ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ
Мэдээлэл холбооны технологийн сургууль

ЛАБ 12
ТАЙЛАН

Шалгасан багш: Б.ТУЯАЦЭЦЭГ /F.SW03/

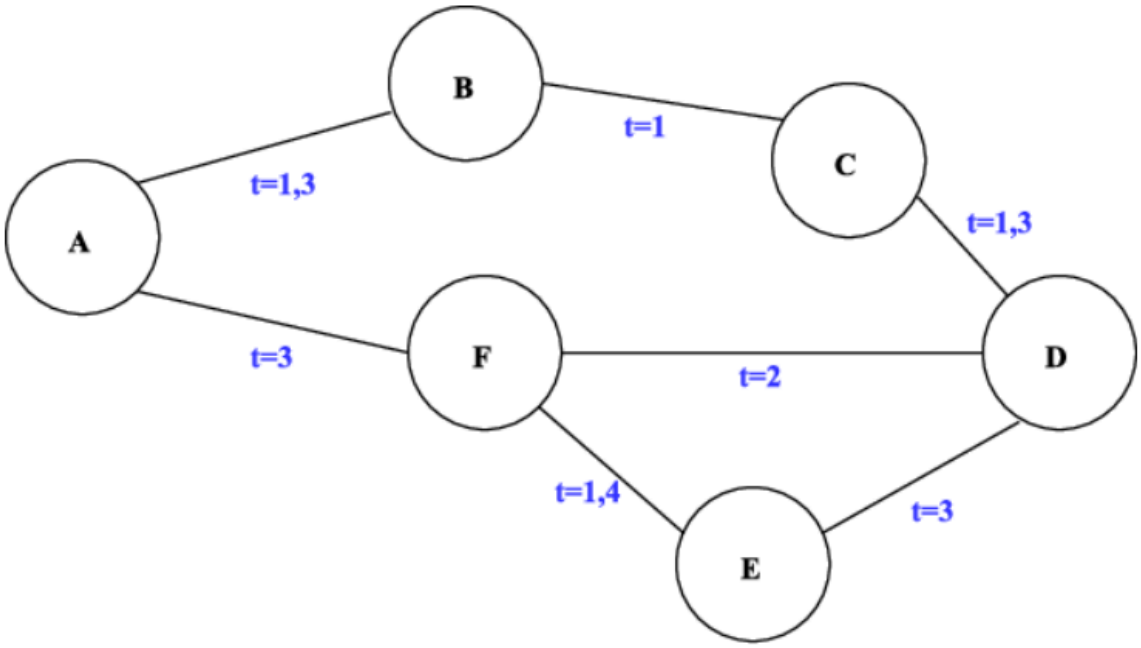
Гүйцэтгэсэн: B221910040 Н.Жавхлантөгс

Улаанбаатар 2025

Динамик Сүлжээ ба DyNetX ашиглах нь

Ерөнхий ойлголт:
Динамик сүлжээ нь оролцогчид (зангилаа), тэдгээрийн хоорондын харилцаа (ирмэг) нь цаг хугацаанд өөрчлөгддөг сүлжээ юм. Энэ төрлийн сүлжээг судлахдаа DyNetX санг ашигладаг ба энэ нь NetworkX сангийн өргөтгөл бөгөөд Python хэл дээр ажиллана.

DyNetX: динамик сүлжээг загварчлахад зориулагдсан сан
Бид өнгөрсөн хичээлүүд дээр статик сүлжээний топологийг судалсан. Гэхдээ бодит амьдрал дээр сүлжээнд оролцогчид үүсэж, зарим тохиолдолд алга болж, заримдаа холбогдох буюу харилцаа үүсгэж, зарим тохиолдолд харилцан хамаарал нь үгүй болж байдаг. Өөрөөр хэлбэл цаг хугацаа өнгөрөхөд зангилаа болон холбоосууд нэмэгдэж, зарим тохиолдолд устдаг. Энэ нь сүлжээний бүтэц болон холбоост байдалд ихээр нөлөөлдөг. Тухайлбал, топологийн өөрчлөлт нь тархах үзэгдэлд ашиглагдана. DyNetx нь цаг хугацаанаас хамаарч хувьсан өөрчлөгддөг графуудыг загварлахад ашиглагддаг. Дараах хэсэгт бид DyNetx санг ашиглан динамик сүлжээнүүдийг байгуулах болон түүн дээр шинжилгээ хийхэд шаардлагатай чухал ойлголтуудыг тайлбарлана. Динамик сүлжээ нь таймстамп буюу цагийн үечлэлүүдэд ирмэг, оройнууд холбогдон үүсэх топологи юм. Жишээг дараах зурагнаас харна уу.



