Semi-Boustrophedon Coverage with a Dubins Vehicle

Jeremy S. Lewis, William Edwards, Kelly Benson, Ioannis Rekleitis, and Jason M. O'Kane

Abstract— **This paper addresses the problem of generating coverage paths—that is, paths that pass within some sensor footprint of every point in an environment—for vehicles with Dubins motion constraints. We extend previous work that solves this coverage problem as a traveling salesman problem (TSP) by introducing a practical heuristic algorithm to reduce runtime while maintaining near-optimal path length. Furthermore, we show that generating an optimal coverage path is NP-hard by reducing from the Exact Cover problem, which provides justification for our algorithm's conversion of Dubins coverage instances to TSP instances. Extensive experiments demonstrate that the algorithm does indeed produce length paths comparable to optimal in significantly less time.**

I. Introduction

Coverage path planning is often described as selecting a path that passes within a given range of every obstacle-free point in an environment [2], [3], [7]. The range is given in terms of a sensor radius or the robot's physical dimensions. Generating this path is fundamental in many problems robots are currently solving or are ideally suited to solve. Tasks such as lawn maintenance, floor cleaning, agriculture, environmental monitoring, humanitarian demining, and numerous naval applications such as mine sweeping and search and rescue can be stated as coverage path planning problems.

The complexity of these problems depends on factors such as the environment complexity and the motion constraints of the covering vehicle. Open, obstacle-free environments are easier to plan for than tightly bounded environments with obstacles. A holonomic vehicle can follow any path in its environment without limitation, while one with non-holonomic constraints is not only restricted in their movement, but may not even be capable of covering a given space. A common movement constraint is a restriction on the minimum turning radius of a vehicle, as angular acceleration is limited in many of the vehicles of practical interest.

This paper considers the coverage problem for the Dubins vehicle [4], which has just this sort of constraint. Without limiting ourselves to one particular robot model, a fixed-wing aircraft performing aerial coverage with a camera or a surface water vehicle performing coverage of bodies of water with sonar are representative vehicles for our plans. In the case of a water vehicle, we assume the environment and a portion of its surroundings are sufficiently deep to avoid collisions. In other words, obstacles are areas where no coverage is necessary but passing through is feasible. Practical implementations often utilize vehicles with Dubins kinematics, such as boats [5] or fixed wing UAVs [19], without addressing the turning radius constraint while solving the coverage problem, leaving the low-level controller to handle the trajectory generation.

The authors are with the Department of Computer Science and Engineering, University of South Carolina, 301 Main St., Columbia, SC 29208, USA. [lewisjs4, yiannisr,jokane]@cse.sc.edu

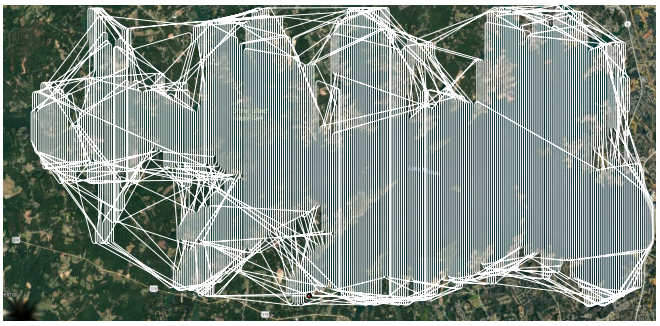

Fig. 1. A path by our planner covering a lake in a search-and-rescue simulation.

The method we use for path planning is an extension of the approach first described by Yu, Roppel, and Hung [25], in which they addressed the problem of visiting every row in a farm field. Their method reduced the constrained coverage problem to solving a variant of the traveling salesman problem (TSP) called set TSP or generalized TSP. The authors mapped each direction in which each row might be covered to a node in the graph and noted that two directions covering the same row would result in two nodes in the same set. They then determined the shortest circuit covering one node in each set and use this to determine the order and direction each row should be visited. Recall, however, that no efficient algorithms are generally believed to exist for TSP, since TSP is an NP-hard problem.

The two contributions of this paper are (1) to show that Yu, Roppel, and Hung's decision to map their coverage problem to an NP-hard problem is appropriate, by proving that the optimal Dubins coverage problem is NP-complete and (2) to introduce an algorithm to greatly improve the runtime of this planner by sacrificing some optimality.

We refer to our planner's paths as semi-boustrophedon because, while we do make use of the boustrophedon cellular decomposition (BCD) [3], we do not limit coverage strictly to boustrophedon paths. Instead, boustrophedon patterns emerge in the path when they are the optimal options for the planner.

We use the vehicle constraints to guide our planner toward circuits that would be useful in a real-world scenario. We assume that a robot is provided with some area for which coverage is desired. While we do not consider the area between regions we cover, we do not specifically avoid them.

II. Related work

Similar to Xu, Viriyasuthee, and Rekleitis [22], we address the problem of coverage path planning for a known environment. Our objective is to produce shortest distance paths, extending the work of [22]. Xu, et al. use a boustrophedon decomposition to form a Reeb Graph of their environments. The cells become the edges of a graph with the critical points as the nodes. They solve the Chinese postman problem (CPP) for this graph and use the results to build a coverage path. We extend their work by choosing a different graph, in which our nodes are the individual passes necessary to cover each cell and edges are the weights to transition from one pass to another. Instead of solving the CPP for our graph, we solve a generalized TSP.

