УДК 621.39:681.3

**Ю.В. Пономарьов, нач. РВВ**
*Житомирський військовий інститут радіоелектроніки*

# СПОСІБ ОБЛІКУ ПОТОЧНИХ ЗНАЧЕНЬ ДЛЯ РОЗРАХУНКУ НАЙКОРОТШИХ МАРШРУТІВ ПЕРЕДАЧІ ІНФОРМАЦІЇ ПОТОЧНІЙ ІТЕРАЦІЇ

*(Представлено доктором технічних наук, професором Ю.І. Лосєвим)*

*Розглянуто класичний матричний алгоритм пошуку найкоротших маршрутів передачі інформації в мережах передачі даних (ПД).*

*Запропоновано новий спосіб обліку поточних значень для розрахунку найкоротших маршрутів передачі інформації в поточній ітерації, а також алгоритм, що побудований завдяки даному способу.*

*Наведений порівняльний аналіз обчислювальної складності вказаних алгоритмів, визначені їх достоїнства й недоліки при практичному застосуванні.*

Залежно від призначення мережі ПД можуть використовувати різні алгоритми пошуку найкоротших маршрутів: розподілені, централізовані, гібридні [1]. У тому випадку, коли мережа ПД призначена для збору, обробки, зберігання та передачі керувальної інформації у складі загального алгоритму маршрутизації, необхідно передбачити використання централізованого алгоритму пошуку найкоротших маршрутів. Оскільки інформацію для вказаних цілей зручно представляти у вигляді матриць, то для розробки вказаного алгоритму доцільно використовувати класичний матричний алгоритм [2]. Однак даний алгоритм має ряд суттєвих недоліків: велику обчислювальну складність; тривалий час розрахунку та великий обсяг проміжної інформації, яку потрібно зберігати у пам'яті ЕОМ. Причиною цих недоліків є те, що задача пошуку найкоротших маршрутів розв'язується в ході послідовних ітерацій, кількість яких, залежно від структури, кількості вузлів і ваг (довжин) каналів зв'язку в мережі ПД, може бути дуже великою. Це пов'язано з тим, що даний алгоритм у ході послідовних ітерацій ($T$) розраховує матрицю довжин найкоротших маршрутів ($A$) з урахуванням використання основної властивості Белмана [3], тобто в кожній ітерації знаходяться найкоротші маршрути, кількість транзитних ділянок в яких не перевищує порядкового номера поточної ітерації.

$$A_{i,j}^{(T+1)} = \min_{n} \left[ A_{i,n}^{(T)} + a_{n,j} \right], \qquad (1)$$

де $a_{n,j}$ – вага (довжина) каналу зв'язку від якогось транзитного вузла ($n$) в маршруті до вузла-адресата ($j$), а $i$ – вузол-джерело.

Це призводить до послідовного (від ітерації до ітерації) поліпшення маршрутів з точки зору їх оптимальності. Проте, як показують результати моделювання, цей спосіб вимагає досить багато ітерацій, особливо при розв'язуванні задач великої розмірності (кількість вузлів у мережі ПД більше десяти). Це, в свою чергу, як вже було сказано, призводить до зростання обчислювальної складності алгоритму і, як наслідок, до більшого часу розрахунку поставленої задачі.

У зв'язку з цим виникає потреба в розробці такого алгоритму, який зміг би суттєво знизити (послабити) вищеперераховані недоліки.

Для цього, з початку, наведемо стислий опис вищевказаного (класичного) алгоритму [2].

Він, в основному, працює з двома робочими матрицями [4]: ваг (довжин) найкоротших маршрутів ($A$) і маршрутних змінних ($M$).

Крім того, для формування робочих матриць використовуються такі вхідні (початкові) дані:

1. $V$ – кількість вузлів у мережі ПД;

2. $B$ – матриця суміжності (розміром $V \times V$), що характеризує топологію (конфігурацію) мережі ПД, тобто наявність (якщо $b_{ij} = 1$) чи відсутність (якщо $b_{ij} = 0$) каналів зв'язку між окремими вузлами комутації.

