Mongolian Journal of Engineering and Applied Sciences, 2023, 5(1)

Компьютерын ухаан

Ойролцоо k хоорондын төв

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

[1]МУИС, МКУТ

Хүлээн авсан 2023.05.24; Хянагдсан 2023.09.16; Зөвшөөрөгдсөн 2023.09.27

*Холбоо баригч зохиогч: gantulgag@seas.num.edu.mn.

Хураангуй

Сүлжээний хоорондын төв (ХТ)-ийг тооцоолох зарим хувилбар (k хоорондын төв, ойролцоо хоорондын төв)-уудыг харьцуулан үнэлнэ. k хоорондын төв (k-ХТ) нь сүлжээний хол байрших тоглогчид хооронд мэдээлэл дамжих боломж багатай байдаг шинж чанарыг загварчлах зорилгоор ХТ-ийг тооцоолохдоо k -с хэтрэхгүй урттай замуудын мэдээллийг авч үздэг. Ойролцоо хоорондын төв (ойролцоо ХТ)-ийн судалгаа сүүлийн жилүүдэд эрчимтэй хийгдэж байгаа боловч k-ХТ-ийг ойролцоолох судалгаа дутмаг байна. Бид энэ өгүүллээр k-ХТ-ийг түүврийн аргатай хослуулан ойролцоо k хоорондын төв (ойролцоо k-ХТ) алгоритмыг зохиомжлов. Ойролцоо k-ХТ-ийн алгоритмын хугацааны үнэлгээ нь $O(log^3(n) * d^k)$ байна, энд n оройн тоо, d нь сүлжээний дундаж зэрэг, k нь авч үзэх замын дээд уртыг илэрхийлнэ. Өргөн хэрэглэгддэг гурван сүлжээн дээр туршилтыг гүйцэтгэв. Ойролцоо k-ХТ-ийн алгоритмын ($k \geq 4$ үед) үр дүн нь ХТ-ийн алгоритмын үр дүнтэй өндөр хувийн корреляци-тай бөгөөд дунджаар 1,000 дахин бага хугацаанд ажиллаж ($k \in \{4; 5\}$ үед) байгаа нь туршилтын үр дүнгээс харагдав.

Түлхүүр үг: нийгмийн сүлжээ, хоорондын төв, комплекс сүлжээний анализ, ойролцоо алгоритм

1 Удиртгал

Хоорондын төв (betweenness centrality) нь сүлжээний бусад оройн хооронд мэдээлэл дамжуулах үйл явц дахь оролцоог үнэлэх замаар оройнуудад эрэмбэ оноох арга юм. Хоорондын төв (ХТ)-ийн оноо нь өндөр бол тэр оройг мэдээлэл дамжуулах үйл явцад чухал үүрэгтэй, сүлжээний гол тоглогчдын нэг гэж үзэх юм. ХТ-ийг сүлжээт систем (networked system)-ийг шинжлэх зарим хэрэглээг товч авч үзье. Хотын одоо байгаа болон шинээр баригдаж буй хэсэг хоорондын зорчигчийн хөдөлгөөний ачаалал, тархалтыг тооцоолох, зорчигчийн зан төлөвийг загварчлахад хоорондын төвийг ашиглах судалгаа [1]-д хийгдсэн байна. Сэтгэн бодох үйл ажиллагааны явцад хүний тархины функциональ сүлжээний бүтцийн шинж чанарыг хоорондын төвийн эрэмбийг ашиглан судалжээ [2]. Уг судалгааны үр дүнд сэтгэн бодох үйл ажиллагаа нь илүү тархсан функциональ сүлжээг бий болгодог бөгөөд энэ нь танин мэдэхүйн ачааллыг тооцоолоход зориулагдсан тархи-компьютерын интерфейсийг төлөвлөхөд тусалдаг болохыг харуулж байна. Өөр нэг судалгаа [3] нь сүлжээний дутуу холбоосыг таамаглахад хоорондын төвийг ашиглажээ. Хууль бус мансууруулах бодис түгээх сүлжээн дэх оролцогчдын байршилд дүн шинжилгээ хийхдээ хоорондын төвийг ашигласан бөгөөд өндөр ХТ-тэй оролцогчид хууль сахиулах байгууллагад баривчлагдах магадлал багатай болохыг харуулсан байна [4].

ХТ-ийн оноог тооцоолох $O(nm)$ хугацааны Брэндсын алгоритм [5] нь сүлжээнд анализ хийхэд өргөн хэрэглэгддэг, энд n сүлжээний оройн тоо, m нь ирмэгийн тоог илэрхийлнэ. Нэг эхлэлтэй богино замын бодлого (single source shortest path problem)-ыг n удаа давтан хэрэглэж, орой бүрийн ХТ-ийн оноог тооцоолдог. Жингүй сүлжээний хувьд түвшний нэвтрэлтээр ($O(n+m)$), харин жинтэй сүлжээний хувьд Дайкстрагын алгоритмаар ($O(m + nlogn)$) [6] богино замуудыг тооцоолдог. Иймд Брэндсын алгоритмын хугацаа нь жингүй сүлжээний хувьд ($O(nm)$), жинтэй сүлжээний хувьд ($O(nm + n^2 * logn)$) байна. Энэ судалгаагаар бид жингүй, чиглэлгүй сүлжээг авч үзнэ.

