Multi-Robot Waypoint Inspection Plan Mixed Integer Linear Programming Project

Juan Carlos Cruz - ira406

ME 6033 Linear and Mixed Integer Optimization

Abstract

This report

Introduction

This work is motivated by a previous project the author has worked on involving a multi-robot approach to outdoor construction site inspection.

Recent use of precast concrete elements which are manufactured off-site and then transported to the construction site has led to the need for more efficient inspection methods. The use of autonomous robots for such inspections is a promising approach as robot platforms can be equipped with a variety of sensors to enable rapid and accurate inspection which can be easily documented in cloud based systems. Due to the nature of construction sites, objects are often moved from location to location, thus requiring a flexible inspection plan that can be adapted to the current state of the site.

One such approach is to use a multi-robot system where one robot type can quickly locate the correct inspection target and then a more specialized robot can perform the inspection. The initial location of the inspection target could rapidly be done by aerial robot platforms, such as drones, which can cover large areas but due to their payload limitations cannot perform the inspection themselves. A ground mobile robot platform could then be dispatched to the location of the inspection target to perform the actual inspection.

In such an approach, there are various details which must then be considered and implemented, however, in this report we will focus on the optimization of the inspection sequence planning.

This paper presents a mixed integer linear programming (MILP) approach to the problem of multi-robot waypoint inspection planning. The following sections will describe the problem formulation, modeling, implementation, and discussion of the results.

Problem Formulation

For the modeling of the problem we consider two robot types, an aerial robot and a ground mobile robot. The number of robots of each type is specified as a parameter of the problem. Each robot type starts a depot location where they are charged and dispatched to the inspection targets.

The inspection targets are marked as waypoints in a 2D plane. For any waypoint, an aerial robot must first be dispatched to the waypoint to verify it's location, only then can a ground robot visit the waypoint. Each robot time has a fixed amount of time it must remain at the waypoint, this time is to simulate processing, verification, and/or inspection time by the respective robot type.

Each robot type has a limited operation time based on it's battery capacity. Each robot type has a fixed speed it operates at. Any dispatched robot must be able to return to the depot before the battery runs out.

For the purpose of this report we consider only a single inspection loop and model the problem such that the maximum number of waypoints are visited within that single loop.

The units used are minutes for time and meters for distance.

In the next section we will describe the mathematical formulation of the problem.

Modeling

Definitions

Sets and Indices

- N: Set of all waypoints indexed by $i \in 1, 2, \ldots, n$
- K: Set of aerial robots indexed by $k \in 1, 2, \ldots, k_{\max}$
- L: Set of ground robots indexed by $l \in 1, 2, \ldots, l_{\max}$
- d_A: Aerial robot depot
- d_G: Ground robot depot

Parameters

- p_i: Location (coordinates) of waypoint $i \in N$
- p_{d_A}: Location of aerial robot depot
- p_{d_G}: Location of ground robot depot
- $\text{dist}(i, j)$: Euclidean distance between locations i and j
- s_A: Speed of aerial robots (distance per minute)
- s_G: Speed of ground robots (distance per minute)
- t_A^{insp}: Inspection time for aerial robots at each waypoint (minutes)
- t_G^{insp}: Inspection time for ground robots at each waypoint (minutes)
- $T_A^{\max}$: Maximum operation time for each aerial robot (minutes)
- $T_G^{\max}$: Maximum operation time for each ground robot (minutes)

Derived Parameters

- $t_{ij}^A = \frac{\text{dist}(i,j)}{s_A}$: Travel time for aerial robots from i to j (minutes)
- $t_{ij}^G = \frac{\text{dist}(i,j)}{s_G}$: Travel time for ground robots from i to j (minutes)
- M_A: Big-M value for aerial robot time constraints
- M_G: Big-M value for ground robot time constraints

Decision Variables

- $w_i^{a,k}$: Binary variable equals 1 if aerial robot k visits waypoint i, 0 otherwise
- $w_i^{g,l}$: Binary variable equals 1 if ground robot l visits waypoint i, 0 otherwise
- a_i^k: Time when aerial robot k completes inspection at waypoint i (minutes)
- g_i^l: Time when ground robot l completes inspection at waypoint i (minutes)
- use_k^a: Binary variable equals 1 if aerial robot k is used, 0 otherwise
- use_l^g: Binary variable equals 1 if ground robot l is used, 0 otherwise

- $z_i^{k,l}$: Binary variable equals 1 if ground robot l visits waypoint i after aerial robot k inspection, 0 otherwise

