View Planning Problem with Traveling Costs: Problem Formulation, Hardness of Approximation, and Approximation Algorithms

Pengpeng Wang [*] , Ramesh Krishnamurti [†], and Kamal Gupta [*]

Abstract

In this paper, we introduce the problem of view planning with traveling cost, or *Traveling VPP*. It refers to planning a sequence of sensing actions with minimum total cost by a robot-sensor system to completely inspect the surfaces of objects with known geometries in a known workspace. The cost to minimize is a combination of the view cost, proportional to the number of viewpoints planned, and the traveling cost for the robot to realize them. First, we formulate Traveling VPP as a combinatorial optimization problem and show it is equivalent to "set covering on a graph". Then, we use the hardness of approximation result for the group Steiner tree problem (GST) to show that Traveling VPP is log-square inapproximable. Also, we show, via a reduction to the group Steiner tree problem, that the existing approximation algorithm for GST applies to Traveling VPP, thereby providing a poly-log approximation ratio. Finally, we give a linear program based rounding algorithm that achieves an approximation ratio of the order of view frequency, defined to be the largest number of viewpoints that see a single surface patch of the object. We conclude with a discussion of realistic issues and constraints if our algorithm were implemented on real robot-sensor systems.

Keywords: view planning with traveling cost; hardness of approximation; integer linear program; linear program; approximation algorithm

1 Introduction

In applications ranging from surveillance to object inspection, an autonomous robot is required to inspect the surfaces of objects or boundaries of the workspace in a large and/or cluttered environment. Every surface of the objects of interest (which could be the whole environment) must be viewed/covered via at least one planned viewpoint of the range sensor. It is desired that the total

[*]School of Engineering Science, Simon Fraser University, Burnaby, B.C., Canada. Email: {pwangf, kamal}@cs.sfu.ca

[†]School of Computing Science, Simon Fraser University, Burnaby, BC, Canada. Email: ramesh@cs.sfu.ca

cost of the plan, consisting of the traveling cost (the total distance traveled by the robot along the planned path, often a measure of the total amount of energy consumed by the robot) and the view cost (proportional to the number of viewpoints planned where each viewing overhead is due to image acquisition, processing, and registration), is minimized, often a requirement for autonomous robotic missions where battery life is a significant issue [1]. We call this problem *Traveling View Planning Problem*, or *Traveling VPP* in short, and formulate it as an optimization problem. We assume the object and environment are of known geometries and call it the "model-based" case.

See Fig. 1 for a simple Traveling VPP example where the robot-sensor system, a mobile manipulator with a range sensor mounted at the manipulator end-effector, is required to inspect the surface of a large object. Compared with just the mobile base, the mobile manipulator gives the sensor additional degrees of freedoms and maneuverabilities. This is illustrated in Fig. 1, where the robot achieves visibility by extending the manipulator over occluding obstacles. The six robot configurations that realize the planned viewpoints are also shown, and the dotted lines between these configurations is the traveling path of the robot. The dotted triangles that are attached to the robot end-effector are the sensor's field of view (FOV) at different configurations. The total cost of such a plan includes the total view cost, proportional to the number of viewpoints planned (six in this case), and the traveling cost, proportional to the length of the path traveled by the robot.

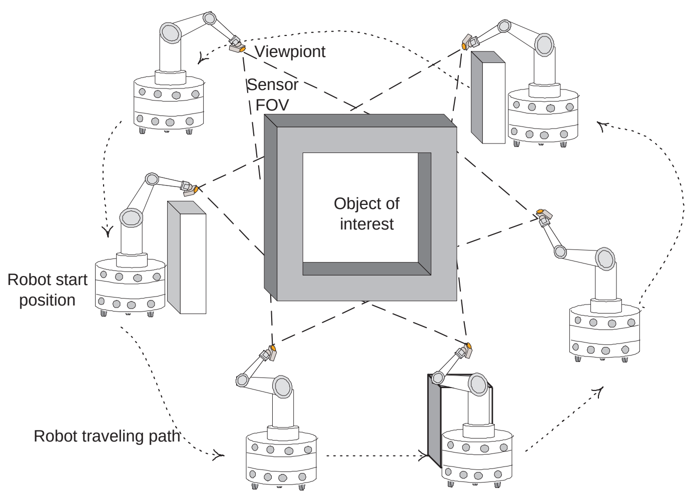

Figure 1: A Traveling VPP instance. It shows 6 planned sensor viewpoints that totally cover the surface of the object of interest, and the robot traveling path to realize them.

Traveling VPP combines elements of both the view planning problem (VPP) and the watchman route problem, and thus generalizes both. VPP refers to planning the minimum number of viewpoints to completely inspect an object surface. It is considered in the robot vision area [2], where often a sensor positioning system is used within a well controlled and limited workspace. These formulations do not consider the traveling cost of the robot, a critical cost, particularly for large workspaces and remote autonomous missions, where power consumption is a critical factor. On the other hand, the watchman route problem, considered in the computational geometry area [3, 4], asks for the shortest tour to inspect the interior of a two-dimensional polygonal region. It does not consider view cost, a critical cost, particularly for the inspection tasks considered here, where each sensor view and the consequent processing are time-consuming. Also, unlike the watchman route problem being restricted to a two-dimensional environment, our *Traveling VPP* formulation uses a graph to encode traveling. Such graphs, called roadmaps, in the root configuration space are commonly used for the path planning problem for high degree-of-freedom robots in the robot motion planning literature [5, 6]. In addition, since we do not assume metrics for the graph, Traveling VPP is applicable to more general cases.

There is some existing work on combining the view and traveling cost in the literature, but not in a unified and global fashion. For example, in [7, 8], the authors considered a local version of the robot exploration problem, "to look around a corner", i.e., to detect an object hidden behind a corner while minimizing the sum of the robot traveling distance and the sensor scan time. The problem is considerably simpler since the goal is local, i.e., the objective is not to cover all the object surfaces.

In [9], the authors considered the combined problem, however, in a "weak sense", since no view cost is considered, thus corresponding to a special case of *Traveling VPP*. They proposed to solve the problem by a decoupled two-level approach, i.e., to plan the minimum number of viewpoints without considering robot traveling cost first and then to solve (approximately) the Metric TSP using the shortest path graph. This two-level decoupled approach would work well for cases where the views considered do not have overlap between their coverage (they become the only choice for a view plan.), or those with large coverage overlap are close to each other (they correspond to similar traveling costs). However, this is not true for a general Traveling VPP setting. For example, as shown in Fig. 2, even assuming that at each level the respective optimization subproblem, obtaining the minimum number of viewpoints and the shortest path tour respectively, is solved optimally, this two-level decoupled approach provides no performance bound (with respect to the optimal solution cost), and can perform arbitrarily poorly. This is easily achieved by pulling the leftmost viewpoint arbitrarily farther from the rightmost ones. This issue occurs because the planned viewpoints at the first level are too far apart for the robot to realize a plan efficiently since no traveling cost is considered at the first stage.

The fact that *Traveling VPP* cannot be satisfactorily solved by decoupled approaches motivated the approach we take in this paper, i.e., to give a unified formulation of the problem with a single objective function that combines both sensing and traveling costs, and to design fast algorithms with guaranteed

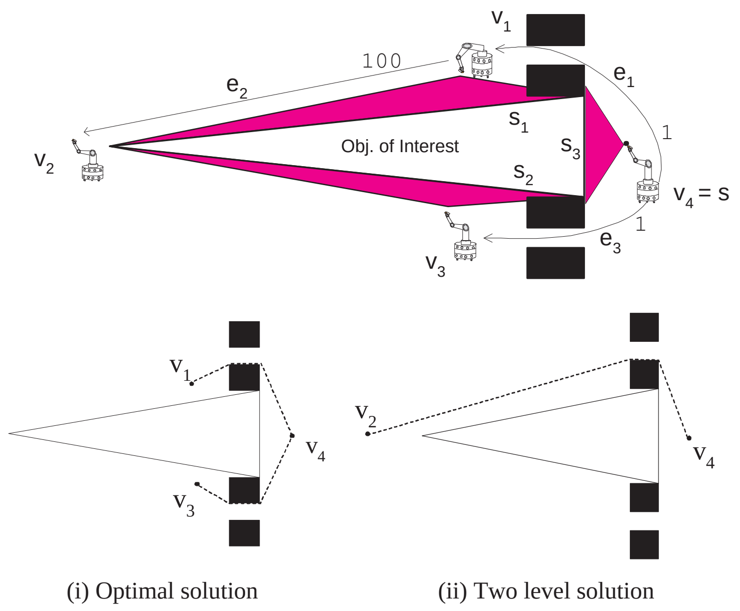

Figure 2: A planar example shows the arbitrarily poor performance of the two-level decoupled approach. The object to inspect, the triangle, has three surface patches, s_1, s_2 and s_3; the four possible sensing positions, or viewpoints, are v_1, v_2, v_3 and v_4, all shown in the top figure. v_4 coincides with the robot start position s. The shaded sensing triangles show the covering relations: viewpoints v_1, v_3 and v_4 cover the surface patches s_1, s_2 and s_3 respectively, while v_2 (sensing triangle of which is not shown) covers both s_1 and s_2. The line segments connecting the views, e_1, e_2 and e_3, denote the robot's traveling path; the numbers on each segment are the respective traveling costs (distances). (The distances are not drawn to scale.) We assume the view and traveling cost are equally weighted in the objective function, i.e. the unit viewing cost (cost for each view) is the same as the unit traveling cost (cost for unit traveling distance). Thus the total cost is the sum of the number viewpoints planned and the total distance of the path planned. The dashed lines shown in the two bottom figures are the planned paths connecting the planned viewpoints. The optimal solution is to take three views at s, v_1 and v_2 using the dashed line segments as the traveling path. The solution given by the decoupled cost can be made arbitrarily poor by pulling v_2 farther to the left.