The way in which we cover individual cells was first demonstrated in the work by Yu, Roppel, and Hung [25]. In their work, the authors map their Dubins coverage problem to a set TSP by an implement-width discretization of their environment. They treat each axis-aligned cell-crossing as a node in a graph. They modify the graph to a set TSP problem with a set containing each direction a cell-crossing might be covered. Finally, they use the TSP circuit to cover the cell. Though a specific environmental decomposition was not presented in that paper, there are many known techniques for dealing with non-convex environments [17] and the Boustrophedon decompositions used in [24] is where we begin our decomposition.

In work by both Huang [10] and Yao [23], the environmental decomposition is used to reduce path length by minimizing the amount of rotation and so environmental partitioning is very important. That line of research is orthogonal to our own; we use a simple axis-aligned decomposition of the environment, and focus instead on allowing the minimum turning radius of the robot to guide our search for good plans.

Gabriely and Rimon [6] also decompose their environment into a graph, using a grid in which cells are the nodes of a spanning tree. Their construction, however, is online and represents what the robot has covered, while our graph is constructed from a map and is used offline to generate the coverage path.

In Kong, Peng, and Rekleitis [12] the environment is decomposed into boustrophedon cells online by multiple robots covering a "virtually bounded" space. Our algorithm could be extended to a multi-robot solution using a similar approach. Likewise, Acar and Choset [1] build a decomposition and graph online to cover all available areas, but also require a backtracking step. Our work is similar in the way we use decompositions for path planning, however our algorithm is offline and single-robot.

The metric traveling salesman problem with Dubins curve constraints, called Dubins traveling salesman problem, has been well studied [16], [21]. It is defined in the same way as the TSP, but adds a new constraint—that the path to visit all nodes must consist of line segments and curves of a given minimum radius. The Dubins TSP is closely related to our problem. We show that the path to cover a corresponding coverage problem can be extracted from a plan to visit all the nodes of a certain Dubins TSP.

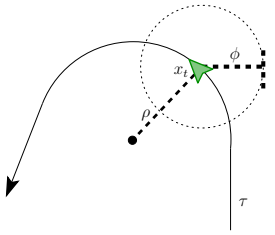

Fig. 2. The robot executing plan τ, at position x_t, with a minimum turn radius ρ, and a sensor footprint ϕ.

Fig. 3. A non-convex, disconnected environment solvable by our algorithm.

The coverage problem for a Dubins vehicle considered in Savla, Bullo, and Frazzoli [20] addresses the coverage problem using a Dubins vehicle from a control-theoretic approach. They make no attempt to discretize their environment or form plans to cover it. Our problem differs in that we are very explicit in our decomposition and use it to guide a structured plan to completely cover all area(s) of interest.

III. Problem statement

This section formalizes the global path planning problem presented in this paper. These plans provide coverage for a given area for a Dubins vehicle—that is, for a robot that moves forward at a constant speed, constrained by a minimum turn radius. We call this problem *optimal Dubins coverage (DCov)*.

A point robot moves in the plane with position and orientation. Its state space is $\mathbb{R}^2 \times [0, 2\pi)$. At any given time t the robot's pose is the tuple (x_t, θ_t), in which $x_t \in \mathbb{R}^2$ and $\theta_t \in [0, 2\pi)$. The robot has a sensor which allows it to observe a disk centered on its position with radius ϕ. The robot's translations are limited by a constraint on its minimum turning radius ρ and a constraint that movement maintains a constant forward speed. These constraints result in a vehicle capable of following only Dubins paths [4]. Figure 2 illustrates this notation.

We consider a bounded polygonal subset of the plane, denoted $P \subset \mathbb{R}^2$. We allow P to be possibly disconnected and/or non-convex areas of interest as illustrated in Figure 3. The goal is to generate a plan τ defined as

$$\tau \colon [0, T] \longrightarrow \mathbb{R}^2 \times [0, 2\pi)\,, \tag{1}$$

in which T is a finite termination time for the plan, obeying the robot's motion constraints. We call τ a *coverage path* if, for every point $p \in P$, there exists a time $t \in [0, T]$ for which

$$|x_t - p| \leq \phi, \tag{2}$$

where $|x_t - p|$ is the Euclidean norm. Our goal is to compute an optimal coverage path, in the sense of minimizing the length of the plan τ.

Optimal Dubins Coverage Problem (DCov)

***Input*:** A polygon P, sensor footprint ϕ, and minimum turning radius ρ.

***Output*:** A plan τ of minimum length, which when followed by a robot with sensor footprint ϕ and minimum turning radius ρ results in coverage of the polygon P.

We later refer to the length of the optimal path for a given P, ϕ, and ρ as DCov(P, ϕ, ρ).

IV. Hardness of Optimal Dubins Coverage

In this section, we establish that DCov, when cast as a decision problem, is NP-Complete. Specifically, we consider the following problem.

Dubins Coverage (Decision Version)

Input: A polygon P, sensor footprint ϕ, minimum turning radius ρ, and maximum path length $d \in \mathbb{R}^+$.

Output: *YES* if there exists a τ, which when followed by a robot with sensor footprint ϕ and minimum turning radius ρ which results in coverage of the polygon P with length such that $|\tau| \leq d$, *NO* otherwise.

The proof, which proceeds by reduction from Exact Cover [8], [9]—a known NP-complete problem—draws heavily from existing hardness proofs for the Euclidean Traveling Salesman (ETSP) [9], [18] and Dubins Traveling Salesman (DTSP) [16] problems.

