**Arianna Bettin – K1763730, Hani Tawil - K1533307**

The plan scenario annotated below, characterizes a model of the upcoming drone delivery amazon is trying to implement for the near future. In this model we govern the logistics of how the drone picks up and delivers (by flight) the package whilst also sustaining a healthy amount of charge to function. We specialize the model further to also take into account the transportation of goods from the warehouse in which the products are stored to the drone-friendly post offices.

## PDDL+

When writing and designing the domain for our problem we wanted to use the PDDL+ features that Dan introduced us to in Lecture 8. He introduced us to specialized actions such as processes and events which essentially acted as actions that would be run as soon as their pre-conditions were met and were especially useful in a hybrid system. In our use case we wanted to use a process action to continuously change the charge of the drone as soon as it is put on charge. Then call an event action to stop the charging as soon as the charge-level reaches 100. In the model the putting and taking off charge are the discrete changes in the world and the charge level increasing would be the continuous change triggered by the discrete change.

However, when trying to run our PDDL code using popf1.1 we realized that the planner was having troubles trying to understand and compile it. We then realized that the features and uses provided by PDDL+ could only be run by a select few planners and those planners could not be run along side ROSPlan and the XIAP interface to be able to portray our work through a eXplainable AI context. We decided to keep the PDDL+ features below in the appendix to show how we would have made use of the advantages that they bring to the domain. Nevertheless, for the second portion of the coursework we collapsed the uses of the process and event into one “RECHARGE-DRONE” durative-action so that the code would be compatible with the ROSPlan interface.

## What is Explainable AI

The growth of Artificial intelligence is increasing rapidly, as machine learning and other AI related algorithms are being used in almost all industries the need for those AI bodies to explain why and how they got to the conclusion they output is all the more necessary. At the moment the use of machine learning algorithms provide a very opaque and un-interactive experience to the users of it. We are expected to blindly trust and accept the answer these AI partners give without any explanation or knowledge as to how and why it got to that answer.

Now with the expansion of Artificial intelligence to higher risk industries such as the oil and gas and health sectors. Now more than ever, we need to be able to justify why the AI algorithms used have given the answer it did. The goal of eXplainable AI is not to just answer back with the obvious answers “because it is a cheaper/faster solution” but to explain the state the AI is in and the conditions/parameters given to it to explain how it drew its conclusion.