Өөр хоорондоо хол зайд байрлах оройнууд хооронд мэдээлэл дамжих нь бодит сүлжээнд харьцангуй ховор байдаг. Хэт урт замуудын мэдээллийг ашиглахгүй байх ХТ-ийн эрэмбийг k-ХТ судалдаг [7–10]. k-ХТ нь сүлжээний мэдээлэл дамжих үйл ажиллагаанд идэвхтэй оролцох холболтуудыг онцлон авч үзэх боломжтой юм. Тодруулбал, сүлжээний өгөгдсөн k (тухайлбал, k=3) тооноос хэтрэхгүй урттай замуудын мэдээллийг ХТ-ийн тооцоололд авч үзэх юм.

Сүлжээний хэмжээ томорсонтой холбоотойгоор Брэндсын алгоритмын хурдыг сайжруулах асуудал сүүлийн жилүүдэд эрчимтэй судлагдаж байна. n удаа (бүх оройноос нэг удаа) нэг эхлэлтэй богино зам (НЭБЗ)-ын бодлогыг давтан бодохын оронд таамгаар зарим (тухайлбал, $O(log^3(n))$ шир-

хэг) оройг түүвэрлэн авч, тэдгээр оройноос НЭБЗ-уудыг олоод ХТ-ийн оноог ойролцоогоор тооцоолох боломжтой [11–14]. k-ХТ-ийг ойролцоолох судалгаа дутмаг байна. Энэ өгүүллээр бид k хоорондын төвийг түүврийн аргаар ойролцоолон, ойролцоо k хоорондын төв (ойролцоо k-ХТ) алгоритмыг дэвшүүлэн, туршилтаар үр дүнгийн үнэлгээ хийж, бусад ХТ-ийн хувилбартай харьцууллаа. Хоёрдугаар бүлэгт ХТ-ийг ойролцоолох түүврийн арга болон ойролцоо k-ХТ-ийн алгоритмыг тодорхойлно. Гуравдугаар бүлэгт туршилтын үр дүн, харьцуулсан үнэлгээг оруулав. Дөрөвдүгээр бүлэгт дүгнэлт, цаашид хийгдэх боломжтой өргөтгөлийн санаа зэргийг тодорхойлов.

2 Ойролцоо k-хоорондын төв

Граф өгөгдөхөд v оройн ХТ (ВС)-ийн утга нь бүх s,t хослолын хувьд s оройгоос t орой хүрэх богино замын хэд нь тухайн v оройгоор дайрсан харьцааны нийт нийлбэрээр тодорхойлогдоно (Тэгшитгэл 1).

$$BC(v) = \sum_{s,t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{1}$$

Олон нийтэд Брэндсын алгоритм гэж танигдсан, жингүй граф дээр ХТ-ийг $O(nm)$ хугацаанд олдог, динамик программчлалын санаан дээр тулгуурласан, суурь алгоритмыг бид цаашид ХТ алгоритм гэх болно [5]. Энэхүү алгоритм нь ХТ утгыг яс (exact) олдог бөгөөд хослолын хамаарлыг хурдан бодох арга дээр тулгуурладаг. Хослолын хамаарал нь

$$\delta_{st}[v] = \frac{\sigma_{st}[v]}{\sigma_{st}}$$

гэж тодорхойлогддог. Өөрөөр хэлбэл s,t хослолын хувьд v орой орсон богино замын тоог s-ээс t хүрэх нийт богино замын тоонд хуваасан харьцаа юм. Тэгвэл ХТ-ийн тодорхойлолтоор Тэгшитгэл 1-ийг дараах байдлаар бичиж болно.

$$BC(v) = \sum_{s,t \in V} \delta_{st}[v]$$

Уг алгоритм нь эхлээд графын бүх оройгоос, олон богино замыг хадгалах чадвартай түвшний нэвтрэлт хийнэ. Түвшний нэвтрэлт s оройгоос эхэлсэн бол бусад бүх v оройн хувьд тухайн орой дээр хамгийн богино замаар ирэх эцгийн олонлог $\pi_s[v]$, v оройд хүрэх нийт богино замын тоо $\sigma_s[v]$-г байгуулна. Тэгвэл s оройгоос эхлэлтэй түвшний нэвтрэлтийн хувьд v орой орсон хослолын хамаарлыг

$$\delta_s[v] = \sum_{t \in V} \delta_{st}[v]$$

гэж тэмдэглэе. Уг тэгшитгэл нь дараах рекурсив бүтцэд задардаг нь уг алгоритмыг хурдан ажиллахад хүргэдэг.

$$\begin{aligned}\delta_s[v] &= \sum_{t \in V} \delta_{st}[v] \\ &= \sum_{t \in V} \sum_{w : v \in \pi_s[w]} \delta_{st}[v,(v,w)] \\ &= \sum_{w : v \in \pi_s[w]} \sum_{t \in V} \delta_{st}[v,(v,w)]\end{aligned}$$

Дээрх тэгшитгэлд бүх w оройгоор нэмж байгаа нь богино замын дарааалалд v орой өвөг нь байх буюу эхлээд v орой, дараа нь (v,w) ирмэгийг ашиглан w ирэх бүх w ирмэгээр нэмж байна. Харин $\delta_{st}[v,(v,w)]$ нь s,t хослолд v оройгоос зөвхөн (v,w) ирмэгийг ашигласан хослолын хамаарлыг илтгэх бөгөөд дараах тэгшитгэлээр тодорхойлогдоно.

$$\delta_{st}[v,(v,w)] = \begin{cases} \frac{\sigma_{sv}}{\sigma_{sw}} & \text{хэрэв } t = w \\ \frac{\sigma_{sv}}{\sigma_{sw}} \cdot \frac{\sigma_{st}[w]}{\sigma_{st}} & \text{хэрэв } t \neq w \end{cases}$$

