MCMCF
A Tool for Network Design

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

1997 September 04

A Multicommodity Flow Example

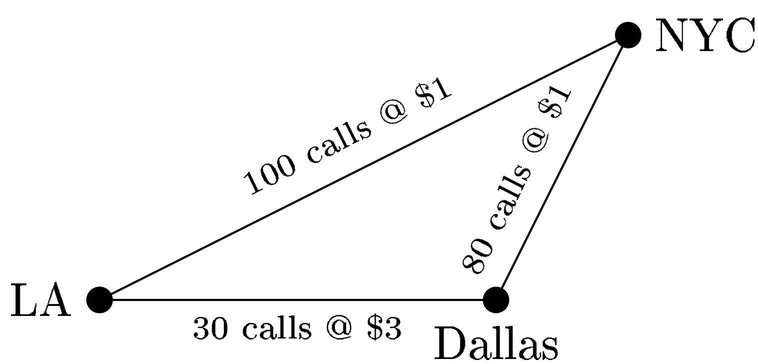
Specify:

- 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



Use: `MCMCF` (Minimum-Cost MultiCommodity Flow)

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 arcs.
- 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 $(1 + \epsilon)$ arc capacity
- flow cost at most $(1 + \epsilon)$ minimum cost

Main idea:

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

Theoretical advantage:

- time: $\tilde{O}(k)$(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 arc

Not smooth!

Solution:

Aggregate into smooth potential function ϕ

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

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 arc costs.
4. Compute single-commodity minimum-cost flow f^*.
5. Improvement step: $(1-\sigma)f + \sigma f^*$.

The Algorithm's Running Time

The theoretical running time is

$$\tilde{O}(\epsilon^{-3}k)(\text{time for min-cost flow})$$

[Karger and Plotkin, 1995],
[Plotkin, Shmoys, Tardos, 1995],
[Leighton et al., 1995].

Advantages:

- (almost) linear dependence on number k of commodities
- uses well-understood single-commodity flow subroutine

The Algorithm's Running Time (cont'd)

Direct implementation runs slower than LP.

Problem:

- pessimistic parameters
- guarantee progress but not practical progress

Solution:

- dynamically adjust parameters

Key Idea:

- use theory to yield practical modifications

Problem Instances

Problem instances from two different families:

multigrid

- two-dimensional grids
- few additional arcs

rmfgen

- series of two-dimensional grids
- connect grids via random node permutation

Commodity sources, sinks, demands randomly chosen.

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)

instance	ϵ	time (seconds) Newton	time (seconds) 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

Choosing α

Constant α in potential function:

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

Theory:

- fixed (large) value $\Rightarrow$ guarantee progress
- progress inversely proportional to α

Practice:

- choose (smaller) value guaranteeing progress
- compute occasionally—expensive

Result: (Sun Enterprise 3000)

		time (seconds)	
instance	ϵ	adaptive	theoretical
rmfgen-d-7-10-020	0.01	56	161
rmfgen-d-7-10-240	0.03	238	738
multigrid-032-128-0080	0.01	42	47

Updating MCF Routine

Theory:

- Use any minimum-cost flow routine.

Practice:

- Costs and capacities do not vary much.
- Simplex MCF can update from feasible flow.
- Use commodity's current flow.

Result: (Sun UltraSPARC-2)

		time (seconds)	
instance	ϵ	updating	no updating
rmfgen-d-7-10-020	0.01	87	180
rmfgen-d-7-10-240	0.01	454	835
multigrid-032-128-0080	0.01	21	37

Small Incremental Flow Change

Theory:

- Flow can change on all arcs.

Practice:

- Flow changes on few arcs.
- Routines for σ use only nonzero differences.

Result: (Sun UltraSPARC-2)

instance	ϵ	time (seconds) use nonzero	use all
rmfgen-d-7-10-020	0.01	87	203
rmfgen-d-7-10-240	0.01	454	972
multigrid-032-128-0080	0.01	21	33

Termination Criteria

Stop algorithm when have ϵ-optimal solution.

