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

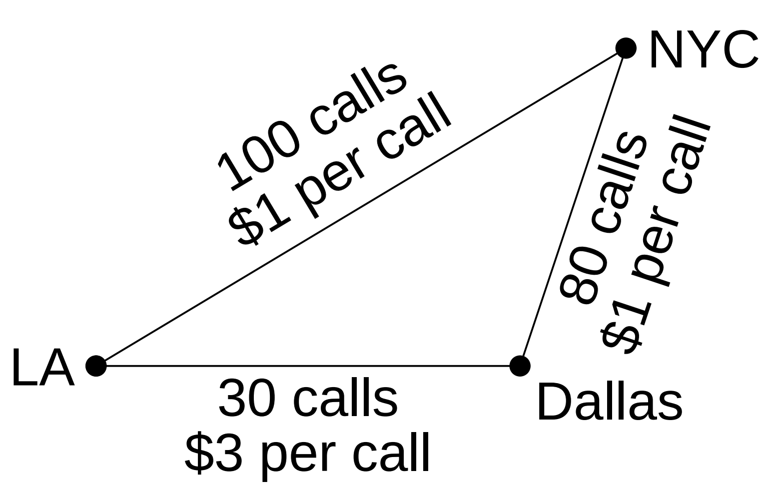
Specify:

- network topology
- edge costs
- peak call demand

Goal: Satisfy peak demand with minimum cost.

Peak Demands

LA–Dallas	35 calls
LA–NYC	80 calls
Dallas–NYC	70 calls



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**:

ϵ-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 ϕ

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

small $\phi \quad \Rightarrow \quad$ good solution

Outline of the Algorithm

Goal: Reduce potential function ϕ.

Main ideas:

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

Until ϵ-optimal solution found:

1. Choose a commodity to improve.
2. Compute $\nabla\phi$.
3. Use $\nabla\phi$ as edge costs.
4. Compute single-commodity minimum-cost flow f^*.
5. Improvement step: $(1-\sigma)f + \sigma f^*$.

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 σ.
- Dynamically adjust α.
- Compute lower bound to determine when solution is ϵ-optimal.
- Restart MCF routine using previous flow.

Choosing the Step Size σ

Improvement step:

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

Theory:

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

Practice:

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

Result: (Sun Enterprise 3000)

		time (seconds)	
instance	ϵ	Newton	theoretical
rmfgen-d-4-12-020	0.01	64	3842
rmfgen-d-7-10-020	0.01	257	15203
multigrid-008-016-0100	0.01	3	95

Comparisons with Linear Programming

MCMCF

ϵ-approximation:

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

CPLEX

dual simplex:

