DOI https://doi.org/10.15407/csc.2022.01.024
УДК 517.7

А.В. АНІСІМОВ, доктор фіз.-мат. наук, професор, декан факультету комп'ютерних наук та кібернетики, Київський національний університет імені Тараса Шевченка
03022, м. Київ, просп. Академіка Глушкова, 4Д, Україна,
a.v.anisimov@knu.ua

ПРО ЗАДАЧУ ЛОКАЛЬНОЇ ОПТИМІЗАЦІЇ ГРАФІВ ТА ЇЇ ЗАСТОСУВАННЯ

Досліджується задача побудови локально-оптимального орієнтовного графа з навантаження ребрами, коли кожному ребру приписана фіксована числова вага. Локальна оптимальність або стабільний незмінний стан поточної розмітки графа дає загальний метод розв'язання багатьох задач, пов'язаних з оптимізаційним пошуком у графах. Показано як із розв'язання загальної задачі побудови локально-оптимального графа випливає задача знаходження шляхів найменшої ваги (вартості). Серед нових застосувань вказується вирішення проблеми асоціативного пошуку в комп'ютерній лінгвістиці (образне мислення) та швидкої взаємної автентифікації в коаліційних угрупуваннях.

Ключові слова: *орієнтовний граф, шлях найменшої вартості, асоціативний пошук, випадковість, швидка автентифікація.*

Вступ

Орієнтовний граф $G=(V,E)$ складається зі скінченної множини вершин V та упорядкованих пар вершин E, що називаються ребрами, $E\subset V\times V$. Пара (*w, v*) інтерпретується, як ребро (дуга), що веде із *w* до *v*. Кожному ребру (*w, v*) приписано постійну вагу t_{wv} — дійсне невід'ємне число. Задача пошуку шляху найменшої вартості, що веде із заданих фіксованої вершини v_0 та вершини v полягає у знаходженні послідовності таких вершин $v_1, v_2, ..., v_{k-1}$, що шлях $\pi = v_0, v_1, ..., v_{k-1}, v_k = v, (v_i, v_{i+1}) \in E$ дає мінімальне можливе значення вагової суми $\sum t_{v_i v_{i+1}}$, $i = 0 \ldots k-1$.

У зв'язку із широкою інтерпретацією числових ваг ребер задача пошуку шляхів найменшої вартості має безліч застосувань і належить до найпоширеніших і найбільш вивчених задач прикладної теорії алгоритмів. У різноманітних модифікаціях вона зустрічається в алгоритмах побудови оптимальних маршрутів для літаючих та інших рухомих об'єктів, у розпізнаванні образів, оптимізації комунікаційних мереж та інтегральних схем, тощо. Усе це зумовлює непослабний інтерес до цієї задачі. Найінтенсивніші дослідження в цьому напрямі припадають на другу половину минулого століття. Загалом описано декілька сотень різноманітних алгоритмів розв'язання цієї проблеми. Найвідомішими є алгоритми Е. Дейкстри [1], Д'Есопо-Пейпа-Мура та декілька паралельних алгоритмів [2–6].

Ми розглядаємо цю задачу в ширшому контексті як загальну задачу локальної оптимізації графа. Вивчається така модель обчислень на графі. У кожній локальній вершині незалежно від інших вершин виконується однакова ітеративна процедура мінімізації деякого невід'ємного числового значення, що приписується цій вершині. Операція такої мінімізації залежить від поточного значення, отриманого в цій верши-

ні при застосуванні попередньої мінімізації та поточних числових даних, що розміщені у вершинах, з яких ведуть ребра до обраної локальної вершини, а також мінімізації ваг цих ребер. Через декілька етапів виконання такої операції граф завжди переходить у стабільний стан: подальша мінімізація вершин не змінює приписаних вершинам значень. У такому стані ми називаємо граф *локально-оптимальним*. Досліджено умови, за яких граф досягає стану локальної оптимізації. При різних виборах інтерпретації функції, що задає вплив сусідніх вершин на значення мінімізації, отримуємо різноманітні варіації застосувань. Наприклад, методом градієнтного спуску зі стану локальної оптимізації легко отримати шляхи мінімальної ваги (вартості) із заданої стартової вершини до всіх вершин. Якщо граф інтерпретувати як семантичну мережу в просторі слів природної мови, то показано, у який спосіб задача пошуку мінімальних шляхів між словами або словосполученнями інтерпретується як розв'язання відомої задачі з комп'ютерної лінгвістики: знайти слово, яке є найбільш асоційованим із заданим вхідним набором слів або речень (образів). Нарешті, через обчислення ваг шляхів у рандомізованому (випадковому з рівномірним розподілом) графі пропонується протокол швидкої автентифікації типу «*свій-чужий*» двох суб'єктів коаліційного середовища, що перебувають у комунікації.