Таким чином, матриця ($B$) задає кількість каналів зв'язку ($Q$) в мережі ПД.

3. $C = \| c_{i,j} \|$ – матриця ваг (довжин) каналів зв'язку (в умовних одиницях).

**Опис алгоритму**

**Крок 1.** За допомогою початкових даних формуються робочі матриці ( $A$ і $M$ ) у першій ( $T = 1$ ) ітерації.

Формування даних матриць здійснюється таким чином:

$$\alpha_{i,j}^{(T-1)} \begin{cases} c_{i,j}, & \text{якщо} \quad b_{i,j} = 1; \\ \infty, & \text{якщо} \quad b_{i,j} = 0, \end{cases} \tag{2}$$

де $a_{i,j}^{(T)}$ – елемент матриці $A^{(T-1)}$, значення якого вказує на вагу (довжину) маршруту від вузла-джерела ( $i$ ) до вузла-адресата ( $j$ );

$$m_{i,j}^{(T-1)} = \begin{cases} i, & \text{якщо} \quad b_{i,j} = 1; \\ 0, & \text{якщо} \quad b_{i,j} = 0, \end{cases} \tag{3}$$

де $m_{i,j}^{(T)}$ – елемент матриці $M^{(T-1)}$, значення якого вказує на номер вузла, найближчого до вузла-адресата ( $j$ ) у найкоротшому маршруті.

**Крок 2.** У ході послідовних ітерацій коректуються значення елементів матриць ( $A$ ) і ( $M$ ).

$$A_{i,j}^{(T)} = \min_{n \in V} [a_{i,n}^{(T-1)} + a_{n,j}^{(1)}], \tag{4}$$

де $\{n\}$ – множина вузлів у мережі ПД; $a_{i,n}^{(T-1)}$ – значення елемента матриці $A$ від вузла-джерела ( $i$ ) до транзитного вузла ( $n$ ) у попередній ( $T-1$ ) ітерації; $a_{n,j}^{(1)}$ – значення елемента матриці $A$ від транзитного вузла ( $n$ ) до вузла-адресата ( $j$ ) у першій ( $T = 1$ ) ітерації.

Значення елементів матриці $M$ дорівнюють тим значенням елементів множини $\{n\}$, для яких виконується (4), тобто:

$$M_{i,j}^{(T)} = M_{n,j}^{(T-1)}. \tag{5}$$

**Крок 3.** Здійснюється порівняння значень елементів матриць ( $A$ ) і ( $M$ ), отриманих у поточній ітерації, зі значеннями елементів цих же матриць, отриманих у попередній ( $T-1$ ) ітерації.

$$A^{(T)} = A^{(T-1)} \;\&\; M^{(T)} = M^{(T-1)}. \tag{6}$$

Якщо умова (6) виконується – кінець роботи алгоритму, якщо ні, то:

$$T = T + 1 \tag{7}$$

і здійснюється перехід до кроку 2.

Оцінимо обчислювальну складність даного алгоритму за кількістю двох основних операцій, які потрібні для пошуку елементів матриць $A$ і $M$: додавання $O(V_+)$ і пошуку мінімуму $O(V_{min})$.

Легко помітити, що для знаходження кожного елемента матриці $A$ і $M$ потрібно провести $V$ операцій додавання та одну операцію пошуку мінімуму. Для знаходження елементів матриць $A$ і $M$ одного рядка потрібно провести $V^2 - V$ операцій додавання та $V$ операцій пошуку мінімуму.

Таким чином, у кожній ітерації для знаходження елементів матриць $A$ і $M$ потрібно буде провести $V^3 - V^2$ операцій додавання та $V^2$ операцій пошуку мінімуму.

Оскільки в мережі ПД, у найгіршому випадку, може знайтися найкоротший маршрут, що складається з максимуму транзитних ділянок $(V-1)$, то для вирішення задачі пошуку найкоротших маршрутів потрібно буде $T = (V-1)$ ітерацій [2]. Тоді обчислювальна складність алгоритму дорівнюватиме:

$$O(V_{алг}^{(1)}) = O(T \times [(V^3 - V^2)_+ + V_{\min}^2]) = O([V-1] \times [(V^3 - V^2)_+ + V_{\min}^2]) \approx O([(V^4 - V^3) + V_{\min}^3]) \tag{8}$$

Зрозуміло, що обчислювальна складність алгоритму дуже велика.

