Optimal Search Mission with Unmanned Aerial Vehicles using Mixed Integer Linear Programming

Erik J. Forsmo, Esten I. Grøtli, Thor I. Fossen and Tor A. Johansen

Abstract—This paper proposes the use of Mixed Integer Linear Programming (MILP) for efficient planning of search missions. The Unmanned Aerial Vehicles (UAVs) taking part in a search mission are assumed to be equipped with cameras or other sensors, with a specified field-of-view below the UAV. The design and implementation of this search algorithm have been made for a general case, such that multiple UAVs with arbitrarily geographically located base stations can take part in the joint search mission and be allocated to different parts of the search area. An algorithm for automatically generation of waypoints inside the defined area has been developed, such that the whole area is covered by the sensors field-of-view when all waypoints have been visited.

I. INTRODUCTION

The use of Unmanned Aerial Vehicles (UAVs) is a field with a lot of potentials. In recent years UAVs have been used for both military operations as well as civilian operations. Task assignment or waypoint visiting for multiple UAVs have been considered in several papers, see for instance [1], [2] and [3]. Path planning of UAVs using MILP has been treated in [4], [5], [6] and [7] to mention a few.

Other approaches to design the path should also be investigated if it is important to have a short computation time. In [8] it is suggested to apply an spiral as the path in search missions. In [9] the use of an Archimedes Spiral is further investigated and an algorithm that automatically generates an Archimedes Spiral and fits the size, direction and other parameters to a given search mission, is developed and discussed. In this approach no optimization problem is solved to find the path the UAV should follow. A four-mode search planning approach is suggested in [10].

When more than one UAV is considered in the search mission, mainly two different strategies for coordinating the search mission can be used. They can briefly be described as [11]:

1) Centralized Optimization Problem: This means that one cost function is applied to coordinate the path for the overall system by collecting optimization variables from each vehicle.
2) Decentralized Optimization Problem: One optimization problem for each vehicle (subsystem) is calculated, and there are restrictions and communication to the neighbour subsystems.

Decentralized trajectory optimization for cooperating UAVs using MILP was treated in [12]. In this paper the Centralized Optimization Problem will be applied.

The main contribution in this paper is the design of an algorithm for generation and placement of waypoints at defined coordinates such that a given search area is covered by the UAVs camera system.

This paper proposes Mixed Integer Linear Programming (MILP) for path planning of defined search areas.

II. WAYPOINT GENERATION

The search mission should be done in a systematic way such that the areas are covered by the on-board camera as fast as possible or with as low energy consumption as possible. If two or more UAVs are available, it is natural to expect that the time spent searching the areas is reduced compared to if only one UAV is being used. However, coordination issues arise when using multiple UAVs for the search mission. Therefore, the UAVs should be coordinated such that:

1) The UAVs do not collide.
2) The UAVs do not search the same areas. A search with too much overlap will be ineffective when it comes to overall search time.
3) When the UAVs have finished the search mission, they should return back to their respective base stations.

Based on these requirements, the path planner should generate a safe and feasible path for all available UAVs, such that they are being used in the most efficient way to search the designated area.

In the waypoint generation algorithm the only information that will be considered are the constraints which limits the search area, and the range of the camera. The search area is assumed to be described by a union of rectangular areas. This implies that the search algorithm can be applied sequentially, for cases where the search area consists of multiple rectangular areas.

Consider a rectangular search area: $x_{\text{min}}$ and $y_{\text{min}}$ are the lowermost $x$- and $y$- coordinates of the search area with respect to a geographic frame, whereas $x_{\text{max}}$ and $y_{\text{max}}$ are the uppermost $x$- and $y$- coordinates.

Generated waypoints are added into the table $wp_{\text{set}} := \{ \begin{array}{c} x_i \ y_i \end{array} \}^T \in \mathbb{R}^2$, where $x_i$ and $y_i$ are the $x$- and $y$- coordinates and index $i$ is a sequence number.

