Mongolian Journal of Engineering and Applied Sciences, 2022, 4–1

Компьютерын Ухаан

Хоорондын төвийн аргаар сүлжээг задлах алгоритм

П.Далайжаргал[1], Г.Гантулга[1,*], З.Идэр[1], Б.Доржнамжирмаа[1]

[1]МУИС, ХШУИС, Мэдээлэл, Компьютерын Ухааны Тэнхим

Received on 2022.4.19; Revised on 2022.11.06; Accepted on 2022.11.29

*Corresponding author: gantulgag@seas.num.mn

Хураангуй

Энэ ажлаар хоорондын төвийн аргаар өндөр эрэмбэ оногдох оройн дэд олонлогийг илрүүлэх дөрвөн шинэ алгоритмыг тодорхойлон, тэдгээрийн үр дүнг харьцуулан шинжилнэ. Оройн дэд олонлогийг устгасны дараа үлдэх сүлжээний бүтэц дэх өөрчлөлтөөр тухайн алгоритмын илрүүлсэн дэд олонлогийн нөлөөг үнэлнэ. Хоорондын төвийн аргаар өндөр эрэмбэ оногдсон к оройг нэгэн зэрэг устгах алгоритмыг сүлжээг задлахад үр дүнтэйгээр ашиглах боломжтойг туршилтын үр дүн харуулав.

Түлхүүр үг: комплекс сүлжээ, граф, сүлжээг задлах, хоорондын төвийн арга

1 Удиртгал

Сүлжээ (граф)-ний зарим орой бусад оройноос чухал үүрэгтэй байдаг. Тухайлбал, агаарын тээврийн сүлжээнд зарим нисэх буудал (жишээ нь Станбул) онцгой чухал нөлөөтэй. Цөөн тооны оройн дэд олонлог устсаны (эсвэл ажиллагаа доголдох) улмаас сүлжээ үйл ажиллагаагаа хэвийн үргэлжлүүлэх боломжгүй болох тохиолдол бий [1]. Иймд сүлжээний бүтцийг судалж халдлагад хэр зэрэг тэсвэртэйг шинжлэн дүгнэх нь практик ач холбогдол өндөртэй бөгөөд эдгээр нь сүлжээ задлах бодлогын гол хэрэглээ юм.

Сүлжээг задлах бодлогын зорилго нь графаас чухал оройнуудыг олж устган, графын жижиглэхэд оршино. Сүлжээг задлах бодлого нь хэд хэдэн хэлбэртэй [2–4] бөгөөд бидний уг ажилд сонирхсон хэлбэр нь $G(V, E)$ граф өгөгдөхөд аль болох цөөн орой устгаж, графын хамгийн том бүрдэл хэсгийн хэмжээг өгөгдсөн L параметрээс бага болгох юм.

Хоорондын төв (betweenness centrality) нь сүлжээний агент (орой)-ууд хооронд мэдээлэл дамжуулах үйл ажиллагаа дахь оролцоог үнэлэн агент бүрд эрэмбэ (оноо) өгөх арга юм [5]. Хоорондын төв (ХТ)-ийн оноо нь өндөр бол тэр агентыг сүлжээний гол төв гэж үзнэ. Энэ оноог тооцоолдог олон аргууд байдаг [6–8]. Сүүлийн жилүүдэд ХТ-ийн оноо өндөртэй топ к оройг илрүүлэх судалгаа эрчимжиж байна [9–12]. Гэхдээ эдгээр топ к орой сүлжээний бүтэц дэх нөлөөллийн судалгаа дутмаг байна.

Иймд энэ ажлаар бид топ k оройны сүлжээний бүтэц дэх нөлөөллийг судлахын тулд сүжлээг хамгийн өндөр хоорондын төвтэй к ширхэг оройг нэг дор устгах замаар сүлжээг задлах арга болон хамгийн өндөр хоорондын төвтэй нэг оройг устгах замаар сүлжээг задлах арга дээр тулгуурласан 4 алгоритм боловсруулсан (2-р хэсэг). Эдгээр 4 алгоритмыг олон нийтэд нээлттэй 4 бодит сүлжээ дээр ажиллуулж гүйцэтгэл, үр дүнг харьцуулсан (3-р хэсэг). Хоорондын төвийг тооцоолох нь хүндрэлийн хувьд $O(nm)$ [6] тул өндөр хоорондын төвийн утгатай к оройг нэг дор устгах нь хугацааны хувьд нэг нэгээр устгаснаас хамаагүй ашигтай бөгөөд шийдийн чанарын хувьд ч сайн болохыг уг судалгаа харууллаа (4-р хэсэг).