Objective Function

Since the goal of the problem is to maximize the number of waypoints visited, and a waypoint is only completely visited when a ground robot has visited it, we can define the objective function as the sum of all waypoints visited by ground robots.

$$\text{Maximize} \sum_{i \in N} \sum_{l \in L} w_i^{g,l} \tag{1}$$

Constraints

Assignment Constraints

To ensure that each waypoint is assigned to at most one aerial robot and one ground robot, we can state that the sum of a robot type which visits a waypoint must be less than or equal to 1.

$$\sum_{k \in K} w_i^{a,k} \leq 1 \quad \forall i \in N \tag{2}$$

$$\sum_{l \in L} w_i^{g,l} \leq 1 \quad \forall i \in N \tag{3}$$

Robot Usage Constraints

In order to capture the usage of a robot type for an inspection we must specify that *A robot is used if and only if it visits at least one waypoint*.

The IF statement can be expressed by stating that if the use binary variable is 1, then the sum of waypoints visited by that robot type must be greater than or equal to 1.

The ONLY IF statement can be expressed by stating that if the sum of waypoints visited by that robot type is greater than or equal to 1, then the use binary variable must be 1.

Since there are n waypoints which could sum up to n, we can multiply the use binary variable by n to ensure that the sum of waypoints visited is less than or equal to n. In the case that the use binary variable is 0, the sum of waypoints visited by the robot would be forced to be 0.

Then for aerial robots we have:

$$\sum_{i \in N} w_i^{a,k} \geq \text{use}_k^a \quad \forall k \in K \tag{4}$$

$$\sum_{i \in N} w_i^{a,k} \leq n \cdot \text{use}_k^a \quad \forall k \in K \tag{5}$$

And for ground robots we have:

$$\sum_{i \in N} w_i^{g,l} \geq \text{use}_l^g \quad \forall l \in L \tag{6}$$

$$\sum_{i \in N} w_i^{g,l} \leq n \cdot \text{use}_l^g \quad \forall l \in L \tag{7}$$

Precedence Constraints

For a ground robot to be able to visit a waypoint, it must have first been visited by an aerial robot. Then for any waypoint visited by the ground robot, the visit binary for that waypoint by an aerial robot must be at least 1.

$$w_i^{g,l} \leq \sum_{k \in K} w_i^{a,k} \quad \forall i \in N, \forall l \in L \tag{8}$$

However, this constraint does not enforce any timing constraints. For that we must add that timestamp the ground robot completes its inspection at a waypoint i, g_i^l, must be greater to or equal to the timestamp of when the aerial robot robot completed it's inspection of the waypoint.

We must be careful here though and filter for any robot type which was not used in the inspection loop and ground robots which are not visiting that specific waypoint.

To do so we can use a big M value to ensure a linear system. The big M value ensures that if both of the aforementioned conditions are met, then the RHS of the following equation will be satisfied.

$$g_i^l \geq a_i^k - M_G \cdot (1 - z_i^{k,l}) - M_G \cdot (2 - \text{use}_k^a - \text{use}_l^g) \quad \forall i \in N, \forall k \in K, \forall l \in L \tag{9}$$

Now to ensure that the binary variable, $z_i^{k,l}$, can be 1 if and only if the ground robot l visits waypoint i after aerial robot k's inspection we must define certain constrains on $z_i^{k,l}$.

The lower bound for this requirement can be stated as:

$$z_i^{k,l} \leq w_i^{g,l} \quad \forall i \in N, \forall k \in K, \forall l \in L \tag{10}$$

$$z_i^{k,l} \leq w_i^{a,k} \quad \forall i \in N, \forall k \in K, \forall l \in L \tag{11}$$

$$z_i^{k,l} \leq \text{use}_k^a \quad \forall i \in N, \forall k \in K, \forall l \in L \tag{12}$$

$$z_i^{k,l} \leq \text{use}_l^g \quad \forall i \in N, \forall k \in K, \forall l \in L \tag{13}$$

The upper bound is set by considering the previous constraint variables as true then subtracting the total -1.

$$z_i^{k,l} \geq w_i^{g,l} + w_i^{a,k} + \text{use}_k^a + \text{use}_l^g - 3 \quad \forall i \in N, \forall k \in K, \forall l \in L \tag{14}$$

Time Constraints

A minimum inspection time at a waypoint can be implemented by specifying the time a robot completes an inspection is greater to or equal to the time it took the robot to complete an inspection at a waypoint. We multiply by the visiting waypoint variable to ensure this is only considered if a robot visited that waypoint.