performances. We show that this unified formulation is equivalent to "set covering on a graph", clearly an NP-complete problem. Approximation algorithms that are fast and provide guarantees on their worst case performance are usually sought for these intractable problems. The worst-case performance of an approximation algorithm is measured by the "approximation ratio", the largest ratio between the cost of the solution provided by the algorithm and the optimal value. See [10] for a review on approximation algorithms for NP-hard problems. The main emphasis of this paper is to design a good approximation algorithm for the traveling VPP.

Secondly, using the hardness of approximation result for the group Steiner tree problem (GST) [20], we show that Traveling VPP is log-square inapproximable. This establishes a lower bound on the best approximation ratio for any approximation algorithm for Traveling VPP.

We then use the approach based on linear program (LP) relaxation for our approximation algorithm. We give a rounding algorithm that takes an optimal relaxed LP solution and outputs an integral Traveling VPP solution. A key result of this paper is that the algorithm has an approximation ratio that is twice the "view frequency", where the view frequency is defined as the maximum number of viewpoints that covers a single surface patch. We then show that this approximation ratio is the same as the *integrality gap* for Traveling VPP. For an ILP formulation, the *integrality gap* is defined as the largest ratio between the optimal values to ILP and to its relaxed LP. In the sense that the integrality gap is widely believed to be a lower bound on the approximation ratio for algorithms based on LP relaxation, the rounding algorithm proposed has the best possible approximation ratio for the ILP formulation. Also, we show, via a reduction from Traveling VPP to GST, that the existing poly-log approximation algorithm for GST [11, 12] is applicable to Traveling VPP and achieves a poly-log approximation ratio. This result parallels that for the set covering problem (SCP) for which the best approximation ratio is either the frequency or a logarithm of the input size.

Furthermore, we give necessary reductions to show that our LP-based rounding algorithm applies to related combinatorial optimization problems such as the GST and the Traveling Purchaser Problem (TPP) [13] and achieves a similar approximation ratio. And finally, we discuss some realistic issues and constraints towards implementing the algorithms on a real system. These include how to generate viewpoints and the traveling graph, and how to accommodate additional constraints.

The rest of the paper is organized as follows. We first give our formulation for Traveling VPP, followed by related work on some combinatorial optimization problems (We delay the discussions on these related work since their relationship to Traveling VPP will become clearer after our formulation is given, the application related issues are abstracted out, and the combinatorial nature of the problem is clear.) We show the hardness of approximation for Traveling VPP; we then give our LP-based rounding algorithm and analyze its performance; finally, we discuss how to incorporate realistic issues towards implementations.

2 Problem Formulation

2.1 Traveling VPP: abstraction to set covering on a graph

In this section, we formulate Traveling VPP as a combinatorial optimization problem combining two well-known problems, namely the set covering problem (SCP) and the traveling salesman problem (TSP).

By considering each object surface patch as an element in a subset that a viewpoint can cover [14], an arbitrary VPP instance is immediately an SCP instance. Since the VPP has an inherent geometric structure, one might hope that VPP is a simpler version of SCP. In the appendix, we show a simple reduction that takes any SCP instance and constructs a VPP instance[1], thus showing the equivalence of the SCP and VPP.

On the other hand, assuming the planned viewpoints are given (or the VPP as a sub-problem is already solved), Traveling VPP is reduced to the Metric Traveling Salesman Problem (Metric TSP) by constructing the shortest path graph on the planned viewpoints [9]. The shortest path graph is defined as the complete graph between the planned viewpoints in which the edge cost between two viewpoints is the length of the shortest path the robot needs to travel to realize them. Since the two problems, SCP (without metrics) and Metric TSP (with metrics), have fundamentally different structures and solving techniques, Traveling VPP is an interesting and non-trivial generalization.

Thus, in the abstract sense, Traveling VPP can also be termed as "Set Covering on a Graph".

2.2 Related work on combinatorial optimization problems

In addition to the related work in the robotics and vision literature mentioned earlier, here we mention some related work in the combinatorial optimization literature, especially the problems of connected facility location and errand scheduling. Two other problems, the group Steiner tree and the traveling purchaser problem, are also related and will be discussed in later sections.

In [15], the problem of connected facility locations is addressed, which, given a set of *facilities* and a set of *clients* both residing in a metric space, asks for a set of *open* facilities connected by a Steiner tree and the *service* assignments between these open facilities and the clients, such that the total cost, which includes both the sum of the service assignment costs and the tree cost, is minimized. The authors give a greedy algorithm with a constant approximation ratio for the metric case. By regarding the clients as the surface patches in the Traveling VPP, and the facilities as the viewpoints, the connected facility location problem is related to the Traveling VPP. However, the visibility relation between viewpoints and surface patches does not assume a metric, and the heuristic used in [15] is not applicable to the Traveling VPP.

[1] In [14], the authors claimed the VPP is isomorphic to the SCP but did not give a concrete reduction from an arbitrary SCP instance to a VPP instance.

In the errand scheduling problem (ESP), a graph with metric costs (the edge costs satisfy the triangular inequality) is given, where each vertex is associated with a subset of errands. A shortest tour is required such that all errands are accomplished [16]. In [16], the author gives an algorithm with an approximation ratio of $3\rho/2$, where ρ is the maximum number of nodes an errand is associated. Traveling VPP is a much more general problem than ESP. First, the graph in Traveling VPP does not assume metrics. Furthermore, in Traveling VPP, even if some viewpoints are in the final solution, the robot may not need to take a view at these points i.e., the robot simply travels through these points (also termed as Steiner nodes [10]). There is no view cost in ESP; and there is no notion of Steiner nodes. This implies that there is no simple reduction from Traveling VPP to ESP.

2.3 Integer program formulation of Traveling VPP

We now give an integer linear program (ILP) formulation for the Traveling VPP.

In our unified formulation of traveling VPP, we make assumptions to abstract out and focus on certain key ingredients, in particular the interplay between SCP and Metric TSP. We assume that the surface patches to be inspected are given, the viewpoints from which a surface can be inspected are also given, and that a graph which encodes the possible robot movements connecting the viewpoints and the robot start position, is also given. We assume binary covering relationship, i.e., a viewpoint either covers a surface patch or does not cover it. These assumptions are based on a realistic scenario and algorithms from the literature can be used to derive these quantities. We discuss these realistic issues with an eye towards implementation in Section 6.

Our formulation of Traveling VPP chooses a subset of the viewpoints and a (Steiner) tree on the graph to connect them, under the (covering) constraint that every surface patch is covered by at least one planned viewpoint and the (connection) constraint that every planned viewpoint is connected to the robot start position via the planned (Steiner) tree. The objective is to minimize the total cost of the plan, defined as the sum of the view costs and the tree cost (the sum of all edge costs in the Steiner tree). The reason for using a (rooted) Steiner tree instead of a tour is that we are able to combine the (rooted) Steiner tree formulation with covering constraints. It is not immediately clear how to combine a tour (or path) constraints (especially the subtour elimination constraints [17]) with the covering constraints. Moreover, Metric TSP can be easily approximated using the solution to the Steiner tree problem [10].

