Optimal placement of communications relay nodes

Oleg Burdakov[a,1], Patrick Doherty[b], Kaj Holmberg[a], Per-Magnus Olsson[b]

[a] *Department of Mathematics, Linköping University, SE-581 83 Linköping, Sweden*
[b] *Department of Computer and Information Science, Linköping University, Sweden*

Abstract

We consider a constrained optimization problem with mixed integer and real variables. It models optimal placement of communications relay nodes in the presence of obstacles. This problem is widely encountered, for instance, in robotics, where it is required to survey some target located in one point and convey the gathered information back to a base station located in another point. One or more unmanned aerial or ground vehicles (UAVs or UGVs) can be used for this purpose as communications relays. The decision variables are the number of unmanned vehicles (UVs) and the UV positions. The constraints are determined by, firstly, a free line of sight requirement for every consecutive pair in the chain and, secondly, a limited communication range. Because of these requirements, our constrained optimization problem is a difficult multi-extremal problem for any fixed number of UVs. Moreover, the feasible set of real variables is typically disjoint. We present an approach that allows us to efficiently find a practically acceptable approximation to a global minimum in the problem of optimal placement of communications relay nodes. It is based on a spatial discretization with a subsequent reduction to a shortest path problem. The case of a restricted number of available UVs is also considered here. We introduce two label correcting algorithms which are able to take advantage of using some peculiarities of the resulting restricted shortest path problem. The algorithms produce a Pareto solution to the two-objective problem of minimizing the path cost and the number of hops. We justify their correctness. The presented results of numerical 3D experiments show that our algorithms are superior to the conventional Bellman-Ford algorithm tailored to solving this problem.

Keywords: OR in telecommunications; Unmanned vehicles; Global optimization; Hop-restricted shortest paths; Label correcting algorithms

1 Introduction

In this paper, we consider the following optimization problem originating from optimal placement of communications relay nodes.

Given a set $X \subset R^n$ and two points s and t in X. One must choose an optimal number of points, say k, in the ordered sequence point $1, \text{point } 2, \ldots, \text{point } k$. We consider separately the two cases: unrestricted and restricted k. The points are to be placed in X in an optimal way subject to some constraints. The position of point i, denoted as x_i, is

[1]Corresponding author. Tel.: +46 13 281473. *E-mail address:* olbur@mai.liu.se

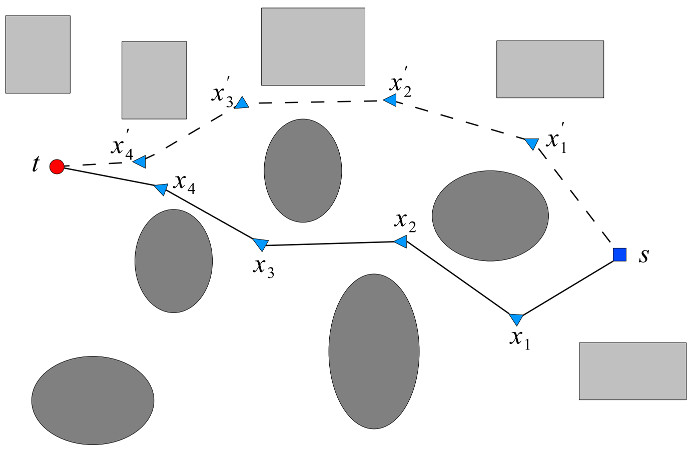

Figure 1: Communications relay

assessed by a merit function $f(x_i)$, and the sum $f(x_1) + \ldots + f(x_k)$ is to be minimized. The point placement should meet the following requirements, in which we denote $x_0 = s$ and $x_{k+1} = t$. Any consecutive pair $(i, i+1)$ in the sequence of points should be placed so that all points of the linear segment

$$[x_i, x_{i+1}] = \{x \in R^n : \ x = \alpha x_i + (1-\alpha)x_{i+1}, \ \alpha \in [0,1]\}$$

belong to the set X. Moreover, it is required that the Euclidean distance $\|x_{i+1} - x_i\|$ does not exceed a given radius $r > 0$.

The described optimization problem can be formulated as follows.

$$\min_{k \in \{0,1,2,\ldots\}} F(k), \tag{1}$$

where the objective function

$$F(k) = \begin{array}{ll} \displaystyle\min_{x_1,\ldots,x_k \in R^n} & \displaystyle\sum_{i=1}^{k} f(x_i) \\ \text{subject to:} & [x_i, x_{i+1}] \subset X, \ i = 0, 1, \ldots, k, \\ & \|x_{i+1} - x_i\| \le r, \ i = 0, 1, \ldots, k, \\ & x_0 = s, \ x_{k+1} = t. \end{array} \tag{2}$$

Our interest in this problem is motivated by the communications relay problems common, for instance, in robotics [2, 3, 4, 13, 14, 16, 19, 25, 28, 29, 30, 31], where it is required to survey some target located in t and convey the gathered information back to a base station located in s. One or more unmanned aerial or ground vehicles (UAVs or UGVs) are used as relays. The set X is defined by the terrain within the area of interest. It is typically the area of interest with removed obstacles which could be, e.g., buildings, hills or mountains. In Fig. 1, X is the white area. It is assumed that the available unmanned vehicles (UVs) are equipped with appropriate sensors to survey the target and also with means to communicate with each other and the base station. We consider here the case when UVs form a chain over which the communication is relayed.

The optimal number of UVs, k, and their optimal positions, $x_1, \ldots, x_k$, are to be determined subject to constraints of the following two types.

First, the communication between any consecutive pair of UVs in the chain can only take place if there is a free line of sight between them (see Fig. 1). This requirement is modeled in (2) as $[x_i, x_{i+1}] \subset X$.

Second, in any consecutive pair, the UVs are not further away from each other than the range of the communication equipment which is characterized by the communication radius $r > 0$. As the range of the equipment is limited, it forces the use of intermediary relay UVs to convey the information back to the base station if the distance between the target and the base station is longer than the communication range. It is assumed, for simplicity, that the range of the base station communications equipment is the same as the UAV communication range. The second requirement justifies the presence of the inequality $\|x_{i+1} - x_i\| \leq r$ in (2).

The function $f(x_i)$ assessing the UV position x_i can be defined in various ways. In Section 2, we suggest to use an obstructed volume as a merit function, whose value is determined by the local terrain around x_i.

Mixed integer programming problems, like (1)–(2), are known to be very difficult to solve [7, 23]. In our case, the difficulties originate not only from the presence of an integer variable, but mainly because of the multi-extremal nature of the continuous optimization sub-problem (2). Moreover, the feasible set in (2) is typically disjoint, in which case there exists at least one couple of feasible points $x = (x_1, \ldots, x_k)$ and $x' = (x'_1, \ldots, x'_k)$ in the kn-dimensional space $R^n \times \ldots \times R^n$ such that any continuous path between them contains infeasible points. The number of disjoint subsets of the feasible set is in practice extremely large. This feature is illustrated in Fig. 1. One can see that there is no *feasible* continuous variation of variables that would be able to transform the feasible sequence of points $x = (x_1, \ldots, x_4)$ to $x' = (x'_1, \ldots, x'_4)$. In this example, there exist a large number of feasible sequences $x = (x_1, \ldots, x_4)$ which are pair-wise disjoint in this sense.

The aim of this paper is to develop an approach that would allow us to efficiently find a reasonably accurate approximation to a global minimum in the problem of optimal placement of communications relay nodes.

