

CASOS



Case Study: Trails

Binxuan Huang

binxuanh@cs.cmu.edu

CASOS

Societal Computing, School of Computer Science, Carnegie Mellon University

Carnegie Mellon

Center for Computational Analysis of
Social and Organizational Systems
<http://www.casos.cs.cmu.edu/>

- Introduction of trail, Markov transition network
- Case study of health trails
- Current research progress
- Hands on session

Introduction

- Trail: a path of an object through time and space

Time	4 pm@Apr. 1	3 pm@Apr. 2	9 am@Apr. 3	1 pm@Apr. 3	2 pm@Apr. 4	4 pm@Apr. 5
Trail 1	L1	L2	L3	L2	L1	L2
Trail 2	L2	L3	L4	L2	L1	L1
Trail 3	L2	L3	L1	L1	L2	L3

	L1	L2	L3	L4
L1	2	3	0	0
L2	2	0	4	0
L3	1	1	0	1
L4	0	1	0	0

Traffic flow network

$$P(L_i \rightarrow L_j) = \frac{N(L_i \rightarrow L_j)}{\sum_j N(L_i \rightarrow L_j)}$$

	L1	L2	L3	L4
L1	0.4	0.6	0	0
L2	0.33	0	0.67	0
L3	0.33	0.33	0	0.33
L4	0	1	0	0

Markov transition network



Introduction

- Adding BEGIN/END locations to capture more information

Time	4 pm@Apr. 1	4 pm@Apr. 1	3 pm@Apr. 2	9 am@Apr. 3	1 pm@Apr. 3	2 pm@Apr. 4	4 pm@Apr. 5	4 pm@Apr. 5
Trail 1	BEGIN	L1	L2	L3	L2	L1	L2	END
Trail 2	BEGIN	L2	L3	L4	L2	L1	L1	END
Trail 3	BEGIN	L2	L3	L1	L1	L2	L3	END

Traffic flow network

	B	L1	L2	L3	L4	E
B	0	1	2	0	0	0
L1	0	2	3	0	0	1
L2	0	2	0	4	0	1
L3	0	1	1	0	1	1
L4	0	0	1	0	0	0
E	0	0	0	0	0	0

Markov transition network

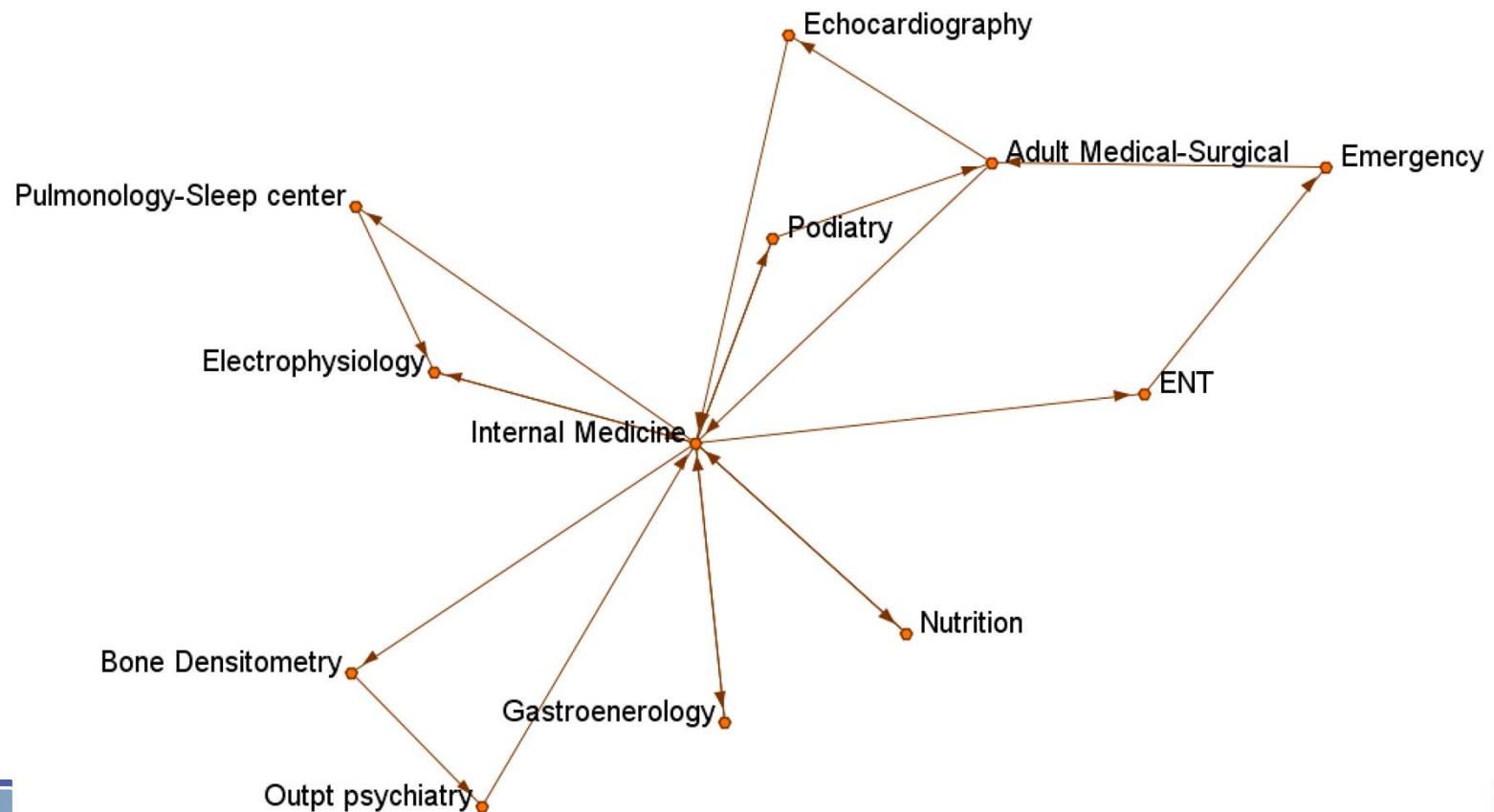
	B	L1	L2	L3	L4	E
B	0	0.33	0.67	0	0	0
L1	0	0.4	0.6	0	0	1
L2	0	0.29	0	0.57	0	0.14
L3	0	0.25	0.25	0	0.25	0.25
L4	0	0	1	0	0	0
E	0	0	0	0	0	0