We denote the set of all viewpoints by $\mathcal{V}$ and index them by i. We denote the set of surface patches by $\mathcal{S}$ and index them by j. We use the notation $i \in \mathcal{V}$ and $j \in \mathcal{S}$ to imply the "ith viewpoint" and the "jth surface patch", respectively. For $i \in \mathcal{V}$, let $\mathcal{S}(i)$ denote the subset of the surface patches that viewpoint i covers; and for $j \in \mathcal{S}$, let $\mathcal{V}(j)$ denote the subset of viewpoints that cover surface patch j. The robot movements are restricted to the graph $G = (\mathcal{V}, E)$, where the node set $\mathcal{V}$ is the set of all viewpoints and s, the starting position of the robot. In case the robot start position does not correspond to a viewpoint, we simply assign the empty set as the set of surface patches it sees. The edge e between two views v_{i_1} and v_{i_2} represents the path from v_{i_1} to v_{i_2}. We use c_e to denote the cost (length) of edge e. We also use $T \subset \mathcal{V} : s \notin T$ to denote a cut or subset of the graph that does not include the robot start position. We use $\delta(T)$ to denote the set of edges that "crosses" T, having one end inside T and the other outside T, i.e., $e = < v_1, v_2 > \in \delta(T) \iff v_1 \in T \wedge v_2 \notin T$ OR $v_2 \in T \wedge v_1 \notin T$. We use w_v, the unit view cost or cost per viewpoint, and w_p, the unit path cost or cost per unit traveling distance. Furthermore, we use F to denote the view frequency, defined as the maximum number of viewpoints that cover a single surface patch, i.e., $F = \max_{j \in \mathcal{S}} |\mathcal{V}(j)|$, where $|\mathcal{A}|$ denotes the cardinality of a discrete set $\mathcal{A}$.

We define a binary variable, y_i, as the indicator whether to take a view at viewpoint i, corresponding to $y_i = 1$, or not, corresponding to $y_i = 0$; we define the binary variable, z_e, as the indicator whether to include the edge e in the robot traveling path, corresponding to $z_e = 1$, or not, corresponding to $z_e = 0$. Thus, the ILP formulation for the traveling VPP is given as:

Traveling VPP (ILP): (1)

$$\min \quad w_v \sum_{i \in \mathcal{V}} y_i + w_p \sum_{e \in E} c_e z_e$$

$$\text{Subject to:} \quad \forall j \in \mathcal{S}\ , \quad \sum_{i \in \mathcal{V}(j)} y_i \geq 1 \tag{2}$$

$$\forall i \in \mathcal{V}, \forall T \subset \mathcal{V} : i \in T \wedge s \notin T, \quad \sum_{e \in \delta(T)} z_e \geq y_i \tag{3}$$

$$y_i, z_e \in \{0, 1\},\ \ i \in \mathcal{V}, e \in E$$

The coverage constraints, (2), require that for each surface at least one view is chosen from its viewpoint set. The connection constraints, (3), require that for each planned view i, i.e., $y_i = 1$, and for every cut T of the vertex set that separates i from the robot start position s, at least one edge that crosses T must be chosen to connect the cut. Such connection constraints are used in the standard (rooted) Steiner tree problem ILP formulation [19], and essentially express the notion that each selected node must be reachable from the start node. Note that the above ILP formulation (1) is not the most compact one, since there are a large number of constraints corresponding to the cuts in the graph. In the following, we will also give a polynomial-sized formulation (especially useful to solve the corresponding relaxed LP). Nonetheless, this formulation gives us a lot of intuition, since it works directly with the edge assignments, and is handy when we analyze the algorithmic performance.

3 Hardness Analysis of Traveling VPP

As the generalization to both SCP and Metric TSP, Traveling VPP is immediately seen to be an NP-hard problem. The hardness of approximation, i.e., the best approximation ratio by any polynomial algorithm, is of great importance to approximation algorithm design. In this section, we use the result in [20] to show the hardness of approximation for Traveling VPP via reductions to the group Steiner tree problem (GST).

GST is defined as follows. Given a graph $G = (V, E)$, where the vertex set V is divided into k distinct groups, $g_1, g_2, \ldots, g_k$, construct the minimum cost Steiner tree to connect at least one vertex from each group. The GST generalizes both the SCP and the Stener tree problem where the best known approximation ratios are $O(\log n)$ and $O(1)$ respectively. In [11, 12], the authors used an LP-based randomized rounding algorithm and a greedy algorithm respectively to achieve the best known poly-log approximation ratio, $O(\log |V| \log\log |V| \cdot \log k \log N)$, where $|V|$, k and N are the number of graph nodes, the number of groups and the maximum cardinality of the groups respectively.

We now show that the Traveling VPP is not approximable within the same poly-log ratio via the reduction given in [11]. By considering each group in any GST instance as the viewpoint set of a surface patch in the Traveling VPP, GST is reduced to a special case of the Traveling VPP where the viewpoint sets that cover different surface patches are exclusive, i.e., they do not share common viewpoints. Thus, the Traveling VPP is certainly at least as hard as GST and cannot be approximated within $\log^{2-\epsilon} |\mathcal{S}|$, following the result of [20], where Halperin and Krauthgamer show that the optimal solution to GST cannot be approximated by any polynomial algorithm within the $O(\log^{2-\epsilon} k)$ ratio, for any $\epsilon > 0$. The question remains whether Traveling VPP is harder to approximate. We show in the following that the inapproximalities of Traveling VPP and GST are of the same order by showing a reduction from Traveling VPP to GST in two steps.

We first show how to reduce an arbitrary Traveling VPP instance to a Traveling VPP instance with 0 view cost. Given an arbitrary Traveling VPP instance with unit view cost and unit traveling cost w_v and w_p respectively, we add a new viewpoint i', for each original viewpoint i, with an identical surface patch set, let the surface patch set for i be empty, and connect i' to i via an edge with cost of $\frac{w_v}{w_p}$. It is easy to see an optimal solution to the reduced (0 view cost) version of Traveling VPP corresponds to an optimal solution to the original Traveling VPP instance since view costs are encoded in the edge costs of the reduced Traveling VPP instance. (Since the surface patch set of the original viewpoint is empty, the new solution has to go to the new copy of the viewpoint, thus incurring the traveling cost $\frac{w_v}{w_p}$ which is equivalent to adding $\frac{w_v}{w_p} \cdot w_p = w_v$ in the objective function.) The size of the resulting instance has $2|\mathcal{V}|$ number of viewpoints and $|\mathcal{V}| + |\mathcal{S}|$ number of edges.

We now show how to construct a GST instance from a special case of a

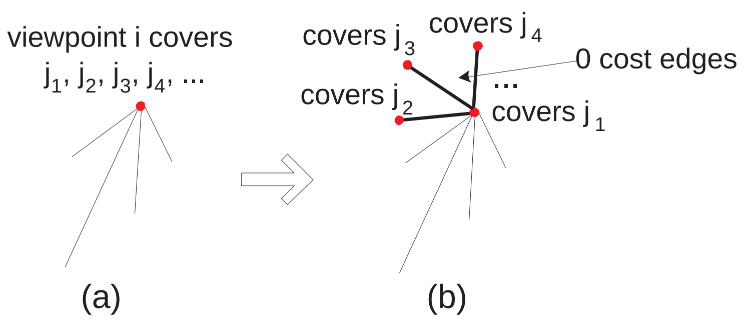

Figure 3: Construction of an GST instance from Traveling VPP with 0 view cost.

Traveling VPP instance with 0 view cost. The idea is to duplicate each viewpoint many times (equal to the number of surface patches it covers) to make the resulting viewpoint sets distinct. Consider such a Traveling VPP instance, i.e., the viewpoint set $\mathcal{V}$, the surface patch set $\mathcal{S}$, the viewpoint set $\mathcal{V}(j) \subseteq \mathcal{V}$ for surface patch $j \in \mathcal{S}$, and the graph G that connects $\mathcal{V}$. We first construct a group g_j for each surface patch j and construct a vertex for each pair of a surface patch j and one viewpoint from its viewpoint set, i.e., (j, i), $i \in \mathcal{V}(j), j \in \mathcal{S}$. We modify the graph G of Traveling VPP accordingly by first constructing a tree with 0-cost edges between the vertices corresponding to the same viewpoint, i.e., $\{(j, i) : j \in \mathcal{S}(i)\}$ (picking an arbitrary vertex (j, i) as the tree root) and then placing the tree root at the node i on G. See Fig. 3. Thus, we have a GST instance on the graph over vertices in the form (j, i) and groups corresponding to the surface patches j. And it is easy to see that an optimal GST solution that picks vertices (j, i) and Steiner tree between them correspond to an optimal solution to Traveling VPP of picking viewpoints i and the resulting Steiner tree connection by collapsing the 0-cost edges. The above GST instance construction produces $O(|\mathcal{V}||\mathcal{S}|)$ number of vertices and $O(E + |\mathcal{V}||\mathcal{S}|)$ number of edges.