The following calculation is done to find the coordinates of the first waypoint:
(\Delta x)^2 + (\Delta y)^2 \leq r_{\text{cam}}^2 \gamma \quad (1)

where \Delta x := w_{\text{set},x,1} - x_{\text{min}} and \Delta y := w_{\text{set},y,1} - y_{\text{min}} are the displacements of the first waypoint from \( x_{\text{min}} \) and \( y_{\text{min}} \). \( r_{\text{cam}} \) is the radius around the UAV which the camera covers, and \( 0 < \gamma \leq 1 \) is a constant which ensures that there is some overlap between the area which is covered by the camera at each waypoint. This number may depend on the UAVs maximum bank and climb angles in cases sensor or camera is not gimbal stabilized.

The approach presented in this paper assumes that the camera range \( r_{\text{cam}} \) is calculated for an constant altitude, and that the UAV altitude remain constant during the search.

Since we assume the image of the camera to be symmetric the distance from \( (x_{\text{min}}, y_{\text{min}}) \) to \( (w_{\text{set},x,1}, w_{\text{set},y,1}) \) is equal along both axes:

\[
(w_{\text{set},x,1} - x_{\text{min}}) = (w_{\text{set},y,1} - y_{\text{min}}) \quad (2)
\]

From equation (1) and (2) the coordinate of the first waypoint can be found:

\[
\Delta x = \frac{r_{\text{cam}}}{\sqrt{2}} \Rightarrow w_{\text{set},x,1} = x_{\text{min}} + r_{\text{cam}} \sqrt{\frac{\gamma}{2}} \quad (3)
\]

\[
\Delta y = \frac{r_{\text{cam}}}{\sqrt{2}} \Rightarrow w_{\text{set},y,1} = y_{\text{min}} + r_{\text{cam}} \sqrt{\frac{\gamma}{2}} \quad (4)
\]

The algorithm for generation of waypoints, that uniformly covers the search area is described by the following pseudocode:

\[
X = \text{ceil} \left[ \left( x_{\text{max}} - r_{\text{cam}} \sqrt{\frac{\gamma}{2}} \right) + \left( x_{\text{max}} + r_{\text{cam}} \sqrt{\frac{\gamma}{2}} \right) \right] + 1;
\]

\[
Y = \text{ceil} \left[ \left( y_{\text{max}} - r_{\text{cam}} \sqrt{\frac{\gamma}{2}} \right) + \left( y_{\text{max}} + r_{\text{cam}} \sqrt{\frac{\gamma}{2}} \right) \right] + 1;
\]

\[
i = 0;
\]

\[
\text{for} \{x = 1 : X\} \text{ do}
\]

\[
\text{for} \{y = 1 : Y\} \text{ do}
\]

\[
\text{if} \{x = 1 \&\& \ y = 1\} \text{ then}
\]

\[
\text{w}_{\text{set},x,1} = x_{\text{min}} + r_{\text{cam}} \sqrt{\frac{\gamma}{2}};
\]

\[
\text{w}_{\text{set},y,1} = y_{\text{min}} + r_{\text{cam}} \sqrt{\frac{\gamma}{2}};
\]

\[
\text{else if} \ y = 1 \&\& \ x \neq 1 \text{ then}
\]

\[
\text{w}_{\text{set},x,i} = \left( x_{\text{min}} + r_{\text{cam}} \sqrt{\frac{\gamma}{2}} \right) + (x - 1) 2r_{\text{cam}} \gamma;
\]

\[
\text{w}_{\text{set},y,i} = \text{w}_{\text{set},y,i-1};
\]

\[
\text{else if} \mod (x, 2) = 1 \text{ then}
\]

\[
\text{w}_{\text{set},x,i} = \left( x_{\text{min}} + r_{\text{cam}} \sqrt{\frac{\gamma}{2}} \right) + (x - 1) 2r_{\text{cam}} \gamma;
\]

\[
\text{w}_{\text{set},y,i} = \text{w}_{\text{set},y,i-1} + 2r_{\text{cam}} \gamma;
\]

\[
\text{else}
\]

\[
\text{w}_{\text{set},x,i} = \left( x_{\text{min}} + r_{\text{cam}} \sqrt{\frac{\gamma}{2}} \right) + (x - 1) 2r_{\text{cam}} \gamma;
\]

\[
\text{w}_{\text{set},y,i} = \text{w}_{\text{set},y,i-1} - 2r_{\text{cam}} \gamma;
\]

\[
\text{end if}
\]

\[
i = i+1;
\]

\[
\text{end for}
\]

\[
\text{end for}
\]

Here, ceil(·) is the ceiling operator such that ceil(x) is the smallest integer not less than x for some scalar x. mod (x, y) is the modulus after division such that mod (x, y) = x − ny for some positive scalars x, y where n is the biggest integer not greater than x/y.

