Relay Positioning for Unmanned Aerial Vehicle Surveillance*

Oleg Burdakov[⋆], Patrick Doherty[†], Kaj Holmberg[⋆], Jonas Kvarnström[†], Per-Magnus Olsson[†,‡]

[⋆] Dept. of Mathematics. E-mail: {olbur, kahol}@mai.liu.se

[†] Dept. of Computer and Information Science. E-mail: {pdy, jonkv, perol}@ida.liu.se

Linköping University, SE-581 83 Linköping, Sweden

[‡] Corresponding author

Abstract

When unmanned aerial vehicles (UAVs) are used for surveillance, information must often be transmitted to a base station in real time. However, limited communication ranges and the common requirement of free line of sight may make direct transmissions from distant targets impossible.

This problem can be solved using relay chains consisting of one or more intermediate relay UAVs. This leads to the problem of positioning such relays given known obstacles, while taking into account a possibly mission-specific quality measure. The maximum quality of a chain may depend strongly on the number of UAVs allocated. Therefore, it is desirable to either generate a chain of maximum quality given the available UAVs or allow a choice from a spectrum of Pareto-optimal chains corresponding to different trade-offs between the number of UAVs used and the resulting quality.

In this article, we define several problem variations in a continuous 3D setting. We show how sets of Pareto-optimal chains can be generated using graph search and present a new label-correcting algorithm generating such chains significantly more efficiently than the best known algorithms in the literature. Finally, we present a new dual ascent algorithm with better performance for certain tasks and situations.

1 Introduction

A wide variety of applications of unmanned aerial vehicles (UAVs) include the need for surveillance of distant targets, including search and rescue operations, traffic surveillance and forest fire monitoring as well as law enforcement and military applications. In many cases, the information gathered by a surveillance UAV must be transmitted in real time to a base station where the current operation is being coordinated. This information often includes live video feeds,

*This is an extended version of Burdakov et al. (2009a).

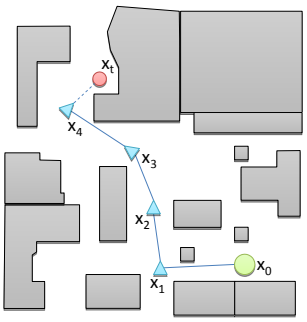


Figure 1: UAVs at x_1, x_2 and x_3 are acting as relays in a city, connecting the base station at x_0 with the surveillance UAV at x_4, surveilling the target at x_t. This is a relay chain of length 5.

where it is essential to achieve uninterrupted communication with high bandwidth. UAV applications may therefore require line-of-sight communications to minimize quality degradation, which is problematic in urban or mountainous areas. Even given free line of sight, bandwidth requirements will also place strict restrictions on the maximum achievable communication range. These limitations are particularly important when smaller UAVs are used, such as the 500-gram battery-powered LinkMAV Micro Aerial Vehicle (Duranti et al., 2007). In these situations, transmitting information directly to the base station can be difficult or impossible.

Free line of sight to the base station could in some situations be achieved through increased altitude. However, surveillance typically also requires free line of sight to the target. Keeping both locations in view simultaneously may require flying at very high altitudes, especially in urban areas with closely spaced buildings. Such altitudes are not always feasible, due to limitations of the UAV or due to aviation regulations. Smaller UAVs may also be unable to fly at sufficient altitude to avoid detection in the case of military or police surveillance, preferring instead to fly at lower altitudes in order to hide behind obstacles. Furthermore, increasing the altitude of the UAV also increases the distance to the base station, which may lead to exceeding the maximum communication range, and increases the distance to the target, thereby decreasing the quality of the information the UAV is able to gather.

As an alternative solution, both intervening obstacles and limited range can be handled using a chain of intermediate *relay UAVs* passing on information from the surveillance UAV to the base station (Figure 1). A surveillance UAV can then be placed freely in a location that yields information of high quality.

We are therefore interested in positioning relay UAVs to maximize the quality of the resulting chain, given a known target position. However, we are also interested in minimizing resource usage in terms of the number of relay UAVs

required. Given these two objectives, a variety of trade-offs are possible. For example, decreasing the distance between adjacent relay UAVs may improve transmission quality but instead requires additional relays.

Unless there is an unlimited number of UAVs, we need the ability to generate the highest quality relay chain possible for a given upper limit on the permitted number of relays. A surveillance mission may also be part of a wider operation, where missions are defined incrementally by ground operators over a period of time and take place concurrently. In this case some relay UAVs may have to be held back for future missions. Thus, it is often important to generate a spectrum of chains with different quality and UAV usage, allowing the ground operator to make a trade-off with full knowledge of the available options. Each such chain should then be *Pareto-optimal* (Miettinen, 1999). A solution is Pareto-optimal if it cannot be improved in any respect without a decrease in quality in another respect. For the problem at hand, this means that the quality of the chain cannot be improved without also increasing the number of relays.

This leads to a continuous bi-objective optimization problem, where algorithms must be sufficiently scalable to enable the use of a large number of comparatively inexpensive miniature UAVs with limited communication range.

The main contribution of this article consists of two new graph search algorithms applicable to relay positioning in discrete space. The first is a label-correcting algorithm applicable to the most complex of the problem variations, that of generating a spectrum of chains that are Pareto-optimal relative to a given discretization and near Pareto-optimal in continuous space. The second algorithm instead uses dual ascent techniques to generate relay chains with an upper bound on the number of relay UAVs. Both algorithms are considerably more efficient than the best known algorithm for these problems in the literature, allowing problems to be solved in less time for any given discretization, or with a denser discretization given a fixed amount of time.

Section 2 introduces several variations of the bi-objective optimization problem in continuous space. Since these variations are intractable in the continuous setting, we discretize the problem in Section 3, generating a corresponding graph where nodes correspond to potential relay positions and edge costs are used to model quality measures. We proceed to present the two new graph search algorithms in Sections 4 and 5. Section 6 contains the result of empirical performance tests in varying environments. Related work is discussed in Section 7, after which we present our conclusions and discuss future work in Section 8.

Though examples in this article focus on the use of UAVs, this method for relay positioning can equally well be used for relay placement for unmanned ground vehicles (UGVs), given a suitable discretization procedure that places nodes on the ground.

2 Relay Positioning Problems

Formally, we define a general relay positioning problem instance as follows.

We assume relays are placed in three dimensions. Let $F \subseteq \mathbb{R}^3$ be the region that is free from obstacles, defining the space through which free line of sight can be achieved. Let $U \subseteq F$ be the region where each individual UAV may safely be placed. This region must only include points sufficiently far away from obstacles for the required safety clearances to be satisfied. No-fly-zones where UAVs are not permitted may also be excluded from U. Let $x_0, x_t \in \mathbb{R}^3 \setminus U$ be the position of a base station and a surveillance target, respectively.

Assume as given two boolean reachability functions. The *communication reachability function* $f_{comm}(x, x')$ specifies whether communication between two entities at points $x, x' \in U$ should be considered feasible. It can be defined by a limited communication radius and a requirement of free line of sight (where all points between x and x' must be in F), by explicit models of 3D wave propagation, or by any other definition appropriate for the problem at hand. The *surveillance reachability function* $f_{surv}(x, x')$ specifies whether a surveillance UAV at $x \in U$ would be able to surveil a target at $x' \in \mathbb{R}^3 \setminus U$. This function must take into account suitable minimum and maximum ranges for surveillance as well as sensor-specific limitations such as cameras that cannot surveil targets in arbitrary directions, for example above the surveillance UAV.

A *relay chain* between x_0 and x_t is defined as a sequence of positions $[x_0, x_1, \ldots, x_k]$, where $\{x_1, \ldots, x_k\} \subseteq U$, such that $f_{comm}(x_i, x_{i+1})$ for all $i \in [0 \ldots k-1]$, and $f_{surv}(x_k, x_t)$. The *length* of a chain is defined as the number of agents required, including the base station: $len([x_0, x_1, \ldots, x_k]) = k + 1$.

A wide variety of problem-specific quality measures can be modeled using the general notion of *cost* and cost minimization. We therefore assume as given a non-negative *communication cost function* $c_{comm}(x, x')$ specifying the cost of relaying information between UAVs located at points $x, x' \in \mathbb{R}^3$. This cost may for example be related to transmission power requirements, risk of interrupted communication or intermittent dropouts, risk of detection by adversaries, the need to enter soft no-fly zones (locations that should preferably be avoided), or weighted combinations of such factors.

Similarly, we assume a non-negative *surveillance cost function* $c_{surv}(x, x')$ specifying the cost of a UAV at $x \in U$ surveilling a target at $x' \in \mathbb{R}^3 \setminus U$. This cost would be inversely related to the quality of the information that can be sensed given specific positions for the surveillance UAV and the target. If a camera is used, for example, higher quality may be achieved at shorter distances and at appropriate camera angles.

The *cost* of a relay chain $[x_0, x_1, \ldots, x_k]$, denoted by $cost([x_0, x_1, \ldots, x_k])$, is then defined as $(\sum_{i=0}^{k-1} c_{comm}(x_i, x_{i+1})) + c_{surv}(x_k, x_t)$.

Given a problem instance as defined above, including the position of the base station and the target, we can now identify a number of interesting single target relay positioning (STR) problems. Some of these problems assume an upper limit on the number of UAVs available, denoted by n_{uavs}.

STR-MinLengthMinCost: Find a relay chain of minimum length among the chains of minimum cost. A solution is a chain s such that for all other chains c, $cost(s) \leq cost(c)$ and $cost(s) = cost(c) \rightarrow len(s) \leq len(c)$. This corresponds to

using the highest quality chain achievable with access to an unlimited number of UAVs, while preferring to use fewer UAVs if this does not compromise quality.

STR-MinCostMinLength: Find a relay chain of minimum cost among the chains of minimum length. A solution is a chain s such that for all other chains c, $len(s) \leq len(c)$ and $len(s) = len(c) \rightarrow cost(s) \leq cost(c)$. This is useful if minimizing the number of UAVs is strictly more important than maximizing quality.

STR-MinCostLimited: Find a relay chain of minimal cost among the chains that use at most n_{uavs} UAVs. A solution is a chain s satisfying $len(s) \leq n_{uavs}+1$, such that for all other chains c, $len(c) \leq n_{uavs} + 1 \rightarrow cost(s) \leq cost(c)$. This corresponds to a desire to find the highest quality relay chain that can be realized within the given limit on the number of UAVs.

