to evaluate what an optimal constraint might be. The time limits we used were 1, 5, and 20 seconds. The former two were used to test time limits applicable to the real-world; the 20 second time limit was used as an upper bound. We were able to see if the first condition described above holds based on the returned average run times, and if the second condition holds based on the average distances travelled.

V. Results

Baseline tests: Figure 1 shows the results of our baseline tests comparing the average distance travelled for the Timed and Exhaustive Bayesian and the Probability-Proximity agents, respectively. The Timed and Exhaustive Bayesian agents performed identically in all cases for these smaller graph sizes, corresponding to the overlapping plotted lines on the figure. This is because the calculation times needed to find the next best path for smaller environments never exceed the time limits set for the Timed Bayesian. Therefore, the Timed Bayesian performs identically to the Exhaustive Bayesian, analyzing each path comprehensively. These results verify the correctness of our algorithm, and show that we met our baseline goal.

Extending tests: We will now analyze the distance and time results collected for environments with 9 and 10 locations.

The results for the average distances travelled by the Probability-Proximity and Timed Bayesian are shown in Figure 2. No units are associated with the distance due to the abstract simulator using number values as weights. We conducted two-tailed, two-sample, equal variance t-tests on

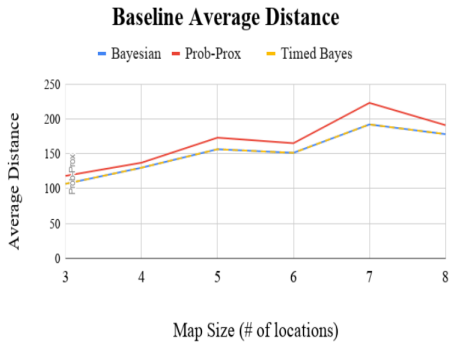

Fig. 1. Comparing average distances travelled for the Probability-Proximity, Timed Bayesian, and Exhaustive Bayesian agents in small environments.

the Probability-Proximity's versus each Timed Bayesian's distances, using a p-value of 0.05 to test for significance. We found that the average distances travelled for the 1 and 5 second time limits, respectively, were significantly longer than that for the Probability-Proximity search for the 10 location trials, the opposite result of what was expected. All other distance differences between the Timed Bayesian and Probability-Proximity agents were non-significant. It appears the 20 second time limit performed better than 1 and 5 seconds and about equally to the Probability-Proximity model. However, even if the 20 second Bayesian was marginally better than Probability-Proximity, this setting was not meant to be reasonable for real-world use. It would thus be impractical to adopt it in actual searches.

The average run times over an entire hunt for the Probability-Proximity and different Timed Bayesian time limit agents are shown in Figure 3. We found that, as expected, the Timed Bayesian has a much higher run time on average than the Probability-Proximity search. The former's average time grows as graph size and time limit increase; the latter's average time relies only on the number of locations, as it calculates a weight for each, making its time essentially constant. Probability-Proximity's run time cannot be seen well on the figure due to being so much smaller than the Timed Bayesian search's. On average it completed a nine location search in 2.6E-4 seconds and a ten location search in 1.4E-4 seconds. Although the run times are an improvement upon the Exhaustive Bayesian search, as we intended, they are not comparable to the path calculation speed of the Probability-Proximity search in large environments.

There were some factors which we believe limited obtaining low distances and low run times for the Timed Bayesian search. The first of these limiting factors, we discovered, was that although the time limit successfully limited the Timed Bayesian search's run time, it also limited the accuracy of its best path predictions. Oftentimes for larger world, the Timed Bayesian search only had time to calculate one step, resulting in sub-optimal path choices that led it to travel farther distances to complete a hunt. Additionally, we found

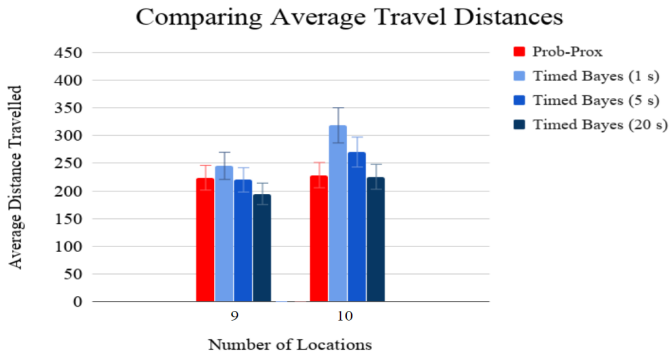

Fig. 2. Comparing average distances travelled by Probability-Proximity and Timed Bayesian agents at different time limits in larger environments.

