Optimization of a Distributed Storage Architecture under Uncertain Demand

A Study in Fast Flow Algorithms

Paul Beaujean[*†] Éric Gourdin[†]

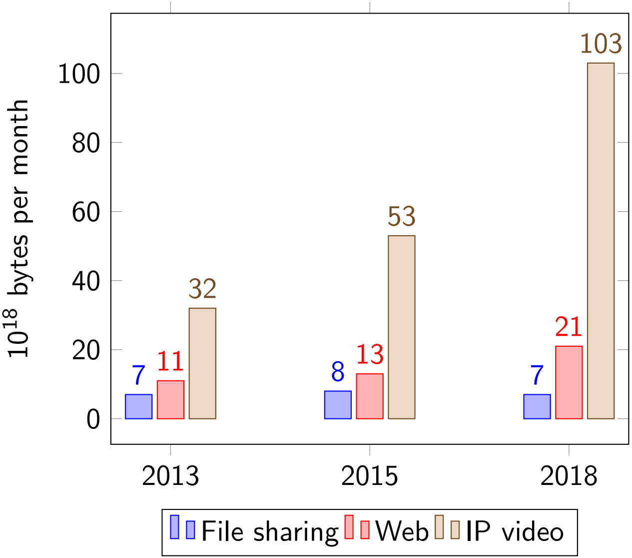
[*]ENSIIE-MPRO

[†]Orange Labs

March 20, 2015

Table of Contents

Context: The rise of IP video



- IP video: 80% of all IP traffic by 2020.

Source: CISCO Visual Networking Index 2014

Content Delivery Network

- distributed storage architecture
- reduced upstream bandwidth consumption
- increased quality of service

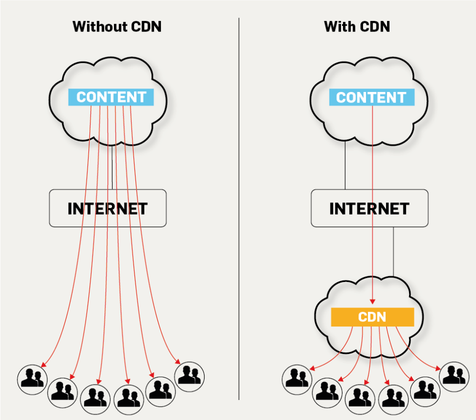
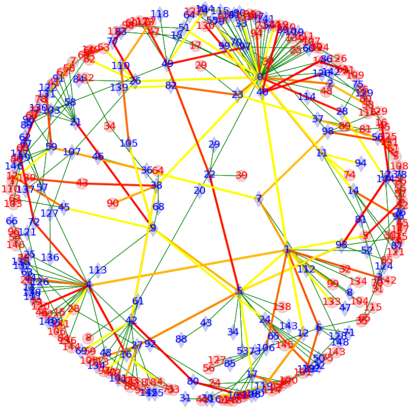


Figure: Caching content closer to demand

Multicommodity Flow = Content Delivery



source	cache server
target	client
capacity	bandwidth
commodity	content

Fractional Multiflow is in P

No combinatorial algorithm known to solve multiflow.

Polynomial-size LP with edge formulation

$$
\begin{aligned}
\max \sum_{s,t \in K} \sum_{a \in \delta^+(s)} & f_a^{s,t} \\
\text{s.t.} \sum_{s,t \in K} & f_a^{s,t} \leq c_a, \ \forall a \in A \\
& \sum_{s,t \in K} \sum_{a \in \delta^-(v)} f_a^{s,t} = \sum_{s,t \in K} \sum_{a \in \delta^+(v)} f_a^{s,t}, \ \forall v \in V \setminus \{s,t\} \\
& \sum_{a \in \delta^-(s)} f_a^{s,t} = 0, \ \forall s \in S \\
& f \in \mathbb{R}_+^{|A \times K|}
\end{aligned}
$$

can be solved in polynomial time with an IPM

$\implies$ $\textsc{Multiflow} \in \mathrm{P}$

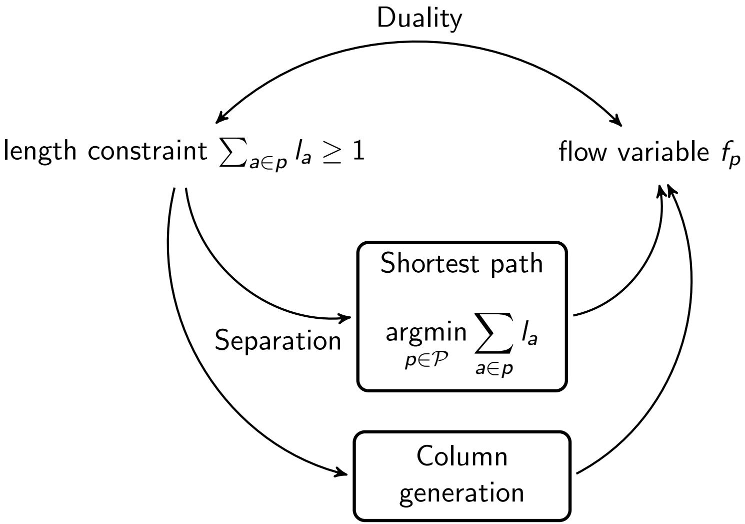
Can it be solved on large instances?