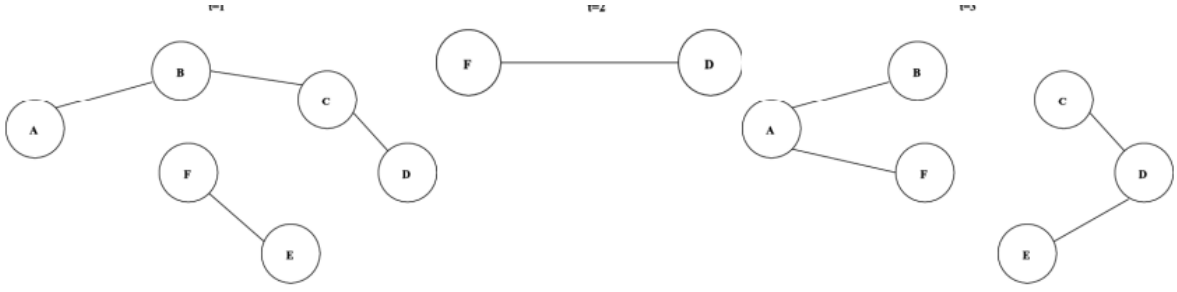
DyNetx нь a Python хэлний нэмэлт сан бөгөөд өмнө ашиглаж байсан networkx сангийн динамик сүлжээг загварчлах зориулж өргөтгөсөн хувилбар юм. Уг сан нь энэ төрлийн сүлжээнд ажиллах алгоритмуудыг агуулна.
судлаачид DyNetx -ийг NDlib -ийг дэмжих нэмэлт санг болгон хөгжүүлсэн. Энэ нь динамик сүлжээний топологийн генерик хөгжүүлэлтийг хийх боломжийг олгодог бөгөөд чиглэлт болон чиглэлт бус графууд дээр ажиллана.

- Snapshot Graphs
- Interaction Networks

Snapshot Graphs (to top)
Ер нь сүлжээний түүхэд тодорхой хугацаанд сүлжээг ажиглаад өөрчлөлтүүдийг нэгтгэн t хугацааны агшинд сүлжээний төлөв, шинжүүдийг агуулсан байдлаар нэг снапсот граф болгон хуваадаг. Томьёолбол,

A Snapshot Graph Gt is defined by a temporally ordered set ⟨ G1 ,G2 …Gt⟩ of static graphs where each snapshot Gi sets of nodes Vi and edges Ei . = (Vi,Ei) is univocally identified by the Снапсот граф Gt нь хугацааны хувьд эрэмбэлэгдсэн графуудын олонлог буюу ⟨ G1,G2 …Gt⟩ бөгөөд эдгээр нь статик графууд байх ба орой Vi болон ирмэгүүдийн Ei олонлогоос бүрдсэн граф Gi = (Vi,Ei) болно.

Сүлжээний агшин зуурын зургийг жишээ нь тогтмол давтамжтайгаар сүлжээний хямрал (бараг) үүсгэдэг үзэгдлийг загварчлахад үр дүнтэй ашиглаж болно. Энэ хувилбарт сүлжээний түүхийг дараалсан агшин агшинд хуваахад контекст хамааралтай түр зуурын цонхыг ашигладаг: сүлжээний амьдралын нарийн, статик, салангид байдлыг тодорхойлсон цаг хугацааны ажиглалтууд. Бидний динамик сүлжээний жишээг авч үзвэл бид дараах агшин зуурын графикуудыг тодорхойлж болно.



