On Solving Oversubscription Planning Problems for Autonomous Underwater Vehicles

Conor McGann, Frederic Py, Kanna Rajan
Monterey Bay Aquarium Research Institute, Moss Landing, California
{cmcgann,fpy,kanna.rajan}@mbari.org

***Abstract*—This paper describes work to allow an Autonomous Underwater Vehicle (AUV) to navigate within a pre-defined planar roadmap to tackle a set of mission goals all of which may not be feasible for execution. The method uses results from an *all-pairs shortest path* calculation to estimate costs for traversal employing *steepest-ascent hill climbing* to identify a feasible subset of goals that have high utility and low cost. It employs *chronological backtracking heuristic search* in a *constraint-based temporal planning paradigm* to insert goals into the plan in the order of traversal to ensure compliance with other system constraints. The algorithm is integrated into an *intelligent executive*, which handles dispatch, monitoring, and repair of the plan as execution unfolds.**

I. Introduction

Imagine an Autonomous Underwater Vehicle (AUV Figure 1) loaded with science instruments, and capable of navigating effectively to great depths, and at low altitudes. Such a vehicle makes for an intrepid *Ocean Explorer*. Exploration often implies a vehicle-operator will not know a-priori what the AUV will find. For AUV's able to adapt missions in-situ based on observations, it becomes harder still to predict a-priori what goals are to be accomplished and ensure feasibility. In the event that not all goals are possible, simply quitting when the vehicle reaches a time and/or energy limit may yield very poor performance, since goals of higher priority may have been deferred till the end of the mission based on the incorrect assumption that there would be time and energy aplenty. A greedy approach that simply tackles the highest priority outstanding goal next can also yield very poor performance since the resulting path traversed may be much more expensive than an optimal path taking into account all goals together. So the core problem we address is one of selecting a subset of goals to tackle, and planning a path to tackle them.

[1] studied this kind of problem in the context of Mars Planetary Rovers. That work observed the strong similarity between *over-subscribed planning* and a variant of the traveling salesman problem called an *orienteering problem*. An orienteering problem is concerned with finding a path through a given set of control points to maximize a reward within a fixed cost bound. [1] noted that AI planning systems are generally not designed to deal with over-subscribed problems since their goal structure is conjunctive (i.e. all or nothing) rather than disjunctive (i.e. goals can be dropped if infeasible). They addressed this deficiency by first solving the abstracted orienteering problem, and then using the solution to this relaxed problem to feed goals to the planner in a prudent order, and guide the search process.

Figure 1: An MBARI AUV at sea

Our work exploits the key ideas underpinning this approach and applies them to a planning scenario on-board an AUV. A plan is a collection of actions used by an onboard executive to dispatch to a lower level functional layer, which in turn actuates subsystems on the AUV. Planning is done incrementally while interleaved with execution in the context of state updates generated by the functional layer (see [2] for details). We use *path-planning* in a distance graph to compute cost estimates for candidate solutions. We then use *local-search* to solve a relaxed form of the orienteering problem, exploiting these estimates. The solution to this relaxed problem provides a goal-ordering heuristic for refinement search in a *partial-order-planner*. The planner is embedded as a specialized deliberative component within an intelligent Teleo-Reactive Executive [2]. This ties the plan and planning to real world state throughout mission execution. Goals may be input at the start of a mission and/or *discovered* during mission execution based on observations of interest. Initial plans may be feasible but can break as execution unfolds, requiring a new plan to be generated.

The paper is laid out as follows. Section II describes the problem in more detail in the context of AUV science missions. Section III describes the design and

implementation of the solver. Section IV briefly outlines the integration of the planner in execution. Section V describes some preliminary results at sea. We conclude with a discussion of related work and some limitations of the current work to be addressed.

II. Problem Description

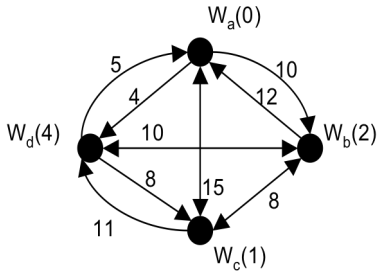

Figure 2: An Orienteering Graph for Benthic Exploration

Consider the scenario where an AUV is employed to seek out hydrothermal vents. The vehicle has been dispatched to map a region of the sea floor. The results indicate a number of locations that merit more detailed study. These locations can be assigned a priority based on some estimate of the probability of finding a vent. The vehicle must formulate a plan to visit these locations in a prudent order and make a set of observations to detect and then document the feature of interest. Moreover, as execution unfolds, the vehicle may spend a lot of time at a site where a vent is found, or move on quickly after determining no vent is present. Suppose we can estimate the costs of traveling from each location, to each of the other locations. Then the orienteering problem can be represented in a graph structure similar to that outlined in Figure 2. Each node (e.g. W_A, W_B, W_C, W_D) indicates a location of interest in the graph. The edges of the graph indicate the cost of traversal from one location to the other. The edges can be directed, which permits representation of different costs depending on the direction traveled. This is especially useful where strong currents play a role. In this example, all the goals occur at a location, though goals could be defined along the path between locations (e.g, "take up to 2 water samples between W_A and W_B"). The over-subscription planning problem is given by:

