Mongolian Journal of Engineering and Applied Sciences, 2023, 5(2)

Компьютерын ухаан

Графын чухал оройг илрүүлэх хувьслын алгоритм

Г.Гантулга[1], П.Батцэнгэл[1], П.Далайжаргал[1,*]

[1]МУИС, ХШУИС, Мэдээлэл Компьютерын Ухааны Тэнхим

Хүлээн авсан 2023.05.12; Хянагдсан 2023.10.19; Зөвшөөрөгдсөн 2023.10.19

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

Хураангуй

Энэхүү судалгааны ажлаар сүлжээ (граф) -ний чухал оройн бодлого (ЧОБ) -ыг хувьслын алгоритм (evolutionary algorithm (EA)) -аар бодох аргыг судална. ЧОБ нь сүлжээнээс хамгийн цөөн тооны оройг устган, үлдэгдэл сүлжээний хамгийн том холбоост бүрдлийн (ХТХБ) хэмжээг өгөгдсөн L параметрээс бага байлгах бодлого юм. Сүүлийн жилүүдэд хийгдэж буй өгөгдөлд суурилсан судалгаагаар байгаль, нийгэм дэх сүлжээ (систем) нь бүлэг бүтэц (community structure)-тэй гэдгийг тогтоогоод байна. Энэ ажилд сүлжээний бүлэг бүтцийн мэдээллийг ЧОБ-д ашиглах шинэ аргыг танилцууллаа. Тодруулбал, бүлэг бүтцийн мэдээллийг генийн дүрслэлээр (representation) ашиглах хувьслын алгоритмыг зохиомжлов. ЧОБ-д өргөн ашиглагддаг зургаан бодит сүлжээн дээр алгоритмын ажиллагааг туршиж, Greedy1, Greedy2, Genetic algorithm гэсэн гурван аргатай харьцуулан, дэвшүүлж буй алгоритмын гүйцэтгэлийг үнэллээ. Дэвшүүлж буй алгоритм том сүлжээн дээр 10-30 дахин богино хугацаанд, 2 өгөгдөл дээр давуу шийд гаргаж байгааг туршилтын үр дүн харуулав.

Түлхүүр үг: Бүлэг бүтэц, чухал оройн бодлого, комплекс сүлжээ, хоорондын төв, хувьсалын алгоритм

1 Удиртгал

Технологи, нийгэм, эдийн засгийн системүүдийг загварчлах үндсэн аргуудын нэг нь граф юм. Сүлжээний оройнууд өөр хоорондоо харилцан адилгүй үүрэг, зорилготой байдаг. Иймээс сүлжээнээс цөөн тооны чухал оройг илрүүлэх нь практик ач холбогдол өндөртэй. Теле-холбооны сүлжээг удирдах [3,4], байгалийн гамшигт үзэгдлийн үед тээврийн сүлжээний ачаалал даах чадварыг үнэлэх [5], цөлжилт явагдаж буй бүс нутгийн гол, мөрний сүлжээний дахин сэргэх чадамж (resilience)-ийг үнэлэх [6], террористын харилцаа холбооны сүлжээг задлах [7] гэх мэт практикийн асуудлыг ЧОБ-оор загварчлан бодох судалгаанууд хийгдэж байна.

Жингүй, чиглэлгүй граф $G(V,E)$ болон бүхэл тоо L өгөгдөхөд сүлжээний бүтцэд нөлөө өндөртэй хамгийн цөөн тооны чухал оройг илрүүлэх асуудлыг ЧОБ-оор судална. Тодруулбал, илрүүлсэн чухал оройг устгасны дараа үлдэх ХТХБ-ийн хэмжээ өгөгдсөн L-ээс бага байх хязгаарлалт хангах ёстой бөгөөд хамгийн цөөн тооны оройн олонлогийг хайж олох бодлого юм [8,9]. ЧОБ-ыг оновчлолын бодлогоор томьёолж болно (Томьёо 1, Томьёо 2), энд $f(S)$ нь зорилгын функц, S нь устгах оройн олонлог, C нь сүлжээнд үлдэх холбоост бүрдлийн олонлог, c_i нь i дугаар холбоост бүрдлийн хэмжээг тус тус тэмдэглэв. Цаашид графын оройн олонлог V, түүний элементийн тоо $V = n$, ирмэгийн олонлог E, түүний элементийн тоо $E = m$ гэж тус тус тэмдэглэнэ.