Локальна оптимізація. Модель обчислень

Нехай $G=(V,E)$ — скінчений орієнтовний граф, V — множина вершин, E — множина ребер, $E\subset V\times V$. Якщо $(u, v)\in E$, то це інтерпретується як орієнтовне ребро, що веде від u до v.

Шлях π від вершини u до v: $\pi=u=v_0,v_1,...,v_k=v,(v_i,v_{i+1})\in E$. Вага π шляху $w(\pi)=\sum t_{v_iv_{i+1}}$, дорівнює сумі ваг всіх відповідних ребер вздовж цього шляху.

Окіл вершини v — це вершини, суміжні з v разом із відповідними ребрами. Для околу вершини v використовуємо позначення $R(v)$. До $R(v)$ входять вершини, що з'єднуються з v вхідними та вихідними ребрами. Тому, відповідно, $R^+(v)=\{(w,(w,v))\mid(w,v)\in E\}$ — множина вершин w, що з'єднується з v вхідними ребрами, $R^-(v)=\{(w,(v,w))\mid(v,w)\in E\}$.

Із кожною вершиною v асоціюємо алгоритм A_v, який виробляє числові значення $D[v]$. Для зручності викладу вважаємо, що $D[v]$ є невід'ємним дійсним числом. Але в деяких застосуваннях $D[v]$ може бути елементом іншої алгебраїчної структури. Вершини v разом з відповідними значеннями $D[v]$ розглядаємо як розмітку μ вершин графа $G,\mu: V\to\mathcal{D}$, де $\mathcal{D}$ позначає множину дійсних чисел. Розмічений граф позначаємо (G,μ).

Позначмо через $+\infty$ спеціальне велике число, що є більшим від будь-якого можливого значення $D[v]$. Вважаймо, що це число є відомим заздалегідь (*apriori*).

Початкова розмітка є такою: $D[v_0]\leftarrow 0$; $D[v]\leftarrow+\infty$ для всіх $v\in V-v_0$. Вершина v_0 є особливо виділеною вершиною графа G. Із цієї вершини починаються всі шляхи в G.

Числовим околом вершини v називаємо множину пар $D^+(v)=\{(D[w],t_{wv})\mid w\in R^+(w)\}$. Ця множина інтерпретується як множина значень, що локально «бачить» алгоритм A_v.

Нехай f_v є функцією, що визначена на множині $D^+(v)$ та приймає невід'ємні дійсні значення. Функція f_v є асоційованою з вершиною v та залежить від локальної числової інформації околу $D^+(v)$ вершини v.

У кожній вершині v виконується простий алгоритм A, який ітеративно виконує таку операцію мінімізації: $D[v]\leftarrow\min\left(D[v],f_V\left(D^+(v)\right)\right)$.

Алгоритм А

Алгоритм A працює нескінченно довго: немає обмежувального параметра умови вико-нання циклу *DO*. Виконання поточної операції мінімізації в тілі циклу називаємо *раундом*.

DO
$D[v]\leftarrow\min\left(D[v],f_V\left(D^+(v)\right)\right)$
END DO

Для запобігання ситуацій взаємного блокування (*deadlock*) вважаймо, що функція при звернені до вершини $w \in R^+(v)$ отримує значення $D[w]$, яке w отримала після виконання свого останнього раунду мінімізації. Це означає, що якщо w виконує свою i-ту локальну мінімізацію в момент обчислення функції f_v, то f_v враховує значення $D[w]$ після виконання цієї i-ї локальної мінімізації w. У програмній реалізації такі ситуації обробляються стандартними методами подолання ситуацій взаємного блокування.

