2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

View Planning Problem with Combined View and Traveling Cost

Pengpeng Wang*, Ramesh Krishnamurti†, and Kamal Gupta*
* *RAMP Lab, School of Engineering Science* † *School of Computing Science*
Simon Fraser University, Canada
{*pwangf, ramesh, kamal*}*@cs.sfu.ca*

***Abstract*— In this paper, we introduce the problem of view planning with combined view and traveling 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 in a known workspace. The cost to minimize is a combination of the view cost, proportional to the number of viewpoints planned, and the traveling cost for the robot to realize them. First, we formulate Traveling VPP as an integer linear program (ILP). The focus of this paper is to design an approximation algorithm that guarantees worst-case performance (w.r.t. the optimal solution cost). We propose a linear program based rounding algorithm that achieves an approximation ratio of the order of view frequency, defined to be the maximum number of viewpoints that see a single surface patch of the object. Together with the result we showed in [22], the best approximation ratio for Traveling VPP is either the order of view frequency or a poly-log function of the input size, whichever is smaller. Motivated from the robot motion planning techniques, where the graph built for robot traveling is a tree, we then consider the corresponding special case of Traveling VPP, and give a polynomial sized LP formulation. We conclude with a discussion of realistic issues and constraints towards implementing our algorithm on real robot-sensor systems.**

I. Introduction

In applications ranging from surveillance to object inspection, an autonomous robot is required to inspect the surfaces of objects or boundaries of the workspace in a large and/or cluttered environment. Every surface of the objects of interest (which could be the whole environment) must be viewed/covered via at least one planned viewpoint of the range sensor. It is desired that the total cost of the plan, consisting of the traveling cost (the total distance traveled by the robot along the planned path, often a measure of the total amount of energy consumed by the robot) and the view cost (proportional to the number of viewpoints planned where each viewing overhead is due to image acquisition, processing, and registration [18]), is minimized, often a requirement for autonomous robotic missions where battery life is a significant issue [17]. We call this problem *Traveling View Planning Problem*, or *Traveling VPP* in short, and formulate it as an optimization problem. We assume the object and environment are of known geometries.

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

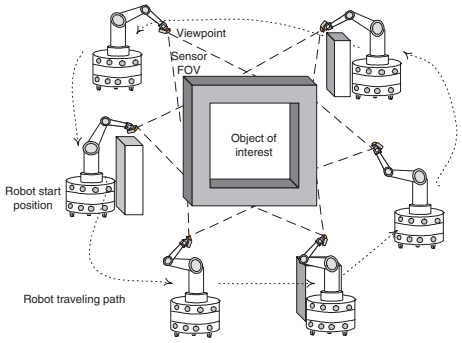


Fig. 1. A Traveling VPP instance. It shows 6 planned sensor viewpoints that totally cover the object surface, and the robot traveling path.

Traveling VPP combines elements of two well-known NP-hard problems, the view planning problem (VPP) and the Metric Traveling Salesman Problem (Metric TSP), and thus becomes NP-hard itself. (Traveling VPP also incorporates the Watchman Route Problem [4] in the computational geometry area. See Sec. II-A for more on it.) VPP refers to planning the minimum number of viewpoints to completely inspect an object surface. We refer to [18] for a survey on VPP, and to [22] for a reduction from the set covering problem (SCP) to VPP, thus showing that VPP cannot be approximated within a logarithmic ratio of its optimal solution cost by any polynomial algorithms, using the result for SCP [7]. VPP is usually 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 traveling cost of the robot, a critical cost, particularly for large workspaces and remote autonomous missions, where power consumption is a critical factor. On the other hand, Metric TSP refers to planning a tour to visit specified vertices on a complete graph with metric. The well-known and straightforward approximation framework is to approximate the tour by solving first a shortest tree

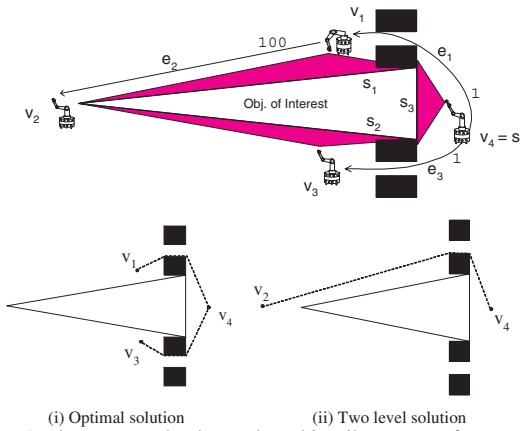
connecting these vertices and constructing a tour from the tree [21]. Note that Metric TSP does not consider view cost, a critical cost, particularly for the inspection tasks considered here, where each sensor view and the consequent processing are time-consuming.

