Visibility-Based Monitoring Of A Path Using a Heterogeneous Robot Team

Parikshit Maini[$], Gautam Gupta[†], Pratap Tokekar[‡], PB Sujit[*]

***Abstract*— We address the problem of visually monitoring a terrain path using ground and aerial robots. This is a coupled problem that involves computation of a guard set for the environment and route planning for a heterogeneous group of robots through the points in the guard set. A terrain path that needs to be monitored can be transformed to generate a 1.5D terrain and a chain visible curve for this terrain can be generated. To efficiently monitor this 1.5D terrain, we present two solutions — a dynamic programming approach that finds the optimal solution but is slower and a integer linear programming solution that is faster in practice and that can take more constraints into account. We perform extensive simulations and do a comparative analysis of the two solution techniques.**

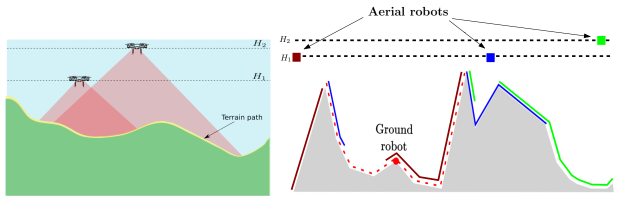
I. Introduction

We study the problem of planning paths for a team of heterogeneous robots consisting of autonomous aerial and ground vehicles to visually monitor an environment. There are numerous high-impact applications where visual coverage of environments with robotic sensors will be transformative. Examples include environmental monitoring [1], precision agriculture [2], search-and-rescue [3], surveillance [4], mapping [5] and border patrolling [6].

In this paper, we focus on the specific problem of visually monitoring a path on a terrain with multiple unmanned vehicles as shown in Figure 1a. The terrain imposes visibility constraints on the vehicles. When UAV is at altitude H_1, its visibility is shortened as the terrain path altitude increases. When the UAV is at altitude H_2, it can sense a larger region of the terrain path. However, operating the vehicle at higher altitudes increases the operational cost and reduces the image resolution on the path. Since the monitoring is performed along a predefined path, we can transform (stretch) the path to a 1.5D terrain. The terrain altitude changes are captured by the height. A sample transformed terrain is shown in Figure 1b, where UAVs can operate at two altitudes H_1 and H_2, while the unmanned ground vehicle (UGV) motion is restricted on the ground.

The goal is to plan routes for the robots such that every point in the environment is eventually observed by one of the robots along its tours. The general problem is challenging since it is a combination of two NP-hard [7] problems: set cover [8] (to find the optimal viewing locations) and traveling salesperson problem [9] (to find routes for the robots to visit the viewpoints). A decoupled approach that first finds the viewing locations, and then finds a path for the robots can be arbitrarily bad even for 1.5D terrains, when the objective is to minimize the makespan of the paths [10]. Instead, we present a dynamic programming algorithm to find the optimal solution. We also present a more general Integer Linear Programming (ILP) formulation that can take into account a number of additional, practical constraints when planning the routes.

[$] Graduate Student, IIIT-Delhi, India.
[†] Undergraduate Student, IIIT-Delhi, India.
[‡] Assistant Professor, Dept. of ECE, Virginia Tech., US.
[*] Assistant Professor, Dept. of ECE, IIIT-Delhi, India.
Maini's research is supported by TCS Research Scholarship.



(a) (b)

Fig. 1: (a) A terrain path that needs to be monitored by multiple vehicles. (b) Illustration of a 1.5D terrain. The dashed line shows visibility region of the ground robot, whereas the solid lines (color coded) show the visibility region of the aerial robots

A. Related Work

Our problem is a generalization of the Watchman Route Problem (WRP) [11]. The objective in WRP is to find a tour of minimum length for a single robot (i.e. watchman) so as to see every point in an input polygon. Carlsson et al. [12] introduced n-WRP where the goal is to find n tours for n homogeneous robots, such that each point in the environment is seen from at least one tour. They showed that the problem is NP-hard. The Watchman Route Problem for a single robot can be solved to optimality in polygons without any holes [12], while an approximation algorithm is proposed for polygons with holes in [13]. Wang et al. [14] studied the 3D Viewpoint Routing Problem for the case of a single robot and developed an $\mathcal{O}(\log m)$ approximation algorithm. Due to the inherent complexity of the problem, there have been many efforts dedicated to finding practical solutions. These include decoupling viewpoints selection and robot routing[15], and self-organizing maps heuristics [16]. The problem naturally becomes more challenging in full 3D (where objects are allowed to float in air). A 2.5D terrain is a better model for practical environments. Efrat et al. [17] studied the problem of finding watchmen tours for n homogeneous robots flying over a 2.5D terrain. A search

