Outline

- Review of the optimizing simulator for large truckload networks

- Why and where to parallelize and distribute the computation

- Comments on results and issues
The Tactical Planning Simulator (TPS)

- Given a snapshot of *drivers* at the beginning of a simulation horizon *and* a set of *loads* available throughout the horizon,

- We want to simulate the assignment of drivers to loads over *time* so as to *maximize* the net contribution of moving the loads,

- Subject to operational and physical *attributes* and *restrictions* of drivers and loads.
The driver-load assignment over time

On Monday of week 1:

- $300
- $150
- $450
- $400
The driver-load assignment over time

On Thursday of week 1:

- $500
- $600
- $180
- $150
The driver-load assignment over time

On Tuesday of week 2:

- Time at home
  - $20
  - $60
  - $130
  - $200
The driver-load assignment over time

On Friday of week 2:

- $160
- $120
- $100
- $300
The resource constraints and attributes

- **Drivers**
  - Category
  - Domicile
  - Location
  - Available time
  - Detailed hours of work according to US rules
  - Preferred/required time-at-home
  - Geographical constraints
  - Varied unit costs

- **Loads**
  - Current location
  - Origin
  - Destination
  - Pickup/delivery time window and type
  - Special driver requirements
  - Revenue components
Solving large problems

- For large problems, it is impractical to solve them as single optimization problems
- Time and space decomposition
- Approximate dynamic programming
Time-location subproblems

- In the *morning* of Monday of week 1 (12 hours):
Time-location subproblems

- In the *afternoon* of Monday of week 1 (12 hours):
Time-location subproblems
In the morning of Monday of week 1:

- Value functions
  - network effect
  - home repositioning
Time-location subproblems

- The algorithm looks like an iterative simulator:
The basic steps in the solution of a time-location subproblem:

1. Generate options for the driver-load assignment and cost them
2. Generate and solve LP corresponding to the optimization problem
3. Collect information to update value functions (e.g., estimate the marginal value of an extra driver with given attributes through numerical derivatives)
4. Implement the solution (update drivers and loads)
Outline

■ Review of the optimizing simulator for large truckload networks

■ Why and where to parallelize and distribute the computation

■ Comments on results and issues
Why parallelize

- Solving one iteration of the problem still takes a bit too long (5500 drivers, 55000 loads, 3 weeks)

- Calibration of value functions may require a few hundred iterations

- Studies with multiple scenarios and/or data sampling

- Sensitivity analysis
Parallelization opportunities

Subproblem solution:

- **70%** 1. Network generation
- **10%** Generate driv-load links
- **60%** Cost links
- **5%** 2. LP solution
- **10%** 3. Numerical derivatives
- **15%** 4. Solution implementation
Parallelization opportunities

- What we have actually implemented:
  
  » *Geography-based parallelization*: distributing the solution of all the location subproblems *within* a given time interval
  
  » *Network costing parallelization*: distributing the costing of the driver-load assignment links during the solution of a given subproblem
Parallelization opportunities

Geography-based parallelization: limitations on the maximum speedup due to imbalances on the geographical decomposition

- largest region: 20% of total solution time
- as time progresses fewer regions become dominant
- the average maximum possible speedup is around 3.5
- instance dependent
**Parallelization opportunities**

Network costing parallelization: maximum achievable speedup (assuming that enough processors are available)

<table>
<thead>
<tr>
<th>Step vs. Platform</th>
<th>Generate Links</th>
<th>Cost Links</th>
<th>LP solution and Num. Deriv.</th>
<th>Solution Implementation</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td>10</td>
<td>60</td>
<td>15</td>
<td>15</td>
<td>100</td>
</tr>
<tr>
<td>Parallel</td>
<td>10</td>
<td>5*</td>
<td>15</td>
<td>15</td>
<td>45</td>
</tr>
</tbody>
</table>

* Time needed to transfer data between processors, assuming a non SMP environment
Parallelization opportunities

- solve individual subproblems, as assigned by the farmer
- send driver-load links to the farmer to be cost out

• controls time synchronization
• controls distribution of parallel tasks
• implements subprob solutions
• updates value functions
• gathers and outputs results

exogenous data

control and data flow

Farmer

Subprob Solver 1
Subprob Solver 2
Subprob Solver 3
Subprob Solver 4

Link Coster 1
Link Coster 2
Link Coster 3

• cost batches of driver-load links, as assigned by the farmer
Parallelization opportunities

- Data flowing from the farmer to the solvers:
  » Before any iteration begins
    • Updated value functions
    • Updated dynamic attributes of exogenous drivers and loads
  » Before each subproblem is solved
    • Time and location of the subproblem to be solved
    • Endogenous drivers and loads to be added to that subproblem
  » As each subproblem is solved
    • Driver-load candidate assignments that have been costed

- Data flowing from the solvers to the farmer as subproblems are solved:
  • Driver-load candidate assignments to be costed
  • Implemented decisions
  • Endogenous drivers and loads to be added to a later subproblem
  • Marginal values to be used to update value functions
Parallelization opportunities

- Data flowing from the *farmer* to the *costers*:
  - Before any iteration begins
    - Updated value functions (home repositioning)
  - As each subproblem is solved
    - Driver-load candidate assignments to be costed

- Data flowing from the *costers* to the *farmer* as subproblems are solved:
  - Driver-load candidate assignments that have been costed
Parallelization opportunities

- Data transfer:
  » Sockets
  » BiteArrays or primitive data types

- Process launching
  » Through the farmer
  » Independent
Parallelization opportunities

- **Pros** of centralizing control and data flow through the *farmer*:
  - Simplified communication network
  - Flexibility in the assignment of processors to tasks
  - Maintenance of information that cannot be easily decomposed (e.g., updates of value functions)
  - Easier recovery in case of processor failure

- **Cons**:
  - Possible data flow bottleneck
  - Greater effort required in streamlining data transfer
  - *Setup cost* of distributing information centrally maintained
Outline

- Review of the optimizing simulator for large truckload networks
- Why and where to parallelize and distribute the computation
- Comments on results and issues
Geography-based parallelization

Variation of the Speedup Ratio with the # of Processors

Data set 2 has twice as many subproblems as data sets 1 and 3
Some comments

» Slowdown for 1 processor, due to the overhead of the data transfer

» Speedup ratio peaks between 6 or 7 processors, possibly due to the overhead of the distribution of centralized data to a higher # of processors

» Implementation written in Java: garbage collection affects parallelization

» More testing is necessary
Geography-based parallelization

Graph follows the life of a geography through the time intervals using different number of processors $\Rightarrow$ garbage collection (GC) hits at different places

- GC effect on one geography is small
- GC in a geography affects all geographies because of sync at the end of a time interval
- the more geographies processed in parallel, the more likely that one will have a GC affecting all