STR-ParetoLimited: Find a complete set of (strongly) Pareto-optimal relay chains using at most n_{uavs} UAVs. A relay chain s is Pareto-optimal if for all other chains c, $len(c) < len(s) \rightarrow cost(c) > cost(s)$ and $cost(c) < cost(s) \rightarrow len(c) > len(s)$: Any chain that is strictly better than s in one respect must be strictly worse in the other respect. A solution set is complete for up to n_{uavs} UAVs if a Pareto-optimal chain is included for every length $len \leq n_{uavs} + 1$ where such chains exist. This corresponds to finding a set of potential tradeoffs between the quality of the relay chain and the required number of UAVs. Using $n_{uavs} = \infty$ yields a complete set of Pareto-optimal chains of unbounded length.

3 Discretization

For the problems defined above, the number of local optima is typically large. Due to obstacles there are also often a large number of disjoint feasible subsets, which makes the continuous problem intractable (Burdakov et al., 2009b). We therefore suggest to discretize the environment and turn to a graph formulation.

Discretization requires selecting a finite set of positions $U' \subseteq U$ to be considered for UAV placement and generating corresponding graph nodes. Nodes must be sufficiently dense that U' remains connected, given limited communication ranges and possibly a line-of-sight restriction. Making good use of the maximum range in all directions, given possible obstacles, also requires high node density. Generating high-quality relay chains in discretized space therefore requires a sufficient node density, even in large obstacle-free areas. This is very different from node placement algorithms for graph-based path planners, where edges can be arbitrarily long and only the total length of a path matters. Thus, using such algorithms unmodified is inappropriate.

For the type of urban and mountainous terrain we are interested in, a simple regular three-dimensional grid has proven quite suitable. As an extension, grid cells could vary in size depending on the local obstacle density. Similarly, to improve connectivity in certain situations, a grid may be augmented with nodes placed randomly with a bias towards generating nodes near obstacles, nodes near the center of the local free space, or nodes in narrow passages (Boor et al.,

1999; Amato et al., 1998; Hsu et al., 2003; Wilmarth et al., 1999).

Each potential UAV position $x \in U'$ is associated with a unique node n. Similarly, the base station at x_0 and the target at x_t are associated with the distinct nodes n_0 and n_t, respectively. The set of all nodes is denoted by N.

Each $x, x' \in U' \cup \{x_0\}$ corresponding to $n, n' \in N$ and satisfying $f_{comm}(x, x')$ is associated with a directed edge $e = (n, n')$ of cost $c_{n,n'} = c_{comm}(x, x')$ representing the possibility and quality of communication between positions x and x'. Similarly, each $x \in U'$ corresponding to $n \in N$ and satisfying $f_{surv}(x, x_t)$ is associated with a directed edge $e = (n, n_t)$ of cost $c_{n,n_t} = c_{surv}(x, x_t)$ representing the possibility and quality of surveilling a target at x_t from a UAV at x. The set of all edges is denoted by E.

For any node $n \in N$, let $n_- = \{n' : (n', n) \in E\}$ refer to its set of predecessor nodes and $n_+ = \{n' : (n, n') \in E\}$ refer to its set of successor nodes. By the *length* of a path, we will refer to the number of edges in the path. This is also referred to as the *hop count*, where each edge corresponds to one hop. The *cost* of a path is then defined as the sum of the edge costs along the path. Thus, a *shorter* path has fewer edges, while a *cheaper* path has lower cost. As for relay chains, $cost(\pi)$ denotes the cost of the path π and $len(\pi)$ denotes its length.

Any path from n_0 to n_t in the graph $G(N, E)$ corresponds directly to a relay chain of identical length and cost from x_0 to x_t. We will sometimes use these terms interchangeably. The graph is directed due to the possibility of asymmetric cost functions. By the construction, there cannot be an edge between n_0 and n_t: The base station itself cannot surveil a target.

The discrete **STR-MinLengthMinCost** and **STR-MinCostMinLength** problems can be solved efficiently by for example Dijkstra's algorithm, using compound path costs (see also Section 4.2). The remaining problems will be considered in more detail below.

4 Generating Pareto-Optimal Relay Chains

To solve the discrete **STR-ParetoLimited** problem, we first consider its relation to the *all hops optimal path* (AHOP) problem (Guérin and Orda, 2002): For each $k = 1, 2, \ldots, L$, find a cheapest path from n_0 to n_t among the paths of length at most k. We call such paths *k-restricted cheapest paths*.

A path of length k is Pareto-optimal if it is a k-restricted cheapest path *and* there exists no strictly shorter path of equal cost. Thus, the problems are not identical. However, given a complete set of k-restricted cheapest paths for all $1 \leq k \leq L$, a complete set of Pareto-optimal solutions of length at most L can be found by grouping the AHOP paths by cost and selecting a path of minimum length in each group. The discretized **STR-ParetoLimited** problem can therefore be solved using an AHOP algorithm.

Conversely, any **STR-ParetoLimited** algorithm can also be used to solve the AHOP problem. If there exists a Pareto-optimal solution of length k, then this path must also be a k-restricted cheapest path. If all Pareto-optimal solutions are longer than k, then all paths from n_0 to n_t are longer than k, and no

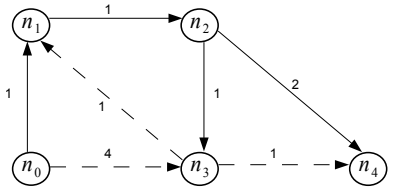


Figure 2: Example graph labeled with edge costs.

k-restricted cheapest path exists. If there are Pareto-optimal solutions strictly shorter than k, any Pareto-optimal path of the greatest length $k' < k$ is a k-restricted cheapest path.

We illustrate these relations in Figure 2. For $n_t = n_4$, there is only one path of length 2, namely $\pi_1 = n_0 \to n_3 \to n_4$ with $cost(\pi_1) = 5$. This path is a 2-restricted cheapest path: There is no cheaper path of length at most 2. Since there is also no equally cheap path of length strictly less than 2, it is also Pareto-optimal. There is one path of length 3, $\pi_2 = n_0 \to n_1 \to n_2 \to n_4$ with $cost(\pi_2) = 4$, which is both a 3-restricted cheapest path and a Pareto-optimal solution. However, consider the path $\pi_3 = n_0 \to n_1 \to n_2 \to n_3 \to n_4$, also of cost 4. Both π_2 and π_3 are 4-restricted cheapest paths, since they have the same minimal cost, but only π_2 is Pareto-optimal.

4.1 Existing Solutions to the AHOP Problem

At its k:th iteration, the standard Bellman-Ford algorithm generates all cheapest paths that originate in the designated start node and make use of at most k edges. These paths are replaced in later iterations, if paths of lower cost are found using additional edges.

Lawler (1976) presented a modified algorithm, originally called the method of successive approximations. In contrast with the Bellman-Ford algorithm, it retains old paths when cheaper but longer paths are found, generating in its k:th iteration a set of k-restricted cheapest paths of minimal length.

Balakrishnan and Altinkemer (1992) introduced a length bound L, truncating the calculation so that only paths of length $k \leq L$ are considered. This results in the best currently known algorithm for the AHOP problem (Figure 3), with a time complexity of $O(L\,|E|)$. It is natural to require that $L \leq |N| - 1$, since no cheapest path can be longer than $|N| - 1$. Therefore, the time complexity of the algorithm can also be expressed as $O(|N||E|) \subseteq O(|N|^3)$.

Though this algorithm is applicable to the discretized **STR-ParetoLimited** problem, it uses a considerable amount of time and is not practically useful for larger problem instances involving large maps, fine-grained discretizations, or long communication ranges. This is especially true in mixed-initiative settings where an operator initiating a surveillance mission expects a prompt result.

```
1  for each n ∈ N do g_0(n) ← +∞,  p_0(n) ← nil
2  g_0(n_0) ← 0
3  for k = 1, ..., L do
4     for each n ∈ N do
5        g_k(n) ← g_{k−1}(n),  p_k(n) ← p_{k−1}(n)
6     for each n ∈ N do
7        for each n′ ∈ n_+ do
8           c ← g_{k−1}(n) + c_{n,n′}
9           if c < g_k(n′) then
10             g_k(n′) ← c,  p_k(n′) ← n
```

Figure 3: Algorithm 1 – Truncated successive approximation algorithm.

k (path length)	g_k (cost)	p_k (predecessor)
1	4	n_0
3	3	n_2

Table 1: Reachability record for n_3 after execution of the new algorithm.

4.2 A New Label-Correcting Algorithm

We will now present a new algorithm that solves the **STR-ParetoLimited** problem, generating for each node a complete set of Pareto-optimal solutions of length at most $n_{uavs}+1$. Like Algorithm 1, it can be viewed as a label-correcting Bellman-Ford-type algorithm. However, through efficient use of preprocessing, a large percentage of the calculations performed by Algorithm 1 can be avoided.

The new algorithm incrementally generates and updates a set of *reachability records* for each node. Each record $\langle k, g_k, p_k \rangle$ indicates that the node can be reached from the base station in k hops at a cost of g_k using the predecessor p_k.

Table 1 shows two reachability records for the node n_3 in Figure 2, sorted by increasing path length. The first record corresponds to using a minimal number of UAVs: The shortest path to n_3 regardless of cost is of length 1. Subsequent records correspond to using increasingly greater numbers of UAVs, yielding paths with progressively lower cost, until we reach the least-cost path using the largest number of UAVs. Any missing intermediate values of k indicate that even if chains of length k exist, they are not Pareto-optimal. For example, Table 1 lacks a record for $k = 2$ because a chain of length 2 would be at least as expensive as one of length 1, the nearest smaller value of k present in the table, so using such a chain would be pointless. A chain of length 3 could be a useful alternative, though, as this would increase quality and reduce the cost to 3.

We will show that after termination, the fact that a target node n_t is associated with a reachability record $\langle k, g_k, p_k \rangle$ corresponds exactly to the existence of a Pareto-optimal relay chain between n_0 and n_t of length $k \leq n_{uavs} + 1$ and

cost g_k, allowing n_t to be reached from the base station node using $k-1$ intermediate UAVs, one of which is the surveillance UAV. A complete relay chain from n_0 to n can always be reconstructed (in reverse order) by considering the reachability record of the predecessor p_k for $k-1$ hops, continuing recursively until n_0 is reached.

Preliminaries. To present our algorithm and justify its correctness, we need the following definitions, illustrated by Table 2.

