Generating UAV Communication Networks for Monitoring and Surveillance

Per-Magnus Olsson, Jonas Kvarnström, Patrick Doherty
Department of Computer and Information Science
Linköping University
SE-581 83 Linköping, Sweden
{perol, jonkv, patdo}@ida.liu.se

Oleg Burdakov, Kaj Holmberg
Department of Mathematics
Linköping University
SE-581 83 Linköping, Sweden
{olbur, kahol}@mai.liu.se

***Abstract*—An important use of unmanned aerial vehicles is surveillance of distant targets, where sensor information must quickly be transmitted back to a base station. In many cases, high uninterrupted bandwidth requires line-of-sight between sender and transmitter to minimize quality degradation. Communication range is typically limited, especially when smaller UAVs are used. Both problems can be solved to creating relay chains for surveillance of a single target, and relay trees for simultaneous surveillance of multiple targets. In this paper we show how such chains and trees can be calculated. For relay chains we create a set of chains offering different trade-offs between the number of UAVs in the chain and the chain's cost. We also show new results on how relay trees can be quickly calculated and then incrementally improved if necessary. Encouraging empirical results for improvement of relay trees are presented.**

***Index Terms*—Unmanned aerial vehicles, UAV surveillance, relay, communication, tree optimization**

I. Introduction

Many applications for unmanned aerial vehicles (UAVs) include the need for surveillance of distant targets. Examples of such activities include search and rescue operations, traffic surveillance and forest fire monitoring as well as law enforcement and military applications. Often the information gathered must be transmitted continuously from a *surveillance UAV* to a base station where the current operation is being coordinated. As this information may include urgent high-volume sensor data such as live video, high uninterrupted communications bandwidth is often required. To minimize quality degradation, UAV applications therefore tend to require line-of-sight (LOS) communications, which is problematic in urban or mountainous areas. The maximum communication range is typically also limited, especially when smaller and lower-cost UAVs are used.

The problem of achieving line-of-sight can in some cases be alleviated by increasing altitude. However, this also requires greater communication ranges, and the airspace at higher altitudes may not be permitted by aviation regulations. Smaller UAVs may also be unable to ascend to sufficient altitude to achieve line-of-sight to both the target and the base station.

Both intervening obstacles and limited range can be handled using a chain of one or more cooperating UAVs acting as *relays*, passing on information flowing to the base station (Figure 1). To realize such a *relay chain*, the UAVs should be positioned in a way that not only allows an uninterrupted flow of information, but also optimizes the quality of the chain. The relay chain is a special case of a *relay tree*, which occurs in simultaneous surveillance of several targets. A relay tree places even higher requirements on cooperation as some UAVs will receive information from several sources.

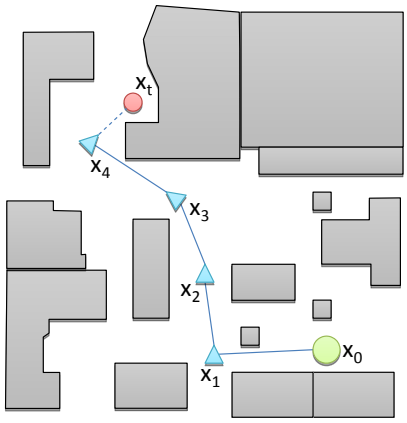

Fig. 1. The relay UAVs at positions x_1, x_2 and x_3 are connecting the base station at x_0 with the surveillance UAV at x_4, surveilling the target at x_t. This is a relay chain of length 5.

Generation of relay chains and trees plays an important part in a larger system involving both human operators and UAVs: as soon as information about a set of surveillance targets is received and a map of the environment is available, a surveillance mission is planned. A broadcast is performed with the intention of finding a set of UAVs (Figure 2) that are available during the required time. The next step is to determine the positions of the UAVs so that all targets are surveilled, given restrictions such as e.g. the number of UAVs and communication constraints. Different options can be explored, such as using the minimum number of UAVs or a larger number to achieve a higher quality relay tree. Once the allocation of UAVs has been determined, each UAV is assigned a position and uses its own path planner to find a flyable trajectory to the destination. As soon as all UAVs have reached their respective positions, the surveillance mission can be initiated.

Algorithms for calculating relay chains and trees should be sufficiently scalable to be able to calculate positions involving a large number of UAVs, to enable the use of relatively inexpensive miniature vehicles with highly limited

Fig. 2. The UASTech Yamaha RMAX helicopter system [1], [2].

Fig. 3. The UAS Tech Lab LinkQuad quadrotor system [3].

communication range, such as the LinkQuad Micro Aerial Vehicle (Figure 3). Furthermore, such calculation must be performed in a timely manner as the ground operator expects a prompt response. In this paper we define relay chains and trees, as well as describe different factors that can be used when evaluating positions for suitability of relay placement. We discuss algorithms for calculating relay chains and trees and present preliminary results for generating such relay trees.

II. Relay Chains and Relay Trees

We will now formally define relay chains and trees as well as give examples of factors that can be used to model whether communication and surveillance can take place.