There is some existing work on combining the view and traveling cost in the literature, but not in a unified and global fashion. For example, in [8], [13], the authors considered a local version of the robot exploration problem, "to look around a corner", i.e., to detect an object hidden behind a corner while minimizing the sum of the robot traveling distance and the sensor scan time. The problem is considerably simpler since the goal is local, i.e., the objective is not to cover all the object surfaces.

In [5], the authors considered the combined problem, however, in a "weak sense", since no view cost is considered, thus corresponding to a special case of *Traveling VPP*. They proposed to solve the problem by a decoupled two-level approach, i.e., to plan the minimum number of viewpoints without considering robot traveling cost first and then to solve (approximately) the Metric TSP using the shortest path graph. This two-level decoupled approach would work well for cases where the views considered do not have overlap between their coverage (they become the only choice for a view plan.), or those with large coverage overlap are close to each other (they correspond to similar traveling costs). However, this is not true for a general Traveling VPP setting. For example, as shown in Fig. 2, even assuming that at each level the respective optimization subproblem, obtaining the minimum number of viewpoints and the shortest path tour respectively, is solved optimally, this two-level decoupled approach provides no performance bound (with respect to the optimal solution cost), and can perform arbitrarily poorly. This is easily achieved by pulling the leftmost viewpoint arbitrarily farther from the rightmost ones. This issue occurs because the planned viewpoints at the first level are too far apart for the robot to realize a plan efficiently since no traveling cost is considered at the first stage.

That *Traveling VPP* cannot be satisfactorily solved by decoupled approaches motivated the approach we take in this paper, i.e., to give a unified problem formulation with a single objective function combining both view and traveling costs, and adopt the approximation algorithm for NP-hard problems as our methodology. Approximation algorithms are algorithms that run in polynomial time and guarantee the algorithmic solution cost (w.r.t. the optimal cost) even in the worst case. The parameter to measure the quality of an approximation algorithm is the *approximation ratio*, defined as the largest ratio between the costs of the algorithmic solution and the optimal solution. See [21] for a detailed survey on approximation algorithms.

We then use the approach based on linear program (LP) relaxation for our approximation algorithm. We give a rounding algorithm that takes an optimal relaxed LP solution and outputs an integral Traveling VPP solution. A key result of this paper is that the algorithm has an approximation ratio that is twice the "view frequency", where the view frequency



(i) Optimal solution (ii) Two level solution

Fig. 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 viewpoints are v_1, v_2, v_3 and v_4, all shown in the top figure. v_4 coincides with the robot start position s. The shaded sensing triangles show the covering relations: viewpoints v_1, v_3 and v_4 cover the surface patches s_1, s_2 and s_3 respectively, while v_2 (sensing triangle of which is not shown) covers both s_1 and s_2. The line segments connecting the views, e_1, e_2 and e_3, denote the robot's traveling path; the numbers on each segment are the respective traveling costs (distances). (The distances are not drawn to scale.) We assume the view and traveling cost are equally weighted in the objective function. 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.

is defined as the maximum number of viewpoints that covers a single surface patch. Also, we show in [22], via a reduction from Traveling VPP to the Group Steiner Tree problem (GST), that the existing poly-log approximation algorithm for GST [9] is applicable to Traveling VPP and achieves a poly-log approximation ratio. This result parallels that for the SCP, for which the best approximation ratio is either the frequency or a logarithm of the input size [21].

The relaxed LP formulation given in this paper may have exponential number of connection constraints. We suggest two ways: to adopt the column generation approach, [6], to practically solve the LP; or to use an alternative LP relaxation formulation. We refer to [22] for details of these two approach and focus in this paper a special case (again a NP-hard problem), that of Traveling VPP where the graph given is a tree. It is motivated by tree structures commonly used in motion planning techniques to explore and represent the connectivity of the planning space ([2], [16]). We give a polynomial sized relaxed LP formulation for the *Traveling VPP on a Tree*, which results in a polynomial algorithm with an approximation ratio of the view frequency. And finally, we discuss some realistic issues and constraints towards implementing the algorithms on a real system. These include how to generate viewpoints and the traveling graph, and how to accommodate additional constraints.

