Property Testing and Linear Sketches on graphs

M. de Rougemont
University Paris II
IRIF-CNRS
Plan

1. Property Testing and Streaming  Property Testing
   1.1 Property Testing of regular words
   1.2 Streaming Property Testing of regular trees (XML valid words)
   1.3 Word2vec

2. Linear Sketches for streaming graphs
   2.1 Twitter streams, Window models
   2.2 Connectivity in O(log n) iterations
   2.3 Generation of spanning trees, Mincuts

3. Conclusion
New algorithmic ideas

Example: ECIS 700: algorithms for Big Data (Univ. Penn.)

http://grigory.us/big-data-class.html

- **Streaming Algorithms** (Frequency moments, Count-Sketch, Linear Sketches)
- Selected topics (Compressed sensing)
- Massively Parallel Algorithms (MapReduce)
- **Sublinear Time Algorithms** (Property Testing)
1. Property Testing

Property Testing (Goldreich et al. 1996)

Distance between input structures: Edit distance (words, trees, graphs)
Query model: given a point $i$, provide $Local(i)$

Approximation: an $\varepsilon$-Tester for $L$

- if $w$ in $L$, then Tester accepts
- if $w$ $\varepsilon$-far from $L$, then $\text{Prob}[\text{Tester rejects}] > 2/3$
- $O(f(\varepsilon))$ queries

Property is Testable if there is an $\varepsilon$-Tester for all $\varepsilon$. We decide a property without reading the whole data.

Sublinear algorithms: $O(\log n \cdot f(\varepsilon))$, $O(n^{0.5} \cdot f(\varepsilon))$, $O(1/ \varepsilon)$, $O(2^{1/\varepsilon})$

In practice: $n=10^9$ $\varepsilon=0.1$ between $10^2$ and $10^3$ Samples.
Tester for regular properties of words

Example: \( L=a^* . b^* \)

\[ w_n = \ a \ a \ a \ b \ b \ a \ a \ a \ b \ b \ b \ b \ a \ b \ b \ b \ b \ b \ b \ b \ \quad n=20 \]

\( 3/n = 0.15 \) close to \( a^* b^* \)

**Tester:** sample 2 random positions \( i, j \)

- \( a, a \) or \( b, b \) or \( a, b \) Accept
- \( b, a \) Reject
Tester for regular properties of words

Automaton: Connected components $C_1 C_2 \ldots C_k$

Accepting Paths: $C_1 \cdot C_2$, $C_1 \cdot C_3 \cdot C_4$  \hspace{1cm} m=3

Tester: m samples with the uniform distribution

\textit{Are the samples compatible for an accepting path?}

Analysis: m-moments (generalization of the k-grams) are necessary

Example: $a^*b^*a^*$ needs m=3 samples

\begin{align*}
\quad a^n b^n b^n a^n \\
\quad b^n a^n a^n b^n
\end{align*}

have the same 2-statistics (densities of pairs of words)
3COL (3-Colorability) on dense graphs

NP-hard

**Tester:**

- k random vertices,
- $H_k$ Induced subgraph
- If $H_k$ is 3-colorable, Accept
- Else, Reject

**Analysis:** If $G$ $\varepsilon$-far from 3COL,

$\text{Prob}[\text{Tester Rejects}] > p(\varepsilon)$
**Unranked ordered trees: XML or Json**

**Ordered unranked tree** represented as a balanced word

```
  a b b b a a b a
```

Testing regular tree properties?


1. **Tester needs** $O(\sqrt{n})$ queries!

2. **Streaming Property Tester:** we see all the data but only keep a small fraction, $O(\text{poly}(\log n/\varepsilon))$
Testing a regular tree?

Testing \( w \in a^*b^* \) requires two samples.

Need \( O(n^4) \) samples to test \( T \in c^*[b^*a^*]^* \): birthday's paradox.
Ordered unranked tree represented as a balanced word

\[ a \ b \ b \ b \ a \ a \ b \ a \]

VPA may accept \[ a \ b \ b \ a \]

Tree automaton vs. VPA

1. Non alternating sequence: \[ a \ b \ a \ a \ b \ a \]

\[ w \in L(A) \text{ iff } \tilde{w} \in L(\tilde{A}) \text{ and } \tilde{A} \text{ is a finite automaton (slicing)} \]

2. General sequence:

Construct \( \tilde{w} \) by compression of the peaks
Streaming Property Testing

Compression step
Streaming Property Testing

Iterated Compression: final slicing word
Main Result \[\text{http://arxiv.org/abs/1505.03334}\]

Theorem: Let A be a VPA for L. There is an \(\varepsilon\)-streaming tester for L and Memory \(O(c \cdot \frac{(\log n)^6}{\varepsilon^4})\).