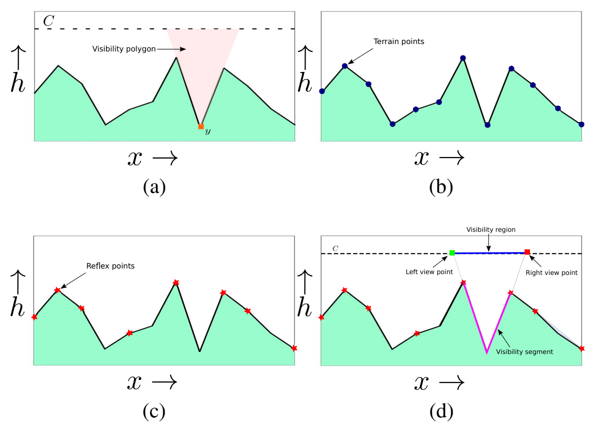


Fig. 2: (a) Terrain with a chain visible curve C and the visibility polygon (b) Terrain points of the terrain (c) Reflex points of the terrain (d) Visibility segment, visibility region and view points (left and right).

strategy for UAVs to detect targets on a 2.5D terrain was proposed in [18], [19]. However, the approach was limited to homogeneous UAVs.

In a recent work, [10] suggests that certain classes of environments have special structures which can be exploited to design good approximation algorithms. For instance, a constant-factor approximation was obtained for the n-homogeneous-WRP [10], [12] in street polygons, a special class of 2D polygons without holes. Encouraged by this, we will seek to characterize the classes of environments for which we can solve the diverse robotic sensors case. When considering a set of heterogeneous robots, it is more appropriate to consider terrains instead of 2D polygons. A 1.5D terrain can be viewed as a 1D heightmap (Figure 1b), whereas a 2.5D terrain can be viewed as a 2D heightmap. Terrains allow us to reason about the visibility of aerial and ground robots. Hence, we address the Heterogeneous Watchman Routing Problem on 1.5 D Terrains (HWRPT).

II. Preliminaries

A. Chain Visibility

A curve C and a set of points X in a 1.5D environment, are said to be chain visible if the intersection of the visibility polygon of a point $y \in X$, i.e. space that has an unrestricted view of y, with C is either an empty set or a connected chain [10]. We refer to this intersection region as the visibility region for the point y on the curve C. In our case, the fixed altitude paths are the ones that correspond to UAV flight altitudes and the set of points X includes the complete entire terrain. Also, a curve is chain visible to any set of points that lie on the curve itself, hence the terrain (UGV path) itself is a chain visible curve for points on the terrain.

Consider the terrain shown in Figure 2(a), where the chain-visibility curve C is located at an altitude from the terrain. The aerial vehicle will be flying at the altitude defined the chain-visible curve. The polygon formed from the terrain location y is the visibility polygon. All the points on the terrain form X.

B. Terrain points, reflex points and visibility segments

We define a terrain point as a point on the terrain at which the slope of the incoming and outgoing edges is different as shown in Figure 2(b). A terrain point at which the slope of the outgoing edge is less than the slope of the incoming edge, computed along the x-axis, is referred to as a *reflex point*. Reflex points for the terrain shown in Figure 2(a) is determined in Figure 2(c). Two consecutive reflex points on the terrain mark a convex polyline when observed from a point on the fixed altitude path or a point on the terrain. We refer to this convex polyline as a *visibility segment*. The visibility segment for two reflex points is shown in Figure 2(d). The projection of the visibility segment along the slope of the visibility segments on to the visibility-chain curve C is called as the *visibility region*. Figure 2(d) shows the visibility region for the earlier visibility segment. By virtue of it being convex, all points on the terrain in a visibility region are visible from any point within the visibility segment. The left end point of the visibility region is called as the left view point and the right end point of the visibility region is called as the right view points. The left and right view points for the above visibility region are shown in Figure 2(d). It may be easily observed that the visibility region of a visibility segment on any chain visible curve to the terrain, is also a chain. Carlsson et. al. [20] use similar ideas within a polygon to compute watchman routes. Note that, for the ground robots, the reflex points itself forms the left and right view points as it follows the terrain.