Дээрх системд $t = w$ байх үе бол илэрхий. Учир нь s оройгоос t хүрэх нийт замын тоо $\sigma_{st} = \sigma_{sw}$ бөгөөд үүний σ_{sv} ширхэг нь v оройд ирээд (v,w) ирмэгийг ашиглана. Харин $t \neq w$ үед s оройгоос t орой хүрэх богино замд w орой орсон байх богино замын тоо $\sigma_{st}[w] = \sigma_{sw} \cdot \sigma_{wt}$ байна. Учир нь s-ээс w хүрэх бүх богино замууд w-ээс t хүрэх богино замуудтай нийлж s-ээс t орой хүрэх богино зам болж чадна. Тэгвэл w оройгоос t хүрэх нийт замын тоо $\sigma_{wt} = \frac{\sigma_{st}[w]}{\sigma_{sw}}$ байх бөгөөд s,t хослолын хувьд v оройд ирээд (v,w) ирмэгийг ашигласан нийт замын тоо

$$\sigma_{sv} \cdot \sigma_{wt} = \sigma_{sv} \cdot \frac{\sigma_{st}[w]}{\sigma_{sw}}$$

байх бөгөөд хослолынх нь хамаарал σ_{st} хуваагж гарч ирнэ. Үүнийг s оройгоос эхлэлтэй богино замд v орой орсон хослолын хамаарлыг олох тэгшитгэлд залгавал

$$\begin{aligned}&\sum_{w\ : v \in \pi_s[w]} \sum_{t\ \in V} \delta_{st}[v,(v,w)] \\ &= \sum_{w : v \in \pi_s[w]} \left(\frac{\sigma_{sv}}{\sigma_{sw}} + \sum_{t \in V/w} \sigma_{sv} \cdot \frac{\sigma_{st}[w]}{\sigma_{sw}} \right) \\ &= \sum_{w : v \in \pi_s[w]} \frac{\sigma_{sv}}{\sigma_{sw}} \cdot (1 + \delta_s[w])\end{aligned}$$

болох бөгөөд $\delta_s[v]$ нь рекурсив хамааралтай болох нь харагдаж байна. Уг рекурсивыг бодохдоо s оройгоос хамгийн хол орших оройгоос эхлэх байгуулалтыг хийснээр ХТ утгыг олох боломжтой. Уг алгоритмын хэрэгжүүлэлт нь Алгоритм 1-тэй төстэй бөгөөд 3, 4-р мөрний оронд бүх V оройгоор гүйх ба 26, 27-р мөрөнд байгаа байгуулалтыг арилгаж, 15-р мөрөнд байрлах гүний хязгаарыг арилгаснаар ХТ алгоритм гарч ирнэ.

k-ХТ алгоритм нь [9]-д дэвшүүлэгдсэн алгоритм бөгөөд ХТ алгоритмаас ялгаатай нь 15-р мөрөн дэх

гүний хязгаарлалт орж ирэх юм. Ингэснээр түвшний хайлт k гүн яваад зогсох бөгөөд бүх оройгоос хийх түвшний хайлтыг хурдасгаснаар k-ХТ алгоритм $O(n{\cdot}d)$ хугацаанд ажиллана. Энд D нь дундаж зэрэг бол k нь хэр гүн явахаас хамаарсан тогтмол болно.

ХТ утгыг бодох нь бүх оройгоос богино зам бодох тул том сүлжээн дээр ажиллуулахад хүндрэлтэй байдаг. Энэ асуудлыг давахын тулд ХТ-ийг ойролцоогоор боддог олон арга дэвшүүлэгдсэний нэг нь ойролцоо ХТ алгоритм юм [12]. Уг алгоритм нь алдааны онолын баталгаа байхгүй ч бодит амьдрал дээр сайн ажилладаг [12] гэдгээрээ олонд танигдсан бөгөөд ажиллах хугацаа нь $O(n \cdot r)$ болно. Энд r нь түүвэрлэж сонгох оройн тоо бөгөөд бүх оройгоос богино зам бодохын оронд санамсаргүйгээр түүвэрлэсэн r ширхэг оройгоос түвшний нэвтрэлт хийж, ХТ утгыг байгуулсныхаа дараа уг мэдээллээсээ бусад оройн ХТ утгыг ойролцоолж байгуулдаг. Алгоритм 1-ээс ялгаатай нь 15-р мөрний гүний хязгаарлалт байхгүй.

Ойролцоо k-ХТ алгоритмын хуурмаг кодыг Алгоритм 1-д харуулав. Доорх хуурмаг код нь [11]-д дэвшүүлсэн алгоритм дээр, [9]-д ашигласан замын уртын хязгаарыг тавьсан бөгөөд эхний ажлаас ялгаатай нь алгоритмын 15, 16-р мөрөнд байгаа хэсэг болно. Уг алгоритм нь санамсаргүйгээр түүвэрлэсэн r ширхэг орой бүрийн хувьд түвшний нэвтрэлтээр бусад бүх орой хүрэх хамгийн богино замын мэдээллийг байгуулдаг (9–19-р мөр). Энэхүү байгуулалтыг хийх явцад хамгийн богино замын урт k-аас бага биш болох үед тухайн оройг түвшний нэвтрэлтийн Q дараалалд оруулахгүй. Энэ нь замын урт нь k-аас их буюу тэнцүү болсон оройгоос цаашаа хайлт хийгдэхгүй гэсэн үг болно. Бид туршилтаа хийхдээ [11]-д дэвшүүлсэн алгоритмын сайжруулсан хувилбар буюу түүвэр оройтой ойр байрлах оройнууд хэт их ХТ утга авч байгаа алдааг багасгасан [12] хувилбар дээр замын хязгаарыг оруулж туршилтыг гүйцэтгэсэн.