By combining the two reduction steps above, an arbitrary Traveling VPP instance with viewpoint set $\mathcal{V}$ and surface patch set $\mathcal{S}$ is reduced to a GST instance with a graph having $O(|\mathcal{V}||\mathcal{S}|)$ vertices, $O(|\mathcal{V}||\mathcal{S}|)$ edges, and $|\mathcal{S}|$ groups. As a result, the Traveling VPP is inapproximable within $O(\log^{2-\epsilon} |\mathcal{S}|)$ ratio of the optimal using any polynomial algorithm. Also the best known approximation algorithms mentioned at the beginning of this section can be applied to the Traveling VPP (after the reductions given above) and the approximation ratio is $O(\log |\mathcal{V}| \log\log |\mathcal{V}| \cdot \log |\mathcal{S}| \log F)$, where F is the view frequency.

4 LP based Algorithms for Traveling VPP (ILP)

In this section, we first give the LP relaxation for the ILP formulation given in Section 2.3, introduce a rounding algorithm to get an integral solution from the LP solution, give the approximation ratio analysis, and then discuss how to solve the LP.

4.1 LP Relaxation for Traveling VPP

By relaxing the binary integral variables, y_i and z_e, to be positive reals, we have the relaxed linear program (LP) formulation given as:

$$\begin{aligned} &\underline{\textbf{LP Relaxation:}} \quad \min && w_v \sum_{i \in \mathcal{V}} y_i + w_p \sum_{e \in E} c_e z_e \\ &\text{Subject to:} \quad \forall j \in \mathcal{S} \ : && \sum_{i \in \mathcal{V}(j)} y_i \geq 1 && (4) \\ &\forall i \in \mathcal{V}, \forall T \subset \mathcal{V} : i \in T \wedge s \notin T : && \sum_{e \in \delta(T)} z_e \geq y_i && (5) \\ & && y_i, z_e \geq 0, \ i \in \mathcal{V}, e \in E \end{aligned}$$

We call the optimal (fractional) solution and the corresponding cost the *LP-optimal solution* and *LP-optimal value* respectively. The LP-optimal solution corresponds to the fractional *LP-optimal viewpoint assignments* and the fractional *LP-optimal edge assignments.* We call the optimal (integral) solution and corresponding cost to the original ILP the *ILP-optimal solution* and *ILp-optimal value*, respectively. The ILP-optimal solution correspond to the integral *ILP-optimal viewpoint assignments* and the integral *ILP-optimal edge assignments.*

4.2 Rounding Algorithm

Let y_i^* and z_e^* denote the Lp-optimal viewpoint assignments and the LP-optimal edge assignments respectively, and let OPT^* denote the LP-optimal value, i.e., $OPT^* = w_v \sum_{j \in \mathcal{V}} y_i^* + w_p \sum_{e \in E} c_e z_e^*$. Let y_i', z_e' denote the algorithmic integral solution by the algorithm *Round and Connect* given below, and let $cost'$ denote the corresponding cost, i.e., $cost' = w_v \sum_{j \in \mathcal{V}} y_i' + w_p \sum_{e \in E} c_e z_e'$. Throughout this paper, we use the superscript $*$ to denote the LP-optimal solution/cost to the corresponding problem instance; and use superscript $'$ to denote a feasible ILP solution/cost. The algorithm *Round and Connect* is given below:

Algorithm *Round and Connect: (take LP-optimal y_i^*, z_e^* as input and output y_i', z_e')*

Step 1. **Initialize.**

```
Set viewpoint choice set $\mathcal{V}^c$ to include all the viewpoints, i.e., $\mathcal{V}^c \leftarrow \mathcal{V}$; the viewpoint solution set $\mathcal{V}'$ to be empty, i.e., $\mathcal{V}' \leftarrow \emptyset$; the uncovered surface patch set $\mathcal{S}^u$ to include all surface patches, i.e., $\mathcal{S}^u \leftarrow \mathcal{S}$
Step 2. Round.
While set $\mathcal{S}^u$ is not empty
    Select the viewpoint $i_{max} \in \mathcal{V}^c$ that covers some uncovered surface patch(es) and has the largest Lp-optimal viewpoint assignment, i.e., $i_{max} = \underset{i \in \mathcal{V}^c:\ \mathcal{S}(i) \cap \mathcal{S}^u \neq \emptyset}{\arg\max} y_i^*$, and add it to $\mathcal{V}'$, i.e., $\mathcal{V}' \leftarrow \mathcal{V}' \cup \{i_{max}\}$
    Delete the surface patch(es) that $i_{max}$ covers from the uncovered surface patch set, i.e., $\mathcal{S}^u \leftarrow \mathcal{S}^u \setminus \mathcal{S}(i_{max})$; and delete $i_{max}$ from the viewpoint choice set, i.e., $\mathcal{V}^c \leftarrow \mathcal{V}^c \setminus \{i_{max}\}$
Output $\mathcal{V}'$, i.e., set $y_i' = 1$ for $i \in \mathcal{V}'$, and set $y_i' = 0$ for $i \notin \mathcal{V}'$.
Step 3. Connect.
Get the optimal solution to the Steiner tree problem to connect $\mathcal{V}'$. Set $z_e' = 1$ for edges in the solution, and $0$ otherwise.
```

In the above algorithm, we iteratively choose the viewpoint with the largest (fractional) LP-optimal viewpoint assignment until all the surface patches are covered. We then feed these chosen viewpoints to a Steiner tree algorithm to get the optimal integral solution, in the Connect step. Note that the Steiner tree problem is an NP-complete problem for a general graph. So practically speaking, we can use a constant-ratio approximation algorithm, for example the one in [19], and incur an additional bounded performance degradation. It is easy to see that the rounding part of the above algorithm (up to the Connect step) runs in polynomial time, $O(|\mathcal{V}||\mathcal{S}|)$.

4.3 Approximation ratio for algorithm *Round and Connect*

It is trivial to see that the solution given by algorithm *Round and Connect* is a feasible integral solution. In the following, we analyze the performance of the algorithm using the fact that the LP-optimal value is a *lower bound* on the ILP-optimal value. We first show that the view part of the cost of the solution given by the algorithm is bounded and then bound the total cost using a feasible *hybrid solution* with integral viewpoint assignments and fractional edge assignments.

4.3.1 View cost analysis

In the following, we show in Lemma 2 that the LP-optimal viewpoint assignments of the chosen viewpoints are lower bounded by $\frac{1}{F}$. This follows immediately from Lemma 1, which is a simple observation based on the feasibility of the LP-optimal solution. In Corollary 1, these results are used to bound the view cost part of the algorithmic solution.

Lemma 1 *For any surface patch, there exists a viewpoint that covers it with* *the corresponding LP-optimal viewpoint assignment greater than* $\frac{1}{F}$*, i.e.,* $\forall j \in \mathcal{S},\ \exists i \in \mathcal{V}(j) : y_i^* \geq \frac{1}{F}$.

Proof. We show this by contradiction. Assume that for some surface patch $j \in \mathcal{S}$, all the LP-optimal viewpoint assignments are strictly less than $\frac{1}{F}$, i.e., $y_i^* < \frac{1}{F}, \forall i \in \mathcal{V}(j)$. By recalling that view frequency F is the maximum number of viewpoint that covers any surface patch (i.e., $|\mathcal{V}(j)| \leq F, \forall j \in \mathcal{S}$), we must have,

$$\sum_{i \in \mathcal{V}(j)} y_i^* < \sum_{i \in \mathcal{V}(j)} \frac{1}{F} = |\mathcal{V}(j)| \cdot \frac{1}{F} \leq 1$$

The above implies that for $j \in \mathcal{S}$, the sum of covering viewpoint assignments is strictly less than 1, or in other words, surface j is not covered. This contradicts the feasibility of the LP solution, specifically the constraints (4). ■

Lemma 2 *The LP-optimal viewpoint assignment for each viewpoint chosen by Algorithm* Round and Connect *is lower bounded by* $\frac{1}{F}$*, i.e.,* $y_i^* \geq \frac{1}{F}, \forall i \in \mathcal{V}'$.

Proof. It is equivalent to show that the above algorithm cannot choose any viewpoint whose LP-optimal viewpoint assignment is less than $\frac{1}{F}$. We show this by contradiction. Assume we choose one viewpoint i with $y_i^* < \frac{1}{F}$. By the *Round and Connect* algorithm, the Round Step, at the iteration when i is picked, it has the maximum LP-optimal viewpoint assignment among the viewpoints that covers the remaining uncovered surface(s). We arbitrarily choose one uncovered surface patch that i covers. By Lemma 1, there exists another i' for which $y_{i'}^* \geq \frac{1}{F}$. This implies $y_{i'}^* > y_i^*$. i' has not yet been chosen, since otherwise all its covering surface pathes including this *uncovered* one would have been deleted from uncovered surface patch set. This contradicts that i has the largest LP solution, y_i^*, among unchosen viewpoints that cover uncovered surface patch(es). ■

Lemma 2 implies that the view cost part of the algorithmic solution is bounded by the view cost of the LP-optimal, as stated in Corollary 1.