The rest of the paper is organized as follows. First, we give our formulation for Traveling VPP, followed by some related work in the combinatorial optimization area. Second, we give an LP-based rounding algorithm and analyze its performance. Third, we give the LP formulation for Traveling

VPP on a Tree. Finally, we discuss how to incorporate realistic issues towards implementations.

II. ILP Formulation and Related Work

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., $e = < v_1, v_2 > \in \delta(T) \Longleftrightarrow v_1 \in T \wedge v_2 \notin T$ 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 traveling 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 $\mathcal{A}$.)

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:

Traveling VPP (ILP): (1)

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

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

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

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

The coverage constraints, (2), require that for each surface at least one view is chosen from its viewpoint set. The connection constraints, (3), require that for each planned view i (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 [10], 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 [22], we 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.

A. Related work in optimization area

As mentioned in the introduction, the Traveling VPP can also be viewed as a generalization of the watchman route problem, i.e., the problem of planning the shortest tour to inspect the interior of a two-dimensional polygonal region, considered in the computational geometry area [4]. 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, Traveling VPP is applicable to more general cases. Such graphs, called roadmaps in the robot configuration space, are commonly used for the high-dimensional path planning problem in robot motion planning literature [14], [15].

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.

The errand scheduling problem (ESP) is defined as: given a graph $G = (V, E)$ with metrics (the edge weights satisfy the triangular inequality), where each vertex is associated with a subset of errands, plan a shortest tour such that the set of all errands 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. (ρ is equivalent to the view frequency F defined above.) Traveling VPP 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 if some viewpoints are chosen in the solution, the robot does not need to take a view at them. They are simply for travel use and do not incur view cost, hence termed as Steiner nodes [21]. Thus the results in [19] does not apply to Traveling VPP. Simple reductions from

Traveling VPP to ESP, for example, adding to the travel cost of each edge $e = (u, v)$ the view cost of both viewpoints u and v, do not work, since this construction requires that the robot take a view at every graph node it travels.

III. LP BASED ALGORITHM FOR TRAVELING VPP

By relaxing the binary integral variables, y_i and z_e, to be positive reals, we have the relaxed LP for Traveling VPP. 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*.

A. Rounding Algorithm

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

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

Step 1. **Initialize.**

Set viewpoint choice set $\mathcal{V}^c$ to include all the viewpoints, i.e., $\mathcal{V}^c \leftarrow \mathcal{V}$; the viewpoint solution set $\mathcal{V}'$ to be empty, i.e., $\mathcal{V}' \leftarrow \emptyset$; the uncovered surface patch set $\mathcal{S}^u$ to include all surface patches, i.e., $\mathcal{S}^u \leftarrow \mathcal{S}$

Step 2. **Round.**

While set $\mathcal{S}^u$ is not empty

Select the viewpoint $i_{max} \in \mathcal{V}^c$ that covers some uncovered surface patch(es) and has the largest LP-optimal viewpoint assignment, i.e., $i_{max} = \arg\max_{i\in\mathcal{V}^c:\ \mathcal{S}(i)\cap\mathcal{S}^u\neq\emptyset} y_i^*$, and add it to $\mathcal{V}'$, i.e., $\mathcal{V}' \leftarrow \mathcal{V}' \cup \{i_{max}\}$

Delete the surface patch(es) that i_{max} covers from the uncovered surface patch set, i.e., $\mathcal{S}^u \leftarrow \mathcal{S}^u \setminus \mathcal{S}(i_{max})$; and delete i_{max} from the viewpoint choice set, i.e., $\mathcal{V}^c \leftarrow \mathcal{V}^c \setminus \{i_{max}\}$

Output $\mathcal{V}'$, i.e., set $y_i' = 1$ for $i \in \mathcal{V}'$, and set $y_i' = 0$ for $i \notin \mathcal{V}'$.

Step 3. **Connect.**

Get the optimal solution to the Steiner tree problem to connect $\mathcal{V}'$. Set $z_e' = 1$ for edges in the solution, and 0 otherwise.

In the above algorithm, we iteratively choose the viewpoint with the largest (fractional) LP-optimal viewpoint assignment until all the surface patches are covered. We then feed these chosen viewpoints to a Steiner tree algorithm to get the optimal integral solution, in the Connect step. Note that the Steiner tree problem is an NP-complete problem for a general graph. So practically speaking, we can use a constant-ratio approximation algorithm, for example the one in [10], and incur an additional bounded performance degradation. It is easy to see that the rounding part of the above algorithm (up to the Connect step) runs in polynomial time, $O(|\mathcal{V}||\mathcal{S}|)$.