Виходячи з цього, потрібно розробити такий алгоритм, який би мав суттєво меншу обчислювальну складність, ніж вищевказаний.

В зв'язку з цим, пропонується спосіб обліку поточних значень матриці $A$, отриманих у поточній ітерації.

Для організації цього способу необхідно, по-перше, відмовитися від основної властивості Белмана (1), а, по-друге, потрібно в якості початкових даних в кожній ітерації (крім першої) мати значення матриць $A$ і $M$, що дорівнюють тим значенням, які отримані в попередній

ітерації. Тільки в цьому випадку можна одночасно оперувати як початковими, так і перерахованими даними в ході поточної ітерації. Отже, при побудові алгоритму, завдяки даному способу, необхідно передбачити наявність двох основних етапів в кожній ітерації (крім першої).

**Етап 1.** Етап присвоювання. На початку поточної ітерації (крім першої) здійснюється присвоювання елементів матриць $A$ і $M$ значенням цих же матриць, що отримані в попередній ітерації, тобто:

$$A^{(T)} = A^{(T-1)} \;\&\; M^{(T)} = M^{(T-1)}, \text{ якщо } T \neq 1. \tag{9}$$

**Етап 2.** Етап корекції значень. Знаходяться найкоротші маршрути з урахуванням поточних значень, отриманих в цій же ітерації за допомогою виразів (10) і (11).

$$A_{i,j}^{(T)} = \min_{n \neq i \neq j \in V} \left[ A_{i,n}^{(T)} + A_{n,j}^{(T)} \right], \text{ якщо } T > 1. \tag{10}$$

$$M_{i,j}^{(T)} = M_{i,n}^{(T)}, \text{ якщо } T > 1. \tag{11}$$

Отже, роботу алгоритму, побудованого завдяки вищевказаному способу, можна стисло подати таким чином:

**Крок 1.** Згідно з виразами (2–3) формуються робочі матриці $A$ і $M$ в ході першої ($T = 1$) ітерації.

**Крок 2.** Згідно з (9) здійснюється перший, а потім, згідно з виразами (10–11), –у ході другої ($T = 2$) ітерації.

**Крок 3.** Перевіряється умова (6), якщо вона виконується – кінець роботи алгоритму, якщо ні – згідно з виразом (7) здійснюється перехід до наступної ітерації (крок 2).

Аналіз виразу (10) показує, що, по-перше, при пошуку кожного елемента матриці $A$ потрібно робити на дві операції додавання менше ($n \neq i \neq j$), а, по-друге, у ході розрахунку цих елементів можливі три випадки:

1. Коли обидва значення ($a_{i,n}^{(T)}$ і $a_{n,j}^{(T)}$) у поточній ітерації ще не скоректовані, тобто вони є початковими у цій ітерації.

2. Коли одне з двох значень ($a_{i,n}^{(T)}$ або $a_{n,j}^{(T)}$) у поточній ітерації вже скоректоване.

3. Коли обидва значення ($a_{i,n}^{(T)}$ і $a_{n,j}^{(T)}$) вже скоректовані.

Процес розрахунку кожного елемента матриці ($A$) згідно з (10) можна докладно показати у вигляді табл. 1 для мережі ПД, що складається, наприклад, з п'яти вузлів комутації ($V = 5$).