Node n is called a *k-hop Pareto* node if there exists a Pareto-optimal path of length k from n_0 to n. We call this path *k-hop Pareto-optimal*. For any k, the set of all k-hop Pareto nodes is denoted by N_k. For example, n_4 is a 2-hop Pareto node as well as a 3-hop Pareto node. Thus, a node may be present in multiple N_k sets. However, n_4 is not a 4-hop Pareto node: Though it can be reached in 4 steps, the resulting path is not Pareto-optimal. We have $N_0 = \{n_0$, as the initial node is the only node reachable in zero steps.$\}$

Our algorithm exploits the following property of Pareto-optimal paths: Any Pareto-optimal path of length k must consist of a sequence of nodes occurring in N_0, N_1, ..., up to N_k. In other words, any non-empty Pareto-optimal path to a node n' must be an immediate extension of a shorter Pareto-optimal path to a predecessor node n.

Theorem 1. *Let $n' \in N_k$. Suppose that*

$$\pi' = n_0 \to \cdots \to n \to n'$$

is a corresponding k-hop Pareto path. Then $n \in N_{k-1}$, and the path π composed by the first $k-1$ hops of this path is a $(k-1)$-hop Pareto path from n_0 to n.

Proof. The proof of these statements follows trivially from the k-hop Pareto-optimality of π', which implies that there is no path π'' from n_0 to n such that:

$$\mathit{len}(\pi'') < \mathit{len}(\pi) \quad \text{and} \quad \mathit{cost}(\pi'') \leq \mathit{cost}(\pi)$$

or such that:

$$\mathit{len}(\pi'') = \mathit{len}(\pi) \quad \text{and} \quad \mathit{cost}(\pi'') < \mathit{cost}(\pi). \qquad \square$$

Theorem 1 implies that lines 7–10 in Algorithm 1 only have to be executed for nodes $n \in N_{k-1}$: Outgoing edges from other nodes cannot yield Pareto-optimal paths and therefore cannot result in updated node costs or new reachability records. As will be shown below, N_{k-1} can be generated incrementally during iteration $k-1$, after which it can be used in the following iteration.

Our algorithm makes use of the following identification of k-hop Pareto nodes: We can find a cheaper path to a node n using k hops than we could using $k-1$ hops if and only if $n \in N_k$, that is, if and only if there is a Pareto-optimal path to n using k hops:

$$n \in N_k \quad \Longleftrightarrow \quad g_k(n) < g_{k-1}(n). \tag{1}$$

Preprocessing. In addition to using the sets N_k, many calculations performed by Algorithm 1 can be avoided if we know the length of the *longest* (and therefore

k	n_0	n_1	n_2	n_3	n_4	N_k	N_k^*
0	0^*	-	-	-	-	$\{n_0\}$	$\{n_0\}$
1	-	1^*	-	4	-	$\{n_1, n_3\}$	$\{n_1\}$
2	-	-	2^*	-	5	$\{n_2, n_4\}$	$\{n_2\}$
3	-	-	-	3^*	4^*	$\{n_3, n_4\}$	$\{n_3, n_4\}$

Table 2: Costs of k-hop Pareto paths, and the sets N_k and N_k^*, for the graph in Figure 2. Asterisks indicate costs corresponding to cheapest paths.

cheapest) Pareto-optimal path from n_0 to each node, disregarding any length bounds. It follows from the definition that such paths must have minimal length among the paths of minimum cost. They will be called *minimum length minimum cost paths*, abbreviated *MLMC-paths*. For node n_4 in Figure 2, there are two paths of minimal cost: $n_0 \to n_1 \to n_2 \to n_4$ and $n_0 \to n_1 \to n_2 \to n_3 \to n_4$. Only the first path is an MLMC-path, as it is of minimal length within the set.

To generate MLMC-paths, we can use compound path costs of the form $\langle \mathit{len}(\pi), \mathit{cost}(\pi) \rangle$. We can then apply a slight modification of Dijkstra's algorithm with a path preference relation where $\langle \mathit{len}_1, \mathit{cost}_1 \rangle < \langle \mathit{len}_2, \mathit{cost}_2 \rangle$ if ($\mathit{cost}_1 < \mathit{cost}_2$) or ($\mathit{cost}_1 = \mathit{cost}_2$ and $\mathit{len}_1 < \mathit{len}_2$). This results in a tree of MLMC-paths from n_0 to all other nodes in N, called an *MLMC-tree*.

Let N_k^* denote the set of all nodes n for which the longest Pareto-optimal path from n_0 to n is of length exactly k. We see that the sets N_k^* partition N and that $N_k^* \subseteq N_k$. We also see that the nodes in N_k^* are exactly the nodes of depth k in an MLMC-tree. In the new algorithm (Figure 4), we therefore begin by creating an MLMC-tree and extracting all N_k^* sets. Table 2 shows these sets for the graph in Figure 2, together with other properties to be explained below.

Let $k^*_{\max}$ denote the height of the MLMC-tree. It is clear that no relay chain of more than $k^*_{\max}$ nodes can be Pareto-optimal, since all longer paths must also be at least as expensive. This will limit the number of relays required for any optimal relay chain for this graph, as well as the depth to which the graph needs to be searched. We therefore extract $k^*_{\max}$ from the MLMC-tree as well.

As a final part of preprocessing, we use the MLMC-tree to create an initial set of reachability records corresponding to the longest Pareto-optimal relay chain for each node. We can then skip the execution of lines 7–10 in Algorithm 1 for the nodes $n' \in N_k^*$ in iteration k, because their reachability records and the costs $g_k(n')$ of the corresponding k-hop Pareto-optimal paths were already extracted from the MLMC-tree. Thus, lines 7–10 need only be executed for edges (n, n') where $n \in N_{k-1}$ and $n' \notin N_k^*$.

Initialization. In addition to the reachability records, our new algorithm (Figure 4) uses $g(n)$ to denote the cost of the cheapest path found so far from n_0 to n. Initially, $g(n) = \infty$ for all nodes except n_0 (line 1).

Lines 2–4 provide initial values for the initial node n_0. Since $N_0^* = \{n_0\}$, a reachability record with $g_0(n_0) = 0$ must have been created during preprocess-

```
0  Calculate MLMC-tree, extract k*_max and all N*_k, generate initial records
1  for each n ∈ N \ {n_0} do g(n) ← +∞
2  for each n ∈ n_0− do                  // Incoming edges...
3     E ← E \ {(n, n_0)}                 // ... are removed
4  N_0 ← {n_0}
5  for k = 1, ..., min{n_uavs + 1, k*_max − 1} do
6     for each n' ∈ N*_k do
7        for each n ∈ n'_− do             // Incoming edges...
8           E ← E \ {(n, n')}            // ... are removed
9     N_k ← N*_k
10    for each n ∈ N_{k−1} do
11       for each n' ∈ n_+ do
12          c ← g_{k−1}(n) + c_{n,n'}    // To n' through n in k hops
13          if c < g(n') then
14             g(n') ← c                 // Lowest cost so far
15             g_k(n') ← c               // Lowest cost in k hops
16             p_k(n') ← n               // Predecessor for k hops
17             N_k ← N_k ∪ {n'}
```

Figure 4: Algorithm 2 – MLMC-tree-based label-correcting algorithm.

ing, which indicates that n_0 can be reached with cost 0 in 0 hops. Furthermore, since the value $g(n)$ where $n \in N_k^*$ is not used in iteration k or any subsequent iteration, no value of $g(n_0)$ needs to be assigned. Clearly, no chain ending in n_0 can improve the value $g_0(n_0) = 0$, so all incoming edges to n_0 can be removed (lines 2–3). Finally, line 4 prepares for the first iteration by setting $N_0 = \{n_0\}$, indicating that any 1-hop Pareto-optimal path must consist of a path to n_0 in 0 hops (the empty path) followed by a single outgoing edge. This line handles all paths of length 0.

Main Algorithm. The longest (and therefore cheapest) Pareto-optimal path to each node was generated during preprocessing. The main part of the algorithm then finds the shortest (and therefore most expensive) Pareto-optimal path, and continues in order of increasing length and decreasing cost.

Each iteration of lines 5–17 considers paths of a particular length $k \geq 1$. The upper bound, $\min\{n_{uavs}+1, k^*_{\max}-1\}$, reflects the fact that (i) we allow at most n_{uavs} UAVs, yielding a total path length of up to $n_{uavs}+1$ when edges to the base station and target are included, (ii) no paths of length greater than $k^*_{\max}$ can be Pareto-optimal, due to the meaning of $k^*_{\max}$, and (iii) any Pareto-optimal path of length exactly $k^*_{\max}$ was already generated during the preprocessing.

When iteration k begins, all reachability records for any node $n' \in N_k^*$ must already have been created: By the definition of N_k^*, no Pareto-optimal path from n_0 to n' can be strictly longer than k hops, and a record for the given value of k was already created during preprocessing. Thus, no new paths ending in any $n' \in N_k^*$ can be Pareto-optimal, and we can remove all incoming edges to n'

without affecting correctness or optimality (lines 6–8). However, we do need to consider longer paths going *through* these nodes (line 9, explained below).

In lines 10–17, we consider all potentially Pareto-optimal relay chains of length k. Recall that any such chain must consist of a path to a node $n \in N_k$ followed by a single outgoing edge from n. We consider each such path in turn, determining its destination n' and calculating its cost c (lines 10–12). If the path has lower cost than the cheapest path found previously (with a cost of $g(n')$) we create a new reachability record with $g_k(n')$ set to the new path cost and $p_k(n')$ set to the new predecessor n (lines 13–16). Note that though reachability records are sparse, we know that any node in the set N_{k-1} does have a reachability record for $k-1$ due to the construction of such sets.

What remains is to prepare for the next iteration by constructing N_k according to its definition and characterization (relation 1). That is, we should construct N_k in such a way that any $(k+1)$-hop Pareto-optimal relay chain necessarily consists of a path of length k to a node $n \in N_k$ followed by a single outgoing edge from n. It is clear that no Pareto-optimal chain would use k hops to reach a node n if it could be reached in $k-1$ hops without incurring additional costs. This is stated in relation (1) which also covers the cases when n cannot be reached at all with paths of length $k-1$. Thus, it is only necessary to consider nodes whose costs were decreased by allowing paths of length k. This is achieved in line 9, for nodes in N_k^* where we found a cheapest path of length k during the preprocessing, and in line 17, for nodes where we found a path of length k and of lower cost in this iteration.

Algorithm Properties. The following result summarizes the main properties of the new label-correcting algorithm (Algorithm 2).