B. 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.

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 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. ■

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

Proof: It is equivalent to show that the above algorithm cannot choose any viewpoint whose LP optimal viewpoint assignment is less than $\frac{1}{F}$. We show this by contradiction. Assume we choose one viewpoint i with $y_i^* < \frac{1}{F}$. By the *Round and Connect* algorithm, the Round Step, at the iteration when i is picked, it has the maximum LP optimal viewpoint assignment among the viewpoints that covers the remaining uncovered surface(s). We arbitrarily choose one uncovered surface patch that i covers. By 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

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

2) Total cost analysis: In the following, after stating the half integrality gap[1] result of the Steiner tree problem [21], 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 [21]. ■

Note that the Connect Step of the algorithm *Round and Connect* corresponds to the Steiner tree problem of connecting $\mathcal{V}'$, the IP optimal solution to which is z_e'. We use OPT_{tree}' to denote the corresponding optimal value. 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: The idea is to utilize an intermediate solution with integral viewpoint assignments and fractional edge assignments, i.e., the set of viewpoints chosen is $\mathcal{V}'$ and the edge assignments are scaled F times from their ILP optimal solution, i.e., $F \cdot z_e^*$. We show the edge assignments of this intermediate solution is a feasible LP solution to the Steiner tree problem to connect $\mathcal{V}'$. Its traveling cost is lower bounded by OPT_{tree}^*, which is not far from OPT_{tree}' due to Lemma 4. This thus establishes the approximation ratio result. Please see [22] for details. ■

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 $2F$. Please also see [22] for a tight example, thus suggesting that the $2F$ approximation algorithm *Round and Connect* is the best possible for the LP relaxation given above.

IV. 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. Second, we show that the *Traveling VPP on a Tree* admits a polynomial sized relaxed LP 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, (3), 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:

$$\begin{aligned}
\min \quad & w_v \sum_{i\in\mathcal{V}} y_i + w_p \sum_{e\in E} c_e z_e \\
\text{Subject to:} \quad \forall j \in \mathcal{S}, \quad & \sum_{i\in\mathcal{V}(j)} y_i \geq 1 \qquad (4) \\
\forall i \in \mathcal{V}, \forall e \in E : e \in p_i, \quad & z_e \geq y_i \qquad (5) \\
& y_i, z_e \geq 0, i \in \mathcal{V}, e \in E
\end{aligned}$$

Since the covering constraints for the above formulation and for *Traveling VPP* are the same, we only need to show the equivalency of the connection constraints, (3) and (5). 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 (3), we have $\sum_{e'\in\delta(T_e)} z_{e'} = z_e \geq y_i$, (5). 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 (3) and (5) are equivalent for the tree case.

Note that in the above formulation, the number of constraints is $O(|\mathcal{S}| + |E||\mathcal{V}|)$, i.e., polynomial and not exponential as in the formulation for the general graph case. In our simulation, using ILOG CPlex LP solver [1], it takes about 30 seconds to solve instances with 1000 viewpoints.

We show in [22] that 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 [9] to achieve a 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 the order view frequency or a poly-log ratio, whichever is smaller. This approximation result parallels that for the set covering problem [21].

V. Incorporating Our Algorithm in Robot Surveillance Task

Our current efforts are to implement the algorithm Round and Connect on a real system, a PowerBot mobile robot [12] equipped with a camera system, to carry out surveillance

[1]Integrality gap for an LP is defined to be the worse-case ratio between the cost of the IP solution and that of its LP relaxation solution [21].

tasks in our lab area, where the map, the lab layout, is given. We briefly mention the realistic issues that needs to be addressed toward this implementation.

Our Traveling VPP formulation assumes the viewpoint set $\mathcal{V}$ and the traveling graph G connecting these viewpoints are given. For given scenes, these viewpoints could be either naturally given, for example, doorways are natural viewpoints to look inside the rooms, or they could be derived from the aspect graph of the scene [3], or by randomly sampling the sensor configuration space, the space of the sensor configurations that uniquely determines the viewpoints [11]. To adequately sample the viewpoint space, multiple viewpoints having the same visibility should be included, since they may correspond to different traveling costs. We assumed a binary coverage relationship mentioned, i.e., a viewpoint can either cover a surface patch or not. In reality, a viewpoint may cover a surface patch only partially. By subdividing surface patches, we can maintain binary coverage relationship. Realistic sensor field of view constraints such as the line of sight constraint (i.e., a viewpoint sees a surface patch only if the line segment that connects them is not occluded), the range constraint (a viewpoint sees a surface patch only if the distance between them is within a range), and the incidence constraint (i.e., a viewpoint sees a surface patch only if the angle between the line connecting them and the surface normal is in a range) can be incorporated via viewpoint and surface patch visibility computations. The Traveling graph would essentially be a roadmap built in the configuration space of the robot. This is a standard and well-studied technique for robot motion planning [14], [15].