III. Problem Statement

Consider an environment $\mathcal{E}$, as shown in Figure 1b, with a 1.5 dimensional terrain. We are given two types of robots: ground robots that can move on the terrain, and aerial robots that can fly at a fixed altitude h (greater than the highest point on the terrain). Each set of aerial robots flying at a given altitude contributes a chain visible curve in addition to the one on the terrain that corresponds to the ground vehicles. Let $\mathcal{C}$ be the set of chain visible curves and $\mathcal{A}_c$ be the set of autonomous robots, ground or aerial, on the curve $c \in \mathcal{C}$.

Let R be the set of reflex points within $\mathcal{E}$. The reflex points combined with the terrain end points comprise the set $\mathcal{R}$. Let ν be the corresponding set of visibility segments. Let $\mathcal{V}_c^L$ and $\mathcal{V}_c^R$ be the set of left and right viewpoints on the chain visible curve c (see Figure 3). $\mathcal{V}_c = \mathcal{V}_c^L \bigcup \mathcal{V}_c^R$, is the set of all viewpoints on the curve c and $\mathcal{V} = \bigcup_{c \in \mathcal{C}} \mathcal{V}_c$ is the set of all viewpoints over all chain visible curves. $\mathcal{V}_c(v), \mathcal{V}_c^L(v)$ and $\mathcal{V}_c^R(v)$ are subsets of viewpoints corresponding to the visibility segment v contained in the respective set.

Our goal is to find routes for each robot to ensure that every point on the terrain is visible on at least one robot route. In addition to a Dynamic Program to optimally solve the problem, we also devise an Integer Linear Program that permits other objective functions that are useful in a practical scenario such as minimizing the total mission cost.

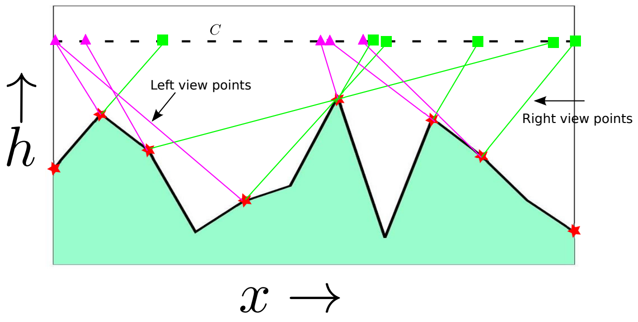


Fig. 3: The set left and right viewpoints corresponding to all visibility regions of the terrain.

In such a scenario, the following Heterogeneous Watchman Routing problem on a 1.5 D Terrain (HWRPT) arises naturally: *Given a 1.5 D terrain and a set of ground and aerial robots with optical sensors, plan routes for the robots such that all points in the terrain are observed by at least one robot and the maximum path cost for a single robot is minimized over all robots.*

IV. Dynamic Programming Approach

In this section we develop an optimal algorithm to solve HWRPT using dynamic programming (DP). Consider the environment as shown in Figure 1b with a 1.5 D terrain and the set of chain visible curves $\mathcal{C}$ that correspond to the available ground and aerial vehicles.

Lemma 1. *Adapted from [10]:* There exists an optimal solution for HWRPT with viewpoints $\mathcal{V}$ such that for any $v \in \mathcal{V}$, if v is the left end-point of a path π, then v must be the right end-point of some visibility region on a chain visible curve corresponding to a visibility segment on the terrain. similarly if v is the right end-point of a path π, then v must be the left end-point of some visibility region

This lemma allows us to restrict out attention only to a set of finite (at most $2|\nu|$) points on each $c \in \mathcal{C}$. Furthermore, we need to consider only the right viewpoints to start a path and only the left viewpoints to end a path (assuming all paths go from left to right, without loss of generality).

Lemma 2. *Let A be the optimal solution to the problem, and let there be two set of paths P and Q such that (i) $P \subset A$ and P contains all paths until some viewpoint $i_c \in \mathcal{V}_c$, when going from left to right, on the chain visible curve c; and (ii) Q contains at most one path from each chain visible curve $c, \forall c \in \mathcal{C}$, starting after i_c. Then, there always exists a non-empty set Q such that $P \bigcup Q \subseteq A$.*

For this to hold, Q must satisfy the property that no visibility region (corresponding to a visibility segment) is completely contained between sets P and Q. That is, there should not be any viewpoint such that the left viewpoint of a visibility segment, $v \in \nu$, lies to the right of i_c and the right viewpoints lies to the left of $i'_c, \forall c \in \mathcal{C}$, where i'_c is the start point of the path in Q on chain visible curve c. If Q does not satisfy this property, then it would not be possible to cover the visibility segment v in any of the paths added later to the set $P \bigcup Q$ and hence v will not be covered. Therefore, $P \bigcup Q$ cannot be a solution, due to partial visibility coverage.