Вважаймо, що виконання операції мінімізації виконується в дискретному часі й може займати декілька одиниць такого часу. Але постулюймо такі обмеження на функцію f_v та виконання алгоритму A:

1) *якщо в околі* $R^+(v)$ *знайдеться вершина* w *така, що* $D[w] \neq +\infty$ $(D[w] < +\infty)$, *то* $f_v(D^+(v)) < +\infty$, *інакше* $f_v(D^+(v)) = +\infty$.

2) *в алгоритмі A для кожної вершини завжди через скінченний проміжок часу виконується звернення до наступного раунду мінімізації.*

Якщо всі функції f_v є однаковими, то для f_v використовуємо позначення f. Надалі ми розглядатимемо тільки такий випадок.

Локальна оптимальність

Алгоритмі A змінює розмітку графа G. Розмічений граф (G, μ) називаємо *локально-оптимальним відносно функції f*, якщо не існує вершини v, $v \in V$, для якої операція мінімізації зменшує поточне значення $D[v]$.

Локальна оптимальність — це властивість розмітки графа. Аналогом локальної оптимальності є відоме математичне поняття нерухомої точки.

Лема 1. Якщо у графі G не існує шляху від вершини v_0 до v, то алгоритм A в вершині v у будь-який раунд видає значення $D[v] = +\infty$.

Доведення. Використаємо індукцію за кількістю вершин $n = |V|$ графа G. Якщо n=2, то твердження є очевидним. Припустімо, що твердження леми виконується для графів із n вершинами і граф G має $n+1$ вершину.

Нехай v — вершина, яка є недосяжною в G із v_0. Припустімо зворотне: алгоритм A в якийсь раунд мінімізації вершини v отримав значення $D[v] \neq +\infty$. Тоді згідно з правилом моделі обчислення 1) в околі $R^+(v)$ має існувати вершина w із поточним значенням $D(w) \neq +\infty$. Розгляньмо граф G_v, який будується за G відкиданням вершини v разом з усіма її ребрами. Граф G_v має n вершин, $v_0 \in G_v$ і $w \in G_v$. У графі G_v немає шляху із v_0 до w, бо якщо би він був, то цей шлях у графі G можна було би продовжити до v. Тому за припущенням індукції у графі G_v, $D[w] = +\infty$. Але тоді й у графі G значення $D[w] = +\infty$ тому що v не входить у + окіл w. Отримуємо протиріччя.

Теорема 1. У графі G існує шлях від v_0 до v тоді й тільки тоді, коли алгоритм A в вершині v у деякому раунді видає $D[v] \neq \infty$.

Доведення. *Необхідність*. Припустімо, що в G існує шлях $v_0, v_1, \ldots v_k = v$, $(v_i, v_{i+1}) \in E$, $i = 0 \ldots k-1$. Тоді внаслідок властивості функції f та правила 2) алгоритму A всі вершини шляху $v_0, v_1, \ldots v_k$ через скінчений проміжок часу мають отримати значення, відмінне від + ∞.

Достатність. Припустімо, що $D[v] \neq +\infty$.

Тоді внаслідок леми 1 в G існує шлях від v_0 до v і, як було показано в доведенні необхідності, в деякому раунді це дає $D[v] \neq \infty$.

Теорема 2. Якщо граф G є ациклічним скінченним графом, то алгоритм A через скінченний час приводить G у стан локальної оптимальності.

Доведення. Використовуємо індукцію щодо кількості вершин n. Якщо n=1 або n=2. Твердження теореми очевидно.

Припустімо, що твердження теореми 2 є вірним для ациклічних графів із n вершинами і граф G має n+1 вершину. Тоді оберімо довільну вершину v, $v \neq v_0$.

Якщо із v_0 не існує шляху до v, то, як показано в теоремі 1, після виконання будь-якої кількості раундів мінімізації $D[v]$ залишається незмінним, $D[v] = +\infty$.

Припустімо, що існує шлях від v_0 до v. Тоді згідно з теоремою 1 після деякого раунду $D[v] \neq +\infty$. Розгляньмо граф G_v, який було

використано під час доведення леми 1. Цей граф будується по G відкиданням вершини v і всіх суміжних ребер. Згідно з припущенням індукції, алгоритм A, що працює у вершинах звуженого графа $G_{\overline{v}}$, переводить цей граф у стан локальної оптимізації. Вершини в околі v також отримують незмінне стабільне значення в $G_{\overline{v}}$. Нехай $w \in R^+(v)$ і $D[v]$ і її незмінне локально-мінімальне значення в $G_{\overline{v}}$.

