BBTM: New life for old ATM paradigms

Vangelis Angelakis*, Alon Efrat†, Eli Packer‡, Valentin Polishchuk*, Leonid Sedov*
*Communications and Transport Systems, ITN, Linköping University, Sweden
†Department of Computer Science, The University of Arizona, USA
‡IBM Research Haifa, Israel

***Abstract*—This paper investigates algorithmic questions related to the possibility of managing UAV traffic with beacon-based navigation, which we dub *BBTM – Beacon-Based Traffic Management*. The specific problem addressed is: How to install the minimum number of beacons in a mountainous terrain to ensure connectivity among a given set of UAS terminals on the terrain? BBTM is relevant for low-cost UAVs operating in remote areas not on time-critical missions, and may also be used as a backup system for better-equipped UAS in case the precise positioning or control information is lost, spoofed or jammed. We give algorithms for the beacon tower placement and evaluate their performance both on synthetic and real-world terrain data; the experiments suggest that our solutions can be used to efficiently quantify costs of establishing direct-visibility routing networks for UAS management.**

I. Introduction

Unmanned aerial vehicles (UAV) operations are picking up, posing the challenge of managing the flow of the new type of air transport. We attend to a fundamental difference between near-future developments of *conventional air traffic management (ATM)* and *UAV traffic management (UTM)* [1]: while the upcoming ATM paradigm hinges on *increasing* aircraft performance and functionalities (with concepts like PBN, SWIM, 4DT, ADS-B, etc. shaping up the vision [2]), the *opposite* trend is clearly picking up in the UAS industry where one (fore)sees the rise of army of mass-produced ubiquitous low-cost autonomous drones launchable by people having no rigorous training in air navigation. Due to these different (in fact, opposing) trends, direct applicability of mainstream ATM research efforts to UTM may be limited; case in point – *separate* theme and tracks for UTM in scientific programs of DASC and other leading forums in aviation.

Nevertheless, it is beneficial for UTM to build up on decades-long experiences earned in ATM and (re)use old working ATM approaches. One of them is beacon-based direct-visibility routing used in the early days of aviation to guide (badly equipped) planes. This paper considers exporting it to management of UAV traffic with beacon-based navigation, which we dub *BBTM – Beacon-Based Traffic Management*. The need to establish guidance and control network for autonomous drones having limited communication and navigation capabilities brings up, in particular, the following algorithmic task:

Given a set of UAV terminals situated in a hilly terrain, find locations on the terrain surface to install (a minimum number of) beacon towers so that between any pair of the terminals there exists a path through the beacons, with the property that consecutive beacons along the path have direct visibility of each other.

A. Motivation

Our focus on the low end of the UAV price spectrum contributes to the potential use cases in yet-to-be-developed areas where the UAS socio-economic potential may be fastest to unleash. While technology advancements make it possible already now to put state-of-the-art communication and surveillance functionalities on every flying robot, less-developed regions would rather crave for cheaper alternatives, possibly with less advanced hard- and software. For one concrete scenario, think about delivery missions (e.g., food and hygiene essentials, mail and packages, archeological artifacts collection bins, everyday supplies, repair work material and such) in rural areas without extensive road network, where low-cost energy-efficient drones can provide boost to the small economies, enable exchange of supplies between local producers, as well as channeling of goods to and from ”mainland” markets [3]. (Naturally, our abstract framework is not limited to rural domains only, but is applicable to urban settings just as well.) To quote Mike Francis, Chief of Advanced Programs & Senior Fellow at the United Technologies Research Center, ”Small UAS are proliferating with these least-capable machines invading the most complex, obstacle-rich environments.” [4]

In particular, we consider the following operational assumptions:

- The UAV’s only navigational capability is to fly towards a beacon in direct line of sight from the drone, i.e., ”home in” on a beacon that is not occluded by the terrain (formally: the segment between the UAV and the beacon does not intersect the terrain). The full flight path of the drone is thus pre-programmed at the origin by specifying the sequence of beacons through which the path goes. That is, the drones are not equipped with precise trajectory control and self-stabilization tools, and rely on the beacons for navigation. (As a by-product, any drone operating under such paradigm, at any time remains visible to at least two beacons; in particular, provided cameras are installed at the beacon towers, it will always be possible to see whether/how the drone is making its way towards the destination, and moreover, provide the flight video to the customer, giving him/her the ”live tracking number”.) BBTM may also serve as a contingency plan for advanced UAVs in case of loss or unavailability of absolute positioning or control signals, which may happen e.g., in confined spaces or near the ground [5].
- The drones are energy-efficient, operating, e.g., on solar power (like, for instance, the unmanned aircraft of Facebook’s initiative internet.org); again, this is relevant for rural-areas applications, where power delivery can be problematic by itself. As a consequence, flying along *shortest* path is not of highest priority in our model: fuel consumption is not an issue, and neither is the duration of flight, since the UAVs perform tasks that are not time-critical (unlike rescue operations, first-aid or disaster relief delivery, military missions, etc.).

In this paper, we present efficient algorithms for placing the beacons, and confirm good performance of the algorithms with the results of computational experiments on synthetic and real-world terrain data. Inter alia, our development provides decision support tools for policy makers to quantitatively assess investments into UAS infrastructure (how many beacons are needed, what height towers, etc.).

B. Related Work

UTM research is booming worldwide; active players include Google, Amazon and NASA with some twenty reports on the subject [6] (see also the recent Nature survey [3] for a technological exposition). Traffic routing paradigms have been previously extensively studied in ATM [7], [8], [9], and naturally (just as we do in this paper), UTM borrows a lot from ATM practice – e.g., one suggestion from Berkeley is the UTM ”parcel” paradigm [10] building upon the currently prevailing ATM idea of sectorizing the airspace and taking responsibility based on the sector ownership.