The main practical significance of our approach is that it allows for finding such a placement in real time mode almost immediately after the target position becomes available. This is achieved by splitting the solution process into the following two stages.

At the presolving stage, an a priori available information about the terrain and the location of the base station is processed as completely as possible. Thus, the major computational efforts are associated with this stage. This makes vanishing the computational cost of the second stage, at which the information about the target position is used.

Our approach is based on a discretization of the set X and a reformulation of our problem as a shortest path problem (see, e.g., [1]) which is known to be solved in a polynomial time of the number of discretization nodes. To avoid confusion, we shall call it a *cheapest path problem*, because both path cost and its length (number of hops) will be considered. We introduce a spatial discretization and present a network formulation in Section 3. The discretization as well as the network problem generation and its solving for every possible discrete target position can all be done at the presolving stage.

In practice, the number of communications relay nodes to be optimally placed is often restricted by some number $K > 0$. This requirement leads to the following reformulation

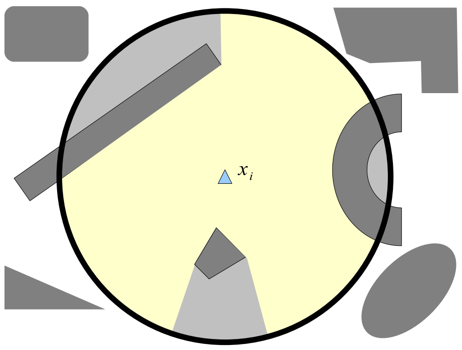

Figure 2: Visible and invisible parts of a ball centered in x_i

of problem (1)

$$\min_{k \in \{0,1,\ldots,K\}} F(k). \tag{3}$$

In our network formulation considered in Section 4, this problem corresponds to a cheapest path problem with *restricted* number of hops. In practice, it may be necessary to solve this problem repeatedly for every new target position and number of available UVs. Therefore, in Section 4, we address also the more general problem of finding optimal placement of a limited number of communications relay nodes for every possible discrete target position. It is reduced to the *all hops optimal path* (AHOP) problem [21]. The consideration of this more general problem allows us to move the major computational burden on the mentioned presolving stage.

The conventional Bellman-Ford algorithm is widely used for finding a tree of *unrestricted* cheapest paths. It has been noticed, e.g., in [1, 23] that the labels generated by this algorithm contain all AHOP solutions. In Section 5, we present the Bellman-Ford algorithm tailored in [23] to solving AHOP problems. We then introduce two new and considerably more efficient label correcting algorithms for solving the same problems and prove their correctness. They can be viewed as modifications of the algorithm of [23]. Their efficiency results from using some peculiarities of the UV-based communications relay problem. The important feature of these algorithms is that for each possible target position they produce a Pareto solution. It is an optimal solution to the multi-criteria problem of minimizing both the path cost and the number of hops. We are interested in minimizing the number of hops because this minimizes the number of UVs required for maintaining the communications relay.

Results of our numerical experiments are presented in Section 6. The considered test problems originate from UAV applications with various 3D terrain topologies. The number of discretization points varies from medium to very large. The experiments show that our algorithms are superior to the Bellman-Ford-type algorithm of [23].

In Section 7, we draw conclusions and discuss future work.

2 Use of obstructed volume for assessing UV position

For our UV applications, we suggest to define the function $f(x_i)$ assessing the UV position x_i as the volume of obstacles and their ‘shade’ within the ball $\{x \in R^3 : \ \|x - x_i\| \leq r'\}$.

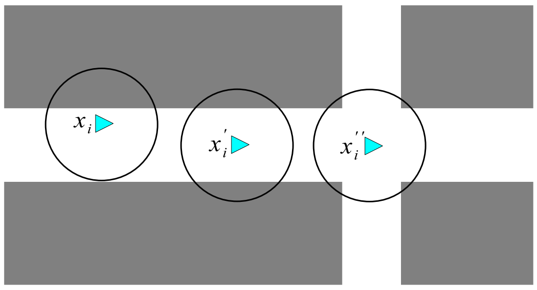

Figure 3: Obstructed volume for three UV positions

Here the radius r' may be either the same as, or different from, the communication radius r. In other words, $f(x_i)$ is the volume of the part of the ball which is invisible from the point x_i (see Fig. 2). Obviously, $f(x_i) = 0$ means that there is no obstructed volume within the ball, and the maximal value of $f(x_i)$ is equal to the volume of the ball. Our desire to find a UV location x_i, which makes the obstructed volume $f(x_i)$ as small as possible, implies maximization of the visible volume of the sphere.

Our choice of $f(x_i)$ is closely related to the concept of an *isovist* introduced in [36]. An isovist, or viewshed, is the area in a spatial environment directly visible from a given point. In our applications this area is restricted by a ball. Isovist is widely used in architectural studies, geoinformation science, computational geometry and computer graphics. It is successfully applied in the problems of line of sight communication, VLSI circuits design, robot and sensor network design, motion planning, architectural and urban planning, computer games etc. Since some of them admit problem formulations similar to (1)–(2), they can be viewed as potential areas for extending the approach presented in this paper.

There exist efficient algorithms that can be used in our applications for computing the obstructed volume $f(x_i)$ (see, e.g., [6, 17, 26, 33, 37]).

With our suggested choice of $f(x_i)$, positions x_i well distant from obstacles are favored over those which are closer to obstacles. This ensures that the better centered points are of preferable choice for placing UVs. Fig. 3 presents an example in which $f(x_i) > f(x_i') > f(x_i'')$, and therefore, the position x_i'' is the most preferable among the considered three alternatives. In what follows, we do not use any specific feature of $f(x_i)$ and assume that it is just a given function.

3 Spatial discretization and network formulation

For the purpose of solving problem (1)–(2) approximately, let us restrict our choice of placing $x_1, \dots, x_k$ by a discrete set of points $D \subset R^n$. This can be done, for instance, by introducing a grid in the area of interest. Then the minimization in (2) is performed over $x_1, \dots, x_k \in D$.

The introduced discretization, in itself, does not make any considerable simplification of problem (1)–(2). Since it still looks intractable, we will construct a network by taking advantage of using such characteristic features of problem (2) as the additive objective function and chain-type constraints. Our network, i.e. a weighted directed graph, will be

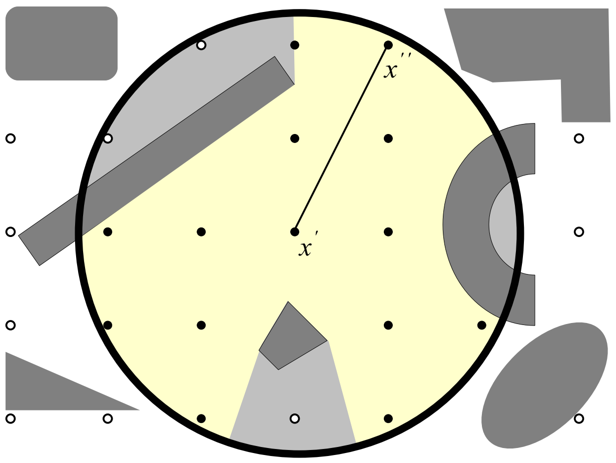

Figure 4: Spatial discretization (the set $D \cap X$ is presented by the dots $\circ$ and $\bullet$). The points x' and x'' are intervisible and meet the limited distance requirement. Nodes $\bullet$ are adjacent to x' in the visibility graph.

constructed on the base of an undirected graph.

