CIS603 Project 2a: JiaoTong Initializer

Due: 17 Nov 2013

This is version 1.1 of this project description

1 Timeline (Tentative)

- *Project 2a* – Compile and run code, and begin it modify it.
- *Project 2b* – Implement and evaluate a simple model-based learner and a satisficing learner
- *Project 2c* – Implement and evaluate a more complex model-based learner and an expert algorithm
- *Project 3* – Implement and evaluate your own winning algorithm. Enter your algorithm in The JiaoTong Cup.

2 Deliverables for Project 2a

To complete this project you should:

1. Determine your team make-up and your team name (make it good – nobody will take you seriously if your team name is stupid). Remember that you can have up to 1 partner on this assignment.
2. Compile and run the code.
3. Implement the stated modification (below).
4. Send an email to `cis603fall13@gmail.com` stating your team members, team name, and that you were able to successfully compile and run the code, that you were able to implement the stated modifications, and that your were able to run multiple versions of the algorithm in JiaoTong simultaneously.

3 JiaoTong – Overview

JiaoTong is a simple transportation simulator that we will use for Projects 2 and 3 in this course. In this game, vehicles interact in the simple transportation network shown in Figure 1. The network consists of 4 nodes (Nodes A, B, C, and D), with roads connecting the nodes as shown. The goal of each vehicle is to move about the network so as to maximize its own utilities.

Please note the following:

- The speed of a vehicle on a given road depends on the number of vehicles currently on that road. As the number of vehicles on a road approaches and surpasses the road's capacity (stated in the silly looking "rainbow" displays), the speed of the vehicles on that road slows.
- Costs: A vehicle accrues a cost of 0.079 happiness units for each second it is moving. The vehicle accrues additional toll charges each time it enters a new road (with the exception of the road connecting Node D to Node A. Tolls can be altered by clicking on them in the GUI.
- Rewards: A vehicle receives a reward when it stops at a node that gives it positive utility. These rewards are normalized to the magnitude of the Costs in the system. Reward values are provided to each vehicle by the JiaoTongServer (more on that later).

Your vehicle is provided with the following information each time it arrives at a node:

- The travel cost and toll charge for travel over the previously chosen link (*travelCost* and *tollCharge*).
- The reward you received for arriving at your current node (*payout*).

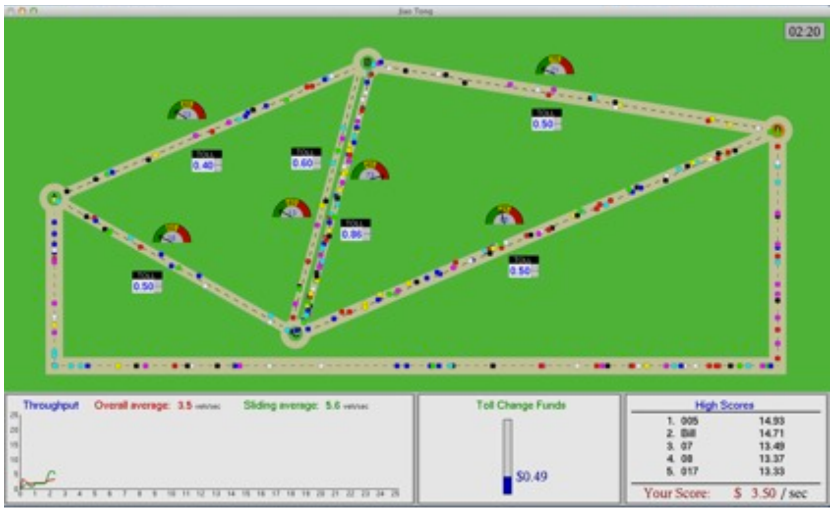

Figure 1: GUI showing JiaoTong.

- The node at which your vehicle is currently located (*currentNode*).
- The payouts you will receive in the future for arriving at each of the four nodes (*currentNodeUtilities*). You must move away from and return to the node your are currently at in order to receive a payout at that node.
- The average number of vehicles that were on the link you just traveled over as well, as the capacity of that link (*aveOnLink* and *capacity*).

These values are passed to your client program and stored in variables in your code so that you can access them.

4 Code

The JiaoTong code we will use in this project uses a client-server model. The server (JiaoTongServer) simulates the world and communicates the status of the world back to clients. The clients (the vehicles) connect to and communicate with the server over TCP/IP sockets. You don't have to do anything to facilitate this communication except to run the programs in the proper order – the code is already provided for you.

