Chapter 15: Network Applications

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

ECLiPSe ELearning

Licence

This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.
How can we get better performance out of a given network?

- Make network transparent
  - Users should not need to know about details
  - Service maintained even if failures occur
- Restricted by accepted techniques available in hardware
  - Interoperability between multi-vendor equipment
  - Very conversative deployment strategies
Reminder: IP Networks

- Packet forwarding
- Connection-less
- Destination based routing
  - Distributed routing algorithm based on shortest path algorithm
  - Routing metric determines preferred path
- Best effort
  - Packets are dropped when there is too much traffic on interface
  - Guaranteed delivery handled at other layers (TCP/applications)

Disclaimer

- Flexible border between CP and OR
- CP is ...
  - what CP people do.
  - what is published in CP conferences.
  - what uses CP languages.
- Does not mean that other approaches are less valid!
Example Network (Uniform metric 1, Capacity 100)

Example Traffic Matrix

Only partially filled in for example

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Using Routing

Demand AC 10

A → R1 → R2 → C

Demand AD 20

A → R1 → R2 → C

Helmut Simonis
Network Applications
Using Routing

Demand BC 10

Using Routing

Demand BD 20
Using Routing

Demand AE 20

A ➔ R1 ➔ R2 ➔ C
B ➔ R3 ➔ R4 ➔ D

Demand BE 20

A ➔ R1 ➔ R2 ➔ C
B ➔ R3 ➔ R4 ➔ D
### Traffic Placement

- **Capacity Management**
- **Other Problems**

### Link Based Model

- **Path-Based Model**
- **Node-Based Model**
- **Commercial Solution**
- **Multiple Paths**

#### Resulting Network Load

![Diagram showing network load](chart)

- **A** to **R1**: 50 units
- **R1** to **R2**: 30 units
- **R2** to **C**: 15 units
- **R1** to **E**: 55 units
- **C** to **R1**: 15 units
- **E** to **R4**: 15 units
- **R4** to **D**: 5 units

#### Considering failure of R1-E

![Diagram showing network load with R1-E failure](chart)

- **A** to **R1**: 35 units
- **R1** to **R2**: 35 units
- **R2** to **C**: 15 units
- **R1** to **E**: 10 units
- **E** to **R4**: 40 units
- **R4** to **D**: 20 units
- **D** to **R3**: 65 units
- **R3** to **R1**: 15 units
- **R1** to **A**: 10 units

*Helmut Simonis  Network Applications*
Can we do better?

- Choose single, explicit path for each demand
- Requires hardware support in routers (MPLS-TE)
- Baseline: CSPF, greedy heuristic

Why not just use Multi-Commodity Flow Problem Solution?

- Can not use arbitrary, fractional flows in hardware
- MILP does not scale too well
Modelling Alternatives

- Link based Model
- Path based Model
- Node based Model

Variants

- Demand Acceptance
  - Choose which demands to select fitting into available capacity
- Traffic Placement
  - All demands must be placed
**Link-Based Model: Intuition**

- Decide if demand $d$ is run over link $e$
- Select which demands run over link $e$ (Knapsack)
- Demand $d$ must run from source to sink (Path)
- Sum of delay on path should be limited (QoS)

---

**Link Based Model**

\[
\min_{\{X_{de}\}} \max_{e \in E} \frac{1}{\text{cap}(e)} \sum_{d \in D} \text{bw}(d)X_{de} \quad \text{or} \quad \min_{\{X_{de}\}} \sum_{e \in E, d \in D} \text{bw}(d)X_{de}
\]

st.

\[\forall d \in D, \forall n \in N : \sum_{e \in \text{OUT}(n)} X_{de} - \sum_{e \in \text{IN}(n)} X_{de} = \begin{cases} -1 & n = \text{dest}(d) \\ 1 & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases}\]

\[\forall e \in E : \sum_{d \in D} \text{bw}(d)X_{de} \leq \text{cap}(e)\]

\[\forall d \in D : \sum_{e \in E} \text{del}(e)X_{de} \leq \text{req}(d)\]

\[X_{de} \in \{0, 1\}\]
Solution Methods

- Lagrangian Relaxation
  - Path decomposition
  - Knapsack decomposition
- Probe Backtracking

