Approximation Algorithms for the Single Robot Line Coverage Problem

Saurav Agarwal(✉) and Srinivas Akella

University of North Carolina at Charlotte, Charlotte, NC 28223, USA
{sagarw10,sakella}@uncc.edu

Abstract. The line coverage problem is the task of *servicing* a given set of one-dimensional features in an environment. Its applications include the inspection of road networks, power lines, and oil and gas lines. The line coverage problem is a generalization of the standard arc routing problems, and is NP-hard in general. We address the single robot line coverage problem where the service and deadhead costs are distinct and asymmetric. We model the problem as an optimization problem that minimizes the total cost of travel on a given graph. We present approximation algorithms to obtain bounded solutions efficiently, using the minimum cost flow problem. We build the main algorithm in stages by considering three simpler subproblems. The subproblems are based on the structure of the *required graph*, i.e., the graph induced by the features that require servicing. We first present an optimal algorithm for the case of Eulerian graphs with only *required* edges. Next we consider general graphs, not necessarily Eulerian, with only required edges and present a 2-approximation algorithm. Finally, we consider the general case with both required and non-required edges. The approximation algorithm is dependent on the Asymmetric Traveling Salesperson Problem (ATSP), and is bounded by $\alpha(C) + 2$, where $\alpha(C)$ is the approximation factor of the ATSP algorithm with C connected components. Our upper bound is also an improvement over the existing results for the asymmetric rural postman problem.

Keywords: Line coverage · Arc routing problems · Path planning

1 Introduction

The *line coverage* problem is the task of servicing linear environment features. This paper considers the single robot version of the problem. The features to be serviced are modeled as 1D segments; all points along the segments must be visited. Examples of line coverage tasks include robotic inspection of road networks, power lines, and oil and gas lines. See Fig. 1 for an example.

In a *coverage* application, the robots are required to visit specified features in the environment. Such features may be a set of 2D regions, 1D line features, or points. *Area coverage*, the coverage of 2D regions, has been widely studied [3,10].

S. M. LaValle et al. (Eds.): WAFR 2020, SPAR 17, pp. 534–550, 2021.
https://doi.org/10.1007/978-3-030-66723-8_32

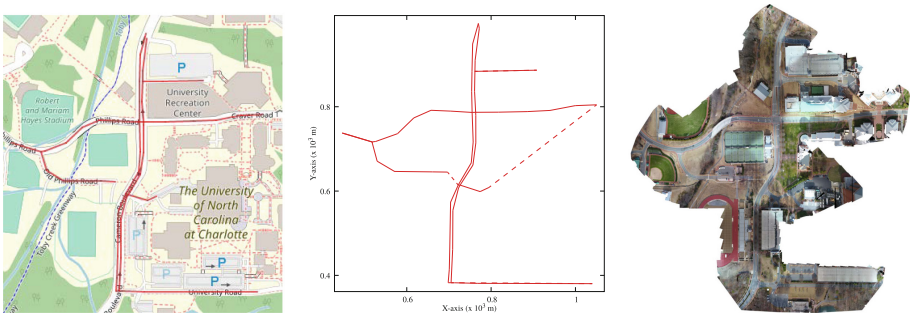

Fig. 1. Line coverage of a road network by a UAV. (left) A region of the UNC Charlotte campus; red lines show required edges to be serviced. Non-required edges, not shown, are straight lines between pairs of vertices. (middle) A tour to cover the road network is shown; dashed segments indicate deadheading travel. (right) An orthomosaic of the road network, from photos taken by the UAV along required edges of the tour.

Point coverage, the coverage of point features, involves node routing problems, such as the traveling salesperson problem and the vehicle routing problem [11]. *Line coverage*, modeled as coverage of all the required edges of an underlying graph, is related to arc routing problems [5]. This problem has received limited attention in the robotics community [1,6,7,12,19], and is the focus of this paper.

The line coverage problem is modeled using a graph. The edges are classified as *required* and *non-required*. Required edges correspond to the linear features to be covered, and the non-required edges can be used to travel from one vertex to another. The vertices represent the end points of the edges.

There are two modes of travel for the robot. The robot is said to be *servicing* a required edge if task-specific actions such as collecting sensor data are performed. Each required edge needs to be serviced exactly once. The robot may travel from one vertex to another without performing servicing. This is known as *deadheading* and both types of edges may be used any number of times for this purpose. There is a service cost and a deadhead cost (e.g., travel time) associated with each edge and they are incurred each time an edge is serviced or deadheaded, respectively. The sum of the service costs and the deadhead costs of the line coverage problem is to be minimized. As task-related servicing is not done during deadheading, deadhead costs are considered to be less than or equal to the service costs. For example, with travel time costs, a UAV servicing an edge by recording images may take longer than when deadheading. In standard arc routing problems, the service and deadhead costs are assumed to be identical, and hence the line coverage problem generalizes the standard problems.

In many robotics applications, the cost of travel is direction dependent. For example, for ground robots, the cost of traveling uphill can be significantly higher than that of traveling downhill. Similarly, for UAVs, the cost of an edge may differ in the two directions due to wind conditions. Hence, we consider the graph to have asymmetric edge costs for both servicing and deadheading.

We now define the *single robot line coverage problem* as the optimization problem of finding a tour that minimizes the total cost while ensuring that each of the required edges is serviced exactly once. The single robot line coverage problem is a generalization of the Rural Postman Problem (RPP) on asymmetric (or windy) graphs [5], described in Sect. 2. The NP-hardness of the RPP on asymmetric graphs implies the single robot line coverage problem is NP-hard. This makes it imperative to develop approximation algorithms. The line coverage problem with multiple robots is presented in [1] along with two heuristic algorithms. The single robot line coverage problem in this paper is a special case of [1], as we do not consider the robot capacity and the demands of edges.