## Explainable Planning

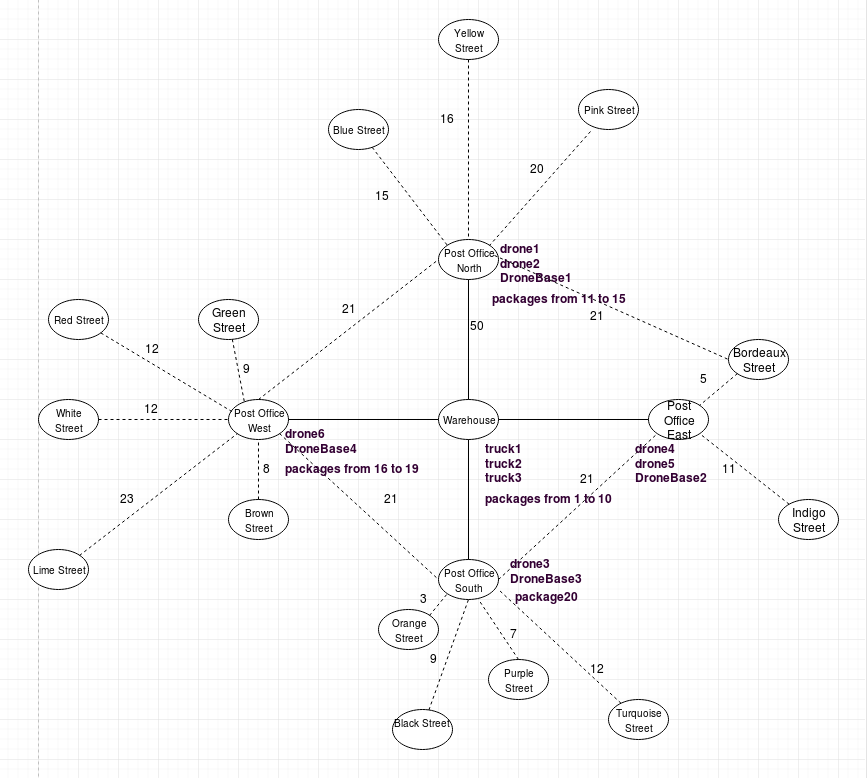
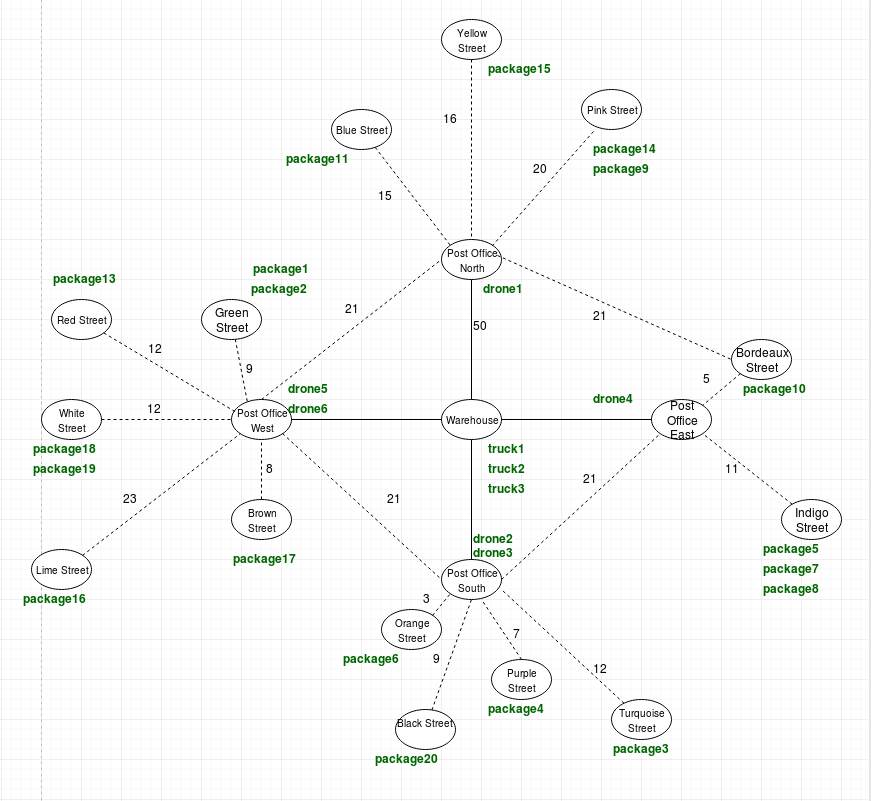
Explainable planning already provides a lot of the foundations for explaining the reasons and providing a certain level of transparency to the user as to how it got to the solution it produced. A lot of this achieved through the execution trace of the plan generated by the planner as this can deliver some sort of causal links between how and why the certain steps were taken in the order they are proposed. Through this trace it exposes to the user a level of computation that a lot of other Artificial intelligent agents do not produce, offering a more transparent experience for the user.

Now with the use of ROSPlan and the XAIPInterface, users are able to ask important explainable AI questions such as:

* Why did/didn’t you do that?
* Why is taking this action more cheaper/efficient than this action
* Why am I not able to this?
* Why do/don’t I need to re-plan at this step

This interface allows users to view the original generated plan then select a step within to suggest a alternative action that can be executed at that time step instead. With this we are able to test whether or not the planner can still complete the problem with the new “forced” action and if so whether or not the new plan generated is more optimal or not since the use of planners does not guarantee optimality for more complex problems.

## Problem file 2 Schemas

**Initial state: Goal state:**

COMPARE THE DIFFERENCES BETWEEN THE 2 PROBLEM FILES AND THE CHALLENGES THEY GIVE:

* Number of packages 5 to 10
* Some initiate in the post office instead of the warehouse
* Some packages in the wrong post office
* Number of post offices

The model provides a good foundation to emphasize the importance of explainable AI by applying many simple constraints which become more and more complex as the problem world grows in size. These constraints include:

* Limited size available in trucks
* Charge management of drones
* Problems where there are more drones than charging bases
* Specific routes that only certain vehicles can take
* A segregation of the map so that certain post offices are more efficient to fly to certain locations.

With these questions we can ask the planner why it choose to make specific moves to try and optimize the world around the constraints listed above.