Ойролцоо k-ХТ-ийн хугацааны төвөг (time complexity) -ийг бодож үзье. Тооцоолох шаардлагатай НЭБЗ бодлогын тоо нь r буюу $O(log^3 n)$ байна. НЭБЗ-д замын уртад хязгаарлалт байхгүй бол $O(n + m)$ байдаг [15]. Түвшний нэвтрэлтийн эхний гүнд d оройд хүрнэ, энд d нь сүлжээний дундаж зэргийг илэрхийлнэ. Хоёрдугаар гүнд ихдээ d^2 оройд хүрэх, гуравдугаар гүнд ихдээ d^3 оройд хүрэх боломжтой гэх мэтээр үргэлжилнэ. Богино замын уртыг k -р хязгаарласнаар түвшний нэвтрэлтийн k -р гүнд ихдээ d^k ($d^k < n$) оройд хүрнэ. Замын уртыг k -р хязгаарласнаар түвшний нэвтрэлт ихдээ $1 + d^1 + d^2 + ... + d^k = O(d^k)$ оройд хүрэх боломжтой. Иймд ойролцоо k-ХТ-ийн хугацааны төвөг нь $O(log^3 n \cdot d^k)$ байна.

Algorithm 1 Ойролцоо k-ХТ алгоритм

ESTIMATED-K-BETWEENNESS(V, r, k)

1: **for** $u \in V$
2: $\quad BC[u] \leftarrow 0$
3: $V' \leftarrow \{V$-ээс санамсаргүй сонгосон r орой$\}$
4: **for** $s \in V'$
5: $\quad$ **for** $u \in V$
6: $\quad\quad dist[u] \leftarrow \infty, \sigma[u] \leftarrow 0, \delta[u] \leftarrow 0, \pi[u] \leftarrow \{\}$
7: $\quad dist[s] \leftarrow 0, \sigma[s] \leftarrow 1$
8: $\quad Q$ дараалалд s-ийг хий
9: $\quad$ **while** $Q \neq \emptyset$
10: $\quad\quad Q$ дарааллаас u оройг гарга
11: $\quad\quad S$ стакт u оройг хий
12: $\quad\quad$ **for** u оройн хөрш v оройн хувьд
13: $\quad\quad\quad$ **if** $dist[v] = \infty$
14: $\quad\quad\quad\quad dist[v] \leftarrow dist[u] + 1$
15: $\quad\quad\quad\quad$ **if** $dist[v] < k$
16: $\quad\quad\quad\quad\quad Q$ дараалалд v оройг хий
17: $\quad\quad\quad$ **if** $dist[v] = dist[u] + 1$
18: $\quad\quad\quad\quad \sigma[v] \leftarrow \sigma[v] + \sigma[u]$
19: $\quad\quad\quad\quad \pi[v]$ олонлогт u оройг хий
20: $\quad$ **while** $S \neq \emptyset$
21: $\quad\quad S$ стакаас u оройг гарга
22: $\quad\quad$ **for** $v \in \pi[u]$
23: $\quad\quad\quad \delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[u]} \cdot (1 + \delta[u])$
24: $\quad\quad$ **if** $u \neq s$
25: $\quad\quad\quad BC[u] \leftarrow BC[u] + \delta[u]$
$\quad$ // бусад оройн BC-г байгуулах
26: **for** $\forall v \in V$
27: $\quad BC[v] \leftarrow BC[v] \cdot \frac{2n}{k}$

3 Туршилт

Энэ бүлэгт бид k-ХТ, оролцоо k-ХТ, оролцоо ХТ [12] алгоритмуудыг ХТ алгоритмтай ажиллах хугацаа болон шийдийн чанарын хувьд харьцуулах болно. Харьцуулалтыг хийхдээ $k \in \{3; 4; 5\}$ гэсэн гурван зайны уртыг k-ХТ алгоритмд сонирхон авч судалсан. Ойролцоолол дээр тулгуурласан алгоритмуудын ойролцооллын түүвэр оройн хэмжээг $\log^3 n$ гэж сонгосон болно. Эдгээр туршилтуудыг Python хэл дээр Networkit [16] санг ашиглан Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz хурдтай, 8GB санах ойтой процессор дээр ажиллуулсан.

3.1 Туршилтын өгөгдөл

Бид бодит амьдрал дээрх 3 том сүлжээ, санамсаргүйгээр үүсгэгдсэн 2 том сүлжээг энэ туршилтад ашигласан. Эдгээр сүлжээг Хүснэгт 1-д харуулав. *Hepth* сүлжээ нь Өндөр Энергийн Физикын Үзэгдэл (High Energy Physics Phenomenology) судалдаг физикийн салбарын судлаачдын эшлэлүүдээр үүсгэгдсэн хамтын ажиллагааны граф юм. *Twitter* сүлжээ нь олон нийтэд нээлттэй их сурвалжуудаас цуглуулсан Twitter-ийн сүлжээний найзуудын граф