Theorem 2. *After iteration k of the algorithm in Figure 4, N_k correctly defines the set of all k-hop Pareto nodes, $g_k(n)$ defines, for each $n \in N_k$, the cost of a corresponding k-hop Pareto-optimal path, and $p_k(n)$ defines the predecessor of n in this path. Moreover, $g(n)$ defines the cost of a cheapest path using at most k hops from n_0 to n for each $n \in N$ such that $n \in N_{k'}^*$ with $k' > k$.*

Proof. The claims of this theorem are easily proved by induction in $k = 0, 1, \ldots, \min\{n_{uavs} + 1, k^*_{\max} - 1\}$ with the use of relation (1) and Theorem 1. □

The difference between the two algorithms is illustrated in Table 2, which shows all the values of $g_k(n)$ produced by Algorithm 2. In contrast, Algorithm 1 produces the values of $g_k(n)$ for every node n and for each $k = 1, \ldots, 4$. Furthermore, it makes the same calculations for producing identical values, for instance, $g_1(n_3) = g_2(n_3) = 4$ and $g_3(n_3) = g_4(n_3) = 3$. For the same node, our algorithm calculates $g_1(n_3)$ only, while $g_3(n_3)$ is provided by the MLMC-tree.

Note that for Lawler's method as well as for the new algorithm, much of the solution is independent of the target location and could be pre-calculated before the target is known. However, the methods do depend on the location of the base station. This is not always known in advance, since base stations can be mobile. Ground operators may then experiment with a variety of hypothetical base station locations before making a decision, after which both the UAVs

and the base station move to their intended positions. For every hypothetical location, a new set of Pareto-optimal chains must be calculated. Similarly, the solution must depend on the graph itself, which may change if new obstacles or no-fly-zones are detected. Even localized changes in the graph can yield major changes in the Pareto-optimal relay chains, again forcing a complete recalculation.

See Burdakov et al. (2009b) for additional details and proofs.

Time Complexity. The preprocessing phase in line 0 is dominated by a single call to Dijkstra's algorithm, which can be performed in $O(|E| + |N| \log |N|)$ time. Line 1 can be performed in $O(|N|)$ time, while lines 2–4 require at most $O(|E|)$ time. Thus, the time complexity of the pre-processing and initialization phase is in $O(|E| + |N| \log |N|) \subseteq O(|N|^2)$.

The main loop is performed at most $k^*_{\max} - 1$ times. Lines 6–8 and lines 10–17 iterate over a subset of the edges in the graph, covering each edge at most once. They consequently require at most $O(|E|)$ time, which is also an upper bound for line 9. The complexity of the loop is therefore in $O((k^*_{\max} - 1)|E|) \subseteq O((k^*_{\max} - 1)|N|^2)$. The overall time complexity of Algorithm 2 is then in $O(k^*_{\max}|N|^2)$. It could also be stated that the complexity is in $O(|N|^3)$. However, it should be noted that $k^*_{\max}$ corresponds to the maximum number of UAVs required to reach any target, which is typically far smaller than the number of nodes ($k^*_{\max} << |N|$). Thus, $O(|N|^3)$ is a significant overestimate for our applications.

Algorithm 1 has a time complexity of $O(|N|^3)$. By applying the widely used improvement of terminating at the earliest iteration where no node cost has been decreased, this bound can be tightened to $O(k^*_{\max}|N|^2)$. Thus, the time complexity of the algorithms would seem to be similar. However, whereas Algorithm 1 always considers every edge in E in every iteration, our estimate that lines 6–17 would cover each edge *at most* once is truly an overestimate, due to the use of preprocessing and node partitioning. As indicated by the empirical testing, far fewer edges are considered in each iteration in practice.

4.3 Generalizations

Algorithm 2 can be modified to work with multiple base stations or pre-placed relays, and can in a single execution generate relay chains to an arbitrary number of potential target locations.

Multiple Base Stations. Suppose that our goal is to transmit information to any of a set of b fixed-location base stations, which are in turn connected to each other and thereby to the ground operator by high bandwidth communication links. We then want to choose the best station to communicate with for each chain length. For example, the highest quality chain of length 5 may lead back to base station 1, while the best chain of length 6 leads to base station 4.

This problem can obviously be solved by running Algorithm 2 for each base station in turn, together with post-processing to determine which base station to use for each chain length. However, this requires time linear in the number of base stations. Instead, we can solve the problem for multiple base stations in

0 Calculate MLMC-graph, extract $k^*_{\max}$ and all N^*_k, generate initial records
1 **for** each $n \in N$ **do** $g(n) \leftarrow +\infty$
2 **for** each $n \in \{n_0^1, \ldots, n_0^b\}$ and each $n' \in n_-$ **do**
3 $\quad E \leftarrow E \setminus \{(n', n)\}$ *// Remove incoming edges*
4 $N_0 \leftarrow \{n_0^1, \ldots, n_0^b\}$
5 *Continue as in Algorithm 2*

Figure 5: Algorithm 3 – Label-correcting algorithm, multiple base stations.

a single call by making use of multiple start nodes.

We define a relay chain between a *set* of b base station locations $B = \{x_0^1, \ldots, x_0^b\} \subseteq \mathbb{R}^3 \setminus U$ and a single target location $x_t \in \mathbb{R}^3 \setminus U$ as a sequence $[x_0^j, x_1, \ldots, x_k]$, where $x_0^j \in B$ is the position of the base station to which information is transmitted and $\{x_1, \ldots, x_k\} \subseteq U$, such that $f_{comm}(x_0^j, x_1)$ and for all $i \in [1 \ldots k-1]$, $f_{comm}(x_i, x_{i+1})$ and $f_{surv}(x_k, x_t)$.

The graph representation is extended analogously, with n_0^1 through n_0^b denoting nodes corresponding to the given base station positions. Graph generation must also create communication links for all base stations: For each base station position $x_0^j \in B$ and each $x \in U'$ corresponding to $n \in N$ and satisfying $f_{comm}(x_0^j, x)$, an edge $e = (n_0^j, n)$ of cost $c_{n_0^j, n} = c_{comm}(x_0^j, x)$ is created.

In the preprocessing step of Algorithm 2, the priority queue of Dijkstra's algorithm must be initialized not with a single node but with the set of all base station nodes $\{n_0^1, \ldots, n_0^b\}$, each having zero cost. The main algorithm must also be altered according to Figure 5, ensuring that incoming edges to all base stations are removed, and that the base stations are part of N_0 as starting nodes for edges resulting in paths of length 1. The resulting time complexity is in $O((b + |N|)^3)$, significantly better than running an $O(|N|^3)$ algorithm b times.

Multiple Potential Target Locations. In certain situations, we may want to pre-calculate sets of Pareto-optimal relay chains to a set of potential target locations. For example, this could allow the use of a denser discretization than could be used in a real-time setting. Depending on the degree of uncertainty as to where a target will turn up as well as the degree of precision desired when a surveillance UAV is placed, such locations could easily number in the hundreds or even in the thousands. Though the problem can again be solved by repeatedly applying Algorithm 2 for each individual target location, the time requirements of doing so may be prohibitive.

A more efficient approach would make use of the fact that in order to find a Pareto-optimal set of paths to the designated target node, Algorithm 2 in fact generates a Pareto-optimal set of paths from the current start node to *all* other nodes reachable within the optional limit on the permitted number of hops. Thus, we merely require two changes to the graph creation procedure. First, instead of generating a single target node, generate one target node n_t^i for every position x_t^i in a set $T = \{x_t^1, \ldots, x_t^t\} \subseteq \mathbb{R}^3$ of potential target positions. Second,

instead of creating incoming edges to a single target node, create edges to all target nodes: For each target position $x_t^j \in T$ and each $x \in U'$ corresponding to $n \in N$ and satisfying $f_{surv}(x, x_t^j)$, create an edge $e = (n, n_t^j)$ of cost $c_{n,n_t^j} = c_{surv}(x, x_t^j)$ representing the fact that a surveillance UAV at x would be able to surveil the target at x_t^j.

As in the case of multiple base stations, executing Algorithm 2 for this extended problem will only take marginally more time than generating relay chains to a single target position, given that the number of potential targets is not too large compared to the number of nodes in U'. Whenever a path to any of the given potential target locations is desired, a set of Pareto-optimal relay chains can efficiently be extracted from the reachability records of the given target node in the usual way.

This is not an optimal solution to the problem of surveilling multiple targets *simultaneously*. However, it can be highly useful in the case where a number of *potential* target positions are known in advance.

5 Generating Chains of Bounded Length

We now turn our attention to the **STR-MinCostLimited** problem, where the objective is to find the cheapest relay chain possible under a given length bound. The discretized version of this problem corresponds directly to the *hop-constrained shortest path* problem (see for example Dahl and Gouveia (2004)), where "shortest" in this case means "lowest cost".

Though only one path is required, the best known algorithm for hop-constrained shortest paths remains the truncated version of Lawler's method of successive approximations (Algorithm 1), which is executed up to the given bound. Consequently, the new label-correcting algorithm presented in the previous section (Algorithm 2) can be used for this problem as well.

However, both of these approaches entail generating a number of relay chains that are eventually discarded, indicating that more efficient algorithms may be found. Additionally, though the path generated in pre-processing has maximum length among all Pareto-optimal paths, the algorithm then generates additional paths in order of *increasing* number of hops. For a particular problem, the algorithm may first generate a path using 17 hops, followed by paths using 8, 9, 10, 12, 14 and 15 hops. Thus, even if the initial path has only slightly too many hops, one must continue until the set of Pareto-optimal solutions is almost complete. There is no advantage to the fact that the first path was "almost" feasible. We therefore present another method explicitly designed to take advantage of this fact.

A Dual Ascent Method. The new method is based on the *dual ascent* approach. In optimization terms, the main idea is to maximize the piece-wise affine Lagrangian dual function by traversing one affine segment at a time. Dual ascent methods have been shown to be very efficient for certain types of problems, especially uncapacitated facility location problems (Erlenkotter, 1978), uncapacitated network design problems (Balakrishnan et al., 1989) multicommodity

1 $\alpha \leftarrow \alpha_0$
2 **loop**
3 Calculate MLMC-tree from n_0 using costs $c'_{n,n'} = c_{n,n'} + \alpha$
4 From the tree, obtain y_n, q_n for all n, and a path π from n_0 to n_t
5 **if** $len(\pi) \leq n_{uavs} + 1$ **then return** π
6 $S \leftarrow \{(n, n') \in E : q_{n'} > q_n + 1\}$
7 **if** $S = \varnothing$ **then fail** *// No path can be shortened*
8 Calculate $\epsilon_{n,n'} \leftarrow \frac{(y_n + c'_{n,n'}) - y_{n'}}{q_{n'} - (q_n + 1)}$ $\forall (n, n') \in S$
9 $\epsilon \leftarrow \min \epsilon_{n,n'}$
10 $\alpha \leftarrow \alpha + \epsilon$

Figure 6: Algorithm 4 – Dual ascent algorithm.

network flow problems (Barnhart, 1993), and other types of location problems (Holmberg and Jörnsten, 1996).