Corollary 1 *Algorithm* Round and Connect *gives an integral solution with view cost at most F times the view cost of the LP-optimal solution, i.e.,* $w_v \sum_{i \in \mathcal{V}} y_i' \leq F \cdot w_v \sum_{i \in \mathcal{V}} y_i^*$.

Proof. By Lemma 2, we have $Fy_i^* \geq 1$, for all the chosen viewpoint $i \in \mathcal{V}'$. It follows that

$$w_v \sum_{i \in \mathcal{V}} y' = w_v \sum_{i \in \mathcal{V'}} 1 \leq F \cdot w_v \sum_{i \in \mathcal{V}'} y^* \leq F \cdot w_v \sum_{i \in \mathcal{V}} y^*$$

■

4.3.2 Total cost analysis

In the following, after stating in Lemma 3 the half-integrality gap result of the Steiner tree problem [10], we show that the solution given by the algorithm

Round and Connect has a total cost at most $2F$ times the LP-optimal value. Since the LP-optimal value is a lower bound on the ILP solution, we now show that the algorithm *Round and Connect* has approximation ratio of $2F$.

Lemma 3 *For the Steiner tree problem, the integrality gap between the IP and its relaxed LP is* 2.

Proof. See Chapter 22 of [10]. ■

Note that the Connect Step of the algorithm *Round and Connect* corresponds to the Steiner tree problem of connecting $\mathcal{V}'$, the ILP-optimal solution to which is z'_e. We use OPT'_{tree} to denote the corresponding optimal value, i.e., $OPT'_{tree} = \sum_{e \in E} c_e z'_e$. Again, we use OPT^*_{tree} to denote the corresponding relaxed LP-optimal value. Now we are ready to show the approximation ratio of algorithm *Round and Connect*.

Theorem 1 *Algorithm* Round and Connect *has the approximation ratio of* $2F$*, i.e.,* $cost' \leq OPT^* \cdot 2F$*.*

Proof. To prove the approximation ratio result, we utilize an intermediate solution with integral viewpoint assignments and fractional edge assignments. We emphasize this solution is only used in the proof and not computed in the algorithm. This solution is denoted by y'_i, z^s_e. The viewpoint assignments are the same as in the algorithm output y'_i, and the edge assignments are scaled by F, i.e., $z^s_e = F z^*_e$. The superscript s denotes it is a solution after scaling. We call it the *hybrid solution*, and denote the total cost of this solution $cost^h$. By Corollary 1 and the edge scaling, we have,

$$\begin{aligned} cost^h &= w_v \sum_{i \in \mathcal{V}} y'_i + w_p \sum_{e \in E} c_e z^s_e \\ &\leq F \cdot w_v \sum_{i \in \mathcal{V}} y^*_i + F \cdot w_p \sum_{e \in E} c_e z^*_e \leq F \cdot OPT^*. \end{aligned}$$

Now, we claim that the hybrid solution is a feasible solution to the LP relaxation of Traveling VPP. Sine the viewpoint assignments of the hybrid solution is exactly the same as in the solution given by the algorithm *Round and Connect*, all the covering constraints, (4), are satisfied by the solution viewpoint set $\mathcal{V}'$. The connection constraints, (5), are also satisfied, since

$$\sum_{e \in \delta(T)} z^s_e = \sum_{e \in \delta(T)} F z^*_e \geq F y^*_i \geq y'_i.$$

The first inequality above is due to the feasibility of the Lp-optimal solution, and the second is due to Lemma 2.

Since all y'_i are integral, z^s_e is a feasible LP solution to the Steiner tree problem to connect $\mathcal{V}'$. It follows immediately that the connection cost $\sum_{e \in E} c_e z^s_e$ is at least the LP-optimal value to connect $\mathcal{V}'$, i.e.,

$$\sum_{e \in E} c_e z^s_e \geq OPT^*_{tree}.$$

Note that the algorithm *Round and Connect* (the Connect Step) gives an optimal integral Steiner tree solution to connect $\mathcal{V}'$. By the integrality gap result for Steiner trees, Lemma 3, this tree cost is at most twice the LP-optimal value for the Steiner tree problem to connect $\mathcal{V}'$, i.e., $\sum_{e \in E} c_e z'_e = OPT'_{tree} \leq 2 \cdot OPT^*_{tree}$. So we have,

$$\sum_{e \in E} c_e z'_e \leq 2 \cdot OPT^*_{tree} \leq 2 \cdot \sum_{e \in E} c_e z^s_e = 2F \cdot \sum_{e \in E} c_e z^*_e$$

(The last equality above is due to the edge scaling.)

Combined with the view cost part of the algorithmic solution (Corollary 1), we have,

$$\begin{aligned} cost' &= w_v \sum_{i \in \mathcal{V}} y'_i + w_p \sum_{e \in E} c_e z'_e \\ &\leq F \cdot w_v \sum_{i \in \mathcal{V}} y^*_i + 2F \cdot w_p \sum_{e \in E} c_e z^*_e \\ &\leq OPT^* \cdot 2F, \end{aligned}$$

which implies the algorithm *Round and Connect* has approximation ratio of at most $2F$. ■

4.3.3 Integrality gap for *Traveling VPP*

Theorem 1 shows that the algorithm *Round and Connect*, recovers an integral solution from any LP-optimal solution to Traveling VPP and the solution cost is within $2F$ times the optimal value. This implies that the integrality gap between ILP-optimal and LP-optimal for Traveling VPP is at most $2F$. In the following, we show that $2F$ is also the integrality gap for *Traveling VPP* by giving an example that achieves this ratio.

Theorem 2 *The integrality gap of the Traveling VPP is* $2F$.

Proof. We show the integrality gap of Traveling VPP is at least $2F$ by giving a Traveling VPP instance where the $2F$ ratio is achieved.

In the Traveling VPP instance in Fig. 4, there are n surface patches. There are n clusters of viewpoints, denoted by $\mathcal{C}_1, \ldots, \mathcal{C}_n$ respectively, each of which corresponds to one surface patch. Each cluster has exactly F viewpoints, each of which covers only the corresponding surface patch of that cluster. We label a viewpoint by the cluster it belongs to and its index within that cluster, for example viewpoint $i_{\mathcal{C}_1,1}$. There are two types of edges, e^1 and e^2, in the graph, superscript 1 or 2 denote the edge type. e^1 edges have the common edge cost $\epsilon \ll 1$ and e^2 edges have the common cost 1. The e^1 edges in each cluster form a complete graph; and the e^2 edges form a complete graph between the clusters. We also use an e^1 edge to connect the robot start position s to a single viewpoint

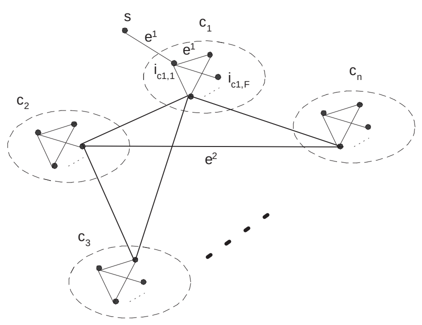

Figure 4: A Traveling VPP instance. There are $n \cdot F$ viewpoints, grouped into n clusters (circled by dashed curves). Each cluster contains exactly F viewpoints. For example, cluster $\mathcal{C}_1$ contains viewpoints $i_{\mathcal{C}_1,1}, \ldots, i_{\mathcal{C}_1,F}$. There are n surface patches (not drawn). All viewpoints in a cluster, $i \in \mathcal{C}_j$, only cover one surface patch indexed the same as the cluster, $j \in \mathcal{S}$. Two types of edges, labeled by e^1 and e^2, connect the viewpoints. Inside each cluster, e^1 edges form a complete graph. Between clusters, e^2 edges form another complete graph among the representative viewpoints of the clusters, one representative per cluster.

of a single cluster. We further assume the view cost is negligible compared with traveling cost, i.e., $w_v \ll w_p$.

It is not difficult to see that the ILP-optimal solution is to choose a single viewpoint from each cluster and construct a tree (which is also a path connecting these viewpoints) using $(n-1)$ e^2 edges, and the corresponding ILP-optimal value is approximately $(n-1) \cdot 1$ (neglecting view cost and e^1 edge costs). The LP-optimal solution, however, is to assign $\frac{1}{F}$ to each viewpoint and $\frac{1}{n-1} \cdot \frac{1}{F}$ to each e^2 edge. (Since e^1 edges do not contribute much to the objective function, we can simply ignore all the e^1 edges in the solution.) This solution is feasible since for any viewpoint, the cuts that separate it from other clusters have at least $n-1$ e^2 edges, and the sum of such edge assignments is at least $(n-1) \cdot \frac{1}{n-1} \cdot \frac{1}{F} = \frac{1}{F}$, the viewpoint assignment. The corresponding LP-optimal value is thus approximately $\frac{1}{(n-1)F} \cdot \left(\begin{array}{c} n \\ 2 \end{array} \right) = \frac{n}{2F}$. (There are all together $\left(\begin{array}{c} n \\ 2 \end{array} \right)$ number of e^2 edges.) So the ratio between ILP and LP-optimal values approaches $2F$ assuming n is large, and the integrality gap for the general problem is at least $2F$.