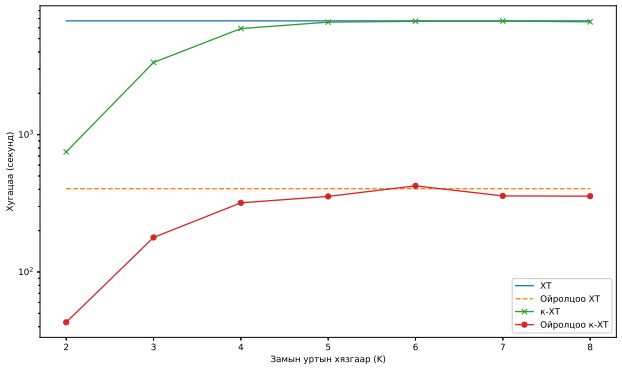
юм. *Amazon* сүлжээ нь худалдан авалтын граф бөгөөд оройнууд нь бараа, ирмэгүүд нь тухайн хоёр бараа хамт худалдан авагдсан гэдгийг илтгэнэ. Эдгээр гурван бодит сүлжээг [17] эх сурвалжаас авах боломжтой. Эрдёс Рени (ER), Ватса Строгац (WS) моделийн дагуу хоёр санамсаргүй сүлжээг Networkit санг ашиглан үүсгэсэн. ER сүлжээг үүсгэхдээ ирмэгийн магадлалыг 0.0004 утгаар сонгон авсан. WS сүлжээг үүсгэхдээ хөршийн тоог 10, ирмэг холбох магадлалыг 0.5 утгаар сонгон сүлжээг үүсгэсэн.

Хүснэгт 1-д өгөгдлийн зарим шинж чанарыг харуулав. *Twitter* нь нөгөө хоёр бодит сүлжээ дундаа хамгийн нягт бодит сүлжээ. *Amazon* нь уг туршилтад ашигласан хамгийн том сүлжээ бөгөөд бусадтай харьцуулахад диаметр томтой, дундаж зэрэг нь бага гэсэн үзүүлэлтүүд нь уг сүлжээ харьцангуй сийрэг сүлжээ болохыг харуулж байна. Харин хоёр санамсаргүй сүлжээ нь тодорхой загварын дагуу үүсгэгдэж байгаа учраас диаметр жижиг, дундаж зэрэг томтой, нягт сүлжээ болно.

3.2 Ажиллах хугацааны харьцуулалт

Харьцуулж буй XT алгоритмуудын ажилласан хугацааг Хүснэгт 2-д харуулав. XT алгоритм алдаагүй XT утга гаргаж байгаа тул эдгээрийн хамгийн удаан нь байна. Харин ойролцоо XT-ыг ашигласнаар хурдыг 3-аас 56 дахин сайжруулж байна (*Hepth* , *Amazon* сүлжээнүүд). Сүлжээ томсох тусам ажиллах хугацааны харьцаа их болж байгааг Хүснэгт 2 -оос харж болно. XT болон ойролцоо XT алгоритмуудын хувьд *Amazon* , *ER* сүлжээнүүд дээр хамгийн удаан ажилласан. XT алгоритм *Amazon* , *ER* сүлжээнүүд дээр харгалзан 65729.39 сек, 20380.81 сек ажилласан бол ойролцоо XT харгалзан 56, 35 дахин хурдан ажилласан байна. Ойролцоо XT алгоритм *Hepth* гэсэн жижиг сүлжээнээс бусад дээр нь хурдыг 17-оос 56 дахин сайжруулсан байна. K-XT алгоритмыг ашиглан замын уртын хязгаар тавьснаар ялангуяа сийрэг граф дээр ойролцоо XT алгоритмаас илүү хугацааг хожиж байгааг туршилтын үр дүн харууллаа. *Hepth* , *Amazon* гэх мэт сийрэг сүлжээнүүд дээр ойролцоо XT харгалзан 3, 56 дахин хурдан ажилласан бол k-XT (удаан хувилбар болох $k = 5$) харгалзан 4, 87 дахин хурдан ажилласан байна. Харин илүү нягт бүтэцтэй *Twitter*, *ER* сүлжээнүүд дээр k-XT алгоритм удаан буюу $k = 5$ үед бараг XT алгоритмтай адил хугацаанд, $k = 4$ үед 2 дахин хурдан ажилласан. Мөн илүү нягт сүлжээний $|E^3|$, $|E^4|$-ийн зөрөө харьцангуй их байгаа нь k-XT-ийн замын урт нэмэгдэхэд хурд нь харьцангуй удаашрахыг илтгэнэ.

Ойролцоо k-XT алгоритм нь k-XT болон ойролцоо XT алгоритмуудын аль алийн давуу талыг агуулах бөгөөд $k = 3$ үед 91-ээс 2281 дахин, $k = 5$ үед 16-аас 4597 дахин хурдан ажилласан. Уг алгоритмын замын уртаар хязгаарлагдах шинж нь XT алгоритмын замын урт тооцоолох үйлдлийг хурдлуулж, том сүлжээ дээр хурдан ажиллах төдийгүй ойролцоололд тулгуурласан шинж нь нягт граф дээр бүх оройг авч үзэлгүй, цөөн түүвэр оройг ашиглан тооцоолол хийх тул хурдыг олон дахин сайжруулах үндэс болно. Ойролцоо k-XT хамгийн удаан хувилбар болох $k = 5$ үед ч бүх сүлжээн дээр бусдаасаа хурдан ажилласан бөгөөд нягт сүлжээнүүд болох *Twitter*, *ER* дээр харгалзан 22, 25 дахин хурдан, сийрэг сүлжээнүүд болох *Hepth*, *Amazon* дээр харгалзан 16, 4597 дахин хурдан ажилласан.



Зураг 1: XT алгоритмуудын ажиллах хугацаа Twitter сүлжээн дээр харуулав. X тэнхлэг замын уртын хязгаар k, Y тэнхлэг ажиллах хугацааг тус тус илэрхийлнэ.