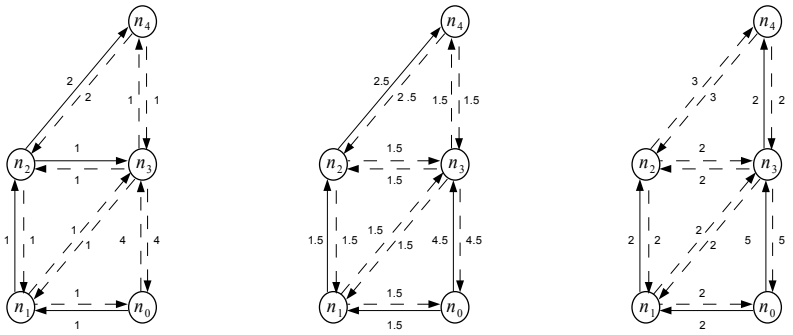
Intuitively, the dual ascent algorithm (Algorithm 4 in Figure 6) begins by creating an MLMC-tree using Dijkstra's algorithm. If the resulting relay chain uses too many hops, the algorithm gradually increases a value $\alpha \geq 0$ that is added to all edge costs, thereby favoring paths that use fewer edges. Each increase ϵ added to α is calculated systematically as the smallest value that causes at least one path to be shortened. Eventually, α reaches a level where the path from n_0 to n_t extracted from the MLMC-tree uses sufficiently few hops, or where the path cannot be shortened, in which case no solution exists.

The algorithm thus makes use of two distinct path cost measures. The *true cost* of a path π is defined only in terms of the communication and surveillance edge costs specified for a particular problem instance and is denoted by $cost(\pi)$. When an MLMC-tree is generated, however, the original edge costs have been incremented by α. We define the *modified cost* of a path π for a given α, denoted by $modc(\pi, \alpha)$, as the cost of π given such modified edge costs. It is clear that $modc(\pi, \alpha) = cost(\pi, \alpha) + len(\pi) \cdot \alpha$: Increasing α penalizes paths in strict proportion to their hop counts.

In line 1, α is given the initial value α_0. In most cases we use $\alpha_0 = 0$, causing the algorithm to begin by generating the cheapest paths possible using original edge costs. Higher values of α_0 decrease the time requirements of the algorithm but also increase the risk of generating paths that are shorter, and therefore more expensive, than strictly necessary.

In line 3, an MLMC-tree is calculated using the current modified edge costs $c'_{n,n'} = c_{n,n'} + \alpha$. This yields, for each node n, its current depth q_n (the number of hops along the MLMC-path from the root to n) and the modified cost y_n of the path from n_0 to n given the current value of α (line 4). If the resulting path π from n_0 to n_t has at most $n_{uavs} + 1$ edges (one to each UAV and one from the surveillance UAV to the target), it is a feasible solution to the hop-constrained problem and can be returned (line 5).

In lines 6–9, the algorithm calculates a suitable ϵ to be added to the α used



(a) The MLMC-tree calculated in iteration 1 is marked by solid edges.

(b) In iteration 2, the edge (n_0, n_3) enters the MLMC-tree while (n_2, n_3) is removed.

(c) The edge (n_3, n_4) enters the MLMC-tree in iteration 3, yielding a new MLMC-path to n_4.

Figure 7: A small example with start node n_0 and goal node n_4.

in the next iteration. The intention is for ϵ to be large enough that at least one path will be shortened, but also small enough that no solution is skipped. The calculation considers all possibilities of shortening the path to any node $n' \in N$ using an edge $(n, n') \in E$ that is not present in the current MLMC-tree. Making no other changes than using (n, n') to reach n' would place n' at depth $q_n + 1$. For this to be a strict improvement, we must have $q_{n'} > q_n + 1$ (line 6).

If no such edges exist, no path in the current tree can be shortened, no solution exists, and the algorithm terminates with failure (line 7). Otherwise, for any edge $(n, n') \in S$, the current path to n' could be shortened by replacing it with the current path to n together with the edge between n and n'. Since this has not already been done, we know that $y_n + c'_{n,n'} > y_{n'}$, that is, going through n is currently more expensive. Making the path through n *equally* expensive[1] entails increasing the cost of the longer path *more* than we increase the cost of the shorter path, by the amount of $(y_n + c'_{n,n'}) - y_{n'}$. This additional cost has to be split by $q_{n'} - (q_n + 1)$ edges (the number of edges by which the path can be shortened), yielding the expression for $\epsilon_{n,n'}$ found in line 8. Finally, we increase α by the minimum of all such $\epsilon_{n,n'}$. This is the smallest increment that guarantees that the tree will change in the next iteration. Using a greater increment would also be possible, but may cause the algorithm to skip solutions.

Example. Consider the objective of finding a relay chain of length at most 2. The algorithm starts with $\alpha_0 = 0$. In Figure 7a, edges included in the MLMC-tree calculated in iteration 1 are solid while other edges are dashed. The MLMC-path from n_0 to n_4 visits the nodes $[n_0, n_1, n_2, n_4]$, with length 3 and cost 4. Note in particular that the edge (n_3, n_4) is not included in the MLMC-tree, as

[1]Recall that the cost function for the MLMC-tree will prefer the shorter of two equally expensive paths. Thus, making the path through n strictly less expensive is not necessary.

reaching n_4 through (n_2, n_4) yields the same cost but requires one edge less. In iteration 1, $S = \{(n_0, n_3)\}$, yielding $\epsilon = \epsilon_{(n_0,n_3)} = \frac{(0+4)-3}{3-(0+1)} = 0.5$. We then get $\alpha = 0 + \epsilon = 0.5$.

In iteration 2 (Figure 7b), the new value of α increases modified edge costs, causing the edge (n_0, n_3) to be included in the tree instead of (n_2, n_3). Though this modifies the path to n_3, the path to the target node n_4 remains the same, now with a modified cost of $1.5 + 1.5 + 2.5 = 5.5$ versus $4.5 + 1.5 = 6$ for $[n_0, n_3, n_4]$. We find $S = \{(n_3, n_4)\}$, and therefore $\epsilon = \epsilon_{(n_3,n_4)} = \frac{(4.5+1.5)-5.5}{3-(1+1)} = 0.5$. This yields $\alpha = 0.5 + \epsilon = 1$, which is applied to all edge costs.

In iteration 3 (Figure 7c), the edge (n_3, n_4) becomes part of the new MLMC-tree, which yields an MLMC-chain to n_4 of length 2 and modified cost 7 visiting the nodes $[n_0, n_3, n_4]$. The true cost of this chain is 5. Thus, the chain is shorter but more expensive than the first relay chain that was found, which had a length of 3 and a true cost of 4. As we have found a relay chain of the desired length, the algorithm terminates.

Formal properties. The two following monotonicity results with respect to α can be shown to hold (Burdakov et al., 2008): *Increasing α cannot increase the length of an optimal path*, and *increasing α cannot decrease the cost of an optimal path.* Therefore, the node prices y_i are nondecreasing and the depths q_i are non-increasing as functions of α.

In each iteration, the algorithm creates an MLMC-tree. We then increase α by the minimum of all $\epsilon_{n,n'}$, which guarantees that the depth of at least one end node n' will decrease (a shorter path will be found). Since node depths cannot increase, the *total depth* T_D of the tree, defined as the sum of the depths of all nodes ($\sum_{n \in N} q_n$), must decrease by at least one.

The minimum total depth of a tree is $|N| - 1$, which occurs if the tree is a star. The maximum possible depth of $|N|(|N| - 1)/2$ occurs when the entire tree is a simple path (i.e. when only two nodes have degree one). Thus, T_D can be decreased at most $|N|(|N| - 1)/2 - (|N| - 1) = (|N| - 2)(|N| - 1)/2$ times, and the number of iterations of Algorithm 4 is bounded by $O(|N|^2)$.

An MLMC-tree can be generated in $O(|E| + |N| \log |N|)$, and the edges in S can be found and tested in $O(|E|)$. We conclude that Algorithm 4 is finite, and that its computational complexity is in $O(|N|^2(|E|+|N| \log |N|)) \subseteq O(|N|^4)$. In practice, however, most relay positioning problems require far fewer than $|N|^2$ iterations.

We refer the reader to Burdakov et al. (2008) for further details.

Optimized generation of MLMC-trees. For all iterations except the first, the generation of an MLMC-tree can be optimized by making use of the fact that an MLMC-tree has already been calculated, albeit with different modified edge costs.

We increase the cost of each edge in the previously created tree by the recently calculated ϵ, and increase the cost of each node n by $q_n \alpha$ (that is, in proportion to its depth). This ensures that the cost y_n of any node n correctly reflects the new modified cost of the path from n_0 to n.

However, the result is not an MLMC-tree: For some nodes, we can find

cheaper paths in the graph than those that are currently included in the tree. Any such paths must include at least one of the edges that yielded the current value of ϵ. We can therefore begin "repairing" the tree starting at the source nodes of those edges, rather than from the root, by initializing Dijkstra's algorithm with a priority queue containing exactly those source nodes. As the tree is traversed in the standard manner, only those nodes to which we find a cheaper path than in the initial solution are added to the priority queue. Thus, parts of the tree where the solution is already optimal will not be visited.

Single target. As specified above, Algorithm 4 calculates paths from the base station to *all* nodes in the graph. However, if we are only interested in a single target at a time, many nodes can be filtered out as irrelevant using two optimizations, both of which are used in our empirical testing.

After the first iteration, only nodes reachable within $n_{uavs} + 1$ hops are relevant to the search. These nodes can be calculated using a single call to Dijkstra's algorithm with edge cost 1, terminating the search after all nodes within the hop limit are visited.

