View Planning Problem with travel costs: Program Formulation, Hardness of Approximation, and Approximation Algorithms

Pengpeng Wang
School of Engineering Science
Simon Fraser University
Burnaby, B.C., Canada

Ramesh Krishnamurti
School of Computing Science
Simon Fraser University
Burnaby, B.C., Canada

Kamal Gupta
School of Engineering Science
Simon Fraser University
Burnaby, B.C., Canada

Technical Report TR 2006-17
School of Computing Science
Simon Fraser University

Abstract

In this report, we tackle the problem of view planning with travel cost, denoted by *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 travel cost for the robot to realize them. We show that the Traveling VPP is equivalent to "Set Covering on a Graph" via reductions from both directions. We use the recent result on the hardness of approximations to show that Traveling VPP is poly-log inapproximable. We give a linear program based rounding algorithm that achieves an approximation ratio of the order of view frequency. Also, we show, via a reduction to the group Steiner tree problem, that the existing approximation algorithm applies to Traveling VPP, thereby providing a poly-log approximation ratio. This parallels the approximation ratio results to the set covering problem, i.e., the best approximation ratio for the set covering problem is either the frequency or the logarithm of the number of elements, whichever is smaller. We then show that our rounding algorithm has a similar frequency factor approximation ratio for other related combinatorial optimization problems by giving the necessary reductions. We also consider several generalizations of Traveling VPP for more realistic models, where image registration constraints and visibility range and angle constraints are accounted for.

1 Traveling VPP: Introduction and Formulation

In this section, we introduce the problem of view planning with travel cost, and formulate as a combinatorial optimization problem.

1.1 Motivations from robotic applications

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 2D or 3D 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. Here we assume line of sight for the sensor which may be subject to occlusions and visibility range. It is desired that the total cost of the plan, comprising the travel cost (the total distance traveled by the robot along the planned path, which is proportional to 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, thus corresponding to intelligent robot behavior. We call this problem *Traveling View Planning Problem*, or *Traveling VPP* in short, and formulate it as an optimization problem. Here 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. 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 travel cost, proportional to the length of the path travelled by the robot.

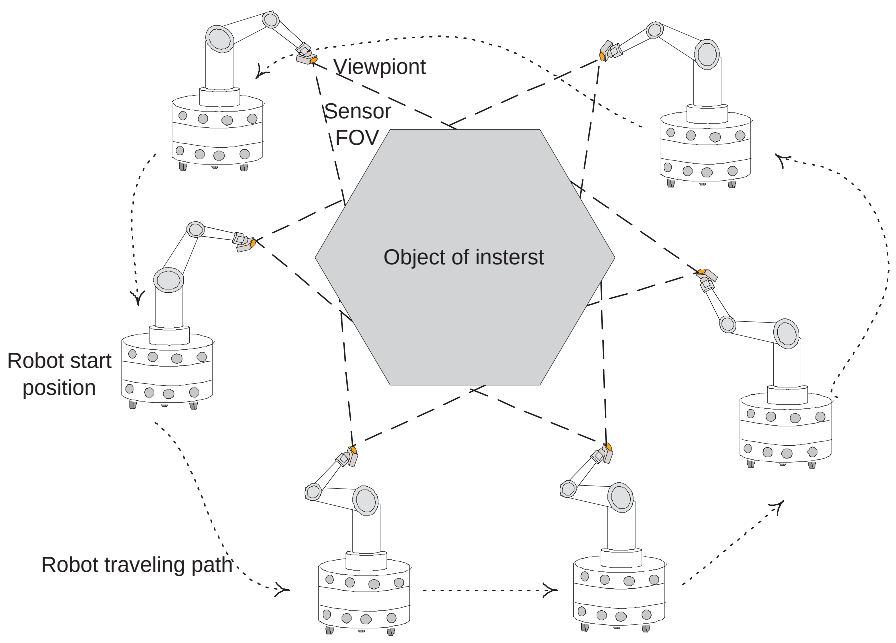


Figure 1: A Traveling VPP instance. It shows 6 planned sensor viewpoints that totally cover the surface of the object of interest, and the robot traveling path to realize them.

The Traveling VPP is clearly a generalization of the view planning problem (VPP), i.e., the problem of planning the minimum number of viewpoints to completely inspect an object surface, considered in the robot vision area [18], where often a sensor positioning system is used within a well controlled and limited workspace. These formulations do not consider the travel cost of the robot, a critical cost, particularly for large workspaces and remote autonomous missions, where power consumption is a critical factor. The Traveling VPP is also a generalization of the watchman route problem, i.e., the problem of planning the shortest tour to inspect the interior of a 2D polygonal region, considered in the computational geometry area [3, 21]. The watchman route problem, however, does not consider view cost, again 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 in 2D environment, *Traveling VPP* uses a graph to encode the traveling. In addition, since we do not assume metrics for the graph, thus Traveling VPP is applicable to more general cases. Such graphs, called roadmaps in the root configuration space, are commonly used for the high-dimensional path planning problem in robot motion planning literature [12, 13].

There are several existing works on combining the view and travel cost in the literature, but not in a unified and global fashion. For example, in [7, 11], 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 travel and the sensor scan time. The problem is considerably simpler since the goal is local, i.e., the objective is not to cover the whole object surface.

In [4], the authors considered the optimization in a global fashion, however, in a "weak sense", since no view cost is considered, thus corresponding to a special case of *Traveling VPP*. They propose to solve the problem by a decoupled two level approach, i.e., to plan the minimum number of viewpoints without considering robot travel cost first and then to solve (approximately) the Metric TSP using the shortest path graph. This two level decoupled approach works 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 travel 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 problem, obtaining the minimum number of viewpoints or the shortest path tour, is solved optimally (as opposed to solving it using approximation algorithms), this two level decoupled approach can perform arbitrarily poorly by pulling the leftmost viewpoint even further apart 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 travel cost is considered at the first stage. This motivates the approach we take in this report, to model the Traveling VPP in a unified formulation and design fast algorithms with guaranteed performances.

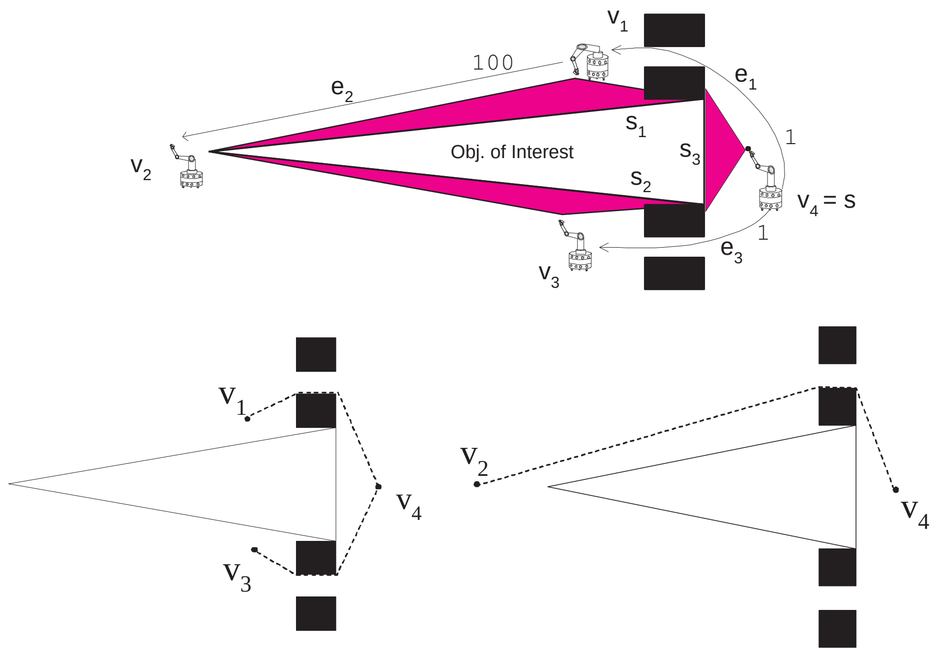
1.2 Traveling VPP: abstraction to set covering on a graph

In this section, we pose Traveling VPP as a combinatorial optimization problem using and generalizing the two well-known problems, namely the set covering problem (SCP) and the metric traveling salesman problem (Metric TSP). We show the equivalence of the view planning problem (VPP) and the SCP via reductions. Considering that the Traveling VPP combines both VPP and the Metric TSP, or in other words, VPP is a special case of Traveling VPP ignoring travel costs, we can also name the Traveling VPP as “set covering on a graph”.