Замын уртын хязгаар XT тооцоолох хугацаанд нөлөөлөх нөлөөллийг *Twitter* сүлжээн дээрх туршилтын үр дүнг Зураг 1-д харуулав. XT болон ойролцоо XT алгоритмуудад уртын хязгаарлалт байхгүй тул ажиллах хугацаа өөрчлөгдөхгүй. Хязгаар $k = 3$-аас эхлэх үед k-XT алгоритмын ажиллах хугацаа XT алгоритмын доороос, ойролцоо k-XT алгоритмын ажиллах хугацаа ойролцоо XT-ийн доороос эхэлж байна. Харин k ихсэх үед ажиллах хугацаа огцом нэмэгдэж байгаа үзэгдэл *Twitter* нягт сүлжээн дээр ажиглагдаж байна. *Twitter* сүлжээний диаметр нь 7 боловч ажиллах хугацаа нь $k = 6$ үед XT алгоритмын ажиллах хугацаатай адил байгаа нь 6 урттай зам графыг бараг бүрхэж байгааг илтгэнэ. Уртын хязгаар 7 хүрэх үед k-XT алгоритм XT алгоритмтай ялгаагүй болно. Ойролцоо k-XT алгоритм 43 секундээс эхлэн, $k = 6$ үед ойролцоо XT алгоритмын хугацаатай адил түвшинд очиж байна. Тэгээд тэр орчимд савлаж байгаа нь ойролцоо XT оройнуудыг санамсаргүй түүвэрлэдэг тул ямар орой сонгосноос хамааран жижиг савлалт үүсгэснээс болж байна. Зургаас үзэхэд $k = 6$ нь ямар хязгаар тавиагүй үеийнхтэй ойролцоо харагдаж байгаа тул $k \in \{4; 5\}$ утгууд магадгүй хамгийн өндөр XT утгатай L оройг жагсаах хэрэгцээнд хангалттай байж болох санааг харуулж байна.

Хүснэгт 1: Туршилтын өгөгдөл. Багануud нь оройн тоо (n), эрмэгийн тоо (m), дундаж зэрэг (d), диаметр (dia), 3 уртаар хүрч болох замын тоо ($|E^3|$), 4 уртаар хүрч болох замын тоо ($|E^4|$).

Сүлжээ	n	m	d	dia	$\|E^3\|$	$\|E^4\|$
Hepth	8,361	15,751	1.88	19	376,431	1,340,125
Twitter	81,306	1,342,310	16.51	7	978,469,063	2,665,262,949
Amazon	334,863	925,871	2.76	47	26,004,103	98,465,328
ER	100,000	2,000,509	20.01	5	2,380,742,533	4,999,949,630
WS	100,000	1,000,000	10.00	6	271,391,046	2,966,315,203

Хүснэгт 2: XT-ын алгоритмуудын ажиллах хугацааг (секунд) харуулав. k нь k-XT бодоход ашигласан замын уртын хязгаар. Харьцааг XT алгоритмын ажилласан хугацаатай харьцуулж гаргасан.

Сүлжээ	XT		k-XT					
			$k=3$		$k=4$		$k=5$	
	хугацаа	харьцаа	хугацаа	харьцаа	хугацаа	харьцаа	хугацаа	харьцаа
Hepth	6.86	-	0.31	21.92	0.65	10.62	1.52	4.53
Twitter	7,163.15	-	748.84	9.57	3,350.81	2.14	5,915.40	1.21
Amazon	65,914.67	-	550.04	119.84	590.68	111.59	751.08	87.76
ER	20,380.81	-	707.24	28.82	10,278.11	1.98	16,722.91	1.22
WS	9,559.03	-	84.40	113.26	1,099.99	8.69	6,413.20	1.49
Дундаж	20,604.90		418.17	58.68	3,064.05	27.00	5,960.82	19.24
	Ойролцоо XT		Ойролцоо k-XT					
			$k=3$		$k=4$		$k=5$	
	хугацаа	харьцаа	хугацаа	харьцаа	хугацаа	харьцаа	хугацаа	харьцаа
Hepth	2.27	3.03	0.08	90.87	0.18	38.73	0.42	16.28
Twitter	403.77	17.74	45.25	158.31	178.45	40.14	318.77	22.47
Amazon	1,171.05	56.29	35.07	1,879.41	11.04	5,969.73	14.34	4,597.31
ER	953.57	21.37	35.83	568.83	495.72	41.11	810.55	25.14
WS	456.84	20.92	4.19	2,281.08	50.31	189.99	297.45	32.14
Дундаж	597.50	23.87	24.08	995.70	147.14	1,255.94	288.30	938.67

3.3 Чанарын харьцуулалт

XT алгоритм дээр замын хязгаар тавих нь хурдан болгож байна. Харин энэ бүлэгт уг үйлдэл шийдийн чанарт хэрхэн нөлөөлж буйг судалцгаая. Бид Спирмений хамаарлын коэффициентыг ашиглан k-XT болон ойролцоолол дээр тулгуурласан алгоритмууд XT алгоритмын үр дүнтэй хэр нийцэлтэй байгааг шинжлэв. Мөн бид XT утгаараа эхний L оройн хэд нь XT алгоритмын эхний $2L$ оройгоос тогтох олонлогт багтаж байгааг ажиглаж, харьцуулалт хийсэн. $L = 10$, $L = \sqrt{n}$ гэсэн хоёр утгыг энэ туршилтад сонгон авсан.