Assume that M UAVs with identical communication capabilities are available and let $U \subseteq R^3$ be the region where the UAVs 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. Assume that $x_0 \in R^3 \setminus U$ is the position of a base station, and that the positions of the surveillance targets are $T = \{t_1, \ldots, t_m\} \in R^3 \setminus U$.

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. 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 $x' \in R^3 \setminus U$.

These reachability functions are typically based on specific characteristics of the equipment used for communication and surveillance, respectively. For example, $f_{comm}(x, x')$ could hold if we predict that the strength of the signal transmitted from x and received at x' is high enough to allow communication. For the problem at hand, f_{comm} would typically be defined in terms of a limited communication range and a line-of-sight requirement in order to minimize quality degradation. For camera surveillance, a maximum surveillance range and a line-of-sight requirement would typically be used. Other than their signatures, no assumptions about the reachability function are made, and more advanced models of e.g. radio wave propagation can be used if desired.

In addition to reachability functions, we also use two cost functions: the *communication cost function* $c_{comm}(x, x')$ and the *surveillance cost function* $c_{surv}(x, x')$ denote the non-negative costs of communication and surveillance, respectively. These functions are valid only if the corresponding reachability function holds between the positions.

The term cost is used in a very broad sense, and can be used to model arbitrary measures for evaluating the suitability of positions for UAV placement. For example, since signal strength decreases with distance, one can create a cost measure where the cost increases with distance. Assuming line of sight, the signal strength decreases with the square of the distance [4]. To some extent the decrease in signal strength can be offset by using communication devices with an error-correcting capability. However, it is likely that some errors are impossible to correct, and that the risk of such errors increases with lower signal strength. This can be modeled by setting a constant communication cost up to threshold distance, above which the cost increases quadratically.

Likewise, the surveillance cost can be related to the quality of the sensed information. For example, if the image quality decreases with increasing distance, the surveillance cost would likewise increase. In some situations it can be useful to use several factors in the cost function, which we can achieve using for example a weighted sum. A lower cost relay chain or tree has a higher quality.

We can now define a *relay chain* between x_0 and a single target t_1 as a sequence of positions $[x_0, x_1, \ldots, x_k, t_1]$, 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 $f_{surv}(x_k, t_1)$. The *length* of a chain is defined as the number of agents required, including the base station: $len([x_0, x_1, \ldots, x_k, t_1]) = k + 1$. The *cost* of a relay chain is defined as $(\sum_{i=0}^{k-1} c_{comm}(x_i, x_{i+1})) + c_{surv}(x_k, t_1)$.

A *relay tree* between x_0 and the targets $\{t_1, \ldots, t_m\}$ consists of a set of relay chains that together form a tree structure. Note that these chains may share nodes, corresponding to one UAV relaying information from several other UAVs. For each target $t_i \in T$ there exists a chain in the relay tree which starts in x_0 and ends in t_i. Let L be the number of UAVs required to realize the tree and let the non-target positions in the tree be denoted by $[x_0, \ldots, x_L]$. Also, let x^- denote the unique predecessor of position x. Then, the total cost is $\sum_{i=1}^{L} c_{comm}(x_i^-, x_i) + \sum_{i=1}^{m} c_{surv}(t_i^-, t_i)$.

We are interested in generating high quality relay chains

and trees relative to quality measures such as the number of UAVs required and/or the total cost of the relay chain. Problem definitions and algorithms for calculating relay chains and trees are available in Section IV and Section V, respectively.

III. Discretization

Finding relay positions that yield high quality relay chains and trees is difficult since the feasible set typically is disjoint due to obstacles, and the number of subsets may be very large. As each subset has at least one local extreme point, the number of local extrema may be very large. Therefore, methods for continuous optimization are very time-consuming and it is not guaranteed that the global optimum is found. For these reasons we suggest to discretize the environment and solve a discrete approximation of the continuous problem. Once a discrete search space has been created, we apply graph algorithms to calculate the relay chains and trees.

The first step of discretizing an instance of the continuous relay positioning problem consists of selecting a finite set of positions $U' \subseteq U$. These are the positions that will be considered for relay and surveillance UAV placement in the discretized version of the problem. Once this selection has been made, a directed graph can be created as follows.

Associate each position $x \in U' \cup \{x_0, t_1, \ldots, t_m\}$ with a unique node, where n_0 denotes the base station node associated with x_0. For convenience, we use the same symbol, T, for the set of nodes $\{\tau_1, \ldots, \tau_m\}$ associated with the surveillance targets $\{t_1, \ldots, t_m\}$ as for the target positions. Let N be the set of all nodes.

For each $x \in U'$ corresponding to $n \in N$ and satisfying $f_{comm}(x_0, x)$, create an edge $e = (n_0, n)$ of cost $c_{comm}(x_0, x)$ representing the possibility of communication between the base station and position x. For each $x, x' \in U'$ corresponding to $n, n' \in N$ and satisfying $f_{comm}(x, x')$, create a directed edge $e = (n, n')$ of cost $c_{comm}(x, x')$ representing the possibility of communication between positions x and x'.