Using beacons for planning robot motion in 2D polygonal domains is the subject of active current work [11], [12], [13], [14], [15], [16], [17], [18]. Similarly to the model in this paper, the robots are assumed to be simply drawn to a sequence of magnet beacons installed in the domain, and the goal is to place the minimum number of magnets to establish connectivity between points in the domain. Differently from our setting, the magnetic force penetrates the domain boundary, i.e., the robot–beacon (in)visibility is not an issue. Also, this line of work either plans a robot path between only *two* terminals, or aims at establishing connectivity between *any* pair of points in the domain (a continuous set). In contrast, our set S of the terminals is discrete but consists from an arbitrary number of points ($|S| > 2$).

Interestingly, already just determining efficiently the visibility graph on the terminals alone (i.e., even without any issue of beacon placement) is a non-trivial problem, even in a 2D polygonal domain (not on the terrain), even in the case when the domain has no holes (i.e., when it is a simple polygon). Fast algorithms for the problem are given in [19] (but no implementation is reported).

Paths with few turning points (as is relevant for us) are known in computational geometry as *minimum-link* paths (minimizing the number of internal vertices is equivalent to minimizing the number of edges, or *links* in the path). The paths have been a classical subject in the field [20, Ch. 25] and were revisited also in the recent works [21], [22], [23], [24].

C. Overview

The rest of the paper is organized as follows: The modeling is described in Section II. In Section III we prove that our problem is NP-hard (even to approximate). We then turn to positive results. As a warmup, in Section IV we first consider the simple case of connecting just 2 terminals, and then in Section IV-B we present a simple greedy heuristic for the general case $|S| > 2$ considered in the rest of the paper. In Section V we give a mathematical programming formulation to our problem. Section VI presents our main contribution – a heuristic solution for the tower placement. In Section VI-A we report the results of experiments with our heuristics both on synthetic and real terrains. Finally, Section VII draws conclusions and outlines directions for future work.

II. Problem formulation and Notation

Our working environment is a terrain $\mathbb{T}$, representing mountains/hills in a rural area (or buildings in an urban setting). We assume that $\mathbb{T}$ is given as a *Digital Elevation Map* (DEM) that specifies the terrain height at every point of a square grid. (Alternatively, a terrain could be specified by its heights at vertices of a triangulation, with the understanding that at other points it linearly interpolates the heights – such a specification, called *Triangulated Irregular Network*, or TIN, is more common in theoretical investigations of terrains; however in TIN terrains even the most basic of our problems are NP-hard [21].)

We are also given a set $S \subset \mathbb{T}$ of $N = |S|$ UAS terminals on the terrain surface, serving as origins and destinations for the drones. We will not be concerned with the specifics of the terminals – they may represent households, farms, shops, storage facilities, settlements in the rural area, etc.

Finally, we are given a number $h \geq 0$ (beacon tower height) representing the elevation at which a beacon can be installed above the terrain.

A. Visibility graph

We say that two terminals $s, t \in S$ see each other if the segment st does not intersect the terrain. Two non-terminal points $p, q \in \mathbb{T} \setminus S$ see each other if the segments pq does not intersect $\mathbb{T}$, after the segment's endpoints are lifted by h above the terrain. Finally, a terminal $s \in S$ sees a non-terminal point p if sq does not intersect $\mathbb{T}$ after p is lifted by h. The *visibility graph* $G(Q)$ of a set $Q \subset \mathbb{T}$ of points on the terrain has a vertex for every point in Q and an edge between vertices $p, q \in Q$ if they see each other.

B. Problem statement

Our goal is establish connectivity between the terminals by installing a set $B \in \mathbb{T}$ of height-h beacon towers so that for any pair of terminals $s, t \in S$ there exists a path s-q_1-$\ldots$-q_k-t through beacons or other terminals (i.e., $q_i \in S \cup B$) such that s sees the first point q_1, the last point q_k sees t, and any pair of consecutive points q_i, q_{i+1} see each other. In summary, our problem is:

GIVEN: Terrain $\mathbb{T}$, the terminals $S \in \mathbb{T}$, the tower height h.

FIND: A set B of tower locations such that $G(B \cup S)$ is connected.

GOAL: Minimize $|B|$ – the number of beacons installed.

Note the implicit assumption that beacons are installed "for free" in the terminals (at height 0). If two terminals see each other, the UAVs can fly between them directly; more generally, if the visibility graph $G(S)$ of the terminals is connected, no beacon towers need to be installed.

III. The hardness

After the visibility graph $G(\mathbb{T} \cup S)$ is built, our problem reduces to finding the smallest set of vertices $B \subseteq \mathbb{T}$ such that $G(B \cup S)$ is connected. This resembles the minimum Steiner tree problem, except that instead of picking edges into the tree, we are picking *vertices* that induce the connected graph. This, vertex-weighted, version of Steiner tree problem is much harder than the standard Steiner tree problem (in which one is picking *edges*):

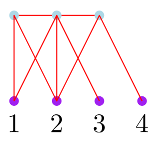


Fig. 1. Purple dots are terminals and blue are possible beacon locations, corresponding to an instance of SetCover with subsets $\{1,2\},\{1,2,3\},\{2,4\}$, over the universe $1,2,3,4$.

Theorem 1. The beacon tower placement for N terminals is NP-hard to approximate to within a factor of $(1-o(1))\ln N$.