In conclusion, since both the upper and lower bounds are $2F$, the integrality gap of Traveling VPP must be $2F$. ■

The integrality gap result for Traveling VPP, Theorem 2, suggests that the $2F$ approximation algorithm *Round and Connect* is the best possible for the LP relaxation given above.

4.4 Solving the relaxed LP

With the algorithm *Round and Connect*, we can recover an integral solution from a relaxed LP-optimal solution, with the approximation ratio of $2F$ for a general *Traveling VPP*. However, the corresponding relaxed LP formulation may have exponential number of connection constraints, (constraint 5). In the following, we first consider a special but important case, the *Traveling VPP on a Tree*, i.e., the graph G connecting the viewpoints is a tree, and give a polynomially-sized LP relaxation formulation. It is motivated by tree structures commonly used in motion planning techniques to explore and represent the connectivity of the configuration space [21, 22]. For general graph case, we suggest two ways here: to use an alternative LP relaxation formulation with polynomial size based on multi-commodity flows; or to adopt the column generation approach [23] to practically solve the LP. Please see the appendix for the derivation of the alternative LP formulation; and see [24] for the development of the column generation approach.

4.4.1 Traveling VPP on a Tree

First, we claim that the approximation ratio of algorithm *Round and Connect* improves to F for *Traveling VPP on a Tree*. This is because there is no integrality gap for "Steiner tree on a tree", since both the ILP-optimal and LP-optimal solutions of "Steiner tree on a tree" correspond to taking the union of the unique paths on the tree that connect the planned viewpoints to the start position.

However, we emphasize that *Traveling VPP on a Tree* is not a simple problem. First, note that the counterexample given in the introduction is also a *Traveling VPP on a Tree* instance. Second, we show, via a counterexample, that a greedy algorithm based on amortized costs can perform quite poorly (linear approximation ratio). The algorithm is to iteratively pick a viewpoint with the least amortized cost, i.e., the sum of the view cost and the shortest path cost to connect to the existing tree, divided by the number of uncovered patch(es) it covers, and iteratively grow the existing tree using this shortest path. Although greedy algorithms based on amortized cost have been shown to achieve the logarithmic approximation ratio (the best approximation ratio) for the SCP [10], it is not so for *Traveling VPP on a Tree*. Consider the example in Fig. 5. We have to choose either viewpoint i_1 (which covers all the surface patches, but connected to the start via a long edge) or all the remaining viewpoints (connected via much shorter edges). It is not difficult to see that the optimal solution is to choose $i_2, \ldots, i_n$ and edges $e_2, e_{i_2}, \ldots, e_{i_n}$, and the cost is roughly 1, the edge cost of e_2. The algorithm, however, will choose i_1 since the amortized cost of i_1, $\approx \frac{n-1}{n-1} = 1$, is less than that of any other viewpoint, $\frac{1+\epsilon}{1} = 1 + \epsilon$. The algorithmic solution cost is thus $(n - 1)$, arbitrarily worse than the optimal value (1). Intuitively, this is because the large cost edge is "underestimated" by the large number of surface patches in the amortized cost.

LP formulation for *Traveling VPP on a Tree*

The *Traveling VPP on a Tree* admits a polynomial-sized relaxed LP formu-

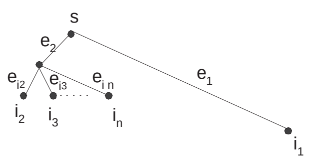

Figure 5: An instance of the Traveling VPP on a Tree. There are n viewpoints, labeled by $i_1, i_2, \ldots, i_n$, and $n-1$ number of surface patches (not drawn). Viewpoint i_1 covers all the surface patch. Each of viewpoints $i_2, \ldots, i_n$ covers only one surface patch, but all together they cover all the patches. Viewpoint i_1 is connected to s via a long edge e_1 with the cost of $n-1$. The remaining viewpoints are first connected to a common node (these connections have negligible costs) and then to s via a short edge e_2 with the cost of $1+\epsilon$, where ϵ is a small positive number. We also assume the traveling cost dominates, and the view cost is negligible.

lation. Since Linear Program is in P [18], we have a polynomial time approximation algorithm (with view frequency as the approximation ratio) for *Traveling VPP on a Tree* by solving first its LP and using *Round and Connect* to recover an integral solution. In the following, we show the polynomial-sized formulation. Intuitively, for a viewpoint to be connected, only the cuts corresponding to the edges on its unique path (to the start s) are needed in the connection constraints, (constraint 5), thus reducing dramatically the LP size.

Let p_i denote the unique path connecting viewpoint i to s. For an edge $e =< i_1, i_2 >$, with i_1 closer to the root of the tree, s, than i_2, we use T_e to denote the subtree of the original tree rooted at i_2, i.e., the subset of tree vertices that are connected to s via e. Note that e is the only edge that crosses the subset T_e, i.e., $\delta(T_e) = \{e\}$. The LP for *Traveling VPP on a Tree* is then given as:

$$\begin{aligned}
&\min && w_v \sum_{i\in\mathcal{V}} y_i + w_p \sum_{e\in E} c_e z_e \\
&\text{Subject to: } \forall j \in \mathcal{S}, && \sum_{i\in\mathcal{V}(j)} y_i \geq 1 && (6)\\
&\forall i \in \mathcal{V}, \forall e \in E : e \in p_i, && z_e \geq y_i && (7)\\
& && y_i, z_e \geq 0, i \in \mathcal{V}, e \in E
\end{aligned}$$

Since the covering constraints for the above formulation and for *Traveling VPP* are the same, we only need to show the equivalence of the connection constraints, (5) and (7). We show this equivalence by reductions in both directions. First, for $i \in \mathcal{V}$ and $e \in p_i$, since T_e is a cut that separates viewpoint i from the start position s and e is the only edge crossing T_e, according to (5), we have

$\sum_{e' \in \delta(T_e)} z_{e'} = z_e \geq y_i$, (7). Second, for any cut T that separates i and s, there must be at least an edge $e \in p_i$ that crosses T, i.e., $e \in \delta(T)$. (Otherwise i and s will not be separated by T.) So $z_e \geq y_i \Longrightarrow \sum_{e' \in \delta(T)} z_{e'} \geq z_e \geq y_i$, hence (5) and (7) are equivalent for the tree case.

Note that in the above formulation, the number of constraints is $O(|\mathcal{S}| + |E||\mathcal{V}|)$.

4.5 Poly-log approximation algorithm via reduction to GST

By the reduction mentioned in Section 3, we can construct from an arbitrary Traveling VPP instance a GST instance with $O(|\mathcal{V}||\mathcal{S}|)$ vertices, $O(|\mathcal{V}||\mathcal{S}|)$ edges, and $|\mathcal{S}|$ groups. We then apply the randomized rounding algorithm in [11] to achieve a $O(\log |\mathcal{V}| \log\log |\mathcal{V}| \log |\mathcal{S}| \log F)$, a poly-log, approximation ratio.

In conclusion, using both the LP based algorithms, deterministic (Round and Connect algorithm) and randomized (the rounding algorithm in [11]) rounding algorithms respectively, we have the approximation ratio of either $O(F)$ or $O(\log |\mathcal{V}| \log\log |\mathcal{V}| \log |\mathcal{S}| \log F)$, whichever is smaller. This approximation result parallels that for SCP.

5 Applications of Traveling VPP Algorithm to Related Problems

In this section, we apply the algorithm Round and Connect to several related problems and extend the best known approximation ratios for these problem from poly-log to the minimum of poly-log and the order of frequency.

5.1 GST

Note that a GST instance is a Traveling VPP instance with 0-view cost. So the frequency bound is just the largest cardinality of the groups and applying algorithm Round and Connect gives us a frequency approximation ratio. Thus by applying the algorithm Round and Connect above and LP randomized rounding in [11], the optimal solution to GST is approximated within the ratio of $\min(O(F), O(\log |V| \log\log |V| \log k \log N))$.

5.2 Traveling Purchaser Problem (TPP)

Given a set of warehouses, $\mathcal{W}$, connected by a graph $G = (\mathcal{W}, E), E \subseteq 2^{\mathcal{W} \times \mathcal{W}}$, and a set of products $\mathcal{P}$ with requirements and the prices of buying a product at a warehouse, $d_{w,p}, w \in \mathcal{W}, p \in \mathcal{P}$, the (unlimited capacitated) Traveling Purchaser problem (TPP) asks for certain warehouses for each product and the tour connection between the planned warehouses with the minimum total cost, the sum of the product purchase cost and the tour cost [13]. In [13], the authors give a poly-log approximation ratio for TPP.