Lagrangian Relaxation - Path decomposition

[Ouaja&Richards2003]

- Dualize capacity constraints
- Starting with CSPF initial solution
- Finite domain solver for path constraints
- Added capacity constraints from st-cuts
- At each step solve shortest path problems
Lagrangian Relaxation - Knapsack decomposition

[Ouaja&Richards2005]
- Dualize path constraints
- At each step solve knapsack problems
- Reduced cost based filtering

Probe Backtracking

[Liatsos et al 2003]
- Start with (infeasible) CSPF heuristic
- Consider capacity violation
  - Resolve by forcing one demand off/on link
  - Find new path respecting path and added constraints with ILP
- Repeat until no more violations, feasible solution
- Optimality proof when exhausted search space
  - Search space often very small
Path-Based Model: Intuition

- Choose one of the possible paths for demand $d$
- This paths competes with paths of other demands for bandwidth
- Usually too many paths to generate a priori, but most are useless

\[
\max \{Z_d, Y_{id}\} \sum_{d \in D} \text{val}(d)Z_d
\]

\[\text{st.}\]

\[\forall d \in D : \sum_{1 \leq i \leq \text{path}(d)} Y_{id} = Z_d\]

\[\forall e \in E : \sum_{d \in D} \text{bw}(d) \sum_{1 \leq i \leq \text{path}(d)} h_{id}Y_{id} \leq \text{cap}(e)\]

\[Z_d \in \{0, 1\}\]

\[Y_{id} \in \{0, 1\}\]
Solution Methods

- Blocking Islands
- Local Search/ Finite Domain Hybrid
- (Column Generation)

[Frei&Faltings1999]
- Feasible solution only
- CSP with variables ranging over paths for demands
- No explicit domain representation
- Possible to perform forward checking by updating blocking island structure
Local Search/Finite Domain Hybrid

[Lever2004]

- Start with (feasible) CSPF heuristic
- Add more demands one by one
  - Use repair to solve capacity violations
- Use Finite Domain model to check necessary conditions
  - Determine bottlenecks by st-cuts
  - Force paths on/off links
- Define neighborhood by rerouting demands currently over violations

Node Based Model: Intuition

- For each demand, decide for each router where to go next
  - Many routers not used
- Treat link capacity with cumulative/diffn constraints
- Pure Finite Domain model, no global cost view
Path placement algorithm developed for Cisco by PTL and IC-Parc (2002-2004)
- Internal competitive selection of approaches
- Strong emphasis on stability
- Written in ECLiPSe
- PTL bought by Cisco in 2004
- Part of team moved to Boston

What happens if element on selected path fails?
- Choose second path which is link (element) disjoint
- State bandwidth constraints for each considered failure case
- **Problem**: Very large number of capacity constraints
Example

Primary/Secondary path for demand AE

Which bandwidth to count?

<table>
<thead>
<tr>
<th>Failed Element Capacity for Path</th>
<th>No Failure</th>
<th>A-R1</th>
<th>R1-E</th>
<th>All Others</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Primary</td>
<td>Secondary</td>
<td>Secondary</td>
<td>Primary</td>
</tr>
</tbody>
</table>
Multiple Path Model

\[
\max \{Z_d, X_{de}, W_{de}\} \sum_{d \in D} \text{val}(d)Z_d
\]

\[
\forall d \in D, \forall n \in N : \sum_{e \in \text{OUT}(n)} X_{de} - \sum_{e \in \text{IN}(n)} X_{de} = \begin{cases} -Z_d & n = \text{dest}(d) \\ Z_d & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases}
\]

\[
\forall e \in E : \sum_{d \in D} \text{bw}(d) \cdot X_{de} \leq \text{cap}(e)
\]

\[
\forall d \in D, \forall n \in N : \sum_{e \in \text{OUT}(n)} W_{de} - \sum_{e \in \text{IN}(n)} W_{de} = \begin{cases} -Z_d & n = \text{dest}(d) \\ Z_d & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases}
\]

\[
\forall e \in E, \forall e' \in E \setminus e : \sum_{d \in D} \text{bw}(d) \cdot (X_{de} - X_{de'} \cdot X_{de} + X_{de'} \cdot W_{de}) \leq \text{cap}(e)
\]

\[
\forall d \in D, \forall e \in E : X_{de} + W_{de} \leq 1
\]

\[
Z_d \in \{0, 1\}, X_{de} \in \{0, 1\}, W_{de} \in \{0, 1\}
\]