2 Арга зүй

Сүлжээний тасрах оройг илрүүлэхэд хялбар (бүх тасрах орой (articulation point)-г илрүүлэх шугаман алгоритм бий [13]) бөгөөд тасрах орой олонтой сүлжээг задлах нь хүндрэл багатай байдаг. Хоорондын төв (ХТ) –ийн аргын сүлжээг задлах нөлөөг тодруулах зорилгоор сүлжээнээс бүх тасрах оройнуудыг устгая. Тасрах оройг устгасны дараа үлдэх сүлжээнд шинээр тасрах орой үүсвэл тэдгээрийн мөн устгая. Тасрах оройгүй болтол энэ үйлдлийн давтах аргаар сүлжээний цөм (core)-ийг үүсгэе [14]. Сүлжээний тасрах оройг устгах замаар үүсгэсэн цөм нь давхар холбоост бүрдэл (ХБ) (bi-connected component) байна. Энэ судалгаанд цөмийг ХТ-ийн аргаар задалж хэмжээг хоёр дахин буруулахад сонгогдох оройн дэд олонлогийг судална.

Харьцуулах дөрвөн алгоритмыг доор тодорхойлов. “Устгах” болон “буцаан нэмэх” гэсэн хоёр үйлдлийн ялгаатай комбинацаар дөрвөн алгоритм үүсгэе. Алгоритм 1 нь зөвхөн устгах үйлдэлтэй. Хамгийн өндөр ХТ-н оноотой нэг оройг устгана. Оройг устгаад ХТ оноог дахин бодно. Цөмийн хэмжээ хоёр дахин бага болох хүртэл устгалыг үргэлжлүүлнэ.

Алгоритм 2 нь мөн зөвхөн устгах үйлдэлтэй. Алгоритм 1-ээс ялгаа нь к оройг нэг дор устгана. к оройг устгаад ХТ оноог дахин бодно. Цөмийн хэм-

жээ хоёр дахин бага болох хүртэл устгалыг үргэлжлүүлнэ.

Алгоритм 3 нь "устгах" болон "буцаан нэмэх" үйлдлүүдийг хослон хэрэглэнэ. ХТ-ийн оноонд үндэслэн сүлжээний бүтцэд чухал нөлөөтэй к оройг илрүүлнэ. Эдгээр к оройноос нөлөөгөөр эрэмбэлэн k/2 оройг буцаан сүлжээнд нэмнэ. Үр дүнд к/2 оройг устгана. Мөн цөмийн хэмжээ хоёр дахин бага болох хүртэл устгалыг үргэлжлүүлнэ. Буцаан нэмэх оройг тодорхойлохдоо оройг эрэмбэлэх шаардлагатай. Нэмэгдэж буй оройнуудыг холбон үүсгэсэн бүрдэл (component)-ийн хэмжээнд үндэслэн эрэмбэлнэ. Үүсэх компонент хэмжээ аль бага байх оройноос эхлэн буцаан нэмнэ.

