AE 740 Project: Final Report

Prepared for: Professor Atkins
Prepared by: Alexey Morozov,
Siddharth Kirtikar,
Anthony D'Amato

December 16, 2009

[1]Department of Aerospace Engineering, The University of Michigan, Ann Arbor, MI 48109-2140

Problem Statement

The package delivery service FEDEX has chosen to automate its cross country package delivery service by purchasing two delivery UAV's to complete state to state pick up and drop off of packages. The two UAV's are not identical, in terms of performance, furthermore the different delivery locations have varying package handling times and refuel rates.

Noting these complications the goal is select routes for the UAV's such that all the packages are picked up and delivered in a timely fashion. Packages may have different delivery priorities, and this must also be accounted for.

A number possible methods for solving this problem are examined, first, we examine the optimal routes by examining scenarios where all factors are known, for example, handling and refuel times. For this case solutions can be obtained by exhaustive search. Second, we consider cases when system specifics are not known. In this case tree search methods are employed using knowledge such as distance between nodes in a given transition.

Assumptions

The problem detailed above, is constrained by a number of simplifying and complicating assumptions.

Aircraft Constraints

- Aircraft have a limited amount fuel.
- Packages are acquired enroute.
- Aircraft are initially empty.
- The aircraft which picks up a specific package may only deliver that package to its delivery node.

Map Constraints

- Map of routes between nodes are known.
- Number of nodes and locations are fixed. The delivery and pick up nodes do not move.

- All nodes may refuel UAV's, however refuel rates may not be uniform.
- The rate at which packages are loaded and unloaded may differ from node to node.

Package Constraints

- All package pickup and delivery locations are known *a priori*.
- Packages have priority ratings. The higher the rating, the greater the incentive to deliver that package first.

Hybrid Automata

In order to simulate the network of drop off and pickup nodes and a fleet of UAV's, we specify a number of variables which we use when simulating the system. First, we define a set of UAV specific variables:

$$\begin{aligned}
&\text{Elapased Time} = t_i \ \{hours\},\\
&\text{Climb and Descend Velocity} = V_{d,i} \ \{kilometers\ per\ hour\},\\
&\text{Altitude} = D_{\text{alt,i}} \ \{kilometers\},\\
&\text{Cruise Velocity} = V_{c,i} \ \{kilometers\ per\ hour\},\\
&\text{Fuel BurnRate} = F_{r,i} \ \{gallons\ per\ hour\},\\
&\text{Fuel Remaining} = F_{l,i} \ \{gallons\},\\
&\text{Distance Traveled} = \delta_i \ \{kilometers\},\\
&\text{Distance Remaining to Next Node} = \epsilon_i \ \{kilometers\},\\
&\text{Accumulated PackagedPriorityCost} = C_{p,i} \ \{hours\},\\
&\text{Number of PackagesinHold} = H_i \ \{packages\},\\
&\text{Node Transitions} = \Gamma_i \ \{dimensionless\},\\
&\text{Transition Number} = S_i \ \{dimensionless\},
\end{aligned}$$

where $\Gamma_i \in \mathbb{R}^m$, and $i : 1 \leq i \leq N_{\text{UAV}}$ is a specific UAV and N_{UAV} is the total number of UAV's. These variables are UAV specific and so each UAV would have a corresponding set of these variables which will be used to define the dynamics of problem. Each variable is self explanatory, and the units are included. Note that $\Gamma_i(S_i)$ is the node location of UAV i at $t_i(k)$.

The second set of variables, also concerns the UAV's flight characteristics but are considered policy, which means that each UAV obeys these rules during flight,

$$\begin{aligned}
&\text{Cruise Altitude} = d_{\text{alt}}\ \{\mathit{kilometers}\},\\
&\text{Climb and Descendangle} = \alpha\ \{\mathit{degrees}\}.
\end{aligned}$$

The third set of variables is node specific, namely, each node has a set of the following variables which specify refueling and package handling time,

$$\begin{aligned}
&\text{Node Latitude} = \lambda_j\ \{\mathit{radians}\}\\
&\text{Node Longitude} = \theta_j\ \{\mathit{radians}\}\\
&\text{Package Handling Time} = P_{H,j}\ \{\mathit{hours\ package}\},\\
&\text{Refuel Rate} = R_{F,j}\ \{\mathit{gallons\ per\ hour}\},
\end{aligned}$$

where $j : 1 \leq j \leq J$ represents a node and J is the total number of nodes. Note that the package handling time refers to the time required to load, or unload a package depending on the action taken at a specified node.

Finally, the last set of variables are package specific,

$$\begin{aligned}
&\text{Pickup Locations} = \Upsilon\ \{\mathit{dimensionless}\},\\
&\text{Dropoff Locations} = \Omega\ \{\mathit{dimensionless}\},\\
&\text{PackagePriority} = p_l\ \{\mathit{dimensionless}\},
\end{aligned}$$

where $l : 1 \leq l \leq L$ represents a package and L is the total number of packages, and $\Upsilon, \Omega \in \mathbb{R}^L$. The vectors Υ and Ω contain the pickup and drop off nodes of all the packages.

The hybrid automata used to simulate the problem given this set of variables can be specified using 6 states. We define the solutions to the dynamics for each node in terms of discrete events based on an index k, this enabled the system to programmed with ease in MATLAB. Furthermore, jump and invariant conditions are also specified in each section. Figure 1 shows the linkage of all the nodes. Note that in the list of dynamics for each node, if a variable is assumed to remain constant, it is excluded from the dynamics for that node, for example, the climb, cruise and descend states do not include H_i since for these states

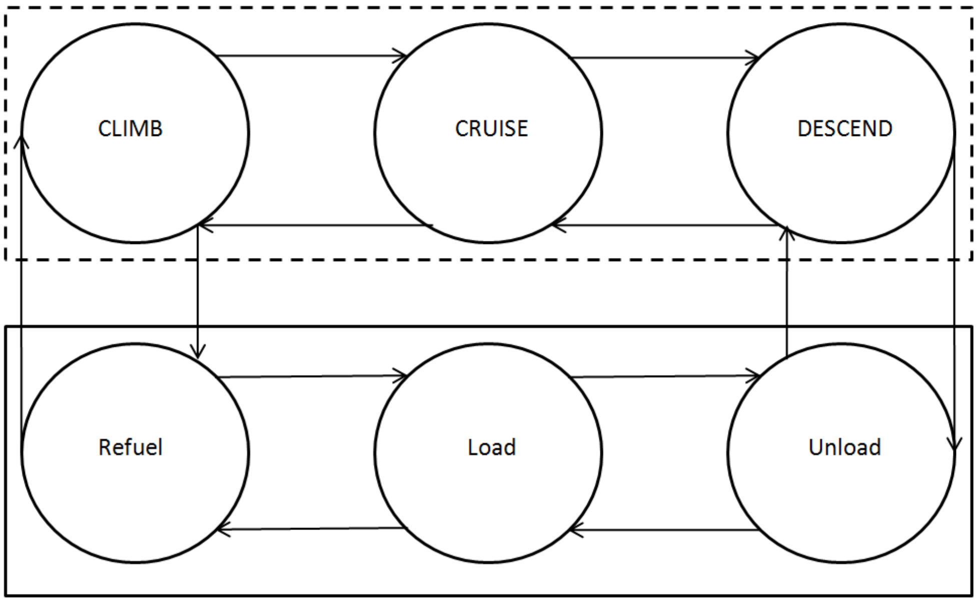

Figure 1: Hybrid Automata for the system. The dashed box represents states which are dependent on UAV characteristics, the solid box are states which are dependent on node characteristics. State dynamics, jump, and invariant conditions are outlined in the body of the section.

$H_i(k+1) = H_i(k)$, note that it is impossible to load/unload in midair.

