Методика масштабування алгоритмів маршрутизації

С.Д. Погорілий, І.В.Пантелєєва

Київський національний університет імені Т.Г.Шевченка

Abstract:

S.D.Pogorily, I.V.Panteleyeva. Scaling methodic of routing algorithms. *Scaling Dijkstra's algorithm was considered and scaling of Floyd-Warshall's algorithm is executed. Strategy of scaling of routing algorithms is offered. Images 3. Ref. 8.*

Key words: routing, Floyd-Warshall's algorithm, Dijkstra's algorithm, scaling, strategy of scaling, shortest path, source vertex.

Вступ

Однією з найбільш важливих задач мереж є задача маршрутизації. Існує багато алгоритмів маршрутизації, але всі вони мають певні недоліки. Серед них:

- неефективність використання у великих мережах;
- фіксована метрика алгоритмів;
- відсутність врахування альтернативних маршрутів.

Сьогодні широко використовуються два протоколи маршрутизації:

- OSPF, який використовує алгоритм Дейкстри;
- RIP, який використовує алгоритм Белмана-Форда.

Більш докладно ці протоколи маршрутизації розглянуто у [5-8].

У 1985 році Гарольд Габов (Harold N. Gabow) опублікував статті [1, 2], в яких запропонував спосіб підвищення швидкості основних мережевих алгоритмів за допомогою масштабування, а також представив масштабовану версію алгоритму Дейкстри.

Масштабування має свої переваги та недоліки:

- покращує часові характеристики алгоритмів маршрутизації;
- масштабовані алгоритми показують високу ефективність в великих мережах;
- використання тільки додатних довжин граней.

Метою цієї роботи є створення методики масштабування алгоритмів маршрутизації на основі алгоритмів знаходження найкоротших шляхів в орієнтованих графах та формування схеми її реалізації для алгоритму Флойда-Уоршалла.

Формалізація алгоритму Флойда-Уоршалла

Задача про пошук найкоротших шляхів між усіма парами вершин вирішується в орієнтованому графі $G = (V, E)$. Час виконання алгоритму Флойда-Уоршалла (Floyd-Warshall algorithm) дорівнює $\Theta(V^3)$. Наявність ребер з від'ємною вагою допускається, але виключаються від'ємні цикли. Детально алгоритм розглянуто у [3,4].

Алгоритм використовує 3 матриці:

- W – матриця суміжності, яка визначається наступним чином:
  - wij = вага ребра (i,j), якщо існує ребро (i,j);
  - wij = ∞, якщо ребра (i,j) не існує;
- D – матриця ваг найкоротших шляхів:
  - D(0) відповідає матриці W, тобто $d_{ij}^{(0)} = w_{ij}$;
  - D(k) визначається співвідношенням $d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$;
  - D(n) дає кінцевий результат $d_{ij}^{(n)} = \delta(i, j)$ (ваги найкоротших шляхів);
- П – матриця передування:
  - П(0) визначається наступним чином:
    - якщо i = j або wij = ∞, то $\pi_{ij}^{(0)} = NIL$;
    - якщо i ≠ j та wij < ∞, то $\pi_{ij}^{(0)} = i$;
  - $\pi_{ij}^{(k)}$ - попередник вершини j на найкоротшому шляху з вершини i, всі проміжні вершини якого належать множині {1,2,..., k} і визначаються наступним чином:

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)}, d_{ij}^{(k-1)} \le d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ \pi_{kj}^{(k-1)}, d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}.$$

Виходячи з визначень, наведених вище, можна скласти наступну процедуру:

```
Floyd_Warshall(W)
1    n ← rows[W]
2    D(0) ← W
3    for k ← 1 to n
4        for i ← 1 to n
5            for j ← 1 to n
6                do d_ij^(k) ← min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
7                   (π_ij^(k) = π_ij^(k-1)) ∨ (π_ij^(k) = π_kj^(k-1))
8    return D(n)
```

Вхідними даними виступає матриця W розміром nxn, яка визначена вище.

Процедура повертає матрицю D(n), яка містить ваги найкоротших шляхів між всіма парами вершин у графі.

