

Polynomial Time Algorithms for Network Information Flow

Peter Sanders (MPII),
Sebastian Egner, and Ludo Tolhuizen (Philips Research)

[Bielefeld 2003]

Overview

- Multicasting with Coding
- How Coding Helps
- Related work
- Linear Coding
- A Simple Algorithm
- Fast Implementation
- Deterministic Implementation
- Generalizations
- Open Problems

Multicasting With Coding

Network: $G = (V, E)$ (acyclic, unit capacity for now)

Source: a node $s \in V$

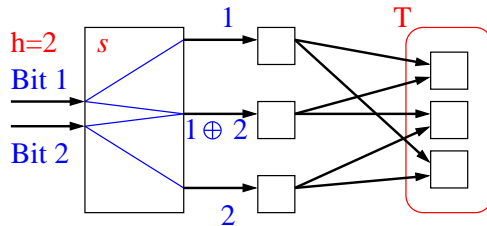
Sinks: $T \subseteq V$

Rate: $h = \min_{t \in T} \text{min cut separating } s \text{ from } t$
 $= \min_{t \in T} \text{max flow from } s \text{ to } t$.

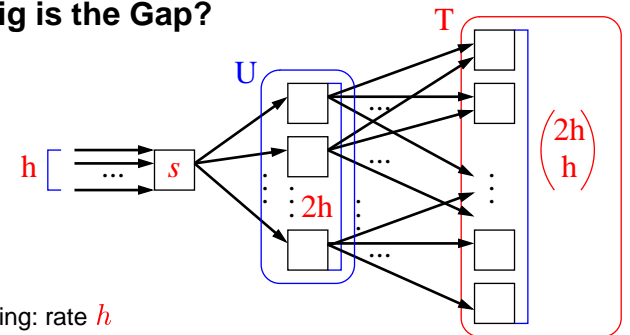
Just pairwise bandwidth?

[Ahlsvede-Cai-Li-Yeung 01]: No

Multicasting at rate h to
 all $t \in T$ simultaneously —
 if **coding** is allowed.



How Big is the Gap?



With Coding: rate h

\rightsquigarrow reconstruct h inputs out of any h -subset of U .

Without Coding: rate < 2

\rightsquigarrow a factor $\Omega(\log |V|)$ gap

Related Work Without Coding

[Carr-Vempala 00, Jain-Mahdian-Salavatipour 03]:

Fractional **Packings of Steiner Trees**.

Equivalent to min weight **Steiner Tree** for undirected graphs via duality.

→ $1 + \epsilon$ approximation is hard. 1.55 approximation.

With Coding

Ahlswede-Cai-Li-Yeung 00: **Random** codes, rate $(1 - \epsilon)h$

Cai-Li-Yeung 02: **Linear** codes, rate h

KoetterMedard02: Algebraic charact. of linear codes, $|\mathbb{F}| = \mathcal{O}(|T||E|)$

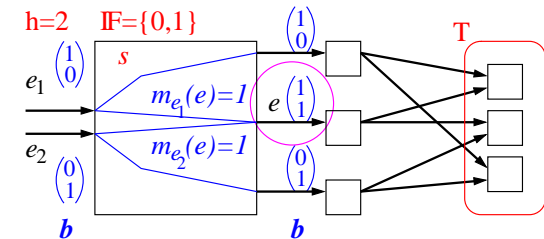
Here: **Polynomial** time construction, $|\mathbb{F}| = \mathcal{O}(|T|)$

independently: [Jaggi/Chou/Jain] a similar (somewhat slower) algorithm

Linear Coding [Cai-Li-Yeung 02]

Edge (v, w) carries symbol $y(v, w) = \sum_{(u,v) \in E} m_{(v,w)}(u, v) y(u, v) \in \mathbb{F}$

Global coding vector: $\mathbf{b}(v, w) = \sum_{(u,v) \in E} m_{(v,w)}(u, v) \mathbf{b}(u, v) \in \mathbb{F}^h$



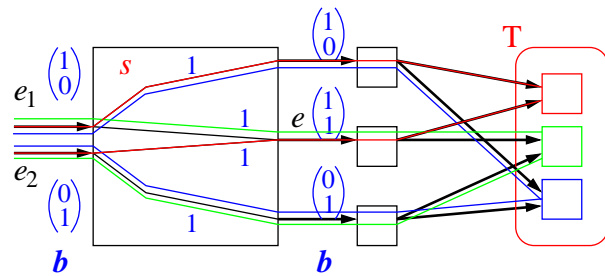
Key observation:

Input can be **reconstructed** from symbols carried by h edges C

⇔

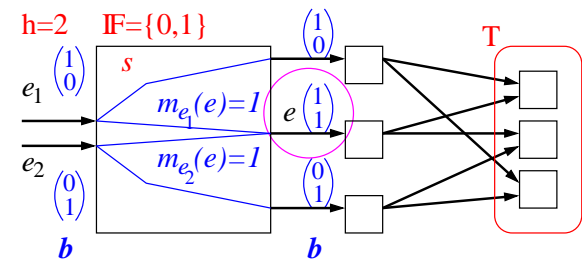
$\{\mathbf{b}(c) : c \in C\}$ is basis of \mathbb{F}^h

How Flows Help



- f^t | h edge disjoint s - t paths (decomposed flows)
- $f_{\leftarrow}^t(e)$ | predecessor of e in f^t
- C_t | one edge from each path in f^t

The LIF Algorithm



Find decomposed flows f_t

foreach vertex v in **topological order** and all $(v, w) \in E$ **do**

 (* **Invariant:** C_t contains the “most recent” edge from each path is f_t *)

 (* **Invariant:** $\forall t \in T : \{\mathbf{b}(c) : c \in C_t\}$ is **basis** of \mathbb{F}^h *)

 Find coefficients $m_{(v,w)}$ that maintain the invariant

return $(h, \{m_e : e \in E\}, \{(C_t, \{\mathbf{b}(c) : c \in C_t\}) : t \in T\})$

A Randomized Implementation

Lemma 1. For random m_e , $e' = f_{\leftarrow}^t(e)$

$$\Pr[\mathbf{b}(e) \text{ is linearly dependent of } \{b(e'') : e'' \in C_t \setminus \{e'\}\}] = \frac{1}{|\mathbb{F}|}$$

Proof: a simple counting argument

Total failure probability at most $\frac{|T|}{|\mathbb{F}|} \leq \frac{1}{2}$ for $|\mathbb{F}| \geq 2|T|$

\leadsto constant expected number of trials per edge

Proof of Lemma 1 (random m_e)

For any $t \in T$, $e \in E$, let $e' = f_{\leftarrow}^t(e)$

Fix $m_e(p)$ for $p \neq e'$. ($|\mathbb{F}|^{|P(e)|-1}$ choices)

Claim:

exactly one choice for $m_e(e')$ makes $\mathbf{b}(e)$ linearly dep. of $B_t \setminus \{\mathbf{b}(e')\}$

$$\mathbf{b}(e) = m_e(e')\mathbf{b}(e') + \sum_{p \in P(e) \setminus \{e'\}} m_e(p)\mathbf{b}(p) = (m_e(e') + \alpha)\mathbf{b}(e') + \mathbf{y}$$

for unique α , \mathbf{y} such that \mathbf{y} is linearly dep. of $B_t \setminus \{\mathbf{b}(e')\}$

$\mathbf{b}(e')$ is linearly indep. of $B_t \setminus \{\mathbf{b}(e')\}$

\leadsto only $m_e(e') = -\alpha$ makes $\mathbf{b}(e)$ linearly dep.

$$\Pr[\mathbf{b}(e) \text{ is linearly dependent of } B_t \setminus \{\mathbf{b}(e')\}] = \frac{|\mathbb{F}|^{|P(e)|-1}}{|\mathbb{F}|^{|P(e)|}} = \frac{1}{|\mathbb{F}|}$$

Fast Implementation I

Flows

Find augmenting paths round robin for each sink
thats good enough

$$\mathcal{O}(h \cdot |T| \cdot |E|)$$

Fast Implementation II

Constant Time Field Arithmetics using Zech Logarithms

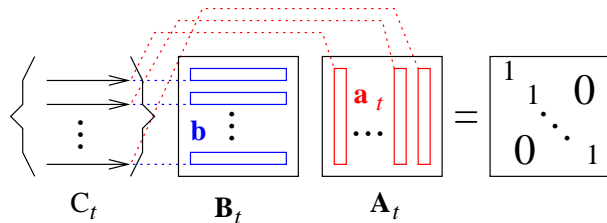
Choose α such that $\{\alpha^0, \alpha^1, \dots, \alpha^{|\mathbb{F}|-2}\} = \mathbb{F} \setminus \{0\}$

$$\alpha^a \alpha^b = \alpha^{a+b} \pmod{|\mathbb{F}|-1}$$

$$\alpha^a / \alpha^b = \alpha^{a-b} \pmod{|\mathbb{F}|-1}$$

$$\alpha^a + \alpha^b = \alpha^a (1 + \alpha^a / \alpha^b) \text{ use lookup table for successor}$$

