The Right Way to Search Evolving Graphs

Jiahao Chen & Weijian Zhang

GABB 2016, Chicago 05-23
The world of Julia

445 contributors to Julia repo
808 packages, 726 authors
6,841 stargazers
549 watchers

https://github.com/jahao/julia-notebooks
Jack likes Les Mis.
Jack tells Elsa about Les Mis
Both Jack and Elsa now like Les Mis.
Elsa tells Leo about Les Mis.
Now everyone likes Les Mis.
Leo heard about Les Mis from Jack.
Leo did not hear about The Hunger Games from Jack.

Information flow on a network is causal.
Information flow on a network is causal.

Leo did not hear about The Hunger Games from Jack.

Unless you have one of these...
Temporal paths: causal connections in space and time

In general, not trivial to find all temporal paths

(Jack, Mon) → (Elsa, Mon) → (Elsa, Tues) → (Leo, Tues)
Temporal paths: causal connections in **space** and **time**

Grindrod et al. (2011) only count static edges

Tang, Musolesi, and Mascolo (2009) only count causal edges
Active and inactive nodes

Temporal paths connect active nodes only
The same node may be **active** or **inactive** at different times.
The forward neighbor of (Jack, Monday) is (Elsa, Monday)
The forward neighbor of (Jack, Monday) is (Elsa, Monday)

The forward neighbor of (Elsa, Monday) is (Elsa, Tuesday)
The forward neighbor of (Jack, Monday) is (Elsa, Monday)
The forward neighbor of (Elsa, Monday) is (Elsa, Tuesday)
(Elsa, Tuesday) is reachable from (Jack, Monday)
All **forward neighbors** are **active nodes**

Forward neighbors

- Jack
- Leo
- Elsa

Active node:
- Elsa
- Leo
- Jack

Day:
- Monday
- Tuesday
Breadth-first search finds forward neighbors
Breadth-first search as a matrix-vector product

Breadth-first search as a matrix-vector product

\[
\begin{bmatrix}
J & E & L \\
0 & 1 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
Breadth-first search as a matrix-vector product

Where does the temporal edge go?
Static graph correspondence

Problem: does not encode time ordering of edges
Static graph correspondence

Problem: does not encode time ordering of edges
Static graph correspondence

(J, Mon) (E, Mon) (L, Mon) (J, Tue) (E, Tue) (L, Tue)

\[
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
Breadth-first search as a matrix-vector product

<table>
<thead>
<tr>
<th></th>
<th>Monday</th>
<th></th>
<th>Tuesday</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Elsa</td>
<td>0</td>
<td>Elsa</td>
<td>0</td>
<td>Elsa</td>
</tr>
<tr>
<td>Jack</td>
<td>1</td>
<td>Jack</td>
<td>0</td>
<td>Jack</td>
</tr>
<tr>
<td>Leo</td>
<td>0</td>
<td>Leo</td>
<td>1</td>
<td>Leo</td>
</tr>
</tbody>
</table>

The diagram on the left shows the relationships between Jack, Elsa, and Leo with static and causal edges. The matrix represents the connectivity between these individuals on two days, Monday and Tuesday.
Breadth-first search as a matrix-vector product

\[
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}
= \begin{bmatrix}
1 \\
0 \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\]
Breadth-first search as a matrix-vector product

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
x \\
A^T x \\
(A^T)^2 x \\
(A^T)^3 x \\
\end{bmatrix}
\]
Problem: adjacency matrix of corresponding static graph is big
Adjacency matrix has structure!

<table>
<thead>
<tr>
<th>(J, Mon)</th>
<th>(E, Mon)</th>
<th>(L, Mon)</th>
<th>(J, Tue)</th>
<th>(E, Tue)</th>
<th>(L, Tue)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Diagonal projection matrix

Can compute matrix-vectors products efficiently

Always 0
Can’t go back in time!
Conclusions

1. Breadth-first search over temporal paths can be expressed as matrix-vector products
2. Evolving graphs correspond to static graphs with special structure in the adjacency matrix, enabling fast matvecs
3. Implemented in EvolvingGraphs.jl in Julia
What’s next?

1. Parallel implementation
2. Shortest temporal path algorithm using semiring algebra
3. Graph centrality algorithms (PageRank, eigenvector centrality)
4. Analyze citation networks
References

- **Evolving graph models**
  - Leskovec et al. (2008) Microscopic Evolution of Social Networks

- **Evolving graph metrics and centralities**
  - Tang, Musolesi, and Mascolo (2009) Temporal distance metrics for social network analysis
  - Tang et al. (2010) Analysing information flows and key mediators through temporal centrality metrics