A Lexicographic Search Method for Multi-Objective Motion Planning

Tixiao Shan and Brendan Englot

Abstract—We consider multi-objective motion planning problems in which two competing resources are penalized hierarchically. The highest-priority resource takes on non-negative values over robot paths, and is frequently zero-valued. This is intended to capture problems in which robots must manage a resource such as collision risk, exposure to threats, or access to valuable measurements, whose consideration is critical in some regions of the environment, and unimportant in others. This leaves freedom for tie-breaking with respect to a secondary resource, which we assume to be a strictly positive quantity consumed by the robot, such as distance traveled, energy expended or time elapsed. We leverage the paradigm of lexicographic optimization and apply it for the first time to graph search over roadmaps. Specifically, our “lexicographic search” is employed in concert with probabilistic roadmaps to solve motion planning problems under various non-negative cost functions motivated by real-world applications.

I. INTRODUCTION

Multi-objective motion planning has been an area of interest in robotics for many years. Continuous multi-objective motion planning in two and three dimensions has been achieved by gradient descent, paired with sampling the Pareto front to identify feasible solutions under added constraints [23]. Genetic algorithms [3, 30] and dynamic programming [11, 13, 22] have also been applied to solve multi-objective motion planning problems. Early work on multi-objective planning over configuration space roadmaps [18] has been succeeded by methods that recover Pareto fronts from probabilistic roadmaps (PRMs) [4, 8]. In pursuit of solutions that can be produced quickly, preferably in real-time, and applied to problems of high dimension, sampling-based motion planning algorithms such as the PRM [16], the rapidly-exploring random tree (RRT) [19], and their optimal variants PRM* and RRT* [15] have been adapted to solve a variety of multi-objective motion planning problems. Such approaches have typically considered the tradeoff between a resource such as time, energy, or distance traveled and a robot’s information gathered [12]. localization uncertainty [1, 20], collision probability [2], clearance from obstacles [17], adherence to a set of rules [26], and exposure to threats [4] and other generalized representations of risk [6, 14].

In this paper, we will consider problems in which two competing resources are penalized hierarchically. The highest-priority resource takes on non-negative values over robot paths, and is frequently zero-valued. This is intended to capture problems in which robots must manage a resource such as collision risk, exposure to threats, or access to valuable measurements, whose consideration is critical in some regions of the environment, and unimportant in others. For example, although some algorithms seek to manage collision risk by adhering closely to the medial axis of the free space [5, 33], this is often overly conservative. Penalizing such risks only in the pertinent regions of the environment, where collision is possible, leaves freedom for tie-breaking with respect to a secondary resource, which we assume to be a strictly positive quantity consumed by the robot, such as distance traveled, energy expended or time elapsed. Such a problem fits nicely into the framework of lexicographic optimization.

The lexicographic method [27] is the technique of solving a multi-objective optimization problem by arranging each of several cost functions in a hierarchy reflecting their relative importance. The objectives are minimized in sequence, and the evolving solution may improve with respect to every subsequent cost function if it does not worsen in value with respect to any of the former cost functions. Use of this methodology has been prevalent in the civil engineering domain, in which numerous regulatory and economic criteria often compete with the other objectives of an engineering design problem. Variants of the lexicographic method have been used in watershed land use planning [21], for vehicle detection in transportation problems [28], and in the solution of complex multi-objective problems, two criteria at a time [9].

Among the benefits of such an approach is the potential for the fast, immediate return of a feasible solution that offers globally optimal management of the primary resource, in addition to locally optimal management of the secondary resource in areas where the primary resource is zero-valued. Due to the fact that the spatial regions in which the primary resource is penalized can be adjusted by the user, such an approach offers an intuitive means for tuning the relative influence of primary and secondary cost functions, in contrast to tuning additive weights on the competing cost functions [26]. Such an approach also stands in contrast to robot motion planning methods that manage the relative influence of competing cost criteria using constraints [1, 2, 6, 12, 20]. Avoiding any potential struggles to recover feasible solutions under such constraints, the lexicographic motion planning problem is unconstrained with respect to the resources of interest.