Analysis:

\[t = O(c_1 \cdot \frac{(\log n)^2}{\varepsilon^2})\] samples of weight \(k = O(c_2 \cdot \frac{(\log n)}{\varepsilon})\)

For each \(u\), \(O(c_3 \cdot \frac{(\log n)^2}{\varepsilon})\) decomposition, Stack \(O(\log n)\)

Hence Memory \(O(c \cdot \frac{(\log n)^6}{\varepsilon^4})\).
We invite you to submit a paper to the Second International Symposium on Web Algorithms (iSWAG 2016).

iSWAG is a symposium that covers research in the areas of algorithms for solving web related problems. The first symposium will be held in Deauville, Normandy, France in June 9-10, 2015.

iSWAG seeks innovative papers with strong applicative impact. The symposium will be highly selective, with both invited talks and refereed full papers presented either in oral sessions or in poster sessions.

Topics covered include but are not limited to:

\[
\begin{align*}
\text{ustat}_1^1(w) &= \begin{cases}
3/n : \text{iSWAG} \\
2/n : \text{algorithms} \\
3/n : \text{symposium} \\
\text{....} \\
\text{....}
\end{cases}, \\
\text{ustat}_2^2(w) &= \begin{cases}
2/m : \text{iSWAG algorithms} \\
1/m : \text{Web algorithms} \\
1/n : \text{symposium Deauville} \\
\text{....} \\
\text{....}
\end{cases}, \\
\text{ustat}_3^3(w) &= \begin{cases}
1/p : \text{Web iSWAG algorithms} \\
1/p : \text{Web algorithms 2016} \\
1/p : \text{symposium Deauville in} \\
\text{....} \\
\text{....}
\end{cases}
\end{align*}
\]
Can we find: $x_1, x_2, \ldots, x_n,$
$$x_{1,2}, x_{1,3}, \ldots, x_{n-1,n}$$

Such that
$$x_i = \text{Prob [ word } i \text{ occurs]} = u_{\text{stat}_1}(w)[i]$$
$$x_{i,j} = \text{Prob [ words } i \text{ and } j \text{ occur in a sentence]} = u_{\text{stat}_2}(w)[i,j]$$

Semidefinite program: Each $x_i$ is a vector
$x_{i,j}$ is $x_i \cdot x_j$ the dot product

$$u_{\text{stat}_1}(w) = \begin{cases} 
\frac{3}{n}: \text{iSWAG} \\
\frac{2}{n}: \text{algorithms} \\
\frac{3}{n}: \text{symposium} \\
\frac{1}{n}: \text{Deauville} \\
\text{...}
\end{cases}$$

Word2vec solution:

[Link to tutorial](http://rare-technologies.com/word2vec-tutorial/)
Applications of Word2vec

Applications:
1. Paris – France + Germany close to ?

2. Blue, white, red, person Which word is the odd one out?

3. Queries to a story: John threw the ball, Mary picked it up. Who has the ball ?

Problems:
Words are static. Transformations \( v'_{\text{ball}} = A_{\text{throw}} \cdot v_{\text{ball}} \)

Games on words: structure of the transformations \( A_1, A_2, \ldots A_m \)
II. Linear Sketches of Graphs

Twitter: Json stream(#s)

@x: Thanks @y about #t

Twitter Graph: G=(V_n, E)

Stream: +e_1, +e_2, +e_3 …… +e_m

Graph H restricted to the last 100 edges

H: +e_{m-100} …… +e_m - e_{m-100} + e_{m+1}
H: +e_{m-99} …… +e_m + e_{m+1}

@x

@y

#t
Stream: \(+e_1, +e_2, -e_3 \ldots -e_m\)
Positive and negative edges

Graph representation:
- Adjacency matrix, list \(O(n^2)\)
- Sketches \(O(n \log n)\)