Solution Method

- Benders Decomposition [Xia&Simonis2005]
- Use MILP for standard demand acceptance problem
- Find two link disjoint paths for each demand
- Sub-problems consist of capacity constraints for failure cases
- Benders cuts are just no-good cuts for secondary violations
The Problem

- How to provide cost effective, high quality services running an IP network?
- Easy to build high quality network by massive over-provisioning
- Easy to build consumer grade network disregarding Quality of Service (QoS)
- Very hard to right-size a network, providing just enough capacity

The Approach

- Bandwidth on Demand
  - Create temporary bandwidth channels for high-value traffic
  - Avoid disturbing existing traffic
- Resilience Analysis
  - Find out how much capacity is required for current traffic
  - Provide enough capacity to survive element failures without service disruption
Background

- Failures of network should not affect services running on network
- Not cost effective to protect connections in hardware
- Response time is critical
  - Interruption > 50ms not acceptable for telephony
  - Reconvergence of IGP 1 sec (good setup)
  - Secondary tunnels rely on signalling of failure (too slow)
  - Live/Live connections too expensive

Approach

- Fast Re-route
  - If element fails, use detour around failure
  - Local repair, not global reaction
  - Pre-compute possible reactions, allows offline optimization
- Link protection rather easy
- Node protection quite difficult
Example Problem

Node j Failure

Traffic Placement
Capacity Management
Other Problems

Bandwidth Protection
Bandwidth on Demand
Resilience Analysis

Helmut Simonis
Network Applications
Node j Failure (Result)

Bandwidth Protection Model
Solution Techniques

[Xia, Eremin & Wallace 2004]

- MILP
  - Use of Karush-Kahn-Tucker condition
  - Removal of nested optimization
  - Large set of new variables
  - Not scalable

- Problem Decomposition
  - Integer Multi-Commodity Flow Problem
  - Capacity Optimization

- Improved MILP out-performs decomposition [Xia 2005]

Cisco Tunnel Builder Pro

- Algorithm/Implementation built by PTL/IC-Parc for Cisco
- Not based on published techniques above
- In period 2000-2003
- Written in ECLiPSe
- Embedded in Java GUI
- Now subsumed by ISC-TEM
Planning Ahead

- Consider demands with fixed start and end times
- Demands overlapping in time compete for bandwidth
- Demands arrive in batches, not always in temporal sequence
- Problem called Bandwidth on Demand (BoD)

Model: BoD

\[
\max \{Z_d, X_{de}\} \sum_{d \in D} \text{val}(d)Z_d
\]

\[
\text{st.}
\]

\[
T = \{\text{start}(d) | d \in D\}
\]

\[
\forall d \in D, \forall n \in N : \sum_{e \in \text{OUT}(n)} X_{de} - \sum_{e \in \text{IN}(n)} X_{de} = \begin{cases} -Z_d & n = \text{dest}(d) \\ Z_d & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases}
\]

\[
\forall t \in T, \forall e \in E : \sum_{d \in D_{\text{start}(d) \leq t, t < \text{end}(d)}} \text{bw}(d)X_{de} \leq \text{cap}(e)
\]

\[
Z_d \in \{0, 1\}
\]

\[
X_{de} \in \{0, 1\}
\]
Solution Methods

- Schlumberger Dexa.net (PTL, IC-Parc)

Schlumberger Dexa.net

- Small, but global MPLS TE+diffserv network
- Oil field services
- (Very) High value traffic
  - Well logging
  - Video conferencing
- Bandwidth demand known well in advance, fixed period
- Low latency, low jitter required
Customer requests capacity for time slot via Web-interface
- Demand Manager determines if request can be satisfied
  - Based on free capacity predicted by Resilience Analysis
  - Taking other, accepted BoD requests into account
- Email back to customer
- At requested time, DM triggers provisioning tool to
  - Set up tunnel
  - Change admission control
- At end of period, DM pulls down tunnel
How much free capacity do we have in network?

