

Scalable, Generic, and Adaptive Systems for Focused Crawling

Georges Gouriten* - georges@netiru.fr

Silviu Maniu[°]

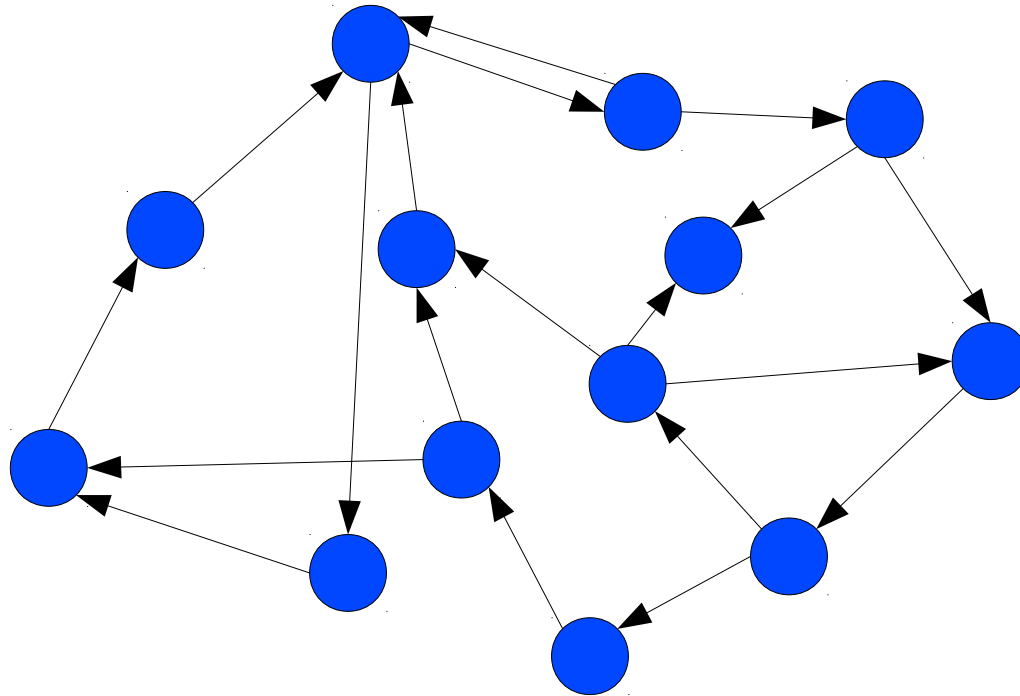
Pierre Senellart*[°]

* Télécom Paristech – Institut Mines-Télécom – LTCI CNRS

[°] Hong Kong University

What is focused crawling?

A directed graph



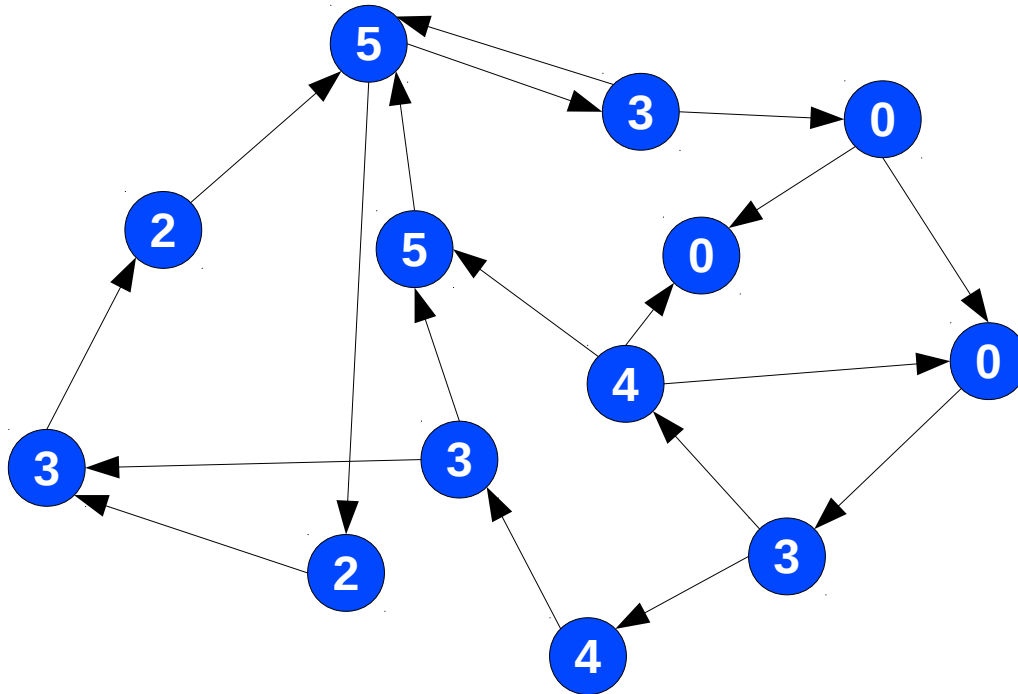
Web

Social network

P2P

etc.