The types identified and defined in the domain are as follows:

* A generic type, **"object",** and its instances, "**place"**, and "**locatable",** instances of which can be associated with a "location" type.
* The **“drone”** (instance of **“vehicle” and “locatable”**) – The machine which will transport packages from the post office to the destination.
* The **“truck”** (instance of **“vehicle” and “locatable”**) – Vehicle which will transport goods only from the warehouse to either post office.
* The **“item”** (instance of **“locatable”**) – The object the drone will carry to its destination
* The **“vehicle”** (instance of **“locatable”**) – The object which encapsulates the drone and truck
* **“droneBase”** is also given a type so that we are able to distinguish between the drop-off locations and the locations in which the drone need to return to and recharge at.

The predicates identified and defined in the domain are:

* **“(at ?item - locatable ?loc - place)”** – this helps represent the locations of a specific object
* **“(available ?d - drone)”** – this shows if a drone is available for transport
* **“(charging ?d – drone)”** - to show that the drone is charging
* **“(in ?item – locatable ?v – vehicle)”** – checks to see what package a specific drone or truck is carrying
* **“(air-link ?x ?y - place)”** – used to recognize a path which only a drone is able to take.
* **“(link ?x ?y - place)”** – used to recognize a path which only a truck is able to take.
* **“(drone-base-available ?db - droneBase)”** – True when no drone is using the base for charging since only one drone can be charged at a time.

The functions identified and defined in the domain are as follows:

* **“(charge-level ?d – drone)”** – the percentage of charge left in the drone
* **“(charge-required ?x ? y – place**)” – the amount of charge required to reach a location.
* **“(flight-time ?x ?y – place)”** – the time it takes to move from one location to another via drone
* **“(drive-time ?x ?y – place)”** – the time it takes to move from one location to another via truck
* **“(loaded-packages ?t - truck)”** – used to track the amount of packages a truck is currently holding

The durative actions that are defined in the domains are:

* **“ATTACH-PACKAGE”** – with “item”, “place” and “drone” as the parameters and the duration of 10 units. To accomplish this action the drone needs to available, both the drone and the package need to be at the same location. This action will make the drone occupied and change the location of the package to the drones location.
* **“RELEASE PACKAGE”** – with the same parameters as “attach package” but with opposite effects.
* **“FLY-DRONES”** - with "drone” and two "location" as parameters. The duration is equal to the "flight-time" function associated with the two "location" points. In short, at the end of the duration, the "location" of the "drone" will change to that of the target "location". The drone needs to make sure that it has enough charge to get to the location and back. It will not take the trip if it cannot make it.
* **FLY-DRONEBASE**” – With the same parameters as “fly-drone” but instead also includes the dronebase object to make sure its going back to the right location as well as calculating the charge required correctly.
* **“RECHARGE-DRONE”** – with “drone”, “place” and “base” as parameters. The duration is equal to 100 minus “charge-level ?d”, therefore the charge rate is 1% for every unit of time. The “drone” must be at the “droneBase” to be recharged and at the end of the duration its charge level will be equal to 100%.
* **“LOAD-TRUCK”** – with “item”, “place” and “truck” as parameters. The duration is equal to 10 units. The “truck” and “item” need to be in the same location and the truck must not have more than 10 packages in it. By the end of the action the item is in the truck and the load of the truck has increased by one
* **“UNLOAD-TRUCK”** – with the same parameters as “load truck” but has the opposite effects.

# PDDL Domain file.

*(define (domain Delivery)*

*(:requirements :strips :typing :time :numeric-fluents :durative-actions :conditional-effects)*