Proof. We follow the reduction from [25] that showed hardness of finding optimal node-weighted Steiner tree, and reduce from SetCover. In the SetCover problem the input is a collection of subsets of an N-element universe, and the goal is to pick the smallest number of the subsets that collectively cover the universe (i.e., so that each of the N elements is contained in one of the picked subsets). Given an instance of SetCover, we create an instance of beacon placement as follows: We start from a terrain $\mathbb{T}$ that has height 0 everywhere (i.e., $\mathbb{T}$ is a plateau, of 0 height). Let $\mathcal{S},\mathcal{B}$ be two parallel segments on the terrain. For each element of the universe, place a terminal on $\mathcal{S}$, and for each subset place a point (the potential beacon location) on $\mathcal{B}$; connect each subset to the elements it contains, and connect all beacons by a segment (Fig. 1). Place the obtained graph on the terrain, and dig deep narrow trenches along its edges; also dig even deeper hole at the points where edges of the graph intersect in their interior. Set $h=0$.

Now, there is no use to place beacons anywhere other than at the locations on $\mathcal{B}$ representing the subsets (since beacons placed elsewhere will see at most 1 terminal); moreover, a set of beacons sees all the terminals iff the corresponding subsets cover the universe. The beacons see each other for free, and hence overall the solutions to beacon placement are in one-to-one correspondence with solutions to the SetCover, with the number of beacons equal to the number of subsets in the cover. The theorem follows from the fact that an optimal solution to SetCover on N elements is NP-hard to approximate to within a factor of $(1-o(1))\ln N$ [26]. $\square$

In the remainder of the paper we give positive results (algorithms) for the beacon placement.

IV. Warmup: Connecting 2 terminals

For starters we consider the simple version when $|S|=2$; say S consists of two terminals s and t. Even though this is a very special case of beacon placement, it allows us to describe the general setup that will be used for the general version, decide on the density of potential tower locations and suggest a simple greedy solution to the general case with $|S|>2$.

The case of 2 terminals reduces to finding shortest (fewest-edges) s-t path in the visibility graph $G(\mathbb{T}\cup s\cup t)$; locations of the internal nodes of the path define the optimal set B of towers. The found path is the *minimum-link* s-t path, and the number of edges (links) in the path is called the *link distance* between s and t.

A. Sparsification

A characteristic feature of link distance is that in many cases, the internal vertices of a minimum-link path may be moved around while still keeping the connectedness of the path; in other words, shifting the path vertices does not influence the link length of the path (in contrast to the Euclidean metric – moving any vertex changes the path's Euclidean length). This suggests a possibility to use a sparsified set $P\subset\mathbb{T}$ of potential tower locations, instead of the full terrain grid $\mathbb{T}$, to adequately represent link distance between points on the terrain.

We experimented with different sparsification levels of the terrain grid $\mathbb{T}$ by generating 10000 random pairs of locations $s,t\in\mathbb{T}$ and computing, for each pair, the minimum-link path through towers installed at 10 sparser grids. Specifically, for each $k=1\dots10$ we defined P_k as the grid that takes only every kth point from $\mathbb{T}$, and found the shortest s-t path in the visibility graph $G(P_k\cup s\cup t)$.

We ran the experiments on both synthetic terrains and real terrains. A 100 of random synthetic terrains were created by erecting walls from a random set of segments in the plane and smoothing the walls with Gaussians; refer to [27, Section 2] for the details (the paper also provides the link to terrain generation code). A real terrain was obtained by image processing (as described in [27, Section 2]).

Figure 2 shows the dependence of the number of links in the solution on the sparsification level. It can be seen that keeping only every fifth grid point

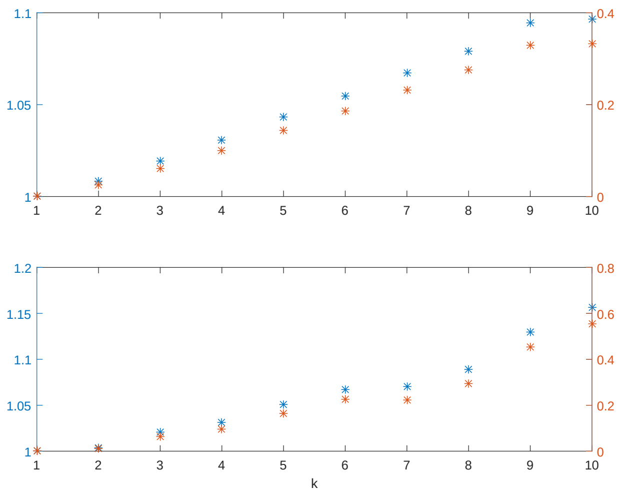


Fig. 2. Left y-axis (blue): The average of the number of links through the grid P_k divided by the number of links through $P_1 = \mathbb{T}$ (the true optimal link distance). Right y-axis (red): The average difference between link distance in P_k and P_1. Top: synthetic terrains. Bottom: real terrain.

maintains a good approximation of the link distance: on average the distance increases by around 5% and less than .17 links (moreover, in about 85% of cases the link distance in P_5 is the same as in $P_1 = \mathbb{T}$). Therefore in what follows we will use the sparsified grid P_5 for the potential tower locations; to avoid the index we will denote P_5 just by P.