Якщо $D[w] = +\infty$, то це значення буде постійним у будь-якому раунді виконання алгоритму A. Якщо $D[w] \neq +\infty$, то, за припущенням індукції, це значення буде мінімальним можливим у $G_{\overline{v}}$. Згідно з твердженням теореми 1, це значення отримано завдяки оптимізації вершин, що утворюють шлях від v_0 до w у графі $G_{\overline{v}}$. Нехай цей шлях буде $v_0, v_1, \ldots v_k = w$. У графі G вершина v не має впливу на значення мінімізації цих вершин, тому що граф G є ациклічним, а отже не існує шляху від v до жодної із вершин $v_0, v_1, \ldots v_k$. Таким чином, операція мінімізації v після того, як усі вершини w в околі $R^+(v)$ стабілізують свої стани, приведе до стабільного незмінного стану вершини v.

Зауважмо, що, якщо прибрати припущення ациклічності графа, то, враховуючи, що не робиться жодних спеціальних припущень стосовно функції f, можливим є варіант входження в цикл вершини, що нескінченно зменшує значення $D[v]$.

Застосування методу локальної оптимізації

Найкоротші шляхи. Розгляньмо класичну задачу знаходження шляхів найменшої ваги від заданої вершини v_0 до всіх вершин графа G. В цьому випадку функція зміни розмітки вершин має найпростіший вигляд:

$$D[v] \leftarrow \min\left(D[v], \min_{w \in R^+(v)} \left(D[w] + t_{wv} \right) \right).$$

У кожній вершині v діє один і той самий алгоритм A:

Алгоритм A

DO

$$D[v] \leftarrow \min\left(D[v], \min_{w \in R^+(v)} \left(D[w] + t_{wv} \right) \right).$$

END DO

Безпосередня інтерпретація теореми 2 дає такий факт:

Теорема 3. Якщо при виконанні алгоритму A $D[v] \neq +\infty$, то $D[v]$ задає вагу деякого шляху від v_0 до v. Якщо вершина v не є досяжною із v_0, то в усіх раундах $D[v] = +\infty$.

Зауважмо таке. Вага кожного ребра задається невід'ємним числом. Тому найменша вага шляху від v_0 до v досягається шляхом, що немає циклів. Тому інтерпретація теореми 2 в цьому разі буде такою.

Теорема 4. При застосуванні алгоритму A до графа G через скінчений час той переходить у стан локальної оптимальності. У стані локальної оптимальності $D[v]$ для всіх вершин дорівнює шляху мінімальної ваги від v_0 до v.

Розгляньмо випадок, коли алгоритм A виконується синхронно в дискретному часі.

Наслідок 1. Час роботи синхронного алгоритму A не перебільшує n тактів, де n — кількість вершин графа G.

Доведення. Довжина будь-якого шляху без циклів не перебільшує n. Тому в синхронному варіанті час роботи алгоритму A до стану локальної оптимальності не перебільшує n.

Уточнюючи вибір вершин w і v в операції мінімізації (це задається застосуванням відповідної евристики), отримуємо всі відомі послідовні та паралельні алгоритми знаходження шляхів мінімальної ваги. Наприклад відомий послідовний алгоритм Е. Дейкстри [1] має такий вигляд:

Алгоритм D (Евристика Дейкстри)

$S \leftarrow \varnothing$;

WHILE $S \neq V$ *DO*

{

вибрати $w \in V - S$ з найменшим значенням $D[w]$;

$S \leftarrow S + w$;

FOR $v \in V - S$ *DO*

$$D[v] \leftarrow \min_{w \in R^+(v)} \left(D[v], D[w] + t_{wv} \right)$$

}

END DO.

В аналогічний спосіб отримуємо інші відомі

алгоритми. Детально це описано в роботі [7].

Образне мислення. Пошук асоціативно зв'язних слів. Задача є такою: вершини графа — слова або словосполучення; ребро від одного слова до іншого відображає безпосередній близький асоціативний зв'язок між словами. Наприклад, асоціативно близькими є такі пари слів: (*квітка, троянда*), (*квітка, кульбаба*), (*хмара, дощ*), (*вода, струмок*), (*метелик, гілка*) тощо. Ребру приписано числову вагу, яка є *обернено пропорційною* силі асоціативного зв'язку. Множину слів мови разом із безпосередніми асоціативними зв'язками можна розглядати як мережеву структуру — граф із заданою вагою ребер.