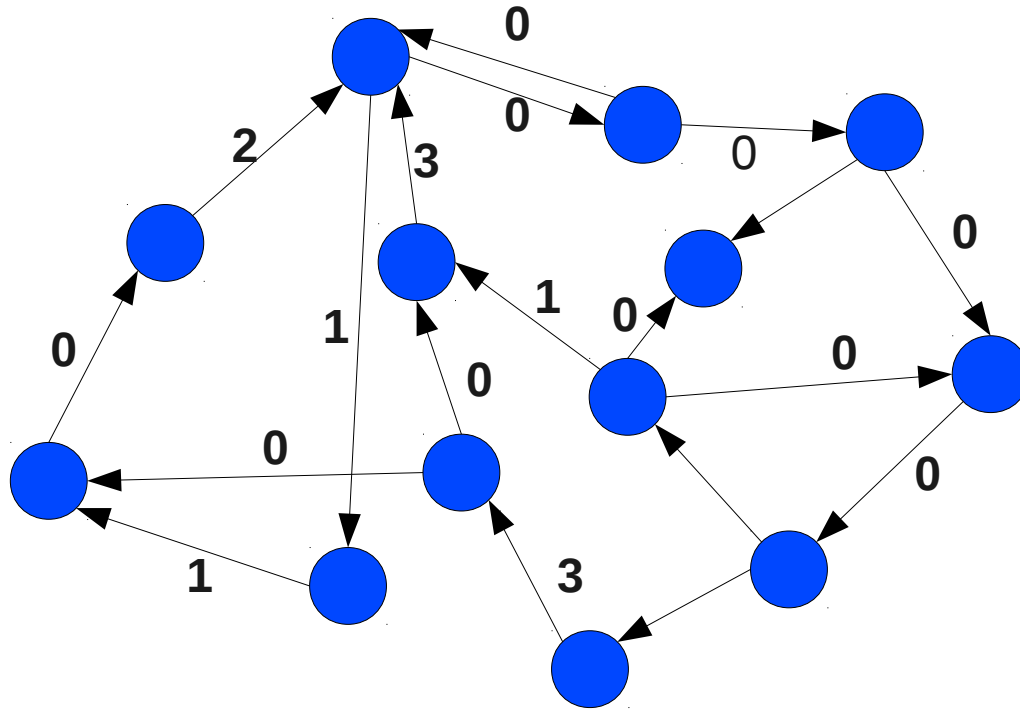
Weighted



Let u be a node,

$\beta(u)$ = count of the word *Bhutan* in
all the tweets of u

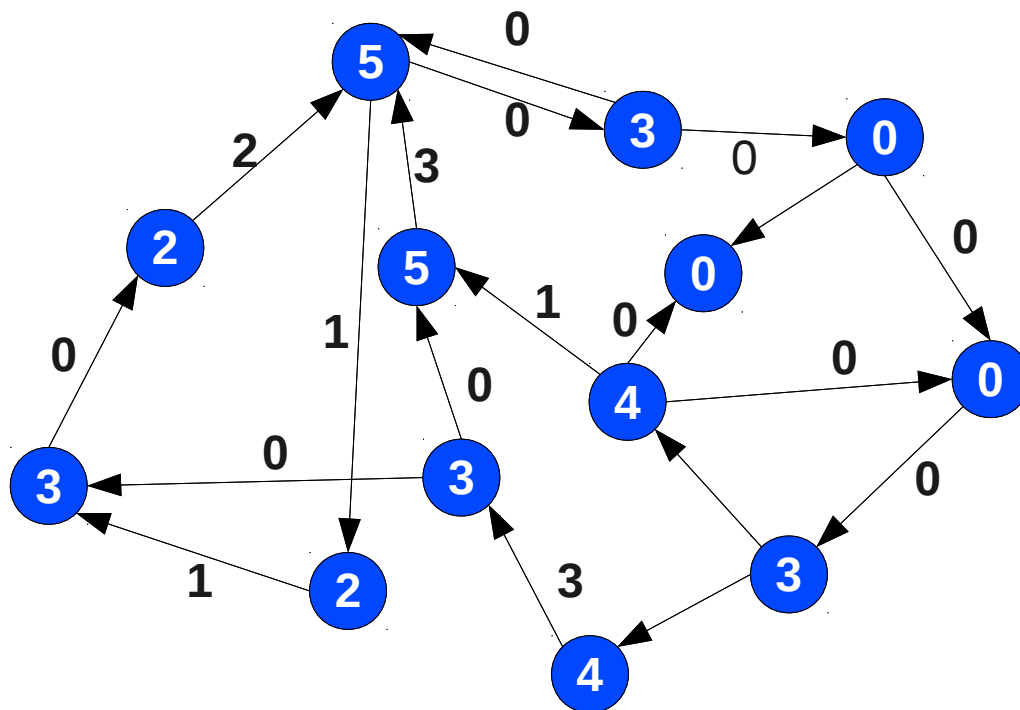
Even more weighted



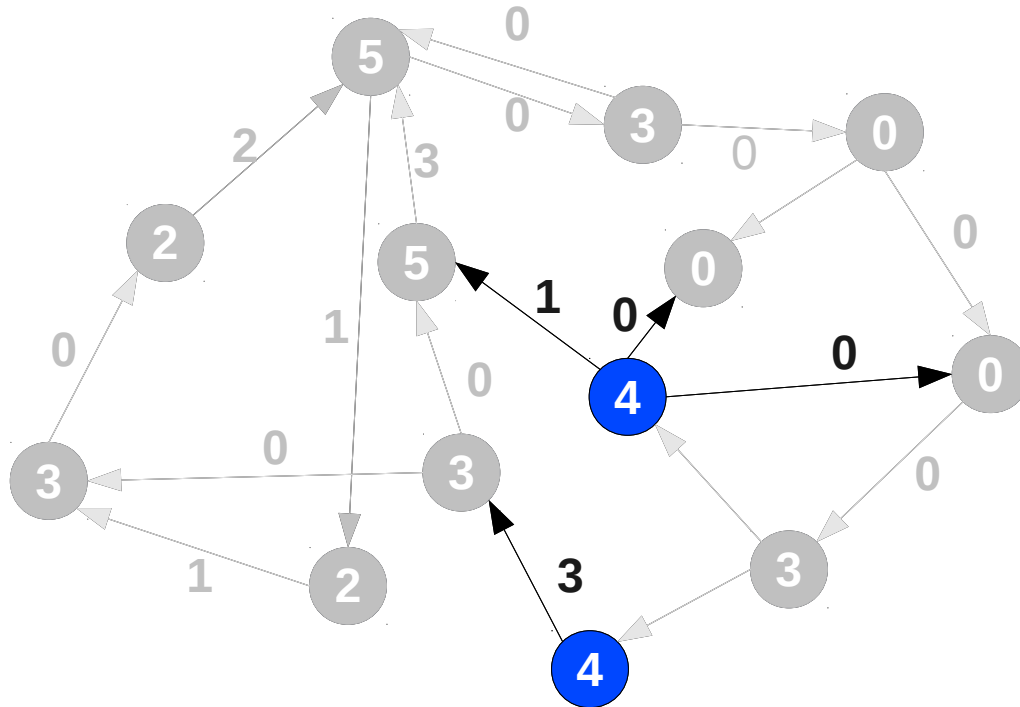
Let (u, v) be an edge,

$\alpha(u)$ = count of the word *Bhutan* in
all the tweets of u mentioning v

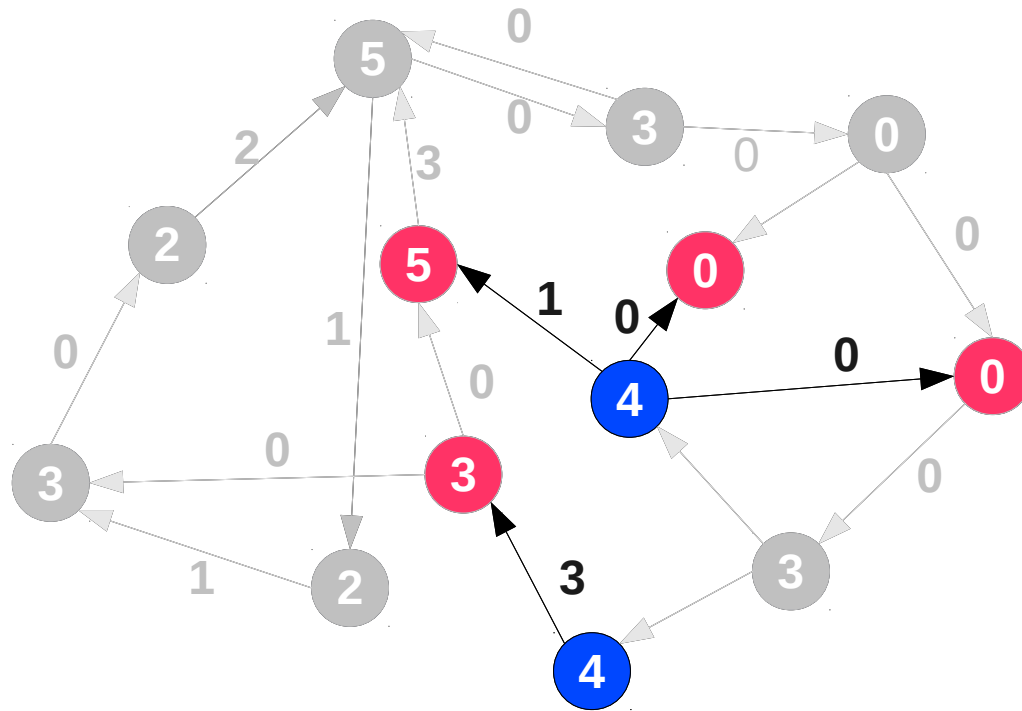
The total graph



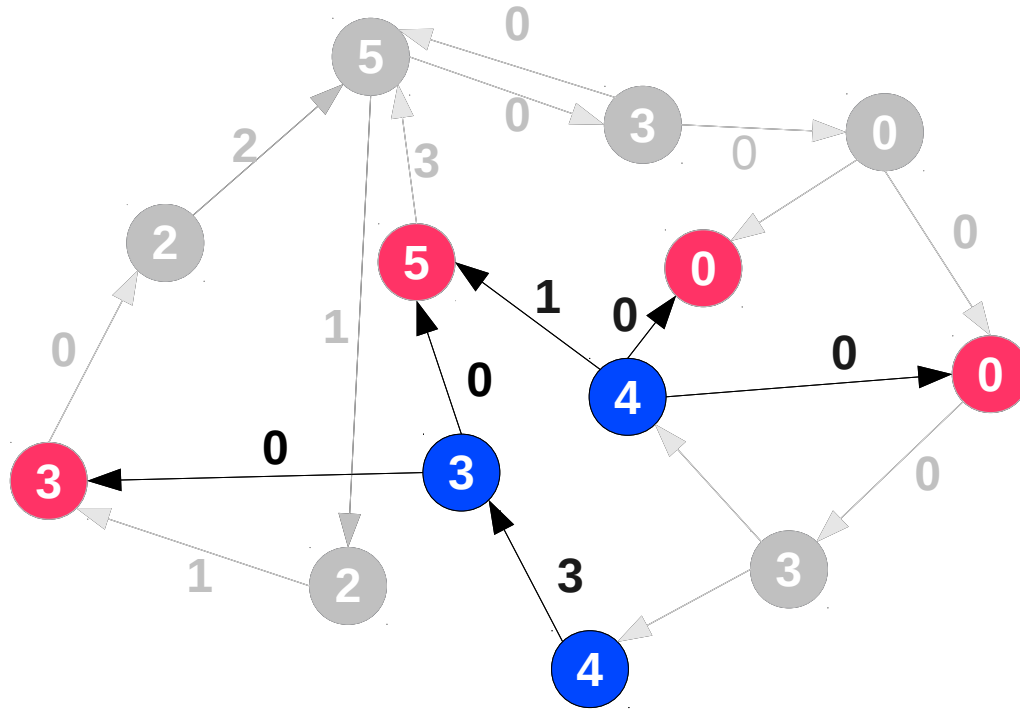
A seed list



The frontier



Crawling one node



A crawl sequence