The principal contribution of this paper is a search method that we employ in concert with PRMs to solve complex multi-objective motion planning problems quickly. Our lexicographic search is compared with a competing method that searches an expanded graph layer by layer, where the edge costs of the graph represent one resource, and the layers of
the expanded graph represent the other. Section II defines the problem of interest and Section III describes and justifies the proposed solution. Section IV presents a computational study comparing the proposed method with the competing expanded graph search. The algorithm’s flexibility is demonstrated over a variety of robot motion planning problems with non-negative primary cost functions inspired by real-world autonomous navigation challenges.

II. PROBLEM DEFINITION

Let $C$ be a robot’s configuration space, which, in the examples to follow, will represent the spaces $\mathbb{R}^d$ and $SE(2)$. $x \in C$ represents the robot’s position and volume occupied in $C$. $C_{\text{obst}} \subseteq C$ denotes the set of obstacles in $C$ that will cause collision with the robot. $C_{\text{free}} = c(C \setminus C_{\text{obst}})$, in which $c(l)$ represents the closure of an open set, denotes the space that is free of collision in $C$. We will assume that given an initial configuration $x_{\text{init}} \in C_{\text{free}}$, the robot must reach a goal state $x_{\text{goal}} \in C_{\text{free}}$. Let a path be a continuous function $\sigma : [0,1] \rightarrow C$ of finite length. Let $\Sigma$ be the set of all paths $\sigma$ in a given configuration space. A path is collision-free if $\sigma \in C_{\text{free}}$ for all arguments, and a collision-free path is feasible if $\sigma(0) = x_{\text{init}}$ and $\sigma(1) = x_{\text{goal}}$. The problem of finding a feasible path may be specified using the tuple $(C_{\text{free}}, x_{\text{init}}, x_{\text{goal}})$. A cost function $c : \Sigma \rightarrow \mathbb{R}_{0}^+$ returns a non-negative cost for all collision-free paths.

Due to our selection of the PRM algorithm for solving the planning problems of interest, we will assume the robot moves through $C_{\text{free}}$ along paths obtained from a directed graph $G(V,E)$, with vertex set $V$ and edge set $E$. An edge $e_{ij} \in E$ is a path $\sigma_{ij}$ for which $\sigma_{ij}(0) = x_i \in V$ and $\sigma_{ij}(1) = x_j \in V$ to which no other members of $V$ belong. $G$ is constructed and stored with no more than one such path for every permutation of two vertices. Two edges $e_{ij}$ and $e_{jk}$ are said to be linked if both $e_{ij}$ and $e_{jk}$ exist. A path $\sigma_{pq} \in G$ is a collection of linked edges such that $\sigma_{pq} = \{e_{p_{1}q_{1}}, e_{p_{2}q_{2}}, \ldots, e_{p_{n-1}q_{n}}, e_{p_{n}q_{n}}\}$.

Our proposed algorithm uses the following primary cost function, which penalizes a generalized representation of the risk accumulated along a path:

$$c_{\text{Risk}}(\sigma) := \int_{\sigma(0)}^{\sigma(1)} Risk(\sigma(s))ds$$

(1)

$$Risk(x) := \begin{cases} R(x), & \text{if } R(x) > \text{RiskThreshold} \\ 0 & \text{otherwise} \end{cases}$$

(2)

where the function $Risk : C_{\text{free}} \rightarrow \mathbb{R}_0^+$ evaluates the risk at an individual robot configuration. We penalize a robot’s risk using the tunable risk threshold $RiskThreshold \in \mathbb{R}^+$. Changing this threshold represents an alteration of the physical boundaries of the workspace in which the primary cost takes on positive values. $R : C_{\text{free}} \rightarrow \mathbb{R}^+$ represents the strictly positive underlying risk at a robot configuration, which is evaluated against $RiskThreshold$ and returned if the threshold is exceeded. We will alternately express $c_{\text{Risk}}(\sigma)$ using the notation $P1(\sigma(0)), \sigma(1))$ to indicate the specific configurations at the limits of the primary-cost integral, and $P1(e_{ij})$ when the path $\sigma$ is comprised of a single edge $e_{ij} \in E$.