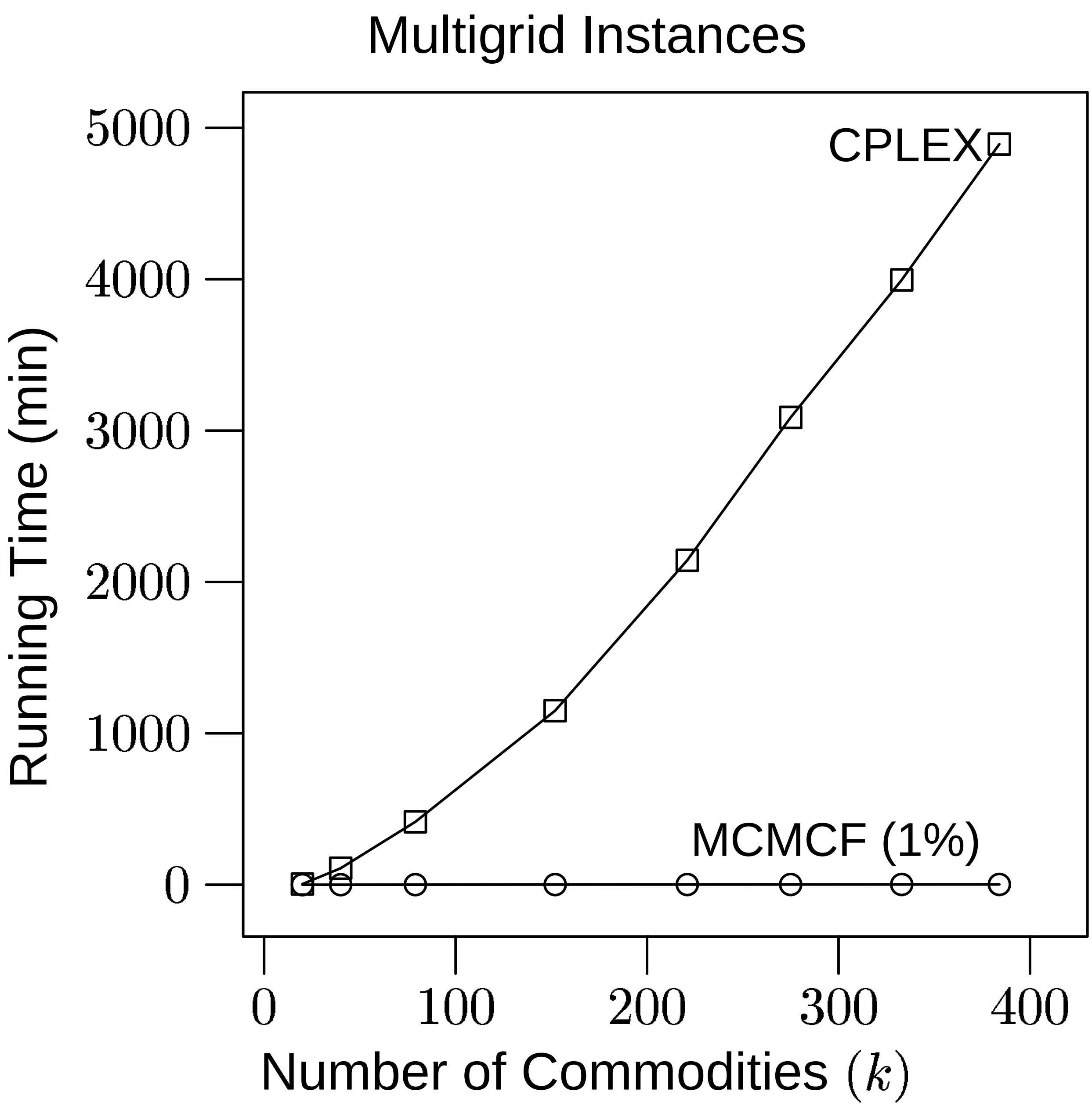
- exact solutions

primal simplex

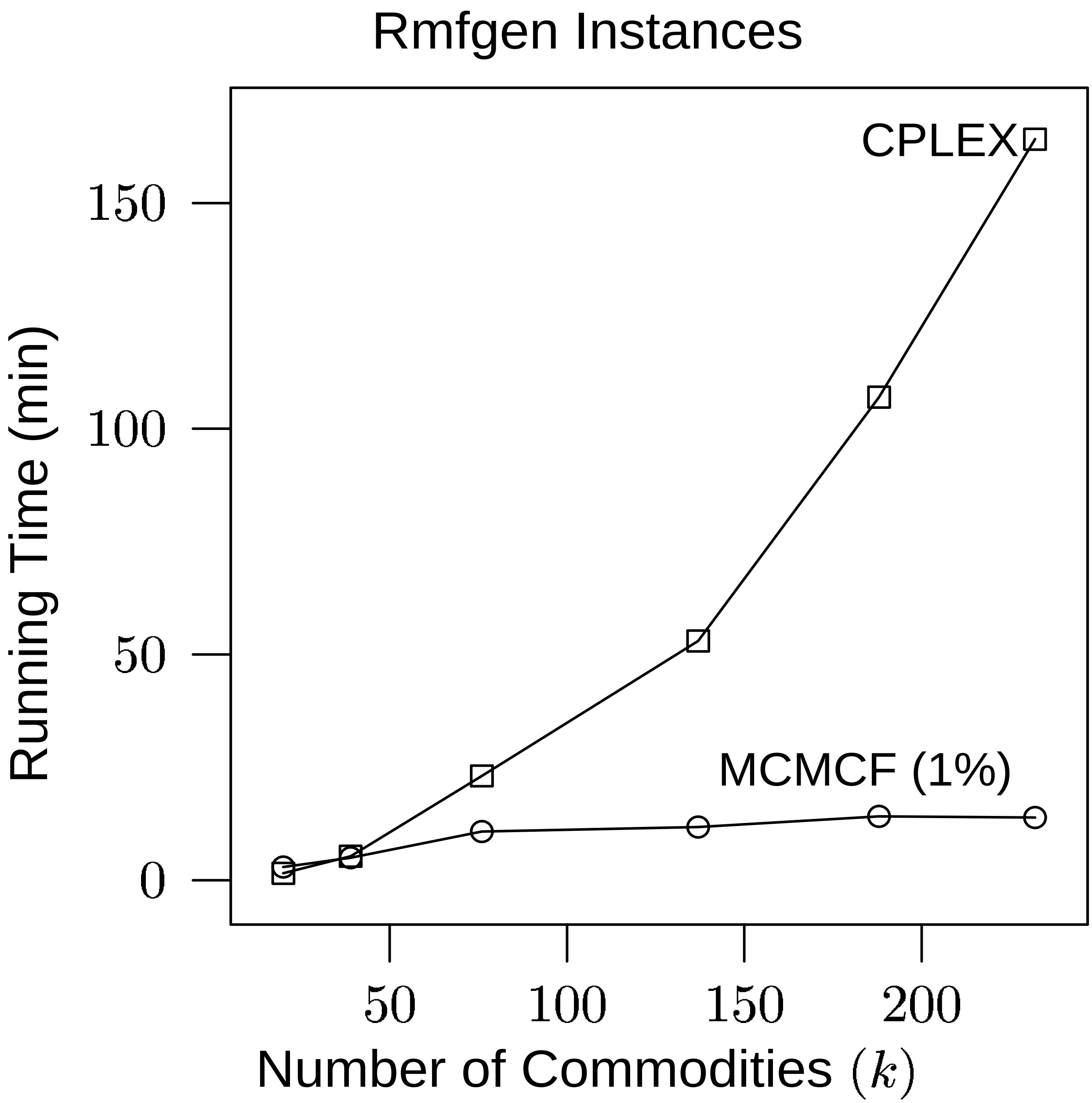
- permits stopping to yield ϵ-approximation
- experimentally 10x slower than dual