First, let us show the reduction from VPP to SCP. By considering each object surface patch as the subset of the viewpoints that can cover it [17], 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 following, 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.

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 $\mathcal{V} = \{v_i : v_i \subseteq \mathcal{S}, j = 1, \ldots, n\}$. (It should be clear from the above why we use the same notation for viewpoints in VPP and for subsets in SCP.) We construct a 2D VPP instance (i.e., viewpoints and objects reside in a 2D Euclidean space populated with obstacles) as in Fig. 3. We first draw a circle and draw a line L (This line does not correspond to any obstacle.) to divide the inner boundary of the circle into two parts. We put n surface patches on the circle's inner surface in only one part, each of which corresponds to an element of the universe of the SCP instance. 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 R distance from line L. (See left figure

[1]In [17], 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.



(i) Optimal solution (ii) Two level solution

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 travel costs (distances). (The distances are not drawn to scale.) We assume the view and travel cost are equally weighted in the objective function, i.e. the unit viewing cost (cost for each view) is the same as the unit travel cost (cost for unit travel 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.

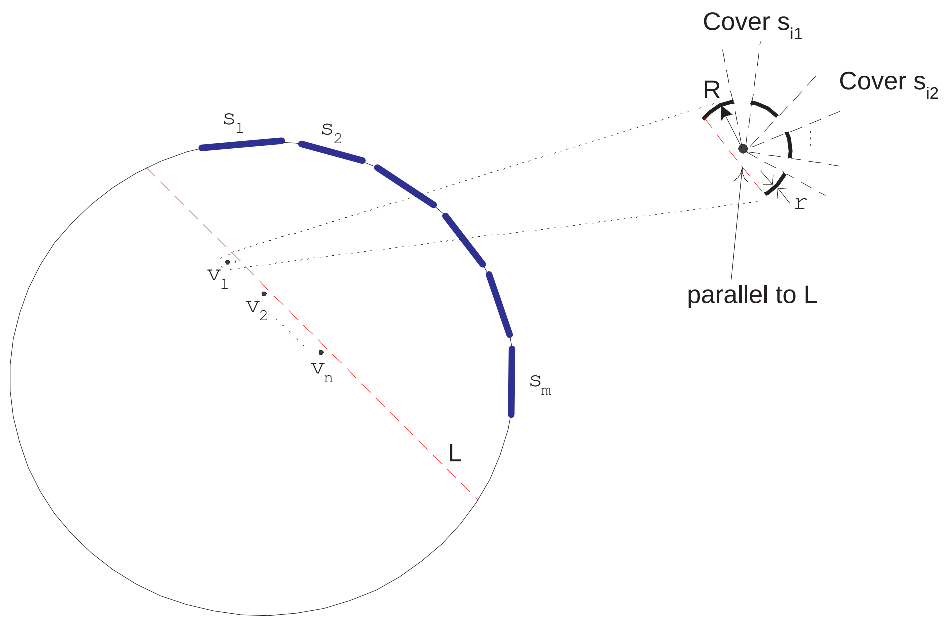


Figure 3: Reduction from an arbitrary SCP instance to a VPP instance. See the text for the reduction.

in Fig. 3.) The interior of the circle is free of obstacle other than those ones we construct around the viewpoints. We construct a half (cut by a line parallel to L) ring of obstacles of radius R and thickness r, with visibility openings, around each viewpoint, such that the viewpoint covers only the surface patches corresponding to those elements in the viewpoint's corresponding subset. For example, in the upper-right zoomed figure in Fig. 3, 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 $s_{i1}, s_{i2}, \ldots$. We further make R and r sufficiently small such that each viewpoint is only occluded by the ring of obstacles around itself, but not by obstacles around other viewpoints. (Since all the viewpoints are on the same line L, one viewpoint can only occlude the other from viewing the surface patch on L, and the surface patches we construct are not on L.) The resulting VPP instance is to ask a plan of 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.

On the other hand, assuming the planned viewpoints are given (or the VPP 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 [4]. (The shortest path graph is defined as the complete graph between the planned viewpoints in which the edge cost between two viewpoints is the shortest path length the robot travels 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. (Please see the appendix for a brief recapitulation of Metric TSP.)

Thus, in the abstract sense, Traveling VPP can also be termed as "Set Covering on a Graph". As an immediate result, Traveling VPP must be at least as hard as both SCP and Metric TSP.

1.3 Related work

In addition to the related works in the robotics and vision literature mentioned above, here we survey some of the related works in the combinatorial optimization literature, especially the problems of connected facility location, and errand scheduling. (Two other problems, group Steiner tree and traveling purchaser, are also related and will be discussed in the following sections.)

In [20], the problem of connected facility location 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 all the clients, such that the total costs, including both the summation of the service assignment costs and the tree cost, is minimized. Using the metric heuristics in their algorithm, the authors give a greedy algorithm with constant approximation ratio. By regarding the clients as the surface patches in the Traveling VPP and regarding 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 in [20] is not applicable to the Traveling VPP.

Given a graph $G = (V, E)$ with metrics (the edge weights of which satisfy the triangular inequality), where each vertex is associated with a subset of errands, the problem of errand scheduling asks for a short tour such that the subsets of the vertices visited is the whole errand set [19]. In [19], the author gives an algorithm with the approximation ratio of $3\rho/2$, where ρ is the maximum number of nodes one errand is associated.

Traveling VPP clearly generalizes ESP in the following senses: the graph in Traveling VPP does not assume metrics; there is no view cost in ESP; and there is no distinction in ESP of viewpoint and Steiner node on the graph. (In Traveling VPP, even some viewpoints are in the final tour planned, the robot does not need to take a view at them, and they are simply for travel use, hence termed as Steiner nodes [22].)

1.4 Integer program formulation of Traveling VPP

In this section, we give an integer linear program (ILP) formulation for the Traveling VPP.

In our unified formulation of travelling VPP, we make assumptions to abstract out and focus on certain key ingredients in the unified formulation, 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 literature can be utilized to derive these quantities. For instance, the set of viewpoints can be derived from the aspect graph [1], partial coverage between viewpoints and surface patches can be incorporated by subdividing the surface patch, sensor field of view constraints can be incorporated via viewpoint and surface patch visibility computations, and the encoding graph would essentially be a roadmap built on the configuration space of the robot [12, 13].

The 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 cost and the tree cost (the sum of all edge costs in the Steiner tree). The reason for using the (rooted) Steiner tree problem instead of Metric TSP is it is easier to combine the (rooted) Steiner tree formulation with covering constraints. Moreover, Metric TSP can be easily approximated using the solution to the Steiner tree problem. (See the appendix for details.)

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 "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 sensing action, 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 travel cost or cost per unit traveling distance, to allow users to specify the relative weights between sensing and traveling. Further, we use F to denote the view frequency, defined as the maximum number of viewpoints that cover a single surface patch, i.e., $F = \max_{j \in \mathcal{S}} |\mathcal{V}(j)|$. ($|\mathcal{A}|$ denotes the cardinality of a discrete set $\mathcal{A}$.)

In the ILP setting, binary integer variables are used to denote a solution. In the following, we define the binary variable, y_i, as the indicator whether to take a view at view cell 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:

<u>**Traveling VPP (ILP):**</u> (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 (implicitly specified by coverage constraints), $y_i = 1$, 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 [9], and essentially express the notion that each selected node must be reached 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 becomes handy when we analyze the algorithmic performance.

