Bedibe: Datasets and Software Tools for Distributed Bandwidth Prediction

Lionel Eyraud-Dubois, **Przemysław Uznański**

Cepage team, LaBRI, Bordeaux, France

Algotel
May 29-June 01, 2012

Network predictions

problem

- given - set of measures over network (latency or bandwidth)
- result - predicting those values over unmeasured pairs of nodes
- usage - p2p, live streaming

Network predictions

problem

- given - set of measures over network (latency or bandwidth)
- result - predicting those values over unmeasured pairs of nodes
- usage - p2p, live streaming

bandwidth vs latency

- there is less research done on bandwidth prediction
- bandwidth measures are harder to obtain (network heavy)

Exitsting algorithms

latency

- GNP (Global Network Positioning)
- Vivaldi

Exitsting algorithms

latency

- GNP (Global Network Positioning)
- Vivaldi

bandwith

- adaptations of latency algorithms
- Sequoia
- LastMile
- DMF (Decentralised Matrix Factorization)

bedibe - introduction

- Bedibe – to simplify development, testing, benchmarking and visualising prediction algorithms.
- language of choice – Python

bedibe - introduction

- Bedibe – to simplify development, testing, benchmarking and visualising prediction algorithms.
- language of choice – Python

common elements

following aspects are common:

- data
- reading data
- preprocessing of data
- comparing results
- visualisation

input

available datasets

- PlanetLab (large scale, worldwide distibuted platform)
- S^3 project by HP – measures on PlanetLab.
- Splay is a framework aimed at simplifying development of large scale distributed platforms.
- We used Splay deployed on PlanetLab to gather several snapshots from Nov 2011 to Feb 2012 (every few days)
- Snapshots $== \approx 3$ measures per every edge for every pair of nodes for ≥ 100 nodes
- `$data/2011-11-16-2337.band:`

```
212.235.189.115; 157.181.175.248; 1556444.955623
192.43.193.71;   193.10.64.36;     762305.70095747
157.181.175.249; 192.42.43.22;     1047101.0775418
...
```

preprocessing

,,real" value over edge

- we have set of measures over edge
- how to choose ,,real" one?
- median, average, first, min, max, . . .

preprocessing

,,real" value over edge

- we have set of measures over edge
- how to choose ,,real" one?
- median, average, first, min, max, . . .

preparing dataset

- hide some data from algorithm == prepare dataset so it's not a full matrix
- later, compare prediction with original dataset

preprocessing

,,real" value over edge

- we have set of measures over edge
- how to choose ,,real" one?
- median, average, first, min, max, ...

preparing dataset

- hide some data from algorithm == prepare dataset so it's not a full matrix
- later, compare prediction with original dataset

example

```
A = selector.select(A, selector.getmedian)
B = selector.select(A, selector.getfirst)
B = selector.neighbours( B, 10 )
```

algorithm

- consider simple algorithm (Last Mile)
- each node has assigned in- and out-going bandwidth, equal to largest in- or out-going connections
- edge is estimated as minimum of out bandwith for sender and in bandwith for receiver
- ```
  def lastmile(A):
      for i in xrange(n):
          inl[i] = max(A[i,j] for j in xrange(n))
          outl[i] = max(A[j,i] for j in xrange(n))
      for i in xrange(n):
          for j in xrange(n):
              B[i,j] = min(outl[i],inl[j])
      return B
  ```
  ```

algorithm cont.

- How to lift function from matrices to data representation used in enviroment?
- ```
  @data_to_mtrx
  def lastmile(INP):
  ```
- We provide plenty of different decorators (for efficient functional-style programming).
  ```

algorithm cont.

- How to lift function from matrices to data representation used in enviroment?
- ```
  @data_to_mtrx
  def lastmile(INP):
  ```
- We provide plenty of different decorators (for efficient functional-style programming).

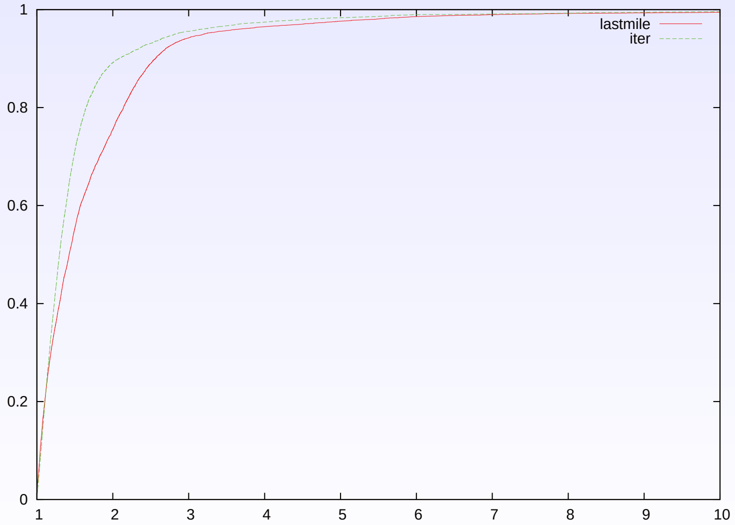
implemented algorithms

- LastMile
- Sequoia
- DMF
  ```

postprocessing

- There are plenty of ways to visualize data, for example as a CDF plot:
- ```
  LM = lastmile.lastmile(A, logexp=False)
  ITER = lastmile.lastmile(A, logexp=True, iterated=True)
  with plot.Plot('exp20') as p:
      p.add_plot(plot.simple_comp(LM, INP),"lastmile")
      p.add_plot(plot.simple_comp(ITER, INP),"iter")
  ```
- As a result, we will get a handy gnuplot script to visualise the plots.
  ```

results



concluding remarks

already done

- common framework
- datasets
- implemented algorithms
- benchmarking

concluding remarks

already done

- common framework
- datasets
- implemented algorithms
- benchmarking

future work

- more algorithms implemented
- better measures
- more features (better visualisation)

Thank you for your attention!