$$f(S) = \min\{|S|\} \tag{1}$$

$$s.t. : \{c_i \in C\} \leq L \tag{2}$$

Анх ЧОБ-ыг А Аруселван нар 2007 [3] онд танилцуулсан. Уг бодлого нь холболтын хэмжүүрээс хамааран хэд хэдэн хувилбартай, өөрөөр хэлбэл нийт хос орой хоорондын холболтын тоо [3,10,11], үлдэх ХТХБ-ийн хэмжээ [1,8], сүлжээний мэдээлэл тархахдаа өгөгдсөн зайнаас их зайд дамжихгүй байна гэсэн нэмэлт шаардлага бүхий [12,13] ЧОБ-ууд томьёологдсон байдаг. ЧОБ болон түүний хувилбарууд нь тооцооллын NP-hard ангилалд багтана [10,14,15]. Бодлогын хувилбарууд болон аргуудын талаар дэлгэрэнгүй мэдээлэл [16]-ээс үзэх боломжтой. Бид энэ ажлаар ХТХБ-ийн хэмжээг өгөгдсөн L тооноос бага болгох хувилбарыг судална. Байгаль, нийгэм дээрх ихэнх сүлжээ нь бүлэг бүтэц бүхий зохион байгуулалттай байдаг бөгөөд сүлжээний нэг бүлгийн гишүүд хоорондоо холбоо (ирмэг) ихтэй харин бусад гишүүдтэй холбоо багатай байна [17,18]. Зураг 1-д гурван бүлэг бүхий жижиг сүлжээг дүрслэв. Эхлээд бүлэг илрүүлэхэд өргөн хэрэглэгддэг стандарт алгоритмын нэг болох Гирван-Нүмэн [17] (Girvan-Newman (GN)) алгоритмыг авч үзье. GN нь ирмэгийн хоорондын төвийг [19–21] ашиглан хамгийн төвд орших нэг ирмэгийг устгаад, дахин хоорондын төв (ХТ)-ийг тооцоолж, өндөр оноотой ирмэгийг устгах байдлаар үргэлжилнэ. Эдгээр үйлдлийн дүнд бүлгүүд үүснэ [17].

Бид GN алгоритмын санааг өргөтгөн ЧОБ-д ашиглав. Өргөтгөлийн санааг гурав хуваан авч үзэж болно: (1) ХТ-ийн мэдээлэлд суурилан нэгэн зэрэг олон орой устгана (GN нь ХТ-ийн мэдээлэлд суурилан өндөр оноотой нэг ирмэгийг устгадаг [17] нь GN-г удаан ажиллах үндэс болдог), (2) ХТ-ийг тооцоолох нь $O(n \cdot m)$ хугацаа [19] шаарддаг тул ХТ-ийг $O(m \cdot log^2(n))$ хугацаанд ойролцоолох [20, 21] алгоритмыг ашиглах, (3) үр дүн бага оройнуудыг буцаан нэмэх зэрэг үйлдлүүд болно. Эхний хоёр өөрчлөлт нь нарийвчлалыг алдаж, хугацаа хожно. Алдсан нарийвчлалыг хугацааны алдагдал багатайгаар нөхөх зорилгоор зарим оройг буцаан нэмэв. GN алгоритмын өргөтгөлөөр илрүүлэх оройн олонлогийг сэжигтэй бүс (critical region), сэжигтэй бүсийг илрүүлэн сүлжээнээс устгах процедурыг сэжигтэй бүсийн устгах (critical region extraction) арга гэж тус тус нэрлэе. Сүлжээний сэжигтэй бүсийн устгах аргыг дараалуулан гүйцэтгэвэл тухайн сүлжээ задран жижиг бүрдэл хэсгүүдэд хуваагдана. ЧОБ-ийг бодох сэжигтэй бүсийг гений дүрслэлээр ашиглах хувьслын алгоритмыг дэвшүүлэв. Зургаан бодит сүлжээн дээр туршиж, алгоритмын гүйцэтгэлийг хэмжив. Өмнө хэвлэгдсэн гурван аргатай үр дүнг харьцуулан үнэлэв. Дэвшүүлж буй алгоритм нь богино хугацаанд сайн шийд гаргаж байгааг туршилтын үр дүн харуулав.