Finally, for each $\tau_i \in \{\tau_1, \ldots, \tau_m\}$ corresponding to t_i and for each $x \in U'$ corresponding to $n \in N$ and satisfying $f_{surv}(x, t_i)$, create a directed edge $e = (n, \tau_i)$ of cost $c_{surv}(x, t_i)$ representing the fact that a surveillance UAV at x would be able to surveil the target at t_i. Let E be the set of all edges.

Then, $G(N, E)$ is a directed graph corresponding to the original continuous problem instance. Note in particular that the target nodes have no outgoing edges and that all their incoming edges satisfy f_{surv}, ensuring that its predecessor in any path from the base station to the surveillance target must be suitable for a surveillance UAV. Note also that most parts of this graph only depend on the environment and not on the position of the base station or the surveillance targets, and can be precalculated.

Choosing the set U' requires some consideration: nodes must be sufficiently dense so that the set remains connected, given a limited communication range and possibly a line-of-sight restriction. A high node density is required to make good use of the maximum range, especially in the presence of obstacles. The generation of high quality relay chains and trees 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, it is inappropriate to use such algorithms unmodified.

For the type of urban and mountainous terrain we are interested in, a regular 3D grid has proven quite suitable. The grid is placed over the terrain and a graph is constructed from the unobstructed grid cells, where the grid cell size depends on the minimum distance between obstacles. To improve connectivity in certain situations, the grid can be augmented by nodes placed more randomly, e.g. with a preference for placing nodes near obstacles or in narrow passages [5], [6].

IV. Calculating Relay Chains

When we generate relay chains, the number of UAVs in a chain is an obvious measure of the relay chain's quality, but one can also benefit from using other, mission-specific, quality measures, even at the cost of using a larger number of UAVs. As the desired trade-off can be difficult to formalize, we calculate a set of relay chains and leave the final choice to the ground operator. For this reason, we find for each $1 \leq k \leq M$, a *Pareto-optimal* [7] relay chain of length k between x_0 and t_1, or determine that no such chain exists. A chain is Pareto-optimal if it is not possible to improve the chain in one aspect without a decline in another aspect. That is, a longer chain can only be Pareto-optimal if its cost is less than the costs of all shorter chains.

A set of Pareto-optimal relay chains can be calculated using a version of the Bellman-Ford algorithm due to Lawler [8]. The algorithm calculates chains in order of increasing length and decreasing cost to all nodes in the graph: in iteration k it finds chains of length exactly k. However, it can require a substantial amount of time, especially for large problems.

To improve the performance we have developed a complete and optimal algorithm that calculates a set of Pareto-optimal relay chains to each node. It uses a preprocessing step to calculate a tree consisting of paths of minimum length among those of minimum cost, called *minimum length minimum cost* (MLMC) paths, from n_0 to each $n \in N$. Such an *MLMC-tree* can be calculated by Dijkstra's algorithm [9] using *compound costs* of the form $\langle c, l \rangle$ where c is the cost of the path, l is its length, and $\langle c_1, l_1 \rangle < \langle c_2, l_2 \rangle$ iff $(c_1 < c_2)$ or $(c_1 = c_2$ and $l_1 < l_2)$. By definition, no path from n_0 to a node $n \in N$ can be cheaper than the MLMC-path to node n, and for each n the MLMC-path provides a lower bound on the path cost and consequently also an upper bound on the path length. This information allows termination of the calculation of paths individually for each node, which can be used to decrease the execution time. For brevity, we refer to the algorithm as Algorithm 1 and the pseudocode is displayed in Figure 4.

Algorithm 1 creates a set of *reachability records* in each node, which can be represented as a table, as shown in Table I. Each such record $\langle k, g_k, p_k \rangle$ signifies that the node can be reached from the base station in k hops with a cost of g_k using

k (path length)	g_k (cost)	p_k (predecessor)
1	6	n_0
3	3	n_7

TABLE I
EXAMPLE OF REACHABILITY RECORDS STORED IN A NODE AFTER EXECUTION OF ALGORITHM 1.

```
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{M + 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'}
```

Fig. 4. Algorithm 1 – MLMC-tree-based label-correcting algorithm.

the predecessor p_k. A complete path to n_0 can be reconstructed (in reverse order) by considering the reachability records of the predecessor p_k for $k-1$ hops and continuing recursively until n_0 is found. After execution of Algorithm 1, each reachability record corresponds to a Pareto-optimal path. The shortest chain is found on the first row, and each consecutive row corresponds to a cheaper chain, until the cheapest chain is found on the last row. Any "missing" values of k, e.g. $k = 2$ in Table I, means that even if a chain of that length exists, it is not Pareto-optimal. Also, as Table I contains no record for any $k > 3$ means that the cost can not be decreased by using more than 3 UAVs.

The following terminology is used: $c_{n,n'}$ is the cost to go from node n to node n', $g(n)$ is the cost of the cheapest path from n_0 to n found so far. The sets of predecessors and successors of node n are denoted by n_- and n_+, respectively. The height of the MLMC-tree is denoted by $k^*_{max} \leq N$, and the sets $N^*_k, 0 \leq k \leq k^*_{max}$, consists of exactly the nodes occurring at depth k in the MLMC-tree. N_k is a sequence of sets, which are characterized by the fact that any Pareto-optimal chain must consist of a Pareto-optimal chain to a node $n \in N_k$ in $k-1$ hops and a single outgoing edge from n.