*(:types*

*place locatable - object*

*vehicle truck drone item dronebase - locatable*

*drone truck - vehicle*

*)*

*(:predicates*

*(at ?item - locatable ?loc - place)*

*(in ?item - locatable ?v - vehicle)*

*(available ?d - drone)*

*(link ?x ?y - place)*

*(air-link ?x ?y - place)*

*(charging ?d - drone)*

*(drone-base-available ?db - dronebase)*

*)*

*(:functions*

*(drive-time ?x ?y - place)*

*(flight-time ?x ?y - place)*

*(charge-required ?x ?y - place)*

*(charge-level ?d - drone)*

*(loaded-packages ?t - truck)*

*)*

*(:durative-action LOAD-TRUCK*

*:parameters*

*(?item - item ?truck - truck ?loc - place)*

*:duration (= ?duration 10)*

*:condition*

*(and (over all (at ?truck ?loc))*

*(at start (at ?item ?loc))*

*(at start (< (loaded-packages ?truck) 10)))*

*:effect*

*(and (at start (not (at ?item ?loc)))*

*(at end (in ?item ?truck))*

*(at end (increase (loaded-packages ?truck) 1)))*

*)*

*(:durative-action UNLOAD-TRUCK*

*:parameters*

*(?item - item ?truck - truck ?loc - place)*

*:duration (= ?duration 10)*

*:condition*

*(and (over all (at ?truck ?loc))*

*(at start (in ?item ?truck))*

*(at start (> (loaded-packages ?truck) 0)))*

*:effect*

*(and (at start (not (in ?item ?truck)))*

*(at end (at ?item ?loc))*

*(at end (decrease (loaded-packages ?truck) 1)))*

*)*

*(:durative-action ATTACH-PACKAGE*

*:parameters*

*(?item - item ?drone - drone ?loc - place)*

*:duration (= ?duration 10)*

*:condition*

*(and (over all (at ?drone ?loc))*

*(at start (at ?item ?loc))*

*(at start (available ?drone)))*

*:effect*

*(and (at start (not (at ?item ?loc)))*

*(at end (in ?item ?drone))*

*(at start (not (available ?drone))))*

*)*

*(:durative-action RELEASE-PACKAGE*

*:parameters*

*(?item - item ?drone - drone ?loc - place)*

*:duration (= ?duration 10)*

*:condition*

*(and (over all (at ?drone ?loc))*

*(at start (in ?item ?drone)))*

*:effect*

*(and (at start (not (in ?item ?drone)))*

*(at end (at ?item ?loc))*

*(at end (available ?drone)))*

*)*

*(:durative-action DRIVE-TRUCK*

*:parameters*

*(?truck - truck ?loc-from - place ?loc-to - place)*

*:duration (= ?duration (drive-time ?loc-from ?loc-to))*

*:condition*

*(and (at start (at ?truck ?loc-from))*

*(over all (link ?loc-from ?loc-to)))*

*:effect*

*(and (at start (not (at ?truck ?loc-from)))*

*(at end (at ?truck ?loc-to)))*

*)*

*(:durative-action FLY-DRONE*

*:parameters*

*(?d - drone ?loc-from ?loc-to - place)*

*:duration (= ?duration (flight-time ?loc-from ?loc-to))*

*:condition*

*(and (at start (at ?d ?loc-from))*

*(over all (air-link ?loc-from ?loc-to))*

*(at start (> (charge-level ?d) (\*2 (charge-required ?loc-from ?loc-to)))))*

*:effect*

*(and (at start (not (at ?d ?loc-from)))*

*(at end (at ?d ?loc-to)))*

*)*

*(:durative-action FLY-DRONEBASE*

*:parameters*

*(?d - drone ?loc-from ?loc-to - place ?db - dronebase)*

*:duration (= ?duration (flight-time ?loc-from ?loc-to))*

*:condition*

*(and (at start (at ?d ?loc-from))*

*(over all (at ?db ?loc-to))*

*(over all (air-link ?loc-from ?loc-to))*

*(at start (> (charge-level ?d) (charge-required ?loc-from ?loc-to))))*

*:effect*

*(and (at start (not (at ?d ?loc-from)))*

*(at end (at ?d ?loc-to)))*

*)*

*(:durative-action RECHARGE-DRONE*

*:parameters*

*(?d - drone ?l - place ?db - droneBase)*

*:duration (= ?duration (- 100 (charge-level ?d))*

*:precondition*

*(and (at start (< (charge-level ?d) 100))*

*(at start (available ?d))*

*(at start (drone-base-available ?db))*

*(over all (at ?d ?l))*

*(over all (at ?db ?l)))*

*:effect*

*(and (at start (not (available ?d)))*

*(at start (not (drone-base-available ?db)))*

*(at start (charging ?d))*

*(at end (= (charge-level ?d) 100)) \*\*NOT SURE ABOUT THIS LINE (also no loadedpackages in problems files)*

*(at end (available ?d))*

*(at end (drone-base-available ?db))*

*(at end (not (charging ?d))))))*