In addition, we define a secondary cost $c_{\text{second}}$ of a path $\sigma$ as follows:

$$c_{\text{second}}(\sigma) := \int_{\sigma(0)}^{\sigma(1)} Second(\sigma(s))ds$$

(3)

where the function $Second : C_{\text{free}} \rightarrow \mathbb{R}_0^+$ represents a strictly positive cost, ensuring that ties in secondary cost do not occur as they do for primary cost. We will alternately express $c_{\text{second}}(\sigma)$ using the notation $SI(\sigma(0)), \sigma(1))$ to indicate the limits of the secondary-cost integral, and $SI(e_{ij})$ when the path $\sigma$ is comprised of a single edge $e_{ij} \in E$.

We will apply these cost functions to the problem of lexicographic optimization, which is formulated as follows [21]:

$$\min_{\sigma \in \Sigma} f_i(\sigma_i)$$

$$\text{subject to } : f_j(\sigma_j) \leq f_j(\sigma_j^*) \quad j = 1, 2, \ldots i - 1; \quad i = 1, 2, \ldots, k.$$  

(4)

The formulation is adapted here to show cost functions that take paths as input. In the current phase of the procedure depicted in (4), a new solution $\sigma_i^*$ will be returned if it does not increase in cost with respect to any of the prior cost functions $j < i$ previously examined. Necessary conditions for optimal solutions of (4) were first established by Rentmeesters [25]. Relaxed versions of this formulation have also been proposed, in which $f_j(\sigma_j) \leq f_j(\sigma_j^*)$ is permitted, provided that $f_i(\sigma_i)$ is no more than a small percentage larger in value than $f_j(\sigma_j^*)$. This approach, termed the hierarchical method [24], has also been applied to multi-criteria problems in optimal control [32].

The scope of this paper is limited to the consideration of bicriteria problems, for which $k = 2$. The procedure outlined in (4) will be applied to probabilistic roadmaps in which $f_1(\sigma) = c_{\text{Risk}}(\sigma)$ and $f_2(\sigma) = c_{\text{second}}(\sigma)$. The fact that, per the user-defined RiskThreshold parameter, there will be regions of the workspace in which the primary cost is zero-valued, ensures that improvements can be made with respect to the secondary cost function without increasing the value of the primary cost function. We also note that the use of (1)-(3) limits our consideration to costs that are additive among the edges of a graph. In addition, the functions that evaluate the cost contributions of individual states along a path, $Risk(x)$ and $Second(x)$, are functions of a single robot configuration only, and are not history-dependent. In the following section we will describe how Dijkstra’s algorithm [27] may be applied to solve (4) in a bicriteria search of a probabilistic roadmap.

III. ALGORITHM DESCRIPTION

Our proposed adaptation of Dijkstra’s algorithm to perform a “lexicographic search” is given in Algorithm 1. Provided with a roadmap $G(V,E)$, a user-defined RiskThreshold, and start state $x_{\text{init}}$ as inputs, the algorithm initializes two sets of labels for each node, one describing the best-so-far primary cost-to-come, and the other describing the best-so-far secondary cost-to-come. In real-time applications of the search, $x_{\text{init}}$ is designated to be the closest configuration in the
Algorithm 1 Lexicographic Dijkstra Search with Two Cost Criteria