Case study

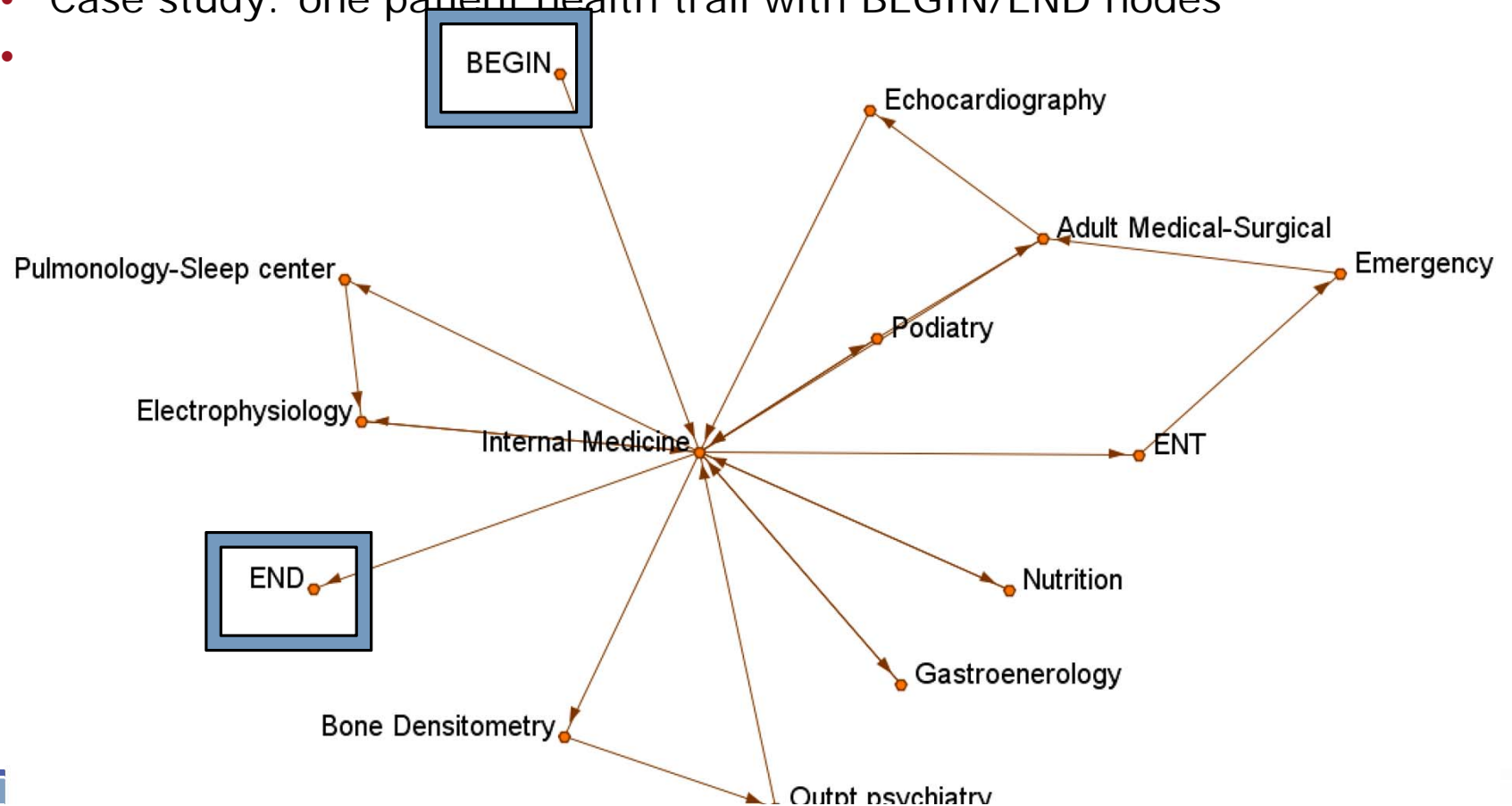
- Case study: one patient health trail

-