In this paper we elucidate the single robot line coverage problem and develop approximation algorithms for it. We develop the analysis in stages from a simpler problem to the most general version. The contributions of the paper are: (1) We pose the single robot line coverage problem as an optimization problem, and develop an integer linear programming (ILP) model. (2) We present an optimal algorithm for Eulerian graphs with all the edges required to be serviced. (3) A 2-approximation algorithm is presented for general graphs, not necessarily Eulerian, with all the edges required to be serviced. (4) An $\alpha(C) + 2$ approximation algorithm is presented for the case with both required and non-required edges, where C is the number of connected components, and $\alpha(C)$ is the approximation factor for an algorithm for the asymmetric traveling salesperson problem.

The practical benefits of our line coverage approach are: (1) The algorithms ensure that the required edges are covered efficiently, thereby optimizing the total cost of tours, e.g., operation time. (2) In contrast to using area coverage, only the relevant features are inspected, thus reducing the inspection time, the amount of sensor data, and the time for data analysis.

2 Related Work

The line coverage problem belongs to the broad class of Arc Routing Problems (ARP). A hierarchy of standard arc routing problems and the line coverage problem for a single vehicle (i.e., a robot in our context), is shown in Fig. 2. The ARP is usually applied to transportation problems in which servicing is related to tasks such as delivery and pick up of goods [5]. Hence the travel times are used as costs, and have the same value whether the edge is serviced or deadheaded. Separate and asymmetric service costs are typically not considered.

Arc Routing Problems for a Single Vehicle: The Chinese Postman Problem (CPP) is to find an optimal tour such that every edge in the given undirected and connected graph is traversed at least once [5]. Edmonds and Johnson [8] used matching and network flows to solve the CPP on general undirected and directed graphs, and Eulerian mixed graphs. They also presented an approximation algorithm for the CPP on mixed graphs that are not necessarily Eulerian. Frederickson [9] presented a 5/3-approximation algorithm for the CPP on mixed graphs by using a combination of two approximation algorithms.

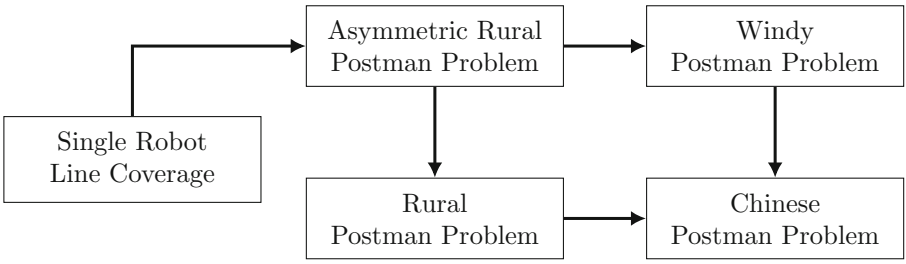

Fig. 2. A hierarchy of arc routing problems with single vehicle/robot. An arrow from problem A to problem B indicates B is a special case of A. The single robot line coverage problem is the most general problem of which the other problems are special cases.

The Windy Postman Problem (WPP) is the CPP with asymmetric edge costs, and is NP-hard [5]. Win [18] solved the WPP for Eulerian graphs in polynomial time using minimum cost flow. Win also designed a 2-approximation algorithm for WPP on general graphs using matching (to make the graph Eulerian) and minimum cost network flow. Raghavachari and Veerasamy [15] gave a 3/2-approximation for the CPP on asymmetric graphs.

When the edges to be serviced are a subset of the edges in the graph, we have the Rural Postman Problem (RPP) [5]. RPP is NP-hard. For RPP, Frederickson noted an exact recursive algorithm that is exponential only in the number of disconnected components. He also mentioned a 3/2-approximation algorithm similar to Christofides' TSP algorithm [4]. The asymmetric RPP considers asymmetric travel costs. van Bevern et al. [2] showed that if the n-vertex Asymmetric Traveling Salesman Problem (ATSP), subject to the triangle inequality, is $\alpha(n)$-approximable in $t(n)$ time, then n-vertex RPP on an asymmetric and mixed graph is $(\alpha(C) + 3)$-approximable in $O(t(C) + n^3 \log n)$ time, where C is the number of weakly connected components in the subgraph induced by required arcs and edges. The single robot line coverage problem is closely related to the asymmetric RPP. However, in asymmetric RPP the costs of deadheading and servicing a required edge are the same, and a required edge can be traversed more than once. Any instance of asymmetric RPP can be converted to that of the line coverage problem by setting the cost of deadheading a required edge to the cost of the edge. In the other direction, we can add non-required edges for each required edge with costs equal to the deadhead costs, and then ensure that each required edge is traversed exactly once in the asymmetric RPP solution.

Asymmetric Traveling Salesman Problem (ATSP): Svensson et al. [16] were recently the first to present a constant-factor approximation algorithm for the ATSP with the triangle inequality. Traub and Vygen [17] very recently improved the approximation ratio to $22 + \epsilon$, $\epsilon > 0$, for the ATSP.

Line Coverage in Robotics: Mixed integer linear programming formulations and heuristic algorithms have been proposed for coverage of road networks [6,12]. Heuristic algorithms for RPP with k vehicles were proposed for coverage of 2D

object boundaries in [7] and for street coverage in [19]. These consider neither asymmetric edge costs nor distinct service and deadhead costs.

3 The Single Robot Line Coverage Problem

We now model the single robot line coverage problem as a graph problem. We are given a connected graph $G(V, E, E_r)$, where V is the set of vertices, E is the set of edges, and $E_r \subseteq E$ is the set of required edges. Note that the set E can contain parallel edges between two vertices, i.e., we allow for G to be a multigraph. The *single robot line coverage problem* is to find a tour that minimizes the total cost of travel on the graph, such that all the required edges in E_r are serviced. The service and deadhead costs are given as inputs along with the graph.

