MCMCF
A Tool for Network Design

Jeffrey D. Oldham
Department of Computer Science
Stanford University
oldham@cs.stanford.edu

1997 November 18

Joint work with Andrew Goldberg,
Serge Plotkin, and Cliff Stein.
A Multicommodity Flow Example

Specify:

- network topology
- edge costs
- peak call demand

Goal: Satisfy peak demand with minimum cost.

Peak Demands

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LA–Dallas</td>
<td>35 calls</td>
<td></td>
</tr>
<tr>
<td>LA–NYC</td>
<td>80 calls</td>
<td></td>
</tr>
<tr>
<td>Dallas–NYC</td>
<td>70 calls</td>
<td></td>
</tr>
</tbody>
</table>

LA Dallas

30 calls

$3 per call

NYC

80 calls

$1 per call

100 calls

$1 per call

Jeffrey D. Oldham (oldham@cs.stanford.edu)
Linear Programming Based Solution

Disadvantages:

**Size:**  Problem specification:  \( O(k + m) \) space

  Linear programs:  \( O(k(n + m)) \) variables
                       \( O(kn + m) \) inequalities

  - \( n \) is the number of nodes.
  - \( m \) is the number of edges.
  - \( k \) is the number of commodities.

**LP solution time:**

  - experimentally quadratic in \( k \)
  - experimentally quadratic in network size

**design tradeoff:**

  - slow, exact solution
  - fast approximation
Combinatorial Solution

Combinatorial program **MCMCF**: 

**\( \epsilon \)-approximation:**

- flow uses at most \((1 + \epsilon)\) edge capacity
- flow cost at most \((1 + \epsilon)\) minimum cost

**Main idea:**

- reduce to single-commodity problems
- relate commodities using potential function

**Theoretical advantage:**

- time: \(\tilde{O}(\epsilon^{-3}k)\) (time for min-cost flow)
- space: \(O(k(n + m))\)

**Practical advantages:**

- trade off time for accuracy
The Potential Function

Problem:
Several objectives:

- minimize total cost
- capacity constraints for every edge

Not smooth!

Solution:
Aggregate into smooth potential function $\phi$

\[
\phi = \exp \left( \alpha \left( \frac{\text{flow's cost}}{\text{desired cost}} \right) \right) + \sum_{\text{edges } e} \exp \left( \alpha \left( \frac{\text{flow}(e)}{\text{capacity}(e)} \right) \right)
\]

small $\phi \quad \Rightarrow \quad$ good solution
Outline of the Algorithm

Goal: Reduce potential function $\phi$.

Main ideas:
- Move in direction $(-\nabla \phi)$.
- Maintain flow satisfying demands.

Until $\epsilon$-optimal solution found:
1. Choose a commodity to improve.
2. Compute $\nabla \phi$.
3. Use $\nabla \phi$ as edge costs.
5. Improvement step: $(1 - \sigma)f + \sigma f^*$. 

Jeffrey D. Oldham (oldham@cs.stanford.edu)
Implementing the Algorithm

Direct implementation runs slower than LP.

Problem:

- pessimistic parameters which guarantee progress but not practical progress

Solution:

Use theory to yield practical modifications:

- Dynamically adjust the step size $\sigma$.
- Dynamically adjust $\alpha$.
- Compute lower bound to determine when solution is $\epsilon$-optimal.
- Restart MCF routine using previous flow.
Choosing the Step Size $\sigma$

Improvement step:

$$(1 - \sigma)f + \sigma f^*.$$ 

Theory:

- fixed step size $\sigma = O(\varepsilon^{-3})$

Practice:

- Compute $\sigma$ to minimize potential function.
- Use Newton-Raphson method.
- Newton requires first and second derivatives.

Result: (Sun Enterprise 3000)

<table>
<thead>
<tr>
<th>instance</th>
<th>$\varepsilon$</th>
<th>time (seconds)</th>
<th>Newton</th>
<th>theoretical</th>
</tr>
</thead>
<tbody>
<tr>
<td>rmfgen-d-4-12-020</td>
<td>0.01</td>
<td>64</td>
<td></td>
<td>3842</td>
</tr>
<tr>
<td>rmfgen-d-7-10-020</td>
<td>0.01</td>
<td>257</td>
<td></td>
<td>15203</td>
</tr>
<tr>
<td>multigrid-008-016-0100</td>
<td>0.01</td>
<td>3</td>
<td></td>
<td>95</td>
</tr>
</tbody>
</table>

Jeffrey D. Oldham (oldham@cs.stanford.edu)
Comparisons with Linear Programming

MCMCF

$\epsilon$-approximation:
- Flow uses at most $(1 + \epsilon)$ edge capacity
- Flow cost at most $(1 + \epsilon)$ minimum cost

CPLEX

dual simplex:
- exact solutions

primal simplex
- permits stopping to yield $\epsilon$-approximation
- experimentally 10x slower than dual

Comparisons performed on a Sun UltraSparc-2.
Dependence on $k$

Multigrid Instances

![Graph showing the dependence of running time on the number of commodities for different solvers, CPLEX and MCMCF (1%).]
Dependence on $k$ (cont’d)

Rmfgem Instances

![Graph showing the dependence of running time on the number of commodities](image)

- CPLEX
- MCMCF (1%)

Running Time (min)

Number of Commodities ($k$)

Jeffrey D. Oldham (oldham@cs.stanford.edu)
Dependence on Problem Size

Tripartite Instances

Running Time (min)

Number of Vertices

CPLEX

MCMCF (2%)
Dependence on the Approximation $\epsilon$

The dependence is asymptotically $O(\epsilon^{-1.5})$. 

Rmfggen Instances
Conclusions

theoretical algorithm

- theoretically fast
- practically slower than LP

practical modifications

- guided by theory

resulting advantages

- yield fast, provably correct implementation
- faster than all other algorithms
- solve larger problems than all other algorithms
- fast approximations—good for design
- trade time for accuracy