Can we solve classical problems?
- Connexity
- Mincut
- \#Spanning trees, forests?
Linear Sketches of Graphs

\[ a_1 = (1, 1, 0, 0, 0, \ldots, 0) \]

\[ (1,2),(1,3),(1,4),(1,5),(2,3),\ldots,(4,5) \quad n(n-1)/2 \]

\[ a_2 = (-1, 0, 0, 0, 1, \ldots, 0) \]

\[ a_3 = (0, -1, 0, 0, -1, \ldots, 0) \]

\[ y_i = M_i \cdot A_i \]

\[ \log n \]

\[ = \]

\[ M \]

\[ a_S = a_1 + a_2 = (0, 1, 0, 0, 1, \ldots, 0) \]

\[ (i,j): 1 \text{ if } i < j \]

\[ -1 \text{ if } i > j \]

Sparse vector
Properties of Linear Sketches

• Linearity for a new edge +\((1, 4)\) or \(-(1, 4)\)

\[
a_1 = (1, 1, 0, 0, 0, \ldots, 0) \\
a_{(1,4)} = (0, 0, 0, 1, 0, \ldots, 0) \quad \text{base vector}
\]

\[
a_1^{\text{new}} = (1, 1, 0, 1, 0, \ldots, 0) \\
y_i^{\text{new}} = M_i \cdot a_i^{\text{new}} = M_i \cdot a_i + M_i \cdot a_{(1,4)}
\]

only computation for \(i=1\) or 4
Properties of Linear Sketches

• Use of sketches: given $S$ find an edge leaving $S$ uniformly

$$y_S = \sum_{i \in S} (M.a_i) = \sum y_i$$

Compressed Sensing: given $y$ and $M$, find a

– Sparse $a_i$

– Unicity

– Reconstruction via Linear Programming

– Sample one of edges uniformly
Compressed Sensing

\( y = M \cdot x \)

\( y \) is the compressed signal

Can we find \( x \), from \( M \) and \( y \) if \( x \) is sparse?

Candes, Tao 2004: yes for certain \( M \)

- Minimize \( L_0 \) is NP-hard
- Minimize \( L_1 \) by a linear program
- Many variations

We don’t search for \( x \), but for a random \( i \) s.t. \( x_i \neq 0 \)
Essentials of Linear Sketches

• Disadvantage

Is \((i,j)\) an edge? Only a probabilistic answer at cost 

\[O(n) \text{ (worst-case)}\]

• Advantage

For any partition \(S_1, S_2 \ldots S_k\) of \(V\), obtain a uniform edge leaving each \(S_i\) at a unit cost.
Connectivity

Distributed algorithm

1. Each node outputs an edge
2. Merge the connected components into new nodes
3. Repeat 1, until no more edges are found.
4. $G$ is connected iff only 1 node is found

$O(\log n)$ iterations
New ideas

1. Generation can be made efficient

2. Many algorithms can be implemented in Map-Reduce

3. #SpanningTrees: count the number of spanning trees

4. Application to Mincuts and Community detection without storing the whole graph.
Newcomers: quantum learning algorithms

• HHL (Harrow, Hassidim, Lloyd) 2009
  – Solve $A.x=b$ in $O(poly(\log n))$

• Scott’s view: [http://www.scottaaronson.com/blog/?p=2196](http://www.scottaaronson.com/blog/?p=2196)
  – K-means
  – SVM
  – Data fitting
  – Pagerank vectors


• Probabilistic algorithms can do almost as well
Counting spanning trees: P (matrix tree theorem), need Adjacency matrix

Counting spanning forests: \#P hard
Uniform Generation

- Approximate uniform generation equivalent to Approximate counting

- AB (Aldous-Broder) random walk:
  - Keep the first edge entering node $i$
  - When all nodes reached, Output Spanning tree

Too long (Cover time)
Distributed Uniform Generation

1. Each node outputs an edge
2. Spanning tree in each connected components: merge into new nodes
3. Repeat 1, until no more edges are found.
4. Uniform Spanning tree

MapReduce implementation
Log n iterations
Online Mincuts

Assume a stream of +/-edges

1. Get k uniform spanning trees
2. On + edges, update the trees, on – edges recompute the spanning trees
3. Keep the most frequent edges
4. Find the Mincut

Assume a random permutation of all the edges:

Mincut=0  Mincut=1 (100%  )  Mincut=2 (50%  )
Conclusion

1. Property Testing
   - Streaming versus Property Tester
   - Streaming Property Tester for VPL (regular trees) $O(\text{poly}(\log n))$
   - $m$-moments statistics

2. Linear Sketches
   - Store $O(n \cdot \log n)$
   - Connexity
   - Spanning trees
   - Mincuts

2. Online Community detection on streams