For each edge in E we associate two directional arcs e and $\bar{e}$, and represent the edge as $(e, \bar{e}) \in E$. If a robot *services* a required edge in E_r in the direction e (or $\bar{e}$), then a service cost $s_c(e)$ (or $s_c(\bar{e})$) is incurred. If a robot traverses an edge without servicing it, the robot is said to be *deadheading*; for example, this occurs when a robot is traveling from a vertex of a required edge to that of another required edge. Both required and non-required edges may be deadheaded. Deadhead costs are denoted by $d_c(e)$ and $d_c(\bar{e})$. We use $s_c(A)$ and $d_c(A)$ to denote the corresponding sums of the service and deadhead costs for a set of arcs A. We denote by $\bar{A}$ the set of arcs oppositely directed to the arcs in A.

We consider the edge costs, for both servicing and deadheading, to be direction dependent. For example, $s_c(e)$ may differ from $s_c(\bar{e})$. The service and deadhead costs can be arbitrary positive numbers, with the constraint that the service cost of an edge is no less than the deadhead cost in the same direction. The costs, such as travel time, affect the objective function of the problem.

3.1 Preliminaries

Let $G(V, E, E_r)$ be a connected undirected graph for the line coverage problem, such that $E_r \subseteq E$. The subgraph $G_r(V_r, E_r)$ induced by the set of required edges E_r is called the *required graph* of G; $V_r \subseteq V$ is the set of vertices that have at least one edge in E_r incident on them. The set of non-required edges is denoted by $E_n = E \setminus E_r$. We define the set of all arcs to be $\mathcal{A} = \bigcup\{e, \bar{e}\},\ \forall (e, \bar{e}) \in E$. Similarly, $\mathcal{A}_r$ is defined for the set of required edges. If an arc a represents the travel direction from vertex u to vertex v, then the vertices u and v are called the tail $t(a)$ and head $h(a)$, respectively. We denote by $H(\mathcal{A}, v)$ all the arcs $a \in \mathcal{A}$ that have v as the head. Similarly, $T(\mathcal{A}, v)$ is defined for the tail. The *degree* of a vertex $v \in V$ is the number of edges incident on v. A *walk* in a graph G is a non-empty alternating sequence $v_1 e_1 v_2 e_2 \ldots e_k v_{k+1}$ of vertices in V and edges in E such that the tail of e_i is v_i and head of e_i is v_{i+1} for all $1 \leq i \leq k$. A *closed walk* is a walk with the same start and end vertices, i.e., $v_1 = v_{k+1}$. An *Euler tour* is a closed walk such that every edge in the graph is traversed exactly once. A graph that has an Euler tour is called *Eulerian*. It is well established that an undirected graph is Eulerian if and only if every vertex has even degree [14].

Let $D(V, A)$ be a *directed graph* (digraph) with V as the set of vertices, and A as the set of (directed) arcs. The digraph is strongly connected if there exists a path from any vertex in V to any other vertex in V. Analogous to the undirected graph, $T(A, v)$ and $H(A, v)$ are defined for the arc set A and $v \in V$. The *indegree* of a vertex $v \in V$, denoted by $\text{indeg}(v)$, is the number of arcs entering the vertex v. Similarly, the *outdegree* of a vertex $v \in V$, denoted by $\text{outdeg}(v)$, is the number of arcs going out of the vertex v. A digraph is Eulerian if and only if the graph is *balanced*, i.e., $\text{indeg}(v) = \text{outdeg}(v), \forall v \in V$. *Imbalance* $\nabla(A, v)$ at a vertex v is given by $\text{outdeg}(v) - \text{indeg}(v) = |T(A, v)| - |H(A, v)|$. Analogous to the undirected graph, a *diwalk* is a sequence $v_1 a_1 v_2 a_2 \dots a_k v_{k+1}$ of vertices and arcs in a digraph $D(V, A)$ such that the tail of a_i is v_i and head of a_i is v_{i+1}. A *closed diwalk* is a walk with the same start and end vertices. An *Eulerian tour* on an Eulerian digraph is a closed diwalk such that each arc is traversed exactly once. An Euler tour can be constructed from an Eulerian graph (or digraph) in $\mathcal{O}(|A|)$ computational time, see e.g., [14].

A *coverage tour* is a closed walk in a graph G such that all the required edges are serviced exactly once. Note that in a coverage tour, a required or a non-required edge may be used multiple times for deadheading. We define the following variables:

$$\begin{aligned} &s_e, s_{\bar{e}} \in \{0, 1\}, \text{ and } s_e + s_{\bar{e}} = 1 \quad \forall (e, \bar{e}) \in E_r \\ &d_e, d_{\bar{e}} \in \mathbb{N} \cup \{0\} \quad \forall (e, \bar{e}) \in E \end{aligned} \tag{1}$$

The variables s_e and $s_{\bar{e}}$ represent the two opposite directions of servicing the edge; exactly one of the two can be equal to 1 for a valid coverage tour. The variables d_e and $d_{\bar{e}}$ represent the number of times an edge is deadheaded in the corresponding direction. For a valid coverage tour, we can create an Eulerian digraph from these variables by adding the corresponding arc as many times as the value of the variable, and a closed diwalk can be obtained in $\mathcal{O}(|E|)$ time. The cost of a coverage tour τ, to be minimized, is given by:

$$c(\tau) = \sum_{(e,\bar{e}) \in E_r} \left[s_e s_c(e) + s_{\bar{e}} s_c(\bar{e}) \right] \quad + \quad \sum_{(e,\bar{e}) \in E} \left[d_e d_c(e) + d_{\bar{e}} d_c(\bar{e}) \right] \tag{2}$$

4 Graphs with All Required Edges

We first consider a simpler version of the problem where all the edges are required, i.e., $E = E_r$ and $E_n = \emptyset$, similar to the Windy Postman Problem (WPP). The single robot line coverage problem is, in fact, a generalization of the WPP since the WPP [18] assumes the service and deadhead costs are the same.