Фіксується вхідний набір слів або частин речень. Треба знайти слово, яке має найбільший асоціативний зв'язок із вхідним набором слів.

А.Г. Івахненко описує декілька експериментів подібного типу [8, с. 59–62]. Наприклад, автор наводить такий тест: Людині надавався текст із невідомим словом-поняттям. У [8] це невідоме поняття позначено як [–]. Треба знайти [–]. Наведімо один із прикладів.

«*Не витрачай свій час на [–]. У нього були хороші [–], а мені не везло*».

Він взяв зі столу [–] і попросив назвати ще одну.

Мене попереджали, щоб я не грав з ним в [–], тому що він махлював».

Досить швидко випробуваний знаходив: [–] = «*карти*»

Подібні завдання мозок людини виконує доволі швидко. Тому алгоритмізація розв'язання подібних задач дає змогу моделювати образне мислення людини й може бути основою систем штучного інтелекту.

У графічній інтерпретації це завдання зводиться до пошуку в семантичному графі визначених мовою асоціативних зв'язків слова або словосполучення, яке має найменшу агрегованну відстань до заданого вхідного набору слів. Агрегування — це деяка функція від множини *відстаней найменшої ваги* до заданого вхідного набору слів. Наприклад, це може бути сума мінімальних відстаней від слова-рішення до вхідного набору.

Інший варіант пошуку найкоротших шляхів в асоціативних мережах було використано в програмі автоматичної генерації віршів. Детально це описано в [9]. Основна ідея алгоритму є такою. Генерується перша фраза — послідовність асоціативно пов'язаних слів. У словнику рим шукається слово-рима до останнього слова першої фрази. Потім від цього слова-рими в графі асоціативних зв'язків шукається мінімальний шлях до останнього слова першої фрази. Вибудовувалася непогана заготовка для комп'ютерного створення віршів. Такі експерименти було здійснено І. Івановою, на той час аспіранткою факультету кібернетики КНУ імені Тараса Шевченка. Зокрема було згенеровано такий фрагмент

«*Рижі жоржини. Огонь. Ад. Гріх.*

Героїн. Героїня. Убивство. Крик. Сміх.

.....»

Тут спочатку рандомізовано формується перша послідовність асоціативно зв'язаних слів. Потім у словнику рим шукається рима до слова «*гріх*». У цьому разі це слово «*сміх*». Обрані слова є асоціативно віддаленими, тобто в універсальному графі асоціативних зв'язків немає ребра, що пов'язує ці два слова. У зворотному порядку шукаємо найкоротший шлях від слова «*сміх*» до слова «*гріх*». Одним із варіантів є вказана друга послідовність.

Скоріш за все, асоціативне мислення людини (образне мислення) побудовано за принципом знаходження оптимальних шляхів у просторі «*слів-понять*». Тому за використання алгоритмів пошуку найкоротших шляхів з'являється можливість побудови системи штучного інтелекту, яка реалізує такий принцип. Простір «*слів-понять*» мови будується за онтологічним принципом.

Аналогічна ідея може бути використана для розв'язання задачі розпізнавання. Тут окіл вершини задає основні ознаки поняття. Ними є вхідні вершини-поняття. Треба знайти вершину-поняття, яка має найкоротший шлях до заданих вхідних вершин.

Автентифікація. Проблема автентифікації суб'єктів, які перебувають у комунікації, має

чимало різноманітних ситуативних особливостей. Універсальна автентифікація за допомогою цифрових підписів потребує збільшення довжини та часу формування повідомлення. Популярні сучасні методи на основі доведень знання *NP*-властивостей із нульовим розголошенням потребують значних обсягів пам'яті для опису *NP*-повних проблем. У багатьох ситуаціях виникає необхідність надшвидкої автентифікації. Ця задача є особливо актуальною для об'єктів критичної інфраструктури та різноманітних коаліційних об'єднань, що включає військові, фінансові, людино-машинні, об'єкти Інтернету речей та інші подібні угруповання. Огляд подано в [10, 11].