The MLMC-tree is calculated on line 0, and k^*_{max} and the N^*_k sets are extracted. Also, initial reachability records are created. Initially, $g(n) = \infty$ for all nodes except n_0 (line 1). As the start node is the only node reachable in zero steps, a reachability record with $g_0(n_0)$ must have been created during preprocessing, and $N_0 = \{n_0\}$ (line 4). No chain to the start node can be cheaper, so all incoming edges to the start node can be removed (lines 2–3).

In lines 5–17, each iteration considers chains of length $k \geq 1$, up to a maximum of $k = 1, \ldots, \min\{M+1, k^*_{max}-1\}$. This is because (i) we allow at most M UAVs, yielding a total chain length of $M+1$ when edges to the base station and target node are added, (ii) no chain longer than k^*_{max} can be of interest as such chains must be at least as expensive as any shorter chain, (iii) any chains of length exactly k^*_{max} were found during the calculation of the MLMC-tree. Setting $M = \infty$ ensures that Pareto-optimal chains of all possible lengths are found.

By the definition of N^*_k, no cheaper chain can be found to any node in N^*_k and the incoming edges to such nodes can be removed without affecting the properties of the algorithm (lines 6–8). However, we need to consider chains going *through* such nodes (line 9), explained further below.

Lines 10–17 considers each outgoing edge of nodes in N_{k-1} and checks whether a cheaper chain has been found. If so, a new reachability record is created and $g(n)$ is set to the cost of the new chain to signify that we are only interested in strictly cheaper chains.

The last thing to do is to prepare for the next iteration by constructing the set N_k. That is, N_k should consist of the nodes to which a cheaper chain was found during the current iteration, or that could not be reached at all with chains of length $k-1$. This is achieved in line 9, for nodes in N^*_k where we found a cheaper chain of length k during preprocessing, and on line 17 for nodes where we found a cheaper chain of length k in this iteration.

The algorithm's time complexity is $O(k^*_{max}|E|) \subseteq O(|N|^3)$ as at most k^*_{max}–1 iterations are performed, each one treating at most $|E|$ edges. However, k^*_{max} is the maximum number of UAVs required to reach any target, and $k^*_{max} \ll N$ typically applies. For more details and proofs, the reader is referred to [10], [11].

V. CALCULATING RELAY TREES

Finding an optimal relay tree is considerably more difficult than finding an optimal relay chain, as finding a relay tree is a variation of the NP-hard *Steiner tree* problem [12]. Therefore, we investigate the problem of finding a feasible relay tree relative to a quality measure such as the number of UAVs or the cost of the tree. The Steiner tree problem in directed graphs is defined as: *Given a network $G = (N, E, c)$ where $c : E \to R^+$ is an edge cost function, a root node $n_0 \in N$ and a non-empty set $T \subseteq N$ of targets, find a tree T_G rooted in n_0 whose leaves are T, such that cost$(T_G) = \sum_{e \in T_G} c(e)$ is minimized*. Let T_p denote the set of *Steiner nodes*, consisting of the set of nodes in T_G that are neither n_0 nor target nodes. Each Steiner node corresponds to a UAV in the relay tree and a tree is *feasible* if $|T_p| \leq M$.

Recall that the directed graph described in Section III does not have any outgoing edges from the target nodes,

1 $T_G \leftarrow n_0$
2 **for** $i = 1, \ldots, |T|$ **do**
3 Calculate the cheapest path q from T_G to $\tau \in T \setminus N(T_G)$
4 $T_G \leftarrow T_G \cup q$ *// Add path to tree*

Fig. 5. Cheapest path heuristic for calculating a relay tree in a directed graph.

thereby making sure that all target nodes are leaves[1]. Even for small problems, the existing algorithms for the directed Steiner tree problem require long execution times [15], [16] or large amounts of memory [17]. Such algorithms are not applicable for calculating relay trees as the graphs used in the relay problems may be large and the algorithms are used in a setting where a ground operator expects a quick response.

Therefore, we adopt another strategy: first we use a heuristic such as the *cheapest path heuristic* [18] to calculate an initial relay tree in a directed graph, and then we incrementally improve the tree through local optimization of subtrees. Below we provide more details about our optimization algorithm, but first we introduce the terminology and describe the cheapest path heuristic.

The relay tree is denoted by T_G, the least cost path from T_G to an unconnected target node is denoted by q and the unconnected target node is denoted by τ. Let the nodes in T_G be denoted by $N(T_G)$.

The cheapest path heuristic (Figure 5) operates iteratively: in each of the $|T|$ iterations, an unconnected target node is connected to an existing relay tree T_G. Initially, this tree only consists of the root node, which corresponds to the base station in our case. In each iteration, the algorithm calculates paths from the nodes in T_G to all unconnected target nodes. From these paths, the cheapest path is selected and added to T_G, thereby connecting a new target as cheaply as possible. Though the solution is only guaranteed to be within a factor $|T|$ from the optimal, the heuristic has in practice proved to be competitive with more advanced methods [19]. The heuristic has a time complexity of $O(|T|(|E| + |N|log|N|))$ as $|T|$ executions of a cheapest path algorithm are performed, each one with a time complexity of $O(|E| + |N|log|N|)$.