In what follows, we write $\mathrm{ETSP}(Q)$ to refer to the length of the shortest path that visits each of the points in a given finite set of points Q. Likewise, we write $\mathrm{DTSP}(Q, \rho)$ for the length of the shortest Dubins curve with turning radius ρ that visits every point in Q.

The Exact Cover problem from which we reduce is defined thusly.

Exact Cover (EC)

***Input*:** A finite set U and a finite family F of subsets of U.

***Output*:** *YES* if there exists a subfamily of F, F' consisting of disjoint sets, such that F' covers U, *NO* otherwise.

Papadimitriou [18] describes a construction that maps an instance of EC to a set of points $Q = \{q_1, \ldots, q_m\}$ in the plane with size $O(n^2)$. The construction also produces numbers $L > 0$ and $\delta > 0$, and provides two guarantees. If the EC instance has a solution, then

$$\mathrm{ETSP}(Q) \leq L. \tag{3}$$

If the EC instance has no solution, then

$$\mathrm{ETSP}(Q) \geq L + \delta. \tag{4}$$

Our reduction from EC to DCov utilizes this construction directly. Given an instance of EC, we form an instance of DCov as follows.

1) Execute Papadimitriou's construction to obtain Q, L, and δ.
2) Choose, for the minimum turn radius ρ, any positive value
$$\rho \leq \frac{\delta}{2m\kappa\pi}, \tag{5}$$
in which $\kappa \approx 2.6575$ is the constant that appears in Ny, Feron, and Frazzoli [16].
3) Choose, for the sensor footprint ϕ, any positive value
$$\phi \leq \delta/(2m) - \kappa\pi\rho. \tag{6}$$
Note that Equation 5 guarantees the existence of a positive ϕ satisfying this constraint.
4) Select $P = \bigcup_{q \in Q} \mathcal{S}(q, \phi)$, in which $\mathcal{S}(x, r)$ refers to an axis-aligned square centered at x, with diagonal length r.
5) Set
$$d = L + \kappa\lceil n/2 \rceil \pi\rho, \tag{7}$$
where n is the number of points in Q.

This construction clearly takes polynomial time. To show that it is indeed a reduction from EC to DCov, we will use three lemmas, two from the literature and one original.

Lemma 1: (Savla, Frazzoli, and Bullo [21], Theorem 4.2) There exists a constant $\kappa < 2.658$ such that, for any finite set of points Q and any turning radius ρ, we have

$$ETSP(Q) \leq DTSP(P, \rho) \leq ETSP(P) + \kappa \left\lceil \frac{n}{2} \right\rceil \pi\rho. \tag{8}$$

Lemma 2: (Savla, Frazzoli, and Bullo [21], Theorem 3.1) For two planar poses whose positions are separated by distance d, the shortest Dubins curve connecting those poses has length at most $d + \kappa\pi\rho$.

Lemma 3: For any set Q of m points in the plane, any sensor footprint ϕ and any minimum turning radius ρ, let

$P = \bigcup_{q \in Q} \mathcal{S}(q, \phi)$. Then we have

$$\mathrm{DCov}\,(P, \rho, \phi) \leq \mathrm{DTSP}(Q, \rho) \tag{9}$$

and

$$\mathrm{DTSP}(Q, \rho) \leq \mathrm{DCov}\,(P, \rho, \phi) + 2m(\phi + \kappa\pi\rho). \tag{10}$$

Proof: For Equation 9, observe that, when the robot is at any point $q \in Q$, its sensor footprint covers all of $\mathcal{S}(q, \phi)$. Therefore, any path which visits each point in Q also covers all of P.

For Equation 10, consider a coverage path τ for P. We form a new path τ' identical to τ, except that we insert some additional path segments to guarantee that τ' passes through each $q \in Q$. Because τ is a coverage path for P and $Q \subset P$, we know that for each each $q \in Q$, there exists some t at which the robot's position $x(t)$ passes within distance ϕ of q, that is, $|x(t) - q| \leq \phi$. At this point, we insert into τ' a Dubins curve from $\tau(t)$ to q (with arbitrary orientation) and from this pose back to $\tau(t)$. Lemma 2 ensures that each of these two path segments have length no longer than $\phi + \kappa\pi\rho$. The total length of all of these 'repairs' is therefore bounded by $2m(\phi + \kappa\pi\rho)$, completing the proof. ■

We can now state the main result of this section.

***Theorem** 1:* DCov is NP-hard.

Proof: Reduction from Exact Cover, using the construction outlined above. We need to show that the EC instance has a solution if and only if the corresponding DCov instance (P, ϕ, ρ, d) has a coverage path of length at most d. We write Q to denote the finite point set generated by Papadimitriou's construction.

Suppose the EC instance has a solution. Then we have

$$\begin{aligned} \mathrm{DCov}(P, \phi, \rho) &\leq \mathrm{DTSP}(Q, \rho) \\ &\leq \mathrm{ETSP}(Q, \rho) + \kappa \left\lceil \frac{m}{2} \right\rceil \pi\rho \\ &\leq L + \kappa \left\lceil \frac{m}{2} \right\rceil \pi\rho \\ &= d. \end{aligned}$$

Here we have used, in order, Lemma 3, Lemma 1, Equation 3, and Equation 7.

