Filtering Undesirable Flows in Networks

Gleb Polevoy Stojan Trajanovski Paola Grosso Cees de Laat

SNE, The University of Amsterdam, The Netherlands

Problems

Consider problems like

- DDoS
- Unimportant flows

Any problem of filtering some "bad" flows to increase the "good" ones.

Needs

While filtering, we need to

- Minimize the effort
- Reasonable time

How?

No theoretical approximations of such filtering.

We

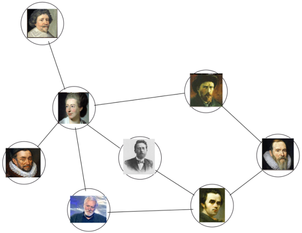
1. formally model
2. prove hardness
3. give a solution

Model

1. The network is a directed capacitated graph $G = (N, E), c \colon E \to \mathbb{R}_+$.
2. A flow f from node o to d along a path, $f = (\underbrace{v(f)}_{\text{value}}, \underbrace{P(f)}_{\text{path}})$, such that

   for every edge e:

$$\sum_{f: e \in P(f)} v(f) \leq c(e).$$



Model – BFF

Definition (Bad Flow Filtering (BFF))

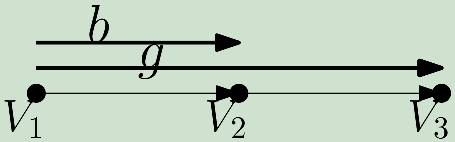
1. *Input:* $(G = (N, E), c \colon E \to \mathbb{R}_+, F, GF, BF, w \colon BF \to \mathbb{R}_+)$.
2. *A solution S is a subset of bad flows to filter.*
3. *A feasible solution is a solution such that the good flows can be allocated values such that the total value of the good flows is the maximum possible.*
4. *Find a feasible solution with the minimum total weight* $w(S) \triangleq \sum_{b \in S} w(b)$.

Model – BFF – Example

The trivial feasible solution *BF* can be very far from the optimum.

Example

- Edge (V_1, V_2) has capacity 2 and (V_2, V_3) has capacity 1.
- $v(b) = v(g) = 1$.
- The optimal solution is $\emptyset$, ∞ times better than everything.



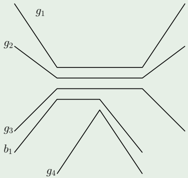
Model – BFF and UIBFF

Definition (Bad Flow Filtering (BFF))

Given $(G = (N, E), c\colon E \to \mathbb{R}_+, F, GF, BF, w\colon BF \to \mathbb{R}_+)$, *minimize* $w(S)$ *such that the total good flow is maximum.*

Definition (Uniform Intersection Bad Flow Filtering (UIBFF))

BFF where every $g \in GF$ *has a set of edges on its path,* $E(g) \subseteq P(g)$, *such that every other good flow* g' *that intersects* g *fulfills:*
i.e. $P(g) \cap P(g') = E(g)$.



UIBFF is Hard

Hardness of approximation

If $\mathrm{P} \neq \mathrm{NP}$, then UIBFF is not approximable within $2^{\log^{1-1/\log\log^c(n)}(n)}$, for $n = |E| + |GF|$ and any $c < 0.5$. Even if no bad edges intersect one another.

General Approximation Technique: Local Ratio

Finding a feasible set of elements S s.t. $w(S) \triangleq \sum_{x \in S} w(x)$ is minimized by manipulating the weights.

1. **If** $\emptyset$ is feasible, **return** $\emptyset$.
2. **Otherwise,** remove the zero-weight elements, solve recursively, and add them afterwards.
3. **Otherwise,** devise an r-effective w_1 and solve recursively w.r.t. $w_2 \triangleq w - w_1$.

Definition (r-effective w_1)

Every feasible solution is an r-approximation w.r.t. w_1.

Theorem (LR theorem)

If a feasible solution is an r-approximation w.r.t. w_1 and w_2, then it is also an r-approximation w.r.t. $w_1 + w_2$.

Reminder of Our Problem

Given $(G = (N, E), c\colon E \to \mathbb{R}_+, F, GF, BF, w\colon BF \to \mathbb{R})$, minimize $w(S)$ such that the total good flow is maximum.

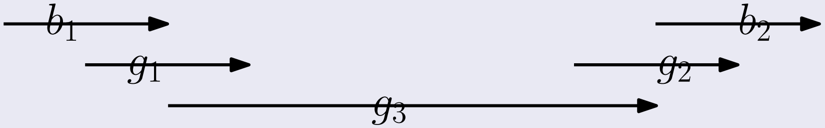
Our Algorithm (Simplified)

1. If filtering cannot increase any good flow, **return** $\emptyset$.
2. **Else, if** there exist bad flows with zero weight, then
   1. remove them,
   2. solve recursively,
   3. add them back.
3. **Else,**
   1. Pick any good flow g that can be increased.
   2. Let all the intersecting good flows that can increase be $g_1, \dots, g_p$. Let $G \triangleq \{g, g_1, \dots, g_p\}$. Let their saturated edges, one from a flow, be $F(G)$, and all the bad flows that contain edges from $F(G)$ be $B(F(G))$.
   3. Let $\delta > 0$ be the minimum weight in $B(F(G))$. Define $w_1 \colon BF \to \mathbb{R}_+$:

   $$w_1 \triangleq \begin{cases} \delta & \textbf{if } b \in B(F(G)), \\ 0 & \textbf{otherwise.} \end{cases}$$

   4. Solve recursively w.r.t. $w - w_1$.

This would be a problem:



However, in UIBFF:

Observation

We can increase the total good flow $\iff$ *we can always increase a good flow by filtering bad ones that intersect it.*

Proof.

