BeeJamA
Honey Bee Inspired Traffic Jam Avoidance

Highly Dynamic and Adaptive Traffic Congestion Avoidance in Real-Time Inspired by Honey Bee Behavior


08.12.2007
Content

- Introduction
- Distributed Traffic Model
  - Quality Rating Function
  - Distributed Traffic Jam Avoidance
    (The BeeJamA Algorithm)
- Simulation Studies
  - Experimental Results
- Conclusion
- Future Work

08.12.2007
Introduction

- Traffic congestion – a highly dynamic distributed problem in metropolitan areas world-wide.
- Serious and complex problems regarding the timely arrival of goods or persons.
- Expertise in developing distributed routing algorithms from Nature (*BeeHive*, *BeeAdHoc*)
  - Dynamic multipath routing algorithms
  - High adaptability / flexibility
  - High throughput
  - High fault tolerance
- Distributed *on-line* traffic control model → *BeeJamA*
Distributed Traffic Model

- Individual hop-to-hop routing
- Probabilistic next hop determination
- Regionally responsible *navigators*
  - Management of local routing tables
  - Real-time response
- Layered routing model
  - *Area-Layer*: nodes correspond to intersections, edges correspond to roads
  - *Net-Layer*: nodes correspond to areas, edges represent roads connecting adjacent areas

Figure 1: Routing Layers

08.12.2007
Distributed Traffic Model (Partitioning)

**Net-Layer**: partitioning into *Foraging Zones* ($\rightarrow \text{IFZ}_{\text{Net}}$-table) and *Foraging Regions* ($\rightarrow \text{IFR}_{\text{Net}}$-table)

- FZ($N$): all nodes (areas) within a limited hop range of node (area) $N$
- FR: fixed partitions with a single *Representative Node* (area)

**Area-Layer**: partitioning into areas

- Size must allow for sufficient routing alternatives
- Real-time constraints must be met
- FZ: each node’s FZ consists of all nodes within its area ($\rightarrow \text{IFZ}_{\text{Area}}$-table)
- *Border-nodes* connect adjacent areas
Quality Rating Function

- Routing tables reflect estimated travel times
- Extensive empirical studies in traffic congestion development
- Functional dependency between vehicle speed and density on a road section

\[ v_{\text{edge}} = \frac{(A - \alpha)}{\alpha^2} \rho^2 + v_{\text{max}} \]

\[ v_{\text{edge}} = \frac{(\rho - \alpha)(B - A)}{\beta - \alpha} + A \]

\[ v_{\text{edge}} = \frac{\beta^3}{\rho^3} B \]

08.12.2007
The *BeeJamA Algorithm*

**Flowchart:**
1. **Lookup destination node in FRM\textsubscript{Area} table**
2. **Is dest. node within current area?**
   - NO: **Lookup dest. region's representative node in FRM\textsubscript{Net}**
   - YES: Move!
3. **Is dest. node's area in current area's FZ?**
   - NO: Get next hop area to dest. region's representative node
   - YES: Get next hop area to dest. node
4. **At destination?**
   - NO: Get next hop to border-node to dest. area
   - YES: Move!
The BeeJamA Algorithm

Vehicle driving from node 4 (area A) to node 14 (area B)

- Get next hop to border-node to dest. area
- Get next hop to dest. node
- Move!

At destination?

NO

Is dest. node?

Net-Layer

Area-Layer

Lookup destination node in FRM_{Area} table

Lookup dest. region’s representative node in FRM_{Net}

Get next hop area to dest. area

08.12.2007
The *BeeJamA* Algorithm

**Table:**

<table>
<thead>
<tr>
<th>FRM\textsubscript{Area}</th>
<th>1</th>
<th>...</th>
<th>7</th>
<th>8</th>
<th>...</th>
<th>14</th>
<th>15</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>...</td>
<td>A</td>
<td>B</td>
<td>...</td>
<td>B</td>
<td>E</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Flowchart:**

1. **Look up destination node in FRM\textsubscript{Area} table**
2. Check if destination node is within the current area.
   - **Yes:** Move to the next hop to the destination node.
   - **No:** Check if at destination.
     - **Yes:** Stop.
     - **No:** Continue.
3. **At destination?**
   - **Yes:** Stop.
   - **No:** Continue.
The BeeJamA Algorithm

1. Look up destination node in FRM_{Area} table.
2. Is destination node within current area? 
   - Yes: Move to destination.
   - No: Look up region's representative node in FRM_{Net}.
3. Is destination node's area within current area's FZ? 
   - Yes: Get next hop to destination.
   - No: Get next hop to border-node to destination.