$$a_i^k \geq t_A^{\text{insp}} \cdot w_i^{a,k} \quad \forall i \in N, \forall k \in K \tag{15}$$

$$g_i^l \geq t_G^{\text{insp}} \cdot w_i^{g,l} \quad \forall i \in N, \forall l \in L \tag{16}$$

The maximum operation time constraint can be implemented by considering the time when the robot completes an inspection at waypoint i plus the time it takes the robot to travel to the waypoint. This time must be less than the maximum allowable operating time of a robot. We enforce this limit by introducing a big M variable for the robot type, such that it can only considered if the robot did visit the waypoint.

$$a_i^k + t_{i,d_A}^A \cdot w_i^{a,k} \leq T_A^{\max} + M_A \cdot (1 - w_i^{a,k}) \quad \forall i \in N, \forall k \in K \tag{17}$$

$$g_i^l + t_{i,d_G}^G \cdot w_i^{g,l} \leq T_G^{\max} + M_G \cdot (1 - w_i^{g,l}) \quad \forall i \in N, \forall l \in L \tag{18}$$

Approximate Route Length Constraints

Our problem formulation shares similarities to the Multiple Traveling Salesman Problem which typically requires routing variables and sub-tour elimination constraints. This approach was attempted by the author but was not satisfactorily implemented, thus a simplification was needed to ensure plans would be feasible and solved quickly (less than hour wallclock time). This is discussed more in the **??** section.

We wish to specify that the total mission time of either robot once it leaves the depot is less than or equal to the maximum allowable operation time of a given robot. Since the exact routes for the robot travel is not known, we estimate expected route travel time to be the one way distance from the depot point to any visited waypoint point. The inspection time for any visited waypoint is then added to this value. The sum of these two values should then be less than or equal to the maximum operating time for that robot platform, where again we use big M value to make the constraint non-binding if a robot is not used.

$$\sum_{i \in N} w_i^{a,k} \cdot (t_{d_A,i}^A) + \sum_{i \in N} w_i^{a,k} \cdot t_A^{\text{insp}} \leq T_A^{\max} + M_A \cdot (1 - \text{use}_k^a) \quad \forall k \in K \tag{19}$$

$$\sum_{i \in N} w_i^{g,l} \cdot (t_{d_G,i}^G) + \sum_{i \in N} w_i^{g,l} \cdot t_G^{\text{insp}} \leq T_G^{\max} + M_G \cdot (1 - \text{use}_l^g) \quad \forall l \in L \tag{20}$$

Implementation

A Python program is created to solve the MILP formulation. We use PuLP, a LP and MILP solving library in Python [1]. The code for the developed model can be found at `https://github.com/jc-cr/multirobot_inspection_optimizer`.

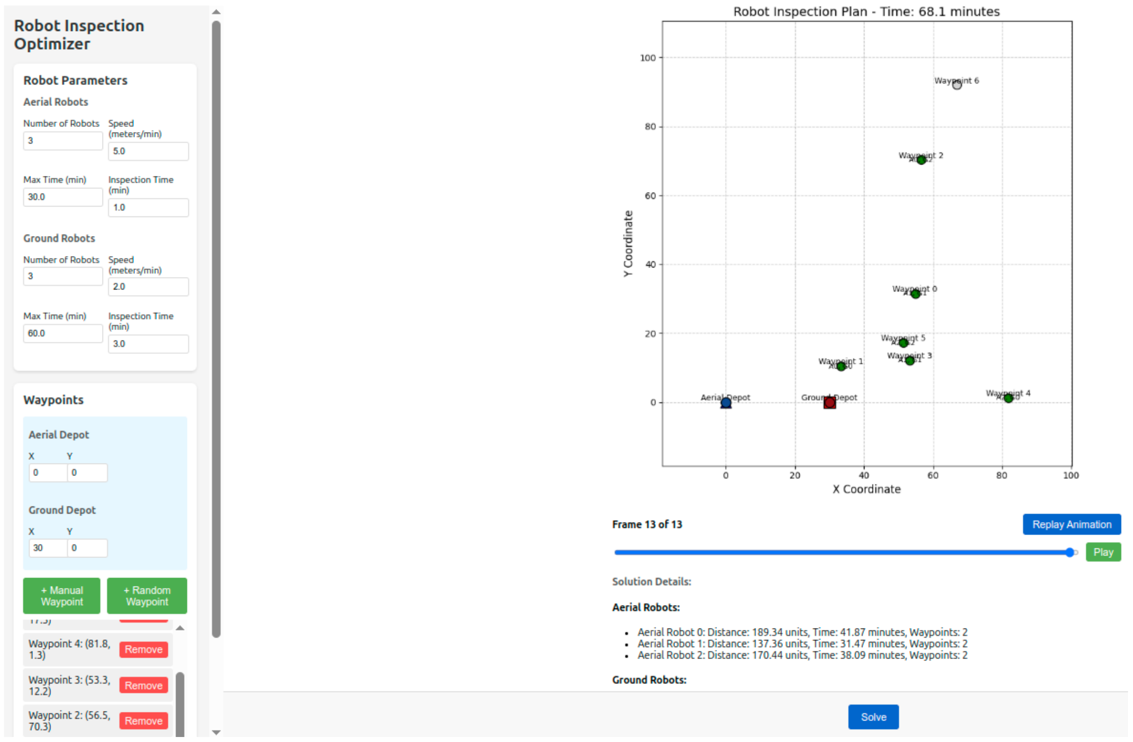
To showcase the models capabilities an interactive demo is created with the same repository. The demo creates a browser based GUI which allows for the user to input robot parameters, set waypoint locations, and then visualize the results. An example of this can be seen in Figure 1.