Розгляньмо задачу автентифікації типу «*свій-чужий*» у коаліційних об'єднаннях. Коаліція потребує наявності довірчої третьої сторони *T*, яка формує певні таємні дані для суб'єктів *A* і *B*, які перебувають у комунікації. Це можуть бути ідентифікаційні параметри або вміння швидко розв'язувати складну задачу. Якщо *A* і *B* роздільно володіють якимось спільним секретом (числом) *S*, то задача легко розв'язується за допомогою обмінів значеннями загальновідомої геш-функції *h*.

Розгляньмо такий протокол.

Умова: *A* і *B* повинні володіти однаковим числом *S*. Фактично *A* володіє числом S_A, *B* володіє числом S_B, *h* — загальновідома геш-функція. *A* і *B* доводять один одному, що вони володіють однаковим числом $S_A = S_B = S$.

Протокол H

1. $B \to A : x$ — випадкове число;
2. $A \to B : \; y_A = h(x + S_A)$;
3. *B* приймає *A*, якщо $y_A = h(x + S_B)$.

Це еквівалентно рівності $S_A = S_B = S$.

З метою взаємної автентифікації цей протокол доповнюється аналогічним повідомленням від *B* до *A*.

4. $B \to A : y_B = h(S_B + y_A)$
5. *A* приймає *B*, якщо $h(S_A + y_A) = y_B$.

Це еквівалентно рівності $S_A = S_B = S$.

Неважко строго довести, що коли б хоч одна із сторін *A* або *B* «*є чесною*» (тобто $S_A = S$ або $S_B = S$), то чесна сторона завжди сприймає чесного партнера й не сприймає нечесного.

Але володіння двома сторонами однаковим секретом у середовищах, що можуть зазнавати атак зловмисників, може бути слабкою стороною під кутом зору безпеки. Динамічне створення спільного секрету стандартними методами «рукостискання» (*handshaking*) типу Діффі-Геллмана або Ель-Гамаля є вразливим до атаки «людина посередині». Тому ставимо умову: *A і B мають володіти різними секретами. Окрім того, вони обидва не повинні знати нічого про секрети другої сторони*. Це має забезпечити довірча третя сторона *T*. При цьому автентифікація має виконуватися в реальному часі без участі довірчої сторони. Розгляньмо такий варіант обмінів між *A* і *B*, використовуючи шляхи у випадкових графах.

Вважаймо, що граф $G = (V, E)$ має $n+1$ вершин, асоційованих із номерами $0,1,...,n$. Із графом *G* однозначно пов'язано дві матриці $A_V = \|e_{ij}\|$, $A_T = \|t_{ij}\|$ розміром $(n+1)\times(n+1)$. A_V — матриця задає структуру графа: $e_{ij} = 1$, якщо із *t* веде ребро до *j*, інакше $e_{ij} = 0$. A_T— матриця ваг ребер, $t_{i,j}$ — вага ребра (*i, j*). Припускаємо наявність циклів, тобто для деяких індексів $t_{ii} \neq 0$, $e_{ii} = 1$.

Нехай $R = \|\tau_{ij}\|$ є матрицею випадкових цілих чисел, $\tau_{ij} \geq 0$. За допомогою матриці A_V *R* визначає випадкові значення ваг A_T, $t_{ij} = \tau_{ij}$, якщо $t_{ii} \neq 0$, $e_{ii} = 1$.

Довірча сторона *T* формує матрицю *R* і передає *R* до *A*.

Нехай *E* — група точок еліптичної кривої над скінченним полем F_P. *P* — точка цієї кривої. Довірча сторона *T* обчислює матрицю точок $R_E = \|\tau_{ij} P\|$ і передає (R_E, P) до *B*.

Вважаємо, що *T* після відповідних передач *R* та R_E вже не зберігає *R* та R_E.

Після комунікацій із *T A* і *B* функціонують незалежно один від одного. Якщо виникає необхідність у *B* автентифікувати *A*, то *B* формує випадковий граф *G* і задає випадковий шлях в цьому графі. Передає цю інформацію до *A*. Своєю чергою у відповідь *A* має обчислити вагу шляху π і передати це значення $w(\pi)$ до *B*. *A* обчислює цю вагу використовуючи матриці $A_V = \|e_{ij}\|$ та $A_T = \|t_{ij}\|$. *B* обчислює значення

точки еліптичної кривої вздовж шляху π, використовуючи матрицю R_E. Нехай це буде значення $w(\pi, P)$. Якщо $w(\pi)P = w(\pi, P)$, то B приймає A, інакше «НІ».