2 Hardness Analysis of Traveling VPP

Although as the generalization to both SCP and Metric TSP, Traveling VPP is immediately an NP-complete problem, its hardness of approximation, i.e., the best approximation ratio any polynomial algorithm can achieve. In this section, we use the result in [10] 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 ratio are $O(\log n)$ and $O(1)$ respectively. In [8, 2], 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. As an interesting robotic application, in [16], the authors use the approximation GST algorithm in [2] to solve the problem of planning tours for the robot arm to achieve at least one configuration from each distinct group of configurations that achieve the same end-effector pose.

It was open whether the gap between the best known SCP and GST approximation ratios could be closed until that recently, Halperin and Krauthgamer show the poly-log inapproximality hardness result for GST, i.e., the optimal solution to GST cannot be approximated by any polynomial algorithm within the $O(\log^{2-\epsilon} k)$ ratio, for any $\epsilon > 0$ [10]. In what remains in this section, we show that the Traveling VPP is not approximable within the same poly-log ratio via the reduction given in [8].

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 a harder problem than GST and cannot be approximated within $\log^{2-\epsilon}|\mathcal{S}|$. The question remains whether the hardness of Traveling VPP is of higher order. We show in the following that the inapproximalities of Traveling VPP and GST are of the same order by showing the reduction from Traveling VPP to GST via two steps.

We first show how to reduce a GST instance from a special case of a 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 exclusive. Consider any Traveling VPP instance, i.e., the viewpoint set $\mathcal{V} = \{v_i, i = 1, \dots, n\}$, the surface patch set $\mathcal{S} = \{s_j, j = 1, \dots, m\}$, the viewpoint set $\mathcal{V}(s_j) \subseteq \mathcal{V}$ for each surface patch s_j, and the graph G that connects $\mathcal{V}$. We first construct a group g_j for each surface patch s_j and construct a vertex for each pair of a surface patch s_j and one viewpoint from its viewpoint set, i.e., $(s_j, v_i),\ v_i \in \mathcal{V}(s_j), s_j \in \mathcal{S}$. 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., $\{(s_j, v_i) : s_j \in \mathcal{S}(v_i)\}$ (picking an arbitrary vertex (s_j, v_i) as the tree root) and then placing the tree root at the node v_i on G. Thus, we have a GST instance on the graph over vertices in the form (s_j, v_i) and groups corresponding to the surface patches s_j. And it is easy to see that an optimal GST solution that picks vertices (s_j, v_i) and Steiner tree between them correspond to an optimal solution to Traveling VPP of picking viewpoints v_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.

We now show how to reduce an arbitrary Traveling VPP instance to a Traveling VPP instance with 0 view cost. For an arbitrary Traveling VPP instance with unit view cost and unit travel cost w_v and w_p respectively, we add a new viewpoint $v_{i'}$, for each original viewpoint v_i, with identical surface patch set, make the surface patch set for v_i to be empty, and connect $v_{i'}$ to v_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 made empty, the new solution has to go to the new copy of the viewpoint, thus incurring the travel 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.

By cascading the two reductions 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 again F is the view frequency.

3 LP based Algorithms for Traveling VPP

In this section, we first give the LP relaxation for Traveling VPP, introduce a novel rounding algorithm to get an integral solution from the LP solution, give the approximation ratio analysis, and then discuss how to solve the LP. We also compare the approximation ratio attained by our algorithm and that by randomized rounding in [8] after the reduction from Traveling VPP to GST is done.

3.1 Relaxed LP 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 of the above LP relaxation 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*.

3.2 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 the ILP solution/cost that may or may not be optimal. The algorithm *Round and Connect* is listed as follows,