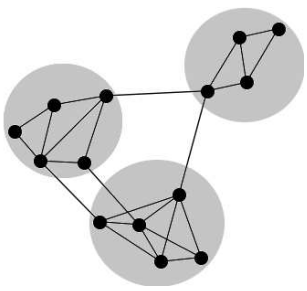
2 Бүлэг бүтэц дээр суурилсан хувьслын алгоритм

ХТ дээр суурилсан хувьслын алгоритм нь граф G, бодлогын хязгаарлалт L, сүргийн тоо N, *repair* операторт ашиглагдах β параметр, хугацаагаар хязгаарлах T_{max}, сайжраагүй олон үе явахад алгоритмыг дуусгах I_{max} параметрүүдийг авна (Алгоритм 1). Хувьслын алгоритмын генүүд нь графын оройнуудаас тогтно. Энэ ажлын нэг шинэлэг тал нь нэг ген нь оройн олонлог байх бөгөөд ихэвчлэн давхардсан тоогоор $\beta\sqrt{n}$ ширхэг оройг агуулах (сэжигтэй бүс) ба энэ тоо хувьсаж өөрчлөгддөг. Харин нэг бодьгалын агуулж буй хромосомын тоо $\frac{n-L}{\beta\sqrt{n}}$ утгаас эхлүүлсэн бөгөөд уг тоо хувьсан өөрчлөгдөнө. Сүргийг эхлүүлэхдээ графыг холбоост бүрдлийн хэмжээг $L/2$ хүртэл задлах сэжигтэй бүсүүдийг цуглуулан тэдгээрээс хромосомын уртаар санамсаргүйгээр сонгон шинэ бодьгалыг үүсгэнэ. Алгоритм 1-ийн 4-р мөрөнд байрлах үндсэн давтал нь I_{max} удаа сайжраагүй ажиллах эсвэл алгоритмын ажиллах хугацаа дууссан үед дуусна. Алгоритм 1-ийн 5-аас 9-р мөрөнд N ширхэг шинэ бодьгалыг агуулах шинэ үеийнхнийг, сүргээс санамсаргүй хоёр бодьгалыг сонгон авч тавь, тавин хувиар генийг нь холих замаар үүсгэнэ. Мөн 5 үе давталт хийгдсэний дараа шинэ үеийг гений өөрчлөлттэйгөөр үүсгэнэ. Үүний дараагаар шинэ үеийг агуулах S олонлогийн бодьгал бүрийн хувьд ХТ-ийн утгыг ашигласан *засварлах оператор* дуудагдах бөгөөд уг оператор нь тухайн бодьгалын генүүдэд байгаа оройнуудыг графаас устгаснаар үлдэгдэл графын хамгийн том бүрдэл хэсэг L-ээс том бол уг нөхцөлийг хангах генүүдийг сүлжээнээс олж нэмж өгөх үйлдлийг хийнэ (Алгоритм 2). Шинэ бодьгалын генүүдийг *засварлах оператор*-ээр зассаны дараа уг бодьгал хамгийн сайн шийдийг агуулж байх бол хамгийн сайн шийдээ шинэчлэн, сул давталтыг тоолох тоолуурыг 0 болгоно (12-15-р мөр). Ийнхүү бүх шинэ үеийнхэн дээр *засварлах оператор* хийгдсэний дараа сүрэгт нийлүүлэн, тэднээс сонгон N бодьгалаас тогтох сүргийг дахин байгуулна. Үүнийг *fitness_selection* үйлдлээр хийсэн болно (Алгоритм 1-ийн 14-р мөр). Уг үйлдэл нь Томьёо 3-аар бодьгалуудыг эрэмбэлж хамгийн эхний N бодьгалыг амьд үлдээх буюу сүрэгт сонгон авна.