Let V_0 be the seed list, a set of nodes,
a *crawl sequence*, starting from V_0 , is

$$\{ v_i, v_i \text{ in } \text{frontier}(V_0 \cup \{v_0, v_1, \dots, v_{i-1}\}) \}$$

Goal of a focused crawler

Produce crawl sequences with
global scores (sum) as high as possible

A high-level algorithm

Estimate scores at the frontier

Pick a node from the frontier

Crawl the node

Supposing a perfect estimator

Finding an optimal crawl sequence offline:
NP-hard

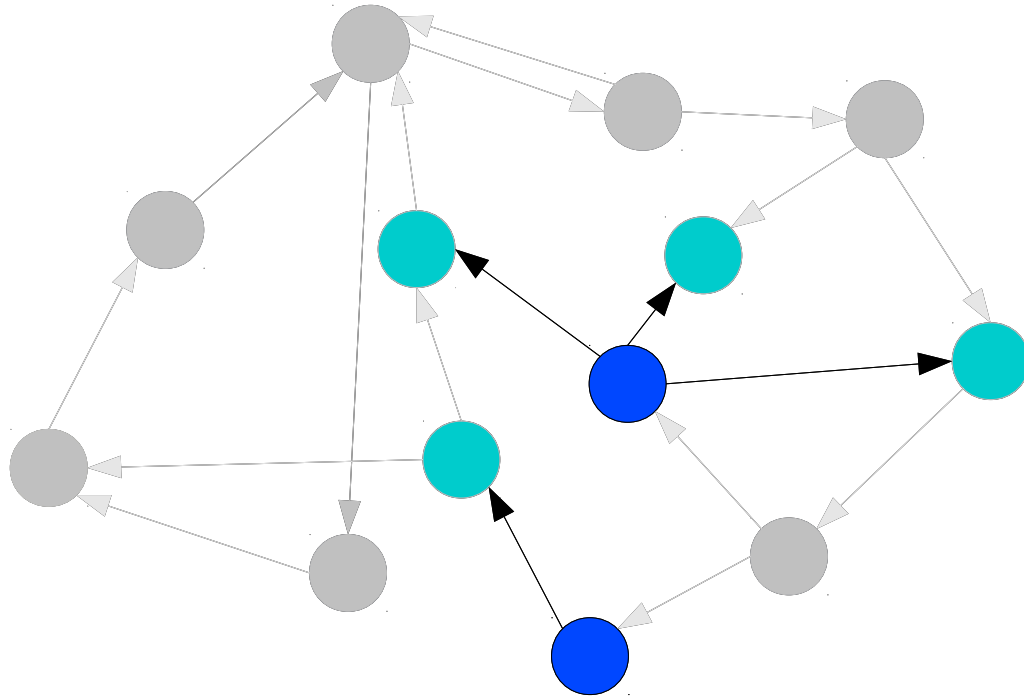
Greedy wins for a crawled graph > 1000 nodes