UIBFF assumes that all the good flows intersect a given good flow at the same edges. □

Our Algorithm – Analysis – Measures

Definition

Given a BFF, let k be the largest possible number of good flows that a given good flow intersects. Formally,

$$k \triangleq \max\left\{\left|\left\{g' \in GF \setminus \{g\} : P(g') \cap P(g) \neq \emptyset\right\}\right| : g \in G\right\}.$$

Definition

For a BFF, let q be the largest number of bad flows that intersect a good flow at any given edge. Formally,

$$q \triangleq \max\{|\{b \in BF : e \in P(b)\}| : g \in G, e \in P(g)\}.$$

Our Algorithm – Analysis – Approximation Ratio

Reminders

$$k \triangleq \max\left\{\left|\{g' \in GF \setminus \{g\} : P(g') \cap P(g) \neq \emptyset\}\right| : g \in G\right\}.$$
$$q \triangleq \max\left\{|\{b \in BF : e \in P(b)\}| : g \in G, e \in P(g)\right\}.$$

$$w_1 \triangleq \begin{cases} \delta & \textbf{if } b \in B(F(G)), \\ 0 & \textbf{otherwise.} \end{cases}$$

w_1 is $q(k+1)$-effective.

Lemma

Any feasible solution S and optimal S^* fulfill: $w_1(S) \leq q(k+1) \cdot w_1(S^*)$.

Proof.

Any feasible solution allows g or at least one of $g_1, \ldots, g_p$ grow, by filtering at least one of the intersecting bad flows. $\Rightarrow w_1(S) \geq \delta$.
Always, $w_1(S) \leq q(k+1)\delta$. □

The correctness and $q(k+1)$-approximation follows by induction.

1. **If** filtering cannot increase any good flow, **return** $\emptyset$.
2. **Else, if** there exist bad flows with zero weight, then
   1. remove them,
   2. solve recursively,
   3. add them back.
3. **Else,**
   1. Pick any good flow g that can be increased.
   2. Let all the intersecting good flows that can increase be $g_1, \ldots, g_p$. Let $G \triangleq \{g, g_1, \ldots, g_p\}$. Let their saturated edges, one from a flow, be $F(G)$, and all the bad flows that contain edges from $F(G)$ be $B(F(G))$.
   3. Let $\delta > 0$ be the minimum weight in $B(F(G))$. Define $w_1 \colon BF \to \mathbb{R}_+$:

$$w_1 \triangleq \begin{cases} \delta & \textbf{if } b \in B(F(G)), \\ 0 & \textbf{otherwise.} \end{cases}$$

   4. Solve recursively w.r.t. $w - w_1$.

contact: G.Polevoy@uva.nl

Conclusions

1. Modeling filtering problems (e.g., DDoS, dispensable flows)
2. Important, but extremely hard to approximate
3. Local Ratio $q(k+1)$ approximation
4. The approximation is tight

Future Work

- Arbitrary intersections (BFF)
- A given allocation algorithm, like max-min fairness

Thank You!

Hardness reduction UIBFF

MMSA_3 to UIBFF

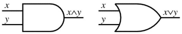
Proof.

Reduction from Minimum-Monotone-Satisfying-Assignment of depth 3 (MMSA_3). An MMSA_3 instance

Input: a monotone (with no negative literals) Boolean formula, which is a conjunction (AND) of disjunctions (OR) of conjunctions, such as $((x_1$ AND $x_3)$ OR $(x_2$ AND $x_3))$ AND $((x_2$ AND x_4 AND $x_5)$ OR $(x_1))$.

The goal: a satisfying assignment that minimizes the number of variables that are assigned 1.

□



Hardness reduction UIBFF - Cont.

MMSA_3 to UIBFF

Proof - Cont.

Satisfying all the disjunctions of the conjunctions is expressed as unblocking all the edges of at least one good flow from all the sets of intersecting good flows. □

MMSA_3		UIBFF
conjunction (AND)	$\longrightarrow$	All the sets of intersecting good flows
disjunction (OR)	$\longrightarrow$	A set of good flows intersecting at an edge
conjunction (AND)	$\longrightarrow$	Edges of a good flow
variables x	$\longrightarrow$	bad flows b_x

Algorithm - The Zero Weight Elements

We remove the zero-weight elements, solve recursively, and add them afterwards.

1. This leaves the solution feasible, since the add the removed afterwards.
2. The recursive invocation returns a $q(k+1)$-approximation w.r.t. the pruned instance. $\Rightarrow$ It is also a $q(k+1)$-approximation w.r.t. the original instance, because we
   1. have the same optimum cost
   2. have the same solution cost

Algorithm - Tightness

example

1. Good flows $g_1, \ldots, g_{n+1}$ with $c(e_i^{(2)}) = 1$.
2. Bad flows $b_{\{1,n\}}, b_{\{2,n\}}, \ldots, b_{\{n-1,n\}}, b_{\{1,2,\ldots,n+1\}}$ with weight 1 each.
3. $m + 1$ copies of the constructed problem instance. The distinct copies intersect only at the edges $e_i^{(2)}$.

Assume the algorithm picks g_n of one of the copies. The next invocation removes all the bad flows from all the copies. This returns the solution BF, while the optimum is $\left\{ b_{\{1,2,\ldots,n+1\}} \right\}$.