Furthermore, we simplify the problem by assuming that there are no external forces on the UAV's, such as wind. In terms of the hybrid automata, although feasibly from the descend state, one might transition to the cruise to maintain altitude, or the unload state to unload cargo, as the problem is set up the automata will never transition from decent to cruise because the aircraft is able to maintain altitude exactly. However, although the dynamics due not account for these external influences, the jump and invariant conditions are specified as if they were possible.

Climb

The dynamics for the climb state of the system are

$$\delta_i(k+1) = \delta_i(k) + \frac{d_{\text{alt}}}{\tan\alpha}, \tag{1}$$

$$t_i(k+1) = t_i(k) + \frac{\delta_i(k)}{V_{d,i}}, \tag{2}$$

$$F_{l,i}(k+1) = F_{l,i}(k) - F_{r,i}t_i(k), \tag{3}$$

$$\epsilon_i(k+1) = \epsilon_i(k) - \delta_i(k), \tag{4}$$

$$D_{\text{alt,i}}(k+1) = d_{\text{alt}}, \tag{5}$$

while in this state the aircraft climbs at the specified climb velocity to the policy specified cruise altitude, at the expense of fuel and time. We note that ϵ which is dependent on the distance between specific nodes, which is calculated using the Haversine formula,

$$d = 2(6373)\sin^{-1}\left(\sqrt{\sin^2\frac{\lambda_1-\lambda_2}{2} + \cos\lambda_1\cos\lambda_2\sin^2\frac{\theta_2-\theta_1}{2}}\right), \tag{6}$$

where d is the distance between nodes 1 and 2. **Jump Conditions:** The system may transition to the cruise state from the climb state, the conditions under which this transition occurs is $D_{\text{alt,i}}(k) \geq d_{\text{alt}}$. Furthermore, if $F_{l,i}(k) < (\frac{\epsilon_i(k)}{V_{c,i}} + 2\frac{\delta_i(k)}{V_{d,i}})F_{r,i}$ then the system will transition to the refuel state. **Invariant Conditions:** The condition under which the system will remain in the climb state is $D_{\text{alt,i}}(k) < d_{\text{alt}}$.

Cruise

The dynamics for the cruise state are

$$t_i(k+1) = t_i(k) + \frac{\epsilon_i(k)}{V_{c,i}} - \frac{\delta_i(k)}{V_{d,i}}, \tag{7}$$

$$\delta_i(k+1) = \delta_i k + \left(\epsilon_i(k) - \frac{d_{\text{alt}}}{\tan-\alpha}\right), \tag{8}$$

$$F_{l,i}(k+1) = F_{l,i}(k) - F_{r,i}t_i(k), \tag{9}$$

$$\epsilon_i(k+1) = \frac{d_{\text{alt}}}{\tan-\alpha}, \tag{10}$$

$$D_{\text{alt,i}}(k+1) = d_{\text{alt}}, \tag{11}$$

while in this state the aircraft cruises at the maximum cruise velocity, at the expense of fuel and time. As the aircraft cruises ϵ decreases indicating that the distance to the next node is decreasing. **Jump Conditions:** The system may transition to the climb state and descend state, the condition under which there is a transition to climb is $D_{\text{alt,i}}(k) \leq d_{\text{alt}}$ and conditions for descend are $D_{\text{alt,i}}(k) \geq d_{\text{alt}}$ or $\epsilon_i(k) \leq \frac{d_{\text{alt}}}{\tan -\alpha}$, the latter condition indicates the required timing for descent to the next node. **Invariant Conditions:** The condition under which the system will remain in the cruise state is $\epsilon_i(k) > \frac{d_{\text{alt}}}{\tan -\alpha}$ and $D_{\text{alt,i}}(k) = d_{\text{alt}}$

Descend

The dynamics for the descend state are

$$t_i(k+1) = t_i(k) + \frac{\delta_i(k)}{V_{d,i}}, \tag{12}$$

$$\delta_i(k+1) = \delta_i(k) + \frac{d_{\text{alt}}}{\tan -\alpha}, \tag{13}$$

$$F_{l,i}(k+1) = F_{l,i}(k) - F_{r,i}t_i(k), \tag{14}$$

$$\epsilon_i(k+1) = 0, \tag{15}$$

$$D_{\text{alt,i}}(k+1) = 0, \tag{16}$$

$$S_i(k+1) = S_i(k) + 1 \tag{17}$$

while in this state the aircraft descends to the next way point for a pickup or drop-off as appropriate. **Jump Conditions:** The UAV will transition back to climb if $\epsilon_i(k) > \frac{d_{\text{alt}}}{\tan -\alpha}$ and $D_{\text{alt,i}} < d_{\text{alt}}$. Furthermore, the UAV will transition to unload if $\epsilon_i(k) = 0$. **Invariant Conditions:** The UAV will remain in descend if $\epsilon_i(k) \leq \frac{d_{\text{alt}}}{\tan -\alpha}$ or $\epsilon_i(k) > \frac{d_{\text{alt}}}{\tan -\alpha}$ and $D_{\text{alt,i}} > d_{\text{alt}}$.

Unload

While in the unload state the dynamics are

$$t_i(k+1) = t_i(k) + P_{H,j} \tag{18}$$

$$C_{p,i}(k+1) = C_{p,i}(k) + t_i(k)p_l \tag{19}$$

$$H_i(k+1) = H_i(k) - 1 \tag{20}$$

Jump Conditions: The system may jump from unload to load when $H_i(k+1) \neq H_i(k)$ indicating a package has been picked up or $\Gamma_i(S_i(k)) \in \Omega$ is not true, namely $\Gamma_i(S_i(k))$, which is the current node, is not in the set of delivery nodes. Furthermore, if $\epsilon_i(k) > 0$ we transition back to the descend state. **Invariant Conditions:** The UAV will remain in the unload state if $\Gamma_i(S_i(k)) \in \Omega$ and $H_i(k+1) = H_i(k)$. We note that the system is constrained to visit a pickup node for a specific package before visiting the unload node.

Load

While in the load state the dynamics are

$$t_i(k+1) = t_i(k) + P_{H,j} \tag{21}$$

$$H_i(k+1) = H_i(k) + 1 \tag{22}$$

Jump Conditions: The system may jump from load to refuel when $H_i(k+1) \neq H_i(k)$ indicating a package has been offloaded or or $\Gamma_i(S_i(k)) \in \Upsilon$ is not true, meaning there is no package to pickup. If $\Gamma_i(S_i(k)) \in \Omega$ we transition to the unload state.

Invariant Conditions: The UAV will remain in the unload state if $\Gamma_i(S_i(k)) \in \Upsilon$ and $H_i(k+1) = H_i(k)$.

Refuel

When in the refuel state the dynamics are

$$t_i(k+1) = t_i(k) + \frac{1}{R_{F,j}}(F_{l,i}(0) - F_{l,i}(k)) \tag{23}$$

$$F_{l,i}(k+1) = F_{l,i}(0) \tag{24}$$

Jump Conditions: The UAV will move to the climb state if $F_{l,i}(k) \geq (\frac{\epsilon_i(k)}{V_{c,i}} + 2\frac{\delta_i(k)}{V_{d,i}})F_{r,i}$, specifically, the UAV has enough fuel to reach the next node. **Invariant Conditions:** The UAV remains in the refuel state when $F_{l,i}(k+1) < F_{l,i}(0)$, specifically, it remains in the refuel node until the tank has been completely filled.

Performance Metrics

The set of transitions that a UAV must complete to deliver a set number of packages is stored in the vector Γ_i. Therefore, for each set of Γ_i there will be an associated time to complete all package deliveries and an associated accumulated priority cost.

In order to define a sense of optimal routing of the UAV's, a cost function must be established. We explore two possible cost functions that might be used to define an optimal package delivery route.