*Таблиця 1*

| Значення $a_{ij}$ | $a_{i,n1} + a_{n1,j}$ | $a_{i,n2} + a_{n2,j}$ | $a_{i,n3} + a_{n3,j}$ | - - | . - | - . | . . |
|---|---|---|---|---|---|---|---|
| | $min\,[a_{i,n} + a_{n,j}]$, при $n \neq i \neq j \in V$ | | | | | | |
| $a_{12}$ | $a_{13} + a_{32}$ | $a_{14} + a_{42}$ | $a_{15} + a_{52}$ | 3 | 0 | 0 | 0 |
| $a_{13}$ | $\mathbf{a_{12}} + a_{23}$ | $a_{14} + a_{43}$ | $a_{15} + a_{53}$ | 2 | 1 | 0 | 0 |
| $a_{14}$ | $\mathbf{a_{12}} + a_{24}$ | $\mathbf{a_{13}} + a_{34}$ | $a_{15} + a_{54}$ | 1 | 2 | 0 | 0 |
| $a_{15}$ | $\mathbf{a_{12}} + a_{25}$ | $\mathbf{a_{13}} + a_{35}$ | $\mathbf{a_{14}} + a_{45}$ | 0 | 3 | 0 | 0 |
| $a_{21}$ | $a_{23} + a_{31}$ | $a_{24} + a_{41}$ | $a_{25} + a_{51}$ | 3 | 0 | 0 | 0 |
| $a_{23}$ | $\mathbf{a_{21}} + \mathbf{a_{13}}$ | $a_{24} + a_{43}$ | $a_{25} + a_{54}$ | 2 | 0 | 0 | 1 |
| $a_{24}$ | $\mathbf{a_{21}} + \mathbf{a_{14}}$ | $\mathbf{a_{23}} + a_{34}$ | $a_{25} + a_{54}$ | 1 | 1 | 0 | 1 |
| $a_{25}$ | $\mathbf{a_{21}} + \mathbf{a_{15}}$ | $\mathbf{a_{23}} + a_{35}$ | $\mathbf{a_{24}} + a_{45}$ | 0 | 2 | 0 | 1 |
| $a_{31}$ | $a_{32} + \mathbf{a_{21}}$ | $a_{34} + a_{41}$ | $a_{35} + a_{51}$ | 2 | 0 | 1 | 0 |
| $a_{32}$ | $\mathbf{a_{31}} + \mathbf{a_{12}}$ | $a_{34} + a_{42}$ | $a_{35} + a_{52}$ | 2 | 0 | 0 | 1 |
| $a_{34}$ | $\mathbf{a_{31}} + \mathbf{a_{14}}$ | $\mathbf{a_{32}} + \mathbf{a_{24}}$ | $a_{35} + a_{54}$ | 1 | 0 | 0 | 2 |
| $a_{35}$ | $\mathbf{a_{31}} + \mathbf{a_{15}}$ | $\mathbf{a_{32}} + \mathbf{a_{25}}$ | $\mathbf{a_{34}} + a_{45}$ | 0 | 1 | 0 | 2 |
| $a_{41}$ | $a_{42} + \mathbf{a_{21}}$ | $a_{43} + \mathbf{a_{31}}$ | $a_{45} + a_{51}$ | 1 | 0 | 2 | 0 |
| $a_{42}$ | $\mathbf{a_{41}} + \mathbf{a_{12}}$ | $a_{43} + \mathbf{a_{32}}$ | $a_{45} + a_{52}$ | 1 | 0 | 1 | 1 |
| $a_{43}$ | $\mathbf{a_{41}} + \mathbf{a_{13}}$ | $\mathbf{a_{42}} + \mathbf{a_{23}}$ | $a_{45} + a_{53}$ | 1 | 0 | 0 | 2 |
| $a_{45}$ | $\mathbf{a_{41}} + \mathbf{a_{15}}$ | $\mathbf{a_{42}} + \mathbf{a_{25}}$ | $\mathbf{a_{43}} + \mathbf{a_{35}}$ | 0 | 0 | 0 | 3 |
| $a_{51}$ | $a_{52} + \mathbf{a_{21}}$ | $a_{53} + \mathbf{a_{31}}$ | $a_{54} + \mathbf{a_{41}}$ | 0 | 0 | 3 | 0 |
| $a_{52}$ | $\mathbf{a_{51}} + \mathbf{a_{12}}$ | $a_{53} + \mathbf{a_{32}}$ | $a_{54} + \mathbf{a_{42}}$ | 0 | 0 | 2 | 1 |
| $a_{53}$ | $\mathbf{a_{51}} + \mathbf{a_{13}}$ | $\mathbf{a_{52}} + \mathbf{a_{23}}$ | $a_{54} + \mathbf{a_{43}}$ | 0 | 0 | 1 | 2 |
| $a_{54}$ | $\mathbf{a_{51}} + \mathbf{a_{14}}$ | $\mathbf{a_{52}} + \mathbf{a_{24}}$ | $\mathbf{a_{53}} + \mathbf{a_{34}}$ | 0 | 0 | 0 | 3 |
| $\Sigma$ | 60 | | | | 10 | 10 | |
| | | | | 20 | 20 | | 20 |