Алгоритм 4 нь Алгоритм 3-тай адил устгах болон буцаан нэмэх үйлдлийг хослон хэрэглэнэ. Гол ялгаа нь өөр схемээр буцаан нэмэх юм. Алгоритм 3-д алхам бүрд k/2 орой буцаан нэмдэг. Харин Алгоритм 4-д хоёр алхам дараалан к оройг устгаад, эдгээр 2к оройг өмнөх устсан оройн олонлогтой нэгтгэнэ. Энэ нийт устсан оройн олонлогоос к оройг буцаан нэмнэ. Мөн адил цөмийн хэмжээ хоёр дахин бага болох хүртэл устгалыг үргэлжлүүлнэ.

```
Algorithm 1 (G)
1 n = number-of-nodes(G)
2 S = ∅
3 while (LLC > n/2)
4         [s, G] = extract-highest-BC-node(G)
5         S = S ∪ s
6 return S
```

```
Algorithm 2 (G, k)
1 n = number-of-nodes(G)
2 S = ∅
3 while (LLC > n/2)
4         [S', G] = extract-highest-BC-nodes(G, k)
5         S = S ∪ S'
6 return S
```

```
Algorithm 3 (G, k)
1 n = number-of-nodes(G)
2 S = ∅
3 while (LLC > n/2)
4         [S', G] = extract-highest-BC-nodes(G, k)
5         [S', G] = Re-insert(G, S', k/2)
6         S = S ∪ S'
7 return S
```

```
Algorithm 4 (G, k)
1 n = number-of-nodes(G)
2 S = ∅
3 step = 0
3 while (LLC > n/2)
4         step = step + 1
5         [S', G] = extract-highest-BC-nodes(G, k)
6         S = S ∪ S'
7         if step  mod 2 = 0
8                 [S, G] = Re-insert(G, S, k)
9 return S
```

3 Туршилт, үр дүн

Энэ бүлэгт өмнөх бүлэгт тодорхойлсон дөрвөн алгоритмаар устгагдах (илрүүлэх) оройн олонлогуудыг харьцуулан шинжилнэ.

3.1 Өгөгдөл, урьдчилсан боловсруулалт

Туршилтад ашигласан сүлжээний мэдээллийг Хүснэгт 1-д доор оруулав. Circuit нь технологийн сүлжээ. Энэ сүлжээний нь логик гайт (logic gate) тэдгээрийн холбоосоор үүснэ. EU нь европын холбооны онгоц буудлуудын хоорондох нислэгийг сүлжээ юм. 2014 оны 2-оос 8-р сард буудлуудын хооронд шууд нислэг бүртгэгдвэл тэдгээр буудлыг хооронд холбосон. Facebook сүлжээ нь Facebook дэх найзын сүлжээ. hepph нь эрдэмтдийн хамтын ажиллагааны сүлжээ. arXiv-н High Energy Physics-Phenomenology ангилалд хэвлэгдсэн өгүүллийн зохиогчид, тэдгээрийн хамтын ажиллагааг харуулах сүлжээ. Хэрэв өгүүлэл хэвлүүлж буй зохиогч v өөр нэг u эрдэмтэнтэй хамтарч бичсэн бол сүлжээний v, u орой хооронд холбоос татна. Нэг өгүүллийг n хүн хамтарч хийсэн бол эдгээр n хүн өөр хооронд холбогдоно [15].

Алгоритм 2, 3, 4-д ашиглагдах к утгыг log(n)-р сонгов, энд n нь цөмийн оройн тоо. Python 3 хэлний Networkit графын санг ашиглан хэрэгжүүлэв. Туршилтыг Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz компьютер ашиглан гүйцэтгэв.

Хүснэгт 1: Сүлжээний мэдээлэл. оройн тоо (V), ирмэгийн тоо (E), цөмийн оройн тоо (Core), цөмийн дундаж зэрэг (deg), цөмийг задлах хязгаар (L)

Сүлжээ	V	E	core	deg	L
circuit	252	399	172	3.3	86
EU	1,191	31,610	789	45.5	395
facebook	4,039	88,234	3,146	49.2	1,573
hepph	12,008	118,489	2,905	26.8	1,453

3.2 Туршилтын үр дүн

Алгоритм 1, 2, 3, 4-р илрүүлэх оройн олонлогуудыг харгалзан S1, S2, S3, S4 гэе. Эдгээр олонлогийн хэмжээг Хүснэгт 2-д харуулав. Эдгээр хэмжээ нь цөмийг задлан хэмжээг L хүртэл буруулахад шаардагдах оройн тоог зааж буй. Иймд цөөн орой устгах алгоритмыг бусдаас үр дүнтэй гэж үзнэ. Өмнөх харьцуулсан судалгаагаар [16] Алгоритм 1-г сүлжээг задлах бодлогод үр дүнтэй гэдгийг мэднэ. Ал-