DyNetX суулгах:

```
!pip install dynetx
```

```
Collecting dynetx
  Downloading dynetx-0.3.2-py3-none-any.whl.metadata (2.9 kB)
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
(from dynetx) (2.0.2)
Requirement already satisfied: future in /usr/local/lib/python3.11/dist-packages
(from dynetx) (1.0.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages
(from dynetx) (4.67.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages
(from dynetx) (3.4.2)
Requirement already satisfied: decorator in
/usr/local/lib/python3.11/dist-packages (from dynetx) (4.4.2)
Downloading dynetx-0.3.2-py3-none-any.whl (39 kB)
Installing collected packages: dynetx
Successfully installed dynetx-0.3.2
```

Сүлжээ үүсгэх, дата унших:

```python
import dynetx as dn
import networkx as nx
import random

def read_net(filename):
  g = nx.Graph()
  with open(filename) as f:
    f.readline()
    for l in f:
      l = l.split(",")
      g.add_edge(l[0], l[1])
  return g

g = dn.DynGraph() # empty dynamic graph
```

Тайлбар:

- read_net() функц нь CSV файлын ирмэгийн мэдээллийг уншиж, граф үүсгэнэ.

Динамик сүлжээ үүсгэх:

```python
g = dn.DynGraph()  # хоосон динамик граф үүсгэх

# 8 снапшот үүсгэх
for t in range(1, 9):
    er = read_net(f'/content/sample_data/got-s{t}-edges.csv')
    g.add_interactions_from(er.edges, t=t)
```

Тайлбар:

- DynGraph нь динамик граф.

- Цагийн үечлэл бүр дээр графыг тэжээж, холбож байна (t=1,2,...,8).

```
g.temporal_snapshots_ids()
```
```
[1, 2, 3, 4, 5, 6, 7, 8]
```
Тайлбар:
- Динамик граф дахь бүх цагийн агшинуудын (snapshot) дугаарыг буцаана.

```
g1 = g.time_slice(1)
```
Тайлбар:
- 1-р снапшотын сүлжээг тусгаарлаж авч g1 хувьсагчид хадгална.

```
type(g1), g1.number_of_nodes(), g1.number_of_edges()
```
```
(dynetx.classes.dyngraph.DynGraph, 0, 0)
```

Тайлбар:
- g1-ийн төрлийг, оройн (node) тоог, ирмэгийн (edge) тоог харуулна.

```
g0_3 = g.time_slice(0, 3)
```

Тайлбар:
- 0–3-р цагийн агшнуудыг хамарсан графыг g0_3 гэж авна.

```
interactions_per_snapshot = {}
for t in g0_3.temporal_snapshots_ids():
    interactions_per_snapshot[t] = len(g0_3.interactions(t=t))
```

Тайлбар:

1. interactions_per_snapshot = {}
   Энэ мөр нь хоосон dictionaries үүсгэдэг. Энэ dictionary-д (temporal snapshot) хийгдсэн харилцаануудын тоо хадгалагдах болно.

2. for t in g0_3.temporal_snapshots_ids():
   Энэ мөр нь графын g0_3-ийн temporal_snapshots_ids функцээс гарч буй бүх цаг хугацааны үеүүдийг (temporal snapshots) гүйлгэж явдаг. t хувьсагч нь тухайн цаг хугацааны үеийг илэрхийлнэ.

3. interactions_per_snapshot[t] = len(g0_3.interactions(t=t))
   Энэ мөр нь тухайн цаг хугацааны үеийн (t) бүх харилцаануудын тоог тоолж, тэр тоог interactions_per_snapshot dictionary-д хадгалдаг. g0_3.interactions(t=t) нь тухайн цаг хугацааны үеийн бүх харилцаануудын жагсаалтыг буцаадаг бөгөөд len() функц нь уг жагсаалтын уртыг тооцоолдог.

```
g1_flat = nx.Graph(g1.edges())
```

Тайлбар:

1. g1.edges()
   Энэ функц нь граф g1-ийн бүх захын (edges) мэдээллийг буцаадаг. Зах нь хоёр дотуур обьектын хоорондох холбоосыг илэрхийлдэг бөгөөд энэ функц нь тухайн графын бүх захын жагсаалтыг буцаана.