1: Inputs: $G(V,E), \text{RiskThreshold}, x_{\text{init}}$
2: SetLabels$_1(V, \infty)$; SetLabels$_2(V, \infty)$;
3: $x_{\text{init}}\cdot\text{label}_1 \leftarrow 0$; $x_{\text{init}}\cdot\text{label}_2 \leftarrow 0$; $x_{\text{init}}\cdot\text{parent} \leftarrow \{\}$;
4: $X_{\text{queue}} \leftarrow \{V\}$; $X_{\text{visited}} \leftarrow \{\}$;
5: while $|X_{\text{queue}}| > 0$ do
6: $X_{\text{min}} \leftarrow \text{FindMinLabel}_1(X_{\text{queue}})$;
7: if $|X_{\text{min}}| > 1$ then
8: $x_i \leftarrow \text{FindMinLabel}_2(X_{\text{min}})$;
9: else
10: $x_i \leftarrow X_{\text{min}}$;
11: end if
12: $X_{\text{visited}} \leftarrow X_{\text{visited}} \cup x_i$; $X_{\text{queue}} \leftarrow X_{\text{queue}} \setminus x_i$;
13: for ($x_j | e_{ij} \in E$) do
14: if $x_j\cdot\text{label}_1 > x_i\cdot\text{label}_1 + PI (e_{ij})$ then
15: $x_j\cdot\text{label}_1 \leftarrow x_i\cdot\text{label}_1 + PI (e_{ij})$;
16: $x_j\cdot\text{label}_2 \leftarrow x_i\cdot\text{label}_2 + SI (e_{ij})$;
17: $x_j\cdot\text{parent} \leftarrow x_i$;
18: else if $x_j\cdot\text{label}_1 = x_i\cdot\text{label}_1 + PI (e_{ij})$ then
19: if $x_j\cdot\text{label}_2 > x_i\cdot\text{label}_2 + SI (e_{ij})$ then
20: $x_j\cdot\text{label}_1 \leftarrow x_i\cdot\text{label}_1 + PI (e_{ij})$;
21: $x_j\cdot\text{label}_2 \leftarrow x_i\cdot\text{label}_2 + SI (e_{ij})$;
22: $x_j\cdot\text{parent} \leftarrow x_i$;
23: end if
24: end if
25: end for
26: end while

roadmap to the robot’s current configuration. A queue $X_{\text{queue}}$ is populated with the nodes of the roadmap.

In each iteration of the outermost while loop, the $\text{FindMinLabel}_1()$ operation returns the set of configurations that share the minimal primary cost-to-come among the nodes in $X_{\text{queue}}$ (Line 6). If $X_{\text{min}}$ contains more than one configuration, secondary costs are used to find the single vertex in $X_{\text{min}}$ offering the minimum secondary cost-to-come, whose neighbors will be examined in detail. The selected node is designated $x_i$ (Lines 8-10). Node $x_i$ is then used to reduce the primary and secondary costs of neighboring nodes $x_j$, if edge $e_{ij}$ exists. If the primary cost from $x_{\text{init}}$ to $x_j$ via $x_i$ is lower than the current cost, $x_j\cdot\text{label}_1$, the primary and secondary costs of $x_j$ are updated by choosing $x_i$ as its new parent (Lines 15 - 17). If there is a tie in primary cost, the secondary cost function will be used once again to break the tie (Line 19 - 22).

Just as the problem formulation in (4) only allows improvements to a solution’s secondary cost when it does not adversely impact the primary cost, the proposed search method only allows improvements to be made in secondary cost when ties occur with respect to primary cost. Our single-source shortest paths solution would take on the same primary cost whether or not these improvements are performed, but the occurrence of ties allows us to opportunistically address an auxiliary cost function in the style of lexicographic optimization. The procedure also takes on the same computational complexity of a standard Dijkstra search, with the number of required operations increasing only by a constant factor. We note that a naive $O(|V|^2)$ implementation of the algorithm is given here for clarity, though its runtime can be improved with the use of data structures such as Fibonacci heaps [10].

![RiskThreshold](image)

**Fig. 1:** An example of the application of the proposed lexicographic search. An obstacle is colored red, and the blue vertex represents the robot’s initial configuration. The green vertex represents the goal configuration, while vertices 2 and 3 are intermediate configurations. Penalized and penalty-free regions are divided by a user-defined RiskThreshold.

The example given in Figure 1 is designed to illustrate the mechanics of the proposed search method. The objective is to find the path of minimum cumulative exposure to risk during a robot’s travel from vertex 1 to vertex 4. Since part of the edge $e_{14}$ is in a penalty region where our chosen representation of risk, the inverse distance transform, exceeds the designated RiskThreshold, the risk cost $PI(e_{14})$ is non-zero. On the other hand, edges $e_{12}, e_{13}, e_{23}, e_{24}$ and $e_{14}$ accumulate zero risk cost, as they all avoid the penalty region. As such, none of the resulting solutions should enter the penalty region, and they should achieve the minimum possible distance traveled subject to the avoidance of the risk penalty.

