SafeRide onCall AI System Design Overview

Michael Benisch, Joshua Butler, Dan Fox, Eden Hochbaum, Victor Naroditskiy
Department of Computer Science
Brown University, Box 1910
Providence, RI 02912
{mbenisch,jtbutler,dmfox,ehochbau,vnarodit}@cs.brown.edu

1 Overview

The most important (and obvious) goal of the AI team is to design and return a shuttle routing schedule that optimizes an objective function which will be described later. This schedule will produce estimated wait times. Less obviously, the AI component needs to be fairly robust in allowing dispatchers to override specific parts of schedules without affecting the overall dispatching structure more than is necessary and the AI component needs to be extensible in allowing for tweaked objective functions that conform to a specific structure, i.e., the objective function would only be improved if average waiting times were decreased with all other variables remaining constant.

The AI system will direct each shuttle at each step of its progression where a step is defined as either dropping a rider off or picking rider up. In other words, the AI system will recommend that a particular shuttle drive to (for example) 40 Brown Street and pick up rider John Smith. Once the driver has signalled arrival and pick-up (or lack of pick-up) the AI system will recommend the next action for that van. Dispatchers (human) will be asked to approve recommendations before drivers are ever alerted.

2 Architecture

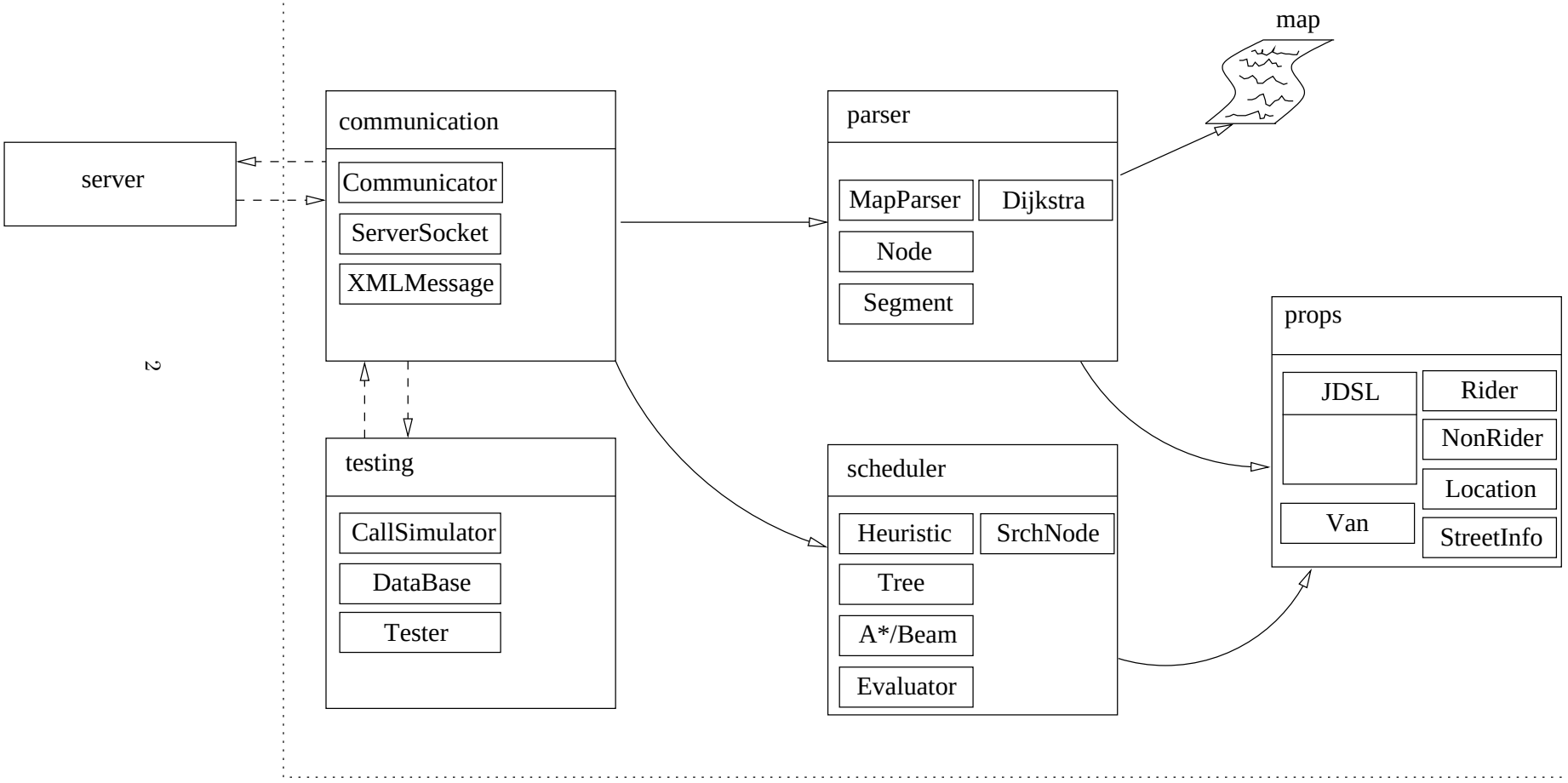
A high-level description of our architecture is shown in Figure 2. The basic flow of information begins with communication between the server and the communication package. The communication package will include a `Communicator` class which will open a dedicated socket. Messages will invoke a call back method on the socket, and trigger the optimization algorithm to begin running. The algorithm will produce a schedule and the information in the schedule (i.e. which actions each van must undertake next) will be entered into the server's SQL database and a message will be sent to the server to indicate the update.

