A Simpler Generalized Shortest Paths Algorithm

Jeffrey D. Oldham
Department of Computer Science
Stanford University
oldham@cs.stanford.edu
http://theory.stanford.edu/˜oldham/

1998 October 01

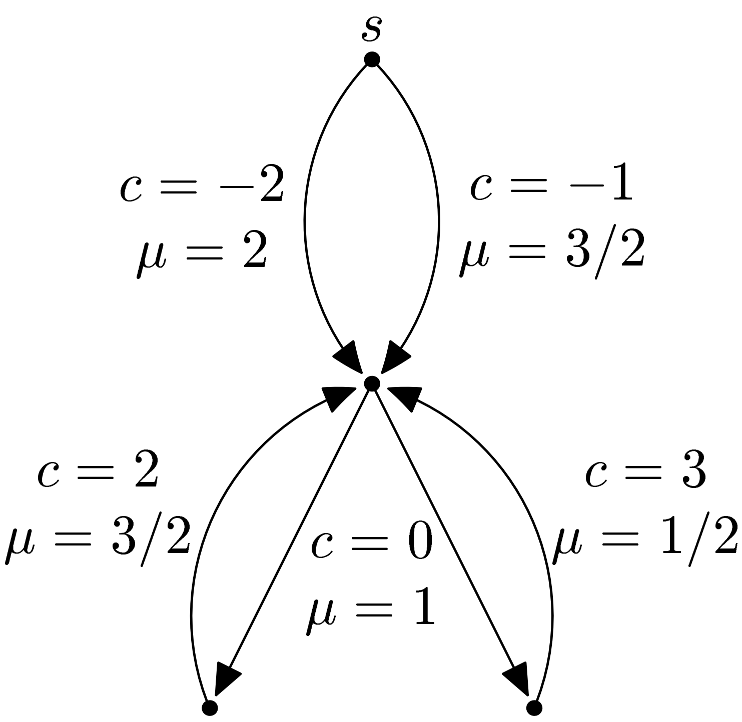
generalized		binary search
+	=	+
shortest paths		Bellman-Ford

An Example

Input:

- vertices and arcs
- arc costs
- flow multipliers
- source vertex

Goal: Remove one unit of flow from source as cheaply as possible.



Modeling Using Generalized Flows

Using flow multipliers permits:

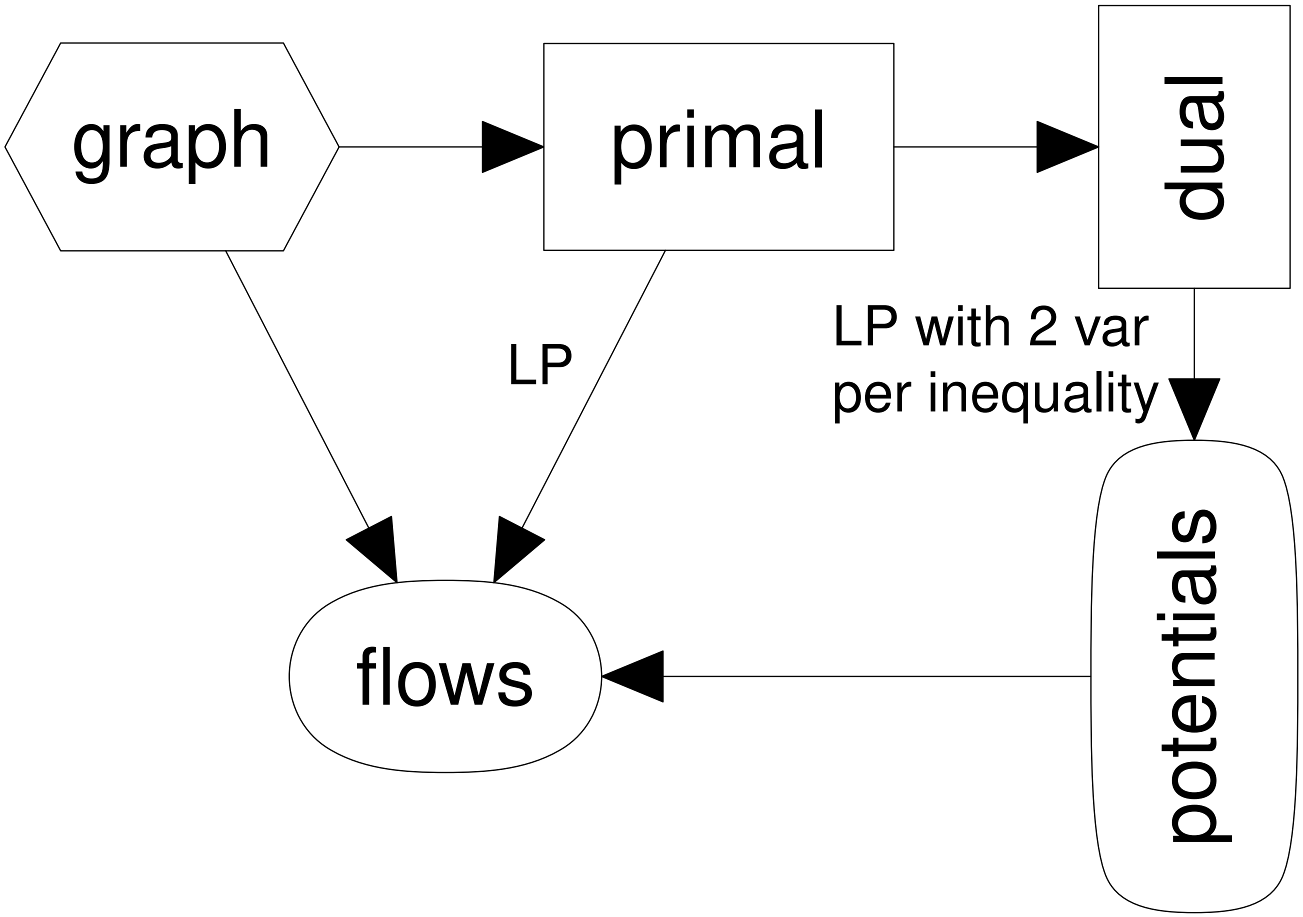
Conversions of Entities:

- Convert from one unit to another.
- Ex: trees to sheets of paper
- Ex: coal to electricity
- Currency conversion network
  - one directed arc for every exchange rate
  - arc costs of zero
  - How can one lose $1 to market inefficiencies?

Modifying Flow Amounts:

- Modify the flow amount of a particular item.
- Ex: breakage during shipment
- Ex: evaporation from water canals
- Ex: accrual of bank interest

Approaches to Solving the Problem



The Main Idea

generalized		binary search		parametric search
+	=	+	=	+
shortest paths		Bellman-Ford		Bellman-Ford

Reduced Costs

dual linear program:

max $\pi(s)$

s.t. $(\forall (v,w) \in A)\,(\pi(v) - \mu(v,w)\pi(w) \leq c(v,w))$.

potential $\pi(v)$ equals cost of eliminating one unit of flow starting at vertex v

reduced cost:

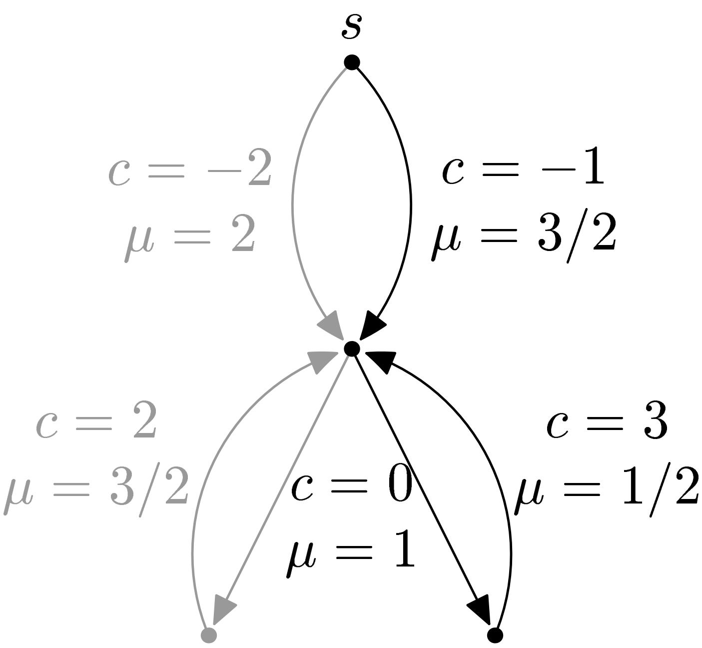
$$c^{\pi}(v,w) = c(v,w) + \mu(v,w)\pi(w) - \pi(v).$$

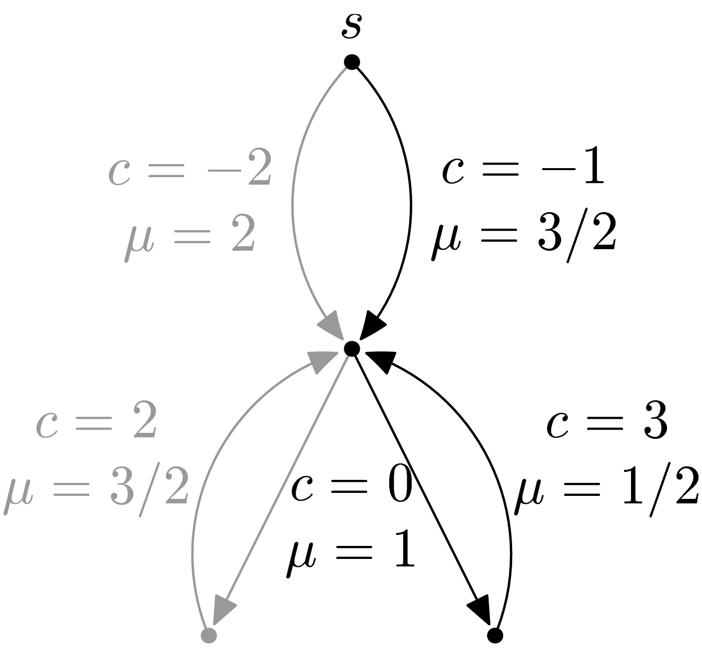
dual linear program (again):

max $\pi(s)$

s.t. $(\forall (v,w) \in A)\,(0 \leq c^{\pi}(v,w))$.