For the other direction, suppose the EC instance has no solution. In that case, we know

$$\begin{aligned} \mathrm{DCov}(P, \phi, \rho) &\geq \mathrm{DTSP}(Q, \rho) - 2m(\phi + \kappa\pi\rho) \\ &\geq \mathrm{ETSP}(Q, \rho) - 2m(\phi + \kappa\pi\rho) \\ &\geq L + \delta - 2m(\phi + \kappa\pi\rho) \\ &\geq L \\ &\geq d \end{aligned}$$

These inequalities derive from Lemma 3, Lemma 1, Equation 4, Equations 6 and 5, and Equation 7 respectively. ■

***Corollary** 1:* DCov is NP-Complete.

Proof: It remains only to show that DCov is in NP. We can use the coverage path τ as the certificate, and compute the region covered by τ, a finite union of circles and rectangles. Then verify (1) that this region is a superset of P using a standard clipping algorithm, (2) that the length of τ is at most d, and (3) that τ is indeed a Dubins curve for turning radius ρ. ■

Algorithm 1 DUBINSCOVERAGE(P, ϕ, ρ)

$\mathcal{D} \leftarrow$ BOUSTROPHEDONCELLDECOMPOSITION(P)
$\mathcal{P} \leftarrow$ REFINEINTOPASSES$(\mathcal{D}, \phi)$
$V \leftarrow \mathcal{P} \times \{\uparrow, \downarrow\}$
$E \leftarrow V \times V$
$w \leftarrow$ COMPUTEWEIGHTS(E, ρ)
$(V', E') \leftarrow$ REDUCEGRAPH(V, E, w)
$(v_1, \ldots, v_n) \leftarrow$ SOLVEGTSP(V', E', w)
$\tau \leftarrow$ CONSTRUCTPLAN$(v_1, \ldots, v_n)$
return τ

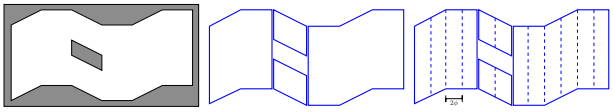

Fig. 4. Decomposing the environment into passes. [left] The original environment P. [middle] A decomposition of P into 4 cells, each y-monotone, via Boustrophedon Cell Decomposition. [right] A refinement the above decomposition into 14 passes of width at most 2ϕ.

V. GRAPH REDUCTION FOR EFFICIENT COVERAGE

In the previous section we showed that unless $P = NP$, DCov cannot be solved optimally by any polynomial time algorithm. Indeed, the best known algorithm, due to Yu, Roppel, and Hung [25], scales quite poorly as the number of passes increases. In addition, the algorithm as originally presented omits a number of geometric details that are essential for a complete implementation. In this section, we introduce an improvement to that algorithm that generates high-quality coverage paths much more efficiently. Algorithm 1 summarizes our approach.

To begin, we require a decomposition of P into simple pieces. To accomplish this, the planner partitions P into monotone regions using the boustrophedon cell decomposition (BCD) algorithm [3] (Alg. 1, line 1).

Next, we further refine the decomposition, cell-by-cell, into a set $\mathcal{P}$ of "passes" $P_i \in \mathcal{P}$ such that each pass is a connected region that can be covered in a single sweep from end-to-end (Alg. 1, line 2). Without loss of generality, we construct these passes with vertical orientations, utilizing divisions parallel to the y-axis; see Figure 4. In a practical deployment the choice for the direction of coverage would be affected by a variety of factors, such as, wind- or sea-current direction, desired sensor placement, location of obstacles, etc [22].

The result is a set of passes, each no more than 2ϕ wide, such that the robot can cover a pass in a single sweep of its sensor. To cover a given pass P_i, a robot must follow a segment of the pass's vertical bisector. We call this segment the *covering path segment* for the pass. The covering path segment, specified by its top point $t\,(P_i)$ and its bottom point $b\,(P_i)$, is defined as the smallest segment along the vertical

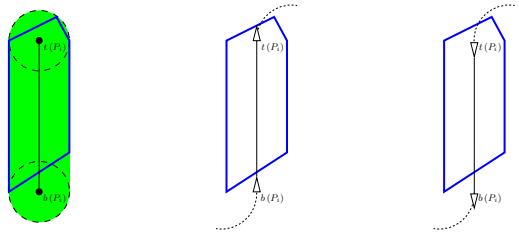

Fig. 5. [left] A pass P_i and its covering path segment. The shaded region area is covered by the robot's sensor as it traverses the covering path segment. [middle] The graph vertex $(P, \uparrow)$ has entry pose $(b(P_i), \frac{\pi}{2})$ and exit pose $(t(P_i), \frac{\pi}{2})$. [right] The graph vertex $(P, \downarrow)$ has entry pose $(t(P_i), \frac{3\pi}{2})$ and exit pose $(b(P_i), \frac{3\pi}{2})$.

bisector of P_i for which

$$P_i \subseteq \bigcup_{q \in \overline{t(P_i)b(P_i)}} \mathcal{B}(q, 2\phi)$$

as illustrated in Figure 5. When every pass $P_i \in \mathcal{P}$ is covered, then P is covered.

We must next map our passes into the vertices of a graph and create the necessary edges. Once we have a graph and the solution to a TSP on the graph, we need only map the circuit to a path. These steps are described below.