горитм 3 нь Алгоритм 1-г гурван сүлжээний хувьд ялсан, нэг сүлжээнд адил үр дүнд хүрсэн байна. Алгоритм 4 нь Алгоритм 1-г хоёр сүлжээний хувьд ялсан, нэг сүлжээнд адил үр дүнд хүрсэн, нэг сүлжээнд ялагдсан байна. Харин Алгоритм 2 нь бусад бүх аргаас тааруу үр дүнг харуулжээ. Тэмдэглэж хэлэхэд Алгоритм 1, 2 нь "буцаан нэмэх" үйлдлийг ашигладаггүй, зөвхөн устгах үйлдэл ашигладгаараа бусад хоёр алгоритмаас ялгаатай. facebook сүлжээнээс Алгоритм 3-н устгах оройг Зураг 1-д дүрслэв.

Хүснэгт 2: Дөрвөн алгоритмаар устгасан оройн олонлогийн хэмжээ

Сүлжээ	|S1|	|S2|	|S3|	|S4|
circuit	5	7	5	5
EU	248	288	237	276
facebook	18	44	16	16
hepph	149	264	132	123

Одоо S1 олонлогийг S2, S3 болон S4 олонлогуудтай харьцуулан судалцгаая. Тэдгээр хос олонлогууд (S1, S2; S1, S3; S1, S4)-н нийтлэг элементийн тоог харьцуулан шинжилгээ хийе. Хүснэгт 3-д S1 олонлогтой бусад олонлогууд хэр зэрэг нийтлэг элементтэй байгаа талаарх мэдээллийг оруулав, энд "тоо" багана нь тухайн хоёр олонлогийн нийтлэг (ижил) элементийг тоог илэрхийлнэ, "хувь 1" багана нь S1-н хэдэн хувийг бусад олонлог агуулж буйг илэрхийлнэ (өөрөөр хэлбэл, хувь 1 = (100*тоо/|S1|)), "хувь 2" багана нь Si (i = 2, 3, 4)-н хэдэн хувийг S1 олонлог агуулж буйг илэрхийлнэ (өөрөөр хэлбэл, хувь 2 = (100*тоо/|Si|), i = 2, 3, 4) . Харьцуулж байгаа хос олонлогийн нийтлэг элементийн хувийг "хувь 1", "хувь 2"-н аль өндрөөр авч үзэх боломжтой.

Facebook сүлжээний хувьд олонлогуудын нийтлэг элементийн хувь өндөр байна. Тодруулбал, S2 олонлог S1 олонлогийг 100 хувь, харин S1 олонлог S3, S4 олонлогуудыг 100 хувь агуулж байна. EU сүлжээн дээр 80-с дээш хувиар өөр хоорондоо адил байна. Hepph сүлжээн дээр S3, S4-н хэмжээ S1 хэмжээнээс бага (Хүснэгт 2-г харна уу) байгаа тул "хувь 2" нь "хувь 1"-с өндөр гарсан байна. Энэ сүлжээнд нийтлэг элементийн хувь 64-с дээш хувьтай гарсан байна. Circuit сүлжээнд нийтлэг элементийн хувь нь бүх хосуудад 60 хувь гарчээ. Хэдий Алгоритм 2, 3, 4 нь нэгэн зэрэг к оройг устгадаг ч нэг нэгээр устгах арга (Алгоритм 1) –тай 60-с дээш хувийн нийтлэг элемент бүхий оройн олонлогийг илрүүлсэн байна. Хоорондын төвийн аргын үндсэн сул тал болох хурдны асуудлыг нэгэн зэрэг олон орой устгах замаар нөхвөрлөх боломжийг энэ туршилтын үр дүн харууллаа.

Алгоритмуудын илрүүлэн, устгаж буй орой, тэдгээрийн сүлжээний бүтэц дэх нөлөөллийн үйл явцыг судалцгаая. Зураг 1-д устсан оройн тоонд харгалзах хамгийн том холбоост бүрдэл (ХБ)-ийн хэмжээг харуулав, энд Алгоритм 3, 4-н эцсийн буцаан нэмэх үйлдэл хийгдээгүй. Бүх тест дээр Алгоритм 1 сүлжээг үр дүнтэйгээр задалж байна. Алгоритм 1-н график зарим сүлжээ (facebook, hepph) -д шаталсан хэлбэртэй дүрслэгдэж байгаа нь хамгийн том ХБ-г задалж чадалгүй хэд хэдэн дараалсан устгал хийснийг илтгэнэ. Алгоритм 4 нь Алгоритм 2, 3-с үр дүнтэй задалсан байна. Сануулж хэлэхэд Алгоритм 3, 4 эцсийн буцаан нэмэх үйлдлийн тусламжтай Алгоритм 1-н үр дүнг гүйцэх боломжтой гэдэг нь Хүснэгт 2-с харагдсан.