Discussion

As mentioned in the Modeling section, the formulation of our model resembles that of the multiple traveling salesman problem (mTSP) . Previous works such as [2] [3], show how such an approach can be used for planning in multi-robot systems.

Typically in a MILP formulation for a mTSP, one would introduce routing variables and constraints for subtour elimination [4]. However, the inclusion of route considerations can slow down the solving speed for the optimization problem and may not even be needed if one only cares about capabilities and cost rather than routing; in such cases various methods have been proposed to approximate travel distances in a problem [5].

In the case of our model, it was noted through visual plotting of random waypoints on a 2D plane that the length of the optimal route could roughly be approximated by the one way distances from the depot



Robot Parameters

Aerial Robots

Number of Robots: 3
Speed (meters/min): 5.0
Max Time (min): 30.0
Inspection Time (min): 1.0

Ground Robots

Number of Robots: 3
Speed (meters/min): 2.0
Max Time (min): 60.0
Inspection Time (min): 3.0

Waypoints

Aerial Depot

X: 0 Y: 0

Ground Depot

X: 30 Y: 0

Waypoint 4: (81.8, 1.3)
Waypoint 3: (53.3, 12.2)
Waypoint 2: (56.5, 70.3)

Frame 13 of 13

Solution Details:

Aerial Robots:

- Aerial Robot 0: Distance: 189.34 units, Time: 41.87 minutes, Waypoints: 2
- Aerial Robot 1: Distance: 137.36 units, Time: 31.47 minutes, Waypoints: 2
- Aerial Robot 2: Distance: 170.44 units, Time: 38.09 minutes, Waypoints: 2

Ground Robots:

Figure 1: Developed browser based GUI showcasing a solved inspection plan.

to each waypoint. From limited testing this method, however, seems to overestimate the time required for completion of an inspection loop.

Conclusion

In this report a Mixed Integer Linear Program was developed to optimize the inspection coverage of a ground and aerial robot fleet. We formulated a model which maximized the number of waypoints covered by N robots with variable parameters and operating capabilities. This formulation was then implemented in a Python program using the PuLP library of solvers and a browser based GUI was developed to allow for an interactive demonstration environment.

References

[1] S. Mitchell, M. O'Sullivan, I. Dunning, and J. Roy, "Pulp: A linear programming toolkit for python," 2022, mIT License. [Online]. Available: https://github.com/coin-or/pulp

[2] P. Oberlin, S. Rathinam, and S. Darbha, "A transformation for a heterogeneous, multi-depot, multiple traveling salesman problem," in *Proceedings of the American Control Conference*, St. Louis, June 2009, pp. 1292–1297, june 10 – 12, 2009.

[3] A. Albert, F. S. Leira, and L. Imsland, "Uav path planning using milp with experiments," *Modeling, Identification and Control*, vol. 38, no. 1, pp. 21–32, 2017.

[4] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *OMEGA: The International Journal of Management Science*, vol. 34, no. 3, pp. 209–219, 2006.

[5] D. Nicola, R. Vetschera, and A. Dragomir, "Total distance approximations for routing solutions," *Computers Operations Research*, vol. 102, pp. 67–74, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054818302673

[6] N. Christofides, A. Mingozzi, and P. Toth, "Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations," *Mathematical Programming*, vol. 20, pp. 255–282, 1981.

[7] G. Laporte, "The vehicle routing problem: an overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, pp. 345–358, 1992.

[8] G. Laporte and Y. Nobert, "A cutting planes algorithm for the m-salesmen problem," *Journal of the Operational Research Society*, vol. 31, pp. 1017–1023, 1980.