2. nx.Graph(g1.edges())
   Энэ мөр нь NetworkX сангийн Graph классыг ашиглан шинэ граф g1_flat үүсгэдэг. Графын бүх захыг g1.edges() функцээс авсан бөгөөд түүгээр шинэ Graph объект үүсгэж байна. Үүний үр дүнд, g1_flat нь g1 графын бүх захтай адил төстэй граф болж үүснэ.

```
import networkx as nx

type(g1_flat), g1_flat.number_of_nodes(), g1_flat.number_of_edges()
```

Тайлбар:

1. type(g1_flat)
   Энэ мөр нь g1_flat объектын төрлийг шалгадаг. Энэ нь NetworkX сангийн Graph класс эсвэл DiGraph (сумын чиглэлтэй граф) зэрэг төрлийн нэг болохыг харуулах болно. Өөрөөр хэлбэл, энэ нь g1_flat графын төрөл юу болохыг мэдэхэд тусалдаг.

2. g1_flat.number_of_nodes()
   Энэ нь g1_flat граф дээрх дотоод узелуудын тоог буцаана. Узел нь графын нэг элемент бөгөөд хоёр эсвэл түүнээс олон захтай холбогддог. Энэ функц нь граф дахь бүх узелуудын нийт тоог өгнө.

3. g1_flat.number_of_edges()
   Энэ нь g1_flat граф дээрх бүх захын тоог буцаадаг. Зах нь хоёр узелын хоорондох холбоосыг илэрхийлдэг бөгөөд энэ функц нь граф дахь бүх захын нийт тоог өгнө.

Динамик сүлжээний хэмжүүрүүд

Inter event time (Global)

Distribution of inter event time (e.g., how much time before a new interaction appears in the graph)

Хэчнээн хугацаанд тухайн графд шинэ холбоосууд үүссэн вэ? гэдгийг глобалаар буюу графын хэмжээнд авч үзвэл.

```
r = g.inter_event_time_distribution()

print(f"Number interactions: temporal distance\t{r}")
```

```
Number interactions: temporal distance  {0: 3307, 1: 8}
```

Тайлбар:

1. r = g.inter_event_time_distribution()
   Энэ мөр нь g графын inter_event_time_distribution() функцыг дууддаг. Энэ функц нь харилцан үйлдлүүдийн хоорондох цаг хугацааны интервал буюу temporal distance буюу цаг хугацааны хуваарилалтыг буцаадаг. Энэ нь граф дахь бүх харилцаануудаас хоорондох цаг хугацааны интервалуудыг (temporal distances) тооцоолдог.

2. print(f"Number interactions: temporal distance\t{r}")
   Энэ мөр нь r хувьсагчийг хэвлэдэг. f-string ашиглан хэвлэхэд, r хувьсагчийн утга нь inter_event_time_distribution() функцээс буцаасан үр дүн (цаг хугацааны интервалуудын тархалт) болно.
   Мөн "Number interactions: temporal distance" гэсэн текстийг хэвлэж, түүний дараа r-ийн утгыг харуулдаг.

Inter event time (Node)

Distribution of inter event time (e.g., how much time before a new interaction involving a specific node appears in the graph)

Хэчнээн удаа тухайн графын тодорхой нэг оройд холбогдсон холбоосууд үүссэн вэ?

```
r = g.inter_event_time_distribution("ARYA")

print(f"Number interactions: temporal distance\t{r}")
```

```
Number interactions: temporal distance  {0: 137, 1: 8}
```

Тайлбар:

1. r = g.inter_event_time_distribution("ARYA")
   Энэ мөр нь g графын inter_event_time_distribution() функцийг ашиглан "ARYA" гэдэг тодорхой узелийн хувьд цаг хугацааны интервалын тархалтыг тооцоолдог. "ARYA" гэдэг нь граф дээрх узелийн нэр бөгөөд уг узелд хийгдсэн харилцан

үйлдлүүдийн хоорондох цаг хугацааны интервалуудыг гаргана.

