ПРО ДЕЯКІ АЛГОРИТМИ ВІДШУКАННЯ ГАМІЛЬТОНОВИХ ЦИКЛІВ НА ЧИСЛОВИХ ГРАФАХ

Т.О. Гришанович

Робота присвячена проблемі відшукання гамільтонових циклів на числових графах. Зокрема, розглядаються алгоритм з поверненням, алгоритм Approx-TSP(G) та алгоритм із поліноміальним часом. Аналізується часова складність даних алгоритмів.

This work is devoted to finding Hamiltonian cycles on numeric graphs. In particular, the algorithm with returning, algorithm Approx-TSP(G) and polynomial algorithms are considered. Analyzed the time complexity of these algorithms.

Робота посвящена проблеме отискания гамильтоновых циклов на арифметических графах. Вчастности, рассматриваются алгоритм с возвращением, алгоритм Approx-TSP(G) и алгоритм с полиномиальным временем. Анализируется временная сложность этих алгоритмов.

Вступ. Широке використання обчислювальної техніки поставило перед всією математикою і перед теорією графів зокрема проблему знаходження не довільних алгоритмів, які дозволяють розв'язати той чи інший клас задач, а таких, які б допускали ефективну практичну реалізацію. Так виникла проблема практичної розв'язності задач: відшукати ефективний або хоча б достатньо простий у практично важливих випадках алгоритм розв'язання задачі [9]. Крім того, безпосереднє і детальне представлення практичних систем, таких як розподілені мережі та системи зв'язку, приводять до графів великої розмірності, успішний аналіз яких в значній мірі залежить від існування «хороших» алгоритмів для них [9].Традиційні способи представлення графів вимагають, як правило, значної кількості пам'яті, а, отже, громіздких обчислень [7]. Тому у таких випадках досить ефективним є використання числових графів.

Вибір саме числових графів для проведення дослідження зумовлений тим, що вони дозволяють значно зменшити витрати кількість пам'яті, а отже, і витрати часу на проведення обчислень [7]. У роботі [5] продемонстровано, що для числових графів вдалось значно зменшити часову скла-

дність алгоритму розкладання графів за допомогою їхніх кістяків (у порівнянні з представленням графів у вигляді матриць суміжності). Тому виникає наступне питання: чи вдасться покращити ефективність за часом уже відомих алгоритмів відшукання гамільтонових циклів за у мови представлення графів у вигляді числових?

Задача відшукання гамільтонових циклів у графі є однією із найстаріших та модельних задач, а також має численні застосування як у самій теорії графів, так і у її прикладних галузях [12]. Найвідомішою проблемою, що ґрунтується на побудові гамільтонових циклів є задача комівояжера.

У даній роботі описано алгоритми відшукання гамільтонових циклів, інтерпретовані для застосування їх на числових графах. Для проведення дослідження було обрано наступні алгоритми: алгоритм із поверненням [2], Approx-TSP(G) [2, 9] та алгоритм із поліноміальним часом [15]. Проведено аналіз часової складності таких алгоритмів для представлення графів у вигляді числових графів та здійснено порівняння отриманих результатів із складностями тих же алгоритмів за умови представлення графів у вигляді матриць суміжності.

Часова складність алгоритму, як функція від розміру вхідних даних, є однією із важливих мір складності алгоритму. Для того, щоб виконати її обчислення, слід визначити час, необхідний для виконання кожної із команд. Для обчислення складності алгоритму будемо використовувати логарифмічний ваговий критерій. Нехай $l(i)$ – логарифмічна функція на цілих числах, яка задається наступними рівностями:

$$l(i) = \begin{cases} \log \mid i \mid +1, & \text{якщо } i \neq 0; \\ 1, & \text{якщо } i = 0. \end{cases}$$

Такий критерій базується на припущенні, що ціна виконання команди (її вага) прямо пропорційна довжині її операндів.[4]

Основні результати. Для проведення дослідження було обрано наступні алгоритми: алгоритм із поверненням [2], Approx-TSP(G) [1, 2] та алгоритм і поліноміальним часом [15]. Кожен із цих алгоритмів був інтерпретований для застосування на такій структурі даних як числові графи.