B. *An iterative solution*

Algorithm 1:

Input : A set S of terminals and a set P of candidate tower locations

Output: A subset $B \subseteq P$ of beacon towers, connecting all the terminals

```
1 $B \leftarrow \emptyset$;
2 $C \leftarrow$ connected components of the visibility
  graph $G(S \cup B)$ on terminals alone;
3 while $|C| > 1$ do
4     $(c, c') \leftarrow$ furthest pair of components in $C$;
5     $b^* \leftarrow$ beacons from $P \setminus B$ connecting $c$ to
      $c'$ with minimum-new-beacons path;
6     $B \leftarrow B \cup b^*$;
7     $C \leftarrow$ connected components of $G(S \cup B)$;
8 end
```

A straightforward way to solve the problem for the general case $|S| > 2$ is to first connect the farthest initially-disconnected terminals (farthest in terms of link distance) with minlink path, and then add the terminals to the current network step-by-step, using the already existing towers for free. The pseudocode for this greedy solution is given in Algorithm 1. Throughout the algorithm we maintain the set C of connected components of the visibility graph $G(S \cup B)$ on the terminals plus the already chosen beacon locations B; initially B is empty and $G(S \cup B) = G(S)$ is the visibility graph on the terminals alone (that is, each component $c \in C$ is a subset $c \subseteq S$ of terminals that can communicate with each other even without any beacons). We merge the components of C iteratively, connecting, in each iteration, the two farthest components c and c', i.e., the components that require installing the maximum number of new towers to be connected (if there is more than one pair with the same link distance, the tie is broken arbitrarily). The new towers are added to B, and the set of connected components is updated. (Note that in addition to merging c and c', other components may become connected "for free" thanks to the new beacons b^*.) The link distance between the components and the updates of the set C are both performed with the help of the visibility graph $G(S \cup P)$, in which the terminals $c \subseteq S$ and the beacons $b_c \subseteq P$ that belong to one component $c \in C$ are replaced by a single vertex c.

Figure 3 shows several iterations of the algorithm on a synthetic terrain.

V. An Integer Program

We can formulate our problem as a multicommodity-flow-based integer program (IP) that has the two sets of variables:

- a variable f_{ij}^{st} for every edge ij of the visibility graph $G(S \cup P)$ and every pair s, t (with $s < t$) of terminals. The variable represents the amount of s-t flow on the edge ij,
- a binary variable x_p for every candidate tower location $p \in P$. The variable indicates whether the tower is installed at p.

The constraints are:

- the flow conservation constraints (2),
- the out- and in-flow constraints (3) and (4) for each commodity (ensuring that every pair of

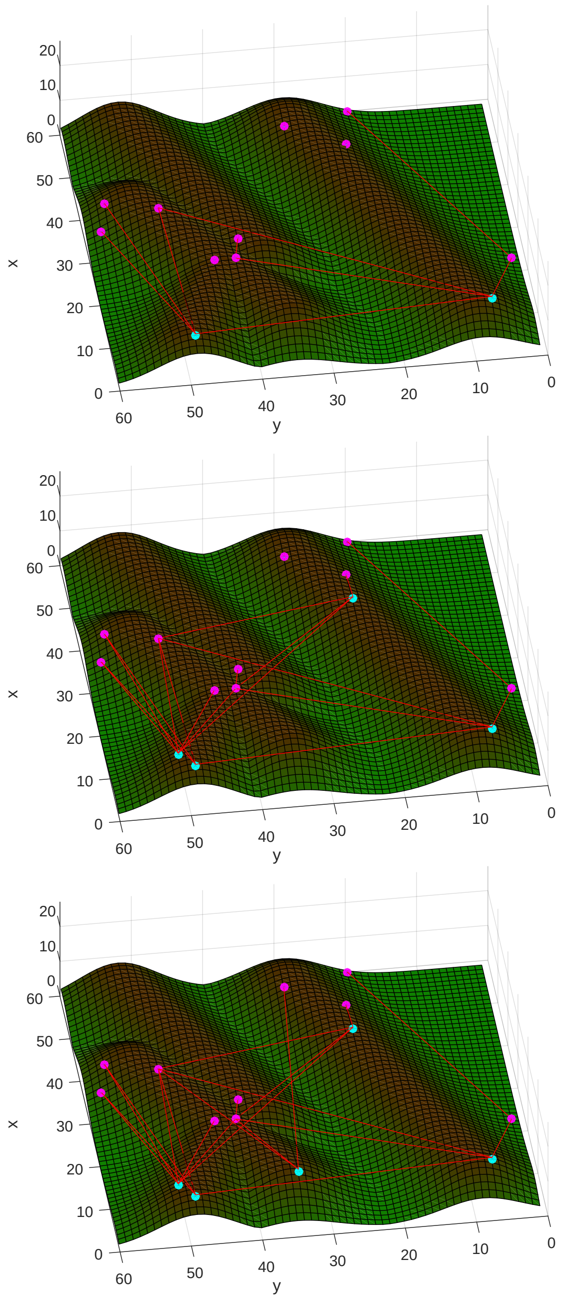


Fig. 3. Top to bottom: The terminal groups are connected one-by-one.

terminals is connected, by sending a unit of flow between the pair) and

- the constraints (5) ensuring that $x_p = 1$ if a path between some terminals uses p (M is a number larger than the maximum possible total flow through a vertex, e.g., $N(N+1)/2$).

The objective is to minimize the number of towers, and the full program looks as follows:

$$\min \sum_{p \in P} x_p \qquad \text{s.t.} \tag{1}$$

$$\sum_{i:ip \in G} f_{ip}^{st} = \sum_{j:pj \in G} f_{pj}^{st} \quad \forall p \in P,\ \forall s,t \in S : s < t \tag{2}$$

$$\sum_{i:it \in G} f_{it}^{st} = \sum_{j:sj \in G} f_{sj}^{st} = 1 \qquad \forall s,t \in S : s < t \tag{3}$$

$$f_{is}^{st} = f_{tj}^{st} = 0 \qquad \forall i,j,\ \forall s,t \in S : s < t \tag{4}$$

$$M x_p \geq \sum_{s,t,i:ip \in G} f_{ip}^{st} \qquad \forall p \in P \tag{5}$$

$$x_p \in \{0,1\} \qquad \forall p \in P \tag{6}$$

While with the increasing computational power one may hope to use the IP to solve our problem optimally, we failed to produce an IP solution with off-the-shelf software (AMPL + CPLEX) for reasonable-size problems.