В табл. 1 уведені наступні позначення, що відповідають:

1. (– –) – першому випадку роботи з даними;
2. (. –) – другому випадку роботи з даними (коли перше значення вже скоректовано, а друге – ще ні);
3. (– .) – другому випадку роботи з даними (коли перше значення ще не скоректовано, а друге – скоректованоно);
4. (. .) – третьому випадку роботи з даними (коли обидва значення вже скоректовані).

В колонках 2, 3, 4 жирним шрифтом наведені значення, що вже скоректовані.

З табл. 1 видно, що розрахунок ведеться тільки у тих випадках, коли $i \neq j$ і $n_{1,2,3} \neq i \neq j$. У результаті цього кількість операцій додавання ($V_+$) при використанні (10) зменшиться, у порівнянні з (4), і буде визначатися в кожній ітерації з виразу:

$$V_+ = V^3 - V^2 - V \times [(V-1) \times (V - (V-2))] = V^3 - V^2 - 2 \times V \times (V-1), \quad (12)$$

де $V^3$ – повна кількість можливих розрахунків; $V^2$ – кількість розрахунків, які необхідно було б зробити у випадку, коли $i = j$; $V \times [(V-1) \times (V-(V-2))]$ – кількість розрахунків, які не обов'язково виконувати для пошуку $(V-1)$ елементів ($a_{ij}$) у $V$ рядках матриці за умови, якщо виконується одна з рівностей: $n = i$ або $n = j$.

Кількість операцій пошуку мінімуму ($V_{min}$) для даного алгоритму у кожній ітерації, крім першої, можна визначити з виразу:

$$V_{min} = V^2 - V, \quad (13)$$

при цьому кожна операція пошуку мінімуму здійснюється з $(V-2)$ значень, тобто на $(V-(V-2))$ значень менше, ніж в алгоритмі (4–5).

Загальна обчислювальна складність алгоритму, побудованого завдяки способу, що розглядається, буде дорівнювати:

$$O(V_{алг}) = O(T \times ([V^3 - V^2 - 2 \times V \times (V-1)]_+ + [V^2 - V]_{min})). \quad (14)$$

Для зазначеного прикладу (табл. 1) необхідна кількість розрахунків (операцій додавання згідно з (12)) у кожній ітерації, крім першої, дорівнює:

$$V_+ = 5^3 - 5^2 - 5 \times 2 \times (5-1) = 125 - 25 - 40 = 60.$$

Крім того, з табл. 1 видно, що з даної кількості операцій (60) у 20 випадках проводиться розрахунок зі значеннями, які ще не скоректовані (перший випадок); у 20 випадках (10 + 10) – зі значеннями, одне з яких вже скоректоване (другий випадок); і в 20 випадках – з вже скоректованими значеннями (третій випадок).

При можливості можна показати, що ця тенденція зберігається для будь-якого значення $V$ в мережі ПД і не залежить від її конкретної топології та ваг (довжин) каналів зв'язку в ній. Тобто з всієї кількості потрібних операцій додавання **2/3** операцій завжди будуть здійснюватися у випадках, коли вже одне або обидва значення скоректовані. Це означає, що вже у поточній ітерації можна знаходити найкоротші маршрути, кількість транзитних ділянок в яких значно перевищує значення номера поточної ітерації, а це, в свою чергу, суттєво впливає як на кількість ітерацій при розв'язанні задачі, так і на кількість потрібних операцій додавання та пошуку мінімуму.