Будемо розглядати зв'язний ненавантажений числовий (натуральний арифметичний) граф $G = (X, U, f)$, що задається кількістю вершин, множиною твірних U, а його функція суміжності має вигляд: $f = (x_i + x_j), (x_i, x_j) \in X$.

Алгоритм із поверненням [2]. Для того, аби перерахувати усі гамільтонові цикли графа, даний алгоритм виконує наступні кроки:

1. Вершина x_1 вибирається в якості відправної і утворює перший елемент множини S. Ця множина буде зберігати у собі вершини ланцю-

га.

2. До S додається наступна можлива вершина, яка суміжні із x_1. Під можливою будемо розуміти ту вершину, яка не входить до множини S.

3. На деякому кроці r виявиться, що неможливо включити вершину x_r до множини S. Це може трапитись у двох випадках:

   (a) Відсутня вершина у графі (відсутня суміжна вершина).

   (b) Ланцюг, що визначається послідовністю вершин в S, має довжину $n-1$, тобто є гамільтоновим. Для цього випадку можливі 2 варіанти:

      i. Існує ребро (x_1, x_r), тому знайдений цикл - гамільтонів.

      ii. Ребро (x_1, x_r) у графі відсутнє, гамільтонів цикл відшукати неможливо.

У випадках 3.a та і 3.b.ii слід вдатися до повернення, в той час як у випадку 3.a.i можна припинити пошук і надрукувати результат (якщо потрібно знайти лише один гамільтонів цикл), або (якщо потрібні всі такі цикли) зробити друк і вдатися до повернення:

1. Повернення полягає у видаленні останньої включеної вершини xr з множини S, після чого залишається множина $S = x_1, ..., x_(r-1)$.

2. Пошук закінчується в тому випадку, коли множина S складається тільки з вершини x_1 і не існує ніякої можливої вершини, яку можна додати до S, так що крок повернення робить множину S порожньою.

До ланцюга S може входити щонайбільше n вершин графа. Сам же алгоритм переглядає усі вершини графа, розглядаючи кожну із них як продовження максимального простого ланцюга. Нагадаємо, що для обчислення часової складності алгоритму використовується логарифмічний ваговий критерій. Підрахувавши кількість виконаних операці та скориставшись наступною теоремою:

Теорема 1. [4] *Якщо* $A(k) = a_m k^m + ... + a_1 k + a_0$ *є поліномом степеня* m, *тоді* $(k) = O(k^m)$,

отримуємо наступне поліноміальне значення для часової складності такого алгоритму за умови представлення графів у вигляді натуральних арифметичних: O(n^2), де n - кількість вершин графа. Для подання графів у вигляді матриць суміжності такий алгоритм має експоненціальне значення часової складності [2]. Зауважимо, що для обчислення часової складності наступних алгоритмів будемо здійснювати аналогічним чином.

Алгоритм Approx-TSP(G) [10]. Даний алгоритм було розроблено спеціально для розв'язання задачі комівояжера (дана задача заключається у знаходженні гамільтонового циклу мінімальної довжини[14]) у метричному просторі. (Задача комівояжера у метричному просторі є частковим випадком задачі комівояжер, де відстані вихідного графа задовольняють нерівності трикутника:

$$d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3),$$

де x_1, x_2, x_3 – вершини графа, $d(x_i, x_j)$ – відстань між заданими вершинами) [10]. Алгоритм Approx-TSP(G) складається із наступних кроків:

1. Побудувати мінімальний кістяк графа. Оскільки граф, що розглядається є ненавантаженим, то для виконання цього кроку алгоритму достатньо побудувати звичайний кістяк графа. Для цього скористаємось алгоритмом пошуку в глибину. Вибирають та відвідують початкову (довільну) вершину графа. x_0. Після цього вибирають ребро (x_0, x_i) інцидентне x_0, та відвідують вершину x_i. Нехай x – остання відвідана вершина. Виберемо довільне ребро (x, x_j), інцидентне x, яке ще не розглядалось. Якщо вершина x_j уже відвідувалась, то шукаємо нове ребро, яке інцидентне x. Якщо y – не відвідана вершина, ідемо до неї та розпочинаємо пошук з цієї вершини. У тому випадку, коли переміщення вперед є неможливим, повертаємось до вершини x (до тієї вершини, по якій ми прийшли до y) та розпочинаємо пошук з неї. В результаті ми повернемось до вершини x_0 та виявимо неможливість продовжувати пошук. Після цього шукаємо нову вершину x_1, яка не відвідувалась раніше, та продовжуємо з неї пошук. Пошук закінчено у тому випадку, коли усі вершини графа були відвідані.

