Peter Glennon

COM 498/499 Research

Instructions on Running

First and foremost, I apologize for how the project is organized. I didn’t do much top-down planning to decide how classes would fit together, and it shows. For instance: MatchingAlgorithms.java contains methods for running a few algorithms, including Greedy Online, but other algorithms have their own classes! And the code for Greedy Online is repeated elsewhere, I believe. Fortunately, this project is not so large, and finding what you need to find shouldn’t be that difficult.

Beyond the code for the Algorithms and Parking Models, there is also a “Matching Algorithm Visualizer”, which was built early on in the project. It was intended as a visual aid to see Greedy Online in action, which it does quite well. However, as I implemented other, more complex matching algorithms, I realized this visual framework wouldn’t be able to show their more detailed inner workings. As a result, the Visualizer is somewhat useless in the scope of this project, but can be used as a simple tool for showing how Greedy Online works (with a small number of nodes). To run it, either compile and run VisualMatchWindow.java, or execute the command “ant run” which will run the ant task “run”, defined in build.xml.

The real purpose of this project is to test matching algorithms against each other. I regret not building a more elegant, flexible testing platform in which a user could select which algorithms to test, the model they ran on, etc. Instead, I just wrote a bunch of messy code for the first tests (Greedy Online vs. Max-Flow/Optimal Offline), and then copied it and changed it around to fit other algorithms. The files beginning with “Test” (TestCampusOne, TestGrid1060, etc.) will run each algorithm (save Permutation) on randomized instance sets. The number of runs per algorithm is determined within each file by a variable called “numRuns”. The files beginning “PermGreedy” are similar, but also run Permutation on the instance sets. To run a test file, just compile and run. While this is not a great solution, it works well enough. The great downside is that, if you want to change something in a test, you would need to do the same in each test file. I would suggest anyone wishing to expand on this project create a more adaptable test framework first.