$$\text{fitness}(P, s) = |s| \cdot (1 - \frac{2 \cdot \text{diff}(P, s)}{n}) \qquad (3)$$

Томьёо 3-т ашигласан $\text{diff}(P, s)$ үйлдэл нь s бодьгалын генд агуулагдах оройнууд, нийт сүргээс хэдэн оройгоор ялгаатай байгааг буцаах үйлдэл ба $|s|$ нь тухайн бодьгалын генд агуулагдах ялгаатай оройн тоо. Иймээс уг функц нь аль болох бусдаасаа ялгаатай оройнуудыг агуулсан болон шийдийн чанар сайтай оройнуудыг авч үлдэх болно.

Алгоритм 2-т харуулсан *засварлах оператор* нь бодьгалын генд байх бүх оройг G графаас утгасны дараа ХТХБ хэсэг нь L-ээс их байх үед граф дээр ХТ-ийн утгыг бодож хамгийн том ХТ утгатай $\beta\sqrt{n}$ ширхэг оройг олж нэмж устгах үйлдлийг давтан гүйцэтгэнэ. Харин давталтын нөхцөл биелэхгүй болох буюу графын хамгийн том холбоост бүрдлийн хэмжээ L-ээс бага болох үед нийт устгасан оройг агуулах S олонлогоос зарим нэг шаардлагагүй орой буюу хамгийн том холбоост бүрдлийн хэмжээг L-ээс их болгохгүй байх зарим оройнуудыг графт буцааж нэмж өгнө. Энэ үйлдэл нь шийдийн чанарыг сайжруулдаг [22]. Ингээд эцэст нь шинэ бодьгалын генийг S олонлогийн оройнуудаас тогтох генүүдээр солино. Уг засварлах үйлдэл хувьслын алгоритм хурдан нийлэхэд туслах боловч ХТ-ийг бодох үйлдлийн давталтаар дахин, дахин хийж байгаа нь алгоритмын ажиллах хугацааг удаашруулна. Гэхдээ ХТ-ийн утгыг яг оновчтой бодох шаардлагагүй тул



Зураг 1: Гурван бүлэгтэй сүлжээ.

Алгоритм 1 Дэвшүүлж буй хувьслын алгоритм (EA)

EA-algorithm$(G, L, \beta, N, T_{max}, I_{max})$
$P \leftarrow Initialize\_population(G, n)$;
$i\_cnt \leftarrow 0, s_{best} \leftarrow V$;
$idle \leftarrow 0, mutation\_interval \leftarrow 5$;
while $idle < I_{max}$ and $time < T_{max}$
  if $i\_cnt \ \% \ mutation\_interval = 0$
    $S \leftarrow mutation(P, N)$;
  else
    $S \leftarrow crossover(P, N)$;
  for $\forall offspring \in S$
    $repair(G, offspring, L, \beta)$;
    if $s_{best} > |offspring|$
      $s_{best} \leftarrow offspring$;
      $idle \leftarrow 0$;
  $P \leftarrow fitness\_sellection(P \cup S, N)$;
  $i\_cnt \leftarrow i\_cnt + 1$;
return s_{best};

Алгоритм 2 Засварлах оператор

repair$(G, offspring, L, \beta)$
$S \leftarrow \{\forall v \in V : v \in offspring\}$;
$G \leftarrow G(V/S, E)$;
while хамгийн том бүрдэл хэсэг$(G) > L$
  $S' \leftarrow \{$ хамгийн том ХТ утгатай n орой$\}$;
  $S \leftarrow S \cup S'$;
  $G \leftarrow G(V/S', E)$;
$S \leftarrow S/\{$шаардлагагүй зарим орой$\}$;
$offspring$-д агуулагдах генийг S-д агуулсан оройгоор нэм;

энд ойролцоогоор бодох [20] алгоритмыг ашиглан ажиллах хугацааг багасгасан.