- The vertex set V consists of $2|\mathcal{P}|$ vertices defined as $V = \mathcal{P} \times \{\uparrow, \downarrow\}$. The interpretation is that visiting a vertex indicates that the robot should cover the given pass by traversing that pass's covering path segment in the given direction. For each *up* vertex $(P_i, \uparrow)$ we define the *entry pose* as $\left(b(P_i), \frac{\pi}{2}\right)$ and the *exit pose* as $\left(t(P_i), \frac{\pi}{2}\right)$. For *down* vertices $(P_i, \downarrow)$, we define entry and exit poses similarly *mutatis mutandis*. Figure 5 illustrates the construction.
- The edge set E contains $4|\mathcal{P}|^2$ elements, connecting all pairs of vertices. For a given edge $e_{ij} \in E$, its weight w_{ij} is defined as the length of a Dubins curve from the exit pose of the source vertex v_i, to the entry pose of the target vertex v_j (Alg 1, line 5).
- Given a circuit that visits, for each pass P_i, either $(P_i, \uparrow)$ or $(P_i, \downarrow)$, we can directly construct a coverage path τ by alternating covering path segments with Dubins curves between the successive passes in the circuit.

To generate the required circuit mentioned above, we solve an instance of the *generalized traveling salesman problem (GTSP)* [15].

Generalized TSP

***Input*:** A weighted graph $G = (V, E)$ and a partition of V into nodesets $S_1, \ldots, S_m$.

***Output*:** The shortest cycle in G that visits each nodeset exactly once.

This problem is readily shown to be NP-hard by reduction from the standard traveling salesman problem and GTSP instances can be converted to instances of asymmetric TSP (aTSP) by a straightforward linear time construction [13].

As in Yu et al., we form a GTSP instance, by partitioning the nodes of our graph into $|\mathcal{P}|$ nodesets

$$\{(P_1, \uparrow), (P_1, \downarrow)\}, \ldots, \{(P_{|\mathcal{P}|}, \uparrow), (P_{|\mathcal{P}|}, \downarrow)\},$$

each containing the two complementary vertices for a single pass. This forces a GTSP solution to visit each pass exactly once, either its $\uparrow$ or its $\downarrow$ vertex. While the authors found no solvers for the GTSP, there exist heuristic optimizers for the aTSP are available [14] that are fast in practice (Alg 1, line 7).

The result of SOLVEGTSP is a circuit providing the order in which each nodeset should be visited. Due to our construction of aTSP from GTSP, the circuit contains each pass in the environment twice, once for the up pass and once for the down pass in either order. To build τ, note that the first occurrence of a pass is the correct direction in which the pass should be covered. For each of these, we link the coverage path segment with a Dubins curve to the first pass in the next node set given in the circuit.

Even with fast heuristic optimizers, GTSP is a difficult problem. Performance is directly related to the complexity of the graph, in that the larger the number of vertices and edges, the larger the search space. Intuitively, the more nodes to visit and more connections to consider, the more possible paths to visit all nodes, and therefore the longer the run time.

To combat this complexity, we considered several heuristics which use subgraphs of G, omitting some vertices and/or edges. The objective is to eliminate nodes or edges which are unlikely to be part of the optimal solution. In the end we determined, experimentally, that the ability to cover a pass in either direction did not significantly improve the path length of coverage, though it did drastically affect the run time. This strategy, which we call **specified directions** decomposition (SDD), is to impose an ordering of passes within each cell of the BCD. We randomly select one of the two directed nodes for the first pass and then alternate that selection of nodes for each pass (Alg 1, line 6). The opposite node for each pass, along with all of its incident edges, is deleted from the graph.

In addition to the specified directions strategy, we also investigated three other graph reduction heuristics whose performance did not prove as effective.

1) **Alternating directions** — Delete all edges that connect vertices with the same direction. That is, we remove any edge from an $\uparrow$ vertex to an $\uparrow$ vertex, or from a $\downarrow$ vertex to a $\downarrow$ vertex. The effect is to force the robot to alternate between upward and downward sweeps, without pre-specifying the coverage direction for each pass.
2) **Restricted weights** — Compute the mean μ and standard deviation σ of the edge weights. Select a parameter $z \in \mathbb{R}$ and delete all edges whose weight exceeds $\mu + z\sigma$. The observation we make is that when all passes are connected to all other passes, there are a very large number of long edges, connecting passes

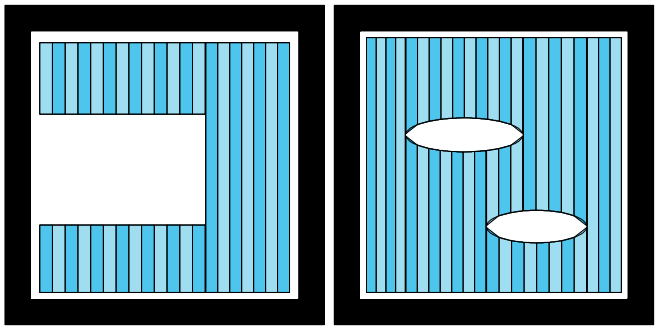

Fig. 6. [left] An environment wrapping around an area which does not requirecoverage. [right] An environment with two areas which do not need coverage.

which would not be directly connected in an optimal plan.

3) **Proximate passes** — Perform deletions based on distance in the x-axis between cells. This approach most closely realizes our goal of allowing paths which cover cells partially–depending on how many times the cell is crossed–without giving up the ability to jump areas of disinterest or obstacles.