Comparisons performed on a Sun UltraSparc-2.

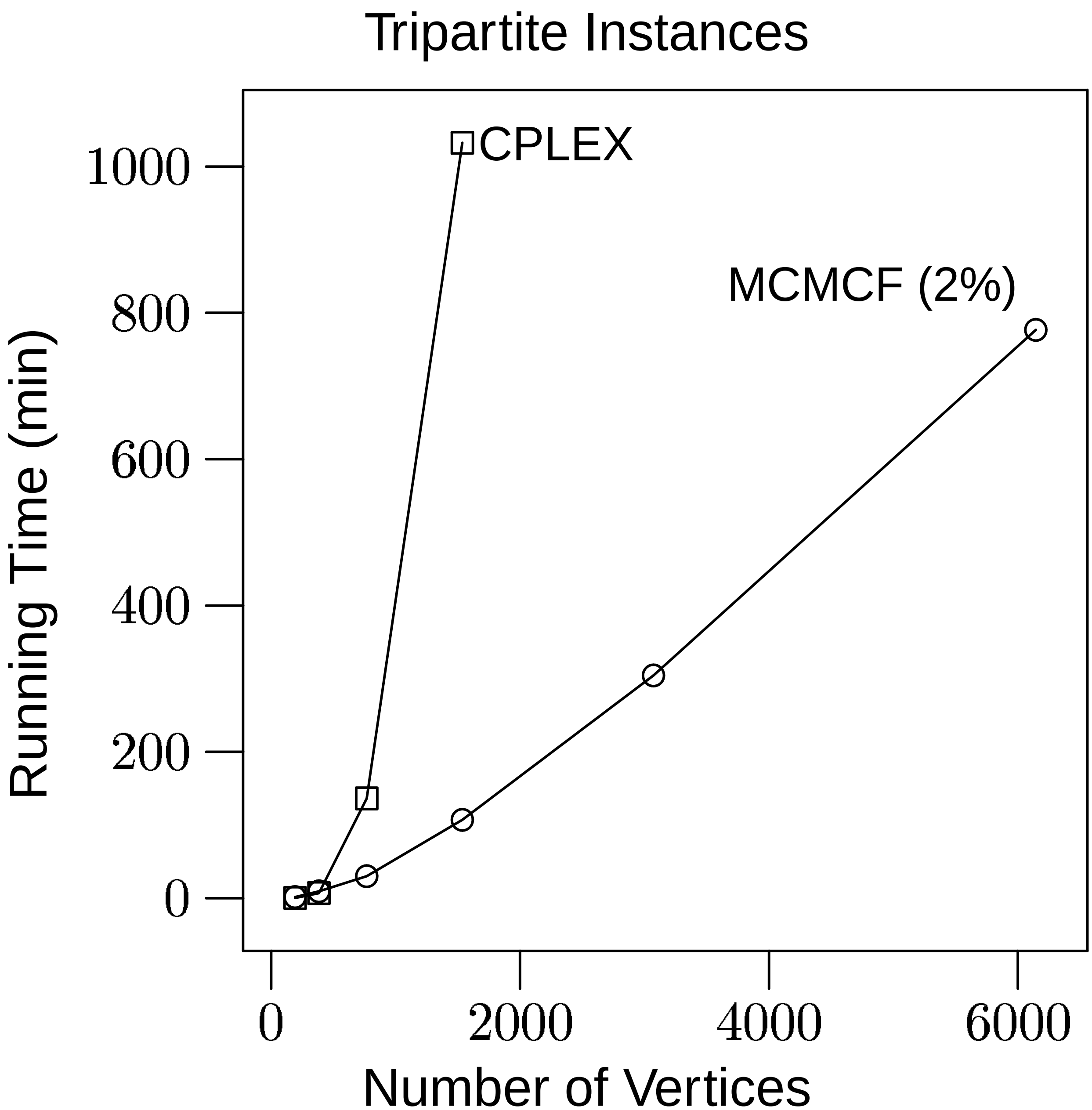
Dependence on k



Dependence on k (cont'd)