During the first iteration of the lexicographic Dijkstra search, $x_1$ is popped from the queue. The parent of $x_2, x_3$ and $x_4$ is set to $x_1$ after this iteration. Then, in the next iteration, both $x_2$ and $x_3$ are returned in $X_{\text{min}}$, as they share an identical zero-valued primary cost. However, $x_3$ has a smaller secondary cost and is moved from $X_{\text{queue}}$ to $X_{\text{visited}}$ as a result. A tie exists in the third iteration when $x_2\cdot\text{label}_1$ and $x_3\cdot\text{label}_1 + PI(e_{3,2})$ are compared, as they both evaluate to zero, requiring a consideration of secondary cost to break the tie. In the fourth and final iteration, $x_4$’s parent is changed from $x_1$ to $x_3$ because $x_3$ can offer the shortest path of zero primary cost from $x_1$.

IV. COMPUTATIONAL RESULTS

To permit a fair evaluation of the proposed algorithm’s computational performance, we must select a suitable basis for comparison. Our first efforts at comparison have involved testing different combinations of multiplicative coefficients on the primary and secondary cost functions, which are summed together into a single composite cost function and
subjected to a standard Dijkstra search over a probabilistic roadmap. However, this approach has been observed to be highly sensitive to the choices of coefficients, where very small differences can have large impact on the quality of the solutions obtained. A systematic procedure for solving a lexicographic optimization using such a method would involve choosing many combinations of coefficients, searching the roadmap with each, and choosing, among the paths offering minimum primary cost, the solution of minimum secondary cost. However, a straightforward linear scaling of coefficients has not been found in practice to thoroughly probe the space of solutions, without being exhaustive.

Instead we turn to a multi-objective search method proposed for graphs over which costs are non-negative, but not strictly positive [29]. This approach entails the search of an expanded graph, which applies one of the competing resources to its edge weights, and the second resource is represented by expanding the original graph across many separate layers, representing different amounts of resource consumption. We will hereafter refer to this competing method as the expanded graph search (EGS), which requires a layer-by-layer Dijkstra search to uncover solutions that trade off consumption of the competing resources. We have encountered the best results by placing the lexicographic problem’s secondary cost, which is strictly positive (typically a robot’s distance traveled), along the edges of the roadmap. The lexicographic problem’s primary cost, which is occasionally zero-valued, accumulates from layer to layer of the expanded graph. At every layer of the graph, the Dijkstra search is constrained by its “budget total,” which cannot be exceeded by paths recovered on this layer. By systematically increasing the allotted resource consumption from layer to layer, and by employing finely-spaced layers, solutions close in value to those produced by the lexicographic search may be recovered. In the limit of many expanded graph layers, all of the solutions recovered by a lexicographic search can be produced by searching the layers of the expanded graph. The relative computational costs of the two methods are compared below, in which linear scaling among layers proves highly effective for the EGS method.

|                  | Total Runtime | Averaged Runtime for Each Path |
|------------------|---------------|-------------------------------|
| Expanded Graph   |               |                               |
| 5 Layers         | 10.771s       | 0.513s                        |
| 20 Layers        | 37.825s       | 1.801s                        |
| 50 Layers        | 92.395s       | 4.399s                        |
| Lexicographic Search | 2.887s      | 0.137s                        |

A. Simulation Results in \( \mathbb{R}^2 \)

A comparison of two methods, EGS and lexicographic search, is employed to verify the effectiveness of motion planning with the proposed method. For the purposes of illustration, our first example considers geometric planning in \( \mathbb{R}^2 \) for a translating point robot. The red and blue points in Figure 2 at (1,1) and (14,1) indicate the robot’s initial configuration \( x_{\text{init}} \) and goal configuration \( x_{\text{goal}} \) respectively. There are two obstacles in the workspace that are colored black, and the workspace is surrounded by walls. The contour lines numbered from 0.5 to 3 represent the distance from each line to the closest obstacle in meters. The primary, risk-based cost is the inverse distance transform, for which many different threshold values are explored. Specifically, we perform searches for 21 different \( \text{RiskThreshold} \) values for obstacle standoff distances spanning from 0 to 2 m with an increment of 0.1 m, using a PRM graph that includes 3,000 samples. The maximum risk budget for EGS is set to 10. The risk budget is discretized evenly from 0 to 10 units divided among 5, 20 and 50 “layers” in each separate trial.