Let $P = \emptyset$ at the start. We can build the solution from left to right by enumerating over all $Q's$ satisfying Lemma 2, then there exists a Q, such that $P \bigcup Q$ will be a subset of the optimal solution by Lemma 2. For the next iteration, we will initialize $P = P \bigcup Q$ and again iterate over all possible $Q's$ satisfying Lemma 2 to obtain new $P \bigcup Q$. This process continues until $Q = \emptyset$, in which case, we cannot add any new paths to P. The optimal solution will be the minimum cost solution from the set of generated solutions from P.

In order to implement the DP algorithm, we need to define a DP table T consisting of $T([i_1, j_1], \ldots, [i_n, j_n], k)$ where $i_c \in \mathcal{V}_c^R, j_c \in \mathcal{V}_c^L, c = 1, \ldots, n$ and $n = |\mathcal{C}|$. The variable k is the iteration counter. An entry in the table gives the minimum cost set P where the last path on curve c is $[i_c, j_c]$. We use the optimal single path algorithm ($OptimalSinglePath(i_c, j_c, c)$) given in [10] to determine the optimal way to cover all viewpoints from i_c to j_c on chain visible curve c.

To fill the rest of the entries in T, we also need to find i'_c and j'_c for each paths being added. Further, we also need to maintain a count on the number of vehicles on each curve corresponding to every $T([i_1, j_1], \ldots)$. We obtain $[i'_c, j'_c]$ as

$$([i'_1, j'_1], \ldots, [i'_n, j'_n]) = arg\min T([i'_1, j'_1], \ldots, [i'_n, j'_n], k') + \infty \times I([j'_1, i_1], \ldots, [j'_n, i_n]), k' = k - 1. \quad (1)$$

This equation represents the process for finding minimum cost set P such that for a given set Q, Lemma 2 is followed. The binary indicator $I([j'_1, i_1], \ldots, [j'_n, i_n])$ represents the fact that all of viewpoint between j'_c and i_c has been covered by one of the paths added.

Once we obtain $[i'_c, j'_c]$, we can compute the entries of the table as

$$T([i_1, j_1], \ldots, [i_n, j_n], k) = \max\{OptimalSinglePath(i_c, j_c, c) \forall c, T([i'_1, j'_1], \ldots, [i'_n, j'_n], k)\}. \quad (2)$$

This equation finds the cost of the $P \bigcup Q$ and updates it in the DP table. Set $[i_c, j_c]$ maybe null, which implies that no path from that curve is being added. However there must be at least one non null entry. The algorithm terminates after $\sum_{c \in \mathcal{C}} \mathcal{A}_c$ iterations.

The complexity of the DP algorithm may be computed in terms of

1) number of iterations = $\sum_{c \in \mathcal{C}} \mathcal{A}_c$.
2) Enumeration of Q: $(n)^{2|\mathcal{C}|}$.
3) Feasibility checking: $|\mathcal{C}| \times (2n)$.

Based on the above three terms, the complexity of the algorithm is $\mathcal{O}((\sum_{c \in \mathcal{C}} \mathcal{A}_c) \times (n)^{2|\mathcal{C}|} \times |\mathcal{C}|)$, where $n = |\mathcal{V}|$ is the number of viewpoints.

V. Integer Linear Programming Approach

We develop an Integer Linear Program to find optimal solutions HWRPT. An ILP allows the development of different and practically useful objective functions. We propose two different objectives and present simulation results for both.

We define a visibility function and a cost function on the viewpoints. The visibility function, $\gamma_c(v) : \nu \rightarrow \wp(\mathcal{V}_c)$ where $\wp(x)$ is the power set of x, is defined for each curve $c \in \mathcal{C}$. $\gamma_c(v)$ is the set of viewpoints on curve c that lie between the left and right end points of the visibility region of visibility segment v on c, thus $\gamma_c(v) \subseteq \mathcal{V}_c$. Also, $\gamma(v) = \bigcup_{c \in \mathcal{C}} \gamma_c(v)$. Cost function, $\alpha^c_{ij} : \mathcal{V}_c \times \mathcal{V}_c \rightarrow \mathbb{R}^+$, returns the cost to travel from i to j on the chain visible curve c, where i and j belong to the set $\mathcal{V}_c$. In addition, we also define a vehicle cost function, $\beta_c : \mathcal{C} \rightarrow \mathbb{R}^+$, that gives the one time cost for using a vehicle on the chain visible curve c.