III. IMPLEMENTATION OF MILP

A. Implementation of UAV dynamics

Since the search area has been defined and waypoints have been generated to ensure complete cover of the search area, we only need to define the path planning problem such that all generated waypoints are visited by the available UAVs.

The equation of motion in continuous time needs to be transformed to discrete time to fit the MILP framework. The discrete time dynamics of the UAV can be written as:

\[
s_{i+1} = A_d s_i + B_d u_i
\]

where \( A_d \) is the discrete system matrix and \( B_d \) is the discrete input matrix. \( s_i \) is the discrete state vector for each time step \( i \) and \( u_i \) is the input vector for each time step \( i \). \( N \) is the number of time steps in the path planning horizon. \( s_i \) and \( u_i \) should be determined by the optimization problem, such that the applied cost function is minimized.

In the implementation of the optimization problem in this paper, the following definition of the state and input vector has been made: \( s := [x \ y \ v_x \ v_y]^T \) and \( u := [f_x \ f_y]^T \). Here, \( x \) and \( y \) are the position components in a local East-North Cartesian coordinate system. \( v_x \) and \( v_y \) are the UAV velocity components, and \( f_x \) and \( f_y \) are the input components to the system.

The aircraft model will be included in the optimization problem, as well as restrictions on the position components, velocity components and input components. The UAV position should be limited to be inside a defined area, due to for instance fuel or battery capacity constraints, communication constraints, or airspace segregation constraints. The velocity components should be limited by the physical velocity constraints of the UAV, which should be given by the aircraft producer. The input components should be limited by an upper value of the effect the engines can give.

The position components, \( x \) and \( y \), are restricted by the following equations:

\[
x_i \leq x_{\text{lim, max}}
\]

\[
x_i \geq x_{\text{lim, min}}
\]

\[
y_i \geq y_{\text{lim, min}}
\]

\[
y_i \leq y_{\text{lim, max}}
\]

To include the restrictions on the velocity components, the magnitude of the \( x \)- and \( y \)- components will be considered. Since this optimization problem is described in the NED frame, information about the aircraft’s orientation is not considered by the path planner. This implies that the UAV can fly along it’s \( x \)- coordinate in the body frame, but in the NED frame this can result in a path along the \( y \)- coordinate. Therefore, we approximate the UAV velocity to be inside a circle in the \( x \)- \( y \)- NED frame, with the centre of the circle in
the origin. The velocity and input components are restricted by the following equations:
\[-f_{\text{max}} \leq f_x \leq f_{\text{max}} \quad \text{and} \quad -f_{\text{max}} \leq f_y \leq f_{\text{max}} \quad (7)\]
\[-V_{\text{max}} \leq v_x \leq V_{\text{max}} \quad \text{and} \quad -V_{\text{max}} \leq v_y \leq V_{\text{max}} \quad (8)\]
Here, \(V_{\text{max}}\) and \(f_{\text{max}}\) is the maximal magnitude of the velocity and input components respectively. There are mainly two aspects that has to be considered to decide the value of \(f_{\text{max}}\) to ensure a feasible path:
1) The maximum centripetal force that can affect the UAV due to a bank turn should be found.
2) The maximum force from the UAVs engine (moving the UAV forward) should be found.

To ensure a feasible path \(f_{\text{max}}\) should be given by:
\[f_{\text{max}} \leq \min(f_{\text{c,\text{max}}}, f_{\text{e,\text{max}}}) \quad (9)\]
Here, \(f_{\text{e,\text{max}}}\) is the maximum centripetal force that can affect the UAV due to a bank turn and \(f_{\text{c,\text{max}}}\) is the maximum force moving the UAV forward.