Theory:

- small $\phi \quad \Rightarrow \quad \epsilon$-optimal

Practice:

- Compute a lower bound using LP dual.
- Compute occasionally—k MCF computations

Comparisons with Linear Programming

`MCMCF`

ϵ-approximation:

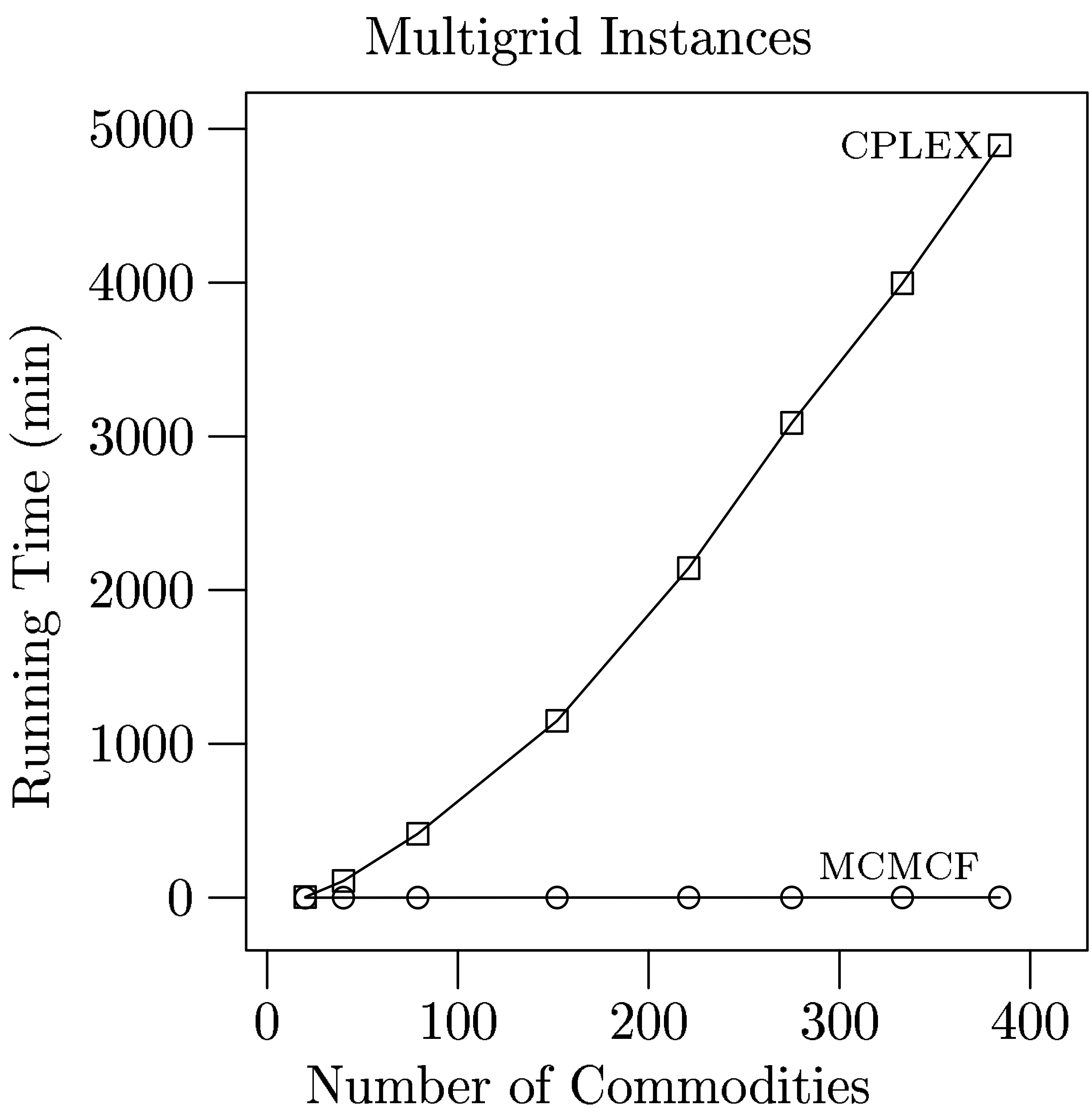
- Flow uses at most $(1 + \epsilon)$ arc capacity
- Flow cost at most $(1 + \epsilon)$ minimum cost