Total time

The first cost function is the total time to deliver all the packages. Namely, the total cost associated with $\Gamma_i \dots \Gamma_{N_{\text{UAV}}}$ is

$$C_{\text{Total}} = \max(t_i), \quad i = 1 \dots N_{\text{UAV}}. \tag{25}$$

where t_i is the time at which the i^{th} UAV, completes the set of transitions Γ_i. If $N_{\text{UAV}} = 1$ then

$$C_{\text{Total}} = t_i. \tag{26}$$

Simply put, assuming all UAV start at the same instant, then C_{Total} is the time at which the final package is delivered.

Priority Cost

The second cost function is also a function of time, but penalizes the selections of $\Gamma_i \dots \Gamma_{N_{\text{UAV}}}$ based on the priority number of a package p_l. Each UAV has an $C_{p,i}$, which is prorogated according to 19. Therefore, the total cost is

$$C_{\text{Total}} = \max(C_{p,i}), \quad i = 1 \dots N_{\text{UAV}}. \tag{27}$$

Therefore, the total cost is dependent on the UAV with the highest $C_{p,i}$.

Optimal Routing

With the hybrid automata defined in Section , and the definitions of cost, we attempt to determine a set $\Gamma_i \ldots \Gamma_{N_{\text{UAV}}}$ such that C_{Total} is minimized. One method for approximating these transitions, is through exhaustive search, brute force computations.

Using uniformly distributed random number we generated random sets of transitions based on a few constraints, namely, that the resulting set of transitions results in all packages successfully delivered. We store C_{Total} for each randomly generated set of transitions, we select the transitions which results in the lowest C_{Total}. Therefore as the number of randomly generated transitions approaches infinity, the probability that we will stumble on an optimal solution approaches 1. To demonstrate this theory, we generate N random Γ_i, for each N we store $\min \Gamma_i$. We repeat this procedure 10 times. Then we compute the standard deviation of the 10 $\min \Gamma_i$. As N increases the standard deviation decreases indicating that there is increasing confidence in the estimated optimal solution, Figure 2 demonstrates this concept.

Single UAV

We demonstrate the method by defining a single UAV with the following performance characteristics, $V_{c,1} = 700$, $V_{d,1} = 500$, $F_{r,1} = 50$, $F_{l,1}(0) = 500$, $\Gamma_1(0) = 29$. We wish to pickup packages at 26, 22, 35 19 and deliver them to 40, 5 ,13,12 respectively. Furthermore, $P_{H,j}$ and $R_{F,j}$, which are the handling and refuel rates, respectively, are the same for all nodes. The priority for each package is also the same.

Figure 3 is the resultant optimal route for minimizing the total time required to deliver the specified packages. We note that there are a number of different routes which result in the same minimum time obtained for this route. This suggests that the optimal solution is not unique.

Figure 4 demonstrates the change in the optimal routing when $P_{H,j}$ and $R_{F,j}$ are no longer uniform but are randomized. We note that when the refuel and handling times are not uniform the number of optimal routes decreases significantly.

Two UAV's

We now introduce another aircraft into the simulation such that the performance characteristics of the two UAV's are as follows, for UAV 1 $V_{c,1} = 700$, $V_{d,1} = 500$, $F_{r,1} = 50$,

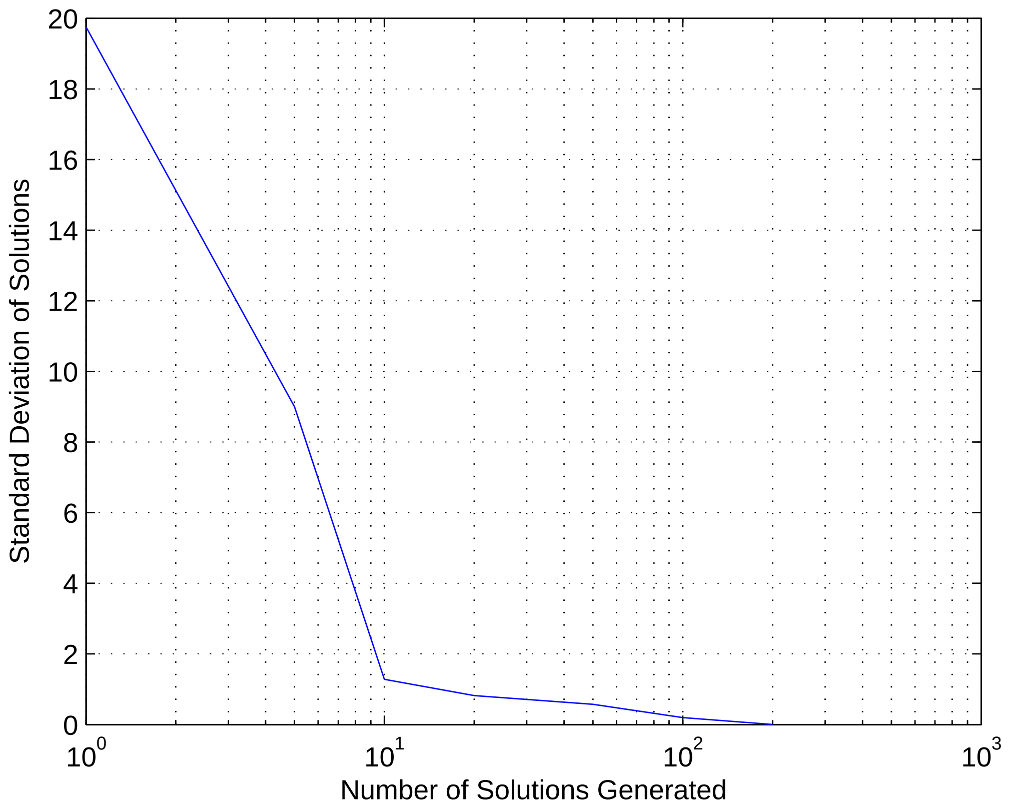

Figure 2: The brute force search generates a finite number of possible solutions, as the the number of possible solutions increases the likelihood that we have found an optimal solution increases. The Y axis is the standard deviation of 10 brute force computations for each finite number of possible solutions. As the standard deviation approaches zero, we gain confidence that we have found an optimal solution.

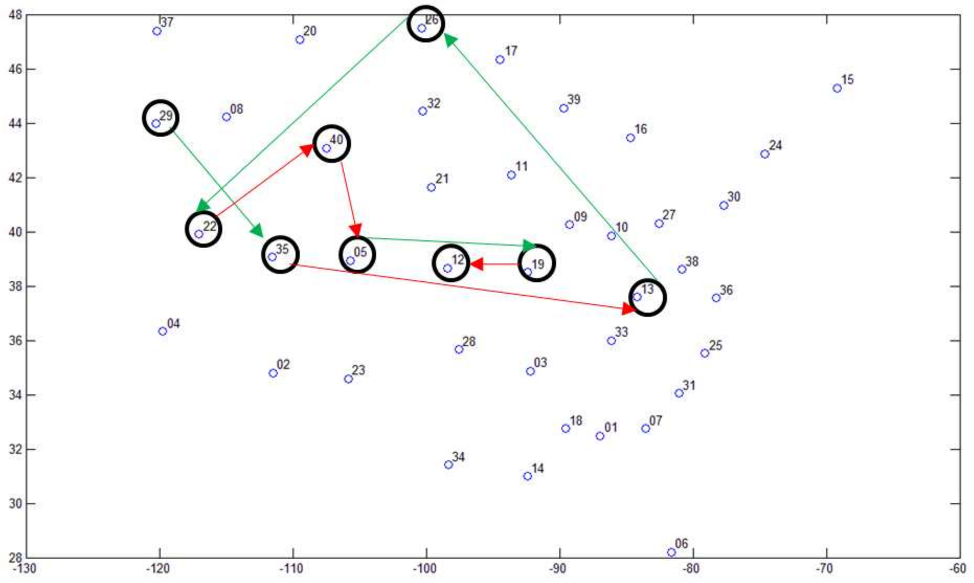

Figure 3: An optimal route to deliver packages is outlined, green arrows represent pickup runs, and red arrows represent delivery runs. In this example, all nodes have the same refueling and package handling rates.