After the initial relay tree has been calculated, our new algorithm is used to incrementally improve the tree. Different *optimization criteria* can be used, such as generating the least cost tree, the tree using the least number of UAVs or the least cost tree using at most M UAVs. Compound costs as described in Section IV can also be used. Depending on the current relay tree, different optimization objectives are used. If the current tree is infeasible, it is first optimized with regards to the number of UAVs. Once a feasible tree has been found, the optimization objective is changed to finding the least cost feasible tree. This optimization objective is also used if the initial tree was feasible. The process of continually improving the tree can be performed until no better tree is found or until the available time is out. The algorithm suggested by Chen [20] can also be used to optimize subtrees but is limited to finding the cheapest subtree or the subtree with the fewest steps. Let a subtree of T_G be denoted by T_s, with $r(T_s)$ denoting the root node and $L(T_s)$ denoting the set of leaves of the subtree. A subtree that is a candidate for replacing T_s is denoted by $T_{s'}$.

The algorithm works by performing a sequence of local optimizations of the relay tree. In each optimization, the algorithm chooses a subtree and calculates a set of subtrees as candidates to replace it. From the set of candidate subtrees, the best candidate is chosen and compared to the original subtree. A replacement is performed only if it yields an improvement. Naturally, each candidate subtree $T_{s'}$ has the same root node and the same set of leaves as the subtree it is intended to replace: $r(T_s) = r(T_{s'})$ and $L(T_s) = L(T_{s'})$. The candidate subtrees are calculated through executing Algorithm 1 starting in the subtree root node as well as in each leaf. The nodes must store several sets of reachability records as each execution creates a set of Pareto-optimal chains from the start node to all reachable nodes. Each reachability records stores information about a path, and records for paths with different start nodes are be combined in order to represent a subtree. As target nodes lack outgoing edges and to maintain consistency with how the edges will be used in the final tree, all executions of Algorithm 1 starting in any of the leaf nodes will use the edges in the reverse direction.

When optimizing T_G, only a few nodes are relevant. We refer to the set of relevant nodes as *key nodes*. This set consists of the root node, the targets and the Steiner nodes with outdegree of at least two. A *key path* is a path containing exactly two key nodes: the start node and the end node. To allow for quickly determining the subtrees for optimization, a *reduced tree* is created. The reduced tree is created by replacing each key path by a single edge. Thus the reduced tree consists of only key nodes and maintains the same topology as the relay tree. Determining the nodes in a subtree for optimization becomes a simple matter of selecting a non-target key node and retrieving its predecessor and successors. The order in which subtrees are optimized can be chosen in many different ways, for example starting with the subtrees furthest from the root node and progressing towards the root node.

Figure 6 displays parts of a relay tree and the corresponding reduced tree, where it is clear that the reduced tree only contains key nodes. The subtree T_s with $r(T_s) = n_2$ and $L(T_s) = \{n_{21}, \tau_1, n_{22}\}$, marked by heavy solid edges, is replaced by the tree $T_{s'}$ marked by heavy dashed edges. The reduced tree on the right has the same topology as the relay tree on left, but consists only of key nodes.

The algorithm pseudocode is displayed in Figure 7, and the preference operator $\prec$ is true if the new subtree $T_{s'}$ is preferable to T_s with respect to the optimization criteria. First a subtree is chosen for optimization (line 2) and then Algorithm 1 is executed starting in the root node and once starting in each leaf (lines 3–5). As the new subtree must contain paths to both $r(T_s)$ and all nodes in $L(T_s)$, only nodes reached by all calculations are considered reachable. In each reachable node, reachability records are combined to create

[1]This would not be the case in an undirected graph, and to avoid that targets would potentially be required to relay information, the partial terminal Steiner tree problem [13], [14] would be used to model our problem.

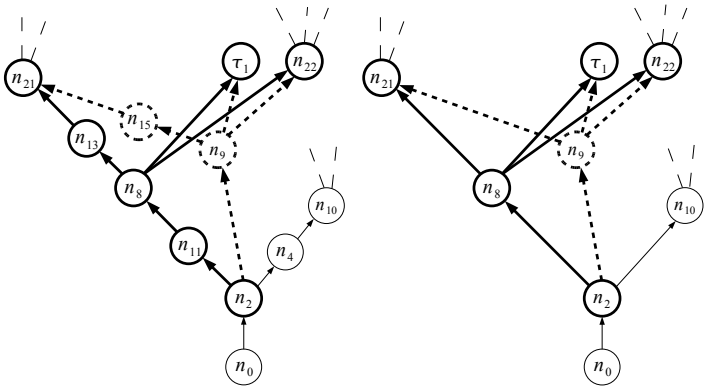

Fig. 6. Relay tree on the left and the corresponding reduced tree on the right. The subtree marked by heavy edges is replaced by the subtree marked by dashed heavy edges.

```
1  while true do
2      T_s ← Choose subtree(T_G)
3      Execute Algorithm 1 starting in r(T_s)
4      for each n ∈ L(T_s) do
5          Execute Algorithm 1 starting in n
6      for each n ∈ N do
7          Determine subtrees(n)
8      T_s' ← Choose best subtree(N)
9      if T_s' ≺ T_s then
10         T_G ← T_G \ T_s          // Remove old subtree
11         T_G ← T_G ∪ T_s'         // Insert new subtree
12         Yield T_G                // Yield improved tree
```