To formalize our network problem, let us regard the discrete points $D \cap X$ as nodes. We denote the set of nodes by N. For simplicity, the same notation will sometimes be used for both points and nodes.

Let the set $E \subset N \times N$ be composed of the couples $x', x'' \in D \cap X$ which both are *intervisible*, i.e. $[x', x''] \subset X$, and meet the limited distance requirement $\|x' - x''\| \leq r$. The set of nodes N and the set of undirected edges E define the *visibility graph* $G = (N, E)$ (see Fig. 4).

Using the available visibility graph G, one can easily answer the practical question about the minimal number of UVs required to establish a relay-type communication link between the base and the terminal UV for the given target location. The breadth-first search [12] is ideally suited for this purpose, because its computational complexity grows linearly with the number of edges.

Visibility graphs are widely used in the areas listed in Section 1 in connection with the notion of isovist and viewshed. For constructing visibility graphs, there exist not only efficient computational algorithms [6, 17, 20, 26, 33, 37] and software [27], but also hardware accelerators [32].

Note that the number of uniformly distributed grid points, which are within the distance r' from x_i and invisible for x_i, is proportional to the obstructed volume. Therefore, this number can be used instead of the obstructed volume to define $f(x_i)$. It can be calculated for a given visibility graph G by subtracting the corresponding node degree, i.e. the number of nodes adjacent to x_i, from the maximum possible number of grid points in a sphere of the radius r'. This way of defining $f(x_i)$ can be extended to the case of nonuniformly distributed grid points.

Assuming that $s, t \in N$, the optimal placing of communications relay nodes is then reduced to finding in this graph a path between the nodes s and t which minimizes the sum $f(x_1) + \ldots + f(x_k)$. Notice that each term in this objective function is associated with

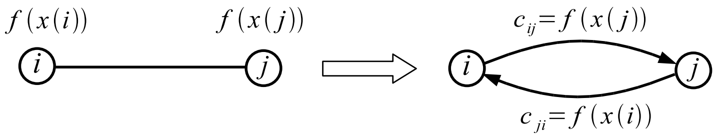

Figure 5: Transition from the undirected graph G to the weighted directed graph $\bar{G}$.

the corresponding node. Since this observation does not allow us yet to apply directly any of the conventional network optimization methods [1, 8] to solving this problem, we will introduce directed edges and define their costs.

We suggest to change from the undirected graph G to a directed one $\bar{G}$ as follows. The set of nodes N remains the same. Any undirected edge $(i, j) \in E$ is substituted by the two directed edges (i, j) and (j, i). The resulting set of edges is denoted by $\bar{E}$. Thus, the directed graph is defined as $\bar{G} = (N, \bar{E})$.

Let $x(j)$ denote the point position associated with the node $j \in N$. Then we assign the cost $c_{ij} = f(x(j))$ to each edge $(i, j) \in \bar{E}$ as indicated in Fig. 5. The introduced edge costs accomplish our definition of the network with the two selected nodes s and t.

If $(i, j) \in \bar{E}$, then by the construction of the visibility graph, it is guaranteed that $[x(i), x(j)] \subset X$ and $\|x(i) - x(j)\| \leq r$. Thus, any feasible placement of communications relay nodes in D composes a path from the node s to the node t, say

$$s \to i \to j \to \cdots \to k \to t.$$

Such placement is characterized by both the *path length* equal to the number of edges (hops) in the path, and the *path cost* equal to

$$c_{si} + c_{ij} + \ldots + c_{kt} = f(x(i)) + f(x(j)) + \ldots + f(x(k)) + f(t),$$

where the term $f(t)$ does not depend on the placement. This allows us to formulate the problem of optimal placement of communications relay nodes as the problem of finding a cheapest path from the node s to the node t.

If the node outdegree is used instead of the obstructed volume, the problem is equivalent to finding a path from s to t with maximal total outdegree of nodes in the path.

It is intuitively clear that in the practically important cases it is natural to expect that the discrete cheapest path solution converges to the global solution to problem (1)–(2) when the discretization is properly refined. In this paper, we do not study this convergence.

There exist efficient algorithms for solving the cheapest path problem (see e.g. [1, 8, 10, 12]). In our notations, the computational complexity of the best of the known practical cheapest path algorithms is of $O(|\bar{E}| + |N| \log |N|)$, where $|A|$ denotes the cardinality of the set A.

It is of practical significance in the UV-based communications relay problem to solve the corresponding cheapest path problem, as quickly as possible, after the target location becomes available. Such problems may arise repeatedly for the same terrain and the same location of the base station, but for different target locations. We make the desired quick solving achievable by presolving the problem in advance. For this purpose, we take into account as much of the a priori available information as possible.

At the presolving stage, we suggest to discretize the area of interest, construct the visibility graph, and then solve the resulting single-source cheapest path problem [1, 8, 12], whose solution is a tree of cheapest paths from s to each node in N. The cheapest path tree composes a special kind of communication map which for each node contains the location of its immediate predecessor in a cheapest path from s to this node.

At the stage when the location of the target becomes available, an optimal placement of relay nodes can be very quickly retrieved from the communication map. The number of required arithmetic operations is proportional to the optimal number of relay nodes. It is vanishing in comparison with $O(|\bar{E}| + |N| \log |N|)$, the best known complexity of the algorithms that can be used at the presolving stage.

4 Restricted number of relay nodes

In this section, we consider problem (2)–(3) assuming that the maximal number of relay nodes K is given. After generating the network as described in Section 3, this problem is reduced to the cheapest path problem with a restricted number of nodes in path, or equivalently, with a restricted number of edges (hops). In the latter problem, $K + 1$ is the maximal number of edges in the optimal path to be found.

In relation to the hop-restricted cheapest path problem, we should mention here the weight-restricted cheapest path problem [15], which can be formulated as follows. Given a directed graph with edge costs c_{ij}, edge weights w_{ij} and the upper limit K for the path weight, find a cheapest path from s to t whose weight does not exceed K. It is known to be an NP-hard problem, however it can be solved in pseudopolynomial time. In [15], one can find an overview of the existing algorithms. Our hop-restricted cheapest path problem is a special case in which $w_{ij} = 1$ for all edges, and it can be solved in polynomial time.

We call a path *k-restricted* if it consists of at most k edges. We will consider here the all hops optimal path (AHOP) problem [21]. It consists in finding a k-restricted cheapest path from a selected node s to all $j \in N$ for each $k = 1, 2, \ldots$. Note that in the mentioned hop-restricted cheapest path problem the value of k is fixed. The reasoning in this section does not take into account any information about the UV origination of the graph $\bar{G} = (N, \bar{E})$. The only assumption is that $\bar{G}$ has no cycle of negative cost.

Let $d(j)$ stand for the depth of node j with respect to node s, i.e. $d(j)$ is the minimal number of edges over all paths from s to j. If there is no path from s to j, we define $d(j) = \infty$. We denote the maximal depth over all $j \in N$ reachable from s by

$$k_{\max} = \max\{d(j) : \ j \in N, \ d(j) < \infty\}.$$

It can be interpreted as the minimal number of UVs which would be definitely sufficient to survey a target at any discrete position, not necessarily in the optimal way.