- Easy for normal network state (OSS tools)
- Challenge: How much is required for possible failure scenarios?
- Consider single link, switch, router, PoP failures
- **Classical solution**
  - Get Traffic Matrix
  - Run scenarios through simulator

How to get a Traffic Matrix?

- Many algorithms assume given traffic matrix
- Traffic flow information is not collected in the routers
- Only link traffic is readily available
- Demand pattern changes over time, often quite dramatically
- Measuring traffic flows with probes is very costly

From a network consultant:

We have been working on extracting a TM for this network for 15 months, and we still don’t have a clue if we’ve got it right.
Idea

- Use the observed traffic to deduce traffic flows
- **Network Tomography** [Vardi1996]
  - All flows routed over a link cause the observed traffic
  - Must correct for observation errors
  - Highly dependent on accurate routing model
- **Gravity Model** [Medina et al 2002]
  - Ignore core of network
  - Assume that flows are proportional to product of ingress/egress size
- Results are very hard to validate/falsify

---

Model: Traffic Flow Analysis

\[
\forall i, j \in N : \quad \min \{F_{ij}\} / \max \{F_{ij}\}
\]

s.t.

\[
\forall e \in E : \quad \sum_{i,j \in N} r_{ij} F_{ij} = \text{traf}(e)
\]

\[
\forall i \in N : \quad \sum_{j \in N} F_{ij} = \text{ext}^{in}(i)
\]

\[
\forall j \in N : \quad \sum_{i \in N} F_{ij} = \text{ext}^{out}(j)
\]

\[
F_{ij} \geq 0
\]
Start with Link Traffic

Setup Model to Find Flows

\[ \begin{align*}
\{AC, AD, BC, BD, AE, BE\} & : 0.0 \ldots 1.0 & \text{Inf}, \\
AC + AD + AE & = 50, \% A R1 \\
0.5 \times BC + 0.5 \times BD + BE & = 35, \% B R1 \\
0.5 \times BC + 0.5 \times BD & = 15, \% B R3 \\
AD + 0.5 \times BD & = 30, \% R1 R2 \\
AC + 0.5 \times BC + AE + BE & = 55, \% R1 E \\
AD + 0.5 \times BD & = 30, \% R2 D \\
0.5 \times BC + 0.5 \times BD & = 15, \% R3 R4 \\
AC + 0.5 \times BC & = 15, \% E C \\
0.5 \times BC & = 5, \% R4 C \\
0.5 \times BD & = 10, \% R4 D
\end{align*} \]
Solve for Different Flows

\[
\begin{align*}
\min(A,C,\min AC), \max(A,C,\max AC), \\
\min(A,D,\min AD), \max(A,D,\max AD), \\
\min(B,C,\min BC), \max(B,C,\max BC), \\
\min(B,D,\min BD), \max(B,D,\max BD), \\
\min(A,E,\min AE), \max(A,E,\max AE), \\
\min(B,E,\min BE), \max(B,E,\max BE), \\
... 
\end{align*}
\]

Results of Analysis

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

Problem solved, no?
### Benchmark Problems

<table>
<thead>
<tr>
<th>Network</th>
<th>Routers</th>
<th>PoPs</th>
<th>Lines</th>
<th>Lines/router</th>
</tr>
</thead>
<tbody>
<tr>
<td>dexa</td>
<td>51</td>
<td>24</td>
<td>59</td>
<td>1.15</td>
</tr>
<tr>
<td>as1221</td>
<td>108</td>
<td>57</td>
<td>153</td>
<td>1.41</td>
</tr>
<tr>
<td>as1239</td>
<td>315</td>
<td>44</td>
<td>972</td>
<td>3.08</td>
</tr>
<tr>
<td>as1755</td>
<td>87</td>
<td>23</td>
<td>161</td>
<td>1.85</td>
</tr>
<tr>
<td>as3257</td>
<td>161</td>
<td>49</td>
<td>328</td>
<td>2.03</td>
</tr>
<tr>
<td>as3967</td>
<td>79</td>
<td>22</td>
<td>147</td>
<td>1.86</td>
</tr>
<tr>
<td>as6461</td>
<td>141</td>
<td>22</td>
<td>374</td>
<td>2.65</td>
</tr>
</tbody>
</table>