A. Formulation

We define three sets of binary variables, S, P and x. S^{ck}_i variables, defined for all $c \in \mathcal{C}, k \in \mathcal{A}(c)$ and $i \in \mathcal{V}_c$, indicate the starting point for each vehicle. $S^{vk}_i = 1$, if k^{th} vehicle on the c^{th} chain visible curve starts at the i^{th} view point. P^{ck}_i variables, defined for each $c \in C, k \in \mathcal{A}(c)$ and $i \in \mathcal{V}_c$, indicate whether or not the k^{th} vehicle on the c^{th} chain visible curve visits the i^{th} viewpoint. x^{ck}_{ij} variables are defined for each $c \in C, k \in \mathcal{A}(c)$ and $i, j \in \mathcal{V}_c$. $x^{ck}_{ij} = 1$, if the k^{th} vehicle on the c^{th} chain visible curve travels from viewpoint i to j on c and 0 otherwise. We also define the set $\delta_c(\mathcal{P})$, $\forall c \in C, \forall \mathcal{P} \subseteq \mathcal{V}_c$. $\delta_c(\mathcal{P})$ defined as, $\delta_c(\mathcal{P}) \equiv \{(i,j) : i \in \mathcal{P}, j \in \mathcal{V}_c \setminus \mathcal{P} \quad \text{or} \quad j \in \mathcal{P}, i \in \mathcal{V}_c \setminus \mathcal{P}\}$, is the set of all edges between the sets $\mathcal{P}$ and $\mathcal{V}_c \setminus \mathcal{P}$. An edge corresponds to a path segment on the visibility curve between any two view points.
Objective Function:

$$\mathcal{F}1 : \quad \min \max_{c \in C,\ k \in \mathcal{A}_c} \sum_{i \in \mathcal{V}_c} (\beta_c \cdot S^{ck}_i + \sum_{j \in \mathcal{V}_c} \alpha^c_{ij} \cdot x^{ck}_{ij})$$

$$\mathcal{F}2 : \quad \min \sum_{c \in C} \sum_{k \in \mathcal{A}_c} \sum_{i \in \mathcal{V}_c} (\beta_c \cdot S^{ck}_i + \sum_{j \in \mathcal{V}_c} \alpha^c_{ij} \cdot x^{ck}_{ij})$$

The objective function in an Integer Linear Program may be designed as any linear combination of the decision variables. Objective function $\mathcal{F}1$ optimizes a $\min - \max$ objective that minimizes the maximum path cost over all vehicles while $\mathcal{F}2$ is designed to minimize the total cost of completing the mission. We report simulation results for both $\mathcal{F}1$ and $\mathcal{F}2$ in Section VI.

The constraints used in the ILP are as follows:

$$\sum_{c \in C} \sum_{k \in \mathcal{A}_c} \sum_{i \in \gamma_c(v)} P^{ck}_i \geq 1, \quad \forall v \in \nu \tag{3}$$

$$\sum_{i \in \mathcal{V}_c} S^{ck}_i \leq 1, \quad \forall c \in C, \forall k \in \mathcal{A}_c \tag{4}$$

$$\sum_{m \in \mathcal{V}_c} S^{ck}_m \geq x^{ck}_{ij}, \quad \forall c \in C, \forall k \in \mathcal{A}_c, \forall i, j \in \mathcal{V}_c \tag{5}$$

$$\sum_{i \in \mathcal{V}_c \setminus j} x^{ck}_{ij} + S^{ck}_j = P^{ck}_j, \forall c \in C, \forall k \in \mathcal{A}_c, \forall j \in \mathcal{V}_c \tag{6}$$

$$P^{ck}_i - \sum_{j \in \mathcal{A}_c} x^{ck}_{ij} \geq 0, \quad \forall c \in C, \forall k \in \mathcal{A}_c, \forall i \in \mathcal{V}_c \tag{7}$$

$$\sum_{i \in \mathcal{P}} (P^{ck}_i - 1) - \sum_{(i,j) \in \delta_c(\mathcal{P})} x^{ck}_{ij} \leq -S^{ck}_m, \quad \forall c \in C, \forall k \in \mathcal{A}_c, \forall m \in \mathcal{V}_c, \forall \mathcal{P} \subseteq \mathcal{V}_c \setminus m \tag{8}$$