The notation $d^*(j)$ will be used for the minimal number of edges (hops) over all *cheapest* paths from s to j. We call a cheapest path tree *least-hops* if, for each $j \in N$, the length of the path in this tree from s to each j is minimal, i.e. it is equal to $d^*(j)$. Let $k^*_{\max}$ denote the height of a least-hops cheapest path tree, which means that

$$k^*_{\max} = \max\{d^*(j) : \ j \in N, \ d^*(j) < \infty\}.$$

It can be interpreted as the minimal number of UVs sufficient for the *optimal* surveillance of a target at any discrete position.

Since the set of cheapest paths is a subset of all paths, we have

$$d(j) \le d^*(j), \quad \forall j \in N,$$

and $k_{\max} \le k^*_{\max}$.

We say that an AHOP problem *admits a trivial solution* if

$$d(j) = d^*(j), \quad \forall j \in N.$$

Then $k_{\max} = k^*_{\max}$. In our UV applications, this case means that, for each target position, there exists an optimal node placement with the minimum possible number of UVs. In [9], we present a sufficient condition for the AHOP problem to admit a trivial solution. This condition is formulated in terms of the edge costs c_{ij}.

The notations $g^*_k(j)$ and $g^*(j)$ will be used for the cost of, respectively, a k-restricted and standard (unrestricted) cheapest path from s to j. If $0 \le k < d(j)$, there is no k-restricted path from s to j, and in this case we define $g^*_k(j) = +\infty$.

Obviously,

$$g^*_0(j) \ge g^*_1(j) \ge \ldots \ge g^*_k(j) \ge g^*_{k+1}(j) \ge \ldots \tag{4}$$

Moreover, $g^*_k(j) \ge g^*(j)$ for all $k \ge 0$. More detailed relations between $g^*_k(j)$ and $g^*(j)$ are presented below by Lemma 1.

Let

$$j_- = \{i \in N : \ (i, j) \in \bar{E}\}, \quad j_+ = \{i \in N : \ (j, i) \in \bar{E}\}$$

denote the sets of all immediate predecessors and successors of node $j \in N$, respectively. In this notation, the optimality conditions for the AHOP problem can be written (see, e.g. [23]) in the form of the recursion relation

$$g^*_k(j) = \min\{g^*_{k-1}(j), \ \min_{i \in j_-}\{g^*_{k-1}(i) + c_{ij}\}\} \tag{5}$$

valid for any $j \in N$ and $k \ge 0$. It can be viewed as a Bellman-type recurrence equation [1, 5, 8, 18].

The optimality conditions allows us to formulate the following lemma. It will be used for proving the correctness of the algorithms presented in the next section.

Lemma 1 *Let a weighted directed graph $\bar{G} = (N, \bar{E})$ with source s contain no negative cycles. Then the optimal solution to the AHOP problem has the following structure.*

a) If, for some $i, j \in N$ and k such that $(i, j) \in \bar{E}$ and $2 \le k \le k^*_{\max}$, the relations $g^*_{k-2}(i) = g^*_{k-1}(i)$ and $g^*_{k-1}(j) > g^*_k(j)$ hold, then $g^*_{k-1}(i) + c_{ij} > g^*_k(j)$.

b) $g^*_k(j) > g^*(j)$ iff $k < d^*(j)$.

c) $g^*_k(j) = g^*(j)$ iff $k \ge d^*(j)$.

Proof. By optimality conditions (5) and the assumptions in a), we obtain the relations

$$g_k^*(j) < g_{k-1}^*(j) \le g_{k-2}^*(i) + c_{ij} = g_{k-1}^*(i) + c_{ij},$$

which prove statement a).

Statements b) and c) simply follow from the fact that $d^*(j)$ is the minimal number of edges for all unrestricted cheapest paths from s to j, and $g^*(j)$ is the cost of such paths. Indeed, if the number of edges in a path is less than $d^*(j)$, the path cost must be larger than $g^*(j)$. Moreover, it is obvious that for $k \ge d^*(j)$ there is no k-restricted path which is cheaper than $g^*(j)$, while there exists a path of the cost $g^*(j)$ and the length $d^*(j)$. $\square$

If an AHOP problem admits a trivial solution, then for all $j \in N$ and $k \ge 0$, by Lemma 1, we have

$$g_k(j) = \begin{cases} +\infty, & \text{if } 0 \le k < d^*(j) \\ g^*(j), & \text{if } k \ge d^*(j) \end{cases}.$$

5 AHOP algorithms

The following algorithm of [23] is based on the recursion relation (5), and it extracts the AHOP solutions from the labels generated by the conventional Bellman-Ford algorithm.

Algorithm 1.

```
1  for each $i \in N \setminus \{s\}$ do
2      $g_0(i) \leftarrow +\infty$
3  $g_0(s) \leftarrow 0$
4  for $k = 1, 2, \dots, |N| - 1$ do
5      for each $j \in N$ do
6          $g_k(j) \leftarrow g_{k-1}(j)$
7      for each $i \in N$ do
8          for each $j \in i_+$ do
9              if $g_{k-1}(i) + c_{ij} < g_k(j)$ then
10                 $g_k(j) \leftarrow g_{k-1}(i) + c_{ij}$
```

At iteration k, Algorithm 1 produces implicitly, for each node $j \in N$, a k-restricted path from s to j of the cost $g_k(j)$, which is an upper estimate for $g_k^*(j)$. Whenever the label $g_k(j)$ is improved, the path is updated. At the end of the k-th iteration, the algorithm generates

$$g_k(j) = g_k^*(j), \qquad \forall j \in N. \tag{6}$$

This statement is justified, e.g., in [1, p. 142].

We introduce here two modifications of Algorithm 1 which are able to take advantage of using some characteristic features of the AHOP problems originating from our UV-based applications. The modifications are built upon the properties of the optimal AHOP solutions presented by Lemma 1.

The computational complexity of Algorithm 1 is $O(|N||\bar{E}|)$ (see e.g. [1]). This estimate is justified by the necessity of performing $|N| - 1$ iterations of the **for** loop in lines 4-10, which contains the **for** loop in lines 7-10 to be performed for all edges.

We observe that iteration $k = k^*_{max}$ results in $g_k(j) = g^*(j)$ for all $j \in N$. This means that $g_k(j)$ has attained its minimal value and would not change at the subsequent iterations. Therefore, Algorithm 1 can be terminated after this iteration.

Clearly $k^*_{max} < |N|$. It should be emphasized that, in the practical placement of communications relay nodes, the value of k^*_{max} mostly depends on the terrain topology. For given area of interest, k^*_{max} does not change much with the increasing number of discrete points $|N|$, if it changes at all. It is natural to expect for properly refining discretization that k^*_{max} tends to the maximal value of $k(t)$ over all possible continuous target positions t, where $k(t)$ is an optimal solution to problem (1)–(2). In other words, k^*_{max} tends to the minimal number of UVs which would be definitely sufficient for the optimal surveillance of a target at any position. For our applications, it is typical that

$$k^*_{max} << |N|. \tag{7}$$

This observation is taken into account in Algorithm 2 (presented below), which can be viewed as a modification of Algorithm 1.

Another observation is based on Lemma 1a). It states that if $g_{k-1}(i)$ did not change at iteration $k-1$, then $g_{k-1}(i) + c_{ij}$ is not able to improve the value of $g_k(j)$, i.e. in this case, line 10 of Algorithm 1 is not performed for node i at iteration k. Therefore, the **for** loop in lines 8-10 can be restricted to only those edges $(i, j) \in \bar{E}$ which begin in nodes i that compose the set

$$V_k = \{i \in N : \quad g_{k-1}(i) \neq g_{k-2}(i)\}.$$

Algorithm 2 gains the most benefit from using this observation.