### TFA Result for Benchmarks

<table>
<thead>
<tr>
<th>Network</th>
<th>Low Simul (%)</th>
<th>High Simul (%)</th>
<th>Obj</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dexa</td>
<td>0</td>
<td>2310.65</td>
<td>1190</td>
<td>11</td>
</tr>
<tr>
<td>as1221</td>
<td>0.09</td>
<td>8398.64</td>
<td>11556</td>
<td>1318</td>
</tr>
<tr>
<td>as1239</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>as1755</td>
<td>0.15</td>
<td>6255.31</td>
<td>7482</td>
<td>699</td>
</tr>
<tr>
<td>as3257</td>
<td>0.04</td>
<td>12260.03</td>
<td>25760</td>
<td>12389</td>
</tr>
<tr>
<td>as3967</td>
<td>0.1</td>
<td>5387.10</td>
<td>6162</td>
<td>500</td>
</tr>
<tr>
<td>as6461</td>
<td>0.28</td>
<td>8688.39</td>
<td>19740</td>
<td>8676</td>
</tr>
</tbody>
</table>
Reduce Problem Size

- Pop Level Analysis
- Only consider flows between PoPs, not routers
- Local area connections typically not bottlenecks
- Modelling routing can be tricky

### PoP Level Results

<table>
<thead>
<tr>
<th>Network</th>
<th>( \text{Low Simul} ) (%)</th>
<th>( \text{High Simul} ) (%)</th>
<th>Obj</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dexa</td>
<td>0</td>
<td>1068.37</td>
<td>557</td>
<td>5</td>
</tr>
<tr>
<td>as1221</td>
<td>0.24</td>
<td>2964.93</td>
<td>3205</td>
<td>424</td>
</tr>
<tr>
<td>as1239</td>
<td>0.63</td>
<td>1401.72</td>
<td>1931</td>
<td>101359</td>
</tr>
<tr>
<td>as1755</td>
<td>0.66</td>
<td>1263.28</td>
<td>526</td>
<td>103</td>
</tr>
<tr>
<td>as3257</td>
<td>0.30</td>
<td>2028.73</td>
<td>2378</td>
<td>2052</td>
</tr>
<tr>
<td>as3967</td>
<td>0.1</td>
<td>1209.37</td>
<td>483</td>
<td>90</td>
</tr>
<tr>
<td>as6461</td>
<td>1.47</td>
<td>951.41</td>
<td>481</td>
<td>768</td>
</tr>
</tbody>
</table>
Increase Accuracy

- LSP Counters
  - In MPLS networks only, provide improved resolution
  - Implementation buggy, not all counters can be used
- Netflow
  - Collect end-to-end flow information in router
  - Impact on router (memory)
  - Impact on network (data aggregation)

### TFA with LSP Counters

<table>
<thead>
<tr>
<th>Network</th>
<th>Low Simul (%)</th>
<th>High Simul (%)</th>
<th>Obj</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dlexa</td>
<td>30.35</td>
<td>249.71</td>
<td>1190</td>
<td>7</td>
</tr>
<tr>
<td>as1221</td>
<td>9.94</td>
<td>685.37</td>
<td>11556</td>
<td>885</td>
</tr>
<tr>
<td>as1239</td>
<td>10.74</td>
<td>1151.03</td>
<td>98910</td>
<td>72461</td>
</tr>
<tr>
<td>as1755</td>
<td>25.29</td>
<td>269.30</td>
<td>7482</td>
<td>397</td>
</tr>
<tr>
<td>as3257</td>
<td>23.77</td>
<td>425.67</td>
<td>25760</td>
<td>5121</td>
</tr>
<tr>
<td>as3967</td>
<td>24.47</td>
<td>300.17</td>
<td>6162</td>
<td>275</td>
</tr>
<tr>
<td>as6461</td>
<td>19.43</td>
<td>477.44</td>
<td>19740</td>
<td>2683</td>
</tr>
</tbody>
</table>
PoP TFA with LSP Counters