Fig. 7. Algorithm for optimizing existing Steiner trees.

a set of trees (lines 6–7), explained further below. The next step consists of choosing the best subtree according to the optimization criteria (line 8). If the new subtree is better than the existing, the old subtree is removed and the new subtree is inserted (lines 9–11). The improved tree is yielded to the user (line 12).

If a replacement is performed, this may allow further optimization of previously optimized subtrees. As an example, after the replacement depicted in Figure 6, the subtree with root node n_0 and leaves n_9, n_{10} is a candidate for further optimization as the old node n_8 was replaced by n_9. After an optimization has been performed, all subtrees involving the root node and all non-target leaves are candidates for further optimization.

Consider a subtree consisting of a root node and two target nodes. After executing Algorithm 1 once starting in each of the three nodes, the sets of reachability records in Table II exist in some node. By choosing a reachability record from each set and combining them, information about $2 \times 1 \times 3 = 6$ different subtrees are created (Table III). The costs and path lengths are added to get each subtree's corresponding characteristics. Only subtrees for which all aspects are not worse are candidates for replacing the old subtree T_s, i.e. a subtree with a longer total path length is only a candidate if it is cheaper than all trees with a shorter total path length.

k	g_k	p_k
1	6	n_0
3	3	n_{23}

k	g_k	p_k
1	13	τ_1

k	g_k	p_k
2	50	n_{32}
4	32	n_{12}
5	21	n_{28}

TABLE II
EXAMPLE OF REACHABILITY RECORDS FOR EXECUTIONS STARTING IN NODES n_0, τ_1 AND τ_2, RESPECTIVELY.

Path length to n_0	Path length to τ_1	Path length to τ_2	Total path length	Total cost
1	1	2	4	69
1	1	4	6	51
1	1	5	7	40
3	1	2	6	66
3	1	4	8	48
3	1	5	9	37

TABLE III
INFORMATION ABOUT THE DIFFERENT SUBTREES CREATED FROM THE REACHABILITY RECORDS IN TABLE II.

VI. EMPIRICAL RESULTS

We tested our optimization algorithm in a semi-randomized urban environment of size 1000×1000×80 meters with 100 tall buildings. The graph had close to 14,000 nodes and four million edges. The reachability functions were based on line-of-sight, with a communication and surveillance range of 100 m. The cost function was based on distance, with a constant cost up to 60 meters, after which the cost increased quadratically. For testing, 100 combinations of base station position and target positions were randomly generated. In this testing, we used 9 targets, clustered with three targets in each cluster. Testing has been performed on a standard PC with a 2.4GHz CPU and 2 GB RAM, but the algorithms can and have been executed using the on-board computers of the UASTech Yamaha RMAX helicopter (Figure 2), which uses the same software architecture as used in the testing [2].

For the test results presented in Figures 8 – 9, we used compound costs (Section IV) in the optimization objective: the first priority was to decrease the number of nodes in the tree. If the number of nodes in several trees are the same, the tree with the least total tree edge cost was chosen. Here $M = \infty$. In Figure 8, the x-axis displays the improvement in the number of nodes in the tree, and the bars show the number of test cases that attained at least that improvement, e.g. 30 test cases attained an improvement of at least 12%. Of the 100 trees, 87 were improved and the best improvement was 21.4%. The mean improvement was 9.3% and the median was 7.4%. The largest decrease of the total tree edge cost was 63.4%, with a mean improvement of 37.7% and a median of 37.5% (Figure 9).

Figure 10 displays how the number of UAVs and the tree's cost change for a certain test case. First we are trying to find a feasible tree, and once a feasible tree is found, the optimization objective is changed to find the cheapest feasible tree. Here $M = 23$, the black solid curve displays the number of UAVs and the green dashed curve displays the tree cost. The initial tree is infeasible as it uses 26 UAVs,

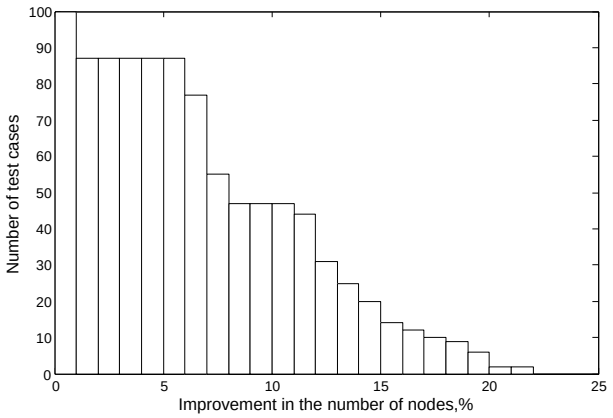

Fig. 8. Number of test cases with a given improvement in the number of nodes.