VI. A HEURISTIC SOLUTION

Our heuristic for the beacon placement is inspired by the approximation algorithm from [25] for the node-weighted Steiner tree. The pseudocode for the heuristic is given in Algorithm 2. Similarly to the greedy solution, in the preprocessing step we compute the visibility graph $G(S \cup P)$; the graph is used to find minimum-(new-)tower paths between connected components of $G(S \cup B)$, i.e., groups of terminals plus existing tower locations that can communicate with each other. (The preprocessing step is not shown in Algorithm 2.)

As in the greedy solution, we initialize by computing the set C of connected components of the visibility graph on the terminals (lines 1–2). Our goal is to connect all components of C into a single

Algorithm 2:

```
Input : A set $S$ of terminals and a set $P$ of candidate tower locations
Output: A tree $T$, on $S$ and a subset $B \subseteq P$ of beacon towers, connecting all the terminals
1  $B \leftarrow \emptyset$;
2  $C \leftarrow$ connected components of the visibility graph $G(S \cup B)$ on terminals alone;
3  while $|C| > 1$ do
4      foreach $b \in P \setminus B$ do
5          $\mathcal{T} \leftarrow$ the shortest paths from $b$ to all components in $C$;
6          Sort $\mathcal{T}$ by increasing number of beacons from $P \setminus B$ in the paths;
7          $f \leftarrow \infty$;
8          $\mathcal{T}' \leftarrow \emptyset$;
9          $i \leftarrow 1$;
10         repeat
11             $f' \leftarrow f$;
12             $\mathcal{T}' \leftarrow \mathcal{T}' \cup \mathcal{T}[i]$;
13             $Q \leftarrow$ the set of yet unchosen beacons that are used in $\mathcal{T}'$;
14             $f \leftarrow \frac{1+|Q|}{i}$;
15             $i \leftarrow i + 1$;
16         until $i \leq size(\mathcal{T})$ and $f < f'$;
17         $b.score \leftarrow f$; $b.Q \leftarrow Q$;
18     end
19     $b^* \leftarrow b \in P \setminus B$ whose $b.score$ is minimum;
20     $B \leftarrow B \cup b^*.Q$;
21     $C \leftarrow$ connected components of $G(S \cup B)$;
22 end
23 $T \leftarrow$ a minimum spanning tree of $S \cup B$
```

component by picking a set B of beacons from the set P of potential tower locations. We merge the components of C iteratively, and the iterations are the main loop of our algorithm (lines 3–22).

In each iteration we consider all beacons $P \setminus B$ that have not been selected so far (lines 4–18). For each such beacon b, we find the shortest paths from b to all components in C (line 5), where the length of a path is defined as the number of yet unchosen tower locations in it (i.e., a path may include both already chosen and not chosen towers, but only the latter towers contribute to the path's length). We sort the paths by increasing length (line 5), and denote this sorted set by $\mathcal{T}$. We consider the paths in $\mathcal{T}$ in its order (lines 10–16), evaluating the expression $f = \frac{1+|Q|}{i}$ when the ith path from $\mathcal{T}$ is considered (i.e., i counts the number of the connected components from C that would be merged if one picks the towers in the first i paths from $\mathcal{T}$); here $Q = (\mathcal{T}[1] \cup \mathcal{T}[2] \cup \cdots \cup \mathcal{T}[i]) \setminus S \setminus B$ is the set of new, (unchosen in the previous iterations) beacons in the first i paths from $\mathcal{T}$. We continue adding the paths from $\mathcal{T}$ (i.e., continue increasing i) as long as f decreases or until all components are merged; in either case, we save the score f and the set of newly chosen beacons Q (line 17).

After going through all beacons in the above manner, we pick a beacon with the smallest f (if several beacons have the same score, the tie is broken arbitrarily), add the corresponding beacons Q to the set of chosen beacons B (line 20), and merge the corresponding components in C (line 21). This ends one iteration.

Finally, we postprocess the result to potentially improve it by finding a minimum spanning tree in the graph on the terminals and the chosen beacons (line 23); this keeps the connectivity but possibly decreases the visibility distance (link lengths), which improves the path quality. (Alternatively one could compute the bottleneck spanning tree where the longest edge is as short as possible.)

A. Experimental results

Figure 4 presents output of the heuristic on a synthetic terrain and two real terrains. We experimented with a large dataset of synthetic terrains (produced as in Section IV), by generating random terrains with various sizes (number of pixels). The experiments were carried out on a single IBM machine with 16GB RAM, processor Intel(R) Core (TM i7-3740QM CPU @ 2.70GHz), and 64-bit operating system.

Figure 5 shows the dependence of computation time on the number of pixels in the terrain. We believe, however, that the large number of pixels of the terrain might not alone imply that the beacon placement in the terrain will be hard. Indeed, on a flat terrain (plateau) where any two points see each other, tower placement is void (no matter how many pixels there are in the terrain). On the other