Алгоритмуудын ажиллах хугацааг Хүснэгт 4-д харуулав. К оройг нэгэн зэрэг устгах нь алгоритмын хурдыг хэд дахин сайжруулж байна. Тухайлбал, hepph сүлжээнд бүх Алгоритмууд хамгийн удаан ажилласан бөгөөд Алгоритм 1-с бусад алгоритмууд 4-5 дахин хурдан ажиллав. "Устгах" болон "буцаан нэмэх" үйлдлүүдийн зөв хослолоор үүсэх аргаар (Алгоритм 3, 4) сүлжээг задлах үр дүнтэй алгоритм боловсруулах боломжтой гэдэг нь харагдлаа. Алгоритм 3, 4 нь сайн шийдийг богино хугацаанд олох боломжтойг харууллаа.

Хүснэгт 4: Дөрвөн алгоритм (Алгоритм 1 (A1), Алгоритм 2 (A2), Алгоритм 3 (A3), Алгоритм 4 (A4))-ын ажиллах хугацаа (секунд)

Сүлжээ	A1	A2	A3	A4
circuit	0.21	0.06	0.08	0.11
EU	402.55	52.48	88.46	103.80
facebook	721.00	210.94	282.60	330.10
hepph	4115.19	812.77	918.55	1019.23

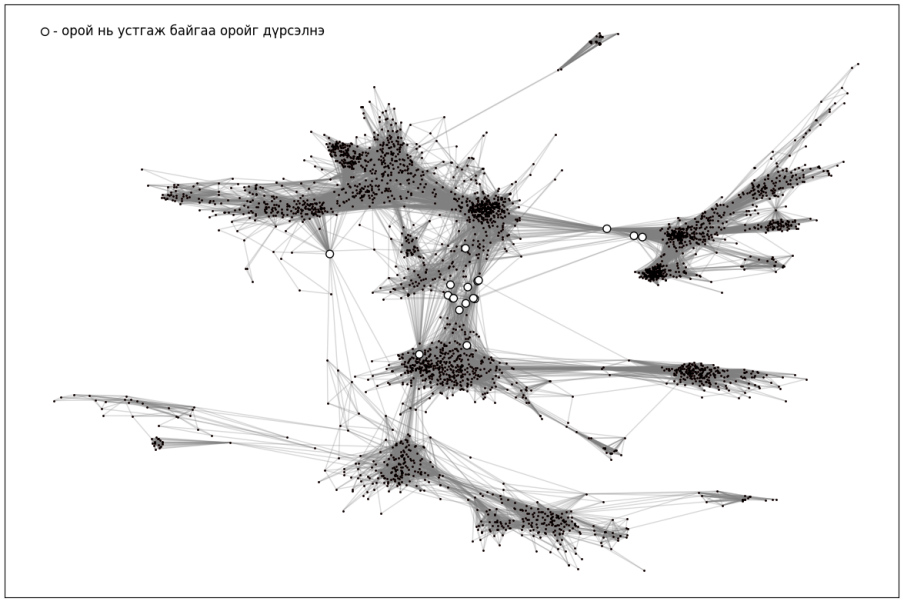
Хүснэгт 4-д харуулснаар оройнуудыг олноор нь устгах нь хугацааг 3-8 дахин хожиж байна. Харин дахин оруулах аргыг нэмж ашигласнаар (Алгоритм 3, 4) шийдийн чанар сайжирч байгаа боловч ажиллах өссөн байна.