2. print(f"Number interactions: temporal distance\t{r}")
   Энэ мөр нь r хувьсагчийг хэвлэдэг. f-string ашиглан хэвлэхэд, r хувьсагчийн утга нь "ARYA" узелд хийгдсэн харилцан үйлдлүүдийн хоорондох цаг хугацааны интервалын тархалтыг илэрхийлэх болно. Тодруулбал, энэ нь "ARYA" узелд хийгдсэн бүх харилцан үйлдлүүдийн цаг хугацааны интервалуудыг үзүүлнэ.

Inter event time (Edge)

Distribution of inter event time (e.g., how much time before a new interaction among two nodes, u and v, appears in the graph)

Хэчнээн удаа u болон v оройнуудыг холбосон ирмэг үүссэн вэ?

```
u = 'JON'

v = 'ARYA'
```

```
r = g.inter_event_time_distribution(u, v)

print(f"Number interactions: temporal distance\t{r}")
```

Тайлбар:

- Энэхүү код нь g граф дээрх хоёр тодорхой (node) болох 'JON' болон 'ARYA'-ийн хоорондох харилцан үйлдлүүдийн цаг хугацааны интервал буюу temporal distance-ийг тооцоолж, үр дүнг хэвлэдэг.

Degree - Оройн зэрэг

Degrees can be queried time-wise Тодорхой эгшинд оройн зэрэг хэд вэ?

```
g.degree(t=2)['ARYA'] # degree of node 0 at time t=2

27
```

Тайлбар:

- Энэхүү код нь g графын "ARYA" node - ийн хугацааны t=2-т байх degree буюу холбогдолын (нэгжүүдтэй холбогдсон захын тоо)-ыг тооцоолж гаргадаг.

Coverage

The ratio of existing nodes w.r.t. the possible ones.

```
g.coverage()
```

```
0.2977216748768473
```

Node contribution

Node u coverage of the temporal graph.

```
g.node_contribution("BERIC")
```

```
0.625
```

Тайлбар:

- g.node_contribution("BERIC") код нь "BERIC" гэдэг node граф доторх хувь нэмрийг тооцоолдог.

Edge contribution

Edge (u, v) coverage of the temporal graph.

```
g.edge_contribution(u, v)
```

```
0.375
```

Тайлбар:

- g.edge_contribution(u, v) код нь u болон v node хоорондох захын хувь нэмрийг тооцоолдог.

Node pair uniformity

Overlap between the presence times of u and v.

```
g.node_pair_uniformity(u, v)
```

```
1.0
```

Тайлбар:

g.node_pair_uniformity(u, v) нь u болон v узелүүдийн хоорондын харилцан үйлдлүүдийн цаг хугацааны жигд байдлыг хэмждэг. Хэрвээ тэдгээрийн хоорондын үйлдлүүд тогтмол хугацаанд болсон бол утга нь өндөр, харин бөөгнөрсөн эсвэл жигд бус бол утга нь бага гарна.

Density

Temporal network density: fraction of possible interactions that do exist in the temporal network

```
g.density()
```

```
0.06686633244351846
```

Тайлбар:
g.density() нь графын төлөөлсөн харилцаанууд хэр нягт байгаа буюу нийт боломжит харилцаан дотроос хэд нь бий вэ гэдгийг тооцоолдог.
Өөрөөр хэлбэл, энэ нь графын зах (edge)-ийн нягтшил буюу орон зайг хэр сайн ашиглаж байгааг харуулна. Утга нь 0–1-ийн хооронд гарна:

- 1 гэвэл бүх боломжит узелүүд хоорондоо холбогдсон (нойтон граф),
- 0 гэвэл ямар ч холболт байхгүй.