The Traveling VPP is a special case of TPP where the prices are uniform for all the product and warehouse pairs. We now show how to reduce an arbitrary TPP instance to an instance of a weighted version of Traveling VPP. By "weighted", we mean the view costs associated with the viewpoints are not uniform. Since this weighted version has exactly the same constraints as Traveling VPP, the algorithm Round and Connect only rounds to 1 the viewpoints with assignments lower bounded by $\frac{1}{F}$. By exactly the same arguments as in Section 4.3, it is clear that the algorithm Round and Connect still has the frequency bound approximation ratio for the weighted version. The reduction from TPP to Traveling VPP is done by "duplicating" a warehouse according to the number of different product prices it offers and connecting them via 0-cost edges.

Formally, we define the price set of a warehouse to the set of different prices it offers for different products, i.e., $D(w) = \{d_{w,p}, w \in \mathcal{W}, p \in \mathcal{P}(w)\}$. We order the set $D(w)$ according to the price entities, and denote the i^{th} smallest price as $d(w)_i$. We then add warehouses $w_i, i = 2, \ldots, |D(w)|$ to the warehouse set $\mathcal{W}$, (We replace the viewpoint w by w_1 with a different product set described as follows.) and add edge $< w_1, w_i >$ with 0 edge cost. The warehouse w_i will cover only the product in its product set that has the price of $d(w)_i$, i.e., $\mathcal{P}(w_i) = \{p \in \mathcal{P}(w) : d(w,p) = d(w)_i\}$. By this construction, each warehouse has a unique price for all the products in its product set. This corresponds to an instance of the weighted Traveling VPP. Note that the frequency bounds for both instances are the same, since we have not increased the number of warehouses that offer a single product.

Thus, by solving the resulting weighted Traveling VPP instance, we can get the $O(F)$ approximation ratio for TPP, where F is defined as the maximum number of warehouses that sell a single product.

6 Issues Towards Implementation

In this section, we discuss several issues and constraints towards implementing our algorithm on a real robotic system.

Our Traveling VPP formulation assumes the viewpoint set $\mathcal{V}$ and the traveling graph G connecting these viewpoints are given. For given scenes, these viewpoints can be derived from the aspect graph of the scene [25], or by randomly sampling the sensor configuration space, the space of the sensor configurations that uniquely determines the viewpoints [26]. We assumed a binary coverage relationship mentioned, i.e., a viewpoint can either cover a surface patch or not. In reality, a viewpoint may cover a surface patch only partially. By subdividing surface patches, we can maintain binary coverage relationship. Realistic sensor field of view constraints such as the line of sight constraint (i.e., a viewpoint sees a surface patch only if the line segment that connects them is not occluded), the range constraint (a viewpoint sees a surface patch only if the distance between them is within a range), and the incidence constraint (i.e., a viewpoint sees a surface patch only if the angle between the line connecting them and the surface normal is in a range) can be incorporated via viewpoint and surface patch visibility computations. The Traveling graph would essentially be a roadmap built in the configuration space of the robot. This is a standard and well-studied technique for robot motion planning [5, 6].

For accurate registration, it is desirable that two planned consecutive viewpoints should have enough overlap in the surface patch sets they cover [2]. This can be incorporated using a set multicover constraint, i.e., each element in the universe needs to be covered by a specified number of subsets in the solution. The idea is as follows. We create new surface patches that are composed of unions of parts of original consecutive patches. The viewpoint set of these created patches are those covering both consecutive patches. By requiring that the viewpoints cover these created surface patches twice, i.e., changing r.h.s. of (2) from 1 to 2 for these patches added, these viewpoints can register them w.r.t. each other and thus the overall viewpoints can all satisfy the overlapping constraints.

7 Conclusion and Future Work

In this paper, we introduced the Traveling VPP, the problem of view planning with traveling costs. We formulated the problem for the model-based case where the geometry of the object to inspect is known. By appropriate reductions, we showed the Traveling VPP is a combination of set covering and TSP. We also showed that it is at least as hard as the GST problem, and hence it is log-square inapproximable. We gave an LP-based rounding algorithm that has the frequency factor approximation ratio. Together with the poly-log approximation ratio achieved via LP-based randomized rounding, Traveling VPP can be approximated within the minimum of a constant times frequency and a poly-logarithmic function of the input size. This parallels the approximation ratio result for the set covering problem. We then discuss several realistic issues and constraints towards implementing our algorithm for a real system.

In the future, we would like to generalize our result to the case where the surface patches to cover and the robot traveling graph are not known in advance, hence the term sensor-based Traveling VPP. It is also interesting to apply the algorithm designed in this paper to where the viewpoint set is a continuous space, for example, the watchman route problem [3]. The idea is to identify the critical points in a watchman route problem and consider the resulting problem using these critical points as the viewpoints in a Traveling VPP instance [27].

Appendix

Reduction from SCP to VPP

An arbitrary SCP instance is given by a universe of elements $\mathcal{S} = \{s_j, j = 1, \ldots, m\}$ and a collection of its subsets $\mathcal{V} = \{v_i : v_i \subseteq \mathcal{S}, j = 1, \ldots, n\}$. We construct a two-dimensional VPP instance (i.e., viewpoints and objects reside in a two-dimensional Euclidean space populated with obstacles) as in Fig. 6. We first draw a circle of radius R_C and let L be a diameter dividing the boundary of the circle into two halves. For one half of the circle, we divide it equally into three parts as shown and put m equally-sized surface patches in the perimeter of the middle part, each of which corresponds to an element of the universe of the SCP instance. The size of each is less than $\frac{R_C \pi}{3m}$. We denote the surface patches using the same labels as those for the elements in the SCP instance, $s_j, j = 1, \ldots, m$, to imply this correspondence. We then create viewpoints, corresponding to the subsets in the SCP instance, and put them on L with distances between two consecutive ones greater than R_O. (See Fig. 6(c).) The interior of the circle is free of obstacles other than those we construct around the viewpoints. We construct a half ring of obstacles of radius R_O and negligible thickness (cut by L). To accommodate n viewpoints, we require that $n \cdot 2R_O < 2R_C$. For the half ring of obstacles around each viewpoint, we make some visibility openings, such that the viewpoint covers only the surface patches corresponding to those elements in the viewpoint's corresponding subset. For example, in Fig. 6(b), for viewpoint v_1 corresponding to subset $\{s_{i1}, s_{i2} \ldots\}$, we make some openings in the half ring of obstacles around v_1 such that it is visible to those surface patches corresponding to elements $s_{i1}, s_{i2}, \ldots$. As show in Fig. 6(c), for viewpoint v_{i+1}, the obstacle around neighboring viewpoints can only occlude its viewing angle of less than 30^o, which implies $s_1, \ldots, s_m$ can only be occluded by obstacle around v_{i+1}. The resulting VPP instance is to plan the minimum number of viewpoints that cover all the surface patches. This is equivalent to asking for the minimum number of subsets, the union of which is the universe. Thus, the reduction from SCP to VPP is constructed.

Alternative polynomial-sized LP relaxation formulation for Traveling VPP

Note that for our Traveling VPP formulation (1), it is the large number of connection constraints using cuts that prevents us from working with its LP relaxation directly and solving for the optimal solution. In the following, we show how to use flow formulation (rather than the cut) for these types of constraints. Thus the resulting LP formulation for Traveling VPP will have a polynomial size.

