B221910009 Д.Нямдорж

Лабораторын ажил 10: Network Dynamics

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.

```
!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-pack
Requirement already satisfied: future in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: decorator in /usr/local/lib/python3.11/dist-
Downloading dynetx-0.3.2-py3-none-any.whl (39 kB)
Installing collected packages: dynetx
Successfully installed dynetx-0.3.2
```

```
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 line in f:
      i = line.split(",")
    g.add_edge(line[0], line[1])
  return g
g = dn.DynGraph()
```

Динамик сүлжээг үүсгэх

Нэг динамик сүлжээг ирмэгүүдийг нэмэх замааар үүсгэлээ. Жишээлбэл, 10 ER графуудыг үүсгэсэн ба энэ нь ижилхэн динамик системийн ялгаатай ялгаатай топологиудыг илэрхийлнэ.

```
for t in range(1,9):
  er = read_net(f'/content/got-s1-edges.csv')
  g.add_interactions_from(er.edges, t=t)
```

Бид снапшотын ID дугааруудыг дараах байдлаар авч чадна

```
g.temporal_snapshots_ids()
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

Цаашлаад бид снапшот бүрийн ID-г нь ашиглан тухайн снапшотод хандах боломжтой

```
g1 = g.time_slice(1)
g1.number_of_nodes()
g1.number_of_edges()
```

```
1
```

Хэрэв хуваагдсан нэг снапшотыг шинжлэхээр бол networkx эсвэл dynetx - ийн функцүүдийг ашиглах боломжтой гэсэн үг.

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

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  {8: 1}
```

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

```
    Number interactions: temporal distance  {}
```

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?

```
print(list(g.edges()))
```

```
    [('T', 'Y')]
```

```
u = 'T'
v = 'Y'
r = g.inter_event_time_distribution(u, v)
print(f"Number interactions: temporal distance\t{r}")
```

```
    Number interactions: temporal distance  {7: 1}
```

Degree - Оройн зэрэг

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

```
g.degree(t=3)['T'] # degree of node 0 at time t=2
```

```
    1
```

Coverage

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


```
g.coverage()
```

```
    1.0
```

Node contribution

Node u coverage of the temporal graph

```
g.node_contribution("Y")
```

```
1.0
```

Edge contribution

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

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

```
1.0
```

Node pair uniformity

Overlap between the presence times of u and v

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

```
1.0
```

Density

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

```
g.density()
```

```
1.0
```

Node Density

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

```
g.node_density(v)
```

```
0.5
```

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)
```

```
    1.0
```

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       1.0
    2       1.0
    3       1.0
    4       1.0
    5       1.0
    6       1.0
    7       1.0
    8       1.0
```

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.

dynetx (Dynamic NetworkX) нь динамик граф/сүлжээ (өөрчлөгдөж буй орой болон ирмэгтэй граф)-ийг удирдаж, тэдгээр дээр анализ хийхэд зориулагдсан Python сан юм. Энэ нь networkx дээр суурилсан бөгөөд динамик сүлжээний хувьд уламжлалт граф алгоритмуудыг хэрэгжүүлэх боломж олгодог.

Төрөл	Тайлбар
Shortest Path	Оройнуудын хамгийн бага тоотой зам (edge count)
Fastest Path	Цаг хугацааны хувьд хамгийн хурдан хүрэх зам (travel time хамгийн бага)
Foremost Path	Хамгийн эрт хүрч болох зам (earliest arrival time)
Shortest Fastest Path	Бүх fastest замуудаас хамгийн бага орой дамжсан зам
Fastest Shortest Path	Бүх shortest замуудаас хамгийн хурдан хугацаанд хүрдэг зам