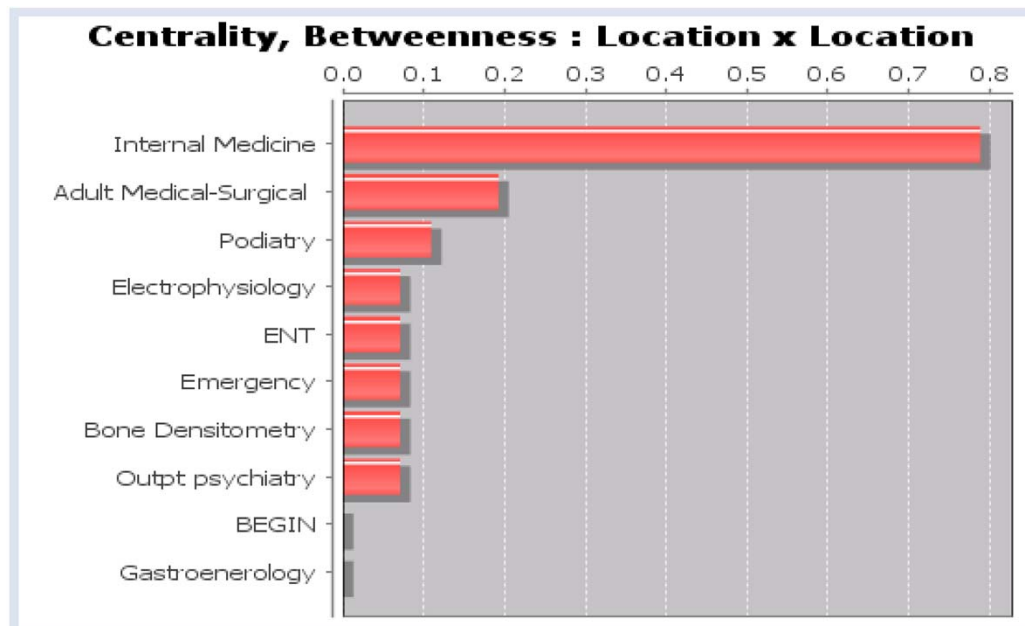
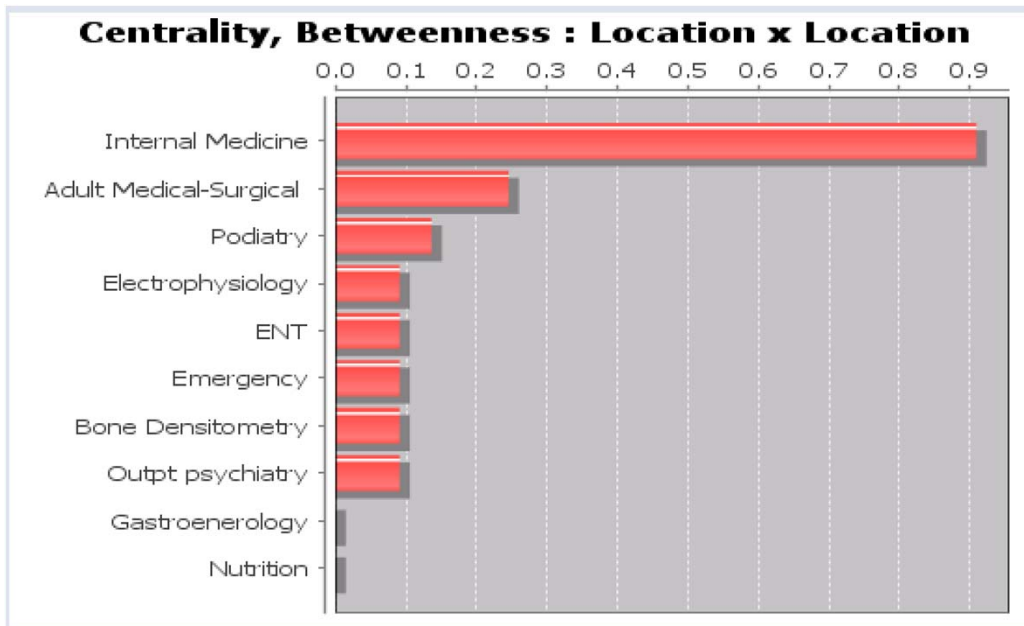
Case study

