Exploring Robustness in Dynamic Graphs

Michael Uftring
Indiana University
I606 - Network Science
Spring, 2018

Overview

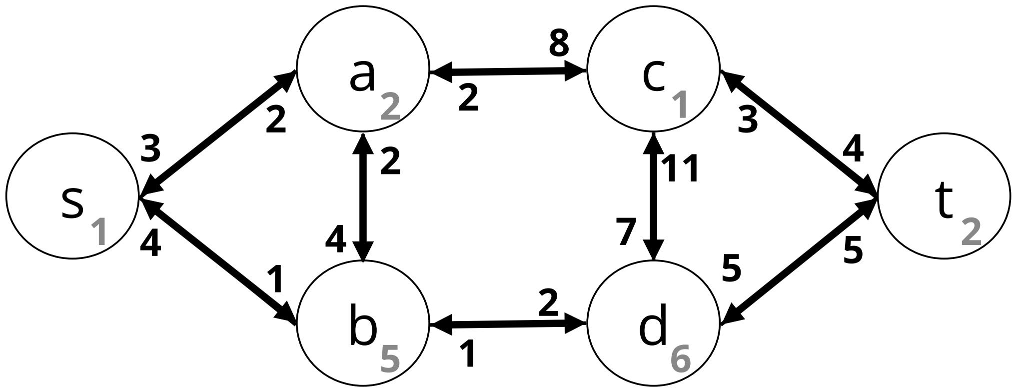
- Dynamic Graphs
- Robustness
- Exploration

Dynamic Graphs

- Graph Theory
- What is a Dynamic Graph?
- Real-world Examples
- Dynamic Graph Theory
- Issues with Algorithms
- Representing Dynamic Graphs

Graph Theory

- A graph G is a pair (V, E), where V is a finite set of vertices or nodes, and E is a set of edges, each being an unordered pair {u, v} of distinct nodes.
- Vertices (nodes) represent elements or components
- Edges (links) represent relationships
- Directed Graphs: edges have a direction
- Weighted Graphs: vertices or edges have a value
- *G = (V, E, f, g)* where *f* and *g* are functions returning weights for nodes and edges



What is a Dynamic Graph?

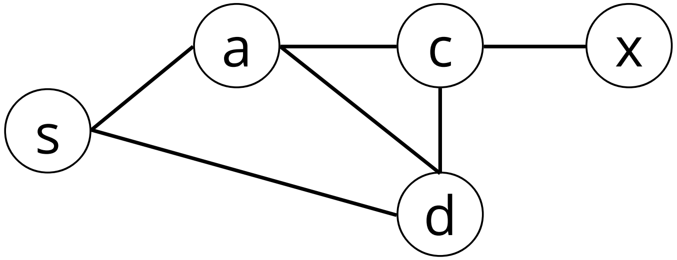
- Graphs which *change over time*
- Recall, graphs and networks involve the following entities:
    - *V* (a set of vertices)
    - *E* (a set of edges)
    - *f* (map vertices to numbers)
    - *g* (map edges to numbers)

weights

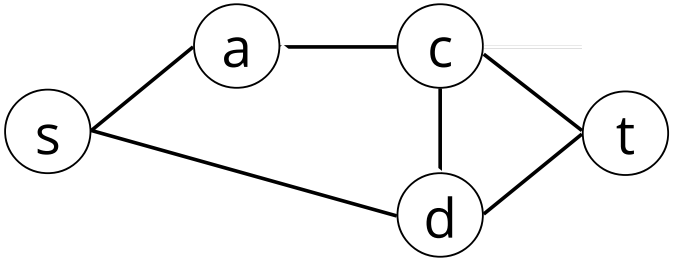
- A dynamic graph is obtained when any of these entities *changes over time*
- Also called:
    - evolving graphs
    - temporal graphs
    - time-varying graphs

Dynamic Graph Example

Time = *t*

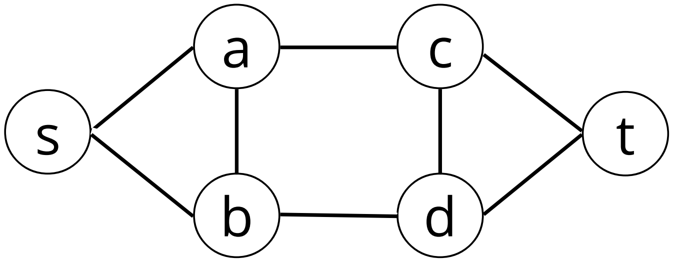


Time = *t+1*



1. remove node x
2. remove edge (a, d)
3. add node t
4. add edge (c, t)
5. add edge (d, t)