We first double the edges in our undirected graph G in the Traveling VPP formulation and build a digraph to direct the flows in the graph. With slight abuse of notation, we denote the resulting digraph by $G = (\mathcal{V}, E)$. For each vertex, or viewpoint, i, let $\mathcal{I}(i)$ denote the set of edges having i as their endpoint and let $\mathcal{O}(i)$ denote the set of edges having i as their start-point. We then define the commodity flow for each viewpoint and edge pair, $f_{i,e}, i \in \mathcal{V}, e \in E$ to reinforce the connection between s and i if i is chosen, i.e., we require $\sum_{e \in \mathcal{O}(i)} f_{i,e} \geq y_i$. The idea is to define each viewpoint i as the source of the commodity i and s as the terminal for all the commodities. We require that y_i amount of commodity i flow from source i to s. Then each edge assignment has to guarantee the flow capacity, i.e., $z_e \geq f_{i,e}, \forall i \in \mathcal{V}$. In addition, we require flow conservation for each flow and for each vertex on the graph that is not

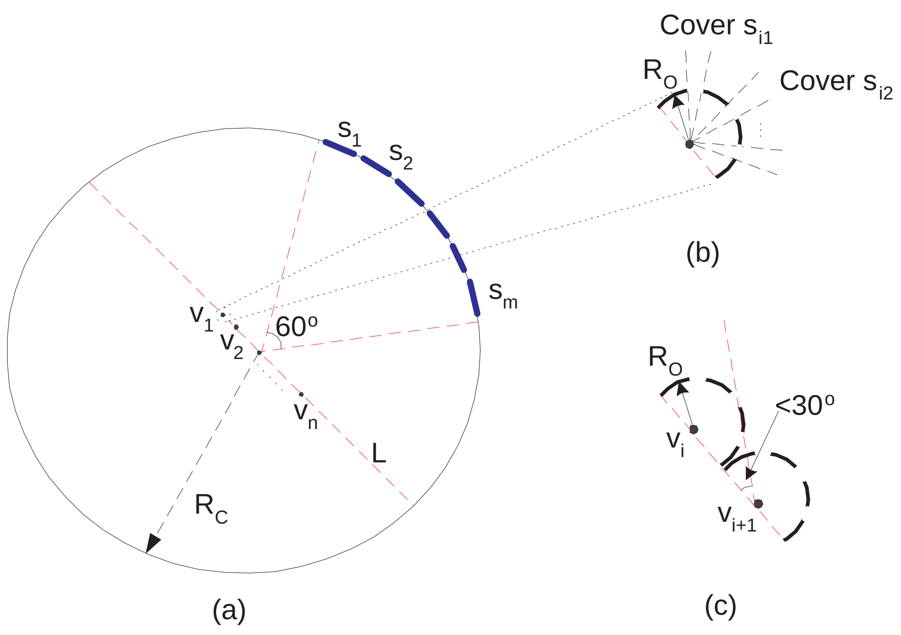

Figure 6: Reduction from an arbitrary SCP instance to a VPP instance. See the text for the reduction.

the terminal of that flow, i.e., $\sum_{e\in\mathcal{I}(i')} f_{i,e} = \sum_{e\in\mathcal{O}(i')} f_{i,e}, \forall i' \notin \{i,s\}$. So the overall relaxed LP is given as:

$$\underline{\textbf{Traveling VPP :}} \tag{8}$$

$$\min \quad w_v \sum_{i\in\mathcal{V}} y_i + w_p \sum_{e\in E} c_e z_e$$

$$\text{Subject to: } \forall j \in \mathcal{S}\ , \quad \sum_{i\in\mathcal{V}(j)} y_i \geq 1$$

$$\forall i \in \mathcal{V} \quad \sum_{e\in\mathcal{O}(i)} f_{i,e} \geq y_i \tag{9}$$

$$\forall i \in \mathcal{V} \quad \sum_{e\in\mathcal{I}(s)} f_{i,e} \geq y_i$$

$$\forall i, \forall i' \notin \{i,s\}, \quad \sum_{e\in\mathcal{I}(i')} f_{i,e} = \sum_{e\in\mathcal{O}(i')} f_{i,e} \tag{10}$$

$$\forall e \in E, \forall i \in \mathcal{V}, \quad z_e \geq f_{i,e} \tag{11}$$

$$y_i,\ f_{i,e},\ z_e \geq 0$$

In the following, we show the equivalence of the above flow-based formulation (8) to the one we give before (1) by showing the feasible solution to one formulation corresponds to a feasible solution to the other.

Lemma 4 *The flow based relaxed LP formulation for Traveling VPP (8) is equivalent to the graph cut based formulation (1).*

Proof. We first show that any feasible solution $y_i, z_e, f_{i,e}$ to (8), corresponds to a feasible solution y_i, z_e to (1). Since the covering constraints are the same for both formulations, we only need to show the connection constraints. For any cut that separates viewpoint i from s, via the flow conservation law, we know the total amount of commodity flow i crossing the cut (from i to s) is at least the amount emanating from i, which in turn is lower bounded by the viewpoint assignment, (9). Since the edge assignment is lower bounded by the flow going through it, (11), we have the total number of edges crossing T is at least y_i.

Second, for any optimal LP solution y_i, z_e to (1), we can pick for each $y_i > 0$ the unique path from i to s in the solution (Note that the optimal LP solution must be a tree, i.e., the positively assigned edges form a tree connection rooted at s, since otherwise we simply delete, and thus reducing the overall cost, some edges while maintaining connectivity. It follows there exists a unique path from any tree node to the root.), and assign the flow for commodity i and edges (directing towards s) on the path to be y_i. It is clear that this construction gives a feasible flow solution to (8). ■

Note that the size of the formulation (8) has $|\mathcal{V}| + |E|$ number of variables and $|\mathcal{S}| + O(|\mathcal{V}||E|)$ constraints.

References

[1] Y. Mei, Y. Lu, Y. Hu, and C.S. Lee. "A case study of mobile robot's energy consumption and conservation techniques". In *Proc. of International Conference on Advanced Robotics* 2005.

[2] W. Scott, G. Roth, and J. Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys*, 35(1), March 2003, pp. 64-96.

[3] W. Chin and S. Ntafos. Watchman routes in simple polygons. *Discrete Comput. Geom.*, 6(1): 9-31, 1991.

[4] X. Tan. Approximation algorithm for the watchman route and zookeeper's problems. *Discrete Applied Mathematics.* 136(2-3): 363-376.

[5] L. Kavraki, P. Svestka, J. Latombe and M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4): 556-580, 1996.

[6] J. Latombe. *Robot motion planning.* Kluwer Academic Publishers, Boston, 1991.

[7] S. Fekete, R. Klein, and A. Nuchter. "Online searching with an autonomous robot". In *Proc. of Workshop on Algorithmic Foundation of Robotics*, 2004.

[8] V. Isler, S. Kannan, K. Daniilidis. "Local exploration: online algorithms and a probabilistic framework". In *Proc. of IEEE ICRA 2003.*

[9] T. Danner and L. Kavraki. Randomized planning for short inspection paths. In *Proc. of IEEE International Conference on Robotics and Automation*, 2002.

[10] V. Vazirani. *Approximation algorithms.* Spinger, 2001.

[11] N. Garg, G. Konjevod, and R. Ravi. A polylogorithmic approximation algorithm for the group Steiner tree problem. *Journal of Algorithms.* 37(1): 66-84, 2000.

[12] C. Chekuri, G. Even, G. Kortsarzc. *A greedy approximation algorithm for the group Steiner problem.* Discrete Applied Mathematics 154 (2006), pp.15-34.

[13] R. Ravi and F. Salman. "Approximation Algorithms for the Traveling Purchaser Problem and its Variants in Network Design". In *Proc. of the 7th Annual European Symposium on Algorithms.* pp. 29-44, 1999.

[14] W. Scott, G. Roth, and J. Rivest. View planning with a registration constraint. In *Proc. 3rd International Conference on 3D Digital Imaging and Modeling*, 2001, pp. 127-134.

[15] C. Swamy and A. Kumar. *Primal-dual Algorithms for Connected Facility Location Problems.* Algorithmica 40 (2004), pp. 245-269.

[16] P. Slavik. *The Errand Scheduling Problem.* Techical Report 97-02. Department. of Computer Science and Engineering, SUNY Buffalo, 1997.

[17] G. Pataki. *Teaching Integer Programming Formulations Using the Traveling Salesman Problem.* SIAM Review 2003. 45(1), pp. 116-123.

[18] C. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithm and complexity.* Prentice Hall, 1982.

[19] M. Goemans and D. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing.* 24(2): 296-317, 1992.

[20] E. Halperin and R. Krauthgamer. "Polylogarithmic inapproximability". In *Proc. of STOC 2003.*

[21] P. Bessiere, J. Ahuactzin, E. Talbi, and E. Mazer. The Ariadne's Clew Algorithm: Global Planning with Local Methods. In K. Goldber, D. Halperin, J. Latombe, and R. Wilson, editors, *Algorithmic Foundation of Robbotics*, pp. 39-47. A K Peters, Ltd., 1995.

[22] S. LaValle and J. Kuffner. Randomized Kinodnamic Planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pp. 473-479, 1999.

[23] G. Desaulniers, J. Desrosiers, and M. Solomon. *Column Generation.* Springer, 2005.

[24] P. Wang, R. Krishnamurti, and K. Gupta. View planning with combined viewing and traveling costs. Technical Report TR 2006-17, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, May 2006.

[25] K. Bowyer, and C. Dyer. Aspect graphs: an introduction and survey of recent results. *International Journal of Imaging Systems and Technology*, 2:315–328, 1990.

[26] H. Gonzalez-Banos and J. Latombe, A randomized art-gallery algorithm for sensor placement. In *Proc. of ACM Symp. on Computational Geometry*, pp. 232–240, 2001.

[27] P. Wang, R. Krishnamurti, and K. Gupta. Generalized watchman route problem with discrete view cost. In preparation.