3 Туршилт

Бид кодоо Networkit графын санг ашиглан Python 3 дээр хөгжүүлсэн. Туршилтыг Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz, 8GB санах ойтой энгийн суурин компьютер дээр хийсэн. Дэвшүүлж буй алгоритмаа CC-CNP бодлогыг бодоход тэргүүлэх алгоритмууд болох Greedy1 (G1), Greedy2 (G2) [1], Genetic Algorithm (GA)-уудтай [2] харьцуулж гүйцэтгэлийг нь хэмжсэн.

3.1 Өгөгдөл

Хүснэгт 1-т харуулсан өгөгдлүүд нь [2] дээр ашиглагдсан өгөгдлүүд бөгөөд техник, биологи, замын мэдээллээр байгуулсан бодит өгөгдлийн граф болно. Хүснэгтэд байгаа өгөгдлийн жижиг графаас томруу дараалаар нь жагсаасан. *Bovine* граф нь үхрийн төрөл зүйл дэх уураг хоорондын холбоо харилцааны граф болно. *USAir97* граф нь 1997 оны АНУ-ын нислэгийн сүлжээгээр байгуулсан граф болно. *HumanDisease* граф нь генетик өвчнүүдийн хоорондын хамаарлын граф. Граф *Openflighs* нь 2011 оны АНУ-ын нислэгийн сүлжээний граф болно. *Facebook* болон *Hepph* нь нийгмийн сүлжээний графууд бөгөөд харгалзан facebook олон нийтийн сүлжээний найзуудын граф, нөгөө нь өндөр энергийн үзэгдэл судалдаг физикийн салбарт ажилладаг эрдэмтдийн харилцааны граф болно. Хүснэгт 1-т эдгээр графуудын оройн тоо, ирмэгийн тоо, ЧОБ-ын хязгаарлалт, дундаж зэрэг, хамгийн том холбоост бүрдлийн хэмжээг харуулсан. Өгөгдөл тус бүр дээр алгоритмын гүйцэтгэлийг хэмжихдээ хүснэгтэд харуулсан бодлогын хязгаарлалт L-ийг ашигласан.

Хүснэгт 1: Туршилтын өгөгдөл. Багануud нь оройн тоо (n), ирмэгийн тоо (m), бодлогын хязгаарлалт (L), дундаж зэрэг (d), хамгийн том холбоост бүрдлийн хэмжээ ($|LCC|$).

Граф	n	m	L	d	$\|LCC\|$
Bovine	121	190	15	3.14	121
USAir97	332	2,126	70	12.81	332
HumanDisease	516	1,188	10	4.60	516
Openflight	1,858	13,900	140	14.96	1,485
Facebook	4,039	88,234	450	43.69	4,039
Hepph	12,008	118,489	3,600	19.74	11,204