4.1 Integer Programming Formulation

We present an integer linear programming (ILP) formulation for the single robot line coverage problem on a connected graph $G(V, E, E_r)$, with given costs.

ILP1 (ILP Formulation 1)

$$\text{Minimize:} \sum_{(e,\bar{e})\in E_r} \left[s_e s_c(e) + s_{\bar{e}} s_c(\bar{e})\right] \quad + \sum_{(e,\bar{e})\in E} \left[d_e d_c(e) + d_{\bar{e}} d_c(\bar{e})\right] \tag{3}$$

subject to:

$$\sum_{a_1\in H(\mathcal{A}_r,v)} s_{a_1} + \sum_{b_1\in H(\mathcal{A},v)} d_{b_1} - \sum_{a_2\in T(\mathcal{A}_r,v)} s_{a_2} - \sum_{b_2\in T(\mathcal{A},v)} d_{b_2} = 0 \quad \forall v \in V \tag{4}$$

$$s_e + s_{\bar{e}} = 1 \quad \forall (e,\bar{e}) \in E_r \tag{5}$$

$$s_e, s_{\bar{e}} \in \{0,1\} \quad \forall (e,\bar{e}) \in E_r, \quad d_e, d_{\bar{e}} \in \mathbb{N} \cup \{0\} \quad \forall (e,\bar{e}) \in E \tag{6}$$

The *balance* (or symmetry) equations (4) state that for each vertex, the number of arc traversals into a vertex should be equal to the number of arc traversals out of the vertex. The *traversing* equations (5) ensure that each required edge is serviced exactly once and in only one direction.

4.2 An Alternative Formulation

We now present an alternative formulation that has fewer variables, and is closely related to the minimum cost flow problem. This formulation will be used to develop efficient algorithms for the single robot line coverage problem. First generate a digraph $D_m(V, A_m)$, using the algorithm MinCostDigraph, which selects the arc with the minimum service cost for each required edge.

We introduce a new variable r_a, for each arc $a \in A_m$, to represent reversal of service direction of the arc a. If an arc's service direction is reversed from a to $\bar{a}$, $r_a = 2$, the imbalance changes by 2, and the total cost changes by $s_c(\bar{a}) - s_c(a)$. We assign *reversal cost* $r_c(a)$ for each arc $a \in A_m$ and set $r_c(a)$ to $\frac{s_c(\bar{a}) - s_c(a)}{2}$.

F2 (Formulation 2)

$$\text{Minimize:} \quad s_c(A_m) \quad + \sum_{a\in A_m} r_a r_c(a) \quad + \sum_{(e,\bar{e})\in E} \left[d_e d_c(e) + d_{\bar{e}} d_c(\bar{e})\right] \tag{7}$$

subject to:

$$-\sum_{a_1\in H(A_m,v)} r_{a_1} + \sum_{b_1\in H(\mathcal{A},v)} d_{b_1} + \sum_{a_2\in T(A_m,v)} r_{a_2} - \sum_{b_2\in T(\mathcal{A},v)} d_{b_2} = \nabla(A_m, v) \quad \forall v \in V \tag{8}$$

$$r_a \in \{0,2\} \quad \forall a \in A_m \tag{9}$$

$$d_e, d_{\bar{e}} \in \mathbb{N} \cup \{0\} \quad \forall (e,\bar{e}) \in E \tag{10}$$

Algorithm 1: MinCostDigraph

Input : Graph $G(V, E, E_r)$, costs
Output : Minimum cost digraph $D_m(V, A_m)$

```
A_m ← ∅;
for (e, ē) ∈ E_r do
    if s_c(e) ≤ s_c(ē) then
        A_m.INSERT(e);
    else
        A_m.INSERT(ē);
end
```

Equation (7) is the modified objective function. The first term in the objective function, $s_c(A_m)$, is the sum of the service costs of all the arcs in the digraph D_m and is independent of the variables. The conditions (8) ensure that the digraph corresponding to a feasible solution will be balanced, i.e., the indegree will be equal to the outdegree at every vertex. The imbalance in the digraph D_m at a vertex v is represented by $\nabla(A_m, v)$. The constraint (9) states that the variable r_a can be 0, implying no reversal of service direction, or 2, for reversal of service direction. When $r_a = 0$, the corresponding ILP1 variable $s_a = 1$, and when $r_a = 2$, the variable $s_{\bar{a}} = 1$.

4.3 A Network Flow Graph Model

Arc routing problems are often solved by modelling them as network flow graphs and finding a minimum cost flow. Using this approach, algorithms for the Chinese Postman Problem and the Windy Postman Problem (WPP) were presented in [8,18]. We now present a network flow graph model for solving the linear relaxation of the formulation F2 and establish their equivalence.

Let $G(V, E, E_r)$ be the input graph. First generate a minimum cost digraph $D_m(V, A_m) = \text{MinCostDigraph}(G(V, E))$. Now construct a network flow graph $D_f(V, A_f)$, in $\mathcal{O}(|V| + |E|)$ time (Algorithm 2), as follows:

1. For each $a \in A_m$, add three arcs a, $\bar{a}$, and a' in A_f with the costs per unit flow $c_f(\cdot)$ and capacities as given in Table 1, which defines the Flow Model. The direction of arc a is same as that in A_m, whereas the direction of arcs $\bar{a}$ and a' are opposite to that of the corresponding arc in A_m.

Table 1. Flow Model (FM): arc costs and capacities for graphs with $E = E_r$.