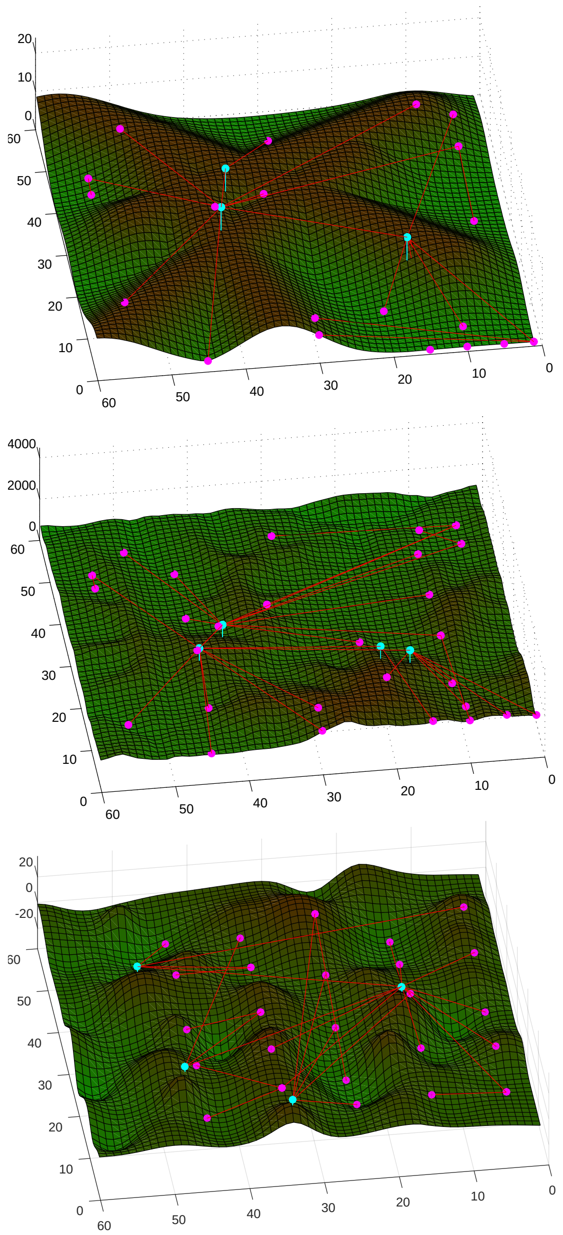


Fig. 4. The terminals (purple dots) are connected via towers (lightblue) and other terminals. Top: A synthetic terrain. Middle: A real terrain somewhere in Arizona (around 36.1°N 112.1°W). Bottom: A real terrain somewhere in Italy (data courtesy of authors of [28]).

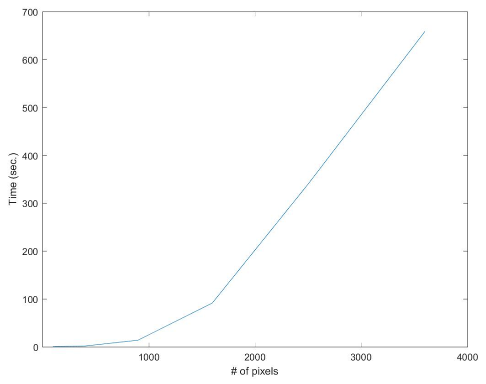
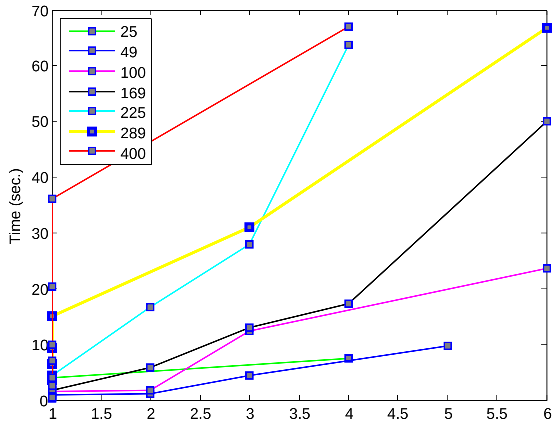


Fig. 5. The running time for synthetic terrains as a function of the number of pixels.

hand, a terrain with a not-too-many pixels may have a complicated visibility structure, making tower placement a challenging task. We therefore looked at the *number of towers in the output* as a factor defining the runtime (the output size, in a sense, measures the "complexity" of an instance – how many beacons are needed in order to connect the terminals). We varied the number of terminals N placed in the terrain, and recorded the running time as a function of the number of beacons (the size of the output), both for synthetic and real terrains (in addition to the terrains shown in Fig. 4, we experimented also with two more Italian terrains from [28]).

Figure 6 summarizes the results (synthetic terrains used in this experiment were mid-size – 40x40). The results confirm that when more towers are needed, it takes longer to place them. It can be also seen that our approach is practical: solutions for both synthetic and real terrains were produced within couple of minutes of computation, even for as many as $N = 400$ terminals. We also tried synthetic terrains as large as 100x100 pixels (Fig. 7) – the solution was found within few hours.

To further investigate the dependence of the running time on the number of beacons in the output, we also ran experiments with changing h (the height of the beacon tower). The parameter h influences the number of towers indirectly – for higher h, one would expect fewer beacons are needed. Our experiments confirm this intuition: as shown in Figure 8, for any N, the running time very



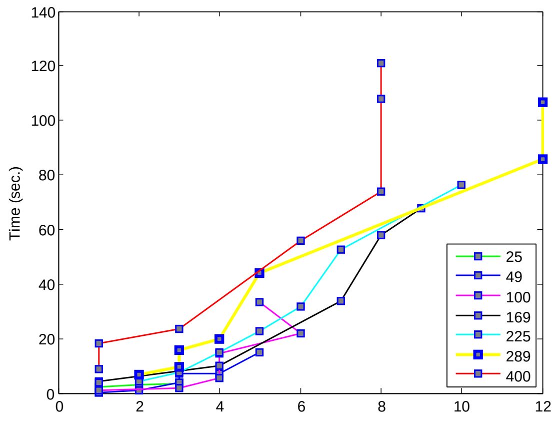


Fig. 6. Number of towers found (output size) and runtimes of our code (in seconds/instance) for different N. Left is synthetic, and right is real terrains.

