Startup Tycoon

Algorithms with a Purpose

October 17, 2015

1 Introduction

Having recently graduated from X Combinator, your startup, Tuber, is struggling to disrupt the status quo in San Franhattan. According to Haxxor News, there's a growing demand for widgets in the city and being the street-smart entrepreneur you are, you decide to pivot your company to on-demand widget delivery. Using your $30 billion of venture capital, you decide to hire the best researchers out of the nearby Cranberry Melon University. However, you aren't alone – you will be joined by others who are also trying to disrupt this nascent market. To take over this niche in the market, your team must develop the best algorithm for delivering widgets! So get out there and deliver some widgets!

2 Terminology

player	a single entity in a game, representing a team of up to 4 humans
game	an iteration of a set of orders and turns on a distinct map
turn	a single time step of a game during which an order may appear and a player may take an action
action	creating a station and/or responding to orders
order	a request for a widget originating from a home on the map
widget	the item being delivered by an order
home	a node on the map that can generate orders for widgets
station	a node on the map that can respond to orders for widgets
map	the map on which orders appear and actions occur, consisting of a collection of nodes, some of which are connected

3 Rules

1. San Franhatten is represented as an unweighted, undirected, connected graph. Some number of "hubs" are chosen on this graph unknown to hte player, and orders appear distributed around these hubs.

2. The game is split into into 1000 time steps (or turns), each of which is capped at 0.5 seconds of real time. Each turn follows this structure:

(a) Generate a new order and add it to the graph.

(b) Remove orders that can no longer be fulfilled.

(c) Ask the player for a set of commands.

(d) Execute each command in order.

3. A single order will possibly appear each turn with a fixed probability. A random hub is selected and a number of steps is sampled from a normal distribution, then the order is placed at a random walk of that many steps away from the hub. An order contains the following:

   - A destination node
   - A bounty for the request (sampled from another normal distribution)

   Orders have arithmetically decreasing value in time. When an unfulfilled order's value reaches zero, it disappears from the map if you do not have a widget in transit, otherwise it gives no money when the order is fulfilled.

4. You can respond to requests still on the map whenever you choose after receiving it (including the time step it was received) by returning a path from the station to the destination including the station and destination. You can respond to more than one order on each turn.

5. A new station can be built on any turn for a price and can begin serving on the same turn (the station build order must precede the widget send order in the command list). Stations can be built on any node in the map, including homes. The cost of buildling a station increases geometrically in the number of stations built.

6. Because Tuber uses pneumatic tubes for delivery, each edge on the path of a widget being delivered (i.e. an "active" order) cannot be used for the duration of transit, i.e. each edge is unusable for time equal to the number of edges in the path.

7. Your final score is the money you have left at the end of the game.

4 Tournament

- All competitors' algorithms will compete in four rounds, with the score equal to a weighted sum of the algorithm's rank.
- The timeouts will not change between testing and competition. All other constants (i.e. those founds in `settings.py` are subject to change. We will not use graphs of over 2000 nodes.

5 Getting Started

- READ THE README.md FILE!

- Read the code, namely:

  `/src/game/player.py` - You will implement your algorithm here.

  `/src/game/base_player.py` - Base class for the Player. Contains utilties for creating your algorithm.

  `/src/game/state.py` - All the state maintained for the game.

  `/src/game/order.py` - Representation of orders in the game.

  `/src/game/settings.py` - Constants and graphs used in the game.

- Familiarize your self with networkx. Start with how to get nodes, edges, and attributes of those nodes and edges. This documentation is linked to from the README.md.