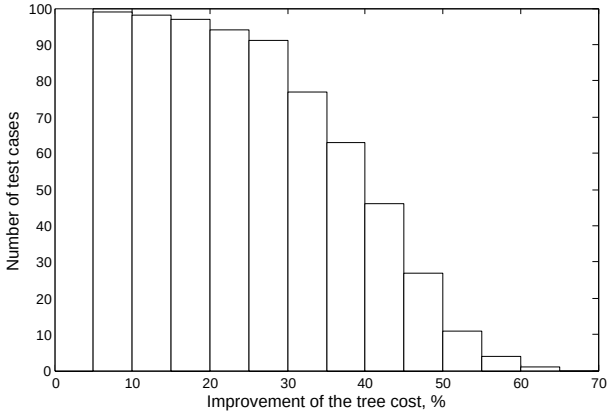

Fig. 9. Number of test cases with a given improvement of the cost of the tree.

and optimization is first performed to find a feasible tree. The second successful optimization finds a feasible tree using 21 UAVs, and the optimization objective is changed to finding the cheapest feasible tree. After that, each new tree decreases the cost, until no lower cost tree is found, after the ninth subtree optimization. In total, the number of UAVs is decreased from 26 to 23 and the cost is only marginally increased. The complete optimization took less than 10 seconds and each improved tree is available to the user as soon as it is calculated.

VII. Related Work

Control behavior for teams of unmanned ground vehicles involving line-of-sight was investigated in Sweeney et al. [21]. In an indoor setting, a lead UGV advances from the base station towards the goal position and incrementally determines where to place relay UGVs along the way in order to maintain communication with the base station. Various strategies are evaluated in terms of time and energy usage. A small survey of positioning algorithms for UGVs is available in Nguyen et al. [22]. The algorithms presented in these articles have several commonalities. No quality or cost measure is used and it is not certain that the goal position will be reached, as no a priori calculation or evaluation of paths is performed.

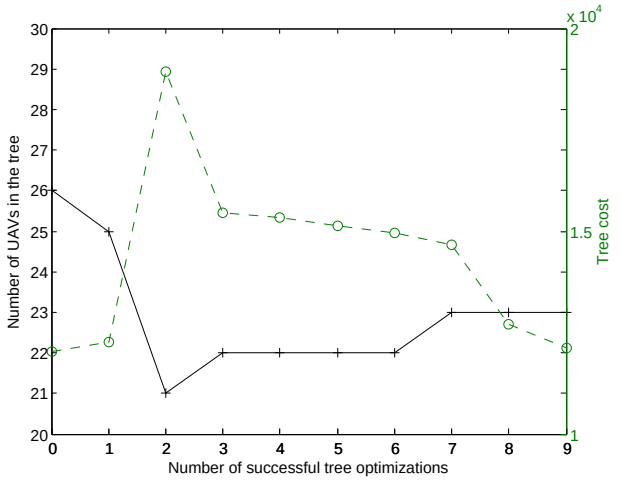

Fig. 10. Minimization of the number of hops is performed until a feasible tree is found, using at most 23 UAVs. Once a feasible tree is found, the optimization objective is changed to finding the cheapest feasible tree.

Arkin and Diaz [23] used a behavior-based architecture to allow teams of ground robots with line-of-sight communication to explore buildings and to find stationary objects, using only limited knowledge about the area in which those objects may be placed.

An algorithm for maintaining LOS between groups of planetary rovers exploring an area is presented by Anderson et al. [24]. The algorithm is based on several heuristics and although testing indicates that the algorithm performs well, it does not guarantee a solution even if one exists. Rooker and Birk present an algorithm for maintaining communication between a group of UGVs exploring an area [25].

The concept of using a UAV as a communication relay, including intended platforms and communications equipment, is discussed in Pinkney et al. [26], but no algorithms are presented. The benefit of using a single relay UAV in an urban environment has also been simulated [27]. Here the UAV works as a relay between two entities on the ground but no algorithm for a priori determining the quality of the UAV's position is provided.

When the number of surveillance targets is greater than the number of UAVs, UGVs must move between targets and surveil them sequentially while maintaining communication with the base station. This can be done by creating a tree rooted in the base station and spanning the targets. Mosteo et al. [28] evaluate several different trees with respect to criteria such as average travel distance for each robot.

Problems that are superficially very similar to the multiple relay positioning problem are encountered in ad-hoc networks, where messages are to be delivered in a network where there is no control of the network topology. Routing algorithms for such networks must be able to handle addition and removal of network nodes at runtime [29]. The range and reliability of ad-hoc networks can be improved by using a UAV as a relay [4]. Wireless Sensor Networks (WSNs) consist of a large number of small sensors that are placed to cover an area [30], [31]. Although there are some similarities with

the problems investigated here, 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. In both ad-hoc networks and WSNs, there is limited control over sensor placement.