Зробимо порівняльну оцінку обчислювальної складності алгоритму (4–5) з алгоритмом (9–11) для конкретного прикладу.

Припустимо, що задана конкретна мережа ПД у вигляді неорієнтованого графа $G(V,Q)$, де $V = 8$ – кількість вузлів комутації в мережі ПД, а $Q = 11$ – кількість дуплексних каналів зв'язку в ній. Даний приклад показаний на рис. 1.

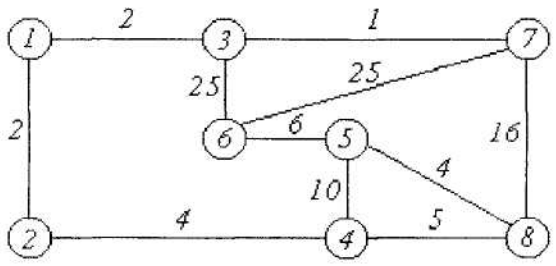


*Рис. 1*

На рисунку цифра в колі визначає номер вузла комутації, а цифра біля дуги – вагу дуплексного каналу зв'язку.

Нижче представлені результати розрахунку даного прикладу за допомогою класичного алгоритму (4–5) у вигляді матриць ( $A$ ) і ($M$) у кожній ітерації.

Слід зауважити, що значення елементів у даних матрицях надані по-різному: ті значення елементів, які відображені без нахилу, означають, що у поточній ітерації в ході розрахунку даних елементів відбулись зміни, тобто їх корекція. В матриці ( $A$ ) ця зміна завжди помітна, так як значення цих елементів у цьому випадку, тобто у поточній і попередній ітераціях, завжди відрізняються. В матриці ( $M$ ) це не завжди відбувається, хоча маршрут і змінився, він все ж таки може проходити через даний останній транзитний вузол. Тому значення елементів в матриці ( $M$ ), які подані без нахилу, остаточно означають, що маршрут змінився незважаючи на те, змінилося чи ні саме значення цього елемента матриці ( $M$ ) у порівнянні з попередньою ітерацією.

Якщо значення елементів матриць ( $A$ ) і ( $M$ ) подані з нахилом, то це свідчить про те, що, незважаючи на їх розрахунок, маршрут не змінився, тобто він, у даному випадку, залишається найкоротшим, а тому на його розрахунок були марно витрачені час і обчислювальні ресурси.

$$A^{(1)} = \begin{Vmatrix} * & 2 & 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & * & 0 & 4 & 0 & 0 & 0 & 0 \\ 2 & 0 & * & 0 & 0 & 25 & 1 & 0 \\ 0 & 4 & 0 & * & 10 & 0 & 0 & 5 \\ 0 & 0 & 0 & 10 & * & 6 & 0 & 4 \\ 0 & 0 & 25 & 0 & 6 & * & 25 & 0 \\ 0 & 0 & 1 & 0 & 0 & 25 & * & 16 \\ 0 & 0 & 0 & 5 & 4 & 0 & 16 & * \end{Vmatrix} \quad M^{(1)} = \begin{Vmatrix} * & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & * & 0 & 2 & 0 & 0 & 0 & 0 \\ 3 & 0 & * & 0 & 0 & 3 & 3 & 0 \\ 0 & 4 & 0 & * & 4 & 0 & 0 & 4 \\ 0 & 0 & 0 & 5 & * & 5 & 0 & 5 \\ 0 & 0 & 6 & 0 & 6 & * & 6 & 0 \\ 0 & 0 & 7 & 0 & 0 & 7 & * & 7 \\ 0 & 0 & 0 & 8 & 8 & 0 & 8 & * \end{Vmatrix}$$