Arc	Description	Unit flow cost $c_f(\cdot)$	Capacity
a	Forward deadheading	$d_c(a)$	∞
$\bar{a}$	Backward deadheading	$d_c(\bar{a})$	∞
a'	Service reversal	$r_c(a) = \big(s_c(\bar{a}) - s_c(a)\big)/2$	2

Algorithm 2: ConstructFlowDigraph

Input : $G(V, E, E_r)$, Digraph $D_m(V, A_m)$, flow model (FM)
Output : Flow Digraph $D_f(V, A_f)$

```
A_f ← ∅;
for a ∈ A_m do
    Insert arcs into A_f, with costs and capacities according to FM;
end
for v ∈ V do
    d(v) = ∇(A_m, v);
end
```

2. For each vertex $v \in V$, assign the following node flow demand:

$$d(v) = \text{outdeg}(v) \text{ in } D_m - \text{indeg}(v) \text{ in } D_m = \nabla(A_m, v) \tag{11}$$

3. Let f_a, $f_{\bar{a}}$, and $f_{a'}$ be the flows along the arcs a, $\bar{a}$, and a', respectively, in A_f, and let the *flow vector* be $\mathbf{f} = [f_{\hat{a}} \mid \hat{a} \in A_f]$. The cost $c(\mathbf{f})$ of a flow $\mathbf{f}$ is:

$$c(\mathbf{f}) = \sum_{\hat{a} \in A_f} c_f(\hat{a}) f_{\hat{a}} \tag{12}$$

Proposition 1. *Let τ be a coverage tour for a graph $G(V, E, E_r)$. Then there exists an equivalent network flow graph D_f with a corresponding flow $\mathbf{f}$, such that:*

1. $c(\tau) = s_c(A_m) + c(\mathbf{f})$, and
2. node flow demand $d(v)$ is satisfied for all $v \in V$.

Proof. The cost of the tour is given by:

$$c(\tau) = \sum_{(e,\bar{e}) \in E_r} \left[s_e s_c(e) + s_{\bar{e}} s_c(\bar{e}) \right] \quad + \quad \sum_{(e,\bar{e}) \in E} \left[d_e d_c(e) + d_{\bar{e}} d_c(\bar{e}) \right] \tag{13}$$

We construct a network flow $\mathbf{f}$ on the flow digraph D_f from the coverage tour τ and the minimum cost digraph D_m. For each arc a in A_m, do the following:

1. if $s_a = 1$ (implying $r_a = 0$), then set $f_{a'} = 0$, $f_a = d_a$, and $f_{\bar{a}} = d_{\bar{a}}$,
2. if $s_{\bar{a}} = 1$ (implying $r_a = 2$), then set $f_{a'} = 2$, $f_a = d_a$, and $f_{\bar{a}} = d_{\bar{a}}$.

Note that s_a and $s_{\bar{a}}$ cannot be both equal to 1 for a feasible coverage tour. For each arc a in A_m we compute its total cost in the flow graph D_f:

1. if $s_a = 1$:

$$\begin{aligned} \text{cost} &= s_c(a) + c_f(a) d_a + c_f(\bar{a}) d_{\bar{a}} \\ &= s_c(a) s_a + s_c(\bar{a}) s_{\bar{a}} + d_c(a) d_a + d_c(\bar{a}) d_{\bar{a}} \end{aligned} \tag{14}$$

2. if $s_{\bar{a}} = 1$:

$$\begin{aligned} \text{cost} &= s_c(a) + 2c_f(a') + c_f(a)d_a + c_f(\bar{a})d_{\bar{a}} \\ &= s_c(a) + s_c(\bar{a}) - s_c(a) + c_f(a)d_a + c_f(\bar{a})d_{\bar{a}} \\ &= s_c(a)s_a + s_c(\bar{a})s_{\bar{a}} + d_c(a)d_a + d_c(\bar{a})d_{\bar{a}} \end{aligned} \tag{15}$$

Since we have an arc in A_m for each edge in E, from Eqs. 13, 14, and 15 we have the first result of the proposition.

As the coverage tour is a closed diwalk, the number of arcs leaving and entering a vertex are equal, i.e, $\text{indeg}(i) = \text{outdeg}(i),\ \forall i \in V$. Thus the flow demands of the vertices in the flow digraph will be satisfied. □

Definition 2. Minimum Cost Flow Problem
Let $D_f(V, A_f)$ be a given digraph, along with costs, capacities, and node flow demands. Then the minimum cost flow problem is to find a feasible flow $\mathbf{f}$ *such that:*

1. *the cost of flow $c(\mathbf{f})$ is minimized, and*
2. *demand $d(v)$ is satisfied for all $v \in V$.*

We next present algorithms for the single robot line coverage problem for the special case when all edges in the graph are required, i.e., $E = E_r$. We first present an optimal algorithm for Eulerian graphs and then extend the algorithm to general graphs. The algorithms are based on solving the linear relaxation using the minimum cost flow problem, as often used in arc routing problems [2,18].

4.4 Eulerian Graphs with All Required Edges

We now consider the case of connected Eulerian graphs with all the edges as required edges, i.e., $E = E_r$.

Theorem 3. *Let $G(V, E)$ be an Eulerian graph for the line coverage problem, with minimum cost digraph $D_m(V, A_m)$, and flow digraph $D_f(V, A_f)$.*

1. *The network flow graph D_f models the linear relaxation of the formulation F2 with the following relation:*

$$f_a = d_a,\ \ f_{a'} = r_a,\ \ f_{\bar{a}} = d_{\bar{a}} \quad \forall a \in A_m \tag{16}$$

2. *The optimal solution for the minimum cost flow gives an optimal and feasible solution for the formulation F2. In particular $f_{a'} = r_a \in \{0, 2\}$, and $f_a, f_{\bar{a}} \in \mathbb{N} \cup \{0\}$.*

Proof. Consider the linear relaxation of the formulation F2. The variables are set to be continuous, and the conditions (9) and (10) are relaxed.

$$\begin{aligned} &r_a \leq 2, && \forall a \in A_m \\ &r_a, d_a, d_{\bar{a}} \geq 0, && \forall a \in A_m \end{aligned} \tag{17}$$