VIII. Conclusion

In this paper, we have described algorithms for generating relay chains for surveillance of a single target and relay trees for simultaneous surveillance of multiple targets. We can quickly generate a set of Pareto-optimal chains and let an operator choose between the alternatives. As generation of optimal relay trees is very computationally demanding, we use a heuristic to quickly find an initial tree and apply a new algorithm to incrementally improve the tree. Improvement can be performed with respect to different criteria such as minimization of the number of UAVs or the cost given a limit on the number of UAVs. New empirical results show that our algorithm can substantially improve the quality of trees in a short amount of time. In the near future we will further investigate how we can improve relay trees and generalize our algorithms to solve a variety of problems involving UAV-based monitoring and surveillance.

Acknowledgment

This work has been supported by: LinkLab (http://www. linklab.se); the ELLIIT Excellence Center at Linköping-Lund for Information 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

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

[2] P. Doherty, P. Haslum, F. Heintz, T. Merz, P. Nyblom, T. Persson, and B. Wingman, "A Distributed Architecture for Autonomous Unmanned Aerial Vehicle Experimentation," in *Proceedings of the 7th International Symposium on Distributed Autonomous Systems*, 2004, pp. 221–230.

[3] UAS Technologies Sweden AB, www.uastech.com.

[4] R. Palat, A. Annamalai, and J. Reed, "Cooperative relaying for ad-hoc ground networks using swarm UAVs," in *Proceedings of MILCOM 2006*. IEEE, 2006.

[5] V. Boor, M. H. Overmars, and A. F. van der Stappen, "Gaussian sampling for probabilistic roadmap planners," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2001.

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

[7] K. Miettinen, *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

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

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms, 1st Edition*. MIT Press, 1990.

[10] O. Burdakov, P. Doherty, K. Holmberg, J. Kvarnström, and P.-M. Olsson, "Positioning unmanned aerial vehicles as communication relays for surveillance tasks," *International Journal of Robotics Research. In press. DOI: 10.1177/0278364910369463*, 2010.

[11] O. Burdakov, P. Doherty, K. Holmberg, and P.-M. Olsson, "Optimal placement of UV-based communications relay nodes," *Journal of Global Optimization. In press. DOI: 10.1007/s10898-010-9526-8*, 2009.

[12] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner Tree Problem*. North-Holland, 1992.

[13] S.-Y. Hsieh and H.-M. Gao, "On the partial terminal Steiner tree problem," *Journal of Supercomputing*, vol. 41, no. 1, pp. 41–52, 2007.

[14] S.-Y. Hsieh and W.-H. Pi, "On the partial-terminal Steiner tree problem," in *Proceedings of the International Symposium on Parallel Architectures, Algorithms, and Networks*, 2008, pp. 173–177.

[15] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li, "Approximation algorithms for directed Steiner problems," in *SODA '98: Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, 1998, pp. 192–200.

[16] L. Zosin and S. Khuller, "On directed Steiner trees," in *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, 2002, pp. 59–63.

[17] M.-I. Hsieh, E. H.-K. Wu, and M.-F. Tsai, "FasterDSP: A faster approximation algorithm for directed Steiner tree problem," *Journal of Information Science and Engineering*, vol. 22, pp. 1409–1425, 2006.

[18] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," *Mathematica Japonica*, vol. 24, 1980.

[19] S. Voß, "Worst-case performance of some heuristics for steiner's problem in directed graphs," *Information Processing Letters*, vol. 48, pp. 99–105, 1993.

[20] N.-P. Chen, "New algorithms for the Steiner tree on graphs," in *IEEE Symposium on Circuits and Systems*, 1983, pp. 1217–1219.

[21] J. Sweeney, T. Brunette, Y. Yang, and R. Grupen, "Coordinated teams of reactive mobile platforms," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, 2002.

[22] H. G. Nguyen, N. Pezeshkian, M. Raymond, A. Gupta, and J. M. Spector, "Autonomous communication relays for tactical robots," in *Proceedings of the 11th International Conference on Advanced Robotics*, 2003.

[23] R. C. Arkin and J. Diaz, "Line-of-sight constrained exploration for reactive multiagent robotic teams," in *7th International Workshop on Advanced Motion Control*, 2002.

[24] 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*. IEEE, 2003, pp. 2266–2272.

[25] M. N. Rooker and A. Birk, "Multi robot exploration under the constraints of wireless networking," *Control Engineering Practice*, vol. 15, no. 4, pp. 435–445, 2007.

[26] F. J. Pinkney, D. Hampel, and S. DiPierro, "Unmanned aerial vehicle (UAV) communications relay," in *Proceedings of MILCOM 1996*. IEEE, 1996.

[27] C. Cerasoli, "An analysis of unmanned airborne vehicle relay coverage in urban environments," in *Proceedings of MILCOM 2007*. IEEE, 2007.

[28] A. R. Mosteo, L. Montano, and M. G. Lagoudakis, "Guaranteed-performance multi-robot routing under limited communication range," in *Distributed Autonomous Robotic Systems 8*, H. Asama, H. Kurokawa, J. Ota, and K. Sekiyama, Eds. Springer Berlin Heidelberg, 2009, pp. 491–502.

[29] M. Mauve, J. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *IEEE Network*, vol. 15, no. 6, pp. 30–39, November/December 2001.

[30] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[31] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 6–28, December 2004.