Algorithm 1 *Round and Connect: (take LP optimal y_i^*, z_e^* as input and output y_i', z_e')*

```
    Step 1. Initialize 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. 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., $\mathcal{V}' \leftarrow \mathcal{V}' \cup \{i_{max}\}$
    Step 3. 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., $\mathcal{V}^c \leftarrow \mathcal{V}^c \setminus \{i_{max}\}$
    Step 4. Stop and output $\mathcal{V}'$ if $\mathcal{S}^u$ is empty, i.e., set $y_i' = 1$ for $i \in \mathcal{V}'$, and set $y_i' = 0$ for $i \notin \mathcal{V}'$; otherwise, go to Step 2.
    Step 5. Connect $\mathcal{V}'$ by the optimal Steiner tree solution. Set $z_e'$ 1 for edges in the Steiner tree 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, Step 5. Note that the Steiner tree problem is again an NP-complete problem for a general graph. So practically speaking, we can use a constant ratio approximation algorithm, for example the one in [9], and incur an additional bounded performance degradation. It is easy to see that the rounding part of the above algorithm (up to Step 5) runs in polynomial time, $O(|\mathcal{V}||\mathcal{S}|)$.

3.3 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.

3.3.1 View cost analysis

In the following, we show that the LP optimal viewpoint assignments of the chosen viewpoints are lower bounded by $\frac{1}{F}$, Lemma 2. This follow immediately from a simple observation based on the feasibility of the LP optimal solution, Proposition 1. The results are then used in bounding the view cost part of the algorithmic solution, Corollary 3.

Proposition 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, Step 2, 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 Proposition 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 3.

Corollary 3 *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

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

■

3.3.2 Total cost analysis

In the following, after stating the half integrality gap result of the Steiner tree problem [22], 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 4 *For the Steiner tree problem, the integrality gap between the IP and its relaxed LP is* 2.

Proof. See Chapter 22 of [22]. ■

Note that Step 5 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 5 *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 Lemma 2 and the edge scaling, we have,

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

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* (Step 5) gives an optimal integral Steiner tree solution to connect $\mathcal{V}'$. By the integrality gap result for Steiner trees, Lemma 4, 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, (The last equality below is due to the edge scaling.)

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

Combined with the view cost part of the algorithmic solution, Corollary 3, we have,

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

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

3.3.3 Integrality gap for *Traveling VPP*

Theorem 5 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 6 *The integrality gap of the Traveling VPP is $2F$.*

Proof. We show the integrality gap of Traveling VPP is at least $2F$ by giving an example where the $2F$ ratio is achieved. In the Traveling VPP instance in Fig. 4, viewpoints $i_{\mathcal{C}_1,1}, i_{\mathcal{C}_1,2}, \ldots, i_{\mathcal{C}_1,F}, \ldots, i_{\mathcal{C}_n,1}, \ldots, i_{\mathcal{C}_n,F}$ are grouped into n clusters, denoted by $\mathcal{C}_1, \ldots, \mathcal{C}_n$ respectively. There are n surface patches. All the F viewpoints in a single cluster only view one surface patch, labeled by the cluster index, i.e., $\mathcal{V}(j) = \{i_{\mathcal{C}_j,k}, k = 1, \ldots, F\}, j \in \mathcal{S}$. There are two types of edges, e^1 and e^2, in the graph. Here we use superscripts 1 and 2 to denote the edge type. e^1 edges have the common edge cost of $\epsilon \ll 1$ and e^2 edges have the common cost of 1. e^1 edges in each cluster form a complete graph; and 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 of a single cluster. We further assume the view cost is negligible compared with traveling cost, $w_v \ll w_p$.

It is not difficult to see that the ILP optimal solution is to choose a single viewpoint from each cluster and construct a tour using $n-1$ e^2 edges, and the corresponding ILP optimal value is approximately $(n-1)\cdot 1$ (by neglecting view cost and e^1 edge costs). The LP optimal solution, however, is to assign

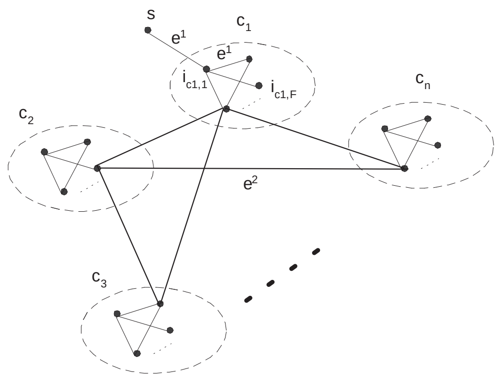


Figure 4: A Traveling VPP instance. There are $n \cdot F$ viewpoints, grouped into n clusters (circled by dashed curves). Each cluster contains exactly F viewpoints. For example, cluster $\mathcal{C}_1$ contains viewpoints $i_{\mathcal{C}_1,1}, \ldots, i_{\mathcal{C}_1,F}$. There are n surface patches (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.

$\frac{1}{F}$ 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 in 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 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 6, suggests that the $2F$ approximation algorithm *Round and Connect* is the best possible for the LP relaxation given above.

3.4 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, (5). We suggest two ways here: to adopt the column generation approach, [5], to practically solve the LP; or to use an alternative LP relaxation formulation. In the following, we first consider a special but important case, the *Traveling VPP on a Tree*.

3.4.1 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, in the following, we analyze the performance of a greedy algorithm based on amortized costs, and show via a counterexample that this algorithm can perform arbitrarily poorly on the *Traveling VPP on a Tree* (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 amortized cost based greedy algorithms have been shown to achieve the logarithmic approximation ratio (the best approximation ratio) for the SCP, it is not so for *Traveling VPP on a Tree* as we show in Fig. 5. In this example, 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, roughly $\frac{n-1}{n-1} = 1$ 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.

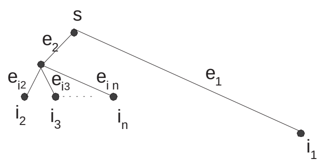


Figure 5: An instance of the Traveling VPP on a Tree. There are n viewpoints, labeled by $i_1, i_2, \ldots, i_n$, and $n-1$ number of surface patches (not drawn). Viewpoint i_1 covers all the surface patch. Each of viewpoints $i_2, \ldots, i_n$ covers only one surface patch, but all together they cover 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.

LP formulation for *Traveling VPP on a Tree*

The *Traveling VPP on a Tree* admits a polynomial sized relaxed LP formulation. Since LP is in P [14], we have a polynomial time and view frequency factor approximation algorithm 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, (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 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:

$$\underline{\textbf{Traveling VPP on a Tree (LP):}} \quad \min \quad w_v \sum_{i \in \mathcal{V}} y_i + w_p \sum_{e \in E} c_e z_e$$

$$\text{Subject to:} \quad \forall j \in \mathcal{S}, \qquad \sum_{i \in \mathcal{V}(j)} y_i \geq 1 \tag{6}$$

$$\forall i \in \mathcal{V}, \forall e \in E : e \in p_i, \qquad z_e \geq y_i \tag{7}$$

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

Since the covering constraints for the above formulation and for *Traveling VPP* are the same, we only need to show the equivalency of the connection constraints, (5) and (7). We show this equivalency by reductions from 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 $O(|\mathcal{S}| + |E||\mathcal{V}|)$, much smaller than the formulation for the general graph case.

3.4.2 Column generation technique applying to Traveling VPP

In this section, we give the dual program for Traveling VPP as follows, and derive the column generation rules for solving the dual program and use the complementary slackness conditions (CSCs) to get the solution.

a. Dual program and complementary slackness conditions for Traveling VPP

Corresponding to the constraints (4), we associate the dual variables, α_j, to each surface patch j; and corresponding to the constraints (5), we associate the dual variables, β_{iT}, to each pair of view node i and cut T that includes i but excludes s. The corresponding dual linear program (DLP) is given as:

$$\begin{array}{rll}
\underline{\textbf{DLP:}} & \max \sum_{j \in \mathcal{S}} \alpha_j & \\
\text{Subject to:} & & \\
\forall i \in \mathcal{V}: & \sum_{j \in \mathcal{S}(i)} \alpha_j - \sum_{T: i \in T} \beta_{iT} \leq w_v & (8) \\
\forall e \in E: & \sum_{i \in \mathcal{V}} \sum_{T: i \in T \wedge e \in \delta(T)} \beta_{iT} \leq w_p\, c_e & (9) \\
 & \alpha_j, \beta_{iT} \geq 0 &
\end{array}$$

As per the duality theorem, a feasible solution to the primal linear program (its dual) provides bounds on the optimal solution to the dual (primal) and achieve the same optimal value when the complementary slackness conditions (CSCs) are satisfied.