Net-Layer

Area-Layer
The BeeJamA Algorithm

1. **Lookup destination node in FRM/Area table**
2. **Is dest. node within current area?**
   - **YES**
     - Get next hop area to border-node to dest. area
   - **NO**
     - **Is dest. node's area in current area's FZ?**
       - **YES**
         - Get next hop area to dest. region's representative node
       - **NO**
         - **At destination?**
           - **YES**
             - Move!
           - **NO**
             - Get next hop to border-node to dest. area

Diagram:

- Areas: IFZ\textsubscript{Net} (A), B, C, E, F, H, I
- Connections: B to C\textsubscript{BB}, C\textsubscript{CB}, C\textsubscript{EB}, C\textsubscript{FB}, C\textsubscript{HB}, C\textsubscript{IB}, E to C\textsubscript{BE}, C\textsubscript{CE}, C\textsubcript{EE}, C\textsubscript{FE}, C\textsubscript{HE}, C\textsubscript{IE}
4→B (via 6)

The BeeJamA Algorithm

Get next hop to dest. node

Get next hop to border-node to dest. area

Is dest. node’s area in current area’s FZ?

NO

YES

Get next area to dest. area

IFZ\_Area  

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(c_{11})</td>
<td>(c_{21})</td>
<td>(c_{31})</td>
<td>(c_{41})</td>
<td>(c_{51})</td>
<td>(c_{61})</td>
<td>(c_{71})</td>
</tr>
<tr>
<td>3</td>
<td>(c_{13})</td>
<td>(c_{23})</td>
<td>(c_{33})</td>
<td>(c_{43})</td>
<td>(c_{53})</td>
<td>(c_{63})</td>
<td>(c_{73})</td>
</tr>
<tr>
<td>5</td>
<td>(c_{15})</td>
<td>(c_{25})</td>
<td>(c_{35})</td>
<td>(c_{45})</td>
<td>(c_{55})</td>
<td>(c_{65})</td>
<td>(c_{75})</td>
</tr>
</tbody>
</table>
The BeeJamA Algorithm

1. Lookup destination node in FRM<sub>Area</sub> table
2. Is dest. node within current area?
   - NO: Lookup dest. region's representative node in FRM<sub>Net</sub>
   - YES: Get next hop area to dest. area
3. Is dest. node's area in current area's FZ?
   - NO: Get next hop area to border-node
   - YES: Get next hop to dest. node
4. Move!
5. At destination?
   - NO: Repeat from step 1
   - YES: End
Simulation Studies

Traffic Simulator

- Realistic traffic model (Nagel/Schreckenberg)
- Cellular automaton based
- Commercially available navigational data (AND)
- Section of the German Ruhr District

Source Nodes: A, B, C, D
Destination Node: E
New Vehicles per Second: 4 (1 per Node)
Simulation Time: 3600 seconds
Dijkstra Update Interval: 600 seconds
Tempo Limits: 135 km/h (Freeways), 85 km/h (Highways)
Max. Speed for Vehicles: 135 km/h

<table>
<thead>
<tr>
<th>Vehicular Density Limits</th>
<th>Highways</th>
<th>$\alpha=35$, $\beta=40$ [vehicles/km], $A=50$, $B=10$ [km/h]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4-Lane-Freeways</td>
<td>$\alpha=40$, $\beta=55$ [vehicles/km], $A=70$, $B=30$ [km/h]</td>
</tr>
<tr>
<td></td>
<td>6-Lane-Freeways</td>
<td>$\alpha=45$, $\beta=65$ [vehicles/km], $A=75$, $B=35$ [km/h]</td>
</tr>
</tbody>
</table>

Figure 3: Realistic Section of the Ruhr District

Table 1: Experimental Setup
Experimental Results

- Traffic congestion avoidance with respect to individual travel times
  - BeeJamA routing against Dijkstra-based fastest path routing (10min update interval)
Conclusion

- We developed our own simulator, due to the lack of software for dynamic, distributed routing algorithms in vehicular traffic networks.
- Distributed layered traffic model based on BeeHive/BeeAdHoc routing.
- Dynamic cost model: Minimal travel times to destinations, congestion avoidance as objectives.
- Although there is no global information the navigators work very efficiently.
- *Demonstration available after the session*
Future Work

- Distributed on-line simulator
  - Realistic communication modeling
  - Dynamic rule sets
  - OpenStreetMap integration
  - Optimal off-line clustering of navigation areas

- Hardware implementation and evaluation
  - OpenMoko handsets
  - Deadline granularity 1 sec.

- Online version of the simulator:
Thank you!