Time = *t+2*



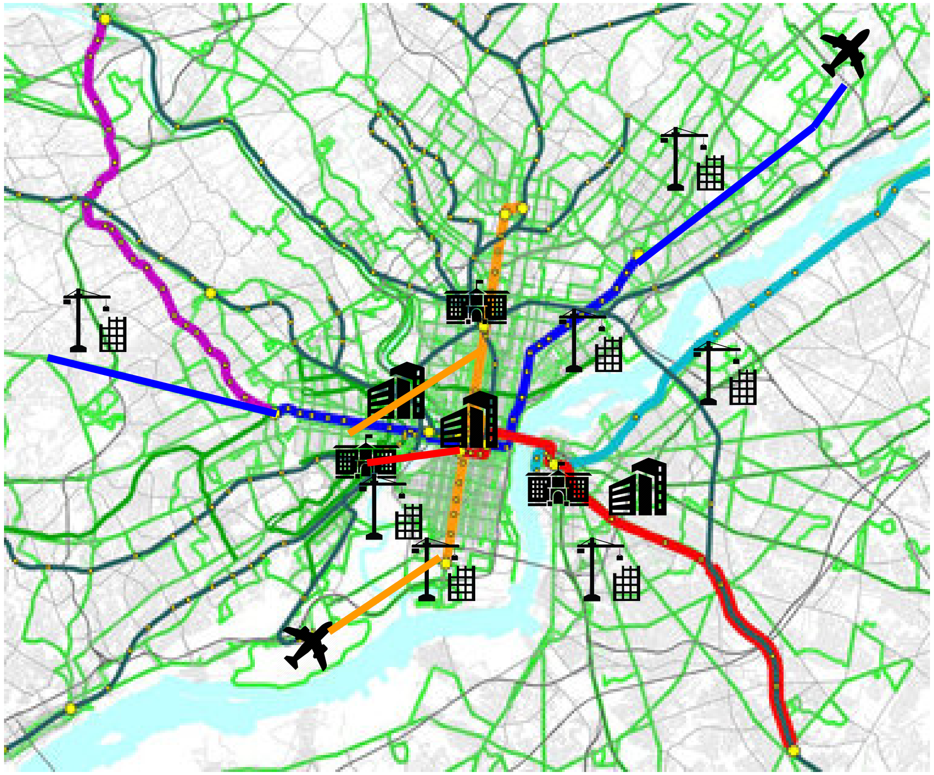
1. remove edge (s, d)
2. add node b
3. add edge (a, b)
4. add edge (b, d)
5. add edge (b, s)

Real World Examples

- Programming Languages
- Computer Networks
- Mobile Communication Networks
- Social Networks
- Metabolic Network
- Neural and Brain Networks
- Transportation

Real World Examples

Transportation Networks



Philadelphia, PA

Legend

- roads (major, minor)
- heavy rail
- light rail, subway
- airport
- business center
- university
- new development

Dynamic Graph Theory

- Types
- Classifications
- Models

Types

- There are several types of dynamic graphs, some of which are:
  - **node-dynamic**: the set of V varies over time; some nodes may be added or removed
  - **edge-dynamic**: the set of E varies over time; edges may be added or removed
  - **weight-dynamic**: the weights on vertices or edges changes

Classifications

- Based on allowed operations:
  - **fully dynamic**: update operations allow unrestricted insertions and deletions of edges or vertices
  - **partially dynamic**: only one type of update operations is permitted
  - **incremental**: only insertions are allowed
  - **decremental**: only deletions are allowed

Models

- The TVG Model
- Barabasi-Albert Model (extended)