Протокол *Authentication*

1. B формує випадкову матрицю A_V, що задає структуру графа без кратних ребер (цикли допустимі). B також задає випадковий шлях $\pi = \langle i_0 = 0,\ i_1, i_2, ..., i_k = n \rangle$ від 0 до n $B \to A : (A_V, \pi)$ і передає цю інформацію до A.

2. A обчислює вагу шляху π. Для цього A використовує матрицю A_T і здійснює сумування ребер вздовж π. $A \to B : w(\pi)$.

3. B обчислює суму точок $\sum t_{i_j i_{j+1}} P = w(\pi, P)$, $j = 0, ..., k-1$, вздовж π.

B приймає A, якщо $w(\pi, P) = w(\pi)P$.

Безпека протоколу *Authentication*. Для точок еліптичної кривої проблема дискретного логарифму вважається обчислювально неможливою. Тому B не знає матрицю ваг $R = \|\tau_{ij}\|$. A не знає навіть точки P та рівняння кривої E. Отже, A і B володіють різними незалежними секретами. Вага шляху є випадковим числом. Тому протокол є стійким до атаки «людина посередині».

Висновки

Задача побудови найкоротших шляхів має багато різноманітних застосувань. У представленій роботі дається алгоритмічне узагальнення цієї задачі. Це дає змогу в уніфікований спосіб отримати всі відомі алгоритми її розв'язння та знайти нові застосування. Зокрема, дано зведення моделювання задачі з образного асоціативного мислення людини до алгоритму побудови найкоротших шляхів. Також до подібних задач пошуку вагової маршрутизації зведено відому криптографічну задачу швидкої

ЛІТЕРАТУРА

1. *Deikstra E.* A note on two problems in connexion with graphs. Numer. Math. 1959. № 1, pp. 269–271.
2. *Moore E.F.* The shortest path through a maze. In Proc. an International Symposium on the Theory of Switching, Part II, Harvard University Press, 1959, pp. 285–292.
3. *Po11ack M.* Solutions of the shortest-rout problems - a review. Oper. Res. 1960, N 8,. pp. 224–230.
4. *Pape U.* Implementation and efficiency of Moore-algorithms for the shortest route problems. Math. Progr, 1974, N 7, pp. 212–2.
5. *Deo N., Pang C. Y., Lord R. E.* Two parallel algorithms for short est path problems. Proc. intern, conf. parallel proc. NewYork: ACM, 1980, pp. 244–53.
6. *Иванов Е. А., Шевченко В. П.* О параллельных вычислениях на графах. Кибернетика. 1934, № 3. С. 89–94.
7. *Анисимов А. В.* Локальный алгоритм для задачи о кратчайшем пути из одного источника. Кибернетика. 1986. №3. С. 57–60
8. *Ивахненко А. Г.* Системы эвристической самоорганизации в технической кибернетике. «Техника», 1971. 371 с.
9. *Анисимов А. В.* Информатика. Творчество. Рекурсия. Киев: Наук. думка, 1988. 224 с.
10. *Li ,N., D. Liu, and S. Nepal* Lightweight mutual authentication for IoT and its applications. IEEE Trans. Sustainable Computing. 2017.2(4), pp.359–370.
11. *Yang T. , G. Zhang, L. Liu, Y. Yang, S. Zhao, H. Sun, and W. Wang* New features of authentication scheme for the IoT: A Survey. In Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things (IoT S&P'19). New York. USA. 2019, pp. 44–49.

Надійшла 06.06.2022

REFERENCES

1. *Deikstra, E.*, 1959. "A note on two problems in connexion with graphs". Numer. Math., N1, pp. 269–271.
2. *Moore, E.F.*, 1959. "The shortest path through a maze". In Proc. an International Symposium on the Theory of Switching, Part II, Harvard University Press, pp. 285–292.
3. *Pollack, M.*, 1960. "Solutions of the shortest-rout problems - a review". Oper. Res., N 8, pp. 224–230.
4. *Pape, U.*, 1974. "Implementation and efficiency of Moore-algorithms for the shortest route problems". Math. Progr, N7, pp. 212–2.
5. *Deo, N., Pang, C.Y., Lord, R.E.*, 1980. "Two parallel algorithms for short est path problems". Proc. intern, conf. parallel proc. NewYork: ACM, pp. 244–253.