Constraint 3 ensures that for each visibility segment v at least one viewpoint in the set $\gamma(v)$ is visited by one of the vehicles, thus ensuring coverage. Each vehicle must start from exactly one point, if at all the vehicle is used (Constraint 4). Constraint 5 ensures that a vehicle must start before it can traverse between any two viewpoints. Vehicle visit variables P^{ck}_i are defined in Constraint 6. A vehicle is said to have visited a viewpoint, if it starts its path from that viewpoint or has an incoming path to that viewpoint. Further, a vehicle may have an outgoing path from a viewpoint, if and only if, it has visited that viewpoint (Constraint 7). Sub-tours that do not include the starting point for the vehicle in use, must be discarded (Constraint 8). As is clearly visible, the number of sub-tour elimination constraints is exponential in the number of viewpoints. The number of viewpoints is proportional to the number of visibility segments and by transitivity, the number of reflex points. To this end, we design a separation algorithm, that generates valid inequalities and adds them to the solver during run time.

B. Branch-and-cut algorithm

As noted in Section V-A, the number of sub-tour elimination constraints is exponential and it is not computationally efficient to enumerate all of them and add to the solver. To address this problem, we implement the ILP formulation using a branch-and-cut framework. Within the branch-and-cut framework, we input a relaxed version of the problem to the solver, that does not include the sub-tour elimination constraints. When the solver obtains an integer feasible solution to this relaxed version, it is checked for feasiblity against the relaxed constraints using a separation algorithm. In case the solution violates a relaxed constraint, the separation routine generates a valid inequality (violated constraint) and adds it to the formulation. The problem is then given back to the solver. This process of adding constraints to the problem sequentially has been observed to be computationally efficient for many variants of the Traveling Salesman Problem [21] and other combinatorial optimization problems [22].

The sepation algorithm to identify and generate valid inequalities for HWRPT operates as follows. It computes the strongly-connected components in the graph defined by the integer solution for each vehicle. Each component P of cardinality greater than one that satisfies the condition $P \subseteq \mathcal{V}_c \setminus \{m\}$, where c is the curve on which the given vehicle traverses and m is the starting point of the vehicle, violates the corresponding constraint (Constraint 8). These violated constraints, given in Eq. (9), are then added to the formulation and the solver is allowed to optimize the problem with the new constraints.

Config #	UAVs	UGVs
1	one at H_1	one
2	one at H_2	one
3	two at H_1	one
4	two at H_2	one
5	one at H_1, one at H_2	one

TABLE I: Environment Configurations

$$\sum_{i\in\mathcal{P}}(P_i^{ck}-1)-\sum_{(i,j)\in\delta_c(\mathcal{P})}x_{ij}^{ck}\le -S_m^{ck}, \tag{9}$$

VI. Simulation Results

We will first show few example scenarios where the paths for different type of vehicles are assigned on the visible curve. We will then follow by Monte-Carlo simulations.

A. *Examples*

Initially, we consider scenarios where only a single vehicle is used and then proceed towards multiple vehicles. For all the scenarios, one UGV and 2 UAVs (U_1 and U_2) are available to determine the path for the vehicles. The visibility curves are on the terrain for the UGV, at 2m and 5m altitudes respectively for the UAVs. If the UAV is used at $2m$ then the operating and movement cost for the UAV is twice that of the UGV. When UAV is used at 5m then the cost of using is twice that of at $2m$. For the scenario shown in Figure 4a, the DP solution assigns the path for UAV (U_1) only, while U_2 and UGV are not used. When the UAV moves from the left side of the assigned path to the right side, then the UAV covers the complete terrain. The path was determined in 0.23s and the cost is 23. Similarly, consider the terrain shown in Figure 4b, where only a single UGV is selected to perform the coverage. The other vehicles are not used in this case. The solution for this terrain was determined in 0.12s and the cost for the UGV is 23.

Now, we consider the scenarios, where, two vehicles are used. For the terrain shown in Figure 5a, the DP solution uses a team composed of U_1 and UGV to monitor the terrain. The solution was determined in 0.25s and the cumulative cost of using these two vehicles is 36.7. In Figure 5b, two UAVs are employed but they hover at the same location on the $2m$ visibility curve. Hence, they contribute only to the operating cost without any path cost. The solution cost is 15.56 and the time taken is 8.43s. These example scenarios show the kind of paths that the DP solution can provide. Note that, the ILP also provides the optimal solution and the solution obtained was the same with similar computational times and hence they are not reported here.