Refresh rate of 1 is better

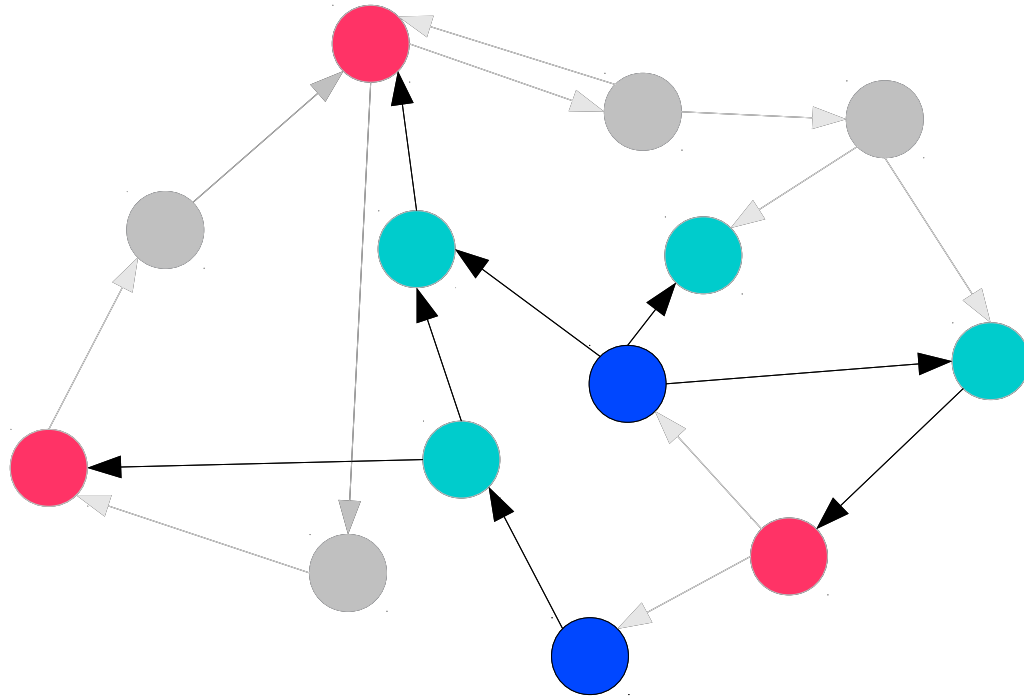
Estimation in practice

Different kinds of estimators

bfs



bfs



bfs

ESTIMATOR 1 (bfs). $\tilde{\beta}(v) = \frac{1}{l(v)+1}$, where $l(v)$ is the distance of v to V_0 .

nr

navigational rank

score propagation from the ancestors of a node

then to the children of a node

nr

$$NR_1(v)^{t+1} = d \times w(v) + (1 - d) \times \text{avg}_{(v,u) \in E'} \frac{NR_1(u)^t}{d_i(u)}$$

$$NR_2(v)^{t+1} = d \times NR_1(v) + (1 - d) \times \text{avg}_{(u,v) \in E'} \frac{NR_2(u)^t}{d_o(u)}.$$

ESTIMATOR 2 (nr). $\tilde{\beta}(v) = NR_2(v)$.

opic

online page importance computation

~ online pageRank computation

opic

1. the node v with the highest cash is selected, and its history is updated with the current cash value $H(v) = H(v) + C(v)$,
2. for each outgoing node u of v , the cash value is updated $C(u) = C(u) + \frac{C(v)}{d_{o(v)}}$,
3. the cash value of v is reset and the global counter incremented, by $G = G + C(v)$ and $C(v) = 0$.

$$2. \rightarrow C(u) = C(u) + \frac{C(v)}{\sum_{(v,w) \in E'} \alpha(v,w) \times C(w)} \times \alpha(v,u) \times C(u)$$