Хүснэгт 3-ээс үзэхэд ойролцоо XT алгоритмын гаралт гурван бодит сүлжээн дээр хамаарлын коэффициент өндөр буюу 0.99 гарч ирж байна. Ойролцоо XT алгоритмын эхний $\sqrt{n}$ болон 10 орой бүгд XT алгоритмтай 100 % нийцтэй бөгөөд ямар алдаагүй үр дүнг харуулав. Харин санамсаргүй сүлжээний хувьд ойролцоо XT алгоритм тийм ч сайн ажиллахгүй байгааг туршилтын үр дүн харуулж байна. Алгоритмын хамаарлын коэффициент нь хоёр сүлжээн дээр харгалзан $0.86, 0.88$ бөгөөд 316 оройгоос *ER* сүлжээн дээр 156, *WS* сүлжээн дээр 186 оройг зөв илрүүлсэн байна. Харин санамсаргүй XT алгоритмын хувьд бүх сүлжээн дээр хамаарал байхгүй гарсан бөгөөд хамгийн ихдээ 5 ширхэг оройг л зөв таньж чадаж байгаа юм. Энд ойролцоо k-XT алгоритмын үр дүнг санамсаргүй XT алгоритмтай харьцуулах үүднээс үр дүнг оруулсан болно.

Ойролцоо k-XT, k-XT алгоритмуудыг $k \in \{3; 4; 5\}$ утгуудаар ажиллуулан харьцуулалтын үр дүнг Хүснэгт 4-д харуулав. K-XT алгоритм гурван бодит сүлжээн дээр ойролцоо k-XT алгоритмаас муу үр дүнг үзүүлж байгаа боловч санамсаргүй сүлжээ дээр илүү сайн байна. Харин ойролцоо k-XT алгоритм гурван бодит сүлжээн дээр k-XT алгоритмаас сайн үр дүнг үзүүлэв. Ойролцоо XT алгоритм сул талыг дагаж санамсаргүй сүлжээн дээр k-XT алгоритмаас муу үр дүнг харууллаа. Ойролцоо k-XT алгоритм $L = 10$ байх үед бодит сүлжээнүүд дээр 6-аас дээш оройг амжилттай олсон бол $k = 4$ үед санамсаргүй сүлжээн дээр бүх 10 оройг олсон. Мэдээж k нэмэгдэх үед k-XT дээр суурилсан хоёр алгоритмын чанар сайжирна. Гэхдээ $k = 4$ үед ойролцоо k-XT санамсаргүй сүлжээн дээр буцаад муу үр дүнг үзүүлж байгаа нь ойролцоо алгоритмын санамсаргүй шинж чанар мөн санамсаргүй сүлжээнүүд тусгай зүй тогтлын дагуу үүсгэгдэж байгаа тул ангилахад бэрх буюу хоорондоо төстэй XT төвтэй оройнууд их үүсгэж буйтай холбоотой гэж тааварлаж байна. Ойролцоо k-XT алгоритм $k = 5$ үед *Twitter*

Хүснэгт 3: Туршилтын алгоритмын хамгийн өндөр XT утгатай L оройн хэд нь XT алгоритмын хамгийн өндөр XT утгатай 2L оройн олонлогт агуулагдаж байгааг харуулав ($L = \sqrt{n}, L = 10$).

Сүлжээ	Ойролцоо XT			Санамсаргүй XT		
	корр	$L = \sqrt{n}$	$L = 10$	корр	$L = \sqrt{n}$	$L = 10$
Hepth	0.99	91/91	10/10	-0.01	5/91	0/10
Twitter	0.99	285/285	10/10	-0.01	5/285	0/10
Amazon	0.99	578/578	10/10	0.00	0/578	0/10
ER	0.86	156/316	4/10	0.00	4/316	0/10
WS	0.88	186/316	1/10	0.00	2/316	0/10

сүлжээн дээр 100 % зөв үр дүнг буцаасан. Харин *Amazon* сүлжээний хувьд хоёр алгоритм тааруу ажилласан бөгөөд 578 оройгоос k-XT 241 оройг, ойролцоо k-XT 270 оройг тус тус зөв олсон.

Ойролцоо k-XT хурдыг олон дахин хожиж байгаа боловч чанарын хувьд ойролцоо XT алгоритмыг аль ч сүлжээний хувь гүйцэнгүй. Харин k-XT болон ойролцоо k-XT алгоритмуудын хооронд сонговол ойролцоо k-XT илүү хурдан, илүү оновчтой үр дүнг буцааж байна. Мөн ойролцоо XT дээр суурилсан хоёр алгоритм бодит сүлжээн дээр сайн гүйцэтгэлийг харуулж байна. Хурд болон чанарын хувьд тохиромжтой утга нь $k \in \{4; 5\}$ бөгөөд ойролцоо k-XT алгоритм харгалзан XT алгоритмаас дунджаар 1256, 939 дахин хурдан ажиллаж байна.

4 Дүгнэлт

Энэ өгүүллээр хоорондын төвийг түүврийн аргаар ойролцоолон бодох ойролцоо k хоорондын төв (ойролцоо k-XT)-ийн алгоритмыг зохиомжлов. Тус алгоритмын хугацааны төвөг (time complexity) нь $O(log^3(n) * d^k)$ байна, энд d нь сүлжээний дундаж зэргийг илэрхийлнэ. Ойролцоо k-XT ($k = 4$ үед) алгоритмаар олдох оройнуудын эрэмбэ нь Брэндсын алгоритм ($O(nm)$ хугацааны алгоритм)-ын эрэмбэтэй 50-с 96-н хувийн корреляци-тай байв. Туршилтаар ойролцоо k-XT (k=4) алгоритм нь Брэндсын алгоритмаас дунджаар 1,256 дахин хурдан ажиллав. Цаашид туршилтын өгөгдлийн тоог нэмэгдүүлэн өөр төрлийн сүлжээний хувьд чанарыг үнэлгээг хийх шаардлагатай. Брэндсын алгоритмыг шууд хэрэглэдэг сүлжээний анализын олон арга, техникүүд байдаг. Тэдгээр аргуудад ойролцоо k-XT -ийг ашиглан үр дүнг харьцуулах нь сонирхолтой судалгаа болох талтай.