Also, with any given value of α, we do not necessarily reach a node in as few hops as possible. We can determine whether there is still a chance of reaching the target node within the hop limit in the following manner. Calculate a "reverse" shortest path tree using edge cost 1, starting at the goal node and following edges backwards, again only considering nodes that are within the hop limit. This yields the *minimum target distance* t_n for each node n, the minimum number of hops required to reach the target from that node. Then, when creating an MLMC-tree and expanding a node n whose depth for the current α is q_n, avoid adding to the priority queue any children n' for which $q_n + 1 + t_{n'}$ is greater than the hop limit. This technique can be also be applied to the optimized generation of the MLMC-tree discussed above.

Generalization to Two Arbitrary Quality Measures. In our problem, there are two conflicting criteria: The quality of a relay chain, modelled by surveillance and communication cost functions, and the number of UAVs required. Thus, we have a bi-objective optimization problem where one objective function can be specified freely while the other is fixed. Let us here mention the possibility of generalizing this problem to allow both objective functions to be specified freely, where we for example wish to explore the trade-off between transmission quality and tolerance to drift.

Algorithm 4 is amenable to being adapted to the generalized problem. For this purpose we introduce an additional edge cost $d_{n,n'}$ corresponding to the second objective being constrained, analogous to the previously defined edge cost $c_{n,n'}$. An upper limit K for the corresponding path cost replaces the hop limit $n_{uavs}+1$. The original **STR-MinCostLimited** problem then corresponds directly to the special case where $d_{n,n'} = 1$ for all edges (n, n'), and the remaining steps of the generalization consist mainly of replacing the implicit use of this value in various parts of the algorithm with the explicit use of a cost that varies for each edge.

First, MLMC-trees and MLMC-paths are modified to use the additional edge

costs instead of hop counts as the secondary objective. In other words, each edge is given the cost $\langle c, d \rangle$ rather than $\langle c, 1 \rangle$, and generating an MLMC-tree results in paths having the lowest "d-cost" among those that have the lowest "c-cost".

Given an MLMC-tree, each node n has a unique path $\pi(n)$ to the root. The node can then be associated with an additional path cost $z_n = \sum_{(n,n') \in \pi(n)} d_{n,n'}$ required for reaching the node, replacing the node depth q_n and analogous to the previously defined path cost $y_n = \sum_{(n,n') \in \pi(n)} c_{n,n'}$.

The MLMC-tree is calculated using the modified edge costs $c'_{n,n'} = c_{n,n'} + \alpha \cdot d_{n,n'}$ instead of $c'_{n,n'} = c_{n,n'} + \alpha \cdot 1$, and the edges that may be used to make paths less expensive in terms of the second objective are calculated as $S = \{(n, n') \in E : z_{n'} > z_n + d_{n,n'}\}$. Finally, $\epsilon_{n,n'} = \frac{(y_n + c'_{n,n'}) - y_{n'}}{z_{n'} - (z_n + d_{n,n'})}$.

The problem now considered is in its general form the Constrained Shortest Path problem (see for example Beasley and Christofides (1989); Dumitrescu and Boland (2003); Carlyle et al. (2008); Muhandiramge and Boland (2009)). This problem is often attacked by Lagrangean relaxation, based on the same Lagrangean dual as the dual ascent method, but with other dual search techniques.

6 Empirical Testing

We will now present the results of empirical testing.

Implementation and Test Systems. All algorithms have been implemented in C++ as part of a modular relay placement service in the UASTech distributed UAV architecture developed at Linköping University (Doherty et al., 2004). This architecture is used in our groundstations as well as on board our larger UAVs, including two slightly modified Yamaha RMAX helicopters. All relay placement algorithms can be executed on board these UAVs. However, our main focus is on the case where a ground operator requests relay chains and approves one of the presented alternatives. This part of the mission setup is better performed on a ground station, which is likely to be more powerful. The ground station then uses the UASTech delegation framework to delegate individual relay or surveillance tasks to available UAVs, which use on-board path planners and related services to execute their part of the mission. As the placement algorithms are more likely to run on a ground station, timings presented below have been generated on a standard PC with a 2.4 GHz Core 2 Duo processor and 2 GB of memory, corresponding to a ground station computer.

Reachability and Cost Functions. A surveillance UAV sending a video feed in real time cannot necessarily re-transmit corrupted or lost packets. Instead, one is likely to use error-correcting codes to correct some but not all errors. The risk of uncorrectable errors increases as the signal-to-noise ratio decreases, leading to a quality loss that can be modeled as a communication cost. Given detailed knowledge about the environment, a cost function can be derived using explicit models of signal propagation, taking into account effects such as reflection and signal absorption. More commonly, costs are calculated in terms

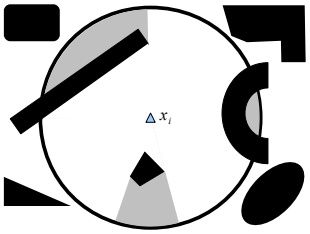


Figure 8: A node and the associated communication radius, visualized in 2D for simplicity. The obstructed volume consists of the black obstacles inside the circle and the shaded volume hidden behind obstacles.

of distance raised to some suitable power (Rappaport, 2002), possibly setting the cost to a constant below a certain distance threshold where transmission errors are more likely to occur for other reasons than distance-related loss of signal strength. This is quite suitable for the purpose of testing the performance of our algorithms. Our first cost function is therefore based on distance, with a constant cost of 300 up to 60 meters, corresponding to the assumption that communication within this distance will be of comparatively constant, but not perfect, quality. After 60 meters, the cost increases with the square of the distance. This tests the case where a wide variety of Pareto-optimal relay chains is generated for any target.

UAVs may be able to relay multiple streams of information, particularly when some streams have lower bandwidth requirements than high-resolution video. We can then benefit from placing relays where they are more likely to be useful for missions initiated in the future. For example, a UAV can communicate with other UAVs located in a sphere whose radius is the maximum communication distance r_{comm}. Given line-of-sight requirements, parts of this volume may be obstructed by obstacles (Figure 8). The greater the obstructed volume is, the smaller the chance that future chains can connect through this relay position. Our second communication cost function $c_{comm}(x, x')$ is therefore proportional to the obstructed volume within a sphere of radius r_{comm} centered at x'. This generally results in considerably fewer Pareto-optimal chains, testing the performance of the algorithms for this end of the spectrum as well.

Finally, our reachability functions were based on free line of sight, with a communication and surveillance radius of 100 meters.

6.1 Generating Pareto-optimal Paths

Empirical performance testing for Pareto-optimal path generation has taken place in three environments, all with a size of 1000 times 1000 meters and a height of 80 meters. The environment graph was constructed using a three-dimensional grid and the grid cell size was varied between 10 and 40 meters. The largest cell sizes corresponds to 33% and 40% of the communication range,

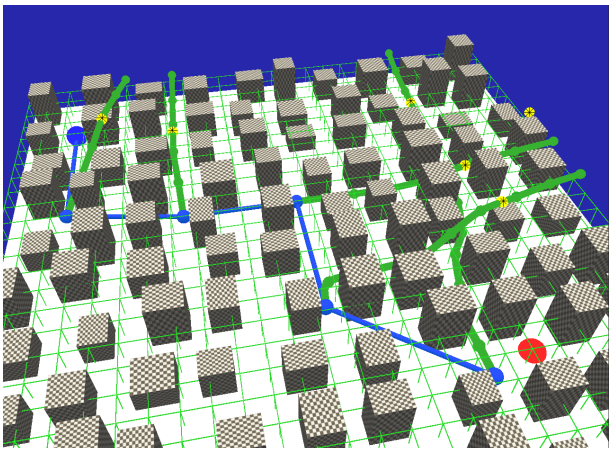


Figure 9: Randomized urban environment.

which we would not necessarily expect to use in practice. We include the results for these discretizations mainly to provide an extended baseline against which the performance of the algorithms can be compared.

The first environment used in testing is an urban environment with semi-random placement of 100 tall buildings, as shown in Figure 9. To reduce clutter, the figure is based on a sparse discretization and shows only the "lowest" level of grid cells. In this figure, we also visualize one specific relay chain generated by Algorithm 2. The base station is in the upper left corner and is connected by dark lines representing communication links to dark spheres denoting intended positions for relay and surveillance UAVs. The target is in the lower right corner and is visible from the last UAV in the chain. Subtasks have been delegated to several other UAVs, which are simulated in anticipation of a sufficient number of hardware platforms being made ready. In the figure, the simulated UAVs have used their path planners to generate individual flight paths (indicated by lines originating at the edges of the map) and are in the process of flying to their intended positions.

The second environment places buildings more randomly, but also reserves space for two wide boulevards where UAVs can fly more freely, intersecting in the center of the environment. Finally, the third environment consists of a 3D model of an emergency services training ground in Revinge in southern Sweden (Figure 10), one of the areas where we regularly perform test flights.

For each of the environments and for each discretization used, we have randomly generated 100 test cases differing in the position of the base station and the target. Table 3 shows the cell size used for each discretization. For the current environments, decreasing cell sizes in the two horizontal directions has a greater impact on the possibility of finding good paths, which is reflected in the selected cell sizes.

The table also shows the average number of nodes and edges for each case,

Figure 10: The Revinge emergency services training ground.

together with the minimum, average and maximum node degree (the number of edges connected to each node). As can be seen, introducing boulevards does not significantly change the number of nodes. However, the somewhat larger open areas do result in a greater number of nodes of high degree, noticeably increasing the total number of edges. The Revinge environment is even more open, with a corresponding increase in both nodes and edges.

The final two columns of the table show the value of $k^*_{\max}$ for each of the two cost functions, also averaged over the 100 random test cases. Recall that $k^*_{\max}$ corresponds to the maximum length of any Pareto-optimal path from the selected base station position. This value is generally greater for the distance-based cost function, where there are greater opportunities to decrease costs by using paths consisting of shorter edges, which requires a larger number of UAVs. It is also clear that using the most coarse-grained discretizations yields noticeably larger values of $k^*_{\max}$ for the distance-based cost function. This function encourages edges of length close to 60 meters, as longer edges may be considerably more expensive. With longer distances between two nodes, finding such edges becomes more difficult, and shorter edges tend to be preferred when generating the least expensive path.

In Figure 11, we compare the time requirements of Algorithm 1 against Algorithm 2 using the distance-based cost measure. To ensure that all Pareto-optimal chains were found, we used $n_{uavs} > k^*_{\max} - 1$. As above, each result is averaged over the 100 randomized instances, and standard deviations are indicated using error bars. For each of the three environments and for each discretization, the new algorithm outperforms Algorithm 1 by a factor of 7–10 for the two coarsest discretizations (33 and 40 meters), and a factor of 12–16 for the remaining discretizations.