CSC of Traveling VPP:

Primal CSC:
$$y_i > 0 \Rightarrow \sum_{j \in \mathcal{S}(i)} \alpha_j = \sum_{T: i \in T} \beta_{iT} + w_v$$
$$z_e > 0 \Rightarrow \sum_{i \in \mathcal{V}} \sum_{T: i \in T \wedge e \in \delta(T)} \beta_{iT} = w_p c_e$$

Dual CSC:
$$\alpha_j > 0 \Rightarrow \sum_{i \in \mathcal{V}(j)} y_i = 1$$
$$\beta_{iT} > 0 \Rightarrow \sum_{e \in \delta(T)} z_e = y_i$$

In the column generation paradigm, we are working with the DLP, which contains a large number of variables, α_j associated with surface patches and β_{iT} associated with view point i and a cut T of view set $\mathcal{V}$ that contains i. The idea is simple and related to the Simplex method: we always work with a reduced version of the LP by only considering a small number of dual variables, corresponding to the basic variables in the Simplex algorithm for LP, and add columns/non-zero dual variables whenever the dual solution can be improved. From the primal program point of view, after solving the LP relaxation using only the basic varialbes, we can use the CSCs to get the corresponding primal variable solution; we check the infeasibility of the primal constraints and for infeasible constraint, add the corresponding dual variable (25). (Please see the appendix for a brief recap for the column generation method.)

b. Column generation algorithm

The infeasibility of the constraint (4) for a surface patch j, or the rule to add α_j, says that we should add α_j to the reduced LP if the sum (over its viewpoint set) of the (fractional) viewpoint assignments is less than 1. We can simply add the fractional assignments of the views that can see surface patch j and compare it with 1. So assuming we have the optimal primal solution to the reduced LP, this takes $O(|\mathcal{V}||\mathcal{S}|)$ time, where $|\mathcal{V}|$ and $|\mathcal{S}|$ are the numbers of surface patches and views respectively. (By some additional data structures, for example the links between surface patch and the views that see it, we can have even faster checking time.)

The rule for adding β_{iT} for a specify viewpoint i asks to check for all the cuts whether there exists one cut T such that the number of edge crossing

T is less than the viewpoint assignment y_i. In the following, we show this corresponds to the classic min-cut problem.

Proposition 7 *The column generation rule for adding the dual variable β_{iT} for a viewpoint and cut pair, for a specific viewpoint i is equivalent to requiring the minimum value of such summation for all possible cuts T to be strictly less than the viewpoint assignment, i.e.,*

$$\min_{T \subset \mathcal{V}: i \in T \wedge s \notin T} \sum_{e \in \delta(T)} z_e < y_i \tag{10}$$

Proof. First, if $\min_{T \subset \mathcal{V}: i \in T \wedge s \notin T} \sum_{e \in \delta(T)} z_e < y_i$ and supposing T' is such a minimizer cut, i.e.,

$$\beta_{iT'} < y_i, \quad T' = \operatorname*{arg\,min}_{T \subset \mathcal{V}: i \in T \wedge s \notin T} \sum_{e \in \delta(T)} z_e,$$

the primal constraint corresponding to the dual variable $\beta_{iT'}$ is not feasible and $\beta_{iT'}$ should be added.

Second, for any dual variable β_{iT} if the primal constraint is infeasible, i.e., $\sum_{e \in \delta(T)} z_e < y_i$, by fixing viewpoint i, we have

$$\min_{T' \subset \mathcal{V}: i \in T' \wedge s \notin T'} \sum_{e \in \delta(T')} z_e \leq \sum_{e \in \delta(T)} z_e < y_i$$

■

By recalling the definition of the min-cut problem as finding the $s-t$ cut (i.e., the cut or partition of the graph vertices separating the vertices s and t on the graph) with the minimum cut capacity (defined as the sum of the capacities of the edges crossing the cut), the subproblem of adding β_{iT} for a specific i is equivalent to the min-cut problem.

Lemma 8 *The subproblem of adding β_{iT} for a specific i is equivalent to min-cut problem by assigning the edge assignments z_e as the capacities for the edges.*

Proof. The proof proceeds by constructing the min-cut problem from the primal solution z_e. By assignment the edge capacities c_e as the edge assignments z_e for all the edges, we have a min-cut problem instance,

<u>Min-Cut Problem</u> Minimize $\sum_{e \in \delta(T)} z_e, \quad \forall T \subset \mathcal{V} : i \in T \wedge s \notin T$

The solution of the above min-cut problem is clearly $\min\limits_{T \subset \mathcal{V}: i \in T \wedge s \notin T} \sum\limits_{e \in \delta(T)} z_e$. So according to the subproblem formulation of adding dual variable β_{iT} for a specific i, (10), we can decide whether to add such dual variable βiT by comparing the optimal solution with y_i. ■

By identifying the second type of subproblem, adding β_{iT}, to be the min-cut problem, we can apply efficient algorithms for solving min-cut problems. Specifically, according to the duality between min-cut and max-flow problems [14], the existing efficient max-flow algorithms can be readily applied.

3.4.3 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 relaxed LP and solving for the optimal solution. In the following, we show how to use flow formulation (rather than the cut) for this type 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 $G = (\mathcal{V}, E)$. We denote the start and end viewpoints of an edge e by $start(e)$ and $end(e)$ respectively. 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 the y_i amount of commodity i to be flowed 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 the terminal of that flow, i.e., $\sum_{e \in \mathcal{I}(i')} f_{i,e} = \sum_{e \in \mathcal{O}(i')} f_{i,e}, \forall i \neq i' \in \mathcal{V}$. So the overall relaxed LP is given as:

Traveling VPP (ILP using flows): (11)

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

$$\text{Subject to: } \forall j \in \mathcal{S}\ , \quad \sum_{i \in \mathcal{V}(j)} y_i \geq 1 \tag{12}$$

$$\forall i \in \mathcal{V} \qquad \sum_{e \in \mathcal{O}(i)} f_{i,e} \geq y_i \tag{13}$$

$$\forall i \in \mathcal{V} \qquad \sum_{e \in \mathcal{I}(s)} f_{i,e} \geq y_i \tag{14}$$

$$\forall i, \forall i' \neq i, \qquad \sum_{e \in \mathcal{I}(i')} f_{i,e} = \sum_{e \in \mathcal{O}(i')} f_{i,e} \tag{15}$$

$$\forall e \in E, \forall i \in \mathcal{V}, \qquad z_e \geq f_{i,e} \tag{16}$$

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

In the following, we show the equivalence of the above flow-based formulation (11) to the one we give before (1) by showing the feasible solution to one formulation corresponds to a feasible solution to the other.

Lemma 9 *The flow based relaxed LP formulation for Traveling VPP (11) 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 (11), 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, (13). Since the edge assignment is lower bounded by the flow going through it, (16), 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[2], 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 (11). ■

Note that the size of the formulation (11) has $|\mathcal{V}|+|E|$ number of variables and $|\mathcal{S}| + O(|\mathcal{V}||E|)$ constraints.

Simulation result on counterexample Fig. 2 We formulated the relaxed LP corresponding to the counterexample given in the introduction, Fig. 2, and used the Matlab optimization toolbox, specifically the linprog()

[2]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 reduce the overall cost, some edges while maintaining connectivity. It follows there exists a unique path from any tree node to the root.

function, for our simulation [?]. [3] Albeit simple, it showcases the power of the LP formulation for *Traveling VPP on a Tree*: the solutions to the LP make close to optimal decisions (within the approximation ratio of F) according to different preferences between traveling and viewing (different choices for w_v and w_p).

We associate decision variables $y_i, i = 1, 2, 3$ to viewpoints $v_i, i = 1, 2, 3$ respectively, and associate $z_k, k = 1, 2, 3$ to edges $e_k, k = 1, 2, 3$. (Viewpoint v_4 coinciding with the robot start s is already in the solution and no decision needs to be made for it.) The resulting LP formulation can be put in the standard form:

$$\begin{array}{ll}
\min & \vec{c}^{\,T}\vec{x} \\
\text{s. t.} & A\vec{x} \le \vec{b};\ \text{and}\ \vec{x} \ge \vec{0}; \\
\text{where} & \vec{c} = \begin{pmatrix} w_v \\ w_v \\ w_v \\ w_p \cdot 1 \\ w_p \cdot 100 \\ w_p \cdot 1 \end{pmatrix} \quad A = \begin{pmatrix} -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{pmatrix} \\
 & \vec{x} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} \quad \vec{b} = \begin{pmatrix} -1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
\end{array}$$

In the above formulation, the first two rows of $\vec{A}$ and $\vec{b}$ correspond to the covering constraints. For example, the first low, $-y_1 - y_2 \le -1$, requires that for surface patch s_1, we must choose at least one of the viewpoints that cover it, v_1 and v_2. The remaining rows of $\vec{A}$ and $\vec{b}$ correspond to the connection constraints. For example, the third row, $y_1 - z_1 \le 0$, requires that for viewpoint v_1 and cut T_{e_1}, we must choose the only crossing edge e_1, if v_1 is chosen.