$$\text{ESTIMATOR 3 (opic). } \tilde{\beta}(v) = \frac{H(v)+C(v)}{G+1}.$$

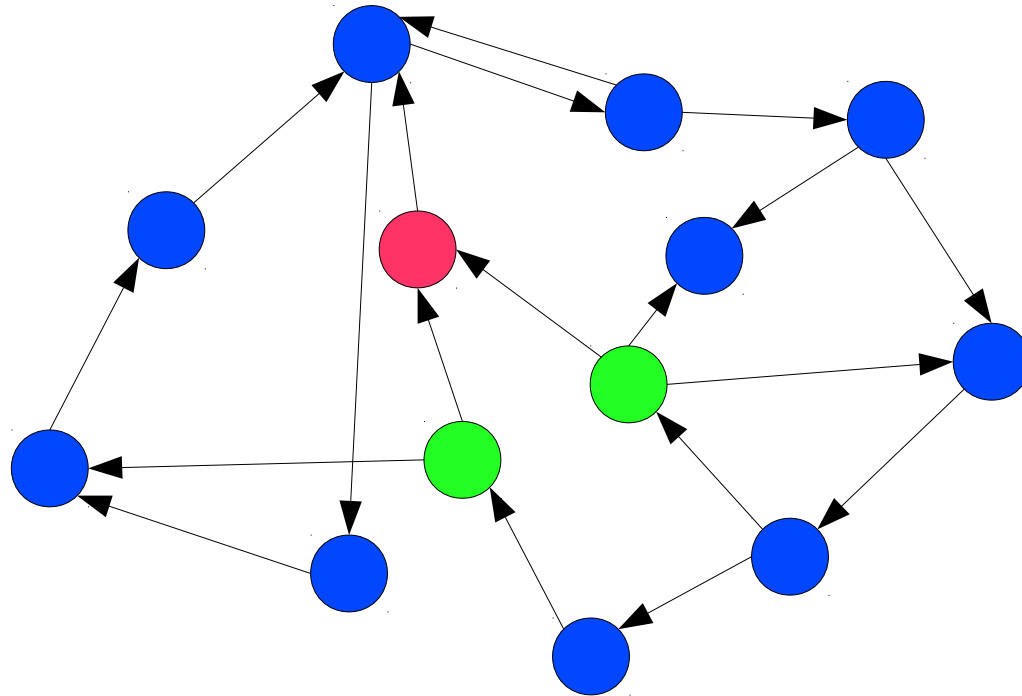
Open spaces in the state-of-the-art

nr has a quadratic complexity

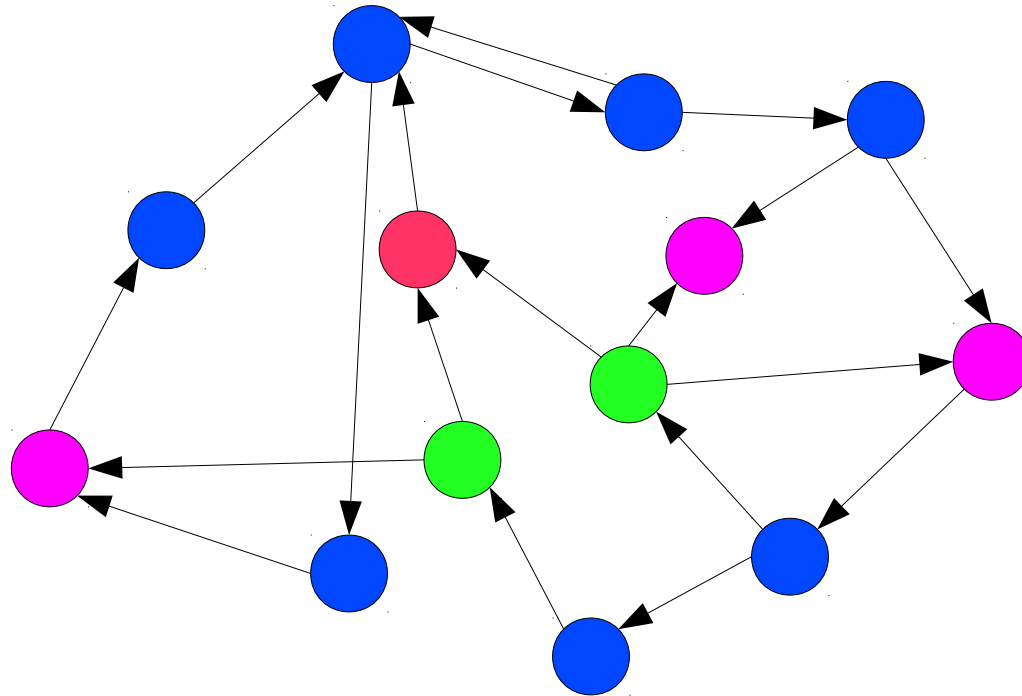
opic focus on popularity

the rest is about how to score

First-level neighborhood



Second-level neighborhood ●



Neighborhood-based estimators

ESTIMATOR 4 (fl_n fl_e fl_ne sl_n sl_e sl_ne).

$$\text{fl_deg} : \tilde{\beta}(v) = d_i(v) = |P(v)|$$

$$\text{fl_n} : \tilde{\beta}(v) = \sum_{u \in P(v)} \beta(u)$$

$$\text{fl_e} : \tilde{\beta}(v) = \sum_{u \in P(v)} \alpha(u, v)$$

$$\text{fl_ne} : \tilde{\beta}(v) = \sum_{u \in P(v)} \beta(u) \alpha(u, v)$$

$$\text{sl_n} : \tilde{\beta}(v) = \sum_{u \in P(v)} \sum_{\substack{w \in V' \\ u \in P(w)}} \beta(w)$$

$$\text{sl_e} : \tilde{\beta}(v) = \sum_{u \in P(v)} \sum_{\substack{w \in V' \\ u \in P(w)}} \alpha(u, w)$$

$$\text{sl_ne} : \tilde{\beta}(v) = \sum_{u \in P(v)} \sum_{\substack{w \in V' \\ u \in P(w)}} \beta(w) \alpha(u, w)$$

deg, e, n, ne

deg: number of neighbors

e: sum of incoming **edges**

n: sum of incoming **nodes**

ne: sum of incoming (**node*edge**)s

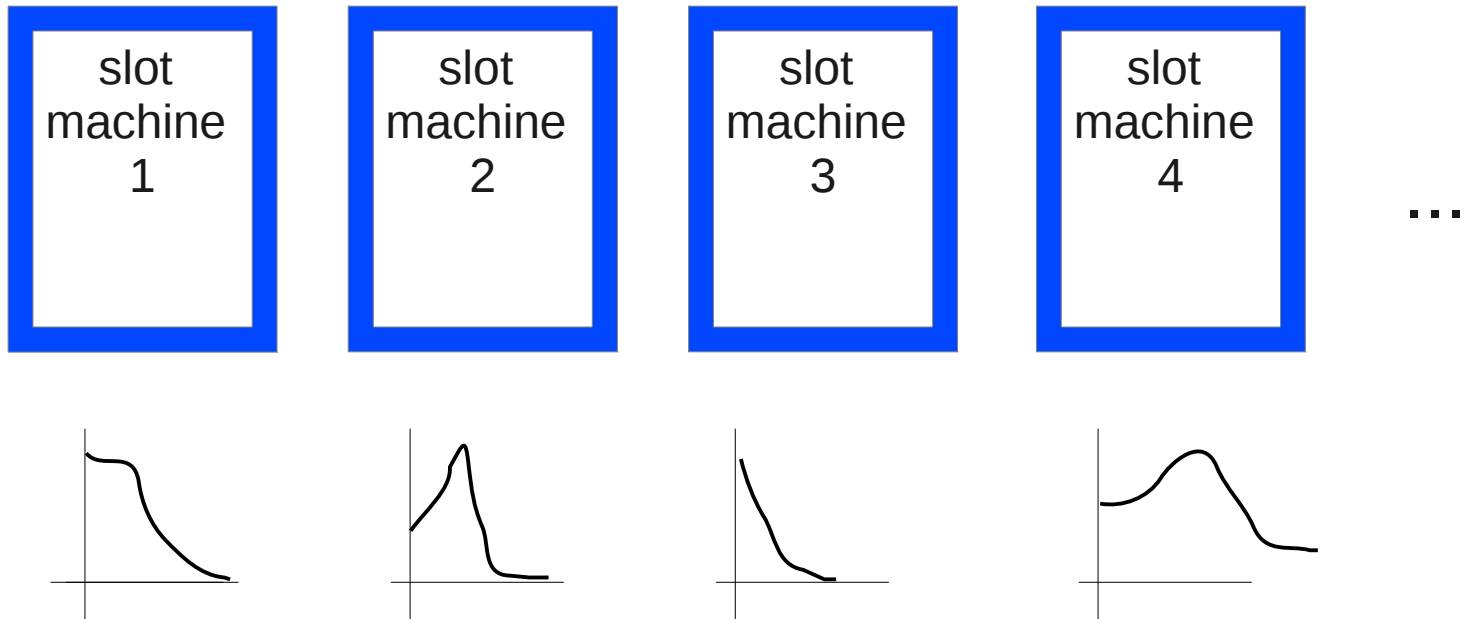
Linear regressions

ESTIMATOR 5 (lr_fl lr_sl).

lr_fl : $\tilde{\beta}(v) =$ trained linear combination of the *fl_* estimators.

lr_sl : $\tilde{\beta}(v) =$ trained linear combination of the *fl_* and *sl_* estimators.

