# CS 182 Project Check-In

Jason Goodman, Raynor Kuang

11/23/16

Here's an outline of our progress and intended plan for the future:

1. We settled on an elevator model where people input destination floors. This is (probably) still NP-complete, but we want to see if we can do better than with traditional inputs since this is technologically feasible. Thus, our assumptions can be summarized as such:

   - Every timestep, an elevator can either
     - travel to an adjacent floor
     - sit with its doors open (let people one/off)
     - sit with its doors shut (killing time)
   - Riders automatically board an open elevator going in the right direction
   - Riders automatically leave an elevator on their floor
   - Elevator state:
     - List of riders (destination and wait time are interesting, and could be packaged as part of the rider information)
     - Floor
   - Elevator state:
     - List of riders (destination and wait time are interesting, and could be packaged as part of the rider information)
     - Floor
   - Simulation inputs:
     - Number of floors n, number of elevators k, elevator capacity c
     - List of rider source/destination/timestep triples
     - Agent function from a list of elevator states to a list of actions (move up/down or sit open/closed)
   - And output a list of wait times.

2. The basic idea for the simulator is that the agent function takes a timestep, list of elevator states (floor, riders), and map from floors to waiting passengers; it then returns an action for each elevator: UP, DOWN, OPEN_UP, OPEN_DOWN, or SIT. Open_up/down means indicate that passengers going up/down should board.

3. On the AI side, we ruled out search techniques because unknown future events matter (e.g., plan to handle a bunch of arrivals on the first floor) and the probability doesn't seem too straightforward to shove into minimax/expectimax. As a result, RL makes the most sense, so we'll use the implementations from the Pacman homework. A significant amount of work here will be designing state spaces and rewards, since the Berkeley team was so nice to make the RL implementation game-design independent. A special case would be a state space that considers passenger direction rather than destination floor; we'll make one of those so we can see how much improvement comes from knowing destinations.

4. Another big chunk of work is modeling distributions of passenger behavior and figuring out how much this model affects performance. We'll start off with simple schedules ("one passenger every minute going from ground to a random floor, waiting a random amount of time, then returning"), then try to model rush hour traffic and exponentially-distributed unstructured traffic that could still lead to weird bottlenecks. We may consider typical distributions across various variables like arrival time and floor, like normal distributions and Poisson distributions. Performance will be evaluated to how well the system does with an arbitrarily large training data set as well as how it responds in real time to variations.