1. A set of *Initial Conditions (I)*. This minimally includes the current time, the current position, and the currently available energy. We can treat *I* as a *partial-plan*.
2. A *Goal Set (G)*. Each goal has a *priority*. Goals are ordered lexicographically based on priority, 0 being the highest priority. Each goal has a *start* and *end* location. This allows goals to be specified along traverses (*start* != *end*) or at locations (*start* == *end*). Finally, each goal has an estimate of *cost*. We assume cost is in terms of time and energy, and derived as a function of distance and speed of the vehicle.
3. A *Cost Map (M)*. This provides cost estimates for the edges in the orienteering graph.
4. A *Cost Bound (B)*. In our case the mission is constrained by finite battery capacity of the AUV and mission duration.

The solution to the over-subscription planning problem is a *feasible plan* for a subset of goals that *maximizes utility*.

III. Solution Technique

Figure 3 presents the algorithm to solve the over-subscription planning problem described above. Line 1 initializes the set O, representing the set of goals that have not yet been submitted for insertion into the partial plan s. It also initializes an empty tour for *t*. Line 2 computes an initial solution using *refinement search* [5]. No goals are submitted. The solution is simply a refinement of the initial conditions, which are presented as a partial plan. This preliminary solution will have no utility since it has no goals. If no consistent and complete solution exists for the initial conditions absent additional goals, then no solution exists and you return as in line 3. As we will see subsequently, the *Cost Map M*, and *Cost Bound B,* can be applied as constraints and heuristics in the ***Refine*** algorithm. Line 4 begins a process of iteration, incrementally extending the solution one goal at a time. First, a new goal, *g*, is selected via ***GoalSelect***. If there are no more goals, the current solution is returned. Goal selection operates as a goal-ordering heuristic to accomplish an efficient tour of locations. It also behaves as a filter, since goals may be excluded to arrive at a *feasible* solution. The new goal is removed *monotonically* from *O* (line 6). If goal *g* cannot be inserted into *s* it is excluded from further consideration. We then solve for the new goal, *g*, given the current partial plan *s*. Goals are *rejectable, that is they can be legally omitted from the plan*. If the goal is *rejected*, its score will not be included in the calculation of utility, and it will not impact any costs either. It is worth emphasizing that the algorithm *does not backtrack at this point to re-consider goals already inserted*. Rather, it will move on to another goal. We will now look at components of this algorithm in more detail.

Algorithm ***OSSolve(I, G, M, B)***

1. O = G; t = ∅; i = 0;
2. s = ***Refine***(I, {}, M, B);
3. if (s == ∅) return ∅; // No initial plan
4. g = ***GoalSelect***(t, i, G, O, M, B);
5. if (g == ∅) return s; // No more goals
6. O = O - g;
7. s = ***Refine***(s, {g}, M, B);
8. goto 4.

Figure 3: The Over-Subscription Solver

Algorithm ***GoalSelect(t, i, G, O, M, B)***

1. R = *rejected*(G);
2. if ($t \cap R \neq \emptyset$) t = *inserted*(*t*);
3. j = 0;
4. while($i < IterationMax \wedge j < PlateauLimit$){
   a. i++; j++;
   b. *t'* = *bestNeighbor*(t, G, O, M, B);
   c. if($t' \geq t$) t = t';
   d. if($t' > t$) j=0; // Off plateau so reset
5. }
6. return *next*(t);

Figure 4: Local Search for Goal Selection

A. Construction of a Cost Map

Before executing ***OSSolve*** we compute the cost map, *M*. The cost map provides *cost estimates* for edges in the orienteering graph; the construction of M is application dependent. In our application, we define a *distance graph* of nodes and edges to define the traversable links. Edge weights are the Euclidean distance between nodes of an edge. The cost map is then constructed for all pairs of *goals* using Dijkstra's algorithm [3] over the distance graph.