Algorithm 3: LineCoverageEulerian

Input : Eulerian graph $G(V, E)$, costs
Output : Optimal coverage tour τ^*

$D_m(V, A_m) =$ MinCostDigraph$(G(V, E))$;
$D_f(V, A_f) =$ ConstructFlowDigraph$(G, D_m$, Flow Model in Table 1);
Compute the minimum cost flow **f** for D_f;
/* Generate Eulerian digraph $D_e(V, A_e)$ */
$A_e \leftarrow A_m$; /* initial service arcs */
for $a \in A_m$ **do**
  if $f_{a'} = 2$ **then**
    Reverse service direction of a to $\bar{a}$;
  A_e.INSERT(f_a copies of a);
  A_e.INSERT($f_{\bar{a}}$ copies of $\bar{a}$);
end
Generate Eulerian diwalk τ^* from D_e;

The network flow model, with the specified relation of the flow variables to the formulation F2 in Eq. (16), is the exact model of the linear relaxation.

The node flow demand for a vertex $v \in V$ is even as the graph D is Eulerian. The capacities defined in Table 1 are either 2 or ∞. Hence the optimal flow is also even for each arc $a_f \in A_f$. As the minimum cost flow algorithm gives integral solutions [14], $f_{a'} = r_a \in \{0, 2\}$. This result can also be shown using total unimodularity of the constraint matrix from the balance equations (8). □

Theorem 4. *Let $G(V, E)$ be an Eulerian graph with service and deadhead costs.* LineCoverageEulerian *given in Algorithm 3 generates an optimal coverage tour τ^* for G, and has time complexity $\mathcal{O}\big((|E| \log |V|)(|E| + |V| \log |V|)\big)$.*

Proof. By Theorem 3, the algorithm gives an optimal solution to the formulation F2. It can be shown that the graph D_e in the algorithm is Eulerian, and an optimal Eulerian diwalk can be generated in $\mathcal{O}(|E|)$ computational time. Hence, the algorithm gives the optimal coverage tour.

MinCostDigraph has complexity $\mathcal{O}(|E_r|)$, the flow graph can be computed in $\mathcal{O}(|V|+|E|)$, and the Eulerian digraph and diwalk can be computed in $\mathcal{O}(|E|)$ time. The computation time of minimum cost flow [13] is $\mathcal{O}\big((|E| \log |V|)(|E| + |V| \log |V|)\big)$, leading to the overall complexity of the algorithm. □

4.5 A 2-Approximation Algorithm for General Graphs with All Required Edges

For a general graph $G(V, E)$, i.e., not necessarily Eulerian, the created minimum cost digraph $D_m(V, A_m)$ may have vertices with odd flow demands (11). As a result, the optimal flow values need not be even. While it is not a problem if d_a or $d_{\bar{a}}$ is odd, we need to assign service directions to the edges for which r_a is 1.

Algorithm 4: LineCoverage-Approx

Input : Graph $G(V, E)$, costs
Output : A coverage tour τ

$D_m(V, A_m) =$ MinCostDigraph$(G(V, E))$;
$D_f(V, A_f) =$ ConstructFlowDigraph$(G, D_m$, Flow Model in Table 1);
Compute the minimum cost flow $\mathbf{f}$ for D_f;
/* Generate Eulerian digraph $D_e(V, A_e)$ */
$A_e \leftarrow A_m$; /* initial service arcs */
for $a \in A_m$ **do**
  if $f_{a'} = 2$ **then**
    Reverse service direction of a to $\bar{a}$;
  else if $f_{a'} = 1$ **then**
    if $s_c(a) + d_c(\bar{a}) \leq s_c(\bar{a}) + d_c(a)$ **then**
      A_e.INSERT$(\bar{a})$;
    else
      Reverse service direction of a to $\bar{a}$;
      A_e.INSERT(a);
  A_e.INSERT(f_a copies of a);
  A_e.INSERT($f_{\bar{a}}$ copies of $\bar{a}$);
end
Generate Eulerian diwalk τ from D_e;

Let the optimal flow be $\mathbf{f}$, and the cost of the optimal tour be c^*. Then the optimal value of the linear relaxation z^* of the formulation F2 is:

$$z^* = s_c(A_m) + c(\mathbf{f}) \leq c^* \tag{18}$$

Let A_u be the set of arcs for which the flow for the arc corresponding to the reversal of service direction is 1, i.e., $A_u = \{a \mid r_a = f_{a'} = 1, a \in A_m\}$. Let D denote the set of arcs corresponding to the service direction and deadheading decided unambiguously by the optimal flow. Then,

$$c(D) = s_c(A_m \setminus A_u) + c(\mathbf{f}) - r_c(A_u) \tag{19}$$

$$\begin{aligned} \text{Thus,} \quad z^* &= s_c(A_m \setminus A_u) + s_c(A_u) + c(\mathbf{f}) - r_c(A_u) + r_c(A_u) \\ &= c(D) + s_c(A_u) + r_c(A_u) \end{aligned} \tag{20}$$

Theorem 5. *Given a connected graph $G(V, E, E_r)$ with $E_r = E$, and costs,* LineCoverage-Approx*, given in Algorithm 4, generates a coverage tour with cost at most twice the cost of the optimal coverage tour in polynomial time.*

Proof. Let A_s be the set of arcs corresponding to A_u with final service directions oriented by the algorithm. The total cost of a solution τ generated by the algorithm is:

$$c(\tau) = c(D) + s_c(A_s) + d_c(\bar{A}_s) \tag{21}$$

Substituting the value of $r_c(A_u) = \frac{s_c(\bar{A}_u) - s_c(A_u)}{2}$ in (20) and using (18) we have,

$$2c(D) + s_c(A_u) + s_c(\bar{A}_u) \leq 2c^*$$
$$\text{or, } c(D) + s_c(A_u) + s_c(\bar{A}_u) \leq 2c^* - c(D)$$