In Algorithm 2, the label $g(j)$ takes the same values at iteration k as $g_k(j)$ in Algorithm 1. If $g(j)$ changes at the k-th iteration, we set $g_k(j) \leftarrow g(j)$, and if not, no label $g_k(j)$ is assigned to node j at iteration k.

Algorithm 2 has an extra feature. At iteration k, it produces for each node $j \in N$ its immediate predecessor $p(j)$ in the k-restricted path from s to j. The cost of this path $g(j)$ gives an upper estimate for $g^*_k(j)$. At the end of the k-th iteration of Algorithm 2, the equality

$$g(j) = g^*_k(j), \qquad \forall j \in N \tag{8}$$

similar to (6) holds, and $p(j)$ is the immediate predecessor of node j in the implicitly produced k-restricted cheapest path from s to j. The cost of this path is equal to $g^*_k(j)$. It will be explained below how to retrieve this path from the available list of predecessors $p_1(\cdot), \dots, p_k(\cdot)$. The mentioned feature was not introduced in Algorithm 1, because we wish to simplify its formal presentation and to focus on its modifications.

As a result of the k-th iteration of Algorithm 2, the set V_{k+1} is composed of those nodes $j \in N$ for which a cheaper path from s to j has been found at this iteration, i.e. $g^*_k(j) < g^*_{k-1}(j)$. Only these nodes are used at the next iteration to possibly improve the value of g_{k+1} for their immediate successors. This allows Algorithm 2 to reduce in practice the total number of elementary operations, although the worst-case estimate remains the same as for Algorithm 1, namely, $O(|N|\,|\bar{E}|)$.

The outlined algorithm can be formally presented as follows.

Algorithm 2.

```
1  **for** each $i \in N \setminus \{s\}$ **do**
2      $g(i) \leftarrow +\infty$
3  $g(s) \leftarrow 0,\ g_0(s) \leftarrow g(s),\ p_0(s) \leftarrow nil,\ V_1 \leftarrow \{s\}$
4  **for** $k = 1, 2, \ldots$ **while** $|V_k| \neq 0$ **do**
5      $V_{k+1} \leftarrow \emptyset$
6      **for** each $i \in V_k$ **do**
7          **for** each $j \in i_+$ **do**
8              **if** $g_{k-1}(i) + c_{ij} < g(j)$ **then**
9                  $g(j) \leftarrow g_{k-1}(i) + c_{ij},\ p(j) \leftarrow i,\ V_{k+1} \leftarrow V_{k+1} \cup \{j\}$
10     **for** each $j \in V_{k+1}$ **do**
11         $g_k(j) \leftarrow g(j),\ p_k(j) \leftarrow p(j)$
```

Algorithm 2 returns a collection of triples $\{k, g_k(j), p_k(j)\}$ produced for each node $j \in N$. We call them *kgp*-triples. The output of Algorithm 2 for node j may not contain *kgp*-triples for some values of k. In fact, in our applications, such collections are typically very sparse - just one or a few *kgp*-triples per one node. Recall that, in (4), the sequence of non-increasing values of $g^*_k(j)$ may remain the same for some number of the consecutive values of k. The *kgp*-triples are produced by Algorithm 2 only for those values of k which are minimal in such series of equal values of $g^*_k(j)$. Therefore, in order to find the value of a, possibly, missing triple $\{k, g_k(j), p_k(j)\}$, it is necessary to find the largest $k' \leq k$ for which a *kgp*-triple $\{k', g_{k'}(j), p_{k'}(j)\}$ exists. Then the desired values are $g_k(j) = g_{k'}(j)$ and $p_k(j) = p_{k'}(j)$. If such k' does not exist, then $g_k(j) = +\infty$ and $p_k(j) = nil$. A k-restricted cheapest path from s to j can be retrieved from the available *kgp*-triples as follows. We set $j_{k'} = j$, and then recursively find

$$j_{k'-1} = p_{k'}(j_{k'}), \quad j_{k'-2} = p_{k'-1}(j_{k'-1}), \quad \ldots, \quad j_0 = p_1(j_1), \tag{9}$$

where $j_0 = s$.

The main properties of Algorithm 2 are summarized in the following theorem.

Theorem 2 *Let Algorithm 2 be run on a weighted directed graph $\bar{G} = (N, \bar{E})$ with source s. Assume that this graph contains no negative cycles. Then after k iterations of the* ***for*** *loop in lines 4-11, equation (8) holds. Moreover, the generated kgp-triples correctly define, for each node $j \in N$, a path of the optimal cost $g^*_k(j)$. The length of this path is minimal over all cheapest k-restricted paths from s to j, if such paths exist, and it is equal to the largest $k' \leq k$ for which a triple $\{k', g_{k'}(j), p_{k'}(j)\}$ exists. Algorithm 2 terminates in $k = k^*_{max} + 1$ iterations.*

Proof. Denote $V^*_1 = \{s\}$ and

$$V^*_k = \{i \in N : \quad g^*_{k-1}(i) \neq g^*_{k-2}(i)\} \tag{10}$$

for $k \geq 2$. By Lemma 1, the optimality conditions (5) can be written as

$$g^*_k(j) = \min\{g^*_{k-1}(j), \min_{i \in j_- \cap V^*_k} \{g^*_{k-1}(i) + c_{ij}\}\}. \tag{11}$$

Then it can be easily shown, by induction in $k = 1, 2, \ldots, k^*_{max}$, that after iteration k, equation (8) holds and $V_k = V^*_k$. By the definition of k^*_{max}, $V^*_k \neq \emptyset$ for these values of k. Therefore, Algorithm 2 can not terminate before iteration $k = k^*_{max} + 1$. By Lemma 1 and equation (8), at the end of iteration $k = k^*_{max}$ we have

$$g(j) = g^*(j), \qquad \forall j \in N.$$

At the next iteration, none of these values of $g(j)$ can be decreased. Therefore, $V_{k^*_{max}+2} = \emptyset$ and Algorithm 2 terminates after iteration $k = k^*_{max} + 1$.

By the assumption of the theorem, a triple

$$\{k', g_{k'}(j_{k'}), p_{k'}(j_{k'})\}$$

was generated by Algorithm 2 for $j_{k'} = j$. Then $p_{k'}(j_{k'}) \in V^*_{k'}$. This, by definition (10), ensures that the triple

$$\{k' - 1,\ g_{k'-1}(p_{k'}(j_{k'})),\ p_{k'-1}(p_k(j_{k'}))\}$$

exists. By reasoning recursively in the same way, we can prove the correctness of the recursive procedure defined by (9). Since $V^*_1 = \{s\}$, we have $j_0 = s$. Thus, (9) defines a path from s to j with the path cost $g^*_k(j)$. Then it can be easily shown, by induction in k, that the length of this path is minimal over all cheapest k-restricted paths from s to j. □

Consider the two-criteria optimization problem in which both the path cost and its length (the number of hops) are to be minimized for each node. Theorem 2 implies that Algorithm 2 generates a Pareto optimal solution to this problem. Algorithm 1 could also produce a Pareto optimal solution if to modify it properly.

Observe that the major computational burden of Algorithms 1 and 2 is associated with the execution of lines 9-10 and 8-9, respectively.

In Algorithm 1, lines 9-10 are executed approximately $|\bar{E}|\ |N|$ times, because each edge (i, j) is used only once for each k.