<table>
<thead>
<tr>
<th>Network</th>
<th>Low Simul (%)</th>
<th>High Simul (%)</th>
<th>Obj</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dax</td>
<td>60.62</td>
<td>145.85</td>
<td>557</td>
<td>3</td>
</tr>
<tr>
<td>as1221</td>
<td>28.49</td>
<td>499.16</td>
<td>3205</td>
<td>271</td>
</tr>
<tr>
<td>as1239</td>
<td>33.36</td>
<td>211.84</td>
<td>1931</td>
<td>2569</td>
</tr>
<tr>
<td>as1755</td>
<td>50.33</td>
<td>169.37</td>
<td>526</td>
<td>46</td>
</tr>
<tr>
<td>as3257</td>
<td>36.82</td>
<td>249.16</td>
<td>2378</td>
<td>640</td>
</tr>
<tr>
<td>as3967</td>
<td>40.72</td>
<td>182.97</td>
<td>483</td>
<td>36</td>
</tr>
<tr>
<td>as6461</td>
<td>34.05</td>
<td>210.93</td>
<td>481</td>
<td>136</td>
</tr>
</tbody>
</table>

Helmut Simonis  
Network Applications  
Network Applications  69

What now?

- Choose some particular solution?
- Which one? How to validate assumptions?
- Massively under-constrained problem
  - $|N|^2$ variables
  - $|E| + 2|N|$ constraints
  - $2|N|^2$ queries
- Ill-conditioned even after error correction
- Aggregation helps
  - We are usually not interested in individual flows
  - We want to use the TM to investigate something else
Resilience Analysis

- How much capacity is needed to survive all reasonable failures?
- Use normal state as starting point
- Consider routing in each failure case
- Aggregate flows in rerouted network
- Calculate bounds on traffic in failure case

Mathematical Model: Resilience Analysis

\[
\forall e \in E : \min \left\{ \frac{F_{ij}}{F_{ij}} \right\} \sum_{i,j \in N} r_{ij}^F F_{ij}
\]

subject to:

\[
\forall e \in E : \sum_{i,j \in N} r_{ij}^F F_{ij} = \text{traf}(e)
\]

\[
\forall i \in N : \sum_{j \in N} F_{ij} = \text{ext}^i_n(i)
\]

\[
\forall j \in N : \sum_{i \in N} F_{ij} = \text{ext}^o(u)(j)
\]

\[
F_{ij} \geq 0
\]
### Results over 100 runs

<table>
<thead>
<tr>
<th>Network</th>
<th>lower bound/simul average</th>
<th>lower bound/simul stdev</th>
<th>upper bound/simul average</th>
<th>upper bound/simul stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>dexta</td>
<td>91.50</td>
<td>0.14</td>
<td>108.28</td>
<td>0.16</td>
</tr>
<tr>
<td>as1755</td>
<td>88.65</td>
<td>0.11</td>
<td>106.08</td>
<td>0.056</td>
</tr>
<tr>
<td>as3967</td>
<td>94.08</td>
<td>0.073</td>
<td>106.88</td>
<td>0.091</td>
</tr>
<tr>
<td>as1221</td>
<td>87.34</td>
<td>0.10</td>
<td>102.05</td>
<td>0.025</td>
</tr>
</tbody>
</table>
## Results with LSP counters

<table>
<thead>
<tr>
<th>Network</th>
<th>Low Simul (%)</th>
<th>High Simul (%)</th>
<th>Obj</th>
<th>Time</th>
<th>Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>dexa</td>
<td>97.76</td>
<td>101.33</td>
<td>3503</td>
<td>36</td>
<td>59</td>
</tr>
<tr>
<td>as1221</td>
<td>98.15</td>
<td>100.69</td>
<td>14191</td>
<td>1840</td>
<td>153</td>
</tr>
<tr>
<td>as1239</td>
<td>99.37</td>
<td>100.38</td>
<td>4499</td>
<td>3974</td>
<td>10</td>
</tr>
<tr>
<td>as1755</td>
<td>99.28</td>
<td>100.66</td>
<td>8409</td>
<td>964</td>
<td>161</td>
</tr>
<tr>
<td>as3257</td>
<td>99.41</td>
<td>100.44</td>
<td>31093</td>
<td>13381</td>
<td>328</td>
</tr>
<tr>
<td>as3967</td>
<td>98.88</td>
<td>101.00</td>
<td>9090</td>
<td>819</td>
<td>147</td>
</tr>
<tr>
<td>as6461</td>
<td>99.44</td>
<td>100.52</td>
<td>24808</td>
<td>8006</td>
<td>374</td>
</tr>
</tbody>
</table>