Guess and Verify





Algorithm: Guess and Verify

Optimal Solution

- consists of path + lossy cycle
- has only zero reduced cost arcs

Comparison Routine

Guess source's potential $\pi(s)$.
Relax all arcs using Bellman-Ford-Moore algorithm.
Verify arcs' reduced costs.
case negative cost arc**:** guess too large
case no zero reduced cost augmented path**:** guess too small
else: guess just right

Algorithm: binary search on source's potential $\pi(s)$

- not strongly polynomial: $n \log \bar{\mu}$ iterations

Megiddo's Parametric Search

- potentials as linear functions of parameter ϖ
  - ϖ abstracts source's potential $\pi(s)$
- left-distributive closed semiring for BFM algorithm

  path concatenation

  $$(c_1 + \mu_1\varpi) \odot (c_2 + \mu_2\varpi) = (c_1 + \mu_1 c_2) + \mu_1\mu_2\varpi$$

  path comparison

  $$(c_1+\mu_1\varpi)\oplus(c_2+\mu_2\varpi) = \text{ max } (\varpi-c_1)/\mu_1, (\varpi-c_2)/\mu_2$$

  - comparing lines can require one BFM computation
- Use parametric BFM with BFM comparison subroutine.
  - each iteration reduces range for ϖ
  - $O(mn^2 \log m)$ time

ϵ-Approximation Generalized Flow Algorithms

Use GSP to yield fully strongly polynomial-time approximation schemes:

- generalized maximum flow problem
- generalized minimum-cost flow problem
- generalized concurrent flow problem
  - strongly polynomial approximation
- generalized multicommodity maximum flow problem
  - strongly polynomial approximation
- generalized multicommodity minimum-cost flow problem

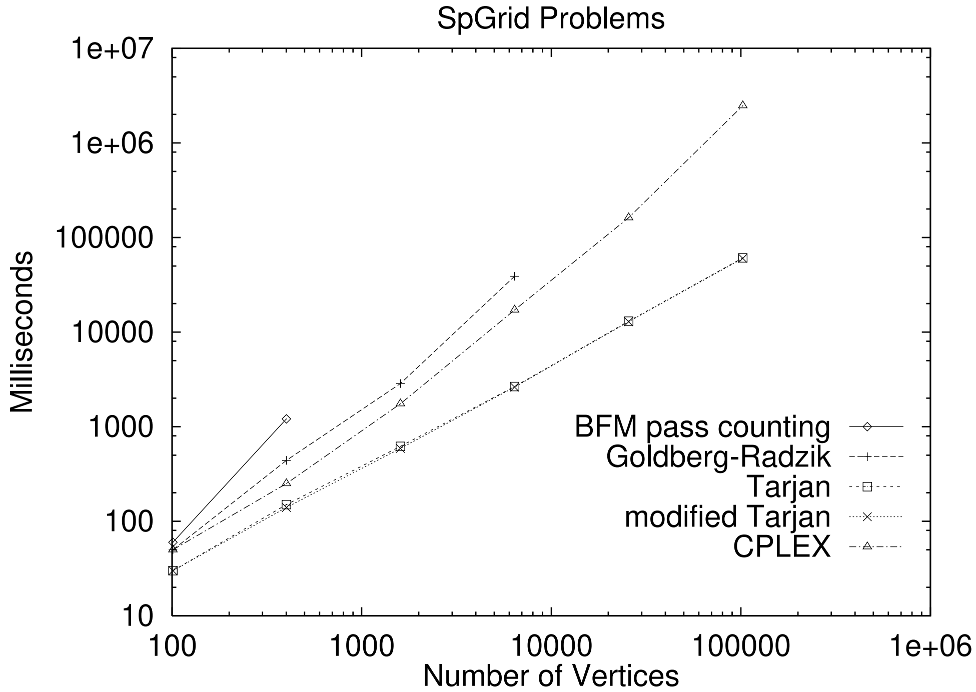
ϵ-Maximum Flow Algorithms

Use Garg-Könemann packing framework [1997].

- costs are exponential functions of flow's use of capacities
- until the flow's cost is large enough,
  1. solve generalized shortest paths
  2. scale flow to obey capacity constraints
  3. add flow to existing solution
  4. update exponential functions
- for multicommodity version,
  - solve one GSP per commodity
  - choose flow for cheapest commodity
  - $O(\epsilon^{-2} km \log m)$ iterations
  - $O(\epsilon^{-2} km^2 n^2 \log^2 m)$ time

Preliminary Implementation Results

Theory: $O(mn^2 \log n)$ time



Conclusions

- a Bellman-Ford-Moore comparison subroutine for generalized shortest paths
- weakly polynomial-time binary search algorithm
- strongly polynomial-time algorithm using Megiddo's parametric search
- combinatorial approximation algorithms for other generalized flow problems