2. Продублювати кожне із ребер знайденого кістяка та в отриманому графі відшукати ейлерів цикл. (Ейлеровим називають цикл, який містить усі ребра та вершини графа[14].)

   Виникає питання: чи існує у графі, що утворений із дерева шляхом подвоєння його ребер, ейлерів цикл?

   Теорема 2 (Ейлера)**.** [14] *Зв'язний граф є ейлеревим тоді і тільки тоді, коли кожна його вершина має парний степінь.*

   Відомо, що кожне дерево із n вершинами має (n-1) ребро. [14] Тобто у дереві кожна із вершин має степінь $q, q = \{ 1, 2, \ldots (n-1)\}$. Розглянемо граф, що утворився із дерева вихідного графа шляхом подвоєння кожного із його ребер. При подвоєнні усіх ребер графа степінь кожної із його вершин збільшиться вдвічі: $q = \{ 2, 4, \ldots 2(n-1)\}$. У

такому випадку, якщо вершина у дереві і мала непарний степінь, то у новоутвореному графі її степінь стає парним. За теоремою Ейлера ейлерів цикл у такому графі існує.

3. У знайденому ейлеревому циклі видалити усі вершини, які повторюються. Отриманий цикл буде гамільтоновим циклом вихідного графа.

Алгоритм Approx-TSP(G) відноситься до наближених алгоритмів і для нього справедлива наступна теорема:

Теорема 3. [1] *Алгоритм Approx-TSP(G) розв'язує задачу комівояжера з помилкою не більше, ніж у 2 рази, якщо виконується нерівність трикутника.*

Для представлення графа у вигляді матриці суміжності [8] час виконання такого алгоритму відомий та становить $O(n^2)$, де n – кількість вершин графа. Оцінимо час роботи такого алгоритму для арифметичних графів. Кістяк для такого виду графів будується за час $O(n)$ [6]. Відшукання ейлеревого циклу має складність $O(m)$, де m – число ребер у графі. Оскільки пошук ейлеревого циклу здійснюється у графі, що утворений із його кістяка шляхом подвоєння ребер, то $m = 2(n-1)$. Тому алгоритм відшукання ейлеревого циклу тут має складність $O(n)$. . Таким чином отримуємо складність алгоритму Approx-TSP(G) для натуральних арифметичних графів $O(n)$.

Алгоритм із поліноміальним часом [15]. Такий алгоритм передбачає виконання наступних кроків:

1. Знайти цикл, який довільним чином містить усі вершини графа. У такому циклі дві сусідні вершини можуть бути не суміжними у вихідному графі. Такі точки будемо називати «break point».

2. Для кожної такої точки, крім одної (її називають «main beak point»), додати нове ребро (слід запам'ятати додані ребра, оскільки кожного разу видаляючи ребро, ми будемо розпочинати роботу алгоритму спочатку).

3. Виправити «main beak point»: вирізати сегмент (частину) із «main beak point» та вставити цей сегмент у деяке довільне місце у циклі.

4. Виправити наступну «main beak point».

5. Виправлення (пункти 2-3) здійснювати до того часу, поки не буде знайдено гамільтонів цикл на графа.

Виникає питання: яким чином здійснювати таку вставку та вирізання? Правило наступне: зробити кількість нових «break point» якнайменшим і кожна нова «break point» повинна відрізнятись від усіх попередніх «main break point». При визначенні наступної «main break point» може трапитись так, що більш ніж, одна «break point» підходить для цього. У такому випадку слід порівняти наступний, але лише один, крок для кожної із цих точок. Також слід уникати вставки одного і того ж сегменту у різні частини циклу послідовно.

Стосовно часу виконання такого алгоритму, то час побудови циклу, що містить усі вершини графа становить $O(n)$, n – кількість вершин графа. Виправляючи кожного разу одну «break point», ми можемо додавати ребро щонайбільше n разів, максимальне ж число самих «break point» теж n. Таким чином отримуємо часову складність $O(n^2)$.