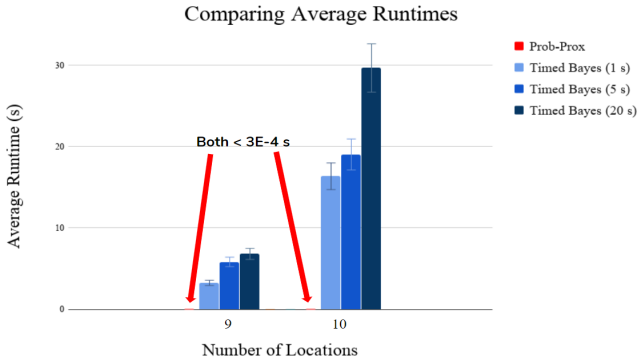

Fig. 3. Comparing average run times of Probability-Proximity and Timed Bayesian agents at different time limits in larger environments.

that the initialization of the saved paths data structure took an unprecedented amount of time. We do not factor in this initialization into the time elapsed under the limit, so this portion takes as much time as needed to iterate through every path and create an entry for it.

VI. Discussion

While our findings indicate that the Timed Bayesian improves upon the performance of the Exhaustive Bayesian in terms of completing hunts in large search spaces, as we were able to run tests on it without major time constraints for 9-10 nodes, it still cannot be considered a reasonable replacement for the Probability-Proximity search in these cases. The modifications we made on the original Bayesian agent did reduce its overall run time; however, the limitations discussed above caused no improvement in the average distance travelled, which is part of an aggregate goal (short calculation time + shortest distance possible) for developing efficient service robot searching techniques. Using the Timed Bayesian model would cause a user to wait upwards of an extra three to thirty seconds for the objects they requested to be found if this time could be greatly reduced by a model that performs similarly.

Improvements must be made to the Timed Bayesian search if it is to be considered a practical replacement for the Probability-Proximity model in large maps. An improved version might arise from a different approach to the algorithm, such as a ”smarter” Bayesian search which calculates the expected cost for a subset of paths rather than stepping down every possible path. For example, many path permutations begin with the same sequences of locations, so if we could detect these near-identical paths and calculate only one instance of them, our calculation time would be reduced. Another possible fix is the use of co-routines to perform context switching between paths, rather than using our path data storing method to remember each path's state. This could potentially eliminate the need to initialize the saved paths data structure, saving considerable time. Finally, using a lower-level language such as C or C++ to implement the algorithm might result in an efficiency increase. Python performs slowly for much of the functionality we require, such as copying hunts from one step to the next. With a lower-level, compiled language, we could attain a greater degree of control over program performance.

These results indicate that more work is needed to be done on an improved Bayesian algorithm before it can be considered a suitable replacement for the Probability-Proximity algorithm in larger environments. For now, having a slightly less comprehensive but a much faster method for object retrieval is better. Speed is the main concern for human convenience, which our algorithm did not ameliorate.

VII. Conclusion

In this paper, we discussed the scavenger hunt problem, extending the work done by the previous scavenger hunt research group that tested search algorithms for an agent in a space with unknown object locations. Our goal was to devise an algorithm that would reproduce the behavior of the Exhaustive Bayesian search while limiting the best path calculation time. We implemented a modified version of the Exhaustive Bayesian Search called the Timed Bayesian Search, which calculates expected path costs in a breadth-first, step-wise manner and whose execution is limited by a predetermined time limit. We then tested our algorithm to assess its functional equivalence to the Exhaustive Bayesian, then determined whether its performance in larger environments improved upon that of the Probability-Proximity Search and could be a better option.

Our results indicate that the Timed Bayesian does not perform significantly better than the Probability-Proximity search in large environments in both average run time needed to calculate the best path and average distance of the best path. Using the Probability-Proximity model is still the most viable solution for larger maps, as the original research group concluded. However, the Timed Bayesian model did improve upon the unreasonable run time of the Exhaustive Bayesian in large spaces. In future work, improvements upon our algorithm could be made with different implementation decisions and optimizations. We hope that our work might function as a stepping stone for more refined Bayesian-like search algorithms.

REFERENCES

[1] S. deBruyn, J. Suriadinata, and E. Tang, "A Scavenger Hunt for Service Robots," p. 1–6, Jun 2019.

[2] R. Casey, A. Chanler, M. Desai, B. Keyes, P. Thoren, M. Baker, and H. A. Yanco, "Good wheel hunting: Umass lowell's scavenger hunt robot system."