Fast Implementation III



Write basis for sink t as matrix \mathbf{B}_t

Maintain $\mathbf{A}_t = \mathbf{B}_t^{-1}$ (update using Sherman Morrison formula)

$$e' \rightarrow e : \mathbf{A}_t \leftarrow \frac{\mathbf{a}_t(e')(\mathbf{b}(e) - \mathbf{b}(e'))}{1 + (\mathbf{b}(e) - \mathbf{b}(e')) \cdot \mathbf{a}_t(e')} \quad \mathcal{O}(h^2)$$

$$\mathbf{x} \text{ linearly dep. of } \mathbf{B}_t \setminus \mathbf{b}(e') \Leftrightarrow \mathbf{x} \cdot \mathbf{a}_t(e') = 0 \quad \mathcal{O}(h)$$

Decoding at t : Matrix-vector multiplication with $\mathbf{A}_t \quad \mathcal{O}(h^2)$

Total Construction Time: $\mathcal{O}(h^2 \cdot |T| \cdot |E|)$

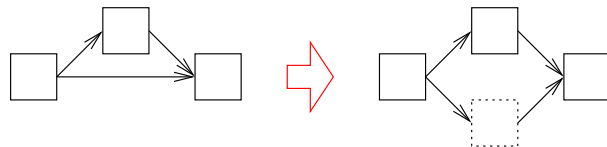
Pipelined Operation

So far: Communicate h symbols **once**

In practice: h symbols **each time step**

Problem: Naive pipelining **mixes** information from different time steps

Solution: Transform G into a **layered network**



Deterministic Implementation

Greedily fix one coefficient $m_e(p)$ after the other

For each sink there is at most one **bad coefficient**

Total time:

for each **edge** $|E| \times$

for each **coefficient** $|P(e)| \leq |T| \times$

for each **sink** $|T| \times$

find one **bad coefficient** $h +$

for **maintaining** $\mathbf{A}_t \quad \mathcal{O}(|E| \cdot |T| \cdot h^2)$

Total time

$$\mathcal{O}(|E| \cdot |T| \cdot h \cdot (|T| + h))$$

Capacitated Edges



Yields **pseudopolynomial** time algorithm

Fully Polynomial Time Approximation

Round capacities appropriately

Cycles [Ahlswede-Cai-Li-Yeung 00]

Make $n = |V|/\epsilon + 1$ copies G_0, \dots, G_n of G

edge $(u, v) \rightsquigarrow (u_i, v_{i+1})$

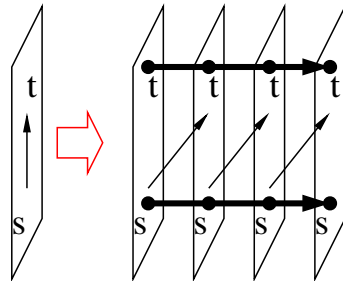
Capacity h connection between s_0 and s_i

Capacity h connection between t_i and t_n

\rightsquigarrow acyclic graph G' with rate $(1 - \epsilon)hn$

Apply LIF to G'

Emulate G' on G .



Open Problems

- Real world applications, e.g. cost efficient multicasting (Akamai?)
- Non-coding: Tighten the gap. Performance guarantees in polynomial time.
- Fully polynomial (scaling?) algorithm for capacitated edges
- Faster/exact handling of cycles
- Multiple sources. How much better? Still simpler?

Notation

V	nodes
E	directed, unit capacity edges (possibly parallel)
s	source node
T	set of sink nodes
h	maximal rate = $\min_{t \in T} s-t$ cut = $\min_{t \in T} \max s-t$ flow
f^t	h edge disjoint $s-t$ paths (decomposed flows)
$f_{\leftarrow}^t(e)$	predecessor of e in f^t
\mathbb{F}	finite field, symbols carried by edges
$m_e : P(e) \rightarrow \mathbb{F}$	local coding coefficients
$\mathbf{b}(e) \in \mathbb{F}^h$	global coding vector
C_t	one edge from each path in f^t