3.2 Ажиллах хугацааны харьцуулалт

EA алгоритмын хугацааны гүйцэтгэлийг G1, G2 болон GA алгоритмуудтай харьцуулсан. Эдгээр алгоритмын үр дүнг [2]-аас авсан болно. Эдгээр туршилтууд нь бидний туршилт хийсэн орчноос өөр орчинд хийгдсэн тул ажиллах хугацааг харьцуулахдаа дараах зөрүүг авч үзэх нь зохимжтой. G1, G2, GA-ийн туршилтууд нь 2.1 GHz хурдтай хоёр AMD Opteron 8425HE процессортой, 16GB санах ойтой компьютер дээр C++ хэл дээр хөгжүүлж туршилтыг гүйцэтгэсэн. Хэдийгээр өөр өөр орчинд ажилласан эдгээр туршилтын үр дүнг харьцуулахад хэцүү боловч CPU-benchmark[1] үйлчилгээнээс харвал бидний хэрэглэсэн процессор 1.48 дахин хурдан гэж үзэж болно. Шударгаар харьцуулахын тулд Хүснэгт 2–т харуулсан эдгээр алгоритмын ажиллах хугацааг хоёр дахин багасган бидний алгоритмтай харьцуулж болно.

Алгоритмуудын ажилласан хугацааг секундээр хэмжин Хүснэгт 2-т харуулав. Хүснэгтээс үзэхэд G1, G2, GA алгоритмуудад яг адилхан хугацааны хязгаарлалт өгөгдсөн бол EA алгоритмд илүү бага хугацааны хязгаарлалт өгч ажиллуулсан. EA алгоритм нь бүлгийн бүтцийг ашиглан хувьслын алгоритмыг илүү хурдан нийлүүлэх тул богино хугацаанд ойролцоо шийд гаргах зорилготой. EA алгоритмыг *Hepph*-ээс бусад сүлжээн дээр 100 секунд, *Hepph* сүлжээн дээр 1000 секундийн хязгаарлалт тавьж ажиллуулсан.

[1] `https://cpu-benchmarks.com/`

3.3 Чанарын харьцуулалт

Шийдийн чанарын хувьд харьцуулж буй алгоритмуудын үзүүлсэн хамгийн сайн үзүүлэлтийг Хүснэгт 3-д харуулав. Энэ нь бодлогын хязгаарлалт L-д багтаахын тулд устгасан оройн хамгийн бага тоогоор харьцуулсан. Уг тоо бага байх тусам гүйцэтгэл сайн байна гэсэн үг. *Bovine* граф дээр бүх алгоритмууд адилхан үр дүнг гаргасан нь энэ үр дүн уг графын хувьд хамгийн сайн шийд гэдэг нь харагдаж байна. Харин *USAir97*, *HumanDisease* графууд дээр алгоритмууд ялгаатай үр дүнг үзүүлсэн байгаа боловч хамгийн сайн үр дүнг GA, EA алгоритмууд харуулсан байна. Харин *Openflight* сүлжээн дээр GA, EA алгоритмууд нөгөө хоёроосоо сайн шийд гаргасан бөгөөд GA алгоритм хамгийн сайн шийд харуулсан нь EA-ийн үзүүлэлтээс 3 оройгоор бага байна. *Facebook* граф дээр бүгд ялгаатай үр дүнг үзүүлсэн бөгөөд EA алгоритм G1, G2, GA-ийн үр дүнгээс харгалзан 176, 525, 28 оройгоор сайн үр дүн гаргалаа. EA-ийн гаргасан шийд нь бусад 3 алгоритмын хамгийн сайн нь болох GA-ийн шийдээс даруй 28 оройгоор бага байна. *Hepph* графын хувьд GA, EA алгоритмууд нөгөө хоёроосоо сайн шийдийг харуулсан бөгөөд хамгийн сайн шийдийг GA алгоритм гаргасан нь EA-ээс 22 оройгоор бага байна.