B. Goal Selection with Local Search

Figure 4 describes a variant of *steepest-ascent hill-climbing* [4] to solve the orienteering-problem. The parameter of particular interest is the tour, *t. where* $t \subseteq G$, This is represented as a sequence of goals. The algorithm begins by computing *R*, the set of rejected goals; this can be a simple lookup. *R* is used to test if a goal that was considered part of the tour was in fact *rejected* rather than *inserted*. That may occur where additional constraints in the full problem description necessitate removal of such goals for completion of a mission (e.g when adverse currents result in longer traversal times). If so, the tour is to include only the goals that have been *inserted* into the plan successfully. The central element of the local search is contained within the loop. In order to allow escape from a local maximum, a new solution that is no worse than the current solution will dominate. In order to avoid thrashing around a plateau in the solution space, a limit is placed on the number of promotions that can occur with no improvement in the score.

The relational operators used to compare tours directly exploit *M* and *B*. For each tour considered, the cost is estimated by summing the shortest path costs along the tour, starting from the beginning. The cost map can provide *O*(1) access times for pair-wise costs if stored in an adjacency matrix, making cost evaluation linear in the length of the tour. Utility is computed according to the sum over the goals in the tour: $\sum_1^n K^{P-p(i)}$ where *K* and *P* are integer constants and *p(i)* is the priority of the i^{th} goal in the tour. The selection of *K* and *P* are specified to ensure a lexicographic ordering of goals (e.g. a goal with priority of 1 dominates any combination of goals with priority > 1). In our tests, *K* is 10 and *P* is 5. Our evaluation policy reflects a preference for feasible plans first, and maximizing utility second. *Feasibility* is evaluated by comparing the tour cost to *B*. Feasible candidates dominate infeasible candidates. Higher utility dominates when both candidates are feasible. Lower cost dominates when both candidates are infeasible. Otherwise a simple comparison of *utility/cost* is used.

The *bestNeighbor* function returns a candidate solution in the neighborhood of the current best tour with the best score – hence a steepest-ascent hill-climb. A neighbor is a tour that is obtained by applying *exactly one* of the following operators to the current tour:

1. *insert* – insert one of the goals in *O*, in one of the positions in t. Removes a goal from *O* if promoted.
2. *swap* – swap 2 goals in t.
3. *remove* – remove one of the goals in t. Inserts a goal in *O* if promoted.

The scoring of a candidate can use the same comparator used in ***GoalSelect***. There are $O(N^2)$ neighbors to consider in the worst case, where N is the number of goals. Not all operators make sense all the time. . For example, there is no insertion possible if all goals are already present in the tour, or if the current tour is infeasible

C. Refinement Search

Recall that the orienteering problem is only an abstraction of the underlying over-subscription problem. In practice,

there can be substantial additional detail in the plan that must be filled in. For example, a detailed path may be traversed to move from one location of scientific interest to another. Furthermore, a range of science observations may be required to accomplish the science goal. All these refinements can impact available time and energy on the AUV. For this detailed planning we use *refinement search*.

Refinement search is the process of solving a planning problem by searching the space of partial plans [5]. The final plan is a monotonic refinement of the initial partial plan. Our work uses $EUROPA_2$ [6] to implement this algorithm an open-source library for constraint-based temporal planning. We exploit the Cost Map in two ways within the $EUROPA_2$ framework:

1. Detailed path planning can exploit the cost map to guide path selection decisions. It can prune infeasible path choices and select choices that minimize the distance to the end goal.
2. Cost information can be used, in conjunction with other elements of the model (e.g. speed) to propagate temporal bounds on actions in the plan, allowing further pruning of the search space.

IV. Integration With Execution

We have two key reasons for placing an *Over-Subscription Planner* on-board. First, there is substantial uncertainty in the actual costs of activities in execution. To avoid being too conservative, we want to reserve the right to

keep goals in the plan that might be feasible until we determine otherwise. If such an event occurs, we must re-plan. Second, we anticipate discovering new goals in-situ, potentially at depths where surfacing the vehicle for guidance is not an option. Consequently, making the planner available on-board is a necessity. To accomplish this, we use T-REX [2], an *Executive* designed to synthesize deliberation and execution in a seamless manner. In this case, the planner is the deliberation algorithm for mission management. T-REX takes care of *inserting observations* in the plan as execution unfolds, *dispatching* the plan for execution, *propagating updates* in the plan to *project plan failures* and *invoking the planner* to re-plan or to insert newly discovered goals as the mission proceeds.