To describe the non-linear constraints given by equation (7) and (8) [13] suggest to approximate the constraints by using \(M\) number of linear constraints:
\[\forall i \in [0, \ldots, N - 1] \quad \forall \text{m} \in [1, \ldots, M] \quad -f_{\text{max}} \leq f_{xi} \sin\left(\frac{2\pi \text{m}}{M}\right) + f_{yi} \cos\left(\frac{2\pi \text{m}}{M}\right) \leq f_{\text{max}} \quad (10)\]
\[\forall i \in [1, \ldots, N] \quad \forall \text{m} \in [1, \ldots, M] \quad -V_{\text{max}} \leq v_{xi} \sin\left(\frac{2\pi \text{m}}{M}\right) + v_{yi} \cos\left(\frac{2\pi \text{m}}{M}\right) \leq V_{\text{max}} \quad (11)\]

### B. Implementation of waypoints
As described in [13], waypoint constraints can be added to the MILP problem in the following way:
\[\forall i \in [1, \ldots, N] \quad \forall c \in [1, \ldots, \text{W}] \quad \forall p \in [1, \ldots, n_p] \]
\[x_{ip} - x_{Wc} - \Delta \leq M_{big}(1 - b_{icp}) \quad (12a)\]
\[x_{ip} - x_{Wc} - \Delta \geq -M_{big}(1 - b_{icp}) \quad (12b)\]
\[y_{ip} - y_{Wc} - \Delta \leq M_{big}(1 - b_{icp}) \quad (12c)\]
\[y_{ip} - y_{Wc} - \Delta \geq -M_{big}(1 - b_{icp}) \quad (12d)\]
\[\forall c \in [1, \ldots, \text{W} - n_p] \quad \sum_{i=1}^{n_p} \sum_{p=1}^{N} b_{icp} = 1 \quad (13)\]

Here, \(x_{ip}\) and \(y_{ip}\) are the \(x\)- and \(y\)- coordinates for UAV \(p\) at time step \(i\). \(x_{Wc}\) and \(y_{Wc}\) are the \(x\)- and \(y\)- positions for waypoint \(c\). \(M_{big}\) is a large positive number which is much bigger than the area in which the UAV is operating. The variable \(\Delta \geq 0\) is included to allow a waypoint to be viewed as visited if the time sample is placed somewhere inside the range \(\Delta\) from the waypoint. This may result in a smoother path. The binary variable, \(b_{icp}\) becomes 1 when UAV \(p\) visits waypoint \(c\) at time step \(i\), and is zero otherwise. Restriction (13) is introduced to require that each waypoint is visited once and only once.

### C. Implementation of the order the waypoints are visited
We require that the waypoints are visited in a specific order and introduce the constraints:
\[\theta_{c+1} \geq \theta_c \quad (14)\]
and
\[\theta_c = \sum_{p=1}^{n_p} \sum_{i=1}^{N} b_{icp} \quad (15)\]
We emphasize that \(c\) is the enumeration of waypoints. With the above constraints, \(\theta_c\) is given the value of the time step at which waypoint \(c\) is visited by a UAV. See [14] for further details.

### D. Implementation of obstacles
When the UAV is in the air it has to consider no-fly areas, terrain, structures and other aircrafts. The obstacles can be non-convex, but for scope of implementation we consider only rectangular obstacles. The position of the obstacle is denoted by the coordinates of its lower left and upper right corner points: \((x_{\text{min}}, y_{\text{min}})\) and \((x_{\text{max}}, y_{\text{max}})\). At every time step \(i\) the position \((x_i, y_i)\) of the vehicle must lie outside of the obstacle.

Let the \(t_{izk}\) be a binary variable and let \(M_{big}\) be a large arbitrary positive number. The obstacle avoidance may then be described by the following MILP [4]:
\[\forall i \in [1, \ldots, N], \forall z \in [1, \ldots, O] \quad x_i \leq x_{\text{min},c} + M_{big} t_{iz3} \quad (16a)\]
\[-x_i \leq -x_{\text{max},c} + M_{big} t_{iz2} \quad (16b)\]
\[y_i \leq y_{\text{min},c} + M_{big} t_{iz3} \quad (16c)\]
\[-y_i \leq -y_{\text{max},c} + M_{big} t_{iz4} \quad (16d)\]
\[\sum_{k=1}^{4} t_{izk} \leq 3 \quad (17)\]
Here, \(O\) is the number of obstacles, and \(t_{izk}\) are binary variables which ensure that at least one constraint is active every time step.

For situations where more than one UAV is taking part in the search mission, a restriction which ensures that the UAVs do not collide with each other should be included in the optimization problem. However in the implementation and simulations done in this paper, the UAVs are allocated to visit waypoints in pre-segregated parts of the search area such that they will not collide.