`CPLEX`

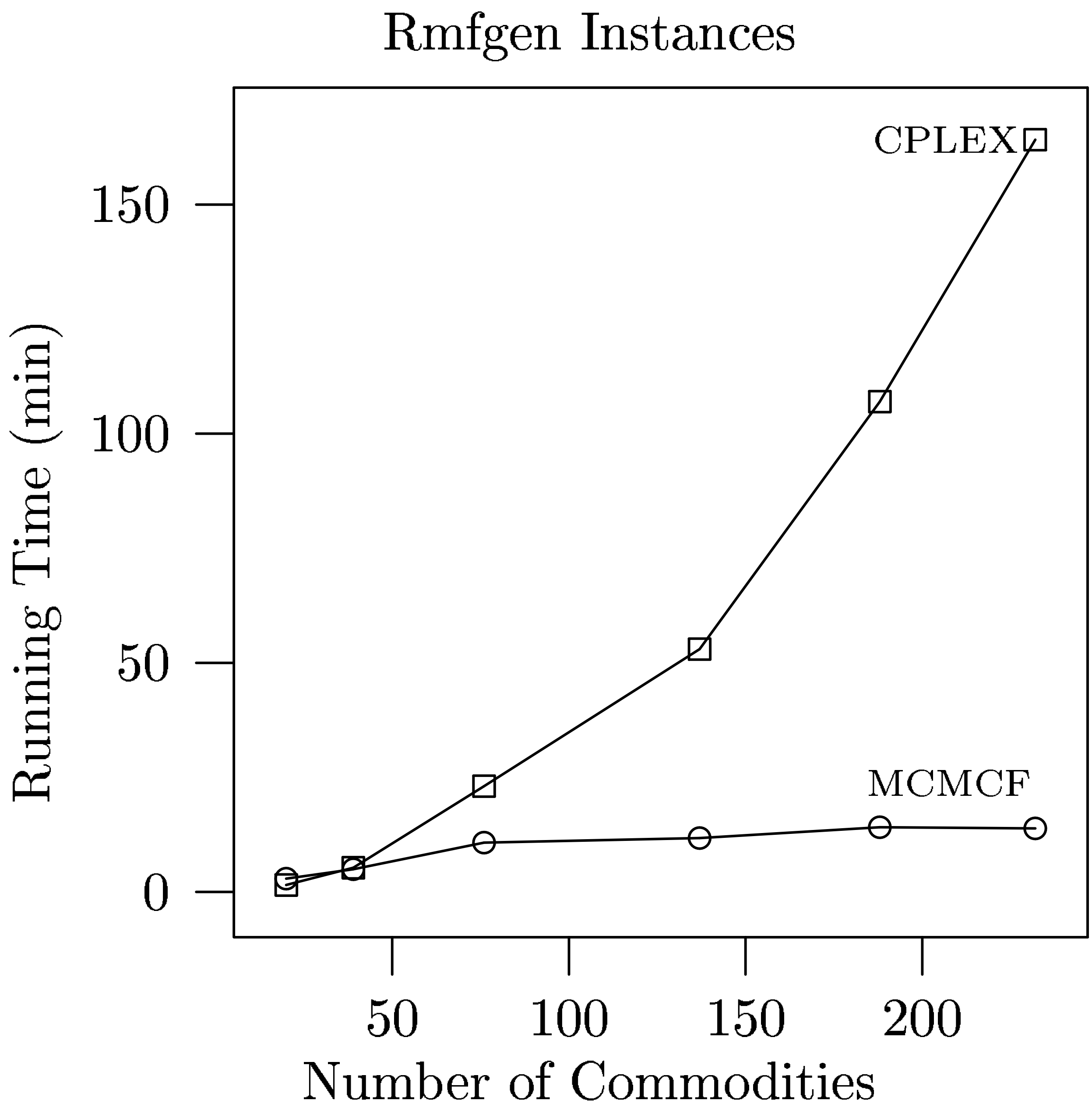
dual simplex: exact solutions
primal simplex