Масштабування алгоритму Флойда-Уоршалла

Визначимо «близьке до оптимального» рішення проблеми найкоротших шляхів як функцію d_{ij} таку, що:

1) $d_{ii} = 0$
2) d_{ij} домінує над довжинами ребер для будь-яких граней (i,j), $l_{ij} + d_i \geq d_j$ (рис. 1)

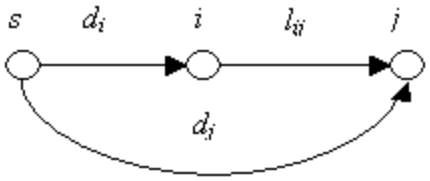

Рисунок 1

3) кожна вершина j має шлях з i, значення довжини якого лежить між d_{ij} та $d_{ij} + m$.

Масштабований алгоритм Флойда-Уоршалла по черзі вибирає як виток вершину $s \in V$. Далі алгоритм знаходить найкоротші шляхи з витоку s в кожну вершину i. Алгоритм найкоротшого шляху базується на процедурі, яка перетворює наближене рішення у точне. Щоб це зробити спочатку підраховують модифіковані довжини ребер l’ij = di + lij – dj. Далі підраховують найкоротші шляхи для модифікованих довжин ребер, використовуючи алгоритм Флойда-Уоршалла. Модифіковані довжини ребер не можуть змінити найкоротші шляхи. Легко бачити, що повний час виконання цієї процедури перетворення дорівнює O(m).

Розглянемо ітеративну версію масштабування, де k = [log2N] + 1 масштабувань відповідають послідовним бітам у бінарному наборі b1, …, bk. Перше масштабування бере ведучий біт чисел b1, і обраховує відстані d_i^1 для кожної вершини i; у загальному випадку масштабування t обраховує відстані d_i^t. Щоб виконати масштабування t + 1, необхідно змінити поточні довжини граней lij на $l_{ij}' = d_i^t + l_{ij} - d_j^t$ і підвищити величину масштабування до 2l’ij + bt+1, де bt+1 - (t+1)-ий біт первинної довжини ребра (i,j). Найкоротший шлях масштабування k – це найкоротший шлях дерева з витоком s для первинного графа. (Щоб доказати це, спочатку помітимо, що в масштабуванні t ребро (i,j) з

первинною довжиною b1, …, bk має довжину $b_1 \cdots b_t + \sum_{x=1}^{t-1} 2^{t-x}(d_i^x - d_j^x)$.

Далі, величини $\sum_{x=1}^{t} 2^{t-x} d_i^x$ відповідають відстаням для довжин ребер $b_1, \ldots, b_k$ як і потрібно [1]).

Передбачається, що існує масив ребер - це дозволяє зменшити час виконання операцій над ребрами до $O(V)$ і проводити підрахунки тільки для масиву існуючих ребер, а не для всіх можливих комбінацій з матриці суміжностей.

Виходячи з визначень, наведених вище, можна скласти таку процедуру:

```
Scalling_Floyd_Warshall(W)
1     n ← rows[W]
2     k ← [log2N]+1
3     for s ← 1 to n
4           for all l do l_{i,j} ← b_1
5           for t ← 1 to k
6                 do    d^t ← Floyd_Warshall(W)
7                       for all l
8                             do    l_{i,j} ← d_i^t + l_{i,j} − d_j^t
9                                   l_{i,j} ← 2 * l_{i,j} + b_{t+1}
10                      for i ← 1 to n do  R_{s,i} ← 2 * R_{s,i} + d_i^t
11    return R
```

Таким чином, масштабований алгоритм виконується за час $O(n^2k) = O(n^2\log_2 N)$, що суттєво покращує часову межу алгоритму Флойда-Уоршалла $O(n^3)$. Рисунки 2 та 3 ілюструють, що час виконання алгоритму є пропорційним $O(n^2\log_2 N)$.