4 Дүгнэлт

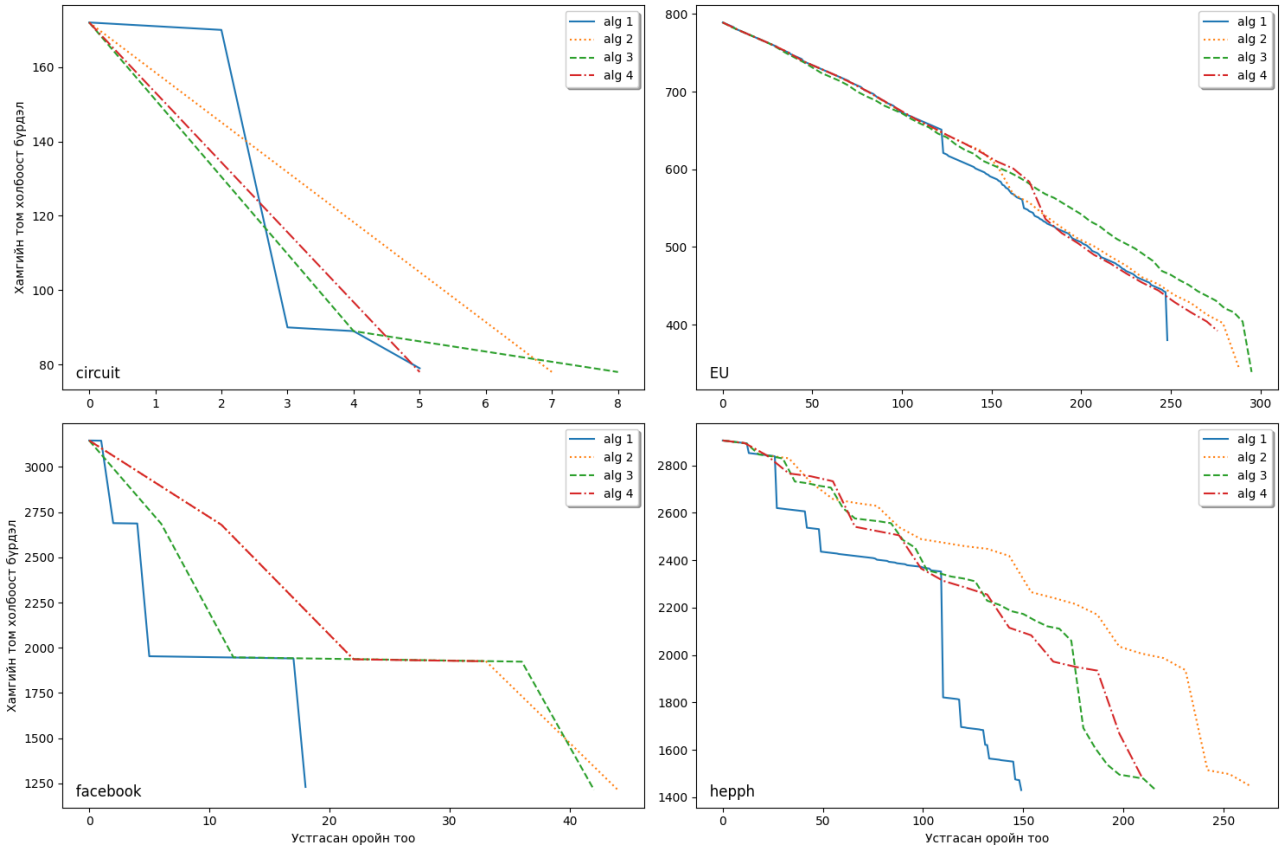
Хоорондын төвийн аргаар сүлжээг задлах дөрвөн алгоритмыг боловсруулан, үр дүнг харьцуулав. Хоорондын төвийн эрэмбээр нэгэн зэрэг олон орой илрүүлэх нь нэг нэгээр илрүүлсэнтэй өндөр хувьтай адил байна. Зөв "буцаан нэмэх" үйлдэл нь хоорондын төвийн аргаар нэгэн зэрэг олон орой илрүүлэх алгоритмын үр дүнг мэдэгдэхүйц сайжруулав. Хоорондын төвийн аргаар нэгэн зэрэг олон орой илрүүлэх алгоритм нь сайн шийдийг богино хугацаанд олох боломжтойг харууллаа.