The TVG Model

- A time-varying graph (TVG) is described by:

$$\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$$

-
  - V is the set of vertices
  - E is the set of edges
  - $\mathcal{T}$ a time span, the lifetime of the system, where $\mathcal{T} \subseteq \mathbb{T}$
  - $\rho : E \times \mathcal{T} \to \{0, 1\}$ presence function, indicates whether an edge is available at a given time
  - $\zeta : E \times \mathcal{T} \to \mathbb{T}$ a latency function
- The model can be extended with node centric functions
  - $\psi : V \times \mathcal{T} \to \{0, 1\}$ presence function, indicates whether a node is available at a given time
  - $\varphi : V \times \mathcal{T} \to \mathbb{T}$ a latency function

A.Casteigts et al, *Time-Varying Graphs and Dynamic Networks*, **Proc. Adhoc-Now'11**, 2010

Barabasi-Albert Model (extended)

- Extensions to the Barabasi-Albert model
- Addresses real-world issues (shortcomings of the BA model)
- Captures a wide range of phenomena that shape the topology of real networks
- Extensions:
  - Initial Attractiveness
  - Internal Links
  - Node Deletion
  - Accelerated Growth
  - Aging

Models capturing the mechanisms that govern the time evolution of a network.

Issues with Algorithms

> “
>
> *In a typical dynamic graph problem one would like to* ***answer queries on dynamic graphs****, such as, for instance, whether the graph is connected or which is the shortest path between any two vertices.*
>
> *The goal of a dynamic graph algorithm is to* ***update efficiently*** *the solution of a problem after dynamic changes, rather than having to recompute it from scratch each time.*

C.Demetrescu et al, *Dynamic Graphs*, **Handbook of Data Structures and Applications, 1ed**, CRC Press, 2001

Example:

finding a path between two vertices in a graph such that the sum of the weights of its constituent edges is minimized

The Shortest Path Problem

- What issues arise in dynamic graphs?
  - In a dynamic graph, vertices and edges may be added, removed, or changed (e.g., weights) over time.
  - The shortest path between nodes a and b at time t_1 may not be the same as at time t_2.
- How can this problem be solved?
  - recompute after every change to the graph <- EXPENSIVE
  - adapt the algorithms to *update* in response to changes
    - *that is, recompute just a portion of the solution given by the events of change*

Shortest path problem, **Wikipedia**, https://en.wikipedia.org/wiki/Shortest_path_problem

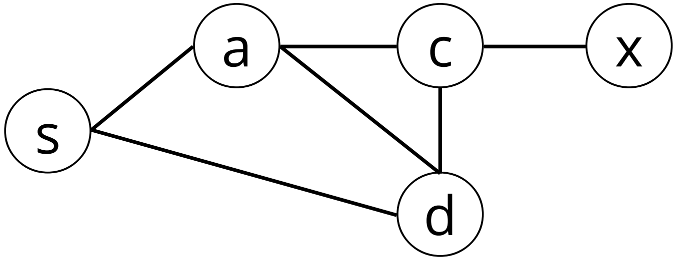
Representing Dynamic Graphs

- A sequence of Adjacency Matrices
- Logic Programming
- Active (Executable) Model

A dynamic graph can be viewed as a discrete sequence of static graphs.

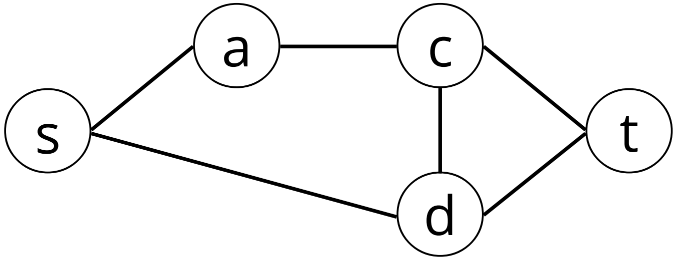
Sequence of Adjacency Matrices

Time = *t*



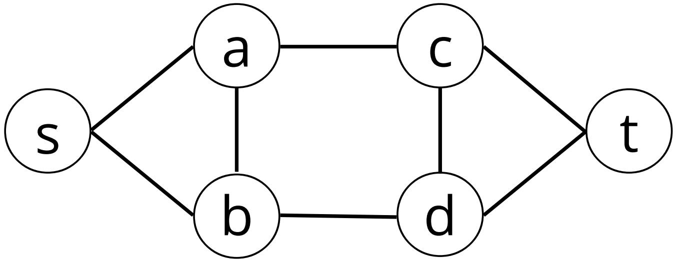
	a	b	c	d	s	t	x
a	0	-	1	1	1	-	0
b	-	-	-	-	-	-	-
c	1	-	0	1	0	-	1
d	1	-	1	0	1	-	0
s	1	-	0	1	0	-	0
t	-	-	-	-	-	-	-
x	0	-	1	0	0	-	0