Методика масштабування алгоритмів маршрутизації

Масштабування алгоритмів маршрутизації ґрунтується на тому, що підрахунки виконуються для вершини-витоку і для уточнення довжини ребра беруться найкоротші шляхи поточної ітерації з вершини-витоку до вершин поточного ребра. Таким чином, щоб вирішити задачу знаходження найкоротших шляхів для всіх вершин графа, необхідно послідовно здійснити масштабований алгоритм для всіх вершин графу, по черзі приймаючи кожну з них за вершину-виток.

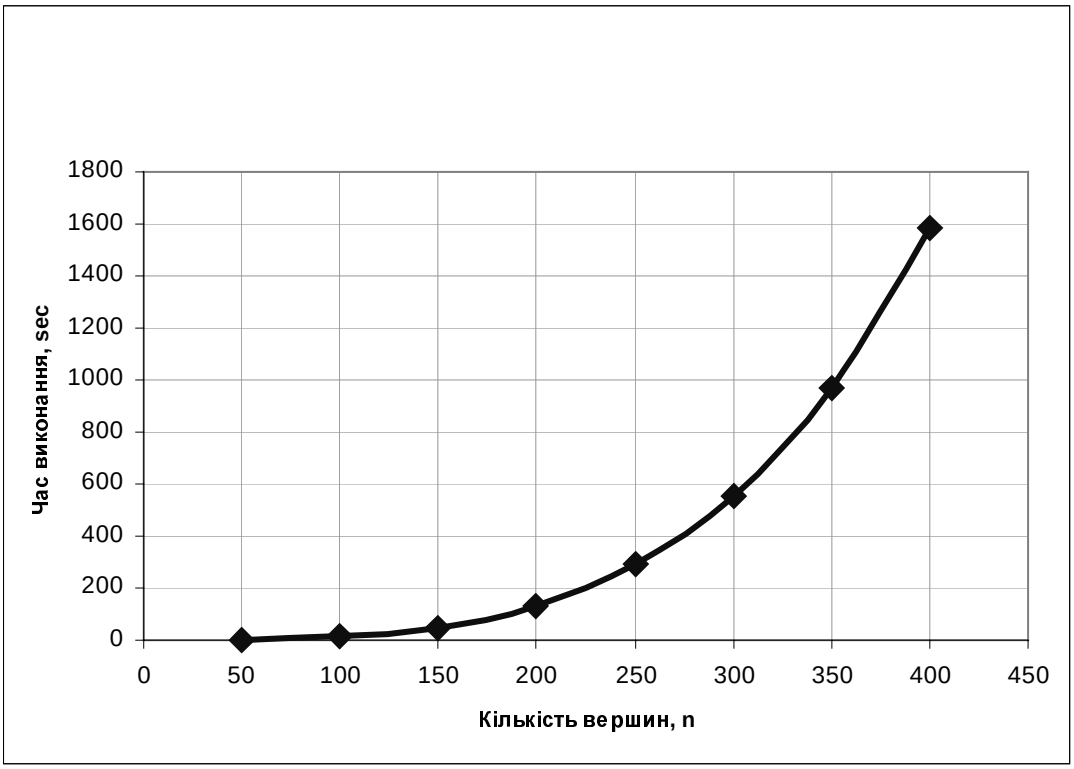

Рисунок 2 – Масштабований алгоритм Флойда-Уоршалла. Залежність t(n), n=[1,50-400], w=[1,5]