The results are listed in Table 1 for different w_v and w_p. The first row in Table 1, $w_v = w_p = 1$, corresponds to the precise case of Fig. 2, same (unit) costs for traveling and viewing. The LP solution (in this case already integral) gives the optimal decision. The last row, $w_v = 1000, w_p = 1$,

[3] We tried both Simplex and Interior Points as the LP solving algorithms. Both converge to the optimal solution really fast: Simplex took around 5 iterations, and Interior Point took around 8 iterations.

w_v, w_p	y_1	y_2	y_3	z_1	z_2	z_3	cost
1,1	1	0	1	1	0	1	4
$\vdots$							
$97, 1$	1	0	1	1	0	1	196
$99, 1$	0.5	0.5	0.5	0.5	0.5	0.5	199.5
$\ldots$							
$101, 1$	0	1	0	1	1	0	202
$\vdots$							
$1000, 1$	0	1	0	1	1	0	1101

Table 1: Simulation results on example in Fig. 2: the LP solutions.

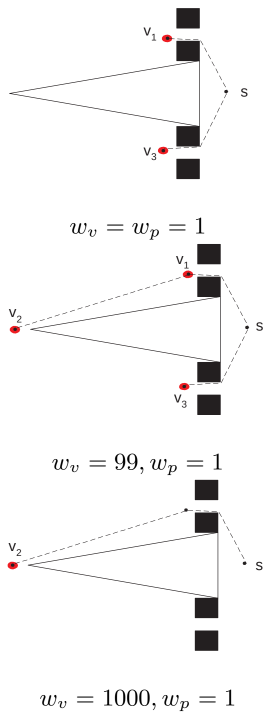


Figure 6: Corresponding *Traveling VPP* solutions. (It is the worst possible solution given by the algorithm *Round and Connect* for $w_v = 99, w_p = 1$.)

corresponds to the case where the view cost is much higher than traveling cost. In this case, the LP solution (again integral), taking viewpoint v_2 that covers more surface patches, again gives the correct decision. The middle row is an interesting case, where the two plans, to take viewpoint v_2 and traveling e_1 and e_2, and to take viewpoints v_1 and v_3 and traveling e_1 and e_3 have exactly the same cost of 200. However, the LP solution makes the choice of distributing equally the task among the two plans. Since the edge assignments (specifically e_1 in this case) need only satisfy the largest assignment of viewpoints connected (y_1 and y_2), half of the edge cost of e_1 is saved in the LP solution. Note also that when applying the algorithm *Round and Connect* for this case, it may choose all the viewpoints and edges and the corresponding total cost is $99 * 3 + 102 = 399$, roughly twice the optimal value.

3.5 Randomized rounding algorithm

By the reduction mentioned above, 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 [8] to achieve a $O(\log|\mathcal{V}|\log\log|\mathcal{V}|\log|\mathcal{S}|\log F)$, a poly-log, approximation ratio. (See also the appendix for an alternative LP formulation for Traveling VPP and the corresponding direct application of the randomized rounding algorithm that achieves the poly-log approximation ratio.)

In conclusion, using both the LP based algorithms, deterministic and randomized 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.

4 Applications of Algorithms 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 polylogarithmic to the minimum of poly-log and the frequency.

4.1 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 the first LP based rounding algorithm gives us a frequency approximation ratio. Thus by applying the algorithm Round and Connect above and LP randomized rounding in [8], the optimal solution to GST is approximated within the ratio of $\min(O(F), O(\log|V|\log\log|V|\log k\log N))$.

4.2 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 [15]. The Traveling VPP is a special case of TPP where the prices are uniform for all the product and warehouse pairs. In [15], the authors give a poly-log approximation ratio for TPP.

Now we 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 the frequency bound approximation ratio still holds for weighted set covering, the rounding algorithm above still has the frequency bound approximation ratio. The reduction is done by "duplicating" a warehouse according to the number of different product prices it offers and connecting them via 0-cost edges.

To make it formal, 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 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 increase 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.

5 Special Cases of Traveling VPP

In the following, we briefly discuss several generalizations of Traveling VPP for more realistic models.

5.1 Image registration constraints

Image registration constraints refer to any two viewpoints planned should have enough overlap in the surface patch sets they cover. This is a crucial requirement in the robot vision application. Here we model it as a set multicover problem, 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 for the intersection of the overlapping surface patches. By requiring 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 registration constraints.

5.2 Metric Traveling VPP with visibility range

Now we investigate a restricted version of Traveling VPP where a point robot is equipped with a range sensor with limited range, i.e., a viewpoint can cover certain surface patches only if they are within a certain visibility range. We call this special case the *Metric Traveling VPP with Visibility Range*. These constraints are motivated from the vision application where range sensors have certain physical limitations. Note that for this special case, the $O(\log n)$ approximation ratio is a lower bound for its hardness of approximation, since one easily construct such a reduction from the SCP as in Fig. 3.

We first give some additional notations for the Metric Traveling VPP with Visibility Range. We denote the environment where both robot and surface patches reside by $\mathcal{P}$. We assume $\mathcal{P}$ is either the two-dimensional or the three-dimensional Euclidean space populated by obstacles, denoted by $\mathcal{P}^2$ and $\mathcal{P}^3$ respectively. For any two entities x_1 and x_2, we denote the Euclidean distance between them by $\|x_1, x_2\|_{\mathcal{P}}$. For robot traveling, we again use a complete graph $G = (\mathcal{V}, E)$ with metrics where the edge distance $c_e, e =< i_1, i_2 >$ between two viewpoints $i_1, i_2 \in \mathcal{V}$ is the shortest distance the robot traveling. Note that the edge cost is lower bounded by the Euclidean distance, i.e., $\forall e =< i_1, i_2 >$: $c_e \geq \|i_1, i_2\|_{\mathcal{P}}$. We denote the visibility range by D. By definition, the necessary condition for a viewpoint i to cover a surface patch j is that the distance between them is upper bounded by D. With a slight abuse of notation, we use $\|i, j\|_{\mathcal{P}}$ to denote this distance. Rigorously, for a surface patch j that occupies a region $\mathcal{R}(j)$ (of co-dimension 1) in $\mathcal{P}$, the distance $\|i, j\|_{\mathcal{P}}$ is the upper bound on the distances between any point belonging to surface patch j and the viewpoint i, i.e., $\|i, j\|_{\mathcal{P}} = \sup_{x \in \mathcal{R}(j)} \|x, i\|_{\mathcal{P}}$. Again, we use w_v, the unit view cost or cost per viewpoint, and w_p, the unit traveling cost or cost per unit traveling distance, to allow users to specify the relative weights between sensing and traveling.

With the above notations and settings, the *Metric Traveling VPP with Visibility Range* is formulated as to plan a subset of the viewpoints, denoted by $\mathcal{V}'$, and a traveling path connecting them on G, denoted by E', such that the total cost, $w_v|\mathcal{V}'| + w_p \sum_{e \in E'} c_e$, is minimum. ($|\mathcal{A}|$ denotes the cardinality of set $\mathcal{A}$.)

5.2.1 A Two-level Decoupled Algorithm for *Metric Traveling VPP with Visibility Range*

In this section, we give a very simple algorithm for *Metric Traveling VPP with Visibility Range*. It solves the problem in two steps. At the first level, it solves the VPP or SCP part of the *Metric Traveling VPP with Visibility Range* greedily, i.e., iteratively picks viewpoint that cover the most *uncovered* surface patches until all surface patches are covered. At the second level, it solves the Metric TSP to connect these picked viewpoints.

5.2.2 Algorithm Analysis

In this section, we analyze the approximation ratio of the two level algorithm presented. We present an alternative algorithm whose performance is no better than the algorithm presented before and then analyze the approximation ratio of this alternative algorithm, which also serves as the approximation ratio of the two level greedy algorithm. In the following, we first give a few definitions to clarify the algorithm and the consequent analysis.

Alternative algorithm: We call $2D$, the double of the visibility range, the *virtual range*. Intuitively, this is the range to locate the viewpoints with related covering functionalities, since for any surface patch j that viewpoint i covers, the other viewpoints covering it must lie within the *virtual range* of i according to the *visibility range*. For viewpoint i, we define the free region within its *virtual range*, i.e., where i can reach using the shortest path of length less than $2D$, the *virtual domain* of i. We call two viewpoints *dependent* if their virtual domains intersect; or *independent* if the virtual domains do not share common part. Note that if two viewpoints are independent, they cannot both cover any surface patch, since otherwise by traveling via the surface patch they would have a shortest path less than the virtual range.

In the following, we give the pseudo codes of the alternative algorithm.

Algorithm 2 *Algorithm for* Metric Traveling VPP with Visibility Range