Бидний дэвшүүлж буй EA алгоритмын хувьд бүх өгөгдөл дээр богино хугацаанд, G1, G2 алгоритмуудаас сайн үр дүнг гаргах хандлага ажиглагдаж байна. Харин GA алгоритмтай харьцуулахад богино хугацаанд 3 өгөгдөл дээр адил хариу гаргаж, 1 өгөгдөл дээр (*Facebook*) 28 оройгоор бага үзүүлэлт, 2 өгөгдөл дээр харьцуулж болохуйц үр дүнг гаргасан байна. EA алгоритм нь бүлэг бүтэц нь илүү тодорхой графууд дээр сайн ажиллах хандлага туршилтаар батлагдаж байна. Зураг 2-т *Facebook*, *Hepph* графуудын бүлгийн бүтцийг харуулахын зорьсон бөгөөд *Facebook* граф дээр бүлгүүд нь илүү тодорхой харагдаж байна. Харин *Hepph* графын хувьд бүлгүүд харагдаж байгаа боловч нэг аварга том бүлгээс тогтож байгаа нь уг сүлжээг задлахад илүү хүндрэлтэй болж байна.

4 Дүгнэлт

Сүлжээний бүлэг бүтэц дээр суурилсан, богино хугацаанд ойролцоо шийд гаргах хувьслын алгоритм дэвшүүлж зургаан өгөгдөл дээр туршив. Дэвшүүлсэн EA алгоритм нь богино хугацаанд сайн үр дүнг өгч байна. Ялангуяа бүлэг бүтэц нь илүү тодорхой сүлжээнүүд дээр сайн гүйцэтгэлийг харуулав. Бүлэг бүтцийн мэдээлэлд суурилсан генийн дүрслэл нь хувьслын алгоритмын нийлэлтийн хурд төдийгүй хайлтын чадварыг сайжруулж байна. Цаашдаа өгөгдлийн тоо, төрөл, хэмжээг нэмэгдүүлэх, тэдгээр өгөгдөл дээр алгоритмыг туршиж параметрүүдийг тохируулах, сайжруулах зэрэг ажлуудыг гүйцэтгэнэ. Буцаан нэмэх үйлдлийн үнэлгээг нарийвчлах шаардлагатай.

Ашиг сонирхлын зөрчилгүйн баталгаа

Ашиг сонирхлын зөрчилгүй болохыг баталж байна.

Зохиогчийн оролцоо

Уг судалгааны дизайн, алгоритмын хөгжүүлэлтийг Г.Гантулга гүйцэтгэв. Туршилтын үр дүнг нэгтгэх, дүгнэх, өгүүллийн бичилтийг бүх зохиогчид хамтран гүйцэтгэв.

Ашигласан ном

[1] Addis B, Aringhieri R, Grosso A, Hosteins P. Hybrid constructive heuristics for the critical node problem. Annals of Operations Research. 2016;238:637–649.

[2] Aringhieri R, Grosso A, Hosteins P, Scatamacchia R. A general evolutionary framework for different classes of critical node problems. Engineering Applications of Artificial Intelligence. 2016;55:128–145.

[3] Arulselvan A, Commander CW, Pardalos PM, Shylo O. Managing network risk via critical node identification. Risk management in telecommunication networks, Springer. 2007:79–92.

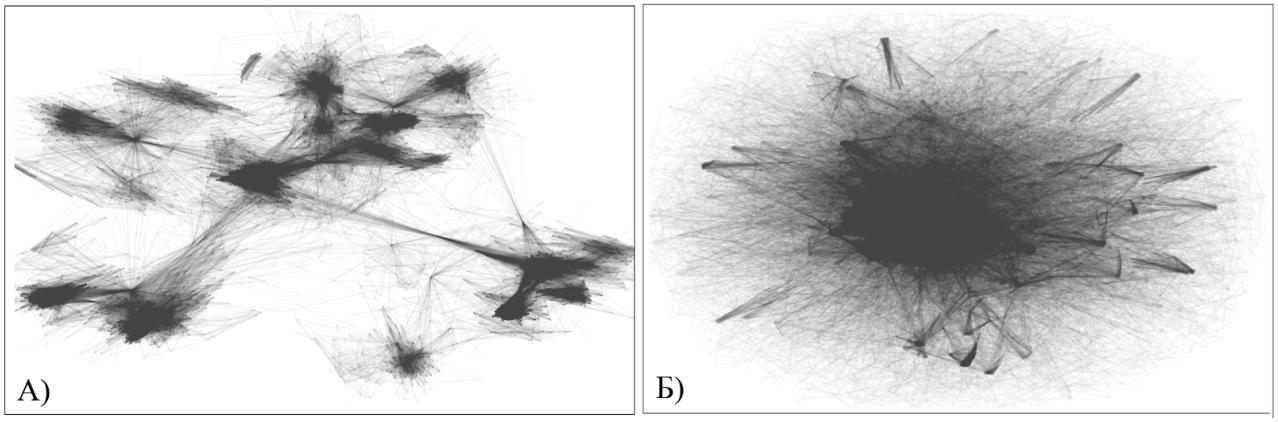
[4] Santos D, de Sousa A, Monteiro P. Compact models for critical node detection in