Similar calculations show that lines 8-9 of Algorithm 2 are executed at most $k^*_{max}|\bar{E}|$ times. In the problems for which (7) holds, this very rough estimate is much better than $|N||\bar{E}|$. To continue with a more refined analysis of the computational burden, we denote:

$$\bar{E}_k = \{(i, j) \in \bar{E} :\ i \in V_k,\ j \in i_+\},$$

$$|\bar{E}_{\max}| = \max\{|\bar{E}_k| :\ 1 \leq k \leq k^*_{\max}\}.$$

Then it is not difficult to verify that the number of elementary operations of Algorithm 2 grows in proportion to the value

$$\sum_{k=1}^{k^*_{max}} |\bar{E}_k|.$$

Since this estimate is bounded above by $k^*_{max}|\bar{E}_{\max}|$, it is typically well below $k^*_{max}|\bar{E}|$ for our applied problems, in which $|\bar{E}_k|$ is far less than $|\bar{E}|$.

Our further progress in reducing the number of elementary operations is related with the use of the cheapest path tree. If the edge costs are non-negative, this tree can be produced, for instance, by Dijkstra's algorithm. Its computational complexity is $O(|\bar{E}| + |N|\log|N|)$. We need the cheapest path tree to possess the least-hops property. This extra property can be assured by a simple modification of the algorithms producing cheapest path trees. For this purpose, not only a path cost should be associated with each node, but also the length of this path. The currently best path can be improved not only when a candidate path has a smaller cost, but also when its length is smaller while the cost is the same as the currently best one (for the formal presentation of this modification, see [9]).

Let $p^*(j)$ denote the immediate predecessor of node $j \in N$ in a given least-hops cheapest path tree. Hereafter we assume that such a tree is available. More specifically, not only the number k^*_{max} and the kgp-triples

$$\{d^*(j), g^*(j), p^*(j)\}, \quad j \in N, \tag{12}$$

are assumed to be available, but also the sets

$$N^*_k = \{j \in N : \ d^*(j) = k\}, \quad k = 0, 1, \ldots, k^*_{max}. \tag{13}$$

All these sets can be generated for a given tree in a linear time of the number of nodes $|N|$.

We observe that, for any node $j \in N$, the inequality in line 9 of Algorithm 2 obviously holds at each iteration $k > d^*(j)$. Therefore, it is not necessary to check this inequality at any of these iterations. Moreover, if the value of $g^*(j)$ is available for a node j, it is natural to skip the execution of lines 9-10 for this node at the iteration $k = d^*(j)$.

For any node $j \in N$, we can avoid the unnecessary execution of these lines at iterations $k \geq d^*(j)$ by excluding at iteration $k = d^*(j)$ node j from all the sets i_+ such that $i \in j_-$. This is the key observation which allows us to efficiently exploit the available least-hops cheapest path tree. It is implemented in the following algorithm.

Algorithm 3.

```
1  for each $i \in N \setminus \{s\}$ do
2      $g(i) \leftarrow +\infty$
3  $V_0 \leftarrow \emptyset$
4  for $k = 0, 1, \ldots, k^*_{max} - 1$ do
5      for each $j \in N^*_k$ do
6          for each $i \in j_-$ do
7              $i_+ \leftarrow i_+ \setminus \{j\}$
8      $V_{k+1} \leftarrow \emptyset$
9      for each $i \in V_k$ do
10         for each $j \in i_+$ do
11             if $g_{k-1}(i) + c_{ij} < g(j)$ then
12                 $g(j) \leftarrow g_{k-1}(i) + c_{ij}$,  $p(j) \leftarrow i$,  $V_{k+1} \leftarrow V_{k+1} \cup \{j\}$
13     for each $j \in V_{k+1}$ do
14         $g_k(j) \leftarrow g(j)$,  $p_k(j) \leftarrow p(j)$
15     $V_{k+1} \leftarrow V_{k+1} \cup N^*_k$
```

Before this algorithm is executed, the sets N_k^* and a least-hops cheapest path tree must be determined.

The main properties of Algorithm 3 can be summarized in the same way as it has been done in Theorem 2 for Algorithm 2. We skip it, because the difference in their formulations and proofs are pretty obvious. It can be easily seen also that the total number of elementary operations is less for Algorithm 3 than for Algorithm 2. This derives from the fact that i_+ in the set of edges associated with the **for** loops in lines 6-7 and 10-12 of Algorithm 3 is a subset of the set of edges which are used at the same iteration of Algorithm 2.

Line 7 of Algorithm 3 is equivalent to the exclusion of the edge (i, j) from the set $\bar{E}$. Since this operation is done at most once for each edge, the computational burden associated with lines 5-7 grows linearly with the number of edges $|\bar{E}|$.

If in our UV applications the terrain topology is not too complicated, the corresponding AHOP problem may admit a trivial solution. In this case, at each iteration of Algorithm 3 we have $i_+ = \emptyset$ in line 7, $V_{k+1} = \emptyset$ in line 13, and $V_{k+1} = N_k^*$ in line 15. It is the most favorable case for Algorithm 3, because the total number of elementary operations associated with lines 8-15 grows linearly with $|N|$.

In general, the smaller the difference is between the node distances $d_k^*(j)$ and $d_k(j)$, the lower the computational burden of Algorithm 3.

6 Implementation issues and numerical experiments

Here we consider test problems originating from optimal placement of UAV-based communications relay nodes. Such problems are characterized by relation (7). Another feature of these problems is that the edge costs are non-negative.

Our test problems cover various 3D terrain topologies. Each row in Table 1 refers to one test case. The first five columns present some features of the generated directed graph $\bar{G} = (N, \bar{E})$. Column 'outdegree' specifies the min/max/average node outdegree.

In column '%kgp > 1', the percentage of nodes j with $d^*(j) > d(j)$ is indicated. Such nodes have more than one *kgp*-triple. Zero value means that the corresponding test problem admits a trivial solution. To increase the portion of nodes with $d^*(j) > d(j)$, we modified some of the test problems by artificially increasing $f(x(i))$ for one arbitrary selected node i which was a successor of node s in the original cheapest path tree. The artificially modified problems are marked by star in the first column of Table 1.

One can see that the number of discretization points varies from medium to very large. In the largest test problem, the area of interest is of the size $1000m \times 1000m \times 60m$, the discretization step is $10m$, and the communication radius $r = 100m$. The obstacles in this problem model an urban environment.

In the previous section, it was mentioned that Algorithm 1 can be terminated after iteration $k = k^*_{max}$. Since the value of k^*_{max} is not available, one can terminate Algorithm 1 as soon as

$$g_k(j) = g_{k-1}(j), \qquad \forall j \in N,$$

results from iteration k.

This idea is widely used in implementations of the Bellman-Ford algorithm. Here it was implemented by setting $flag \leftarrow true$ at the very beginning of iteration k, and if the inequality in line 9 of Algorithm 1 holds at least once, we set $flag \leftarrow false$. Then the **for** loop in lines 4-10 is terminated as soon as $flag = true$ at the very end of iteration k. We shall refer to this modification as Algorithm $1'$.

This simple idea provides a speedup factor of about $|N|/k^*_{max}$, which is approximately 10^3 in the largest of the test examples considered below. Due to (7), the speedup factor grows linearly with $|N|$. One should bear this in mind, when we compare our Algorithms 2 and 3 with Algorithm $1'$, which is actually the conventional Bellman-Ford algorithm tailored to solving our UV-related problems.

We were using C++ for implementing our algorithms. Algorithm 2 was implemented with no change.