For accurate registration, it is desirable that two planned consecutive viewpoints should have enough overlap in the surface patch sets they cover [18]. This can be incorporated using a set multicover constraint, i.e., each element in the universe needs to be covered by a specified number of subsets in the solution. The idea is as follows. We create new surface patches that are composed of unions of parts of original consecutive patches. The viewpoint set of these created patches are those covering both consecutive patches. By requiring that the viewpoints cover these created surface patches twice, i.e., changing r.h.s. of (2) from 1 to 2 for these patches added, these viewpoints can register them w.r.t. each other and thus the overall viewpoints can all satisfy the overlapping constraints.

VI. Conclusion and Future Work

In this paper, we introduced the Traveling VPP, the problem of view planning with combined view and traveling cost. We formulated Traveling VPP into an ILP, and gave an LP-based rounding algorithm that has the frequency factor approximation ratio. Together with the poly-log approximation ratio achieved via a reduction to GST, Traveling VPP can be approximated within the minimum of a constant times frequency and a poly-logarithmic function of the input size. Also for Traveling VPP on a Tree, a special case motivated from the robot motion planning literature, we gave a polynomial sized LP. We then discuss several realistic issues and constraints towards implementing our algorithm for a real system.

In future, we would like to generalize our result to the case where the surface patches to cover and the robot traveling graph are not known in advance. 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. We are currently working on it [23].

References

[1] Ilog cplex. http://www.ilog.com/products/cplex.

[2] P. Bessiere, J. Ahuactzin, E. Talbi, and E. Mazer. The ariadne's clew algorithm: Global planning with local methods. In K. Goldber, D. Halperin, J. Latombe, and R. Wilson, editors, *Algorithmic Foundation of Robbotics*, pages 39–47. A K Peters, Ltd., 1995.

[3] 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.

[4] W. Chin and S. Ntafos. Watchman routes in simple polygons. *Discrete and Computational Geometry*, 6(1):9–31, 1991.

[5] T. Danner and L. Kavraki. Randomized planning for short inspection paths. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 971 – 976, 2002.

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

[7] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634 – 652, July 1998.

[8] S. Fekete, R. Klein, and A. Nuchter. Online searching with an autonomous robot. In *Proc. of Workshop on Algorithmic Foundation of Robotics*, pages 350–365, 2004.

[9] N. Garg, G. Konjevod, and R. Ravi. A polylogorithmic approximation algorithm for the group steiner tree problem. *Journal of Algorithms*, 37:66–84, 2000.

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

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

[12] http://www.activrobots.com/ROBOTS/power.html.

[13] V. Isler, S. Kannan, and K. Daniilidis. Local exploration: online algorithms and a probabilistic framework. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 1913 – 1920, 2003.

[14] 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.

[15] J. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[16] S. LaValle and J. Kuffner. Randomized kinodnamic planning. In *Proc. of IEEE International Conf. on Robotics and Automation*, pages 473–479, 1999.

[17] Y. Mei, Y. Lu, Y. Hu, and C.S. Lee. A case study of mobile robot's energy consumption and conservation techniques. In *Proc. of International Conference on Advanced Robotics*, pages 492– 497, 2005.

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

[19] P. Slavik. The errand scheduling problem. Technical Report 97-02, Sate University of New York at Buffalo, 1997.

[20] C. Swamy and A. Kumar. Primal-dual algorithms for connected facility location problems. *Algorithmica*, 40:245–269, 2004.

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

[22] P. Wang, R. Krishnamurti, and K. Gupta. View planning problem with combined view and traveling costs: Problem formulation, hardness of approximation, and approximation algorithms. Technical Report TR2006-17, Simon Fraser University, Burnaby, B.C., Canada, May 2006. Available at ftp://fas.sfu.ca/pub/cs/TR/2006/CMPT2006-17.ps.

[23] P. Wang, R. Krishnamurti, and K. Gupta. Generalized watchman route problem with discrete view cost. Submitted to *23rd Annual ACM Symposium on Computational Geometry*, 2007.