Note that $s_c(A_s) + d_c(\bar{A}_s) \leq s_c(\bar{A}_u) + d_c(A_u)$ because we selected the service and deadheading directions to minimize the sum of the costs for individual arcs in A_u. Furthermore, $d_c(a) \leq s_c(a)$ for $a \in A_m$. Hence,

$$\begin{aligned} c(\tau) &\leq c(D) + s_c(\bar{A}_u) + d_c(A_u) \\ &\leq c(D) + s_c(\bar{A}_u) + s_c(A_u) \leq 2c^* - c(D) \leq 2c^* \end{aligned}$$

The complexity of the algorithm is determined by the minimum cost flow problem, which can be solved in $\mathcal{O}\big((|E| \log |V|)(|E| + |V| \log |V|)\big)$ time [13]. □

5 Rural Graphs

We now consider graphs where all the edges need not be required edges; we call these *rural graphs* since they arise in the Rural Postman Problem (RPP). This version of the single robot line coverage problem corresponds to the RPP on asymmetric graphs with distinct service and deadhead costs. We first extend the algorithms in the previous section to the case where the *required graph*, i.e., the graph induced by required edges, is connected.

5.1 Rural Graphs with a Single Connected Component

Let $G(V, E, E_r)$ be a graph such that $E_r \subseteq E$ and the required subgraph $G_r(V_r, E_r)$ induced by the required edges E_r consists of a single connected component. Similar to the algorithm for graphs with all required edges, we present a network flow model that is a linear relaxation of the corresponding formulation. First generate a minimum cost digraph $D_m(V, A_m)$. Now construct a network flow graph $D_f(V, A_f)$ using the algorithm ConstructFlowDigraph modified as follows: For each $a \in A_m$ add three arcs a, $\bar{a}$, and a' in A_f, and for each edge $(e, \bar{e}) \in E_n$ where $E_n = E \setminus E_r$, add two arcs b and $\bar{b}$, with the costs per unit flow and capacities in Table 2. LineCoverageConnected (Algorithm 5) gives a coverage tour for G with an approximation factor of 2, the proof for which is similar to that given in Theorem 5.

5.2 Rural Graphs with Multiple Connected Components

We now consider rural graphs for which the subgraph G_r induced by the required edges may have multiple connected components. For such graphs, Algorithm 5 may output a digraph that is disconnected even though the individual connected components are Eulerian. We develop an $\alpha(C) + \beta$ approximation algorithm where C is the number of connected components in G_r and β is the approximation factor for the single robot line coverage problem on rural graphs with a single

Table 2. Arc costs and capacities for network flow model for rural graphs.

Arc	Description	Unit flow cost $c_f(\cdot)$	Capacity
a	Forward deadheading	$d_c(a)$	∞
$\bar{a}$	Reverse deadheading	$d_c(\bar{a})$	∞
a'	Service reversal	$r_c(a)$	2
b	Non-required forward deadheading	$d_c(b)$	∞
$\bar{b}$	Non-required reverse deadheading	$d_c(\bar{b})$	∞

connected component. The α approximation factor depends on the approximation algorithm for the asymmetric traveling salesperson problem (ATSP) [16,17], and a β of 2 was discussed in the previous subsection.

The output digraph D_e of Algorithm 5 is processed to find strongly connected components. We then create an auxiliary graph $G_0(V_0, E_0)$ with V_0 consisting of one arbitrary vertex from each of the connected components in D_e. G_0 is a complete graph with E_0 consisting of edges between all pairs of vertices in V_0. Each edge in E_0 has two weights corresponding to the shortest deadhead cost paths in the two directions. We can use constant-factor approximation algorithms for the ATSP to find a tour connecting all the vertices in G_0; see Sect. 2. The arcs in the ATSP tour are then added to the disconnected diwalk generated from algorithm LineCoverageConnected to obtain a connected coverage tour.

Theorem 6. *The single robot line coverage problem can be solved in polynomial time with an approximation factor of $\alpha(C)+\beta$, where $\alpha(C)$ is the approximation factor for an algorithm for the asymmetric traveling salesperson problem with C vertices, and β is the approximation factor for line coverage on rural graphs with a single connected component.*

Proof. Let the optimal coverage tour be τ^*, and $\tilde{\tau}$ be the output of Algorithm LineCoverageConnected with a corresponding digraph $\tilde{D}_e$. Note that $\tilde{\tau}$ need not be a feasible coverage tour as $\tilde{D}_e$ may contain multiple strongly connected components. However, the number of strongly connected components in the digraph $\tilde{D}_e$ will be no more than the number of connected components C in the required graph G_r. As we are not considering connectivity constraints between the components in $\tilde{D}_e$, the solution is an approximation result to a relaxation of the original problem. Hence, $c(\tilde{\tau}) \leq \beta c(\tau^*)$.

Let V_0 be a set of vertices such that no two vertices belong to the same strongly connected component in $\tilde{D}_e$. Then $|V_0| \leq C$. The tour τ^* will visit each of the vertices in V_0 because each vertex in V_0 is part of a connected component. For the graph G_0, let T^* be the optimal ATSP tour and T be the ATSP tour returned by the $\alpha(C)$-approximation algorithm. Then $c(T) \leq \alpha(C)c(T^*) \leq \alpha(C)c(\tau^*)$.