$F_{l,1}(0) = 500$, $\Gamma_1(0) = 29$ and UAV 2 $V_{c,2} = 600$, $V_{d,2} = 600$, $F_{r,1} = 60$, $F_{l,2}(0) = 600$, $\Gamma_1(0) = 15$. Note that the second UAV has a slower cruise speed, but faster climb and decent speed, it also has a larger fuel tank, but a higher fuel consumption rate.

We again assume the refuel and handling times are uniform, and simulate the system to obtain the minimum time solution. Figure 5 shows the paths of both UAV's.

We again note that the optimal route is not unique.

The final case we analyze is optimization of priority cost. The packages are assigned priorities 50, 10, 50, 3, respectively. Noting that packages 1 and 3 have high priority numbers, we would expect that these would be picked up and delivered first. Figure 6 shows the routes of the two aircraft, and indeed the solution calls for packages 1 and 3 to be picked up and delivered first in order to minimize the priority cost. Furthermore, we note that 1 and 3 are picked up and delivered by separate UAV's, which means both high cost packages are delivered simultaneously.

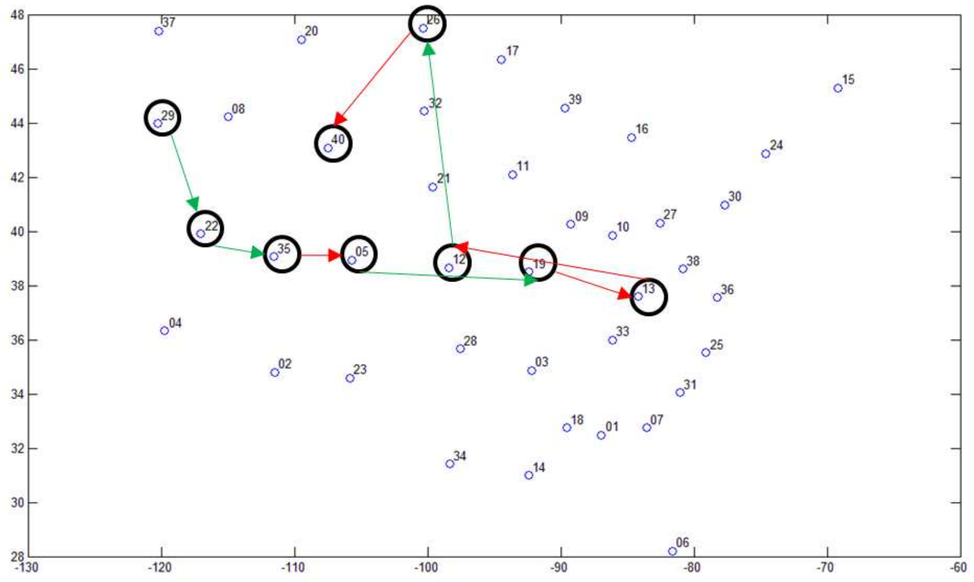

Figure 4: An optimal route to deliver packages is outlined, green arrows represent pickup runs, and red arrows represent delivery runs. In this example, nodes have randomized refueling and package handling times.

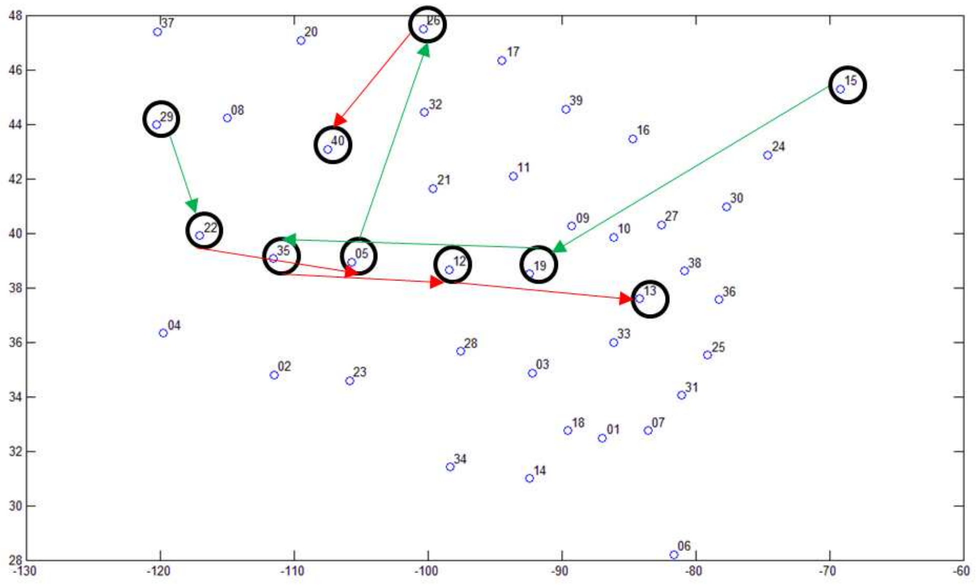

Figure 5: An optimal route to deliver packages is outlined, green arrows represent pickup runs, and red arrows represent delivery runs. In this example, UAV 1 starts at node 29 and UAV2 at 15. All refueling and package handling times are uniform.

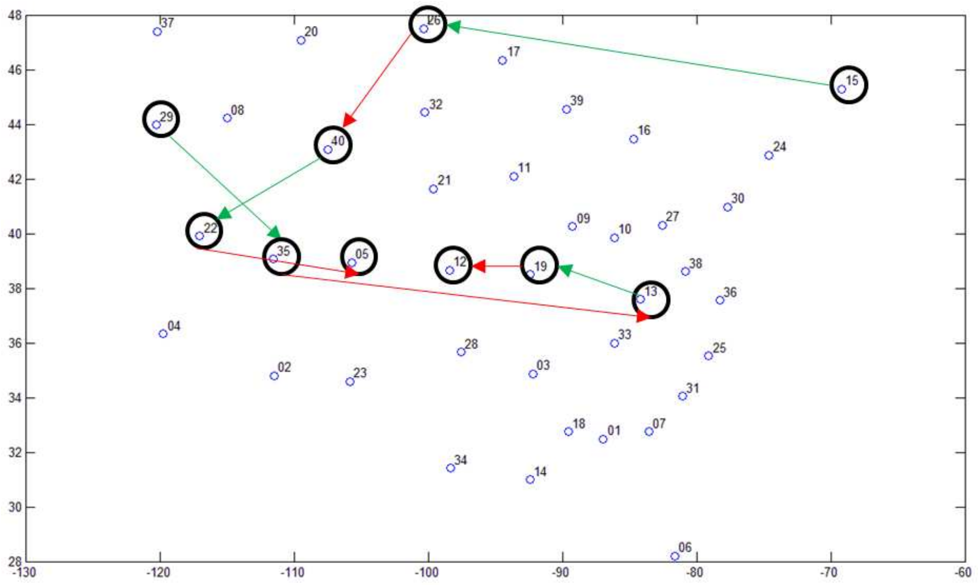

Figure 6: An optimal route to deliver packages is outlined, green arrows represent pickup runs, and red arrows represent delivery runs. In this example, UAV 1 starts at node 29 and UAV2 at 15. All refueling and package handling times are uniform. However, the priority rating of each package is different.

Search and Planning

We now discuss the search and planning aspects of the project. In the previous sections solutions were obtained for scenarios when all environmental variables were known, such as refuel times, package handling times, etc. We now examine search methods that utilize limited amounts of dats, such as distance between nodes.

The search and planning part of this project was completed in three main stages. These stages are briefly described in this introduction and are further discussed in self-titled subsections.