VI. Experiments

We used C++ to implement our algorithms in simulation. To solve GTSP instances, our implementation uses *LocalSolver*, "a hybrid mathematical programming solver," [14] as our TSP solver. LocalSolver does not solve the TSP exactly, rather optimizes toward a solution. Because LocalSolver does not compute an optimal solution, it requires a termination condition specified either in time or in iterations. Iterations are linked to the complexity of the problem, so we use this as our termination metric–LocalSolver can complete more iterations per second for simpler problems. Due to the construction of an aTSP graph from a GTSP graph and the use of an optimizer, there are times when the path returned is invalid. This is discussed in Section VI-B.

A. Comparison Experiments

To compare the algorithms, we conducted experiments on two simple environments, shown in Figure 6. The left environment represents an area with a large hole in the middle with no interest for coverage. The right environment containing two smaller holes. The left environment might represent a crop-duster spraying farmland with a farm house in the middle. The right could represent covering a body of water with islands which do not need to be covered. In a set of experiments, we compare the run time to solve the coverage problem using each algorithm on both environments. In all of the experiments, we chose an arbitrary 5,000,000 iterations as our halting condition. This gives us a means by which we may judge the improvement our different algorithms may provide.

1) Figure 7: The first experiment repeatedly covers the left environment from Figure 6 holding the sensor ϕ, constant and varying the minimum turning radius ρ. The restricted weights decomposition, though having the worst runtime growth, eventually leads to the best paths. The proximate passes decomposition does well in runtime against the complete graph, but suffers in path length. The clear winner in this case, as the problem gets harder, is the SDD. Not only are its paths of nearly identical length as the complete graph, its runtime growth is markedly better as the problem's difficulty increases. The alternating edges decomposition did not show up in this graph, due to the degeneracy described above.

2) Figure 8: Our second experiment again uses the left environment of Figure 6. However, this time we keep ρ constant and vary ϕ. Again, it is clear that the SDD algorithm is an improvement over the complete graph in run time. The gap is smaller this time, appearing to be a constant offset of the complete decomposition. When the ϕ is very small, implying more passes required to cover the environment and a larger problem, the difference in runtime is obvious. The other graph decompositions perform on par with the complete decomposition. For graphs with few nodes, there is not a clear winner.

3) Figure 9: The third experiment uses the right environment shown in Figure 6, holds ϕ constant while varying ρ and again demonstrates the runtime improvement with SDD. This experiment demonstrates the largest improvement in runtime when using the SDD. As the problem becomes more difficult, through increasing ρ, the growth rate of SDD appears to be possibly sub-linear. The proximate passes decomposition also performs well as difficulty increases. It begins at the same starting point as restricted weights and the complete graph, but maintains a much better search time as the problem difficulty increases. Again, however, its path length is sub par compared to the other decompositions. Though it is unclear why, the restricted edges decomposition results in similar length paths as the complete graph, but with a much worse run time.

4) Figure 10: In the fourth experiment we held ρ constant and varied ϕ repeatedly solving the coverage problem for the right environment depicted in Figure 6. Again, the SDD has a better run time than the complete graph with similar path lengths. The biggest run time difference is experienced when the number of vertices is large. The restricted weights graph decomposition experiences points at which its run time is worse than the complete graph, but it and the others perform as well as the complete graph.

B. Search & Rescue Simulation

In contrast to the rectilinear environments, what follows is a simulation of a search and rescue scenario. We assume that a person is lost somewhere on a lake as shown in Figure 1. The person is wearing a life-preserver which would offer an average of one half meter of visible area across. By Johnson's criteria [11], this would require around 6 pixels of height or width to recognize therefore a resolution of 12 pixels/meter is required. Given a standard 1080p camera with a 60° field of view, we determine that a UAV must fly at a height of 78m to have a cone of 90m for the required resolution. We executed

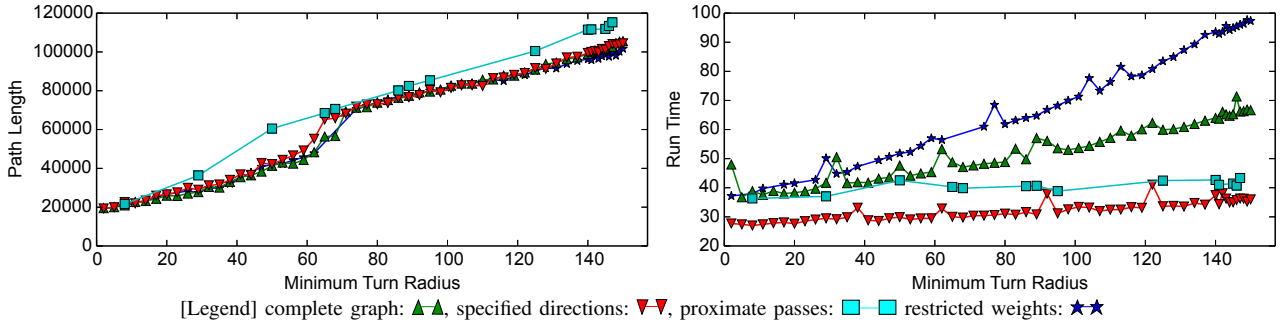

Fig. 7. Coverage of the left environment Figure 6 with ϕ constant at 10.0, varying ρ.