Зохиогчийн оролцоо

Уг судалгааны дизайныг П.Далайжаргал гаргаж, Г.Гантулга алгоритмын хэрэгжүүлэлтийг гүйцэтгэв. Туршилтын үр дүнг нэгтгэх, дүгнэх, өгүүллийн бичилтийг бүх зохиогчид хамтран гүйцэтгэв.



Зураг 1: facebook сүлжээнээс Алгоритм 3-р устгагдах оройн олонлогийг дүрслэв



Зураг 2: Дөрвөн алгоритмаар устгах оройн тоо хамгийн том холбоост бүрдэлийн хэмжээнд хэрхэн нөлөөлж байгааг харуулав.

Хүснэгт 3: S1 олонлогийг бусад олонлогуудтай харьцуулан, нийтлэг элементийг тоо болон хувиар харьцуулав

	S1, S2			S1, S3			S1, S4		
Сүлжээ	тоо	хувь 1	хувь 2	тоо	хувь 1	хувь 2	тоо	хувь 1	хувь 2
circuit	3	60	43	3	60	60	3	60	60
EU	243	98	84	189	76	80	230	93	83
facebook	18	100	41	16	90	100	16	90	100
hepph	126	84	48	85	57	64	84	56	68

Санхүүжилт

Энэхүү судалгааны ажил нь Монгол Улсын Их Сургуулийн өндөр түвшний судалгааны төслийн (P2020-3978) санхүүжилтээр хэрэгжсэн судалгааны ажлын нэг хэсэг болно.

Ашиг сонирхлын зөрчилгүйн баталгаа

Ашиг сонирхлын зөрчилгүй болно.

Ашигласан ном

[1] Albert R, Hawoong J, Albert-László B. Error and Attack Tolerance of Complex Networks. Nature. 2000;406(6797):378-82.

[2] Arulselvan A, Commander CW, Shylo O, Pardalos PM. Cardinality-constrained critical node detection problem. In: Performance models and risk management in communications systems. Springer; 2011. p. 79-91.

[3] Salemi H, Buchanan A. Solving the distance-based critical node problem. INFORMS Journal on Computing. 2022.

[4] Lalou M, Tahraoui MA, Kheddouci H. The critical node detection problem in networks: A survey. Computer Science Review. 2018;28:92-117.

[5] Freeman LC. A Set of Measures of Centrality Based on Betweenness. Sociometry. 1977;40(1):35-41.

[6] Brandes U. A faster algorithm for betweenness centrality. The Journal of Mathematical Sociology. 2001;25(2):163-77.

[7] Geisberger R, Peter S, Dominik S. Better approximation of betweenness centrality. In Proceedings of the Meeting on Algorithm Engineering Expermiments. 2008:90-100.

[8] Tsalouchidou I, Baeza-Yates R, Bonchi F, Liao K, Sellis T. Temporal betweenness centrality in dynamic graphs. International Journal of Data Science and Analytics. 2020;9(3):257-72.

[9] Riondato M, Evgenios MK. Fast Approximation of Betweenness Centrality through Sampling. The Journal of Mathematical Sociology. 2016;30(2):438-75.

[10] Riondato M, Eli U. ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages. ACM Transactions on Knowledge Discovery from Data. 2018;12(5):1-38.

[11] Fan C, Li Z, Yuhui D, Muhao C, Yizhou S, Zhong L. Learning to Identify High Betweenness Centrality Nodes from Scratch: A Novel Graph Neural Network Approach. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 2019:559-68.

[12] Zhang Q, Rong-Hua L, Minjia P, Yongheng D, Guoren W, Ye Y. ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages. ArXiv:210710052 [Cs]. 2021.

[13] Tarjan RE, Uzi V. An Efficient Parallel Biconnectivity Algorithm. SIAM Journal on Computing. 1985;14(4):862-74.

[14] Tian L, Bashan A, Shi DN, Liu YY. Articulation points in complex networks. Nature communications. 2017;8(1):1-9.

[15] Leskovec J, Jon K, Christos F. Graph Evolution: Densification and Shrinking Diameters. ACM Transactions on Knowledge Discovery from Data. 2007;1(1):1-41.

[16] Wandelt S, Sun X, Feng D, Zanin M, Havlin S. A comparative analysis of approaches to network-dismantling. Scientific reports. 2018;8(1):1-15.