Multi-armed bandits (1)



Multi-armed bandits (2)

Budget n , how to maximize the reward?

Balance exploration and exploitation

Applied to focused crawling

Slot machines: estimators

Reward: score of the top node

mab_ε

- probability $1-\varepsilon$: slot machine with the highest average reward
- probability ε : random slot machine

ESTIMATOR 6 (mab_ε). $\tilde{\beta}(v) = \text{output of an epsilon-greedy strategy.}$

mab_ε-first

steps $[0, \lfloor \varepsilon \times N \rfloor]$: random slot machine

steps $[\lfloor \varepsilon \times N \rfloor + 1, N]$: slot machine with the highest average reward

ESTIMATOR 7 (mab_ε-first). $\tilde{\beta}(v) = \text{output of an epsilon-first strategy.}$

mab_var

Succession of ε -first strategies, with a reset every r steps, r varying with the context

ESTIMATOR 8 (mab_var). $\tilde{\beta}(v) =$ *output of an epsilon-first with variable reset strategy.*

Their running times

Expected running times

Twitter API for one week:

- 3s
- 200,000 nodes

One domain website for one week:

- 1s
- 600,000 nodes

Experimental framework (1)

Dataset	Nodes (million)	Non-zero nodes (%)	Edges (million)	Non-zero edges (%)
BRETAGNE	2.2	2.0	35.6	0.5
FRANCE	"	19.2	"	6.8
HAPPY	16.9	11.0	78.0	2.4
JAZZ	"	0.6	"	0.1
WEIRD	"	3.2	"	0.4

Experimental framework (2)

– *Graph score*

10 seed graphs

1 seed graph:

50 seeds picked randomly among non-zero β

Arithmetic average of the crawl scores (sum)

– *Global score*

Normalization with a baseline -- *relative score*

Geometric average among the five graphs

Datasets and code are online

<http://netiru.fr/research/14fc>

To measure the running times

Same crawl sequence: the oracle

Storage in RAM (20G)

3.6 GHz

The running times (ms)

Dataset	Evaluator	100	1,000	10,000	100,000
FRANCE	nr	2,832.1	19,720.5	N/A	N/A
	opic	1.9	2.5	4.6	4.7
	ne_fl	0.2	0.1	0.1	0.1
	lr_fl	0.2	0.2	0.1	0.1
	mab_var_fl	0.6	0.3	0.2	0.2
	ne_sl	8.5	27.1	2.0	6.1
	lr_sl	8.5	27.2	2.0	6.1
HAPPY	nr	45,965.7	105,209.3	N/A	N/A
	opic	1.8	1.6	1.9	2.5
	ne_fl	0.3	0.1	0.2	2.1
	lr_fl	0.5	0.1	0.2	2.1
	mab_var_fl	1.1	0.3	0.5	3.9
	ne_sl	111.1	24.5	63.3	240.5
	lr_sl	111.4	24.5	63.3	241.0

nr

$$NR_1(v)^{t+1} = d \times w(v) + (1 - d) \times \text{avg}_{(v,u) \in E'} \frac{NR_1(u)^t}{d_i(u)}$$

$$NR_2(v)^{t+1} = d \times NR_1(v) + (1 - d) \times \text{avg}_{(u,v) \in E'} \frac{NR_2(u)^t}{d_o(u)}.$$

ESTIMATOR 2 (nr). $\tilde{\beta}(v) = NR_2(v)$.