$$A^{(2)} = \begin{Vmatrix} * & 2 & 2 & 6 & 0 & 27 & 3 & 0 \\ 2 & * & 4 & 4 & 14 & 0 & 0 & 9 \\ 2 & 4 & * & 0 & 31 & 25 & 1 & 17 \\ 6 & 4 & 0 & * & 9 & 16 & 21 & 5 \\ 0 & 14 & 31 & 9 & * & 6 & 20 & 4 \\ 27 & 0 & 25 & 16 & 6 & * & 25 & 10 \\ 3 & 0 & 1 & 21 & 20 & 25 & * & 16 \\ 0 & 9 & 17 & 5 & 4 & 10 & 16 & * \end{Vmatrix} \quad M^{(2)} = \begin{Vmatrix} * & 1 & 1 & 2 & 0 & 3 & 3 & 0 \\ 2 & * & 1 & 2 & 4 & 0 & 0 & 4 \\ 3 & 1 & * & 0 & 6 & 3 & 3 & 7 \\ 2 & 4 & 0 & * & 8 & 5 & 8 & 4 \\ 0 & 4 & 6 & 8 & * & 5 & 8 & 5 \\ 3 & 0 & 6 & 5 & 6 & * & 6 & 5 \\ 3 & 0 & 7 & 8 & 8 & 7 & * & 7 \\ 0 & 4 & 7 & 8 & 8 & 5 & 8 & * \end{Vmatrix}$$

$$A^{(3)} = \begin{Vmatrix} * & 2 & 2 & 6 & 16 & 27 & 3 & 11 \\ 2 & * & 4 & 4 & 13 & 20 & 5 & 9 \\ 2 & 4 & * & 8 & 21 & 25 & 1 & 17 \\ 6 & 4 & 8 & * & 9 & 15 & 21 & 5 \\ 16 & 13 & 21 & 9 & * & 6 & 20 & 4 \\ 27 & 20 & 25 & 15 & 6 & * & 25 & 10 \\ 3 & 5 & 1 & 21 & 20 & 25 & * & 16 \\ 11 & 9 & 17 & 5 & 4 & 10 & 16 & * \end{Vmatrix} \quad M^{(3)} = \begin{Vmatrix} * & 1 & 1 & 2 & 4 & 3 & 3 & 4 \\ 2 & * & 1 & 2 & 8 & 5 & 3 & 4 \\ 3 & 1 & * & 2 & 8 & 3 & 3 & 7 \\ 2 & 4 & 1 & * & 8 & 5 & 8 & 4 \\ 2 & 4 & 7 & 8 & * & 5 & 8 & 5 \\ 3 & 4 & 6 & 8 & 6 & * & 6 & 5 \\ 3 & 1 & 7 & 8 & 8 & 7 & * & 7 \\ 2 & 4 & 7 & 8 & 8 & 5 & 8 & * \end{Vmatrix}$$

$$A^{(4)} = \begin{Vmatrix} * & 2 & 2 & 6 & 15 & 22 & 3 & 11 \\ 2 & * & 4 & 4 & 13 & 19 & 5 & 9 \\ 2 & 4 & * & 8 & 18 & 25 & 1 & 13 \\ 6 & 4 & 8 & * & 9 & 15 & 9 & 5 \\ 15 & 13 & 18 & 9 & * & 6 & 20 & 4 \\ 22 & 19 & 25 & 15 & 6 & * & 25 & 10 \\ 3 & 5 & 1 & 9 & 20 & 25 & * & 16 \\ 11 & 9 & 13 & 5 & 4 & 10 & 16 & * \end{Vmatrix} \quad M^{(4)} = \begin{Vmatrix} * & 1 & 1 & 2 & 8 & 5 & 3 & 4 \\ 2 & * & 1 & 2 & 8 & 5 & 3 & 4 \\ 3 & 1 & * & 2 & 4 & 3 & 3 & 4 \\ 2 & 4 & 1 & * & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & 8 & * & 5 & 8 & 5 \\ 2 & 4 & 6 & 8 & 6 & * & 6 & 5 \\ 3 & 1 & 7 & 2 & 8 & 7 & * & 7 \\ 2 & 4 & 1 & 8 & 8 & 5 & 8 & * \end{Vmatrix}$$