- The first stage consisted of coding the most basic structures and functions needed for our project (for example, coding the map and package structures, writing the package-delivery scheduler and incorporating search algorithms, etc.).
- The second stage involved having to change the grid to the actual map of the US and allowing for ”optimum” solutions.
- The third involved upgrading to 2 UAVs on the US map.

Starting Simple

Before going into details, it is important to talk about the general direction in which the search and Planning part of this project was approached. It was decided to use the search software that was developed in the last homework and possibly change and augment it as needed. First order of business was to decide if this software was to be kept in C or be converted over to MATLAB. Since we wanted to produce lots of plots of our simulations, MATLAB looked very tempting while plotting in C seemed like a possible, but rather time consuming solution. So it was decided upon converting C code over to MATLAB. It was discovered that this conversion process is not as straightforward as one would think, and indeed was abandoned after hours of fruitless work.

The next feasible solution that came up was to take the best of two worlds (speed of C and graphical abilities of MATLAB). The C code would stay in C, but on MATLAB side it would be augmented with ’bookkeeping and plotting attachments.’ As far as interfacing the two goes, MATLAB would write a text file and call the executable (compiled from C-code) on that text file. The executable, in turn, would do all the search-related manipulations and

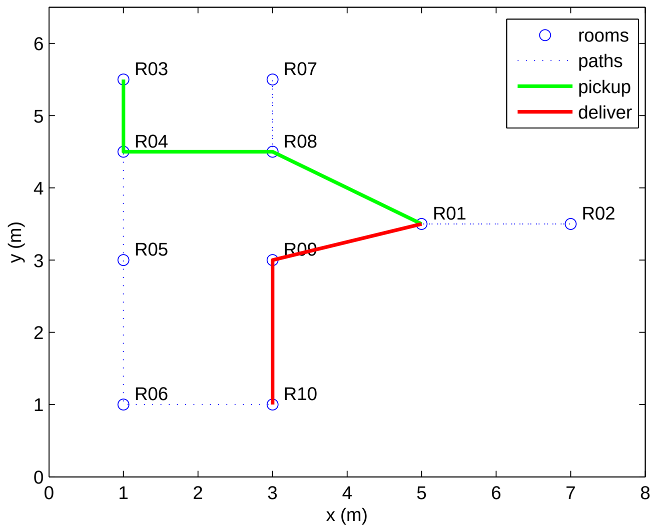

Figure 7: The rooms example. PSA (or UAV) starts in room 3, picks up package in room 1, and delivers it to room 10.

return the search results in another text file to MATLAB. At that point MATLAB could use these results to schedule package delivery, plot trajectories and so on.

After some time, the iterative process of coding and troubleshooting yielded a working simulation on a simple grid with a single package. Let's take a look at the general structure of this implementation. The grid was based on the last homework (problem about the PSA[1]). We assumed that PSA had to move perpendicular to the air-lock door while traveling between rooms. Accordingly, the costs of traveling from one room to another one were calculated using the Manhattan-block distance concept. However, the heuristic for the A-star and Greedy search methods was picked to be the straight line distance. This heuristic is admissible because it is always optimistic (the true cost the PSA will inquire between rooms is Manhattan-block distance, which is greater or equal to the straight line distance). A sample pick up and delivery process is shown in Figure 7. The PSA starts in room 3, but there is a package to be picked up in room 1 and delivered to room 10.

While this result looks rather simple, the code that produces this example can be rather confusing at times. However, it includes lots of comments as shown in Appendix 1 (files rooms.m (needs some other small files) and mysearch.c (needs map_us.h)). This code utilizes a very particular package-delivery strategy, which is important to mention here. What it does is just calculate a cumulative cost (both to pick up and to deliver) for each package and

[1]http://psa.arc.nasa.gov/

pick the cheapest one. Then it flies over to pick up the package and then, to deliver it. At this point, it recalculates the costs for the rest of the packages from the current position and picks the cheapest package again, and so on until all packages are delivered. This strategy is very simple to implement, but at the same time, it is easy to show that it is very wasteful (for example, it doesn't pick up or deliver packages en-route to other packages). This strategy can be thought of being greedy and in some senses it is similar to the greedy search method. Nonetheless, this strategy was a starting point and by building on this simple case our team was able to produce better scheduling algorithms shown in the next sections.

Maps and Searching for Optimal Solutions

At this stage in this project, the US map was incorporated into the simulations. Several simplifying assumptions were made: only geographical centers of the states were included, only contingent states were included, and finally only the states whose centers were 'sufficiently far apart' were included. These assumptions don't have any logical objective besides simplifying the mapping process and decreasing the computational intensity. Once the US map was programmed in, the greedy strategy, described in the previous section, was tested in a setting with four packages (shown in Figure 8). As it can be seen, this strategy performs just as it was expected to - it doesn't pick up packages optimally, but at least it works.

So the next step was to design a strategy that would do a little bit better job. The resulting algorithm operates somewhat similarly: it calculates the costs to pick up or deliver (previously picked-up) packages from current location and picks the cheapest action. That way, the algorithm doesn't 'lock up' on one package and indeed can pick up/deliver packages as it goes. This strategy is shown in Figure 9. The performance of this algorithm was tested in a set of circumstances and was found to perform better then the greedy one (in terms of distance-cost inquired while performing the pick-ups and delivery). However, it can be shown that in certain settings even this algorithm fails to give 'completely optimum' solutions (just as greedy search can return suboptimal solutions). However, some incite was gained in the nature of the problem. (This and the greedy strategy, implemented in the MATLAB-code, are shown in Appendix 2 (file main.m).)

While working on this project it was also discovered that the process both strategies were trying to implement was very similar to the search process (for example A*). However, the difference is that the goal of this problem is to deliver 4 packages and they can't be delivered until they are picked up. So, in comparison to the simple navigation problem, where there is only one destination, here one has $2n$ destinations (n = number of packages,

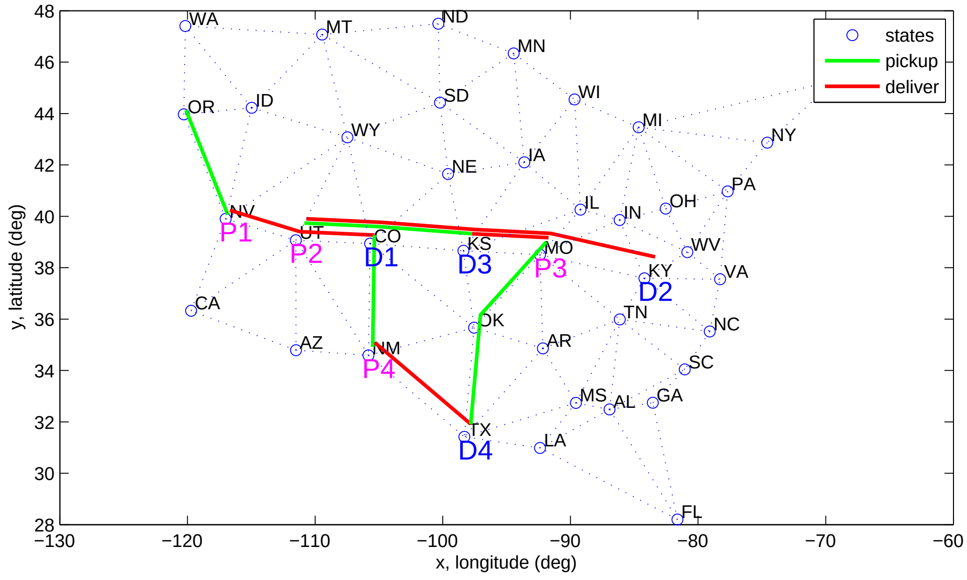

Figure 8: The greedy strategy example. UAV starts in WA, package pick up locations are labeled with P# and delivery locations with D#.

