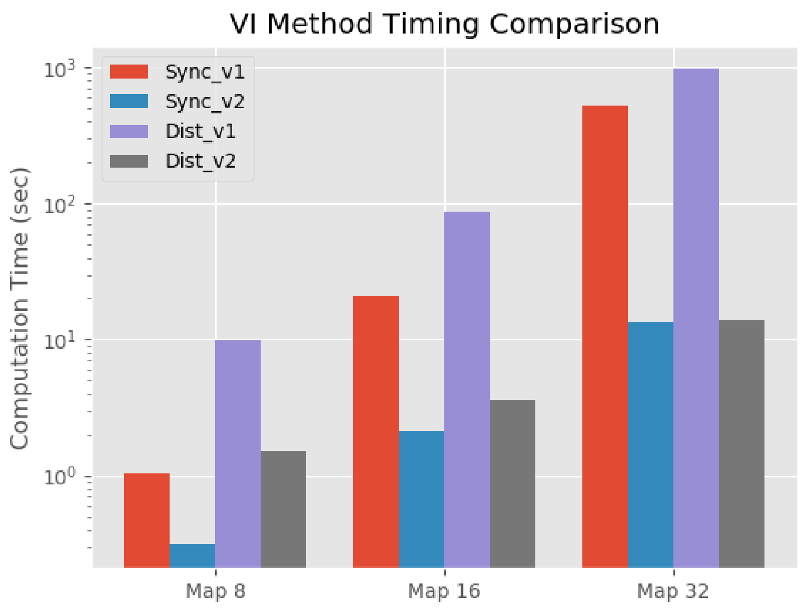
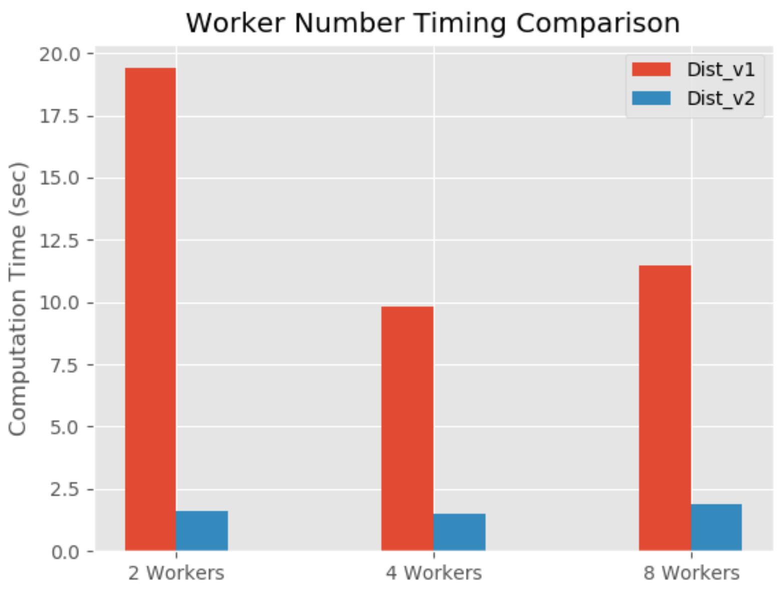
CS 533 Homework 2: Distributed Synchronous Value Iteration

1. Value Iteration Implementation Method Timing Comparison Plot:



2. Number of Workers Timing Comparison Plot:



3. The second distributed method is much faster than the first because it avoids making a remote worker for each state. While creating parallel processes can speed things up by doing computation in parallel, there is an overhead for each worker that is created. If too many workers are created, the overhead cost of creating these workers could offset the time savings of the parallel computation. Such is the case with the first distributed method which makes a separate process for each state. This is far too much overhead, thus causing the speed difference between the two distributed methods.

4. The best non-distributed approach was version 2, which used the GetSuccessors method as opposed to checking every possible transition state. The best distributed approach was version 2, which partitioned the state space according to the number of workers as opposed to making a worker for each state. Which approach performs faster depends on the size of the state space. For the 8x8 map, the non-distributed approach is faster, but on the larger maps the distributed approach starts to perform faster. This is because parallel processing gives multiplicative speed up, but also has a constant overhead cost. For smaller maps, the non-distributed version's computation time is already so small that dividing that time by 4 is outweighed by the constant overhead cost of creating the remote workers. However, as the non-distributed's computation time increases as the state size increase, the multiplicative parallel speed starts to outweight the constant overhead and the distributed version becomes faster than the non-distributed.