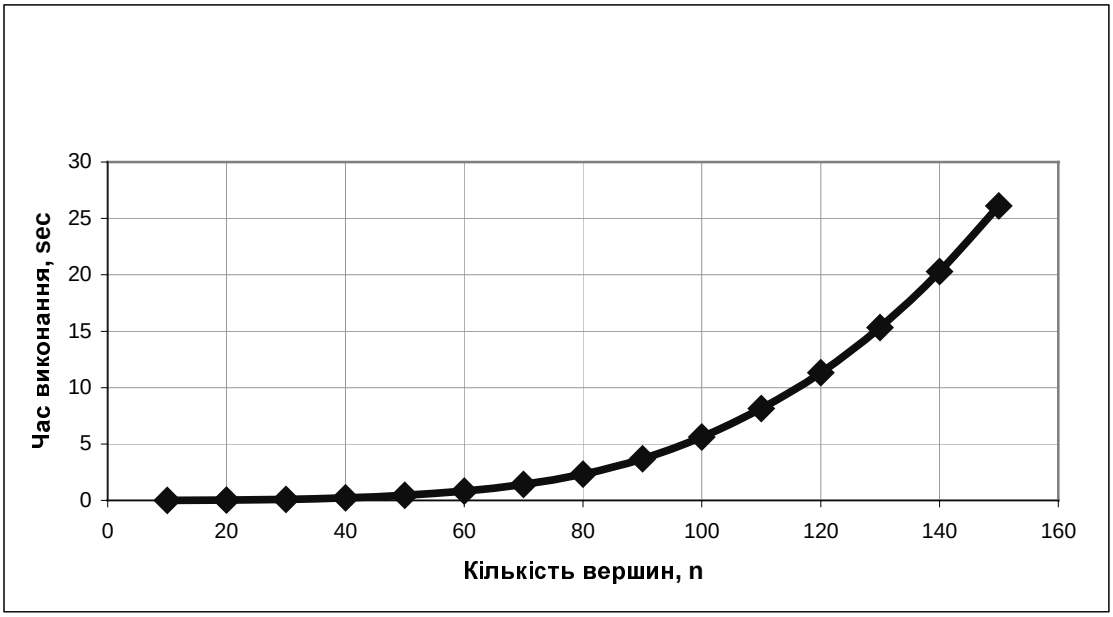

Рисунок 3 – Масштабований алгоритм Флойда-Уоршалла. Залежність t(n), n=[1,10-150], w=[1,5]

Ґрунтуючись на розглянутих масштабованих алгоритмах Дейкстри [1] і Флойда-Уоршалла, можна скласти методику масштабування:

1 $k \leftarrow [\log_2 N]+1$
2 *для всіх вершин-витоків* $s \in V$
3 *do* *для всіх ребер* $(i,j) \in E$ *set* $l_{i,j} \leftarrow b_1$ *of* $l_{i,j}$
4 *for* $t \leftarrow 1$ *to* k
5 *do* $d^t \leftarrow$ *Алгоритм маршрутизації(W)*
6 *для всіх ребер* $(i,j) \in E$ *set* $l_{i,j} \leftarrow d_i^t + l_{i,j} - d_j^t$
7 *для всіх ребер* $(i,j) \in E$ *set* $l_{i,j} \leftarrow 2 * l_{i,j} + b_{t+1}$
8 *для всіх вершин* $i \in V$ *set* $R_{s,i} \leftarrow 2 * R_{s,i} + d_i^t$
9 *return R*

В залежності від алгоритму маршрутизації та кількості вершин-витоків R може бути як масивом, так і матрицею.

Висновки

1) Проведено масштабування алгоритму Флойда-Уоршалла і отримано покращену часову межу.

2) Розроблено методику масштабування алгоритмів маршрутизації і наведено схему масштабування.

3) Обґрунтовано переваги масштабованих алгоритмів маршрутизації.

Література

1. H. N. Gabow. Scaling algorithms for network problems, J. Of Comput. System Sci., 31 (1985), pp. 148–168.
2. H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems, SIAM J. Comput., 18 (1989), pp. 1013–1036.
3. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы, построение и анализ, 2-е издание. М.:Издательский дом «Вильямс», 2005, с. 718 – 722.
4. Роберт Седжвик. Фундаментальные алгоритмы на С++. Алгоритмы на графах. СПб: ООО «Диасофт ЮП», 2002, 496с.
5. http://citforum.ru/nets/semenov/4/44/rip44111.shtml
6. http://citforum.ru/nets/semenov/4/44/osp44112.shtml
7. Погорілий С.Д., Калита Д.М. Про підхід до формалізації протоколів маршрутизації на прикладі протоколу OSPF. // Вестник МСУ. 2000. №4. с. 79-85.
8. Погорілий С.Д., Калита Д.М. Оптимізація алгоритмів маршрутизації з використанням систем алгоритмічних алгебр. УСиМ 2000, №4 с. 20-30.