B. *Monte-Carlo results*

We generated random environments with number of terrain points varying from 5 to 30 in steps of 5. Five random terrains were generated for each size. Two different altitudes, H_1 and H_2 (Figure 1b), were chosen to place chain visible curves corresponding to aerial vehicles and the terrain is the

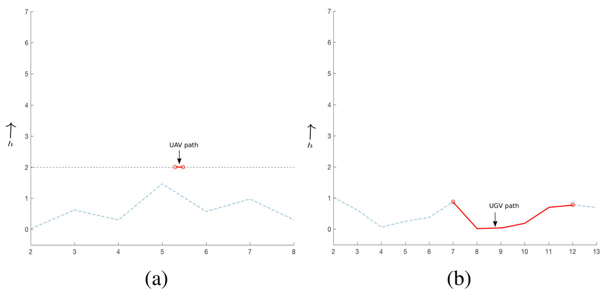


Fig. 4: The visible curve for the UAV is placed at 2m altitude. (a) The DP solution assigns a small path for U_1 to view the complete terrain. (b) The DP algorithm assigns only a UGV for the terrain.

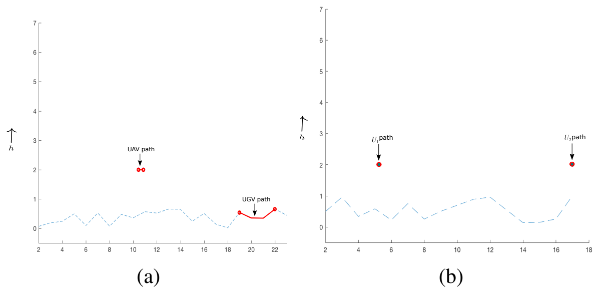


Fig. 5: The visible curve for U_1 is placed at 2m altitude. (a) The DP solution assigns a small path for U_1 and the UGV to view the complete terrain. (b) The DP algorithm assigns U_1 and U_2 both of them at $2m$ altitude only.

visible chain curve for the UGV. The simulations were run on all environments for each robot combination as given in Table I. DP was implemented in MATLAB and simulations were run on a 4-core Intel Core i7 processor. The ILP was solved using a branch-and-cut framework developed in C++ using IBM ILOG CPLEX library version 12.7. We utilized the lazy constraint callback functionality of the CPLEX library to implement the separation algorithm for dynamic constraint addition.

The time complexity for DP is exponential in the number of chain visible curves. While it was able to compute optimal solution for instances upto 25 terrain points, the DP does not scale well to larger instances and quickly becomes intractable (see Table II). The ILP formulation however, though exponential in the number of constraints, performs well in a branch-and-cut framework. The addition of constraints during run time, allows it to scale better. It was able to solve instances of size upto 30 terrain points. Even for 30 terrain points the average computation time for ILP is similar to that of DP with 25 terrain points. As can be seen from Tables III and IV, the growth in computation time is more gradual in case of ILP. The computation time for DP grows exponentially as the number of chain visible curves increases and hence we do not have computation time for DP for larger terrains.

Another interesting observation is the trend of computation time for ILP with different objective function. In case of objective function $\mathcal{F}1$ (Table III), the growth of computation time is faster in comparison to objective function $\mathcal{F}2$ (Table IV) The versatility of ILP in allowing different objective functions as well as addition of practically relevant constraints make it a useful technique.

Size	Computation Time in seconds				
	1 x H1	1 x H2	2 x H1	2 x H2	1 x H1, 1 x H2
5	0.65	0.57	0.9	0.60	0.68
10	0.59	0.57	0.64	0.62	0.57
15	0.74	0.85	0.9	0.95	0.75
20	1.99	2.02	1.91	2.01	1.97
25	77.8	75.03	74.68	74.61	70.97.

TABLE II: Simulation results computed using the Dynamic Program to minimize robot path makespan.

VII. Conclusions

This work addresses the persistent monitoring application in a 1.5 D terrains using a heterogenous group of ground and aerial robots. We develop two techniques to solve the problem. The first technique uses a Dynamic Programing to compute the optimal solution. The second technique develops an ILP formulation and uses a branch-and-cut framework to find optimal solutions. Since the problem is a combination of two NP-hard problems, set cover and traveling salesman problem, the solution techniques do not scale well. However, the ILP allows the computation of solutions for terrains with up to 30 terrain points optimally. For larger instances the ILP formulation may be used to compute feasible solutions with small acceptable tolerances. A natural extension of the problem is take vehicle refueling constraints into account [23]. Another line of work involves developing fast heuristics to both the original problem and the fuel constrained version.