V. Results

The Over-Subscription Planner was integrated with T-REX and tested in simulation as well as at sea in the course of a series of sea trials in the Monterey Bay, California. Figure 5 gives an example of the scenarios used to test the algorithm. Nodes define positions of interest in latitude and longitude. Links are directed edges between node pairs. The horizontal and vertical scales of the test area are

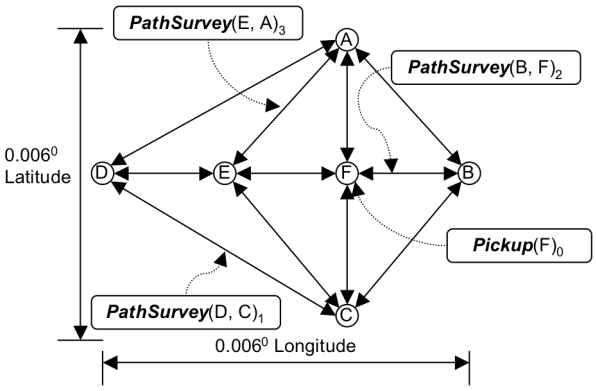

Figure 5: Multi-Objective Mission for Sea-Trials

approximately 0.006 decimal degrees longitude and latitude respectively. 4 goals are shown. The *science goals* were to conduct a *path survey* from a start node to an end node. A path survey enables the AUV to travel the shortest path between these nodes within a depth range while running a range of science instruments. The mission also includes an operator goal to have the vehicle ready for *pickup* at the designated node. The subscript attached to each goal indicates its priority. Throughout mission execution, the vehicle is subject to an operations constraint on the length of time the vehicle may remain submerged after which it must pre-empt its work and surface to localize using GPS. The duration of this task depends on the conditions at the surface. With high sea state, the wave action can inhibit the GPS. The vehicle will wait at the surface until it obtains the desired number of GPS hits. The mission is subject to a maximum duration constraint.

We have 2 key mechanisms to adjust the problem. By reducing the mission duration we can make the problem over-subscribed. By forcing the vehicle to linger when localizing we can simulate a case where unforeseen execution delays cause the plan to break. Using these methods, we have successfully demonstrated a number of cases that were over and under subscribed during execution. For example, given the goal set in Figure 5, we constrained the mission to terminate in 1000 seconds, and localize every 240 seconds with a delay at the surface of 150 seconds. The initial position was close to *F*. The initial plan correctly dropped the survey goal along *EA*, but included all other goals. The planned path was *DCBF*. The delay accrued at the surface resulted in a projected failure during the traverse *DC*. Dropping the goal to survey along BF repaired the plan. The final path was *DCF*.

VI. Related Work

The combination of an orienteering solver and a refinement search planner was first described in [1]. Our approach differs by using local search to solve the orienteering problem coupled with a cost estimation method based on a distance graph. In contrast, [1] uses a plan-graph based technique for cost estimation and beam-search to solve the orienteering problem. The distance graph is very appealing in our domain because of the strong dominance of navigation in time and energy consumption.

Our work is further discriminated as it is used on-board an AUV rather than off-board for Mars Rovers. To our knowledge, no over-subscription planning technique has been applied in underwater robotics.

VII. Conclusions

Early results indicate that the combination of an orienteering solver and a refinement search planner can be effective in enabling an AUV to tractably handle over-subscribed planning problems on-line. Moreover, cost estimation using graph-based path planning techniques can be applied to solve both the abstracted orienteering problem and the subsequent plan refinement algorithm for goal insertion. The algorithms proposed in this paper have not yet been subjected to rigorous empirical study to understand in more detail how well they scale in theory or in practice, or to compare their performance against other comparable techniques. Finally, we are seeking to apply these methods to missions of scientific interest in the context of unstructured ocean exploration.

VIII. Acknowledgements

This research was supported by the David and Lucile Packard Foundation. We thank NASA Ames Research Center for making the EUROPA planner available. We thank Dave Smith also of NASA Ames, for discussions and suggestions that helped this work evolve. We also wish to thank our colleagues at MBARI - Rob McEwen, Rich Henthorn, Hans Thomas, Duane Thompson, Doug Conlin

and the captain and crew of the RV Zephyr - for their help in integration and deployment.

IX. REFERENCES

[1] David E. Smith. “Choosing Objectives in Over-Subscription Planning,” Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004), June 3-7 2004, Whistler, British Columbia, Canada.

[2] Conor McGann, Frederic Py, Kanna Rajan, Richard Henthorn, Rob McEwen. “A Deliberative Architecture for AUV Control”. ICRA 2008. Pasadena, California, USA.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. “Introduction to Algorithms”. 2nd edition, McGraw-Hill Book Company, Boston, MA, 2001, pXX-YY.

[4] Zbigniew Michalewicz, David B. Fogel. “How to Solve It: Modern Heuristics”, Springer-Verlag, 2000, p43-45.

[5] Subbarao Kambhampati, Craig A. Knoblock, Qiang Yang. "Planning as Refinement Search: A Unified Framework for Evaluating Design Tradeoffs in Partial-Order Planning", *Journal of Artificial Intelligence*, volume 76, number 1-2, pages 167-238, 1995.

[6] Frank, J., A. Jonsson, “Constraint-based attributes and interval planning,” Journal of Constraints, vol. 8, Oct. 2003.