Time = *t+1*



	a	b	c	d	s	t	x
a	0	-	1	0	1	0	-
b	-	-	-	-	-	-	-
c	1	-	0	1	0	1	-
d	0	-	1	0	1	0	-
s	1	-	0	1	0	0	-
t	0	-	1	1	0	0	-
x	-	-	-	-	-	-	-

Time = *t+2*



	a	b	c	d	s	t	x
a	0	1	1	0	1	0	-
b	1	0	0	1	1	0	-
c	1	0	0	1	0	1	-
d	0	1	1	0	0	1	-
s	1	1	0	0	0	0	-
t	0	0	1	1	0	0	-
x	-	-	-	-	-	-	-

Logic Programming

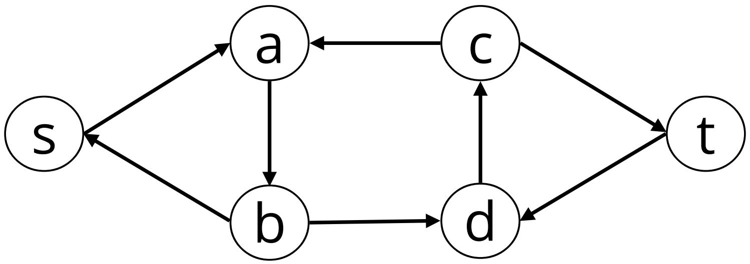
- Based on a subset of predicate logic (Horn Logic)
- Predicates can also be viewed as relations
- **Fact**: a predicate or relation over some terms, states the predicate is unconditionally true
- **Rule**: predicates used for stating conditional truths, can be a conjunction, inclusive or exclusive disjunction
- Every graph can be represented as a simple logic program

Something to think about...

Example: *reachability*

- Facts used represent the directed graph

- Rules represent the notion of reachability



F.Harary, G.Gupta, *Dynamic Graph Models*, **Mathl. Comput. Modelling**, Vol.25, No.7, pp.79-87, 1997

Robustness

- What is Robustness?
- Evaluating Robustness

What is Robustness?

- How does a graph (or network) fare in the presence of faults?
- What are *faults*?
    - **Failure**: a random node or link fails
    - **Attack**: a specific node or link is targeted, and fails

*Many natural and social systems have a remarkable **ability to sustain their basic functions even when some of their components fail**.*

Robustness:

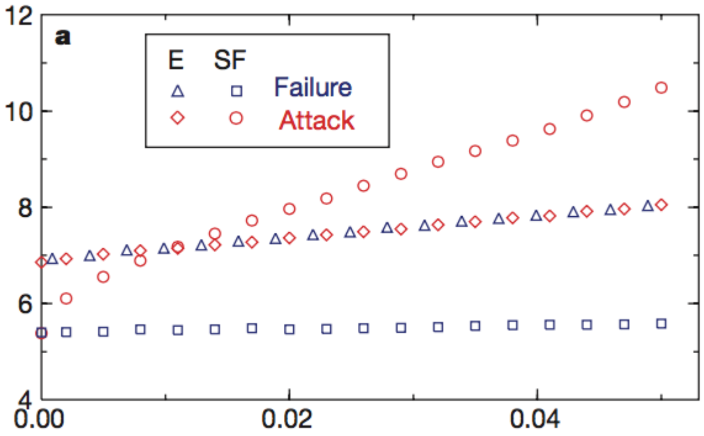
*the primary interest is in **knowing the impact of node failures** on the integrity of a network*

A.L.Barabasi, *Network Science*, Cambridge University Press, 2016

Evaluating Robustness

- Tolerance
  - tolerance is not shared by all networks

	failure	**attack**
Exponential	tolerant	tolerant
Scale-free	tolerant	vulnerable



Using *diameter* as the measure, what is the impact of **failure** and **attack** on Exponential and Scale-free networks?

a measurement, or a perception

R.Albert, H.Jeong, A.L.Barabasi, *Error and attack tolerance of complex networks*, **Nature**, 406, pp. 378-382, 2000

Exploration

- Evaluation
- Experiment
- Results
- Demonstration
- Conclusion
- What's Next?