Node Density
Intersection among the temporal presence of the edge (u, v) and the joint temporal presences of u and v

```
 g.node_density(u)
```

```
0.2295760082730093
```

Тайлбар:
g.node_density(u) нь u node - ийн ойр орчмын харилцааны нягтшлыг (density) тооцоолдог.
Өөрөөр хэлбэл, u node-той холбогдсон хөрш узелүүд хоорондоо хэр сайн холбогдсон байгааг хэмждэг. Утга нь 0–1-ийн хооронд гарна:

- 1 бол хөршүүд нь бүгд хоорондоо холбоотой,
- 0 бол огт холбогдоогүй.

Pair Density

Intersection among the temporal presence of the edge (u, v) and the joint temporal presences of u and v.

```
g.pair_density(u, v)
```

0.375

Тайлбар:

g.node_density(u) нь u node-ийн ойр орчмын харилцааны нягтшлыг (density) тооцоолдог. Өөрөөр хэлбэл, u node-тэй холбогдсон хөрш node-үүд хоорондоо хэр сайн холбогдсон байгааг хэмждэг. Утга нь 0–1-ийн хооронд гарна:

- 1 бол хөршүүд нь бүгд хоорондоо холбоотой,
- 0 бол огт холбогдоогүй.

Snapshot Density

Density of a temporal network at time t.

```
for t in g.temporal_snapshots_ids():

  print(f"{t}\t{g.snapshot_density(t)}")
```

```
1	0.06971428571428571

2	0.05886627906976744

3	0.06608969315499606

4	0.04535563715490276

5	0.05640222190571144

6	0.05404055538907202

7	0.1271604938271605

8	0.20473898556090336
```

Энэ код нь графын түүхэн агшин бүр дэх нягтшлыг (density) хэвлэдэг.

- g.temporal_snapshots_ids() – графын бүх цаг хугацааны агшнуудыг авна.
- g.snapshot_density(t) – тухайн агшин дахь графын нягтшлыг тооцоолно.
- print(...) – цаг хугацаа ба тэр үед ямар нягт граф байсан мэдээллийг хэвлэнэ.

Path analysis

Computes the time respecting paths among u and v within [start, stop]

```
import dynetx.algorithms as al

paths = al.time_respecting_paths(g, "GENDRY", "GREY_WORM", start=1, end=5)
```

Энэ код нь "GENDRY" node-оос "GREY_WORM" node хүртэлх цаг хугацааг хүндэтгэдэг замуудыг (time-respecting paths) 1-ээс 5 хүртэлх хугацаанд хайж гаргаж ирнэ.
Тайлбар:

- al.time_respecting_paths(...) — динамик граф доторх цагийн дарааллыг баримталсан замуудыг олдог функц.
- g — граф объект
- "GENDRY" — эхлэх node
- "GREY_WORM" — төгсгөл node
- start=1, end=5 — зөвхөн 1–5 хугацааны хоорондох замуудыг хайна

```
p = paths[0] # example of identified paths. Each list element is a tuple
of the form (fr

p
```

Энэ код нь paths жагсаалтаас эхний замыг (paths[0]) авч p хувьсагчид хадгалж байна.

- paths нь time_respecting_paths() функцээр олдсон цаг хугацааг хүндэтгэдэг замуудын жагсаалт юм.
- Зам бүр нь тупль хэлбэртэй ((from, to, time)) элементийн дараалалтай байна.

***Даалгавар. ***

dynetx санг судална уу. Тодорхой нэг статик снапшотын хувьд эсвэл глобалаар буюу динамик сүлжээний хэмжээнд shortest, fastest, foremost, fastest shortest, shortest fastest замуудыг ол. Эдгээр замуудын ялгааг тайлбарла.

Замуудын ялгаа

Замын төрөл	Тайлбар
Shortest path	Хамгийн бага ирмэг бүхий зам (үндсэн граф дээр)
Fastest path	Хамгийн бага хугацаанд очдог зам (цаг хугацаанд суурилсан)
Foremost path	Хамгийн анхны очих боломжит зам
Fastest shortest	Хамгийн бага ирмэгтэй, тэдгээрийг хамгийн хурдан дамждаг зам
Shortest fastest	Хамгийн хурдан зам дотроос хамгийн богино ирмэгтэйг сонгоно