3 Search Space

The search space in our problem is defined as follows (for example see Figure 3):

- Each node contains the following state information:
  - Which passengers are in each van
  - Which passengers are unassigned
- Branches represent an *action* that a particular van will take. Actions can be one of the following:
  - Pick up a caller waiting for a ride
  - Drop off a rider

SafeRide AI System Overview



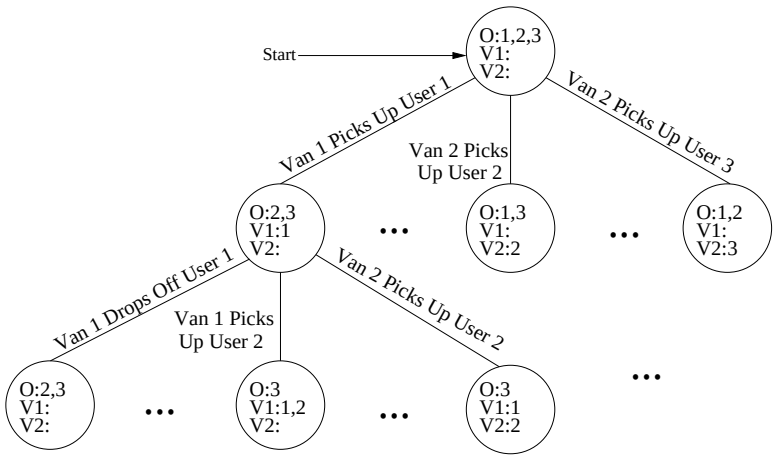


Figure 2: Search Space Example

- The depth of our search space = O(m), where m is the total number of people waiting for rides and riding in vans. In worst case the depth is exactly $2m$ when everyone is waiting for a ride. In this case we need to pick up each person and then drop him off. At each level of the tree we make a decision about picking up or dropping off one person.
- End nodes of the tree contain schedules for each van. The schedules specify pick-up and drop-off order for each of the m people.

4 Objective Function

The algorithm tries to minimize two measures for each ride. First is the error in the wait time estimate, the amount by which the van's pickup time p_i for a ride is later than the projected pickup time ep_i given to the caller. The second is the total trip time, the difference between the time d_i the rider arrives at her destination and the the time c_i she requested the ride.

$$\sum_i \alpha \max(0, p_i - ep_i)^\gamma + \beta \left(d_i - c_i\right)^\mu$$

5 Heuristic Function

The heuristic function finishes potential schedules in the following way:

- First drop off everyone in each car in the quickest order.
- For each unassigned user, i, ordered by estimated (absolute) waiting time
  - Assign user i to the earliest available van.
  - Calculate the group associated with that user as anyone within $r = \frac{d_i}{K}$ where: d_i is the length of user i's trip.
- Remove all the users in the group from further consideration.
- Pick up that group in any order and drop them off in any order.