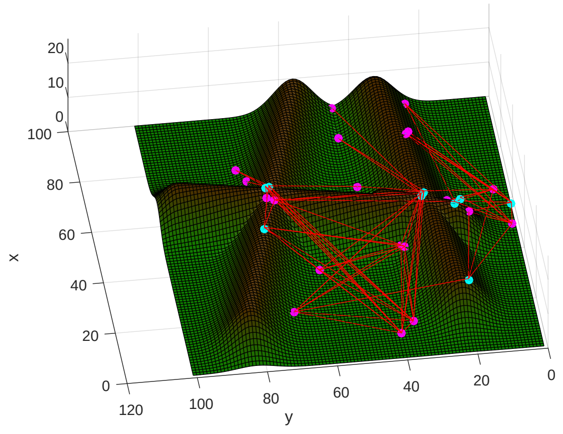


Fig. 7. The solution on a 100x100 synthetic terrain.

clearly monotonically decreases with increasing h.

VII. Conclusion and Discussion

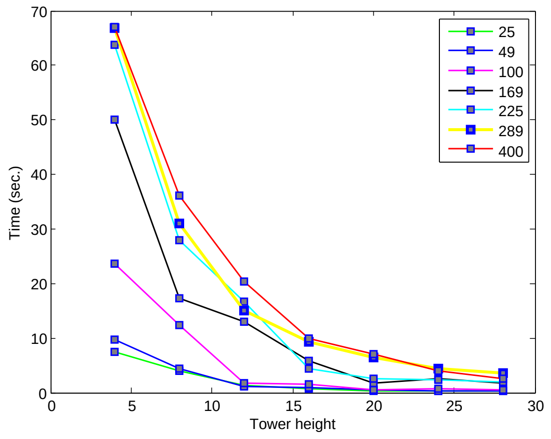
We presented an efficient heuristic for placing beacon towers in a terrain to establish connectivity between UAV terminals under the introduced BBTM paradigm. We also experimented with a greedy solution whose simplicity may come at the expense of reduced solution quality, and with an IP approach which gives optimal solutions but can handle only small problem instances. Many extensions and directions for future work are possible; some are outlined below.

a) Visibility distance: We did not have any upper bound on the distance between visible points, which may be unrealistic. Such a bound can be easily assumed within our framework by removing the overly long edges from the visibility graph.

b) Tower heights: In our model, all towers had the same given height h. Our approach still applies if different-height towers can be installed – we would just have a node in the visibility graph for every possible height. More interestingly would be to stay within the uniform-height framework and to find "threshold" values for h, at which the solution size $|B|$ changes (it does not make sense to use towers of non-threshold heights since for every non-threshold height there exists a lower threshold height providing a same-size solution).

c) Higher connectivity: A standard way to increase robustness of interconnection is to require that the network is K-connected for some $K > 1$. However, the standard graph-theoretic K-connectivity is a wrong requirement for BBTM (Fig. 9). Instead, another measure could be used for the network robustness: resilience to "geographically correlated attacks" [29] – a topic for future research.

d) Path complexity: One may introduce an upper bound M on the total number of intermediate beacons that the UAV traverses when flying through our network from one terminal to another. (Note that if the link distance between some terminals is larger than M, the instance of the problem becomes infeasible; the infeasibility, however, may be detected by running our algorithm for minimum-link path from Section IV for every pair of the terminals.) Handling such a constraint is an open problem.



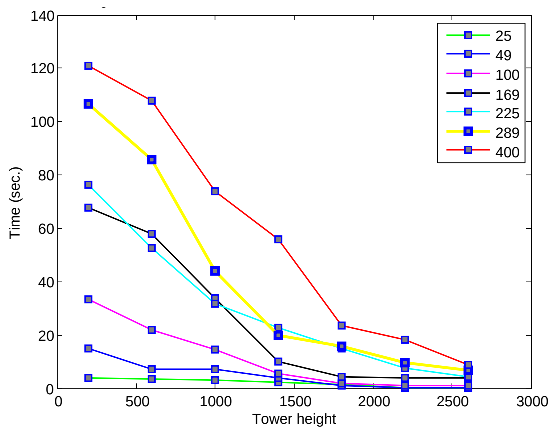


Fig. 8. Runtimes of our code (in seconds/instance) as a function of h for different N. Left is synthetic, and right is real terrains.

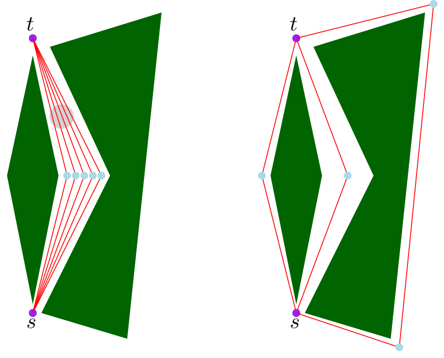


Fig. 9. K-connectivity may not be the best measure of robustness. Lightblue dots are beacon towers, darkgreen polygons are obstacles (breaking the direct visibility), red lines are UAV paths. Left: 5 s-t paths running close to one another provide 5-connectivity, but they can all be killed by a single small atmospheric event (lightgray disk). Right: 3 paths make s and t less connected in the graph-theoretic sense, but more robust to failures.

VIII. Acknowledgements

This research is part of the UTM-OK project supported by the Swedish Transport Administration (Trafikverket). We thank participants of the 2nd IBM Workshop on Geometric Problems in Sensor Networks and Robotics for discussions.

References

[1] J. Geller, T. Jiang, D. Ni, and J. Collura, “Traffic management for small unmanned aerial systems,” in *Transportation Research Board 95th Annual Meeting*, 2016.

[2] SESAR, “European ATM Master Plan,” 2012.

[3] D. Floreano and R. J. Wood, “Science, technology and the future of small autonomous drones,” *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.

[4] M. Francis, Accessed May 4, 2016. [Online]. Available: https://www.linkedin.com/company/aiaa

[5] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. A. Chatzichristofis, L. Kneip *et al.*, “Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in gps-denied environments,” *Robotics & Automation Magazine, IEEE*, vol. 21, no. 3, pp. 26–40, 2014.