4.1 JiaoTongServer

Before running the JiaoTongServer, you must compile it on the machine that you will be using. If you are using a Mac, you compile this code by typing `make` (in a terminal) in the directory in which the JiaoTongServer code is located. If you are using Linux, you also do this by typing `make`, but you must first switch out the makefile for the file that is labeled `makefile_for_linux` (just rename the files). If you are using Windows, a project to compile the code using Visual Studio is provided. Visual Study can be downloaded and compiled for free. You can also use other programming environments if you desire.

If you are using Linux or Mac, you'll need to have OpenGL installed on your machine (I think I've included the files already you'll need for Windows). Here's directions for Linux users (Mac users should be able to find something similar):

```
http://stackoverflow.com/questions/3907064/installing-opengl-and-openal-in-ubuntu
```

Once you have compiled the JiaoTongServer, you can run it from the command-line. If you are using Linux or Mac, use the following command:

```
./JiaoTongServer theworld [numVehicles] [GameLength]
```

where `[numVehicles]` is the number of vehicles in the game, and `[GameLength]` is the number of minutes the game will last. If you are using Windows, go to the `JiaoTongServer_Win/Debug` directory, and type (in the command-line):

```
JiaoTongServer_Win.exe theworld [numVehicles] [GameLength]
```

where the variables are defined the same as for Linux/Mac users. Once you run this command, the JiaoTongServer will wait until `numVehicle` clients connect to it.

4.2 Clients – Vehicle Intelligence

Basic clients are provided in C/C++, Java, and Matlab. Use the language that you are most comfortable with. These clients cause an agent to move about the world randomly. You will add to this code to make the agents more intelligent throughout Projects 2 and 3.

Instructions for compiling and running the code for each language are provided below.

4.2.1 C/C++

If you are using a Mac (and probably Linux), you compile this code by typing `make` (in a terminal) in the directory in which the source code is located. If you are using Windows, a project to compile the code using Visual Studio is provided. Once compiled, you can run this code from the command-line with the following command:

```
./RandomAgent localhost [ID] [name]
```

where `[ID]` is the connection order of the clients (first to connect is 0, the second is 1, etc.), and `[name]` is your team name.

4.2.2 Java

You can compile this code by typing `javac *.java` in a command-line (in the directory that contains the source code). Note that you'll need to have Java installed. You then run the code using the following command-line syntax:

```
java Duder localhost [ID] [name]
```

where `[ID]` is the connection order of the clients (first to connect is 0, the second is 1, etc.), and `[name]` is your team name.

4.2.3 Matlab

If you have Matlab installed on your computer, all you have to do to run this code is to go to the directory with the source code, and then type:

```
Duder('localhost','[ID]','[name]')
```

where `[ID]` is the connection order of the clients (first to connect is 0, the second is 1, etc.), and `[name]` is your team name.

4.2.4 Clients for other programming languages

If you want to use another programming language (such as Python), that is fine, but you'll need to write the client yourself. In the last section of this write-up, there is a description of the communication protocol that is used.

5 Tasks: Project 2a

5.1 Task 1 – Establish a team name

Once you've decided if you are working alone or if you are working with somebody else (i.e., determine your team members), come up with a really cool team name.

5.2 Task 2 – Compile and run the code that is provided

Compile and run the code that is provided. To do this, first run the JiaoTongServer:

```
./JiaoTongServer theworld 1 1
```

Then, run one client using [ID]=0 and [GameLength]=1. For example, if I were running the C/C++ code and my team name were `GradeGrubber`, I would type:

```
./RandomAgent localhost 0 GradeGrubber
```

If everything is working correctly, the JiaoTong GUI will then pop up on your screen and you should see a little circle (your vehicle) randomly moving around in the world. The circle will stop moving after one minute (the length of the game you specified when starting the JiaoTongServer). You can see the relative "joy" of your agent plotted at the bottom of the screen (only the first 8 characters of your team name will be displayed).

5.3 Task 3 – Create a more structured random agent

We have talked about normal-form games in class a lot. JiaoTong isn't a normal-form game, but there are various ways that we can structure it so that it looks like one. Let's look at one way to do this, and then you'll create a client that implements this conversion, and then chooses its actions randomly.

5.3.1 Convert JiaoTong into something like a normal-form game

An agent has essentially five cycles. The first four cycles involve moving from A through B or C (or both) and then to D and back to A again. These paths are:

Cycle 1: $A \rightarrow C \rightarrow D \rightarrow A$
Cycle 2: $A \rightarrow B \rightarrow D \rightarrow A$
Cycle 3: $A \rightarrow C \rightarrow B \rightarrow D \rightarrow A$
Cycle 4: $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$

There is also one other cycle, which is to move back and forth between B and C. This cycle can be defined as:

Cycle 5: $B \rightarrow C \rightarrow B$

This means that your agent really will just repeatedly pick a cycle (an action) to run through. Since the other vehicles will be doing the same thing, this seems very much like a multi-player repeated game.