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

Dependence on k

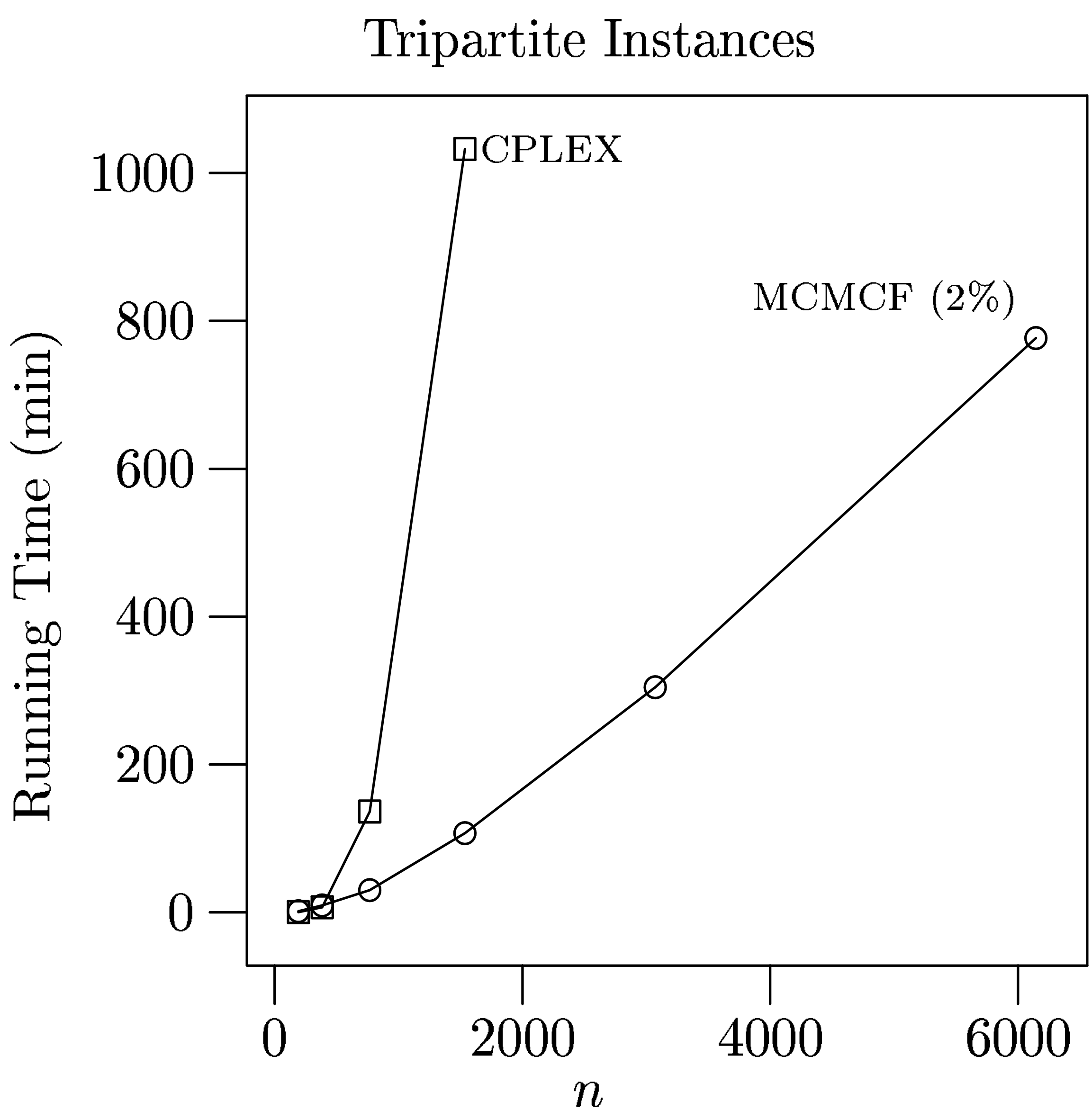


Dependence on k (cont'd)