The paths recovered from the use of different \( \text{RiskThreshold} \) values are depicted in different colors. When \( \text{RiskThreshold} \) is equal to 0 and 0.1, the shortest safe path that connects \( x_{\text{init}} \) and \( x_{\text{goal}} \) passes the narrow passage between the left obstacle and the wall. When \( \text{RiskThreshold} > 0.1 \), zero-risk paths cannot be found.
Fig. 3: The solutions generated by two competing methods are plotted in different colors. The objective is to find the minimal risk path from $x_{\text{init}}$ (red point) to $x_{\text{goal}}$ (black point) with respect to a set of different $\text{RiskThreshold}$ values.

anymore due to the fact that the start and goal locations are in the regions where risk is penalized. The narrow passage is also no longer safe, as its width is no more than 0.4 units. As a result, the robot travels through the wider passage between the two obstacles. As the value of $\text{RiskThreshold}$ is increased incrementally, the passage between the two obstacles is not safe again when $\text{RiskThreshold} \geq 1.2$. Longer paths are preferred due to the goal of minimizing the robot’s cumulative exposure to risk.

As we can see in Figure 3(d), (e) and (f), the paths recovered from the lexicographic method represent the minimum-risk paths with respect to the specific spatial boundaries, $\text{RiskThreshold}$, within which risk is penalized. Some paths obtained from the EGS method are superior in distance cost. As the expanded graph’s layers are discretized more finely, the paths produced by EGS approximate the paths from the lexicographic search method.

Using C++ implementations of the two algorithms run on a laptop with an Intel i7 2.5GHz processor, equipped with 16GB RAM, algorithm runtime results are shown in Table 1. All quantities described here are mean values averaged over 100 trials of each method. Apparently, lexicographic search offers a substantial computational advantage over EGS, as only 2.887s is needed to recover 21 optimal paths. Since discretization of the budget is required by EGS, the roadmap is searched on the resulting layers that represent different levels of budget. If the budget is discretized finely enough, EGS can recover all optimal paths for a given start and goal location. However, if the budget is not discretized finely enough, we are left to guess where finer discretization is needed, and we must reason in unintuitive units - the cost accumulated over the course of path, which may represent risk, exposure to threats, or similar. In addition, the maximum budget for EGS also needs to be guessed. On the one hand, if the maximum budget is too big, it needs to be discretized more finely which will cause unnecessary computational burden. On the other hand, some paths will never be found if the maximum budget is given a small value.

B. Planning over a Terrain Map

Further simulations are performed over a terrain map shown in Figure 4. $\text{RiskThreshold}$ is now defined as the maximum safe altitude for the robot, per a modified $\text{Risk}(x)$ function defined in Equation (5). $\mathcal{H}(x)$ returns the altitude of config-
Fig. 4: An example shows the paths produced by the lexicographic method in a terrain map with 11 different RiskThreshold values. The color of the terrain changes from blue to yellow as the altitude increases. Risk is penalized when the altitude of robot exceeds RiskThreshold.

When H(x) exceeds RiskThreshold, the robot has greater likelihood of being exposed to hostile threats due to increased visibility. Figure 4(a) shows the single-source shortest paths to all configurations in the roadmap after one lexicographic search when RiskThreshold = 10m. In Figure 4(b), the robot starts from (0,0,3.6) and its destination is at (150,100,30). Eleven RiskThreshold values, which increase from 10m to 35m with an increment of 2.5m, are used in this example. When RiskThreshold is small, the path stays in the valley before climbing the hill to reach the goal location. When RiskThreshold is increased, the robot has more freedom to climb the hill with an altitude less than RiskThreshold to achieve a shorter path. Averaged over 100 trials, each lexicographic search spends 0.329s per path recovered from a graph with 5,000 samples.