Зохиогчийн оролцоо

Уг судалгааны дизайн, алгоритмын санааг П.Далайжаргал гаргаж, Г.Гантулга алгоритмын хэрэгжүүлэлт, туршилтыг гүйцэтгэв. Туршилтын үр дүнг нэгтгэх, дүгнэх, өгүүллийн бичилтийг бүх зохиогчид хамтран гүйцэтгэв.

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 болохыг баталж байна.

Ашигласан ном

[1] Sevtsuk A. Estimating pedestrian flows on street networks: revisiting the betweenness index. Journal of the American Planning Association. 2021;87(4):512-26.

[2] Makarov VV, Zhuravlev MO, Runnova AE, Protasov P, Maksimenko VA, Frolov NS, et al. Betweenness centrality in multiplex brain network during mental task evaluation. Physical Review E. 2018;98(6):062413.

[3] Zhou M, Jin H, Wu Q, Xie H, Han Q. Betweenness centrality-based community adaptive network representation for link prediction. Applied Intelligence. 2022;52(4):3545-58.

[4] Morselli C. Assessing vulnerable and strategic positions in a criminal network. Journal of Contemporary Criminal Justice. 2010;26(4):382-92.

[5] Brandes U. A faster algorithm for betweenness centrality. Journal of mathematical sociology. 2001;25(2):163-77.

[6] Dijkstra EW. A note on two problems in connexion with graphs. In: Edsger Wybe Dijkstra: His Life, Work, and Legacy; 2022. p. 287-90.

[7] Borgatti SP, Everett MG. A graph-theoretic perspective on centrality. Social networks. 2006;28(4):466-84.

Хүснэгт 4: k-ХТ алгоритмын хамгийн өндөр ХТ утгатай L оройн хэд нь ХТ алгоритмын хамгийн өндөр ХТ утгатай $2L$ оройн олонлогт агуулагдаж байгааг харуулав ($L = \sqrt{n}, L = 10$). Сайн үр дүнгүүдийг тодруулав.

	$k=3$					
	k-ХТ			Ойролцоо k-ХТ		
Сүлжээ	корр	$L=\sqrt{n}$	$L=10$	корр	$L=\sqrt{n}$	$L=10$
Hepth	0.99	51/91	**7/10**	0.83	**71/91**	**7/10**
Twitter	0.89	179/285	5/10	0.70	**235/285**	**9/10**
Amazon	0.83	241/578	4/10	0.19	**270/578**	**8/10**
ER	1.00	**292/316**	**8/10**	0.40	39/316	0/10
WS	0.99	**257/316**	**5/10**	0.24	36/316	0/10
	$k=4$					
Hepth	0.99	54/91	**8/10**	0.96	**72/91**	7/10
Twitter	0.98	217/285	7/10	0.96	**280/285**	**10/10**
Amazon	0.90	260/578	5/10	0.50	**335/578**	**6/10**
ER	1.00	**303/316**	9/10	0.76	112/316	**10/10**
WS	1.00	**300/316**	8/10	0.62	86/316	**10/10**
	$k=5$					
Hepth	0.99	56/91	**8/10**	0.97	**76/91**	**8/10**
Twitter	1.00	253/285	9/10	0.99	**285/285**	**10/10**
Amazon	0.91	282/578	5/10	0.74	**366/578**	**9/10**
ER	1.00	**316/316**	**10/10**	0.86	159/316	3/10
WS	1.00	**298/316**	**9/10**	0.84	145/316	2/10

[8] Brandes U. On variants of shortest-path betweenness centrality and their generic computation. Social networks. 2008;30(2):136-45.

[9] Pfeffer J, Carley KM. k-centralities: Local approximations of global measures based on shortest paths. In: Proceedings of the 21st International Conference on World Wide Web; 2012. p. 1043-50.

[10] Kourtellis N, Alahakoon T, Simha R, Iamnitchi A, Tripathi R. Identifying high betweenness centrality nodes in large social networks. Social Network Analysis and Mining. 2013;3:899-914.

[11] Brandes U, Pich C. Centrality estimation in large networks. International Journal of Bifurcation and Chaos. 2007;17(07):2303-18.

[12] Geisberger R, Sanders P, Schultes D. Better approximation of betweenness centrality. In: 2008 Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX). SIAM; 2008. p. 90-100.

[13] Riondato M, Upfal E. Abra: Approximating betweenness centrality in static and dynamic graphs with rademacher averages. ACM Transactions on Knowledge Discovery from Data (TKDD). 2018;12(5):1-38.

[14] Cousins C, Wohlgemuth C, Riondato M. BAVARIAN: Betweenness centrality approximation with variance-aware Rademacher averages. ACM Transactions on Knowledge Discovery from Data. 2023;17(6):1-47.

[15] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms. MIT press; 2022.

[16] Staudt CL, Sazonovs A, Meyerhenke H. NetworKit: A tool suite for large-scale complex network analysis. Network Science. 2016;4(4):508-30.

[17] Leskovec J, Krevl A. SNAP Datasets: Stanford Large Network Dataset Collection; 2014. `http://snap.stanford.edu/data`.