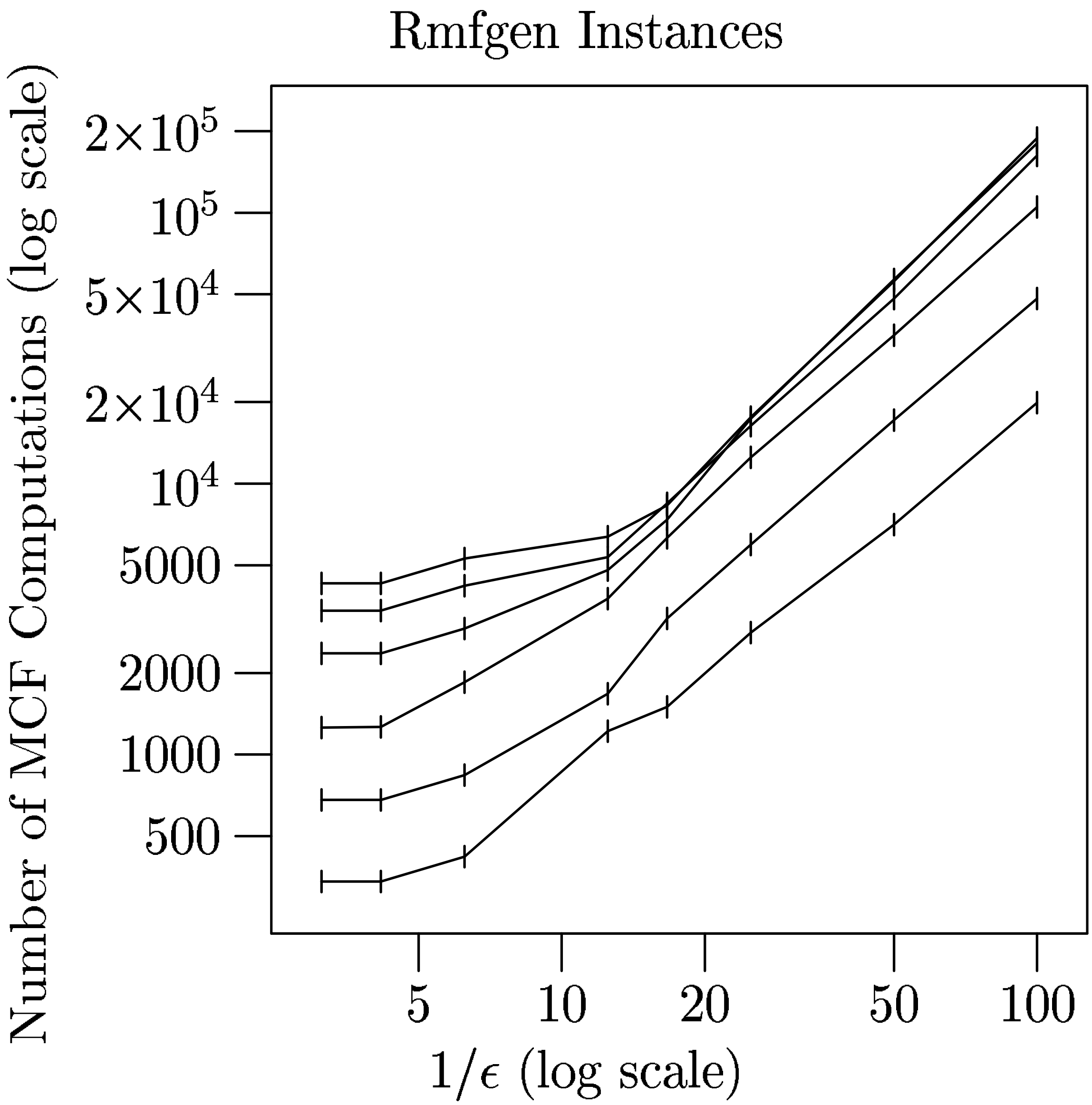
Dependence on Problem Size

The tripartite generator was designed to produce problems difficult for `MCMCF` to solve.



Dependence on the Approximation ϵ

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



Conclusions

theoretical algorithm

- theoretically fast
- practically slower than LP

practical modifications

- guided by theory
- yield fast, provably correct implementation

resulting advantages

- faster than all other algorithms
- solve larger problems than all other algorithms
- fast approximations—good for design
- trade time for accuracy