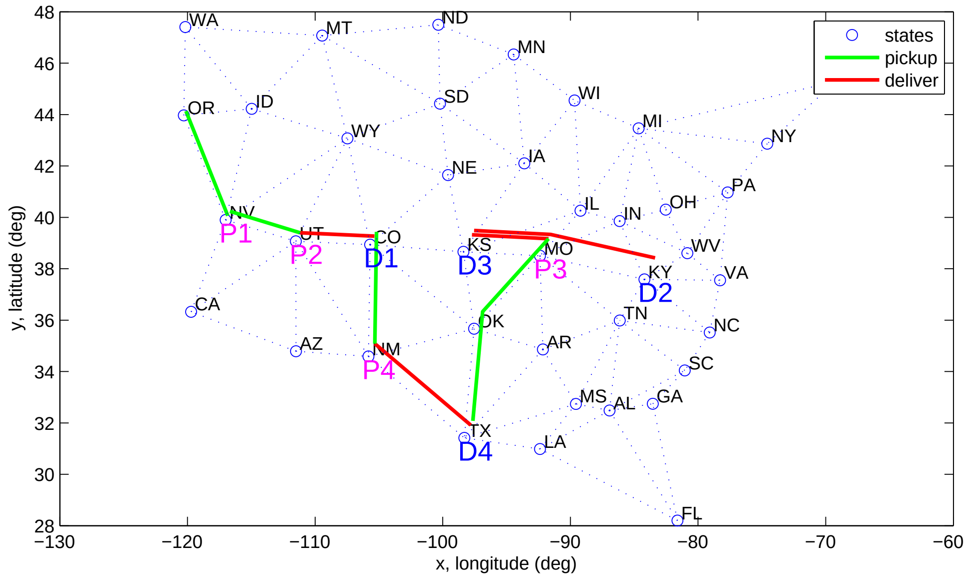

Figure 9: The improved strategy example. UAV starts in WA.

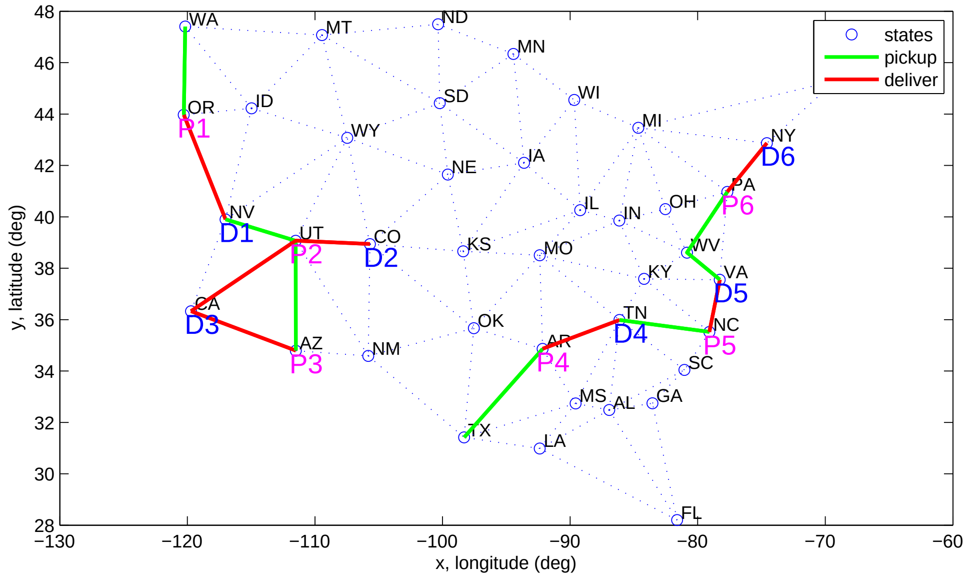

Figure 10: The multiple UAV's strategy example. UAV1 starts in WA, UAV2 starts in TX.

it's $2n$ if no packages are collocated) and the order you can reach them is subject to certain restrictions. Accordingly, it was attempted to formulate this problem in more general way (using tree structure) and define an admissible heuristic, but these attempts didn't result in working solution in time allotted. However, the upgraded greedy strategy was extended to working on two UAV's, as described in the following section.

Multiple UAV Systems

One solution that was considered for this multi-UAV problem was just to take the code that worked with one UAV and run it multiple times. However, simulations of that solution quickly unearthed certain interference issues. Nonetheless, after these issues were dealt with, this solution yielded an intermediate result that looks rather convincing (Figure 10). Here, the package locations were selected rather carefully because if one UAV picks up a package right now, the second UAV can deliver it, because being picked-up is a property of the package and currently has nothing to do with the UAV number. However, if there were more time allowed for the project this issue would be dealt with properly.

The last notion worth mentioning here is that of optimality. So far all discussion of optimality involved optimizing over distance. However, real life is more complicated than that. In real life there is wind that can help UAV get to some point faster or instead slower;

there are airports that have really slow fuel pumps which makes refueling slower at those nodes and so on. In this section we assumed that the communication hub (be that satellite or some people in some central ground location) is not operational. Therefore we have no knowledge of particular characteristics of the nodes and accordingly, we can't optimize over time. The best we can do is optimize over the distance (assuming map is known to every UAV). However, the previous section discussed the various concepts that allow this project to account for 'real life' constraints.

All the code written for this section is available online at this website.[2] Also, see Appendix 1 for a printout of the main files for this project.

Communication

The most essential component of our delivery system is to establish reliable communication between our UAV's and Satellite. In this section, we analyze how under different scenarios and constraints we can improve the performance metric of our system. First we briefly address the components involved the communication system. We use a geostationary communication satellite stationed in space for the purpose of receiving requests and transmitting the optimal route and decision for each UAV. Satellite commutation incorporates a delay of the order of 300 ms each way but for the end user it is not of much importance. Furthermore, we model a fixed known latency for all the communication links associated with the Satellite. Next we have a server, which can be on the ground or incorporated in the satellite itself. We install our algorithm for computing optimal routes and other decision making parameters in the server. The difference between the two cases namely servers on the ground or on the satellite itself is a matter of fixed latency which can be neglected in comparison to the request rate (λ) and the delivery rate. Hence, we analyze the case when server is installed in the satellite itself. Next, we have UAV's which we have assumed that cannot communicate with each other, and they can only communicate with the Satellite. Scenario when satellite is not visible is not addressed, as the decision making process is left to UAV's with no appropriate way to communicate the package request unless every node/station broadcasts whenever they receive a package and UAV's that are locally accessible to these stations can receive the requests for package delivery, which in this case won't be centralized. We have not implemented this scenario in our search and planning section. The diagram below shows the basic components of our system and how they are linked.

[2]umich.edu/~morozova/um/AE740/FP_Final_Set.zip

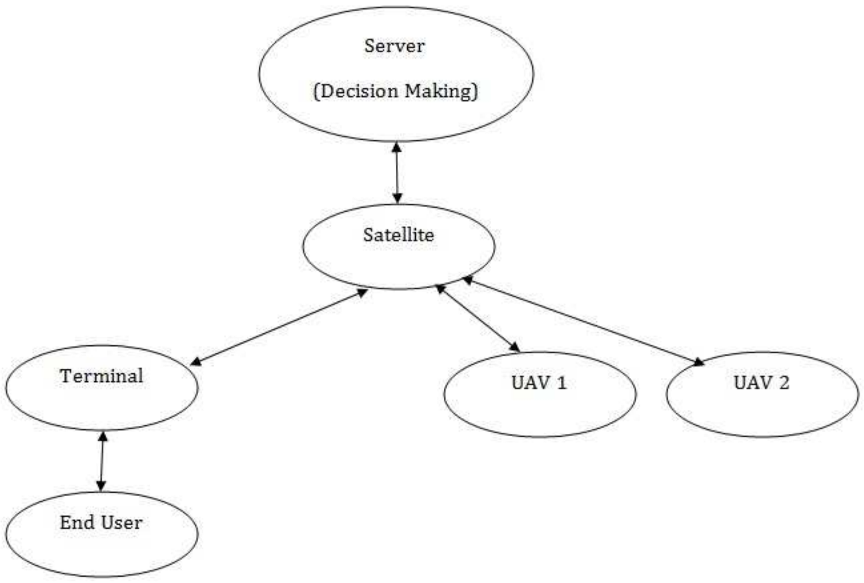

Figure 11: The communication links above restricts our UAV's to communicate with each other or the Base Stations.