Figure 12 shows the corresponding results for the cost function based on obstructed volume. Due to the nature of this function, there tend to be fewer opportunities to improve existing paths in each iteration. The new algorithm is able to take advantage of this fact to a greater degree than Algorithm 1, resulting in a speedup by approximately a factor of 45 to 75.

World	Cell size (meters)	$\|N\|$	$\|E\|$	Node degree Min/avg/max	$k^*_{\max}$ Vol.	Dist.
Randomized urban	40x40x40	1,022	21,815	5/21/35	14.1	23.6
	33x33x40	1,791	53,767	6/30 /47	14.9	24.1
	25x25x25	3,796	257,711	11/67/113	13.9	20.1
	20x20x20	7,846	1,102,063	21/140/221	13.5	19.0
	15x15x20	13,852	3,741,793	30/270/434	12.4	17.7
	12x12x20	22,008	9,286,019	40/421/680	12.5	18.1
	10x10x20	31,807	18,753,049	61/589/948	12.3	17.8
Urban with boulevards	40x40x40	989	27,455	1/27/42	12.7	22.1
	33x33x40	1,781	62,547	1/35/50	12.6	23.2
	25x25x25	3,775	339,148	2/89/135	12.0	18.4
	20x20x20	7,941	1,456,997	7/183/278	13.3	17.6
	15x15x20	13,813	4,850,503	11/351/531	12.8	16.9
	12x12x20	21,885	11,970,877	11/546/831	14.4	18.1
	10x10x20	31,965	24,452,326	15/764/1154	12.7	17.4
Revinge	40x40x40	1,170	38,903	1/33/42	13.9	23.0
	33x33x40	2,049	86,351	1/42/51	13.9	24.0
	25x25x25	4,480	479,167	2/106/137	13.7	20.0
	20x20x20	9,341	2,047,203	7/219/280	13.4	18.5
	15x15x20	16,273	6,784,873	11/416/532	12.3	17.3
	12x12x20	25,663	16,762,197	1/653/832	12.4	18.0
	10x10x20	37,307	33,976,220	1/910/1157	12.3	17.5

Table 3: Information about worlds and discretizations in empirical testing.

6.2 Generating Cheapest Paths with Length Bounds

Though the label-correcting algorithm is very efficient, the dual ascent algorithm provides better performance for **STR-MinCostLimited** in environments where obstacles are dense. We have therefore compared the two algorithms for a randomized urban environment that is very similar to the one used previously, but with a larger number of smaller buildings. To test the more difficult cases where many different Pareto-optimal relay chains exist, we used only a distance-based cost function.

We randomly generated 100 test cases with different base station and target positions. Since the length of the cheapest and longest Pareto-optimal relay chain varied, we use relative hop limits to ensure that results are comparable across all test cases. For example, given that the cheapest relay chain has a length of 14, the time reported for the relative hop limit of -3 for this chain corresponds to the time required to find a hop-constrained relay chain of length at most $14 - 3 = 11$.

Figure 13 shows the result for a grid cell size of 20 meters. Let us note that the irregular shape of the curves is due to the the fact that not all test cases are solvable for all possible relative hop limits: The results for relative length -10

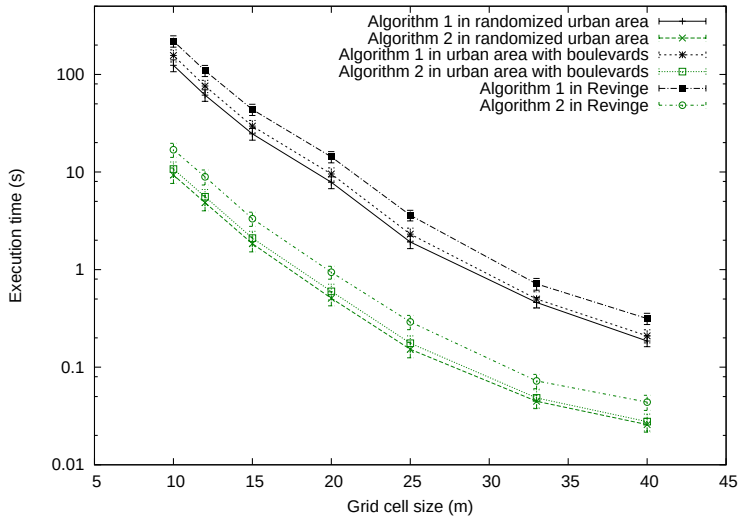


Figure 11: **STR-ParetoLimited** timing results using distance-based costs.

are averaged over considerably fewer (and on average more difficult) test cases than the results for relative length 0.

At the rightmost part of the curves, we can see a general pattern caused by the structure of the algorithms: After generating the cheapest and longest path, Algorithm 2 generates all paths in order of increasing length. Therefore, for any given test case, the maximum of the curve is at relative length -1. For the dual ascent algorithm, the longest paths can generally be created quickly, as they require few iterations. The shortest paths can also often be created more quickly, as the associated hop limits are quite strict and allow us to constrain the search space more effectively. This leads to a comparatively greater performance advantage at the ends of the spectrum.

7 Related Work

A variety of problems related to the multiple relay placement problem defined above have been discussed in the literature.

The problem of controlling teams of communicating unmanned ground vehicles has been investigated by several researchers (Sweeney et al., 2002; Nguyen et al., 2003). The goal is to position UGVs to create a relay chain between a base station and a known target, commonly with the requirement of line of sight between robots to assure sufficient signal strength. Various strategies for UGV movement and placement are evaluated, where a lead UGV advances from the

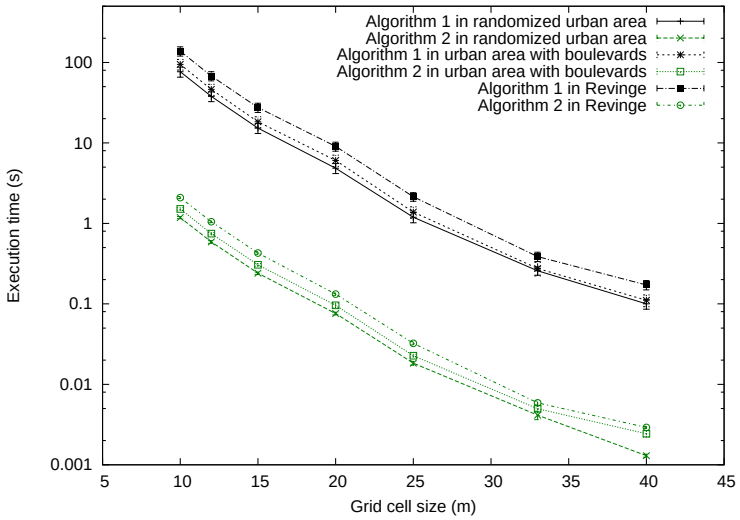


Figure 12: **STR-ParetoLimited** timing results using costs based on obstructed volume.

base station towards the goal position and incrementally determines where to place relay UGVs along the way. Arkin and Diaz (2002) use similar algorithms to find a relay chain between a base station and a target located at an unknown position. However, these algorithms solve a different problem than the one we are interested in: As no a priori calculation or evaluation of paths is performed, it is not certain that a relay chain is found if one exists, or that a chain has maximum quality for the number of relays used.

Centralized optimization of an explicit objective function has been applied to exploration of an indoor area (Rooker and Birk, 2007). The search space is discretized both spatially and temporally, and each discrete location is assigned a utility based e.g. on whether it is previously unexplored and whether it is possible to communicate with team members and with the base station. In each time step, an optimization is performed to determine which actions yield the highest total utility for the team. If the number of targets is greater than the number of robots available, the robots must move between targets while at the same time maintaining communication with a base station (Mosteo et al., 2008). One possibility to create a patrol route is to calculate a tree rooted in the base station, spanning all targets. Several different tree types are possible, such as depth-limited or minimal spanning trees, or trees based on a travelling salesman tour. The trees are evaluated with respect to different criteria, e.g. average travel distance. As we are interested in finding multiple relay chains to a single target

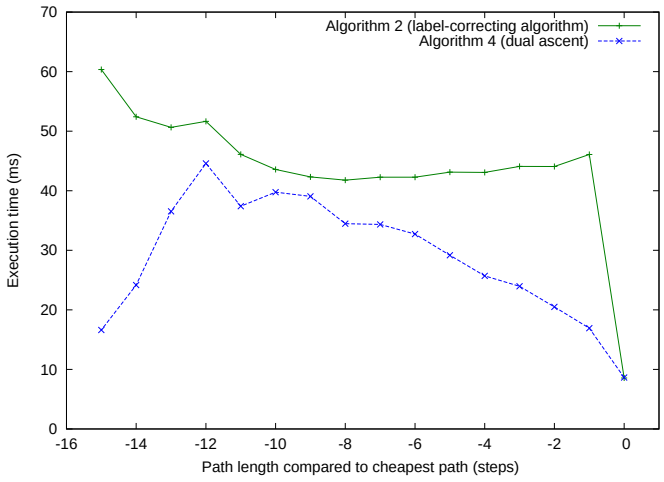


Figure 13: **STR-MinCostLimited** timing results for dense urban environments.

at a known location, and guaranteeing that a solution is found if one exists, there is a significant difference in the problems we aim to solve.

Beard and McLain (2003) instead plan paths for one team member at a time, in a sequential decentralized manner. Any previous plans are taken as input when the next team member plans its trajectory. This is applied in the context of motion planning for several UAVs which must sense as many targets as possible while at the same time avoiding threats and communicating with team members. A dynamic programming approach is used, which is polynomial in the number of nodes in the network but exponential in the number of lookahead steps. Spanos and Murray (2005) investigate how robots can maintain communication while moving from one configuration to another. A distributed algorithm is presented. It is shown that while certain reconfigurations are possible using the presented algorithm, other require global decision making. While Spanos and Murray view maintaining connectivity as a constraint, others see it as a utility that can be optimized in the case of sparse connectivity (Fridman et al., 2007). When a group of agents must move between two positions and the current motion plan does not take communication constraints into consideration, small changes can be made to the trajectory, thereby improving the connectivity of the network. Both cooperative and non-cooperative algorithms are presented and evaluated. As would be expected, the cooperative algorithm gives a slightly greater improvement in network connectivity. These motion planning problems are of limited relevance to the *placement* problems that we are interested in.