```
    Step 1. Solve the SCP greedily:
        Iteratively choose the viewpoint that covers the most uncovered surface patches until all surface patches are covered.
        Output the chosen viewpoint set $\mathcal{V}'$.
    Step 2. Choose independent viewpoints:
        Iteratively choose a viewpoint from $\mathcal{V}'$ that is independent from already chosen viewpoints in this step.
        Output the chosen viewpoint set $\mathcal{V}''$.
```

Step 3. Solve the Metric TSP to connect $\mathcal{V}''$

Step 4. Connect the viewpoints in $\mathcal{V}' \setminus \mathcal{V}''$ to its nearest neighbor in $\mathcal{V}''$ using the shortest path.

Note that the above solution takes $|\mathcal{V}'|$ number of viewpoints, same as in the simple greedy algorithm, and constrains that the solution tour is to go to $\mathcal{V}''$ first and then to detour to $\mathcal{V}' \setminus \mathcal{V}''$. So the solution cost is clearly no better than the simple greedy algorithm presented in the previous section. We also call the viewpoints in the set $\mathcal{V}''$ *centers* (since they act like the cluster center of the viewpoints in $\mathcal{V}'$) and thus $\mathcal{V}''$ the *center set*.

Analysis: In the following, we first show that the optimal solution has to travel to every virtual domain of $\mathcal{V}''$. We then lower bound the optimal solution cost and upper bound the algorithmic solution cost. Combining both bounds gives us the algorithmic approximation ratio. We use OPT_{SCP} to denote the optimal SCP solution cost to cover all the surface patches, i.e.,

$$OPT_{SCP} = \min_{\mathcal{U} \subseteq \mathcal{V}:\ \underset{i \in \mathcal{U}}{\cup} \mathcal{S}(i) = \mathcal{S}} |\mathcal{U}|.$$

We call the tour that connect virtual domains of $\mathcal{V}''$ a *domain tour*, i.e., a tour to connect at least one viewpoint lying in each of the virtual domains of $\mathcal{V}''$. We use OPT_{Tour} to denote the optimal domain tour distance. We use OPT_{TVPP} to denote the optimal solution cost to *Metric Traveling VPP with Visibility Range*. We denote the algorithmic solution cost by $cost'$, the view cost and traveling parts of which are denoted by $cost'_{view}$ and $cost'_{travel}$ respectively.

Lower bound on OPT_{TVPP}: In this section, we give a lower bound on optimal solution cost to *Metric Traveling VPP with Visibility Range.* This lower bound is based on the simple observation that any feasible solution has to choose at least one viewpoint in the virtual domain of every viewpoint in $\mathcal{V}''$. This implies that the tour in the solution has to visit every virtual domain.

Proposition 10 *Any feasible solution has to choose at least one viewpoint from every virtual domain of $\mathcal{V}''$.*

Proof. For any viewpoint $i \in \mathcal{V}''$, we arbitrarily pick one surface patch from its surface patch set $j \in \mathcal{S}(i)$. Note that due to visibility range constraints, any viewpoint i' of its viewpoint set, i.e., $\forall i' \in \mathcal{V}(j)$, has to lie in the virtual domain of i. We show this by the metric (triangular) constraints, $\|i, i'\| \leq \|j, i\| + \|j, i'\| \leq D + D = 2D$. ■

As the immediate consequence, we have that the traveling cost of the optimal solution to *Metric Traveling VPP with Visibility Range* is the optimal tour cost to visit every virtual domain of the centers. So we have the following,

Lemma 11 *The optimal cost to* Metric Traveling VPP with Visibility Range *is at least the sum of the minimum view cost (ignoring traveling) and the optimal tour cost to visit every virtual domain of the centers, i.e.,*

$$OPT_{TVPP} \geq w_v OPT_{SCP} + w_p OPT_{Tour} \tag{17}$$

Upper bound on algorithmic solution cost: Note that we use the greedy SCP algorithm to get $\mathcal{V}'$. Assume the best approximation ratio for the SCP is A. Note that $A = \min(F, \log |\mathcal{S}|)$, where F is the frequency bound on the SCP, i.e., the maximum number of viewpoints that covers a single surface patch, $F = \max_{j \in \mathcal{S}} |\mathcal{V}(j)|$. We have [22]

$$|\mathcal{V}'| \leq OPT_{SCP} \cdot A.$$

So the view cost of the algorithmic solution is upper bounded, i.e.,

$$cost'_{view} \leq w_v \cdot OPT_{SCP} \cdot A \tag{18}$$

In analyzing the traveling cost of the algorithmic solution, note that we can easily get a tour to connect the centers by traveling first on the optimal domain tour and then detouring to the centers if needed. So the solution to Step 3. above is at most $OPT_{Tour} + |\mathcal{V}''| \cdot 4D$. By (if needed) again detouring from the centers to the remaining viewpoints $\mathcal{V}' \setminus \mathcal{V}''$, incurring for each such viewpoint a detour cost of $8D$, we have a lower bound on the travel cost of the algorithmic solution, i.e.,

$$cost'_{travel} \leq w_p[\, OPT_{Tour} + 4D|\mathcal{V}''| + 8D(|\mathcal{V}'| - |\mathcal{V}''|)\,] \leq w_p[\, OPT_{Tour} + 8D|\mathcal{V}'|\,]. \tag{19}$$

In conclusion, by both Eqs. (18,19), the total cost of the algorithmic solution is upper bounded, i.e.,

$$cost' \leq (w_v \cdot OPT_{SCP})A + (w_p \cdot OPT_{Tour}) + w_p \cdot 8D \cdot OPT_{SCP} \cdot A, \tag{20}$$

Total cost analysis: Combining the lower bound of the optimal solution cost and the upper bound of the algorithmic solution cost, we have the approximation ratio of the alternative algorithm is $(1+\frac{w_p}{w_v}8D)A$, where again $A=\min(F,\log|\mathcal{S}|)$.

Again, this alternative algorithm chooses the same number of viewpoints as the two-level algorithm and uses the optimal domain tour plus detours, first to the centers and then to the remaining viewpoints $\mathcal{V}''$, rather than the optimal tour between $\mathcal{V}''$ chosen by the two-level algorithm. So the cost of solution by the two-level algorithm is at most that of the alternative algorithm. And we conclude that the approximation ratio for the two-level algorithm is also $(1+\frac{w_p}{w_v}8D)A$.

Note that the above approximation ratio showed above nicely factors the tradeoff between view and travel: on the one hand, if the view cost is dominant, $w_v \gg w_p$, the problem is dominated by the associated view planning part and the approximation ratio is that of the view planning problem; on the other hand, if the range D is small, the problem is reduced to simply the traveling part, since A in this case is approximately 1 (the viewpoint has to be at the same location as the surface patch in order to cover it, which is only possible solution).

5.3 View travel costs tradeoff

We now consider the restricted version where viewpoints can be clustered, i.e., they can be grouped such that the union of surface patches covered by different groups are exclusive from each other. If the viewpoint distances within each group are upper bounded by D, the approximation ratio is $\frac{w_pD}{w_v}\log|\mathcal{S}|$ if we use the greedy algorithm for set covering. The reason is that for any Steiner tree connection the number of edges chosen is upper bounded by the number of viewpoints chosen minus one. Thus the travel cost is upper bounded by the optimal view cost times $\frac{w_pD}{w_v}$. And the overall cost, as the sum of the view and travel costs, is bounded by $\frac{w_pD}{w_v}\log|\mathcal{S}|$.

6 Conclusion and Future Work

In this report, we cover the problem of view planning with travel costs, Traveling VPP. We formulate the problem for the model-based case where the geometry of the object to inspect is known. By reduction from the set covering problem to the 2D VPP, we show the Traveling VPP is a combination of set covering and Metric TSP. As a problem harder than GST, Traveling VPP is poly-log inapproximable. We show our LP-based deterministic rounding algorithm 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 constant times frequency and a poly-logarithmic function of the input size. This parallels the approximation ratio result for set covering problem. We then discuss several special cases of Traveling VPP and give some weak approximation ratio results using the set covering heuristics.

In the future, we would like to generalize our model based 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. 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.

7 Appendix

7.1 LP-based method for solving IP

For integer linear programs, it is generally convenient to relax the integral variables to be fractional, solve the corresponding relaxed LP, and recover the integral solution from the fractional solution. Integral solution recovery can be done for example by (randomly) rounding the fractional LP solution to integers, called LP rounding. This LP-based approach plays a central role in the design of approximation algorithms. See [22] for a detailed treatment of this topic. The intuitive idea is that the relaxed LP corresponds to a constrained continuous optimization that can be solved efficiently. The integral solution can be relatively easily recovered without much performance sacrifice, if the original IP optimal solution is not far from this relaxed LP solution. This is characterized by the important concept of integrality gap, defined as the supremum of the ratio between optimal value to the ILP and to its LP-relaxation. It is generally believed that any LP-based approximation algorithms will have at least the integrality gap as the approximation ratio. Also, in analyzing LP relaxation based approximation algorithms, the optimal value of the LP relaxation is used as a lower bound on the IP optimal value. For example, if a rounding algorithm is shown to increase within a certain ratio the fractional LP optimal solution to get the integral solution, this ratio is an upper bound on the approximation ratio. This is exactly what we did for the *Round and Connect* algorithm analysis.