Let τ be the coverage tour obtained by adding to $\tilde{\tau}$ the arcs from T. Then $c(\tau) = c(\tilde{\tau}) + c(T) \leq \beta c(\tau^*) + \alpha(C)c(\tau^*) = (\alpha(C) + \beta)c(\tau^*)$. □

Algorithm 5: LineCoverageConnected

Input : Rural graph $G(V, E, E_r)$ with connected G_r, costs
Output : A coverage tour τ, Eulerian digraph $D_e(V, A_e)$

$D_m(V, A_m) =$ MinCostDigraph$(G_r(V, E_r))$;
$D_f(V, A_f) =$ ConstructFlowDigraph$(G, D_m$, Flow Model in Table 2);
Compute the minimum cost flow **f** for D_f;
`/* Generate Eulerian digraph` $D_e(V, A_e)$ `*/`
$A_e \leftarrow A_m$; `/* initial service arcs */`
for $a \in A_m$ **do**
  if $f_{a'} = 2$ **then**
    Reverse service direction of a to $\bar{a}$;
  else if $f_{a'} = 1$ **then**
    if $s_c(a) + d_c(\bar{a}) \leq s_c(\bar{a}) + d_c(a)$ **then**
      A_e.INSERT$(\bar{a})$;
    else
      Reverse service direction of a to $\bar{a}$;
      A_e.INSERT(a);
  A_e.INSERT$(f_a$ copies of $a)$;
  A_e.INSERT$(f_{\bar{a}}$ copies of $\bar{a})$;
end
for $(b, \bar{b}) \in E_n$ **do**
  A_e.INSERT$(f_b$ copies of $b)$;
  A_e.INSERT$(f_{\bar{b}}$ copies of $\bar{b})$;
end
Generate Eulerian diwalk τ from D_e;

Corollary 7. *Combining Theorems 5 and 6, and noting that the number of connected components C is usually small in practice [2], we observe:*

1. *The single robot line coverage problem has an $\alpha(C)+2$ approximation factor. This also improves the previously best known approximation result of $\alpha(C)+3$ for the asymmetric rural postman problem [2].*
2. *If $C \in \mathcal{O}(\log n)$, a $\mathcal{O}(C^2 2^C)$ dynamic programming algorithm gives the optimal ATSP solution in polynomial time, giving a 3-approximation algorithm.*

6 Conclusion

We considered the line coverage problem for a single robot where the service and deadhead costs are considered distinct, and are asymmetric. We presented approximation algorithms to obtain bounded solutions efficiently, using the minimum cost flow problem. We built the main algorithm in stages by considering three simpler subproblems, based on the structure of the *required graph*, i.e., the graph induced by the features that require servicing. We first presented an optimal algorithm for the case of Eulerian graphs with only *required* edges. Next we considered the case of graphs that are not necessarily Eulerian with only

required edges and presented a 2-approximation algorithm. Finally, we considered the general case of *rural graphs* with both required and non-required edges. The approximation algorithm here is dependent on the asymmetric traveling salesperson problem (ATSP), and is bounded by $\alpha(C) + 2$, where $\alpha(C)$ is the approximation factor of the ATSP algorithm with C connected components. This upper bound in also an improvement over the existing results for the asymmetric rural postman problem.

Future work includes improving the approximation factor of 2 for rural graphs. Generalizing the analysis to consider robot capacities and edge demands in order to develop approximation algorithms for the line coverage problem with multiple robots is another important direction.

Acknowledgments. This work was supported in part by NSF awards IIS-1547175, IIP-1439695 and IIP-1919233, a UNC IPG award, and DARPA HR00111820055.

References

1. Agarwal, S., Akella, S.: Line coverage with multiple robots. In: IEEE International Conference on Robotics and Automation, Paris, France (2020)
2. van Bevern, R., Komusiewicz, C., Sorge, M.: A parameterized approximation algorithm for the mixed and windy capacitated arc routing problem: theory and experiments. Networks **70**(3), 262–278 (2017)
3. Choset, H.: Coverage for robotics - a survey of recent results. Ann. Math. Artif. Intell. **31**(1), 113–126 (2001)
4. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, 388, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA (1976)
5. Corberan, A., Laporte, G. (eds.): Arc Routing: Problems, Methods, and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia (2014)
6. Dille, M., Singh, S.: Efficient aerial coverage search in road networks. In: AIAA Guidance, Navigation, and Control Conference, Boston, MA, USA, pp. 5048–5067 (2013)
7. Easton, K., Burdick, J.: A coverage algorithm for multi-robot boundary inspection. In: IEEE International Conference on Robotics and Automation, Barcelona, Spain, pp. 727–734 (2005)
8. Edmonds, J., Johnson, E.L.: Matching, Euler tours and the Chinese postman. Math. Program. **5**(1), 88–124 (1973)
9. Frederickson, G.N.: Approximation algorithms for some postman problems. J. ACM **26**(3), 538–554 (1979)
10. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robot. Auton. Syst. **61**(12), 1258–1276 (2013)
11. Macharet, D.G., Campos, M.F.M.: A survey on routing problems and robotic systems. Robotica **36**(12), 1781–1803 (2018)
12. Oh, H., Kim, S., Tsourdos, A., White, B.: Coordinated road-network search route planning by a team of UAVs. Int. J. Syst. Sci. **45**(5), 825–840 (2014)
13. Orlin, J.B.: A faster strongly polynomial minimum cost flow algorithm. Oper. Res. **41**(2), 338–350 (1993)

14. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall Inc., Upper Saddle River (1982)
15. Raghavachari, B., Veerasamy, J.: Approximation algorithms for the asymmetric postman problem. In: Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1999, USA, pp. 734–741 (1999)
16. Svensson, O., Tarnawski, J., Végh, L.A.: A constant-factor approximation algorithm for the asymmetric traveling salesman problem. In: 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, pp. 204–213 (2018)
17. Traub, V., Vygen, J.: An improved approximation algorithm for ATSP. In: 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, pp. 1–13 (2020)
18. Win, Z.: On the windy postman problem on Eulerian graphs. Math. Program. **44**(1), 97–112 (1989)
19. Xu, L., Stentz, A.: An efficient algorithm for environmental coverage with multiple robots. In: IEEE International Conference on Robotics and Automation, Shanghai, China, pp. 4950–4955 (2011)