- Case study: one patient health trail with BEGIN/END nodes



Case study

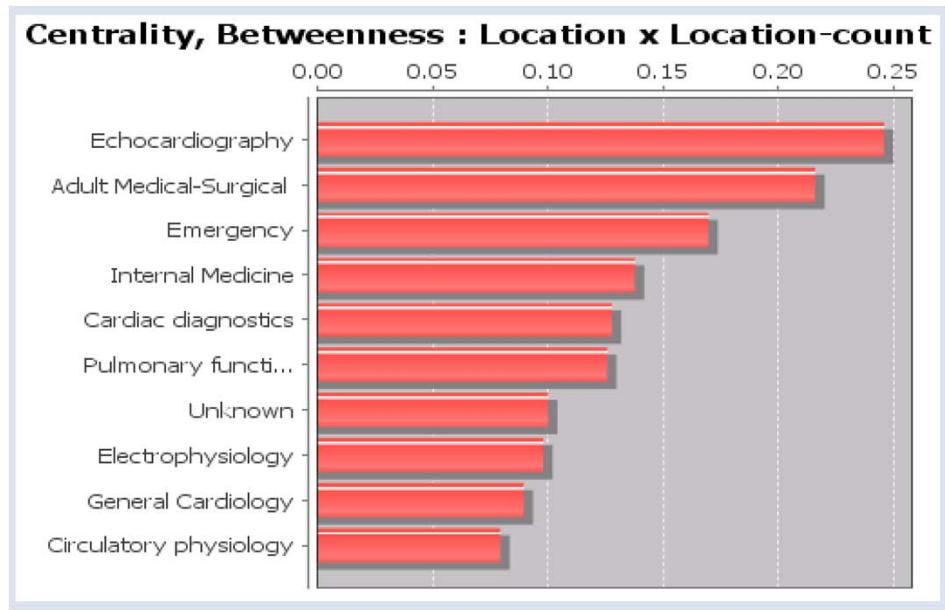
- Compare between one patient health trail with/without BEGIN/END
- Without BEGIN/END



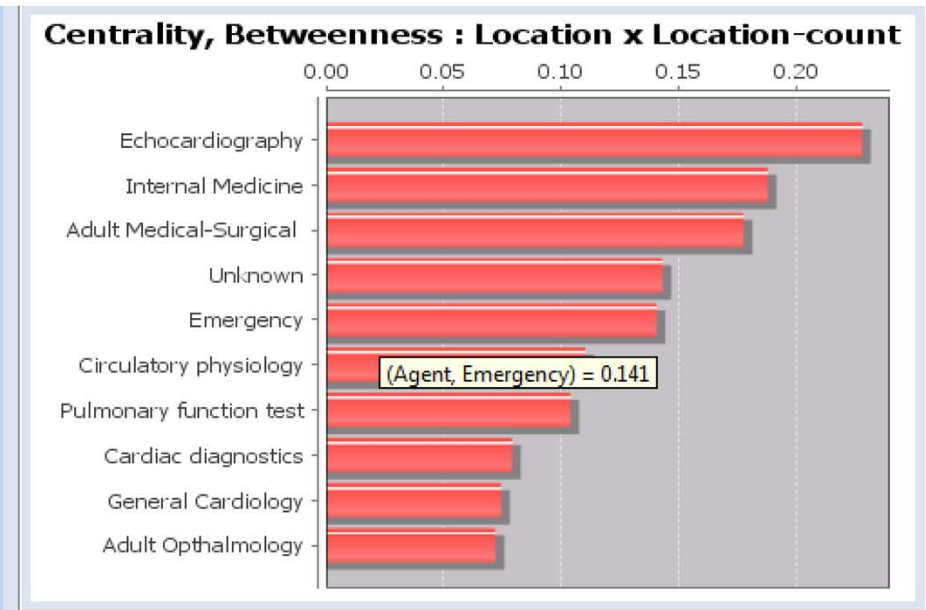
Case study

- Compare between group of health trails with/without BEGIN/END

Without BEGIN/END



With BEGIN/END



Current progress: Overcome the Low Time Resolution in Trails

- | Time | 4 pm@Apr. 1 | 3 pm@Apr. 2 | 9 am@Apr. 3 | 1 pm@Apr. 3 | 2 pm@Apr. 4 | 4 pm@Apr. 5 |
|---------|-------------|-------------|-------------|-------------|-------------|-------------|
| Trail 1 | L1 | L2 | L3 | L2 | L1 | L2 |
| Trail 2 | L2 | L3 | L4 | L2 | L1 | L1 |
| Trail 3 | L2 | L3 | L1 | L1 | L2 | L3 |

Lower time resolution ↓ Broken point

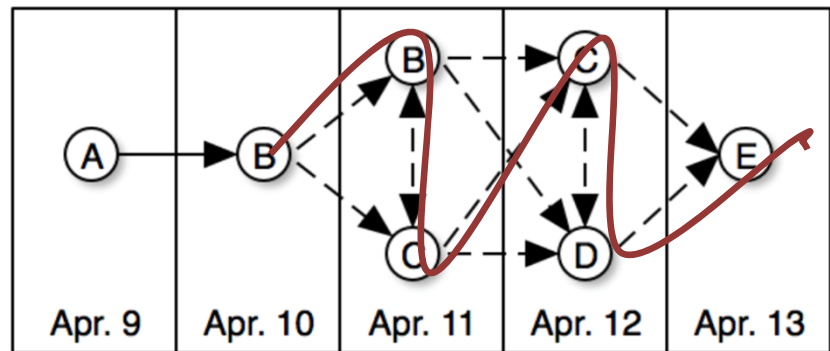
Time	Apr. 1	Apr. 2	Apr. 3	Apr. 3	Apr. 4	Apr. 5
Trail 1	L1	L2	L3	L2	L1	L2
Trail 2	L2	L3	L4	L2	L1	L1
Trail 3	L2	L3	L1	L1	L2	L3