Our implementation of Algorithm 3 was based on the following observation. One can skip the **for** loop in lines 5-7 of Algorithm 3 if line 10 is changed as follows

```
10        for each $j \in i_+$ such that $d^*(j) > k$ do
```

We shall refer to this implementation as Algorithm $3'$. More sophisticated implementations of our algorithms and their detailed comparison will be the main subject of a separate paper.

Algorithm $3'$ requires that k^*_{max} and the part of *kgp*-triples presented by (12) are available. They are produced by Dijkstra's algorithm modified in the way outlined in the previous section. Dijkstra's algorithm was implemented with the use of the standard heap algorithms available in the C++ template library.

The sets N^*_k (13) are also required to be available for Algorithm $3'$. They are produced at the preprocessing stage by sorting the nodes $j \in N$ in the increasing order of $d^*(j)$ with subsequent identification of the segments $N^*_1, N^*_2, \ldots$ in the obtained sequence of sorted nodes.

The numerical results presented here were produced on a PC running under Windows Vista with an Intel Core 2 Duo processor (2.4 GHz, 2GB RAM). Only one core was involved in the computational process. The CPU time was measured in milliseconds averaged over 1000 runs. We used the O2 optimization flag in the Microsoft Visual Studio 2008 C++ compiler.

The CPU time of running Algorithms $1'$, 2, $3'$ and Dijkstra's algorithm are presented by the corresponding columns in Table 1. We shall refer to the combination of Algorithm $3'$, Dijkstra's and preprocessing algorithms as the combined Algorithm $3'$. The run time of the preprocessing algorithm is not reported here because it was less than 1% of the run time of the combined Algorithm $3'$ presented by the last column.

Table 1 allows us to compare the run time of Algorithms 2 and the combined Algorithm $3'$ vs the run time of Algorithms $1'$. One can see that the speedup factor of Algorithm 2 ranges from 10 to 63. The combined Algorithm $3'$ is, in general, faster than Algorithm 2, especially in the cases of large scale problems. The combined Algorithm $3'$ provides a speedup with the factor ranging from 10 to a few hundreds. It should be emphasized that the run time of this combination is, in the most cases, less than twice the time of running Dijkstra's algorithm. The results presented by Table 1 show the high efficiency of the new algorithms.

Table 1: Graph characteristics and run time in milliseconds.

$\lvert N\rvert$	$\lvert\bar{E}\rvert$	outdegree	k^*_{max}	%kgp > 1	Alg $1'$	Alg 2	Alg $3'$	Dijkstra	$3'$DP
742	63 876	1/68/43	16	52.2	35	3.5	0.013	0.041	0.054
763	7 732	2/6/5	20	16.4	9	0.3	0.004	0.009	0.015
1 395	14 332	3/6/5	35	18.4	35	0.5	0.018	0.021	0.042
1 654	66 680	5/24/20	27	68.4	69	4.3	1.152	0.695	1.849
1 654	66 680	5/24/20	24	68.4	74	4.2	1.163	0.117	1.280
1 993	20 544	2/6/5	33	35.2	45	1.0	0.035	0.047	0.086
3 411	244 492	9/48/35	19	74.3	175	15.7	6.9	3.3	10.2
5 294	2 269 268	3/316/214	13	2.1	1 053	42.0	17.6	23.4	35.9
*5 294	2 269 268	3/316/214	13	30.6	1 178	65.2	44.9	23.4	68.3
5 294	2 272 256	3/316/214	14	0	1 117	40.8	12.3	16.9	29.2
*5 294	2 272 256	3/316/214	14	2.1	1 120	46.9	12.5	17.4	29.9
5 804	1 248 772	20/173/108	19	0.2	823	24.3	6.8	10.1	16.9
*5 804	1 248 772	20/173/108	19	3.1	878	34.5	6.7	10.1	17.1
7 117	3 251 472	3/316/228	15	0	1 736	58.3	17.9	25.1	43.0
*7 117	3 251 472	3/316/228	15	3.3	1 756	87.9	18.7	33.8	52.5
9 226	10 381 952	71/965/563	10	22.0	3 845	252.3	170.3	107.1	278.2
*9 226	10 381 952	71/965/563	10	43.5	3 853	366.3	410.7	82.4	493.2
9 226	46 172 304	228/4793/2502	5	6.8	10 199	882.4	439.7	326.1	765.1
*9 226	46 172 304	228/4793/2502	5	24.3	10 299	1 076.6	414.1	328.9	743.0
15 105	12 622 644	52/560/418	13	0	6 228	225.9	81.5	97.1	179.9
*15 105	12 622 644	52/560/418	13	2.3	6 220	202.8	81.5	97.3	180.0
15 105	12 622 644	52/560/418	14	0	6 866	226.4	81.4	130.9	213.7
*15 105	12 622 644	52/560/418	14	2.4	6 756	274.8	83.0	98.4	182.7
18 801	26 450 548	114/1134/703	13	0.2	12 595	480.9	164.9	195.7	362.8
*18 801	26 450 548	114/1134/703	13	1.3	12 430	518.1	165.9	196.6	364.7
38 542	56 654 428	64/1198/735	17	0.1	36 354	1 060.4	380.4	580.9	968.0
*38 542	56 654 428	64/1198/735	17	2.3	35 111	1 412.9	301.5	436.0	824.0

7 Conclusions and future work

The two main results of this paper are the following.

First, the multiextremal problem (1)–(2), for which it is intractable to directly calculate any optimal solution, has been reduced to an easy-to-solve cheapest path problem, which yields a reasonably accurate global optimum for a properly refined discretization. The practical importance of this approach is that the major computational burden falls on the presolving stage, owing to which a real-time decision on the optimal placement of the relay nodes can be made almost immediately after the target position becomes available.

Second, new algorithms for finding hop-restricted cheapest paths have been developed. The important property of these algorithms is that for each possible target position they produce a Pareto solution. It is an optimal solution to the multi-criteria problem of minimizing the path cost and the number of hops. The practical efficiency of our algorithms originates from their ability to take into account some peculiarities of the considered UVs placement problem. Our numerical experiments show that the new algorithms are reasonably fast in solving the single source restricted cheapest path problem AHOP. Their computational time is a small multiple of the time required by Dijkstra's algorithm to solve the single source unrestricted cheapest path problem. The new algorithms are superior to the conventional Bellman-Ford algorithm tailored to solving AHOP problems.

As an alternative approach, we develop separately a dual ascent algorithm. It is applied to finding a restricted cheapest path for given initial and terminal nodes. It is based on the Lagrangian relaxation of the constraint which limits the number of hops. Preliminary results are encouraging. They are reported in [9].

We plan to extend our approach to the case of nonuniform UVs with an individual range of the communication equipment. In this case the inequality constraints in (2) take the form

$$\|x_{i-1} - x_i\| \leq r_i, \;\; \|x_{i+1} - x_i\| \leq r_i.$$

The approach presented in this paper can be easily modified if it is necessary to incorporate some other type of constraints of practical importance. For instance, the visibility graph can take into account the presence of areas where any placement of autonomous vehicles is prohibited, while intersections of the communication lines with such areas are admitted. Moreover, the extra requirement, that the autonomous vehicles can not be placed too close to each other, can also without difficulty be taken into account in the process of generating the visibility graph.