Quadratic complexity, with large constant factors

Their precision

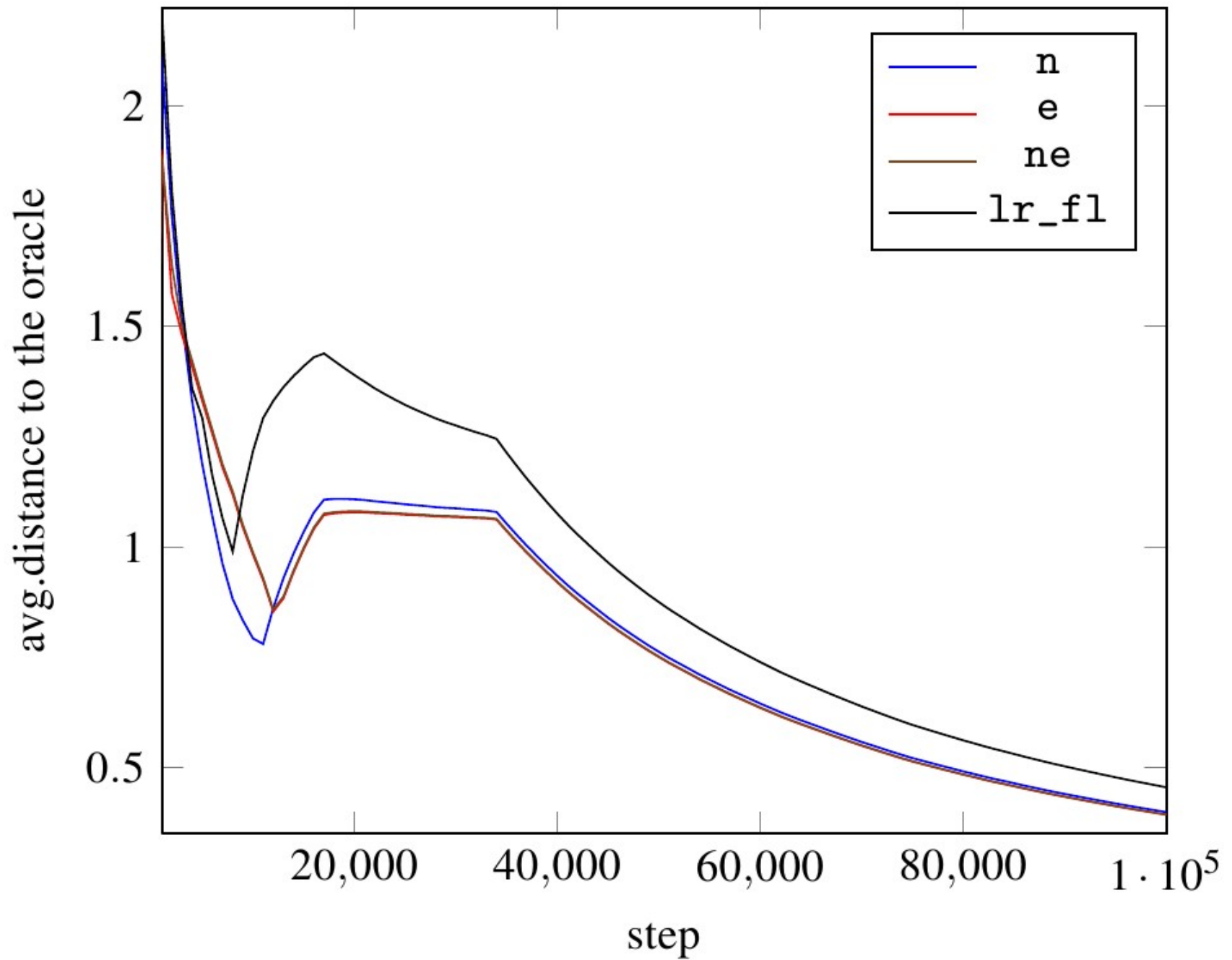
The precision

Same crawl sequence: the oracle

Precision: distance of the top node to the actual top node

Arithmetically averaged over a window of 1000 steps

For bretagne



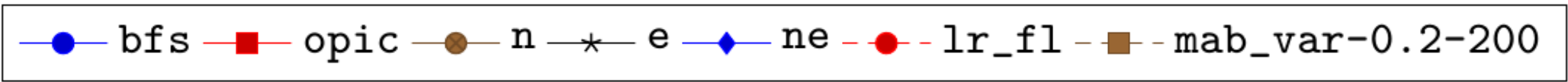
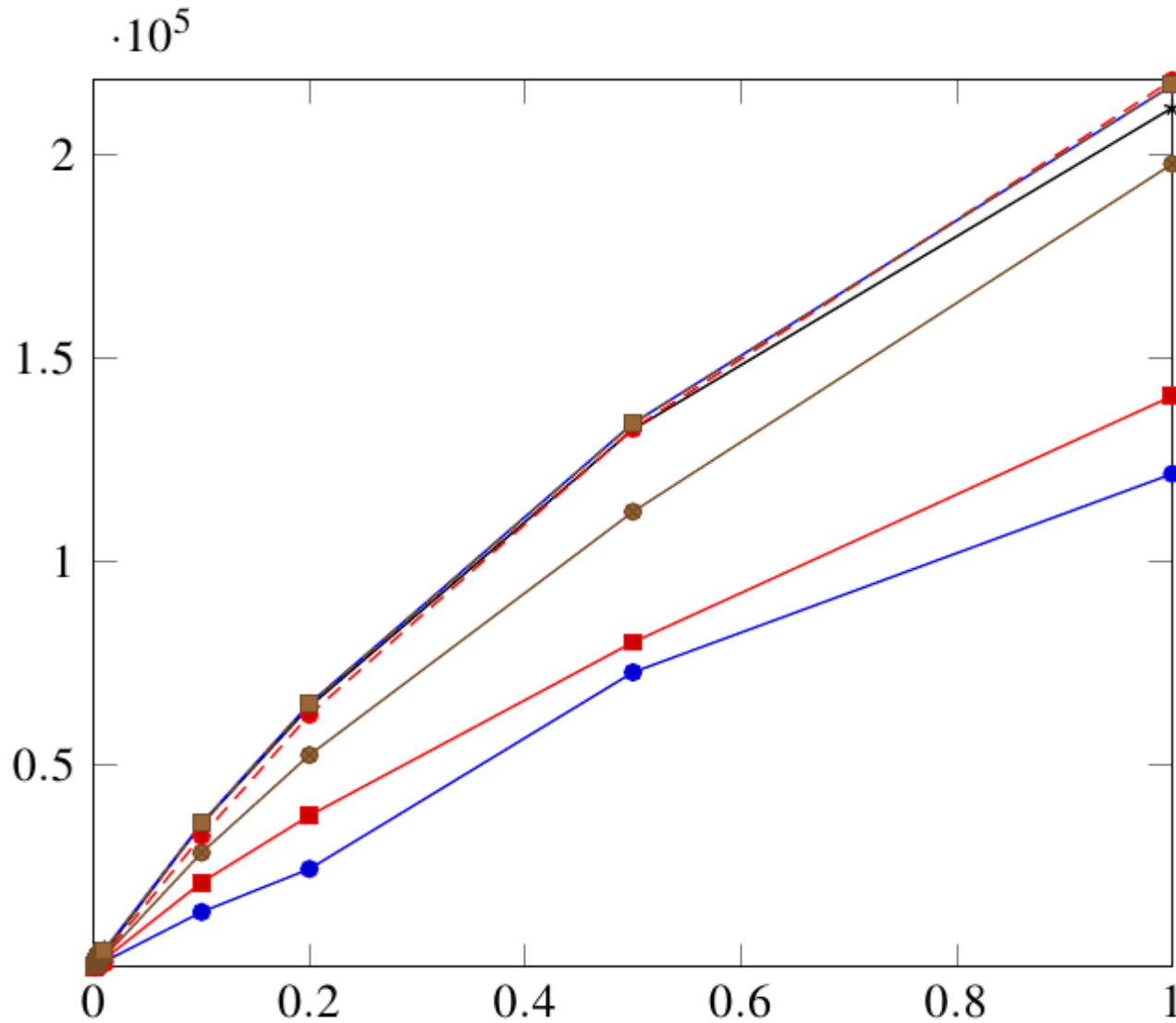
Their ability to lead crawls

Leading the crawl

Different crawl sequences:

defined by the top estimated nodes

Average graph scores for France



The multi armed-bandits

Type	100	1,000	10,000	100,000
ϵ	0.450	0.481	0.477	0.495
ϵ -first	0.409	0.501	0.484	0.490
var-0.1-1000	0.383	0.439	0.420	0.494
var-0.2-200	0.427	0.413	0.461	0.458

All the estimators

Estimator	100	1,000	10,000	100,000
bfs	0.147	0.132	0.130	0.207
opic	0.283	0.184	0.205	0.287
n	0.358	0.280	0.362	0.467
e	0.594	0.560	0.457	0.377
ne	0.583	0.570	0.466	0.378
lr_fl	0.325	0.382	0.466	0.504
mab_var-0.2-200	0.427	0.413	0.461	0.458

Conclusion

What we learnt

Generic model

NP-hardness offline

Refresh rate of 1

Greedy

Neighborhood features

Linear regressions

Multi-armed bandit strategy

Future work

Approximation of the optimal score

Distributed crawl

Recrawling nodes

Further multi-armed bandits comparisons

Thank you.