Current progress: Overcome the Time Resolution Issue in Trails

- For a location sequence in a broken point: Find a location sequence with maximum transition probability product
 - $p(l_1 \rightarrow l_2 \rightarrow \dots \rightarrow l_n) = p(l_2|l_1)p(l_3|l_2) \dots p(l_n|l_{n-1})$
- Relation with Asymmetric Traveling Salesman Problem(ATSP): visit each location exactly once and find the minimum travelling cost(NP-hard)

• -lo

Time	2016.4.9	2016.4.10	2016.4.11	2016.4.11	2016.4.12	2016.4.12	2016.4.13
Location	A	B	B	C	C	D	E



Current progress: Overcome the Time Resolution Issue in Trails

- Four algorithms
 - Random: Randomly pick the location order
 - Greedy: At each step, select next location with maximum transition probability.
 - Exact algorithm: enumerate all the possible routes.
 - Ant Colony System(ACS): an approximate algorithm designed for ATSP
- Partition trails when time intervals are larger than a threshold.

Time	Apr. 1	Apr. 1	Apr. 2	Apr. 3	June 3	June 4	June 5	June 5
Trail 1	BEGIN	L1	L2	L3	L2	L1	L2	END

Time	Apr. 1	Apr. 1	Apr. 2	Apr. 3	Apr.3	June 3	June 3	June 4	June 5	June 6
Trail 1	BEGIN	L1	L2	L3	END	BEGIN	L2	L1	L2	END



Current progress: Overcome the Time Resolution Issue in Trails

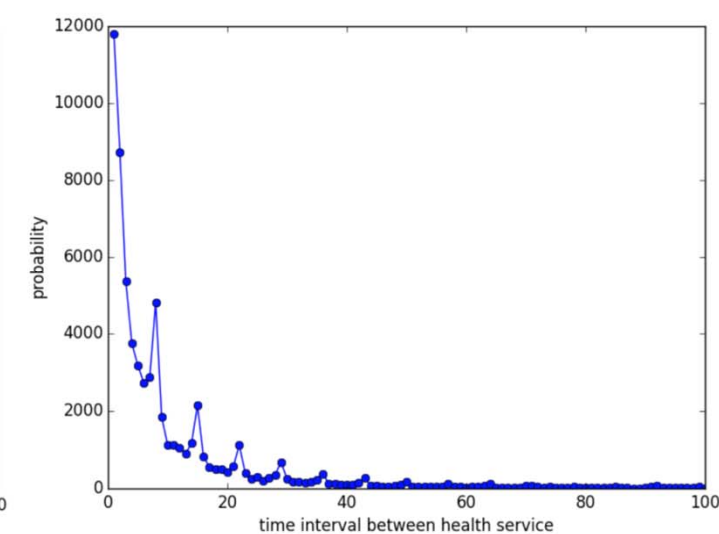
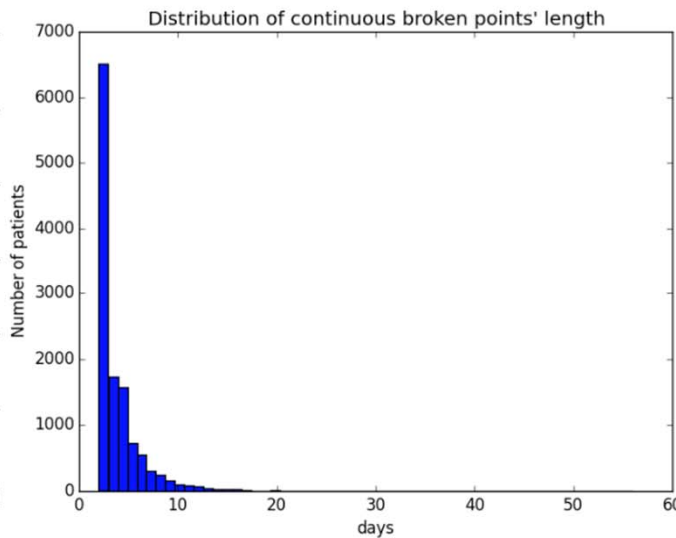
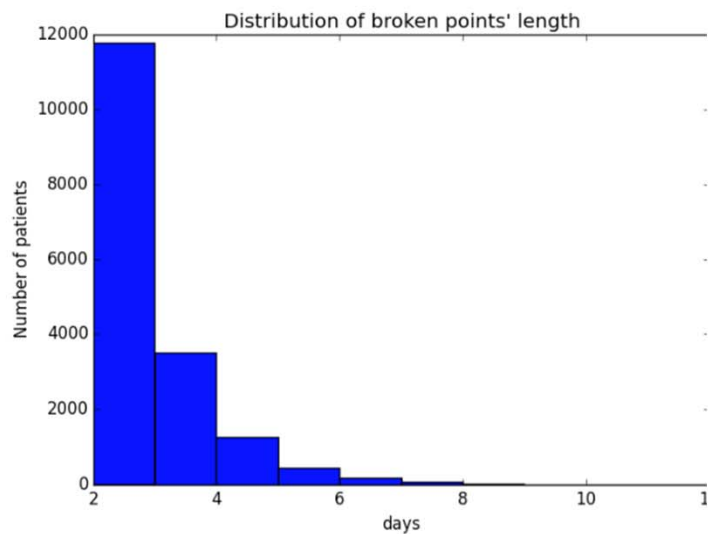
- Datasets:
 - Health record data
 - Location: health service
 - Agent: patient
 - Record: (patient, health service, date)