### E. Objective Function
In this implementation we seek an optimal solution that minimizes the time the UAVs use from their base station, through all the waypoints and back to the base station. This is to ensure that the search is done as fast as possible. Therefore, we define the cost function in order to minimize the search time. To handle multiple aircrafts, we implement the constraints [14]:
\[\forall p \in [1, \ldots, n_p], \forall i \in [1, \ldots, N]\]
\[ \theta_p^{\text{finish}} \leq M^{\text{finish}} (1 - b_{icp}) + ib_{icp} \quad (18) \]
\[ \theta_p^{\text{finish}} \geq i (1 - b_{icp}) \quad (19) \]

where the binary variable \( b_{icp} \) is equal to 1 if and only if vehicle \( p \) return to base station \( c \) at time step \( i \). \( M^{\text{finish}} \) is a constant which is sufficiently large, for example \( M^{\text{finish}} := N \) as suggested in [14]. These constraints enforce \( \theta_p^{\text{finish}} \) to be the number of time steps elapsed before vehicle \( p \) return to its base station. Since we want to minimize the overall mission time, the time elapsed until the last vehicle arrives the final waypoint, we introduce the variable \( \eta^{\text{finish}} \) and require that \( \forall p \in [1, ..., n_p] \)

\[ \eta^{\text{finish}} \geq \theta_p^{\text{finish}} \quad (20) \]

We set our cost function to be minimized as

\[ f^{\text{finish}} = \eta^{\text{finish}} \quad (21) \]

see [14], [15] for details.

IV. SIMULATIONS AND RESULTS

In this section results from the simulations are presented. In the simulations the discretization time was set to 3 seconds. The following scenarios have been considered:

- Simulations of a search mission when one UAV is applied.
- Simulations of a search mission where two UAVs are applied. Initially, one of the UAVs is located at \((x, y) = (0,0)\) while the other UAV is located at \((x, y) = (400,0)\).
- The UAVs initial positions have been required to be the last waypoint the UAVs should visit, to make sure the UAVs return to their base stations after ended search.
- The value \( \gamma \) in the waypoint generation algorithm has been set to 0.8.
- Identify the time it takes to solve the optimization problem when the restriction (14) is implemented and when it is not implemented.
- Identify the time it takes from the solver starts to solve the optimization problem, and until the UAVs has fulfilled the search. These results will be applied in the discussion on the overall time it takes to complete a search mission, both for the case where the restriction (14) is included and not included in the optimization problem. The time a UAV use to complete the search has been found by investigating at which time step the UAV arrives at the base station. The time step, the UAV arrives at the base station on, is multiplied by the discretization time \( T_d \) to calculate the search time in seconds.
- Identify that the camera system on the UAV covers the defined search area to verify that the waypoint generation algorithm works efficiently. The camera range, \( r_{\text{cam}} \), is changed to verify that the waypoints are placed at different places when the camera range changes.

In this paper the simulations have been done on a computer with the following specifications:

- Operating System: Windows 7 enterprise 64-bit
- MILP solver: Gurobi Optimizer 4.5
- Simulation environment: YALMIP
- Processor: Intel(R) Core(TM)i5 -2500 CPU @ 3.30GHz (2 CPUs)
- Memory: 8.00 GB RAM

During the simulations the variable, \( M_{\text{big}} \), was set to 1000. \( x \) and \( y \) are restricted by: \( x_{\text{min}} = 0, x_{\text{max}} = 850, y_{\text{min}} = 50 \) and \( y_{\text{max}} = 550 \).

A. Case 1 - one UAV and no restriction on the order the waypoints are visited

This first case considers a search mission where only one UAV participate in the search, and its base is located at the following coordinates: \((x,y) = (0,0)\). The camera range \( r_{\text{range}} \) has been set to 200 meters.

The time horizon was set to be 150 seconds. The MILP problem took 74 seconds to be solved, and the UAV used 135 seconds to fly the search mission. The UAV path is shown in Figure 1 with the camera range included around each time step.