Next we address the issue of what data needs to be sent on each of these links. Firstly, all the communication links modelled as Gaussian random processes. Since all the information we deal with is digital, it converts to a binary symmetric channel in our digital framework. The simplest communication link of receiving request from the end user consists of 3 basis parameters (Pickup location, Delivery Location, Priority). The priority parameters takes into account the sensitivity of the item that has being delivered. Next the satellite forwards the request to the server and server computes the optimal route decisions back to the satellite to broadcast. Satellite communication is assumed to be in broadcast mode of operation. Next we assume that while the satellite is computing the optimal routes, it will not process any feedback or communication from the Satellite and all the information will be stored in a queue. When it comes to satellite communication with UAV's we consider a latency of 300 ms each way to measure the process time and the delivery confirmation timing. Upon receiving decision from the server, satellite broadcasts the information to all the UAV's and the UAV its meant for the one that interprets its meant for that UAV, sends an acknowledgement back to the Satellite along with the message its has received. Then Satellite confirms the data and broadcasts it again to all the UAV's. This keeps on going until they agree on exactly one decision message. Once the decision is confirmed, the UAV in action transmits a final acknowledgement to the Satellite to inform the server that the request has being received. This takes approximately 2 - 10 seconds, since we don't proceed unless the both match on the same message, the probability of error in communicating the precise message comes out to be of the order of 10^{-8} with the SNR of 40dB. The plot below shows how probability of error varies with SNR.

Now, we consider the following different scenarios and constraints associated with our satellite and UAV's capabilities

1. All the package requests are available at t=0 and an optimal decision is made by our server in a kind of "offline" mode.

2. Package request arrival is a random process and is modelled as Poisson Process with rate λ.

Case 1:

Since all the communication links are modelled as Gaussian random processes, in our digital framework, our channel error characteristic converts it to a binary symmetric channel.

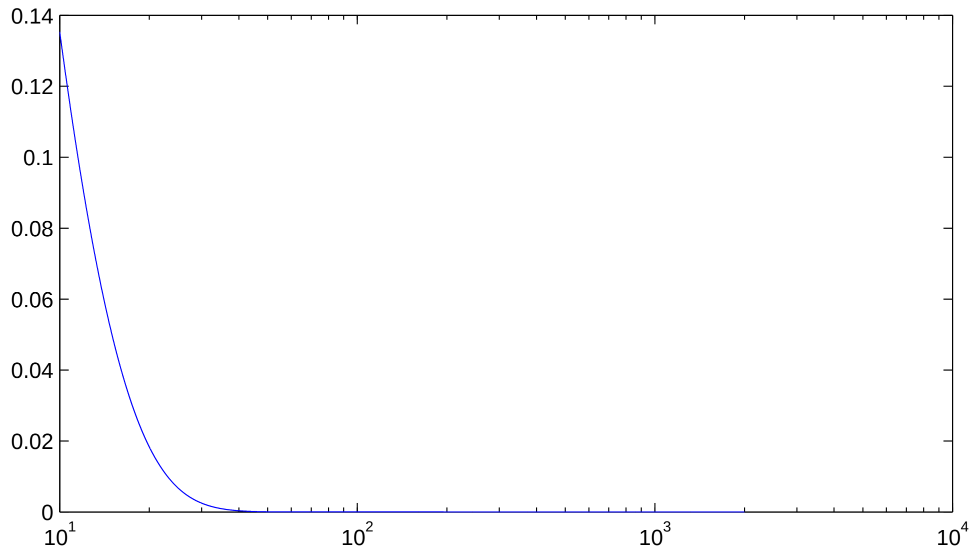

Figure 12:

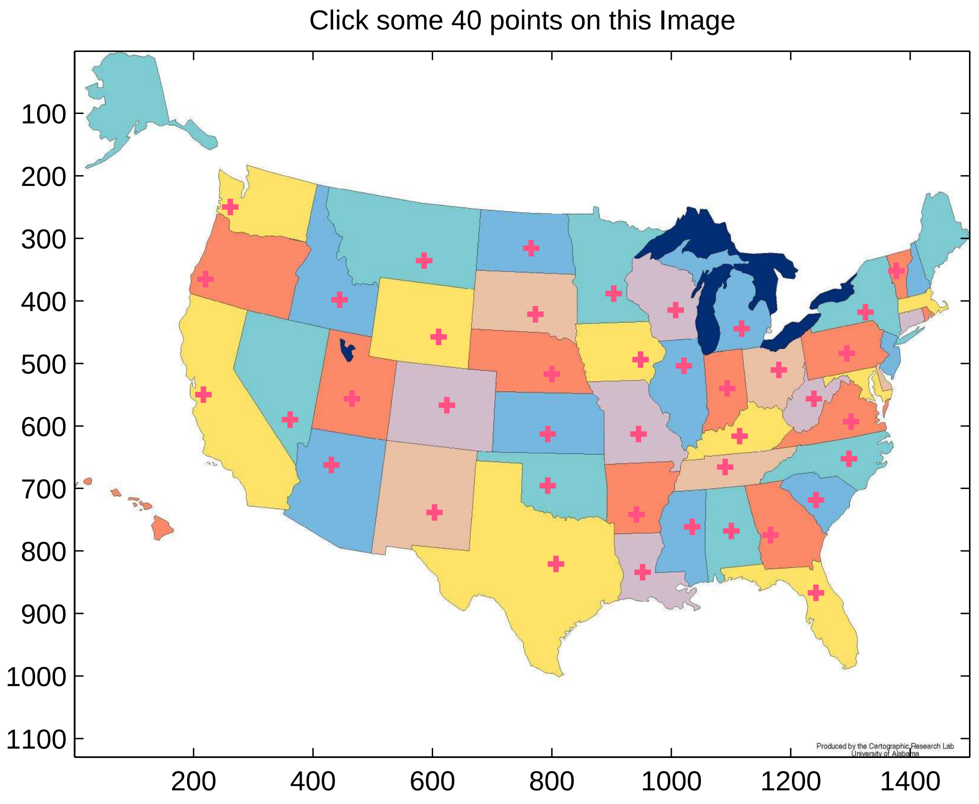

Figure 13:

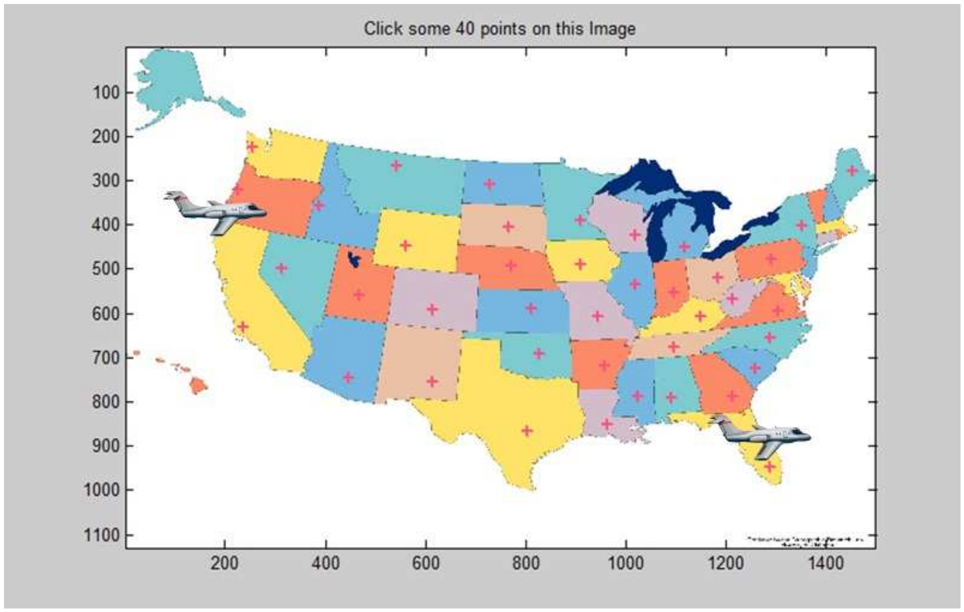

Figure 14:

We dealt with a lot of error correcting codes ranging from LDPC to Turbo Coding and after considering an essential tradeoff between the simplicity of the algorithm and efficient communication, we decided to use veterbi algorithm for our communication. Below is the probability of error Vs SNR plot. Note that the probability of error is computed after both messages are matched. Hence, the error is of the order of 10-9.