Хүснэгт 2: Ажилласан хугацааны харьцуулалт (секундээр)

Граф	G1	G2	GA	EA
Bovine	100	100	100	100
USAir97	300	300	300	100
HumanDisease	300	300	300	100
Openflight	4,000	4,000	4,000	100
Facebook	3,000	3,000	3,000	100
Hepph	10,000	10,000	10,000	1,000

Хүснэгт 3: Алгоритмуудын чанарын харьцуулалт

Граф	G1	G2	GA	EA
Bovine	4	4	4	4
USAir97	34	40	33	33
HumanDisease	51	50	49	49
Openflight	194	206	184	187
Facebook	472	821	324	296
Hepph	1,416	1,572	1,228	1,255



Зураг 2: Facebook (А), hepph (Б) графуудын бүлэг бүтэц.

telecommunication networks. Electronic Notes in Discrete Mathematics. 2018;64:325–334.

[5] Cantillo V, Macea LF, Jaller M. Assessing vulnerability of transportation networks for disaster response operations. Networks and Spatial Economics. 2019;19:243–273.

[6] Sarker S, Veremyev A, Boginski V, Singh A. Critical nodes in river networks. Scientific Reports. 2019;9(1):1–11.

[7] Tian L, Bashan A, Shi DN, Liu YY. Articulation points in complex networks. Nature communications. 2017;8(1):14223.

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

[9] Veremyev A, Boginski V, Pasiliao EL. Exact identification of critical nodes in sparse networks via new compact formulations. Optimization Letters. 2014;8:1245–1259.

[10] Arulselvan A, Commander CW, Elefteriadou L, Pardalos PM. Detecting critical nodes in sparse graphs. Computers & Operations Research. 2009;36(7):2193–2200.

[11] Aringhieri R, Grosso A, Hosteins P, Scatamacchia R. Local search metaheuristics for the critical node problem. Networks. 2016;67(3):209–221.

[12] Alozie GU, Arulselvan A, Akartunalı K, Pasiliao Jr EL. Efficient methods for the distance-based critical node detection problem in complex networks. Computers & Operations Research. 2021;131:105254.

[13] Salemi H, Buchanan A. Solving the distance-based critical node problem. INFORMS Journal on Computing. 2022;34(3):1309–1326.

[14] Gaarey M, Johanson D, Stockmayer L. Some simplified np-complete graph problem. Theoretical Computer Science;1(1976):237–267.

[15] Addis B, Di Summa M, Grosso A. Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth. Discrete Applied Mathematics. 2013;161(16-17):2349–2360.

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

[17] Girvan M, Newman ME. Community structure in social and biological networks. Proceedings of the national academy of sciences. 2002;99(12):7821–7826.

[18] Fortunato S, Newman ME. 20 years of network community detection. Nature Physics. 2022;18(8):848–850.

[19] Brandes U. A faster algorithm for betweenness centrality. Journal of mathematical sociology. 2001;25(2):163–177.

[20] 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.

[21] 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.

[22] Fan C, Zeng L, Feng Y, Xiu B, Huang J, Liu Z. Revisiting the power of reinsertion for optimal targets of network attack. Journal of Cloud Computing. 2020;9:1–13.