[6] NASA. [Online]. Available: http://utm.arc.nasa.gov/

[7] J. Krozel, J. S. B. Mitchell, A. Pääkkö, and V. Polishchuk, “Throughput/complexity tradeoffs for routing traffic in the presence of dynamic weather,” *International Conference on Research in Air Transportation*, 2010.

[8] J. Krozel, J. Mitchell, V. Polishchuk, and J. Prete, “Airspace capacity estimation with convective weather constraints,” in *AIAA Guidance, Navigation, and Control Conference*, 2007.

[9] S. Yang, J. S. B. Mitchell, J. Krozel, V. Polishchuk, J. Kim, and J. Zou, “Flexible airlane generation to maximize flow under hard and soft constraints,” *Air Traffic Control Quarterly*, vol. 19, no. 3, p. 211, 2011.

[10] A. G. Foina, C. Krainer, and R. Sengupta, “An unmanned aerial traffic management solution for cities using an air parcel model,” in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*. IEEE, 2015, pp. 1295–1300.

[11] M. Biro, J. Gao, J. Iwerks, I. Kostitsyna, and J. S. B. Mitchell, “Beacon-based routing and coverage,” *21st Fall Workshop on Computational Geometry (FWCG 2011)*, 2011.

[12] S. W. Bae, C.-S. Shin, and A. Vigneron, “Tight bounds for beacon-based coverage in simple rectilinear polygons,” in *LATIN 2016: Theoretical Informatics*. Springer, 2016, pp. 110–122.

[13] M. Biro, J. Gao, J. Iwerks, I. Kostitsyna, and J. S. B. Mitchell, “Combinatorics of beacon-based routing and coverage,” *Proc. the 25th Canadian Conf. Comput. Geom.(CCCG 2013)*, vol. 1, no. 2, 2013.

[14] M. Biro, “Beacon-based routing and guarding,” Ph.D. dissertation, Stony Brook University, 2013.

[15] M. Biro, J. Iwerks, I. Kostitsyna, and J. S. B. Mitchell, “Beacon-based algorithms for geometric routing,” in *Algorithms and Data Structures*. Springer Berlin Heidelberg, 2013, pp. 158–169.

[16] T. C. Shermer, “A combinatorial bound for beacon-based routing in orthogonal polygons,” in *Proceedings of the 27th Canadian Conference on Computational Geometry, CCCG 2015, Kingston, Ontario, Canada, August 10-12, 2015*, 2015.

[17] B. Kouhestani, D. Rappaport, and K. Salomaa, “On the inverse beacon attraction region of a point,” in *Proceedings of the 27th Canadian Conference on Computational Geometry, CCCG 2015, Kingston, Ontario, Canada, August 10-12, 2015*, 2015.

[18] M. Biro, J. Gao, J. Iwerks, I. Kostitsyna, and J. S. B. Mitchell, “Beacon-based structures in polygonal domains,” *1st Computational Geometry: Young Researchers Forum (CG: YRF 2012)*, 2012.

[19] B. Ben-Moshe, O. Hall-Holt, M. J. Katz, and J. S. B. Mitchell, “Computing the visibility graph of points within a polygon,” in *Proceedings of the Twentieth Annual Symposium on Computational Geometry*. New York, NY, USA: ACM, 2004, pp. 27–35.

[20] J. E. Goodman and J. O’Rourke, *Handbook of Discrete and Computational Geometry*, ser. CRC Press series on discrete mathematics and its applications. Chapman & Hall/CRC, 2004.

[21] I. Kostitsyna, M. Löffler, V. Polishchuk, and F. Staals, “On the complexity of minimum-link path problems,” in *Symposium on Computational Geometry*, 2016.

[22] J. S. B. Mitchell, V. Polishchuk, and M. Sysikaski, “Minimum-link paths revisited,” *Computational Geometry Theory and Applications*, pp. 19–22, 2014.

[23] J. S. B. Mitchell, V. Polishchuk, M. Sysikaski, and H. Wang, “An optimal algorithm for minimum-link rectilinear paths in triangulated rectilinear domains,” in *Automata, Languages, and Programming*. Springer Berlin Heidelberg, 2015, pp. 947–959.

[24] V. Polishchuk and M. Sysikaski, “Faster algorithms for minimum-link paths with restricted orientations,” *Algorithms and Data Structures*, pp. 655–666, 2011.

[25] P. N. Klein and R. Ravi, “A nearly best-possible approximation algorithm for node-weighted Steiner trees,” *J. Algorithms*, vol. 19, no. 1, pp. 104–115, 1995. [Online]. Available: http://dx.doi.org/10.1006/jagm.1995.1029

[26] I. Dinur and D. Steurer, “Analytical approach to parallel repetition,” in *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, D. B. Shmoys, Ed. ACM, 2014, pp. 624–633.

[27] A. Efrat, M. Nikkilä, and V. Polishchuk, “Sweeping a terrain by collaborative aerial vehicles,” in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 4–13.

[28] P. Magillo, L. D. Floriani, and F. Iuricich, “Morphologically-aware elimination of flat edges from a TIN,” in *21st SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2013, Orlando, FL, USA, November 5-8, 2013*, C. A. Knoblock, M. Schneider, P. Kröger, J. Krumm, and P. Widmayer, Eds. ACM, 2013, pp. 244–253.

[29] P. K. Agarwal, A. Efrat, S. K. Ganjugunte, D. Hay, S. Sankararaman, and G. Zussman, “The resilience of WDM networks to probabilistic geographical failures,” *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1525–1538, 2013. [Online]. Available: http://dx.doi.org/10.1109/TNET.2012.2232111