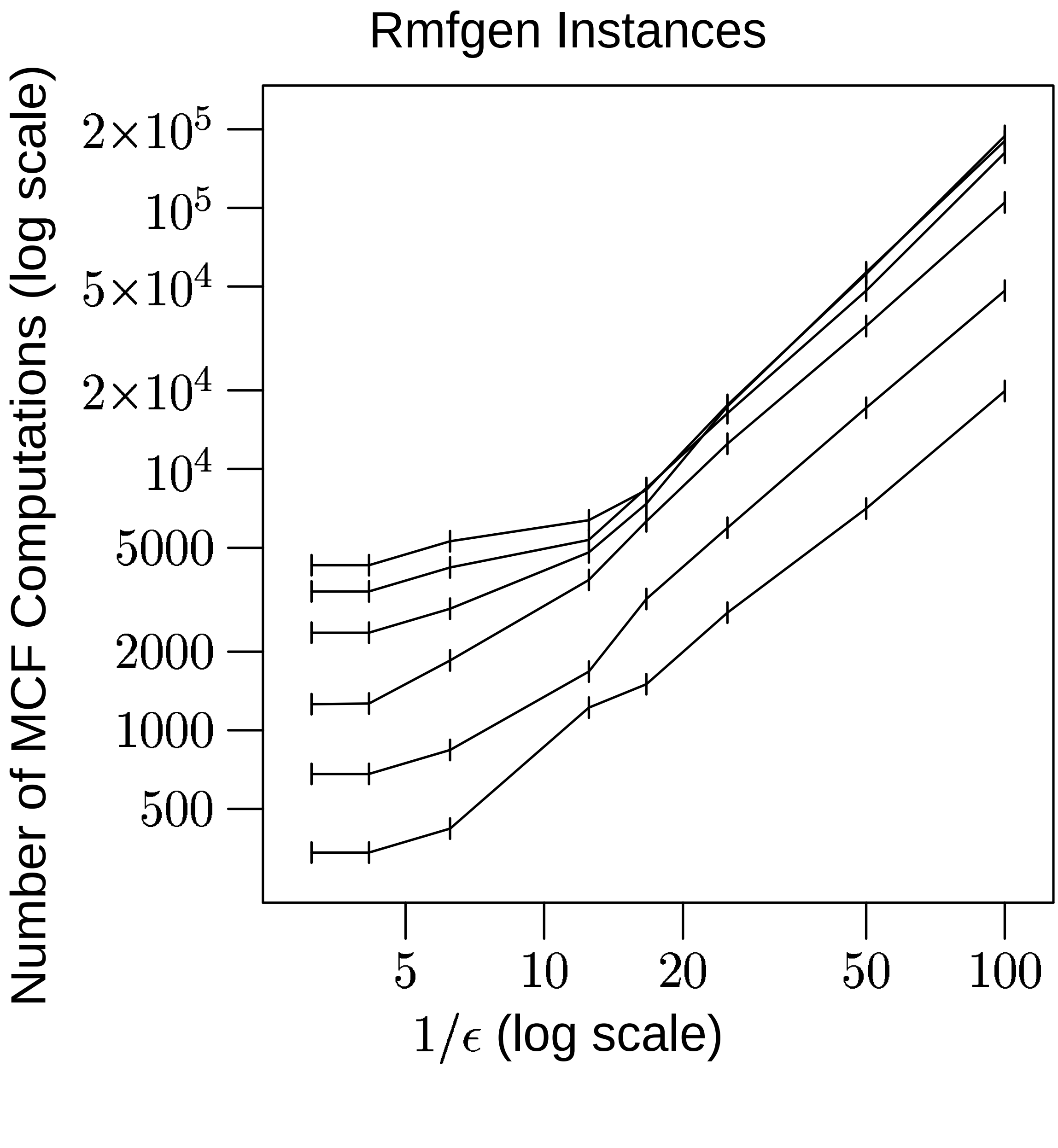


Dependence on Problem Size



Dependence on the Approximation ϵ

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



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