	# of Records	# of Agents	# of Locations
Health data	94885	5055	115

Use 814 unbroken trails as testing data

Current progress: Overcome the Time Resolution Issue in Trails

- Statistics of health dataset



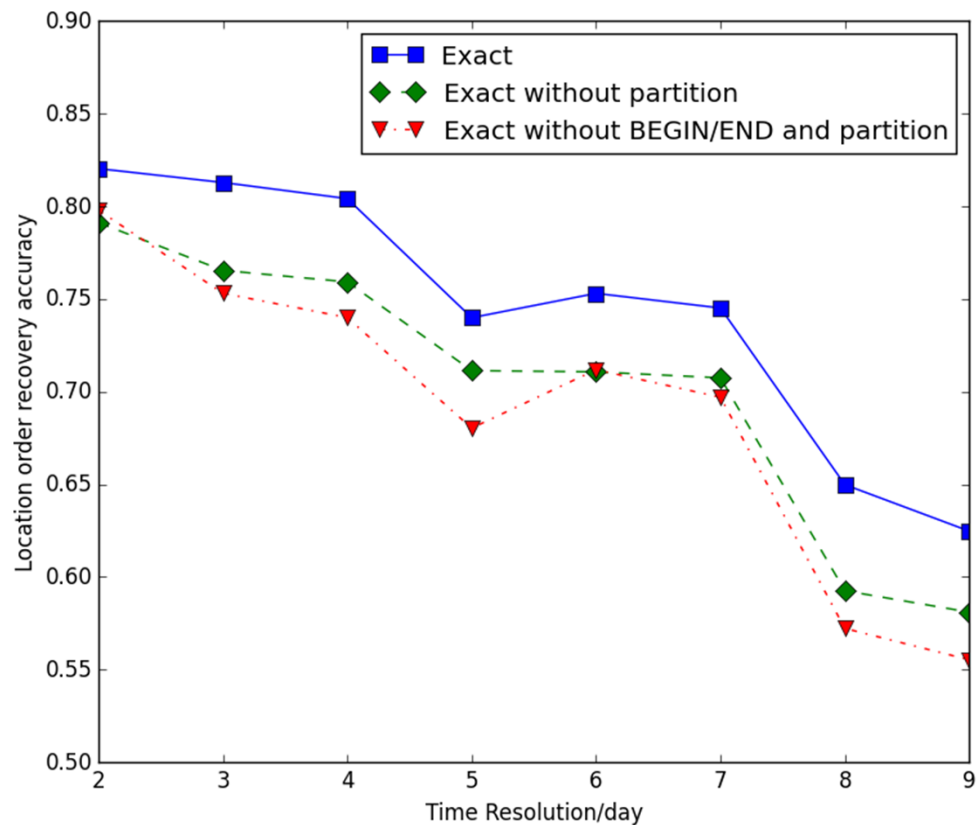
Current progress: Overcome the Time Resolution Issue in Trails

- Experiments setup
 - Artificially change the time resolution of 814 unbroken health trails
 - $timestamp' = \lfloor \frac{timestamp}{resolution} \rfloor * resolution.$

Time resolution(day)	2	3	4	5	6	7	8	9
# of broken trails	164	213	250	263	268	278	294	297
# of broken points	178	251	296	319	340	353	374	389
Avg. length of broken points	2.073	2.223	2.284	2.426	2.482	2.586	2.591	2.627
Avg. length of cont. broken points	2.121	2.364	2.449	2.650	2.749	2.916	2.909	3.041
Max. length of broken points	4	5	6	5	6	6	6	7

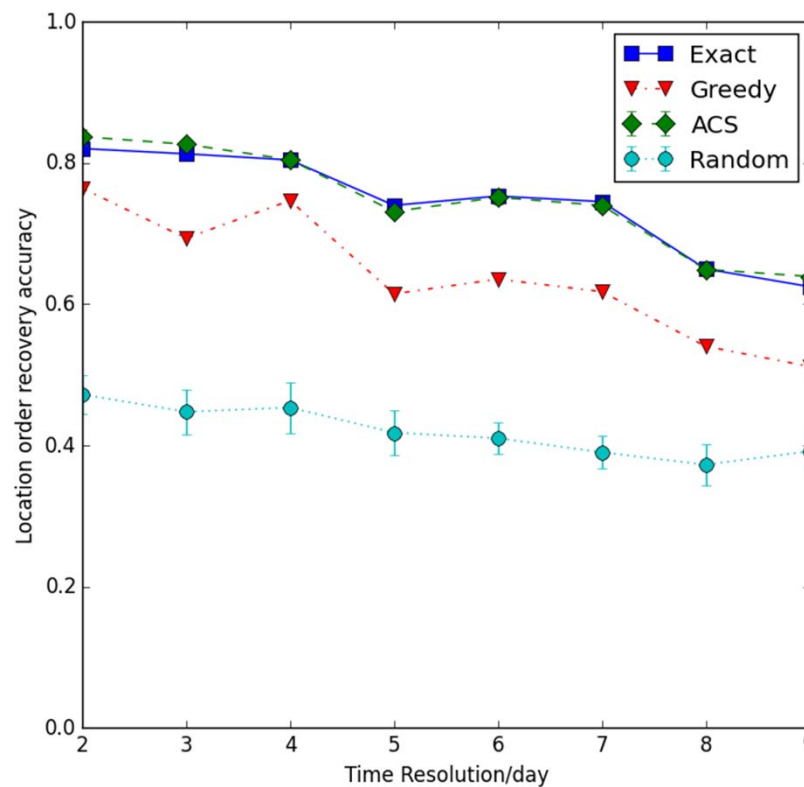
Current progress: Overcome the Time Resolution Issue in Trails