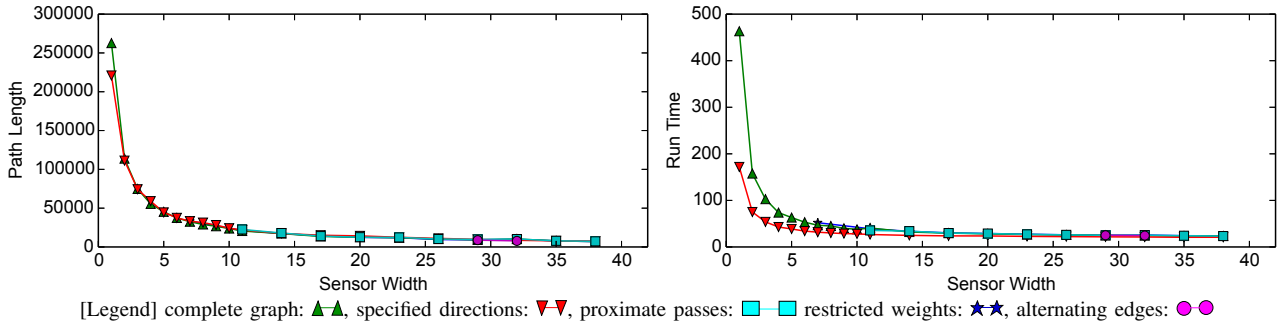

Fig. 8. Coverage of the left environment in Figure 6 with ϕ varying, and ρ constant at 15.0.

each of our decompositions and report the results in the following table, allowing LocalSolver 5,000,000 iterations to optimize each TSP. It is evident that SDD is the superior

Decomposition	**Run Time**	**Distance**
Complete graph	4m 34s	3662km
Specified directions	2m 26s	3532km
Alternating directions	4m 25s	FAIL
Restricted Weights	4m 52s	3520km
Proximate Passes	2m 7s	FAIL

algorithm in this simulation. Even though the complete decomposition contains the same path found in the SDD, it was not found due to the volume of nodes and edges being considered by the optimizer. Additionally, the SDD found the path in roughly half the time. The restricted weight decomposition, discarding edges with "too high" resulted in the longest run time in this simulation, but the best path length. The alternating directions and proximate passes decompositions failed to generate a valid path. This occurs due to the reduction from GTSP to aTSP described in [15]. Because LocalSolver is an optimizer, not an exact solver, it gets stuck in local optima. This behavior may lead to paths which violate the Noon and Bean's construction. The violation results in paths which do not always visit all node sets of GTSP and therefore do not cover an environment.

VII. Summary

This paper presented our solution for the coverage problem using a mobile robot with kinematic constraints. We showed that the Dubins Coverage problem is an NP-complete problem by reduction from the Exact Coverage Problem. We presented an algorithm for generating Dubins Paths which when followed provide paths by which a robot capable of following a Dubins Path might cover an environment. Next we reduced the number of ways the robot might cover an environment by considering less options, yet did not substantially increase the length of coverage plans. Finally, we presented the results of our simulation by showing a side-by-side comparison of the algorithms' performance across two different environments varying the size of the robot's sensor footprint and minimum turn radius.

The analysis of those comparisons led us to the conclusion that there are a large number of edges which only seem to serve to make the problem larger without adding any useful choices. Our attempts to reduce the number of edges seems mostly successful when we chose to only connect cells within relatively close proximity of each other. We also noted that forcing the problem to cover passes in a specific direction did not seem to hurt the length of the best path and resulted in the expected solution speed up.

In the future, we will continue the exploration of this problem by considering both a physical implementation as well as a multiple robot formulation. Multiple robots often make tasks easier, when good cooperative planners can be found to split a task. A good planner to break up the environment such that n robots might work together to cover the environment could not only offer the linear decrease in calculation time, but could also offer a linear decrease in the time to execute the plan. Additionally, we are currently developing a fleet of six ASVs based on the Mokai ES-Kape at our university. Deploying the presented algorithm on one of these vessels will provide field testing validation. Furthermore, a multi-robot coverage extension of the above described approach will be tested with SONAR mapping as the target application.

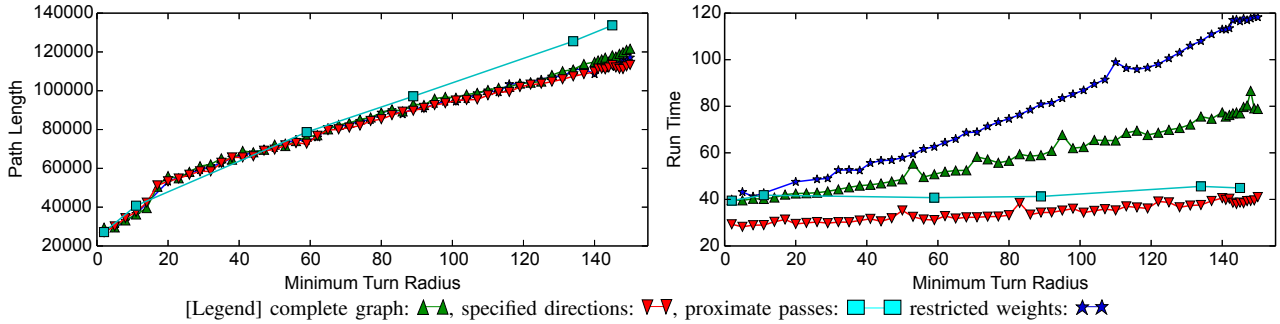

Fig. 9. Coverage of the right environment in Figure 6 with ϕ constant at 10.0, varying ρ.