If it is required to avoid any placement of UVs too close to obstacles, this can be implemented, for instance, by introducing weights in computing the obstructed volume. Those parts of the ball which are more close to its center should have higher weights. Alternatively, one can add to the standard obstructed volume a term which somehow penalizes the presence of obstacles in the immediate vicinity of the UV.

Note that our approach can be evidently extended to the case where each term of the objective function in (2) depends not only on x_i, but also on the position of the previous point in the sequence of relay nodes. This refers to the objective function of the form:

$$\sum_{i=1}^{k+1} f(x_{i-1}, x_i), \tag{14}$$

for which the edge cost can be defined as $c_{ij} = f(x(i), x(j))$.

For UV applications, we plan to consider objective functions of a more general type. The objective functions of the form presented by (14) provide a possibility of assessing not only the UV positions, but also the communication quality.

Our intention is also to address the optimal placement of communications relay nodes in the case of multiple targets and restricted number of UVs.

It should be emphasized that there are some other applied problems, different from those considered here, which could gain from applying our approach either directly or in a modified form. These problems are characterized by the necessity to restrict the number of turns in optimal piecewise linear paths [34]. One can find such problems, for instance, in automated VLSI circuit design [24, 35] and in robotic motion planning [11, 22].

Acknowledgments

This work is partially supported by grants from the Swedish Foundation for Strategic Research (SSF) Strategic Research Center MOVIII, the Swedish Research Council Linnaeus Center CADICS, LinkLab and the Linköping University Center for Industrial Information Technology (CENIIT). The authors would like to thank Jonas Kvarnström for his valuable comments and suggestions helped to improve the presentation of their results.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, 1993.

[2] S.O. Anderson, R. Simmons, and D. Goldberg. Maintaining Line of Sight Communications Network between Planetary Rovers. In *Proceedings of the 2003 IEEE/RSJ International Conference of Intelligent Robots and Systems*, pages 2266–2272. IEEE, 2003.

[3] D.A. Anisi, P. Ögren, and X. Hu. Communication constrained multi-UGV surveillance. In *Proc. of the* 17th *IFAC World Congress*, Seoul, South Korea, 6-11 Jul. 2008.

[4] R.C. Arkin and J. Diaz. Line-of-Sight Constrained Exploration For Reactive Multiagent Robotic Teams. In *7th International Workshop on Advanced Motion Control*, pages 455–461, 2002.

[5] R. Bellman. On a Routing Problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[6] B. Ben-Moshe, P. Carmi, and M.J. Katz. Approximating the Visible Region of a Point on a Terrain. *GeoInformatica*, 12(1):21–36, 2008.

[7] D.P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 1995.

[8] D.P. Bertsekas. *Network Optimization: Continuous and Discrete Models.* Athena Scientific, 1998.

[9] O. Burdakov, K. Holmberg, and P.-M. Olsson. A dual ascent method for the hop-constrained shortest path problem with application to positioning of unmanned aerial vehicles. Technical Report LiTH-MAT-R-2008-07, Department of Mathematics, Linköping University, 2008.

[10] B.V. Cherkassky, A.V. Goldberg, and T. Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical Programming*, 73(2):129–174, 1996.

[11] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT Press, 2005.

[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms, 2nd Edition.* MIT Press and McGraw-Hill, 2001.

[13] P. Doherty. Advanced research with autonomous unmanned aerial vehicles. In *Proceedings on the 9th International Conference on Principles of Knowledge Representation and Reasoning*, 2004.

[14] P. Doherty and P. Rudol. A UAV search and rescue scenario with human body detection and geolocalization. In *20th Australian Joint Conference on Artificial Intelligence (AI07)*, 2007.

[15] I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003.

[16] M. Dynia, J. Kutylowski, F.M. auf der Heide, and J. Schrieb. Local strategies for maintaining a chain of relay stations between an explorer and a base station. In *SPAA '07: Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, pages 260–269, New York, NY, USA, 1 January 2007. ACM Press, New York, NY, USA.

[17] L. De Floriani and P. Magillo. Algorithms for visibility computation on terrains: a survey. *Environment and Planning B: Planning and Design*, 30(5):709 728, 2003.

[18] L.R. Ford, Jr., and D.R. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[19] A. Fridman, J. Modi, S. Weber, and M. Kam. Communication-based Motion Planning. In *Proc. of 41st Annual Conference on Information Sciences and Systems*, pages 382–387. IEEE, 2007.

[20] S.K. Ghosh. *Visibility Algorithms in the Plane.* Cambridge University Press, 2007.

[21] R. Guérin and A. Orda. Computing shortest paths for any number of hops. *IEEE/ACM Transactions on Networking*, 10(5):613–620, 2002.

[22] S.M. LaValle. *Planning Algorithms.* Cambridge University Press, 2006.

[23] E.L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, 1976.

[24] F.T. Leighton and A.L. Rosenberg. Three-dimensional circuit layouts. *SIAM Journal on Computing*, 15(3):793–813, 1986.

[25] A. Moitra, R.M. Mattheyses, V.A. DiDomizio, L.J. Hoebel, R.J. Szczerba, and B. Yamrom. Multivehicle reconnaissance route and sensor planning. *IEEE Transactions on Aerospace and Electronic Systems*, 39(3):799–812, July 2003.

[26] G. Nagy. Terrain visibility. *Computers & graphics*, 18(6):763–773, 1994.

[27] Karl J. Obermeyer. The VisiLibity library. `http://www.VisiLibity.org`, 2008.

[28] G.A.S. Pereira, A.K. Das, V. Kumar, and M.F.M. Campos. Decentralized motion planning for multiple robots subject to sensing and communication constraints. In *Proc. of the Second Multi-Robot Systems Workshop*, pages 267–278. Kluwer Academic Press, 2003.

[29] N. Pezeshkian, H.G. Nguyen, and A. Burmeister. Unmanned Ground Vehicle Radio Relay Deployment System for Non-line-of-sight Operations. In *Proc. IASTED Int. Conf. on Robotics and Applications.* ACTA Press, 2007.

[30] M.F.J. Pinkney, D. Hampel, and S. DiPierro. Unmanned aerial vehicle (uav) communications relay. In *Military Communications Conference MILCOM'96*, volume 1, pages 45–51. IEEE, 1996.

[31] T. Schouwenaars, A. Stubbs, J. Paduano, and E. Feron. Multivehicle path planning for nonline-of-sight communication. *Journal of Field Robotics*, 23(3-4):269–290, 2006.

[32] K. Sridharan and T.K. Priya. A hardware accelerator and fpga realization for reduced visibility graph construction using efficient bit representations. *IEEE Transactions on Industrial Electronics*, 54(3):1800–1804, June 2007.

[33] A.J. Stewart. Fast horizon computation at all points of a terrain with visibility and shading applications. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):82–93, Jan-Mar 1998.

[34] R.J. Szczerba, D.Z. Chen, and K.S Klenk. Minimum turns/shortest path problems: a framed-subspace approach. In *Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics*, volume 1, pages 398–403. IEEE, 1997.

[35] D. Szeszler. *Combinatorial Algorithms in VLSI Routing.* PhD thesis, Budapest University of Technology and Economics, 2005.

[36] C.R.V. Tandy. The isovist method of landscape survey. In H.C. Murray, editor, *Symposium on Methods of Landscape Analysis*, pages 9–10. Landscape Research Group, London, 1967.

[37] A. Turner, M. Doxa, D. O'Sullivan, and A. Penn. From isovists to visibility graphs: a methodology for the analysis of architectural space. *Environment and Planning B: Planning and Design*, 28:103–121, 2001.