\[
Risk(x) := \begin{cases} 
C(x) & H(x) > RiskThreshold \\
0 & H(x) \leq RiskThreshold 
\end{cases}
\]  

\[ (6) \]

D. Real-world Results with Differential Constraints

We have also performed a test on a mobile robot, the Turtlebot 2, with motion planning subject to Dubins constraints. Our aim is to verify the viability of the lexicographic search method for a real-world mobile robot, in which the robot is mapping an a priori unknown environment and iteratively replanning. The robot’s goal was to make progress toward a designated waypoint while maintaining a safe standoff from obstacles, and curbing the length of its route when not in danger of collision. A Hokuyo UTM-30LX laser scanner, which has a 30 meter range and 270° field of view, was mounted on the top of the robot. Equations (1)-(2) in Section II are used here once again to compute primary cost, as the robot’s risk-based primary objective is once again represented by the inverse distance transform. The robot’s RiskThreshold is set to 1, penalizing travel within one meter of the nearest obstacle.

As is shown in Figure 6(b), the goal state of the path is close to an obstacle, and also to areas of the map whose contents are initially unknown. This is because the workspace is not fully explored yet and the reach of the PRM is limited. However, as the robot makes forward progress and the map is expanded, this part of the path remains close to the medial axis between the surrounding obstacles and the boundaries of the environment, as this mitigates the level of risk cost accumulated. The second path depicted achieves an improved clearance from obstacles, as the graph is now expanded and the robot has a better understanding of the scenario, and the third path depicted achieves the best clearance yet. In this real-world test, 500 nodes are included in each roadmap constructed. The time required to generate the roadmap and search for a path was found to be less than one second on average per iteration.

V. CONCLUSIONS AND FUTURE WORK

We have proposed a lexicographic search method intended for use in multi-objective robot motion planning problems, in which two competing resources are penalized. The highest-priority resource takes on non-negative values over robot paths, and is frequently zero-valued. In such instances, we have the freedom to additionally optimize a secondary resource
Fig. 5: In this underwater scenario, there are eight pilings standing in the vicinity of a seawall. A USBL (green point in (b) and (c)) positioning system is assumed to be used here for acoustic positioning. The color of objects in the map changes from red to purple as their depth increases. Start and goal positions are colored red and blue respectively. (b) An aerial view of the scenario. (c) A side view of the scenario, with obstacles omitted.

Fig. 6: A Hokuyo laser scanner is mounted on the top of a Turtlebot 2 indoor ground robot. The obstacles are colored black and the unknown regions of the environment are colored gray. The robot is permitted to travel in the light-colored region. The path planned and executed by the robot at each step in time depicted is colored red, and the robot’s probabilistic roadmap is colored green.

to the extent that it does not diminish the primary resource. Over such problems, we have demonstrated that the proposed search method is capable of producing high-quality solutions in superior runtime to the search of an expanded graph that presents an alternative approach to solving lexicographic motion planning problems. The lexicographic search has also been shown to offer flexible applicability to a variety of relevant cost functions, in which tuning the algorithm’s penalty threshold effectively problems the tradeoffs among competing cost criteria. Future work entails the extension of this method beyond bicriteria problems to more complex and extensive cost hierarchies, and to challenging objectives that are history dependent, such as localization uncertainty.

REFERENCES

[1] S.D. Bopardikar, B. Englot, and A. Speranzon, “Multiobjective Path Planning: Localization Constraints and Collision Probability,” IEEE Transactions on Robotics, 31(3):562-577, 2015.
[2] A. Bry and N. Roy, “Rapidly-Exploring Random Belief Trees for Motion Planning Under Uncertainty,” Proceedings of the IEEE International Conference on Robotics and Automation, 723-730, 2011.
[3] O. Castillo, L. Trujillo, and P. Melin, “Multiple Objective Genetic Algorithms for Path-Planning Optimization in Autonomous Mobile Robots,” Soft Computing, 11(3):269-279, 2000.
[4] Z. Clawson, X. Ding, B. Englot, T.A. Frewen, W.M.