Висновки. У запропонованій роботі здійснено порівняння часової ефективності алгоритмів вішукання гамільтонових циклів для двох способів представлення графів: матриці суміжності та числові графи. Зокрема, для роботи із числовими графами уже відомі алгоритми було інтерпретовано для роботи саме із такими структурами даних. Проведений аналіз отриманих результатів свідчить про те, що для алгоритму із поверненням та алгоритму Approx-TSP(G) значення часової складності для числових графів у порівнянні із поданням їх у вигляді матриць суміжності вдалось покращити: для алгоритму із поверненням отримали поліноміальне значення оцінки, а для Approx-TSP(G) отримали зменшення степеня полінома. Для алгоритму із поліноміальним часом, як для матриць суміжності, так і для натуральних арифметичних графів отримали поліноміальне значення часової складності. Тобто використання числових графів дозволяє збільшити ефективність за часом розглянутих алгортмів.

1. **Алгоритмы.** Построение и анализ / [Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.; пер. с англ. И. В. Крисикова]. – М. : издательский дом «Вильямс», 2005. – 1293.
2. **Асанов М.О.** Дискретная математика: графы, матроиды, алгоритмы / Асанов М.О., Баранский В.А., Расин В.В. – М., Ижевск : РХД, 2001. – 288 с.
3. **Асельдеров З.М.** Представление и восстановление графов / Асельдеров З.М. – К. : Наукова думка, 1991. – 312 с.
4. **Ахо А.** Построение и анализ вычислительных алгоритмов / Ахо А., Хопкрофт Дж., Ульман Дж.; пер. с англ. А. О. Слисенко. – М. : Мир, 1979. – 536 с.
5. **Гришанович Т.О.** Про часову складність алгоритму розкладання числових графів за допомогою їх кістяків / Т. О. Гришанович // Вісник Київського національного університету. - 2011. - №2. - С.83-87.
6. **Донец Г.А.** Об оценке сложности алгоритмов в арифметических графах / Г.А. Донец, Ю.И. Неженцев // Методы решения задач нелинейного и дискретного программирования. – 1991.– №1. – С.79-87

7. **Донец Г.А.** Об общем представлении числовых графов / Г. А. Донец, И. Э. Шулинок // Теорія оптимальних рішень. – 2004. – №3. – С. 18-25.

8. **Касьянов В.Н.** Графы в программировании: обработка, вазуализация и применение / В.Н. Касьянов, В.А. Євстигнеев. – СПб. : БХВ-Петербург, 2003. – 1106 с.

9. **Кристофидес Н.** Теория графов. Алгоритмический поход / Кристофидес Никос; пер. с англ. Г. О. Гаврилов. – М.: Мир, 1978. – 432 с.

10. **Куликов А. С.** Эффективные алгоритмы: конспект лекций [Електронний ресурс] / А.С. Куликов // Лабораторія математической логики – Режим доступу: http://logic.pdmi.ras.ru/

11. **Неженцев Ю. И.** Об оценке сложности алгоритмов поиска в арифметических графах / Ю.И. Неженцев // Математические методы и программное обеспечение в системах принятия решения и проектирования. - К: Ин-т кибернетики им. В.М. Глушкова АН УССР. – 1990. – С. 14-24.

12. **Скобелев В.Г.** Локальные алгоритмы на графах / Скобелев В.Г. – Донецк. : ИПМП НАНУ, 2003. – 218 с.

13. **Шинкаренко В.І.** Особливості практичного застосування показників обчислювальної складності алгоритмів / В.І. Шинкаренко // Проблеми програмування. – 2008. – № 2-3. – С. 12-20.

14. **Эвнин А.Ю.** Дискретная математика: конспект лекций / Александр Юрьевич Эвнин. – Челябинск : Издательство ЮУрГУ, – 1998. – 177 с.

15. **Lizhi Du.** A Polynomial Time Algorithm for Hamilton Cycle (Path) / Du Lizhi // IMECS 2010: International MultiConference of Engineers and Computer Scientists, marc 17-19, 2010: proceedings. – Hong Kong, 2010. – vol. 1, P.21-25.