## Results over 100 runs (with LSP Counters)

<table>
<thead>
<tr>
<th>Network</th>
<th>lower bound/simul</th>
<th>upper bound/ simul</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>average</td>
<td>stdev</td>
</tr>
<tr>
<td>dexa</td>
<td>99.60</td>
<td>0.029</td>
</tr>
<tr>
<td>as1755</td>
<td>99.31</td>
<td>0.016</td>
</tr>
<tr>
<td>as3967</td>
<td>99.41</td>
<td>0.014</td>
</tr>
<tr>
<td>as1221</td>
<td>98.10</td>
<td>0.025</td>
</tr>
</tbody>
</table>
Perspectives

- High polynomial complexity
- Possible to reduce number of queries
  - Small differences between failure cases
  - Many queries are identical or dominated
- Possible to reduce size of problem dramatically
- Integrate multiple measurements in one model
- **Which other problems can we solve without explicit TM?**

Problem

- Which links should be used to build network structure?
- Link speed is related to cost
- Model simple generalization of path finding
- Assumptions about routing in target network?
Model

\[
\min_{\{X_{de}, W_{ie}\}} \sum_{e \in E} \sum_{1 \leq i \leq \text{alt}(e)} \text{cost}(i, e) W_{ie}
\]

\[
\forall d \in D, \forall n \in N:\quad \sum_{e \in \text{OUT}(n)} X_{de} - \sum_{e \in \text{IN}(n)} X_{de} = \begin{cases} -1 & n = \text{dest}(d) \\ 1 & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases}
\]

\[
\forall e \in E:\quad \sum_{d \in D} \text{bw}(d) X_{de} \leq \sum_{1 \leq i \leq \text{alt}(e)} \text{cap}(i, e) W_{ie}
\]

\[
\forall e \in E:\quad \sum_{1 \leq i \leq \text{alt}(e)} W_{ie} = 1
\]

\[
W_{ie} \in \{0, 1\}
\]

\[
X_{de} \in \{0, 1\}
\]

Issues

- Real-life problem not easily modelled
- Possible choices/costs not easily obtained (outside US)
- Choices often are inter-related
- Package deals by providers
- Some regions don’t allow any flexibility at all
Problem

- How to set weights in IGP to avoid bottlenecks?
- Easy to beat default values
- Single/equal cost paths required/allowed/forbidden?

Model

$$
\min_{\{Y_{id}, W_e\}} \quad \max_{e \in E} \frac{1}{\text{cap}(e)} \sum_{d \in D} \text{bw}(d) \sum_{1 \leq i \leq \text{path}(d)} h^0_{id} Y_{id}
$$

st.

$$
\forall d \in D : \quad \sum_{1 \leq i \leq \text{path}(d)} Y_{id} = 1
$$

$$
\forall d \in D, 1 \leq i \leq \text{path}(d) : \quad P_{id} = \sum_{e \in E} h^0_{id} W_e
$$

$$
\forall d \in D, 1 \leq i, j \leq \text{path}(d) : \quad P_{id} = P_{jd} \implies Y_{id} = Y_{jd} = 0
$$

$$
\forall d \in D, 1 \leq i, j \leq \text{path}(d) : \quad P_{id} < P_{jd} \implies Y_{jd} = 0
$$

$$
Y_{id} \in \{0, 1\}
$$

integer \quad W_e \geq 1

integer \quad P_{id} \geq 0$$
Solution Methods

- Methods tested at IC-Parc
  - Branch and price
  - Tabu search
  - Set constraints
- Very hard to compete with (guided) local search

Further Reading

Network problems can be solved competitively by constraint techniques.

Hybrid methods required, simple Finite Domain models usually don’t work.

Constraint based tools commercial reality.

Open Problems
- How to make this easier to develop?
- How to make this more stable to solve?