The reward your vehicle gets for selecting a particular cycle is simply the reward it gets for visiting each node in the path (don't count the starting node), minus the costs associated with moving through the path.

For example, if the agent gets a reward of zero for visit Nodes A, B, and C, and a reward of 5 for visiting Node D, then its reward for taking Cycle 1 would be:

$$0 + 5 + 0 - \mathit{toll}(A \to C) - \mathit{tCost}(A \to C) - \mathit{toll}(C \to D) - \mathit{tCost}(C \to D) - \mathit{toll}(D \to A) - \mathit{tCost}(D \to A),$$

where $\mathit{toll}(A \to C)$ is the toll charge for taking the road from Node A to Node C and $\mathit{tCost}(A \to C)$ is the time cost for traversing the road from Node A to Node C.

Given a set of actions and the corresponding payouts for each path (which will need to be learned from experience), you should be somewhat comfortable with at least starting to reason about what your agent should do. At this point, we won't do anything strategic. Rather, your job is to implement an agent that randomly selects a new cycle each time it completes a cycle. It then traverses the path specified by the cycle, and upon completion selects a new cycle. Note that if it selects Cycles 5 when it is on Node A or D, it will need to first move to Node C or Node B and then begin the cycle.

5.3.2 Change the client code

Adapt the client code (in the language you prefer) to the algorithm just stated. To do this, you'll need to alter the selectAction method so that it randomly selects a path and then traverses the path, rather than just randomly picking a new road when it reaches each node.

5.4 Task 4 – Get multiple versions of your random agent working

Get multiple vehicles (each running the code you created in Task 3) running at once in JiaoTong. You do this by running the JiaoTongServer with the desired number (N) of vehicles, and then running N versions of your client code (each with a different ID). For the time being, I'll let you figure out how to run multiple clients in an easy way (batch files/shell scripts for C/C++ and Java) and there's a way to do it for Matlab as well. I'll specify things on this point more fully in the future.

Get a lot of vehicles running in the world ($N = 30+$) at once so you can see what happens.

6 Communication Protocol

If you want to create a client using a language other than C/C++, Java, or Matlab, you can do so. You'll need to connect to the JiaoTongServer using a TCP/IP socket on port 3000+[ID]. Once you've established a connection with the server, the server will you a string specifying the layout of the world. This string will have the following structure:

```
[posx] [posxy] [numlinks] [link1] [link2] [link3] [link4]\n
[posx] [posxy] [numlinks] [link1] [link2] [link3] [link4]\n
[posx] [posxy] [numlinks] [link1] [link2] [link3] [link4]\n
[posx] [posxy] [numlinks] [link1] [link2] [link3] [link4]\n
```

Each line contains information about one node in the world. The first two values are doubles that give the (x, y)-coordinates of the node, the third number is an integer that indicates the number of roads that lead out of that link, and the last four numbers are integers that specify the destination nodes (-1 if the link does not exist).

The server also then sends you your start node with the string: "`[startNode]\n`".

After receiving this information, you are required to send a string with the following structure: `GradeGrubber\n`, but substituting your team name for "GradeGrubber." You then send a string with the structure "`[red] [green] [blue]\n`", where [red], [green], and [blue] are floats/doubles specifying the rgb color of your agent.

After this initial exchange, you are ready to start playing the game. To do this, you'll alternate between receiving a message indicating you've arrived at a new node and sending a message saying where you want to go. When you arrive at a node (and at the very start of the game), you'll received a message with the following format:

```
Results: [travelCost] [payout] [tollCharge]\n
Position: [currentNode]\n
Utilities: [payout0] [payout1] [payout2] [payout3]\n
LinkState: [aveOnLink] [capacity]\n
```

where [travelCost], [payout], [tollCharge], [payouti], and [aveOnLink] are doubles, and the others are integers. [payouti] specifies the reward you'll get when you reach node i. The other variables are self explanatory (I hope).

After you receive this message, you are expected to send a message indicating which link you want to take. If you are going to a node that gives you a positive payout, you send the message: "`[link] Y\n`". If you are going to a link with zero payout, you give the message "`[link] N\n`". In each case, [link] is the link you want to travel on, where 0 is the first link, and 1 is the second link (coming out of the current node).

If you are confused, come and ask me.