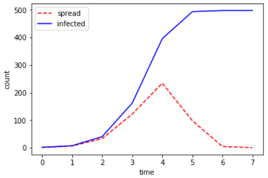
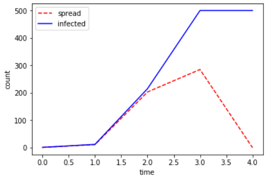
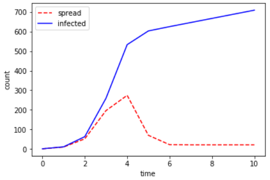
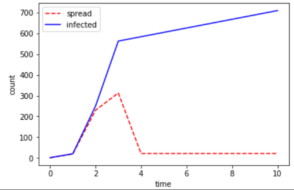
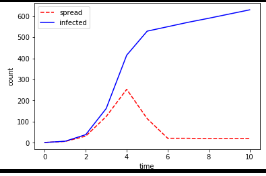
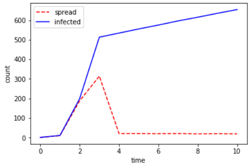
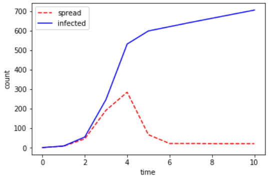
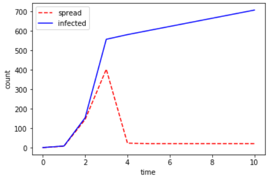
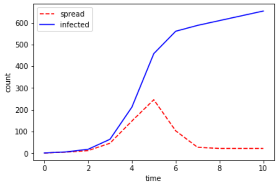
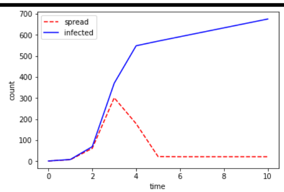
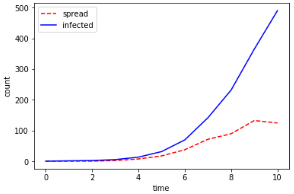
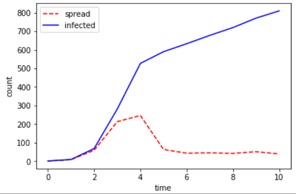
Evaluation

- Software packages and libraries evaluated
  - NetworkX
  - DyNetX
  - NDlib
  - Gephi
  - Cubix

Experiment

- Spreading as a measure of Robustness
  - Jupyter Notebook
  - NetworkX
  - Gephi
  - DIY Model
- Scenarios
  - spreading with additive
  - random detractive
  - targeted detractive (invasive)

Results

<table>
<tr><th></th><th>Erdos-Renyi</th><th>Scale-Free</th></tr>
<tr><td>spreading only</td><td>
</td><td>
</td></tr>
<tr><td>additive only</td><td>
</td><td>
</td></tr>
<tr><td>random detraction</td><td>
</td><td>
</td></tr>
<tr><td>targeted detraction</td><td>
</td><td>
</td></tr>
<tr><td>invasive detraction</td><td>
</td><td>
</td></tr>
<tr><td>extremely invasive</td><td>
</td><td>
</td></tr>
</table>

Demonstration

Jupyter Notebook + Gephi

Conclusion

- Robustness of Dynamic Networks
- Considering Spreading as the measure
- Random Networks (Erdos-Renyi model)
  - tolerant to random and targeted failures
  - vulnerable to only the most invasive
- Scale-Free Networks (Barabasi-Albert model)
  - tolerant to random and small-scale targeted failures
  - some vulnerability in invasive attacks
  - hidden strength in low-degree nodes?

What's Next?

- Enhance Simulation Models
  - order of actions, specify using rates
- Longer Running Simulations
- Different Models
  - cascading, link-centric
- Different Spreading Phenomena
  - reinforcement
  - resilience: vaccination - recovery
- Richer Tracking of Dynamic Processes
- Visualization in Cubix

Thank you!