The concept of using a UAV as a communication relay, including intended

platforms and communications equipment, is discussed in Pinkney et al. (1996), but no algorithms are presented. In limited cases, where a single relay UAV is sufficient, the surveillance UAV could plan and fly a trajectory to the goal while the relay UAV continuously attempts to maintain line of sight to both the surveillance UAV and the base station (Schouwenaars, 2006). The benefit of using a single relay UAV in an urban environment has also been simulated (Cerasoli, 2007). Here the UAV works as a relay between two entities on the ground. The focus is to determine the percentage of an urban area with acceptable coverage for UAVs positioned at different heights. Only a single relay is used and no algorithm for positioning the UAV is provided.

Palat et al. (2005) have investigated using a swarm of UAVs to improve the range and reliability of an ad-hoc network. A significant increase in range is achieved compared to using a direct ground link with the same transmission power. Message routing in ad-hoc networks may superficially seem similar to the multiple relay positioning problem. However, in ad-hoc networks there is little or no control of the network topology, and it is essential for routing algorithms to handle addition and removal of nodes at runtime (Johnson and Maltz, 1996; Li and Rus, 2000; Mauve et al., 2001), which is not the case for us. Additionally, we are only interested in transmitting information between the surveillance UAV and the base station, as opposed to between arbitrary nodes in a large network.

Wireless sensor networks (WSNs) consist of a large number of small sensors that are placed to cover an area (Akyildiz et al., 2002). Routing in WSNs has attracted quite a bit of research recently, due to e.g. power constraints (Al-Karaki and Kamal, 2004). Although there are some similarities with our problem, there are also considerable differences: WSNs must be able to handle frequent sensor failures, and relays are often also sensors and should be placed accordingly. Simonetto et al. (2008) investigate another case, where a set of mobile sensors explore a variety of indoor environments and a set of mobile robots provide assistance in maintaining communication with a stationary base station. Several reactive and pro-active algorithms based on the use of dynamic potential fields are evaluated. This differs from our problem in that our target is stationary and in that we prefer to pre-calculate a set of alternative solutions that guarantee connectivity.

8 Conclusions and Future Work

The use of relay chains is essential to a large variety of UAV and UGV applications where communication range is limited, including but not limited to surveillance tasks. We have presented two new algorithms for two variations of the static target relay positioning problem. Both algorithms build on a discretization of the continuous problem, and produce relay chains representing distinct trade-offs between the number of UAVs used and the quality of the chain in terms of a user-specified quality measure.

The first algorithm uses label-correcting graph search, building on a preprocessing phase to efficiently generate a complete set of Pareto-optimal relay

chains, allowing the ground operator to select a chain representing a suitable trade-off. The algorithm is demonstrated to be significantly faster than standard algorithms on a variety of problems, which can also enable the use of finer-grained discretizations.

The second algorithm uses a dual ascent technique to efficiently generate a high-quality relay chain given an upper limit on the number of UAVs available, and tends to provide even higher performance in obstacle-dense environments.

We have also presented extensions to these algorithms, allowing relay chains to be generated from multiple base stations, from pre-placed relay UAVs, and to multiple potential targets. Finally, we have discussed a means of extending the second algorithm to produce chains representing trade-offs between two arbitrary quality measures.

In our current work, we make use of regular grids for node placement in the discretized environment. In the future, we intend to investigate the impact of using a variety of other strategies, replacing or augmenting the grid approach. We are also working on generation of relay trees for surveillance of multiple static targets, as well as on methods for surveilling moving targets.

Funding

This work has been supported by LinkLab (`www.linklab.se`); the ELLIIT network organization for Information and Communication Technology; the Swedish Foundation for Strategic Research (SSF) Strategic Research Center MOVIII; the Center for Industrial Information Technology CENIIT [grant number 06.09]; and the Linnaeus Center for Control, Autonomy, Decision-making in Complex Systems (CADICS), funded by the Swedish Research Council (VR).

References

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422.

Al-Karaki, J. N. and Kamal, A. E. (2004). Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28.

Amato, N. M., Bayazit, O. B., Dale, L. K., Jones, C., and Vallejo, D. (1998). OBPRM: An obstacle-based PRM for 3D workspaces. In Agarwal, P. K., Kavraki, L. E., and Mason, M. T., editors, *Robotics: The Algorithmic Perspective: 1998 Workshop on the Algorithmic Foundations of Robotics*, pages 155–168. A.K. Peters.

Arkin, R. C. and Diaz, J. (2002). Line-of-sight constrained exploration for reactive multiagent robotic teams. In *AMC 7th International Workshop on Advanced Motion Control*, pages 455–461.

Balakrishnan, A. and Altinkemer, K. (1992). Using a hop-constrained model to generate alternative communication network design. *ORSA Journal on Computing*, 4(2):192–205.

Balakrishnan, A., Magnanti, T. L., and Wong, R. T. (1989). A dual-ascent procedure for large-scale uncapacitated network design. *Operations Research*, 37(5):716–740.

Barnhart, C. (1993). Dual-ascent methods for large-scale multicommodity flow problems. *Naval Research Logistics*, 40(3):305–324.

Beard, R. W. and McLain, T. W. (2003). Multiple UAV cooperative search under collision avoidance and limited range communication constraints. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, volume 1, pages 25–30. IEEE.

Beasley, J. and Christofides, N. (1989). An algorithm for the resource constrained shortest path problem. *Networks*, 19(4):379–394.

Boor, V., Overmars, M. H., and van der Stappen, A. F. (1999). Gaussian sampling for probabilistic roadmap planners. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1018–1023. IEEE.

Burdakov, O., Doherty, P., Holmberg, K., Kvarnström, J., and Olsson, P.-M. (2009a). Positioning unmanned aerial vehicles as communication relays for surveillance tasks. In *Proceedings of the 5th Robotics: Science and Systems Conference (RSS)*, Seattle, Washington.

Burdakov, O., Doherty, P., Holmberg, K., and Olsson, P.-M. (2009b). Optimal placement of UV-based communications relay nodes. Technical Report LiTH-MAT-R-2009-03, Linköping University, Department of Mathematics. Accepted for publication in *Journal of Global Optimization*. DOI: 10.1007/s10898-010-9526-8.

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

Carlyle, W. M., Royset, J. O., and Wood, R. K. (2008). Lagrangian relaxation and enumeration for solving constrained shortest-path problems. *Networks*, 52(4):256–270.

Cerasoli, C. (2007). An analysis of unmanned airborne vehicle relay coverage in urban environments. In *Proceedings of the Military Communications Conference (MILCOM)*, pages 1–7. IEEE.

Dahl, G. and Gouveia, L. (2004). On the directed hop-constrained shortest path problem. *Operations Research Letters*, 32(1):15–22.

Doherty, P., Haslum, P., Heintz, F., Merz, T., Nyblom, P., Persson, T., and Wingman, B. (2004). A distributed architecture for autonomous unmanned aerial vehicle experimentation. In *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, pages 221–230, Toulouse, France.

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

Duranti, S., Conte, G., Lundström, D., Rudol, P., Wzorek, M., and Doherty, P. (2007). LinkMAV, a prototype rotary wing micro aerial vehicle. In *Proceedings of the 17th IFAC Symposium on Automatic Control in Aerospace*.

Erlenkotter, D. (1978). A dual-based procedure for uncapacitated facility location. *Operations Research*, 26:992–1009.

Fridman, A., Modi, J., Weber, S., and Kam, M. (2007). Communication-based motion planning. In *Proceedings of the 41st Annual Conference on Information Sciences and Systems*, pages 382–387. IEEE.

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

Holmberg, K. and Jörnsten, K. (1996). A dual ascent procedure for the exact formulation of the simple plant problem with spatial interaction. *Optimization*, 36:139–152.

Hsu, D., Jiang, T., Reif, J., and Sun, Z. (2003). The bridge test for sampling narrow passages with probabilistic roadmap planners. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 4420–4426.

Johnson, D. B. and Maltz, D. A. (1996). *Dynamic Source Routing in Ad Hoc Wireless Networks*, volume 353 of *The Kluwer International Series in Engineering and Computer Science. Mobile Computing*, chapter 5, pages 153–181. Kluwer Academic Publishers, Norwell, MA.

Lawler, E. L. (1976). *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York City, NY.

Li, Q. and Rus, D. (2000). Sending messages to mobile users in disconnected ad-hoc wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 44–55. ACM.

Mauve, M., Widmer, J., and Hartenstein, H. (2001). A survey on position-based routing in mobile ad hoc networks. *IEEE Network*, 15(6):30–39.

Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers.

Mosteo, A. R., Montano, L., and Lagoudakis, M. G. (2008). Guaranteed-performance multi-robot routing under limited communication range. In *Proceedings of the 9th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, pages 491–502. Springer Berlin Heidelberg.

Muhandiramge, R. and Boland, N. (2009). Simultaneous solution of lagrangean dual problems interleaved with preprocessing for the weight constrained shortest path problem. *Networks*, 53(4):358–381.

Nguyen, H. G., Pezeshkian, N., Raymond, M., Gupta, A., and Spector, J. M. (2003). Autonomous communication relays for tactical robots. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR)*.

Palat, R., Annamalai, A., and Reed, J. (2005). Cooperative relaying for ad-hoc ground networks using swarm UAVs. In *Proceedings of the Military Communications Conference (MILCOM)*, volume 3, pages 1588–1594. IEEE.

Pinkney, F. J., Hampel, D., and DiPierro, S. (1996). Unmanned aerial vehicle (UAV) communications relay. In *Proceedings of the Military Communications Conference (MILCOM)*, volume 1, pages 47–51. IEEE.

Rappaport, T. S. (2002). *Wireless Communications: Principles and Practice*. Prentice Hall, Upper Saddle River, NJ.

Rooker, M. N. and Birk, A. (2007). Multi robot communication under the constraints of wireless networking. *Engineering Control Practice*, 15(4):435–445.

Schouwenaars, T. (2006). *Safe Trajectory Planning of Autonomous Vehicles*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics.

Simonetto, A., Scerri, P., and Sycara, K. (2008). A mobile network for mobile sensors. In *Proceedings of the 11th International Conference on Information Fusion (Fusion)*, pages 1–8. IEEE.

Spanos, D. P. and Murray, R. M. (2005). Motion planning with wireless network constraints. In *Proceedings of the American Control Conference*, pages 87–92. IEEE.

Sweeney, J., Brunette, T., Yang, Y., and Grupen, R. (2002). Coordinated teams of reactive mobile platforms. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 299–304. IEEE.

Wilmarth, S. A., Amato, N. M., and Stiller, P. F. (1999). MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1024–1031. IEEE.