georges@netiru.fr

Finding the optimal crawl sequences
in a known graph

PTime many-one reduction from the
LST-Graph problem

Problem remains hard if nodes, not edges, are
weighted

A subtree rooted at r is seen as a crawl sequence
starting from r

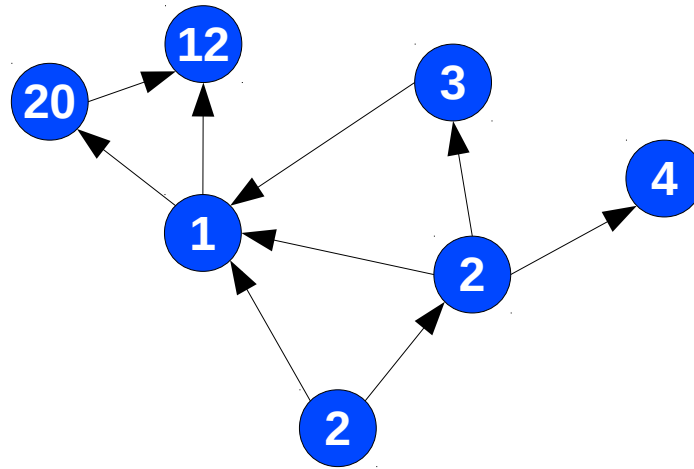
Free edges are added to the graph to allow free
crawls from the seed to any potential root of a
subtree

Rich friends will make you richer

The greedy strategy

Node picked = $\operatorname{argmax}(\beta(v))$, v in frontier

Is not always optimal



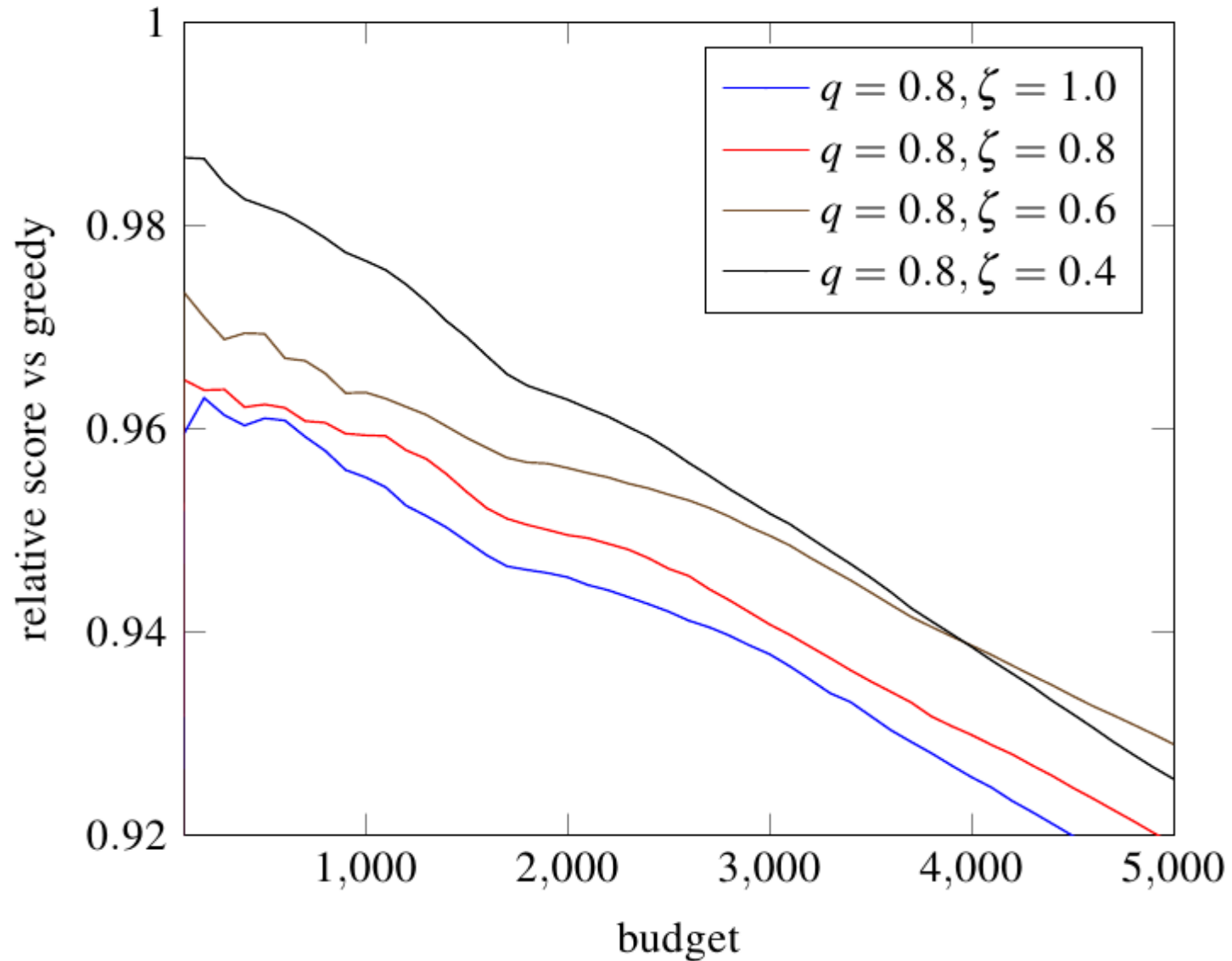
The altered greedy strategy

Node picked =

probability q : $\operatorname{argmax}(\beta(v))$

probability $1-q$: random v so that,
 $\max(\beta(u)) - \beta(v) \leq \zeta \times \max(\beta(u))$

Altered greedy vs greedy for jazz



The refresh rate disadvantage

When estimation takes too long

input : seed subgraph G_0 , budget n
output : crawl sequence V with a score as high as possible

```
1  $V \leftarrow ()$ ;  
2  $G' \leftarrow G_0$ ;  
3 budgetLeft  $\leftarrow n$ ;  
4 while budgetLeft  $> 0$  do  
5     frontier  $\leftarrow$  extractFrontier( $G'$ );  
6     scoredFrontier  $\leftarrow$   
       estimator.scoreFrontier( $G'$ , frontier);  
7      $r \leftarrow$  getRefreshRate();  
8     NodeSequence  $\leftarrow$   
       strategy.getNextNodes(scoredFrontier,  $r$ );  
9      $V \leftarrow (V, \text{NodeSequence})$ ;  
10    for  $u$  in NodeSequence do  
11         $G' \leftarrow G' \cup \text{crawlNode}(u)$ ;  
12        budgetLeft = budgetLeft  $- r$   
13 return  $V$ 
```

The score degradation (%) at different steps

Refresh rate	100	1,000	10,000	100,000
2	0.4	2.2	3.9	6.4
8	1.3	6.5	12.8	18.3
32	6.6	6.5	17.5	24.3
128	38.8	10.7	19.9	29.5
1024	38.8	74.3	25.8	35.9