B. Case 2 - one UAV and a restriction on the order the waypoints are visited

This case considers the same scenario as Case 1, but the restriction on waypoint visiting (14) has been included in the optimization problem. The time horizon \( N \) was in this case increased to 180 seconds, to be able to find a feasible solution to the optimization problem. The computation time was 24 seconds and the UAV used 156 seconds to fly the search mission. The path is shown in Figure 2 with the camera range included around each time step. In this case it takes longer time to do the search compared to the case where there were no restrictions in the waypoint visitation order. This is reasonable because when more constraints are added to the optimization problem, this will affect the optimization problem.
problem and can result in a solution where the cost-function is bigger.

C. Case 3 - the camera range, $r_{cam}$, is changed

If the camera on the UAV has a smaller range, this will result in closer waypoints. In Figure 3 the search area was defined to be the same as in Case 1 and Case 2, but the camera range has been set to be 178 meters, which is 22 meters smaller than the simulations in Case 1 and Case 2. As can be seen this new camera range results in more waypoints which have to be visited to ensure that the area is covered by the camera. In this case the horizon was 180 seconds. The computation time took 185 seconds and the UAV used 169 seconds to complete the search.

D. Case 4 - Two UAVs and no restriction on the order the waypoints are visited

This case considers two UAVs taking part in the search mission. In this simulation no collision avoidance constraints have been implemented, and there is no restriction on which waypoints that should be visited by which aircraft. The time horizon was in this case set to be 90 seconds, and the optimization problem took 7 seconds to be solved. The resulting paths are shown in Figure 4 where the camera range has been plotted with a circle around each UAV sample.

E. Case 5 - Two UAVs and a restriction on the order the waypoints are visited

In this case there is included a restriction which tells which waypoints each of the UAVs are allocated to visit. The scope of this implementation is to find a simple and general distribution of tasks for cases where UAVs are distributed along the search area. The UAV with base in (0,0) is responsible for visiting the waypoints $c \in \{1, ..., c_1\}$, where $c_1$ is defined by the following equation:

$$c_1 := \text{ceil} \left[ \frac{W}{n_p} \right]$$

(22)

The UAV with base in (400,0) is responsible for visiting the remaining waypoints $c \in \{c_1 + 1, ..., W\}$. For each of the UAVs the restriction (14) is included such that the UAVs visit their waypoints in a given order. When the horizon was 90 seconds the optimization problem took 6 seconds to be solved. The resulting paths are shown in Figure 5 where the camera range has been plotted with a circle around each UAV sample.
V. CONCLUSION AND DISCUSSION

This paper considers an approach using MILP to solve path planning problems related to search missions. Waypoints are generated and the defined search area is covered by the UAVs sensor system. The optimal path and inputs to the system are found for each time step, by solving a Centralized Optimization Problem from the initial to the final state, over a given time horizon, N. This fixed horizon strategy requires that the time horizon is chosen long enough with respect to the operation area and the physical limitations given by UAV dynamics and the waypoints. If the time horizon is chosen too short the optimization problem will be infeasible. If the area of operation becomes large, this requires a long time horizon which again will result in a more computationally challenging optimization problem.

As was shown by the simulation results of this paper, we were able to reduce the optimization horizon quite substantially with two UAVs, which in turn reduced the computation time.

For a UAV which takes part in a search mission it has to be expected that the flight area will be significantly larger than the quite limited area considered in this paper. This implies that the time horizon must be large in order to find a feasible solution to the optimization problem. One practical way to deal with this is to split the mission into multiple simpler missions.

During a search and rescue operation, where the target is for example missing people in need, the time taken from the decision is made about making a search with UAVs to the search is conducted, is important. It is therefore reasonable to find methods to minimize the time it takes to compute the optimal state and input variables, such that the overall time for both path-computation and the time it takes to complete the flight is as small as possible.

In Case 5 it was shown that an implementation where the waypoints were dedicated to be visited by a given UAV reduced the computation time significantly. Generally, the results show that if it is possible to reduce the complexity of the optimization problem, the computation time is reduced.

To reduce the computation time further, the optimization problem described in this paper could be solved in a Receding Horizon framework as described in for instance [16]. In Receding Horizon Control (also called Model Predictive Control) an input trajectory that optimizes the plant’s output over a period of time, called the planning horizon, is calculated. The input trajectory is implemented over a shorter execution horizon, and the optimization is performed again starting from the state that is reached. This may make the path planner more robust, since the path will be calculated for a defined time horizon.

REFERENCES