- The effects of BEGIN/END nodes and partitions



Current progress: Overcome the Time Resolution Issue in Trails

Comparison between four algorithms



Hands on

- Data preparation: program function call data

id	timestamp	location
0	553987065457672.00	class edu.umd.cs.findbugs.PluginLoader
0	553987065574331.00	class edu.umd.cs.findbugs.PluginLoader
0	553987065768508.00	class java.lang.Class
0	553987065819048.00	class edu.umd.cs.findbugs.PluginLoader
0	553987100679470.00	class java.net.URL
0	553987102005655.00	class edu.umd.cs.findbugs.PluginLoader
0	553987102202112.00	class edu.umd.cs.findbugs.PluginLoader
0	553987102260252.00	class edu.umd.cs.findbugs.PluginLoader
0	553987102331691.00	class edu.umd.cs.findbugs.PluginLoader
0	553987102384890.00	class edu.umd.cs.findbugs.PluginLoader
0	553987102425550.00	class edu.umd.cs.findbugs.PluginLoader
0	553987476519118.00	class edu.umd.cs.findbugs.PluginLoader
0	553987476619437.00	class java.net.URL
0	553987476664276.00	class edu.umd.cs.findbugs.PluginLoader
0	553987476741035.00	class java.lang.String
0	553987476797655.00	class edu.umd.cs.findbugs.PluginLoader



- Data import

Import Data into ORA-NetScenes
✕

What would you like to do?

- ✦ Design a meta-network
- [-] Import Excel or text delimited files
 - ✦ Rectangle of link values (a matrix)
 - ✦ Table of network links
 - ✦ Table of node attributes
 - ✦ Advanced table
 - ✦ Ego network transition tables
- [+] Import from another network analysis tool
- [+] Import from another tool
- [-] Import XML network data
 - ✦ DyNetML
 - ✦ GraphML
- [+] Import other data formats
 - ✦ Import Email
 - ✦ Import from a database

Description

Import single-mode table data (.csv or tab delimited) with a single ego node column and one or more path columns. Transition links are created for an ego based on the change of values in each path column from one time period to the next.

Sample

Ego Name	City
Adam	Pittsburgh
Adam	Seattle
Adam	Boston
Bob	Boston
Bob	New York
Bob	Pittsburgh
Carl	Phoenix

Cancel
< Back
Next >
Finish

- Data import

Import Data into ORA-NetScenes

Step 1: Select an ego-network file:

The file must have an Ego node column, and one or more Path node columns. Each path column will produce a Path x Path transition network where link values record the number of times an ego transitioned from one path node to another.

Use only lines where column has value:

Step 2: Select how the file is ordered:

Rows are sorted first by ego name and then by date

Sort rows by column which has

Date processing options:

Aggregate dates by

Window for transitions



- Data import

Import Data into ORA-NetScenes

Step 1: Select an ego column and optional entry/exit state columns:

Ego column: Create Entry nodes:

Class: One entry node: Create Exit nodes:

Name: Column values: One exit node: Column values:

Create new ego nodes during import

Step 2: Select one or more path columns:

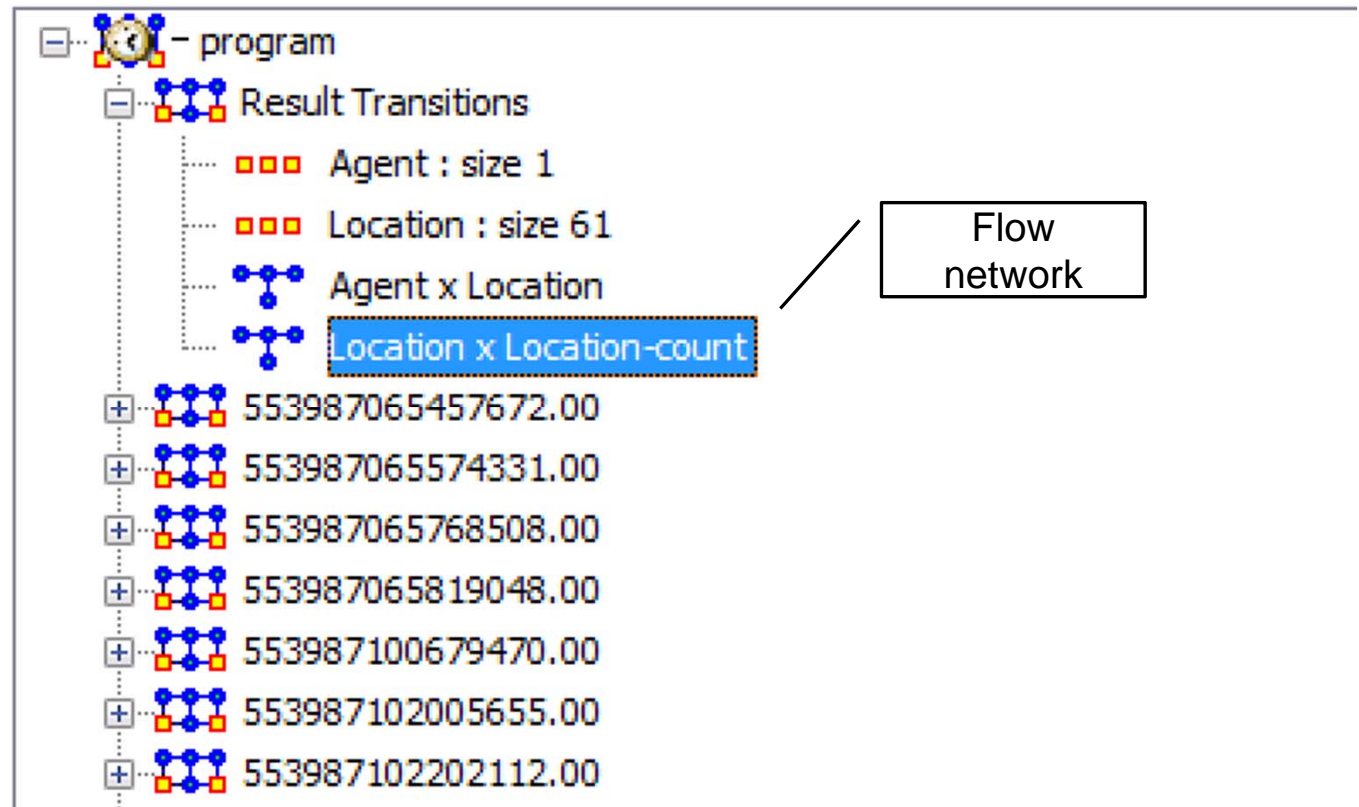
<input type="checkbox"/> id column:	<input type="checkbox"/> timestamp column:	<input checked="" type="checkbox"/> location column:
Class: <input type="text" value="<Select...>"/>	Class: <input type="text" value="<Select...>"/>	Class: <input type="text" value="Location"/>
Name: <input type="text"/>	Name: <input type="text"/>	Name: <input type="text" value="Location"/>
Transition network name: <input type="text"/>	Transition network name: <input type="text"/>	Transition network name: <input type="text" value="Location x Location"/>
Ego path network name: <input type="text"/>	Ego path network name: <input type="text"/>	Ego path network name: <input type="text" value="Agent x Location"/>

Create new path nodes during import

Cancel < Back Next > Finish

Hands on

- Data transform



Hands on

- Data transform

The screenshot shows a software window titled "Network: Location x Location-count". It has two tabs: "Info" and "Editor". The "Editor" tab is active, and a menu is open with the following options:

- Convert Links
- Remove Links
- Highlight
- Hide
- Sort Rows
- Sort Columns

The "Convert Links" sub-menu is expanded, showing the following options:

- Symmetrize by method
- Binarize** link values ($x \neq 0 \Rightarrow x = 1$)
- Collapse** link values ($a \leq x \leq b \Rightarrow x = 1$)
- Negate** link values ($-x$)
- Invert** the link values ($1/x$)
- Logarithm** of the link values ($\log_{10}(x)$)
- Absolute value** of the link values ($|x|$)
- Scale** the link values ($c * x$)
- Row Sum Normalize** the link values ($x_{ij}/(\text{sum of row } i)$)
- Column Sum Normalize** the link values ($x_{ij}/(\text{sum of column } j)$)
- Sum Normalize** the link values ($x_{ij}/(\text{sum of all values})$)
- Increment** the link values ($c + x$)
- Subtract** link values ($c - x$)
- Remove self-loops (diagonal)

Hands on

- Data transform

The screenshot shows a software window titled "Network: Location x Location-count". It has two tabs: "Info" and "Editor". The "Editor" tab is active, and a menu is open under the "Convert Links" button. The menu items are:

- Symmetrize by method
- Binarize link values ($x \neq 0 \Rightarrow x = 1$)
- Collapse link values ($a \leq x \leq b \Rightarrow x = 1$)
- Negate link values ($-x$)
- Invert the link values ($1/x$)**
- Logarithm of the link values ($\log_{10}(x)$)
- Absolute value of the link values ($|x|$)
- Scale the link values ($c * x$)
- Row Sum Normalize the link values ($x_{ij}/(\text{sum of row } i)$)
- Column Sum Normalize the link values ($x_{ij}/(\text{sum of column } j)$)
- Sum Normalize the link values ($x_{ij}/(\text{sum of all values})$)
- Increment the link values ($c + x$)
- Subtract link values ($c - x$)
- Remove self-loops (diagonal)

Hands on

- Analysis

Keyframe: Result Transitions

Meta-Network Name: Result Transitions

Meta-Network Time:

Filename:

Generate Reports... Visualize Measure Charts...

General statistics: _____