Size	Computation Time				
	1 x H1	1 x H2	2 x H1	2 x H2	1 x H1, 1 x H2
5	0.0	0.0	0.0	0.0	0.0
10	0.1	0.1	0.1	0.1	0.1
15	0.4	0.3	0.4	0.4	0.5
20	3.7	2.6	6.1	0.7	1.0
25	16.4	9.5	10.0	14.0	12.3
30	93.9	27.5	201.9	145.0	133.5

TABLE III: Simulation results computed using the ILP formulation to minimize robot path makespan.

Size	Computation Time				
	1 x H1	1 x H2	2 x H1	2 x H2	1 x H1, 1 x H2
5	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.1	0.0	0.0
15	0.4	0.2	0.4	0.3	0.4
20	0.8	0.4	0.7	0.4	0.5
25	3.5	2.6	3.4	1.9	2.3

TABLE IV: Simulation results computed using the ILP formulation to minimize total mission cost.

The ILP allows inclusion of initial feasible solutions to speed up computation and the heuristics play an important role for the same.

References

[1] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *IEEE Robotics and Automation Magazine*, vol. 19, no. 1, pp. 24 –39, Mar 2012.

[2] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[3] R. R. Murphy, *Disaster robotics*. MIT press, 2014.

[4] R. W. Beard, T. W. McLain, D. B. Nelson, D. Kingston, and D. Johanson, "Decentralized cooperative aerial surveillance using fixed-wing miniature uavs," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1306–1324, 2006.

[5] P. Maini and P. B. Sujit, "On cooperation between a fuel constrained uav and a refueling ugv for large scale mapping applications," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2015, pp. 1370–1377.

[6] C. C. Haddal and J. Gertler, "Homeland security: Unmanned aerial vehicles and border surveillance." Library of Congress, Washington DC Congressional Research Service, 2010.

[7] V. Vazirani, *Approximation algorithms*. Springer Publishing Company, Incorporated, 2001.

[8] H. Brönnimann and M. T. Goodrich, "Almost optimal set covers in finite VC-dimension," *Discrete & Computational Geometry*, vol. 14, no. 1, pp. 463–479, 1995.

[9] S. Arora, "Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems," *Journal of the ACM*, vol. 45, no. 5, pp. 753–782, 1998.

[10] P. Tokekar and V. Kumar, "Visibility-based persistent monitoring with robot teams," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 3387–3394.

[11] W. Chin and S. Ntafos, "Optimum watchman routes," in *Proceedings of the ACM Symposium on Computational Geometry*.

[12] S. Carlsson, H. Jonsson, and B. J. Nilsson, "Finding the shortest watchman route in a simple polygon," *Discrete & Computational Geometry*, vol. 22, no. 3, pp. 377–402, Oct 1999.

[13] J. S. Mitchell, "Approximating watchman routes," in *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, 2013, pp. 844–855.

[14] P. Wang, K. Gupta, and R. Krishnamurti, "Some complexity results for metric view planning problem with traveling cost and visibility range," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 3, pp. 654–659, July 2011.

[15] E. Packer, "Computing multiple watchman routes," *Experimental Algorithms*, pp. 114–128, 2008.

[16] J. Faigl, "Approximate solution of the multiple watchman routes problem with restricted visibility range," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1668–1679, Oct 2010.

[17] A. Efrat, M. Nikkilä, and V. Polishchuk, "Sweeping a terrain by collaborative aerial vehicles," in *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*.

[18] A. Kolling, A. Kleiner, M. Lewis, and K. Sycara, "Pursuit-evasion in 2.5 d based on team-visibility," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4610–4616.

[19] A. Kleiner, A. Kolling, M. Lewis, and K. Sycara, "Hierarchical visibility for guaranteed search in large-scale outdoor terrain," *Autonomous Agents and Multi-Agent Systems*, vol. 26, no. 1, pp. 1–36, 2013.

[20] S. Carlsson and B. J. Nilsson, "Computing vision points in polygons," *Algorithmica*, vol. 24, pp. 50–75, 1999.

[21] M. Grötschel, M. W. Padberg, *et al.*, "Polyhedral theory," *The traveling salesman problem*, pp. 251–305, 1985.

[22] P. Maini, K. Sundar, S. Rathinam, and P. B. Sujit, "Cooperative planning for fuel-constrained aerial vehicles and ground-based refueling vehicles for large-scale coverage," *CoRR*, vol. abs/1805.04417, 2018. [Online]. Available: http://arxiv.org/abs/1805.04417

[23] P. Maini, K. Yu, S. P.B, and P. Totekar, "Persistent monitoring with refueling on a terrain using a team of aerial and ground robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.