Tuber Drive

Algorithms with a Purpose

February 4, 2017

1 Preface

Algorithms with a Purpose will take place on Saturday, Febuary 4, 2017 at 12:00pm until 8:00pm. The main event room will be in Rashid auditorium (GHC 4401), coding will take place between 12:30pm and 6:30pm, and judging will occur from 6:30pm to 8:00pm.

2 Introduction

Tuber, the company that you founded about a year-and-a-half ago, has really taken off! You're dominating the market for widget delivery and have expanded to multiple cities besides your home base of San Franhattan. However, you're losing a substantial chunk of profit to your suppliers, who have hiked up their prices due to increased demand and popularity. As such, you have decided to program some roving robots to drive around to search and mine for widget resources, which will hopefully be much cheaper in the long run, but other startups have caught wind of your plan and are starting to program their own mining robots as well, hoping to ride on the widget boom. In order to secure your place as the premier widget company, you and your team must develop an algorithm for retrieving the most widget resources!

3 Terminology

player	A single entity in a game, representing a team of up to 3 humans
field	The game board; a grid consisting of robots, resources, and beacons
robot	A harvester robot that you will provide the logic to make moves
base	The starting location where robots can deposit resources
resource	A finite widget resource that robots can mine and transport
beacon	A persistent marker that robots can place and detect
terrain	The type of a given tile. Indicates whether a position contains a resource or is reachable by the robot
action	What the robot can do each turn (see Rules)
view	The limited information that a particular robot receives to plan an action

4 Rules

1. The field is represented as a grid of size between 50×50 and 200×200, depending on the board. Resources are generated at positions on the grid unknown to the player.

2. The player starts with 50 robots, all initialized at the base at the center of the field.

3. The game is split into into 1000 time steps (or turns), and is capped at five minutes total for each board, such that your final score for a board is your score at five minutes. At each turn, the robots execute their `get_move()` program, performing one action each.

4. An action can be one of the following:

   - Move north, east, south, west, northeast, northwest, southeast, or southwest.
   - If on a resource, mine some amount of the resource
   - If at base, deposit resources

   At the same time as one of the above, a robot may drop a beacon on the tile it currently occupies

5. Every robot, before performing an action, get a `view` (a Python list) containing a tuple of information about each tile currently in their field of vision:

   - The type of terrain of the tile (e.g. resource, base, mountain)
   - Number of robots currently on the tile
   - List of beacons on the tile

6. There are five colors of beacons: red, orange, yellow, green, or blue. Beacons stack and are persistent; it is up to the player how to interpret beacon colors or patterns.

7. There are four types of terrain:

   - Base: The player's home base and every robot's starting position. Resources must be deposited here to add to the total score.
   - Resource: A tile which contains some amount of resource. A robot can check how much of a resource is remaining, what kind of resource it is, and the value of the resource.
   - Plain: Normal, traversable terrain.
   - Mountain: A non-traversable tile. Also used to denote the edge of the playing field, and in a robot's view, tiles a robot cannot see.

8. TIP: Robots do not move simultaneously on each turn; they each get their own view and perform actions one after another.

5 Tournament

- All competitors' algorithms will be run on every tournament board, with the **final score being a weighted sum of the scores from each board**.
- The timeouts will not change between testing and competition.
- The sample maps will be used in the tournament, and will be weighted slightly more than the unrevealed maps.

6 Running Your Code

- If you want to run the game with a randomly generated board: `python3 gameMain.py`
- If you want to run the game with a preset json file: `python3 gameMain.py <name_of_json_file>`. 2 sample maps, `map1.json` and `map2.json` have been provided for you.
- At the end of the game, a `map.txt` file will be generated. Copy this file into the `visual` folder and type `python -m http.server` from the `visual` folder. Afterwards, direct your browser to `localhost:8000` to view the output. Every time you update the `map.txt` file, you will have to re-launch the server.

7 Getting Started

A few tips:

- Read the code, namely:

  `/src/PlayerRobot.py` - You will implement your algorithm here in `get_move()`.

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

  `/src/board.py` - The playing board for the game which keeps the game state.

  `/src/constants.py` - Constants used in the game. You can edit this to test your algorithm locally.

You have access to the following from `/src/robot.py` and `/src/tile.py` - the full list is enumerated in the sample algorithm in `PlayerRobot.py`:

- `Robot.get_fov()` (int): Range of vision that robot can see. Vision/2 to the left and Vision/2 to the right.
- `Robot.get_max_capacity()` (int): Max amount of resources that the robot can hold.
- `Robot.get_pickup_amount()` (int): Amount of resources the robot can pick up every turn
- `Robot.held_value()` (int): Value of resources currently held

- `Robot.storage_remaining()` (int): Amount of resources the robot can still pick up
- `Tile.CanMove()` (bool): Determines whether or not robot is able to move to that tile
- `Tile.GetType()` (TileType): Enum in constants.py that determines what type of tile it is
- `Resource.Value()` (int): Value of 1 unit of resource
- `Resource.AmountRemaining()` (int): Number of units of resource remaining
- `Marker.GetColor()` (MarkerType): Enum in constants.py that determines color of marker