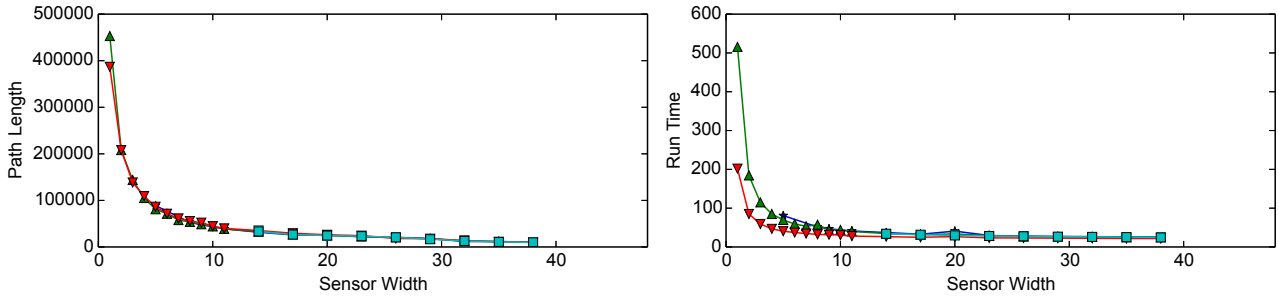

Fig. 10. Coverage of the right environment in Figure 6 with ϕ varying, and ρ constant at 15.0.

ACKNOWLEDGEMENT

The authors would like to thank the generous support of the Google Faculty Research Award and the National Science Foundation grants (NSF 0953503, 1513203, 1526862, and 1637876). This work was supported in part by the South Carolina Honors College Science Undergraduate Research Funding Program

REFERENCES

[1] E. U. Acar and H. Choset, "Sensor-based coverage of unknown environments: Incremental construction of morse decompositions," *The International Journal of Robotics Research*, vol. 21, no. 4, pp. 345–366, April 2002.

[2] H. Choset, "Coverage for robotics - a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.

[3] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon decomposition," in *International Conference on Field and Service Robotics*, 1997.

[4] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[5] P. M. Forooshani and M. Jenkin, "Sensor coverage with a heterogeneous fleet of autonomous surface vessels," in *IEEE International Conference on Information and Automation*, 2015, pp. 571–576.

[6] Y. Gabriely and E. Rimon, "Spiral-stc: an on-line coverage algorithm of grid environments by a mobile robot," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2002.

[7] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[8] M. Garey and D. Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[9] R. Graham, M. Garey, and D. Johnson, "Some np-complete geometric problems," in *Proc. ACM Symposium on Theory of Computing*, 1976.

[10] W. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Proc. the IEEE Int. Conf. on Robotics and Automation*, vol. 1, 2001, pp. 27 – 32.

[11] J. Johnson, "Analysis of image forming systems," in *Proceedings of the Image Intensifier Symposium*, 1958, pp. 244–273.

[12] C. S. Kong, A. P. New, and I. Rekleitis, "Distributed coverage with multi-robot system," in *Proc. IEEE International Conference on Robotics and Automation*, 2006.

[13] R. Kumar and H. Li, "On asymmetric TSP: Transformation to symmetric tsp and performance bound," University of Kentucky, Department of Electrical Engineering, Tech. Rep., Feb 2014.

[14] "LOCALSOLVER," 2017, http://www.localsolver.com.

[15] C. E. Noon and J. C. Bean, "An efficient transformation of the generalized traveling salesman problem," *INFOR: Information Systems and Operational Research*, vol. 31, no. 1, pp. 39–44, 1993.

[16] J. L. Ny, E. Feron, and E. Frazzoli, "On the dubins traveling salesman problem," *IEEE Trans. Automat. Contr.*, vol. 57, pp. 265–270, 2012.

[17] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines." in *Journal of Field Robotics*, vol. 26, no. 8, 2009, pp. 651–668.

[18] C. H. Papadimitriou, "The euclidean travelling salesman problem is np-complete," *Theoretical Computer Science*, vol. 4, no. 3, pp. 237 – 244, 1977.

[19] L. Paull, C. Thibault, A. Nagaty, M. Seto, and H. Li, "Sensor-driven area coverage for an autonomous fixed-wing unmanned aerial vehicle," *IEEE transactions on cybernetics*, vol. 44, no. 9, pp. 1605–1618, 2014.

[20] K. Savla, F. Bullo, and E. Frazzoli, "The coverage problem for loitering dubins vehicles," in *Decision and Control, 2007 46th IEEE Conference on*, Dec 2007, pp. 1398–1403.

[21] K. Savla, E. Frazzoli, and F. Bullo, "On the point-to-point and traveling salesperson problems for dubins' vehicle." in *American Control Conference*, June 2005, pp. 786–791.

[22] A. Xu, C. Viriyasuthee, and I. Rekleitis, "Efficient complete coverage of a known arbitrary environment with applications to aerial operations," *Autonomous Robots*, vol. 36, no. 4, pp. 365–381, 2014.

[23] Z. Yao, "Finding efficient robot path for the complete coverage of a known space," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Beijing, China, 2006, pp. 3369–3374.

[24] X. Yu and J. Y. Hung, "Coverage path planning based on a multiple sweep line decomposition," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, Nov 2015, pp. 4052–4058.

[25] X. Yu, T. A. Roppel, and J. Y. Hung, "An optimization approach for planning robotic field coverage," in *Proc. Annual Conference of the IEEE Industrial Electronics Society*, 2015.