Case 2:

In this section, we consider that the request are not all known prior to making a decision, but are modelled as a Poisson random process. The difficulty in this scenario is not about communication but being able to make an appropriate decision at the server end and being able to process the regular feedback of location (GPS) from the UAV's to incorporate in decision making. The diagram below shows a particular realization of the timing of arrivals of these requests. This issue is not completely addressed in our search and planning section. Nevertheless, we model this scenario as being tough to maintain communication, and we use the same rate as we use in the previous section. Since, the communication channel is worse in this case, the error probabilities are a bit higher.

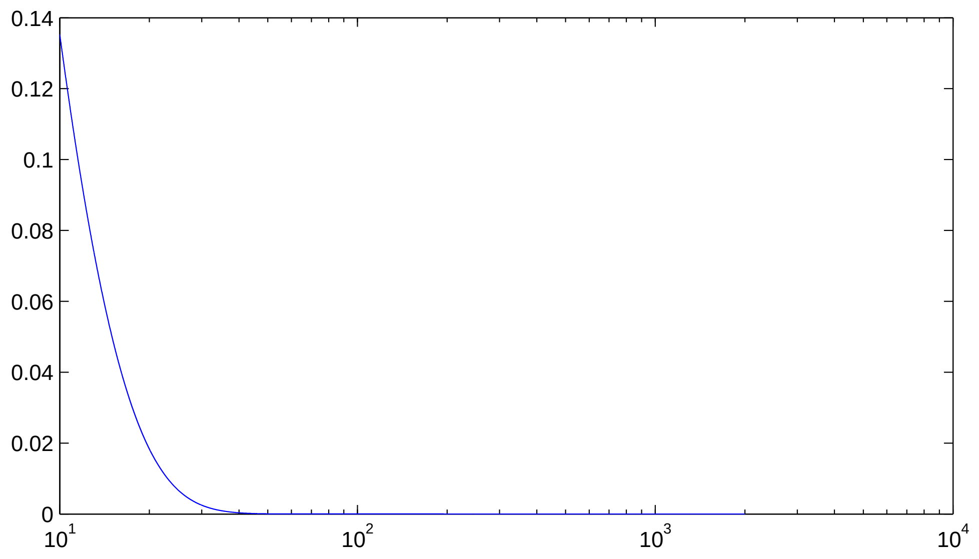

Figure 15:

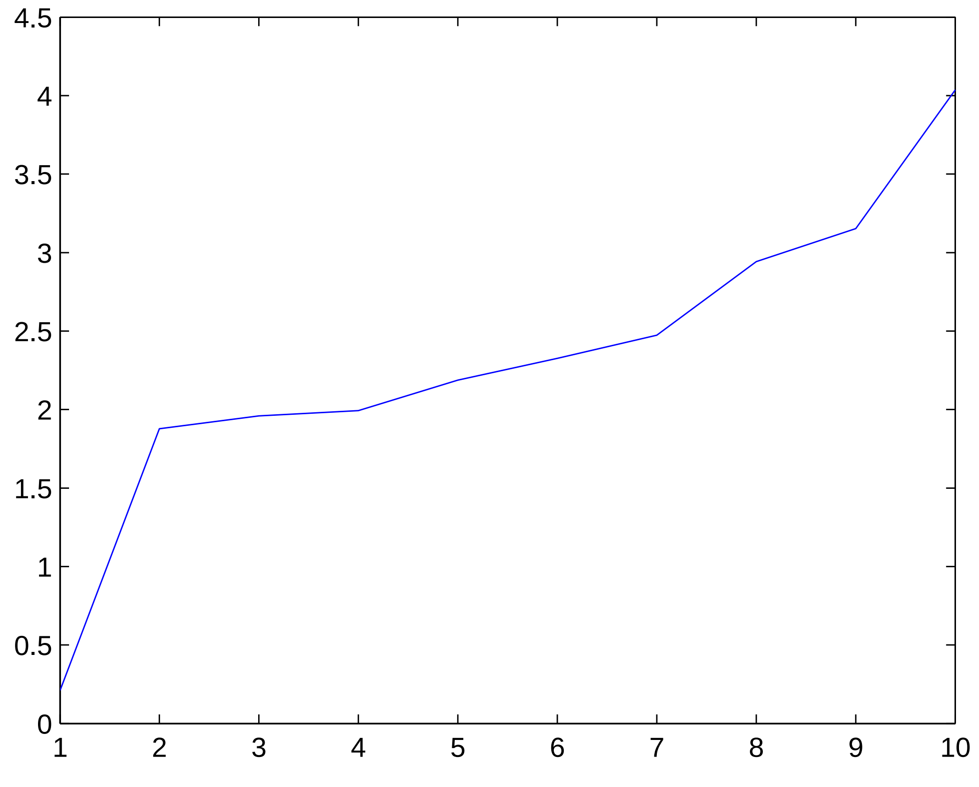

Figure 16:

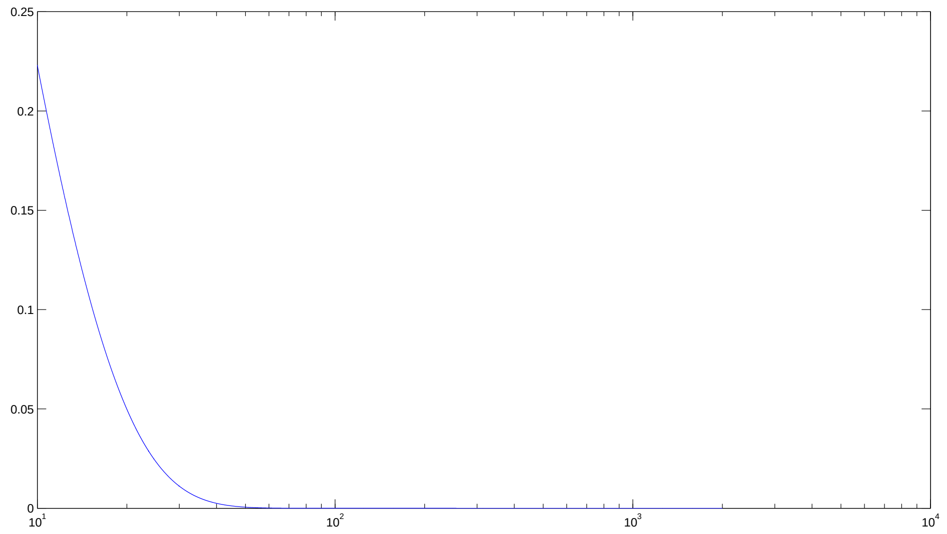

Figure 17:

Conclusion

In conclusion we demonstrated concepts to select routes for UAV's to pickup packages at one location and deliver them to another. We examined exhaustive search methods based on the hybrid automata outlined for the system. Secondly we discussed and demonstrated methods for search and planning when not all of the system variables are known.