7.2 Column generation method

As a quick recap, let us consider the following generic LP in its matrix form. (All the matrices and vectors are of suitable dimensions.)

$$\begin{aligned} \text{Generic LP:} \quad & \max\ c^T x \\ \text{Subject to} \quad & Ax = b \end{aligned} \tag{21}$$

By decomposing x into basic and nonbasic variables, denoted by x_B and x_N respectively, $x = \begin{bmatrix} x_B \\ x_N \end{bmatrix}$, (Remember that basic variables assume nonzero values and nonbasic variables have to be zero.) the above LP can be represented by:

$$\begin{aligned} \text{Generic LP:} \quad & \max\ c_B^T x_B + c_N^T x_N \\ \text{Subject to} \quad & A_B x_B + A_N x_N = b \\ & x_B, x_N \geq 0 \end{aligned} \tag{22}$$

In the above, we decompose the constraint matrix A into two blocks of columns $A = [A_B, A_N]$ according to the corresponding basic and nonbasic variables.

The Simplex method first finds the optimal solution to a reduced problem described by only basic variables, and then adds one or several variables of x_N if the corresponding coefficient is positive (Remember we are working with a maximization problem.), i.e.,

$$c^T x = c_B^T A_B^{-1} b + (c_N^T - c_B^T A_B^{-1} A_N) x_N.$$

For $x_j \in x_N$, if the corresponding coefficient, $c_j - c_B A_B^{-1} A_j$ is positive, x_j will be included in the basic variables. (After solving the reduced LP, zero valued basic variables will be dropped out of basic variables and become nonbasic.) So the rule to add columns or new variables is

$$c_B^T A_B^{-1} A_j < c_j \tag{23}$$

To get a better understanding of the above column generation rule, let us take a look at the DLP to the above given LP (using dual variables y)

$$\begin{aligned} \text{DLP:} \quad & \min y^T b \\ \text{Subject to} \quad & y^T (A_B | A_N) \geq c^T \\ & y^T \geq 0 \end{aligned} \tag{24}$$

Using the complementary slackness conditions (CSCs), since the basic primal variable x_B are strictly positive, the corresponding dual variables satisfy

$$y^T A_B = c_B^T$$

So the column generation rule, after solving the CSCs, is also given by:

$$y^T A_j < c_j \tag{25}$$

Equivalently, the infeasibility of the dual program is checked.

7.3 An alternative relaxed LP formulation for Traveling VPP and the randomized rounding algorithm

Now we define a commodity j associated with each surface patch j. By connecting each surface patch j to all the viewpoints in its viewpoint set $\mathcal{V}(j)$ by directed edges (surface patch as the start of the edge and the viewpoint as the end), and by doubling the graph edges to make them directed, we have a digraph $G' = (V', E')$ with $V' = \mathcal{V} \cup \mathcal{S}$ and $E' = E \cup \{< j, i >: j \in \mathcal{S}, i \in \mathcal{V}(j)\}$. The number of nodes in the graph is $|\mathcal{V}| + |\mathcal{S}|$ and the number of edges is $O(|\mathcal{V}||\mathcal{S}|)$. We then define each surface patch as a source to flow its commodity to the common terminal s, the robot start position. We then define the commodity flow variables f_{ie} for commodity j for the pairs of surface patch j and those edges e either in the original graph or between j and its viewpoint set members. Similarly, for a node in the graph G', we define $\mathcal{I}(v)$ as the set of edges having v as their endpoint and let $\mathcal{O}(v)$ denote the set of edges having i as their start-point. Now v can be either a surface patch, for which there are only outgoing edges, or a viewpoint.

The ILP is given as follows:

$$\begin{aligned}
\min \quad & w_v \sum_{i \in \mathcal{V}} y_i + w_p \sum_{e \in E} c_e z_e \\
\text{s.t. } \forall j \in \mathcal{S} \text{ ,} \quad & \sum_{e \in \mathcal{O}(j)} f_{j,e} \geq 1 && (26) \\
\forall j \in \mathcal{S} \text{ ,} \quad & \sum_{e \in \mathcal{I}(s)} f_{j,e} \geq 1 && (27) \\
\forall j \in \mathcal{S}, \forall i \in \mathcal{V}, \quad & \sum_{e \in \mathcal{I}(i)} f_{j,e} = \sum_{e \in \mathcal{O}(i)} f_{j,e} && (28) \\
\forall e \in E, \forall j \in \mathcal{S}, \quad & z_e \geq f_{j,e} && (29)
\end{aligned}$$

$$\forall i \in \mathcal{V}, \forall j \in \mathcal{S}, \quad y_i \geq f_{j,<j,i>} \tag{30}$$
$$y_i,\ f_{j,e},\ f_{j,<j,i>},\ z_e \in \{0,1\},$$
$$i \in \mathcal{V}, j \in \mathcal{S}, e \in E, <j,i>: j \in \mathcal{S}, i \in \mathcal{S}(j)$$

It is not difficult to see the equivalence of our Traveling VPP IP to the above IP. On the one hand, first, for any feasible IP solution to the above flow formulation, for any surface patch j, there exist an unity flow $f_{j,e}$ for an edge e connecting j to one viewpoint in its viewpoint set. By choosing the viewpoint e connecting in the Traveling VPP solution, the surface patch covering constraints are all satisfied. Second, for any viewpoint thus chosen, let us follow the flow $f_{j,e}$, by flow conservation law, there must exist a path connecting j, the source of the commodity flow j, to s the terminal of all commodity flows. By picking all such paths we have a valid tree connection between chosen viewpoints. On the other hand, for any feasible IP solution to Traveling VPP, i.e., a set of viewpoints that cover all the surface patches and a tree connection connecting the viewpoints, for a surface patch j, we can arbitrarily pick a chosen viewpoint i that covers it and assign a unity commodity flow $f_{j,<j,i>} = 1$; then we simply follow the tree connection from i to s, $path(i,s)$ and assign unity flows for all the edges directing to s on the path, i.e., $f_{j,e} = 1, e \in path(i,s)$. This way, we have a feasible solution to the above flow formulation. In conclusion, the above IP formulation based on flows is equivalent to the Traveling VPP.

By relaxing the above viewpoint edge assignment variables and commodity flow variables to be positive reals, we have the relaxed LP formulation.

In the following, we use the same rounding technique as in [8] and achieve a poly-log approximation ratio.

7.3.1 Randomized rounding algorithm and performance analysis

First assume the Traveling VPP instance is for tree cases, and denote the parent edge of an edge e by $p(e)$, the path, i.e., union of edges, from edge e to s including e by $path(e)$. (Again, if it is a general graph, we embed it into a tree and have an additional $O(\log|\mathcal{V}|)$ loss in the approximation ratio [6].) We denote the LP optimal solution to the above formulation by $f^*_{i,e}, z^*_e, y^*_i$ and the corresponding LP optimal cost by OPT^*. We round the edge assignment to 1 using probability $\frac{z^*_e}{z^*_{p(e)}}$. Those edges that do not have parent edges, i.e., connect directly to the root s are rounded to 1 with probability of z^*_e. We round the viewpoint assignment to 1 with probability $\frac{y^*_i}{z_{e(i)}}$, where $e(i)$ is the edge incident on i that leads to the tree root s. We then choose only those edges that are connected to s by chosen edges, i.e., the chosen edges form a connected component that includes s, and we choose those chosen viewpoints that is connected to s via the chosen edges. We then choose the viewpoint i such that $f^*_{j,<j,i>} = z^*_e, e \in \mathcal{O}(i)$. We discard those chosen edges and viewpoints that are not connected to s via chosen edges. So the probability of an edge e chosen in the solution is $\prod_{e' \in path(e)} \frac{z_e *}{z^*_{p(e)}} = z^*_e$. Similarly the probability of a viewpoint i chosen in the solution is y^*_i. And the expected cost of the solution is $\sum_{i \in \mathcal{V}} w_v y^*_i + w_p \sum_{e \in E} c_e z^*_e = OPT^*$.

Using the same technique as in [8], one can show that the by running $O(\log F \log |\mathcal{S}|)$ rounding iteration, all the surface patches are covered by some viewpoints connected via the constructed tree to s. Rather than repeating, we refer [8] for the detailed analysis. This implies the approximation ratio for this randomized algorithm is $O(\log F \log |\mathcal{S}|)$ and $O(\log F \log |\mathcal{S}| \log |\mathcal{V}|)$ for graphs.

References

[1] 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.

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

[3] W. Chin and S. Ntafos. Optimum watchman routes. In *Proc. of Annual Symposium on Compuational Geometry*, 1986.

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

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

[6] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proc. of STOC 2003*, California, USA.

[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] 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.

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

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

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

[12] 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.

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

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

[15] 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.

[16] M Saha, T. Roughgarden, J. Latombe, and G. Sánchez-Ante. Planning tours of robotic arms among partitioned goals. *International Journal of Robotics Research.* 25(3) March 2006, pp. 207-224.

[17] 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.

[18] 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.

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

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

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

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