Notes, ElliotZ

Started 03/19/2024.

Last Updated 03/31/2024.

These are my thoughts for the project tasks given the idea of a university transportation system. We will use a few key locations around the U of A to develop and demonstrate our system. The objective is to help busy students better utilize their valuable resources by maximizing the efficiency of their commutes across campus. In the future, this system could be scaled to help people plan larger, longer, and more complex travel routes, i.e. involving flight plans, shuttles, rentals, public transit, etc.

Some of the things we need to start gathering/defining to implement the project:

* List of locations to travel between.
  + Parking lots, bike racks, etc. We need to have a clear list of available transportation options before we can develop all these.
  + Buildings
  + Straight line distances between locations to use as a heuristic when optimizing paths for distance.
  + Minimum time between locations to use as a heuristic when optimizing paths for time.
* Definitions of a few traffic (environmental?) conditions to test against.
  + Consider class changes, 5:00 in the morning, etc.
  + This would be data provided to our system from an external entity, for testing we will have a variable that we can change to specify one particular condition at a time.
    - Maybe simulate a broken-down streetcar? Happened to me the other day…
* List of transportation options.
  + Plan at least one public transit, such as a bike share. This will be the subject of our AIPS, which will automatically redistribute transportation devices as necessary (assume autonomous navigation).
  + Transportation options that may be available to individual users.
* List of user disabilities
  + Plan for at least one handicap (being wheelchair bound, for instance) that may limit the user’s options.

High-level Description of System Behavior

The first major component of our system is a mobile app hosted on a webserver and connected to a database. Users will create a profile where they enter important personal information, such as disabilities or transportation options uniquely available to them (skateboard, Manny?). When the user plots a route, the system must generate the graph dynamically. That is, it must consider what nodes to include in the graph, it must identify valid connections between nodes, and it must be able to assign costs to the edges. These will all depend on the parameters outlined above. Then, the system must find the optimum path (optimize for either time or distance) using a search algorithm. (A\*, AO\*?)

The second major component of our system will be some form of public transit that requires an AIPS to manage. I’m thinking of something like a bike share, where there are a few strategic locations around campus that feature special bike racks (“depots”) where people can come up and take a bike using their Catcard. Assuming the bikes can navigate autonomously, the AIPS would determine when and where to redistribute bikes between depots. This deserves more discussion. There may be opportunities to implement predicate logic here as well…

Project Tasks

1. System Requirements
2. Overall System Architecture
3. Component Interactions
4. Key Concepts Applied
   1. Path Planner
   2. AIPS
   3. Predicate Logic?

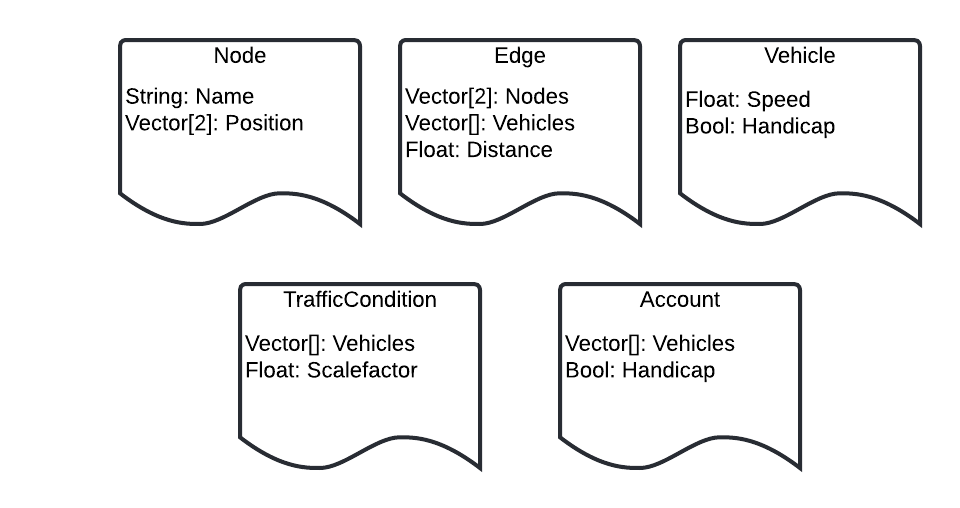
Deliverables

1. Code
2. Report
3. Presentation

System Architecture (Draft)

A diagram of a computer

Description automatically generated



**Node**

**Name**: A string that uniquely identifies each node.

**Position**: An X-Y coordinate pair that will be used to calculate straight line distance between nodes. The straight-line distance can be used as a heuristic that always underestimates.

**Edge**

**Nodes**: Pair of nodes that the edge connects.

**Vehicles**: List of vehicles that can traverse that edge. For instance, passing through the student union would be limited to foot traffic, while a road (including sidewalk) can be used by both foot traffic and automobiles.

**Distance**: The actual length of the edge. We’ll need to decide on units. When optimizing for distance, this will be identical to the cost of the edge. When optimizing for time, the cost of the edge will have to be calculated using this property and the speed of the vehicle in question.

**Vehicle**

**Speed**: The speed of the vehicle. Units will have to be consistent with distance.

**Handicap**: Whether or not the vehicle is an option for handicapped users. Note that this will depend both on the type of vehicle and the edge that the vehicle is associated with. For instance, some vehicles will outright be unavailable to handicapped users (i.e., skateboard). Others will be available, but only on certain routes (i.e., where ramps exist).

**TrafficCondition**

**Vehicles**: List of vehicles that the traffic condition affects.

**Scalefactor**: A scale factor that will be used to adjust the speed of affected vehicles.

**Account**

**Vehicles**: List of vehicles available to the user. This one may get tricky. For example, consider the case where the user parks their car; how will the system remember where the car was parked when determining what vehicles are currently available to the user from any given point?

**Handicap**: Whether or not the user is handicapped.

So, what needs to be done next?

* Formalize system requirements.
* Agree on a text file format for the various classes that can be read in by the application code as if from a database and begin developing the software framework to do so.
* Start defining some nodes, edges, vehicles, traffic conditions, and user accounts! They should be numerous and robust enough to really test out our system. Capture them using the text file format agreed on above.
* Work out the details of the AI production system. I’ve intentionally tried to avoid specifics to see what ideas the team has.