6. *Ivanov, Ye.A., Shevchenko, V.P.*, 1934. "On parallel computing on graphs". Cybernetics, N 3, pp. 89–94 (In Russia).
7. *Anisimov, A.V.*, 1986. "Local algorithm for the problem of the shortest path from one source". Cybernetics. 1986. N 3, pp. 57–60 (In Russia).
8. *Ivakhnenko, A.G.*, 1971. Systems of heuristic self-organization in technical cybernetics. K.: Tekhnika, 371 p. (In Russia).
9. *Anisimov, A.V.*, 1988. Informatics. Creation. Recursion. Kyiv: Nauk. dumka, 1988. 224 p.
10. *Li,N., D. Liu, and S. Nepal*, 2017."Lightweight mutual authentication for IoT and its applications". IEEE Trans. Sustainable Computing. 2(4), 359–370.
11. *Yang T. , G. Zhang, L. Liu, Y. Yang, S. Zhao, H. Sun, and W. Wang*, 2019. "New features of authentication scheme for the IoT: A Survey". In Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things (IoT S&P'19). New York. USA, pp. 44–49.

Received 06.06.2022

A.V. Anisimov, Doctor of Physical and Mathematical Sciences, Professor, Department of Information Systems, dean of the faculty of Computer Science and Cybernetics, Taras Shevchenko National University of Kyiv, Glushkov ave. 4D, 03022, Kyiv, Ukraine, a.v.anisimov@knu.ua

THE PROBLEM OF LOCAL OPTIMIZATION OF GRAPHS AND ITS APPLICATIONS

Introduction. We study the problem of constructing a locally optimal directed graph when each edge is assigned a fixed numerical weight. Due to the broad interpretation of the numerical weights of edges, the problem of finding the least-cost paths has many applications and is one of the most common and studied problems in the applied theory of algorithms. In various modifications, it can be found in algorithms for constructing optimal routes for flying and other moving objects, in pattern recognition, optimization of communication networks and integrated circuits. The property of local optimality is defined through a dynamic process that at every round assigns to each vertex the minimal value depending on its current value and sums of values stored in neighborhood vertices with weights of incoming edges. The graph is locally optimal when any vertex cannot allow further minimizations.

Purpose. The purpose of the article is to investigate the problem of constructing a locally-optimal graph, establish properties of such graphs, and derive new applications.

Methods. We consider the problem of finding the shortest paths in a broader context as a general computational problem of local graph optimization. This model of computations on a graph is studied. Each local vertex, regardless of the other vertices, performs the same iterative procedure of minimizing some non-negative numerical value assigned to that vertex. The operation of such minimization depends on the current value obtained in this vertex when applying the previous minimization and sums of the current numerical data located in the vertices from which the input edges lead to the selected local vertex and weights of the seedges. After several stages of such an operation, the graph always comes into a stable state: further minimization of vertices does not change the values assigned to the vertices. In this state, we call the graph locally optimal.

Results. The conditions under which the graph reaches the state of local optimization are investigated. With different choices of interpretation of the function that determines the influence of neighboring vertices on the vertex minimization value, we obtain a variety of applications. For example, the gradient descent method from the local optimization state makes it easy to obtain the minimum weight (cost) paths from the given starting vertex to all vertices. If the graph is interpreted as a semantic network in the space of natural language words, it is shown how the problem of finding the minimum path among words or phrases is interpreted as solving a known problem in computational linguistics: find the word most associated with a given input set words or sentences (images). Finally, by calculating the weights of the paths in randomized (uniformly distributed) graphs, a fast-authentication protocol is proposed for the two coalition entities acting in a malicious environment.

Conclusions. The local optimality is a stable constant state of the current graph marking reachable by using the dynamic minimization process. Local optimality provides a common method for solving many problems related to search engine optimization. It is shown how from solving the general problem of constructing a locally-optimal graph it follows the problem of finding the paths of least weights (cost). New applications include solving the problem of associative search in computational linguistics (creative thinking) and fast mutual authentication in coalition groups.

Keywords: *directed graph, shortest paths, associative search, randomness, fast authentication.*