Implicit exponential-size path formulation

$$
(\Pi)\begin{cases}
\max & \displaystyle\sum_{p\in\mathcal{P}} f_p \\
\text{s.t.} & \displaystyle\sum_{p\ni a} f_p \leq c_a,\ \forall a \in A\ (l_a) \\
& f \in \mathbb{R}_+^{|\mathcal{P}|}
\end{cases}
$$

$$
(\Delta)\begin{cases}
\min & \displaystyle\sum_{a\in A} c_a l_a \\
\text{s.t.} & \displaystyle\sum_{a\in p} l_a \geq 1,\ \forall p \in \mathcal{P}\ (f_p) \\
& l \in \mathbb{R}_+^{|A|}
\end{cases}
$$

Column generation



Primal-dual approximation algorithm

Fully Polynomial Time Approximation Scheme

An algorithm that returns a solution of value within ϵ of the optimal value:

- in time bounded by a polynomial in the instance size,
- in time bounded by a polynomial in ϵ^{-1}.

There are combinatorial algorithms that achieve such guarantees.

How to design one?

- leverage the multiplicative weights algorithm
- reuse ideas from column generation

Multiplicative weights

Background

Discovered several times

- 1950's: game theory
- 1970's: machine learning
- 2000's: design of competitive online algorithms
- 2010's: approximation algorithms for SDP

Definition

Given a finite set of actions with bounded gain

$$\mathcal{A} = \{a \mid g_a^t \in [0,1]\}$$

Associate weights

$$\forall a \in \mathcal{A} \begin{cases} \lambda_a^1 & = 1 \\ \lambda_a^{t+1} & = \lambda_a^t (1 + \epsilon\, g_a^t) \end{cases}$$

A surprising result

The multiplicative weights algorithm

- At round t, choose action a with probability $\frac{\lambda_a^t}{\sum_a \lambda_a^t}$

has a guaranteed gain of

$$G \geq (1-\epsilon)G_a - \frac{\ln|\mathcal{A}|}{\epsilon}, \ \forall a \in \mathcal{A}$$

$G = \sum_t \mathbb{E}(g^t)$ expected gain of the algorithm

$G_a = \sum_t g_a^t$ gain of action a

Mapping multiplicative weights concepts to multiflow

Mapping concepts

actions	$\mathcal{A}$	$\rightarrow$	A	arcs
gain	g_a	$\rightarrow$	f_a/c_a	arc congestion
compound gain	λ_a	$\rightarrow$	l_a	length (compound congestion)

Iterative primal-dual algorithm

Successively adds flow to the graph

Choosing arcs by constraint separation

Instead of choosing arcs randomly:

- compute shortest path w.r.t. the length l_a^t
- all arcs on that path receive maximum feasible flow
- probabilities are set a posteriori

Combining gain guarantee and properties of flows

Using the gain guarantee

$$G \geq (1-\epsilon)G_* - \frac{\log m}{\epsilon}$$

and simple flow considerations

- $G_* \begin{cases} \text{gain of the best action in hindsight} \\ \text{congestion rate of the most congested arc} \end{cases}$
- $F/F^* \geq G$: approximation ratio bounds expected gain

results in

- If $G_* = \log m \; \epsilon^{-2}$ then $F/G_* \geq (1-2\epsilon)F^*$ $\leftarrow$ AS
- Runtime: $O(km^2 \log m \; \epsilon^{-2})$ $\leftarrow$ FPT

Numerical experiments

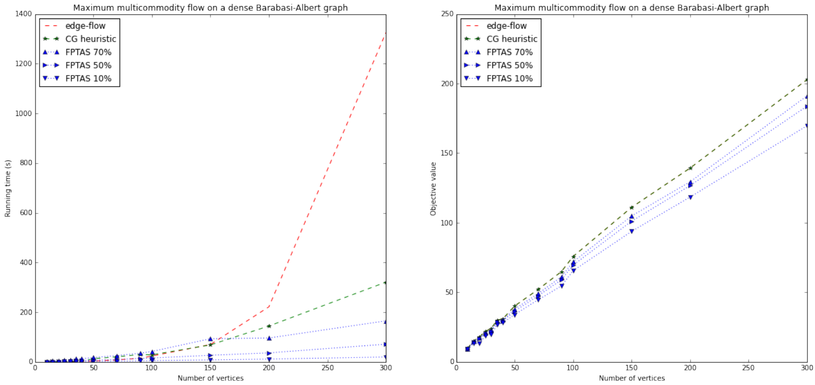


Figure: Performance of LP, column generation, and FPTAS

Conclusion

Highlights

- CDN design naturally involves multiflows
- Approximation: an interesting alternative to exact algorithms

Achievements

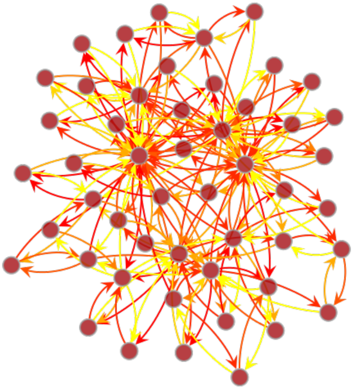
- Studied a theoretical problem in-depth
- Implemented a state-of-the-art algorithm
- Integrated it in an Operations Research library

Perspectives

- Studying the matrix multiplicative weights algorithm
- Applying it to solve SDP approximately

during a PhD at Orange Labs and Université Paris-Dauphine

Thank you!



Any questions?