$$A^{(5)} = \begin{Vmatrix} * & 2 & 2 & 6 & 15 & 21 & 3 & 11 \\ 2 & * & 4 & 4 & 13 & 19 & 5 & 9 \\ 2 & 4 & * & 8 & 17 & 24 & 1 & 13 \\ 6 & 4 & 8 & * & 9 & 15 & 9 & 5 \\ 15 & 13 & 17 & 9 & * & 6 & 19 & 4 \\ 21 & 19 & 24 & 15 & 6 & * & 25 & 10 \\ 3 & 5 & 1 & 9 & 19 & 25 & * & 14 \\ 11 & 9 & 13 & 5 & 4 & 10 & 14 & * \end{Vmatrix} \quad M^{(5)} = \begin{Vmatrix} * & 1 & 1 & 2 & 8 & 5 & 3 & 4 \\ 2 & * & 1 & 2 & 8 & 5 & 3 & 4 \\ 3 & 1 & * & 2 & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & * & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & 8 & * & 5 & 3 & 5 \\ 2 & 4 & 1 & 8 & 6 & * & 6 & 5 \\ 3 & 1 & 7 & 2 & 4 & 7 & * & 4 \\ 2 & 4 & 1 & 8 & 8 & 5 & 3 & * \end{Vmatrix}$$

$$A^{(6)} = \begin{Vmatrix} * & 2 & 2 & 6 & 15 & 21 & 3 & 11 \\ 2 & * & 4 & 4 & 13 & 19 & 5 & 9 \\ 2 & 4 & * & 8 & 17 & 23 & 1 & 13 \\ 6 & 4 & 8 & * & 9 & 15 & 9 & 5 \\ 15 & 13 & 17 & 9 & * & 6 & 18 & 4 \\ 21 & 19 & 23 & 15 & 6 & * & 25 & 10 \\ 3 & 5 & 1 & 9 & 18 & 25 & * & 14 \\ 11 & 9 & 13 & 5 & 4 & 10 & 14 & * \end{Vmatrix} \quad M^{(6)} = \begin{Vmatrix} * & 1 & 1 & 2 & 8 & 5 & 3 & 4 \\ 2 & * & 1 & 2 & 8 & 5 & 3 & 4 \\ 3 & 1 & * & 2 & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & * & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & 8 & * & 5 & 3 & 5 \\ 2 & 4 & 1 & 8 & 6 & * & 6 & 5 \\ 3 & 1 & 7 & 2 & 8 & 7 & * & 4 \\ 2 & 4 & 1 & 8 & 8 & 5 & 3 & * \end{Vmatrix}$$

$$A^{(7)} = \begin{Vmatrix} * & 2 & 2 & 6 & 15 & 21 & 3 & 11 \\ 2 & * & 4 & 4 & 13 & 19 & 5 & 9 \\ 2 & 4 & * & 8 & 17 & 23 & 1 & 13 \\ 6 & 4 & 8 & * & 9 & 15 & 9 & 5 \\ 15 & 13 & 17 & 9 & * & 6 & 18 & 4 \\ 21 & 19 & 23 & 15 & 6 & * & 24 & 10 \\ 3 & 5 & 1 & 9 & 18 & 24 & * & 14 \\ 11 & 9 & 13 & 5 & 4 & 10 & 14 & * \end{Vmatrix} \quad M^{(7)} = \begin{Vmatrix} * & 1 & 1 & 2 & 8 & 5 & 3 & 4 \\ 2 & * & 1 & 2 & 8 & 5 & 3 & 4 \\ 3 & 1 & * & 2 & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & * & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & 8 & * & 5 & 3 & 5 \\ 2 & 4 & 1 & 8 & 6 & * & 3 & 5 \\ 3 & 1 & 7 & 2 & 8 & 5 & * & 4 \\ 2 & 4 & 1 & 8 & 8 & 5 & 3 & * \end{Vmatrix}$$

$$A^{(8)} = \begin{Vmatrix} * & 2 & 2 & 6 & 15 & 21 & 3 & 11 \\ 2 & * & 4 & 4 & 13 & 19 & 5 & 9 \\ 2 & 4 & * & 8 & 17 & 23 & 1 & 13 \\ 6 & 4 & 8 & * & 9 & 15 & 9 & 5 \\ 15 & 13 & 17 & 9 & * & 6 & 18 & 4 \\ 21 & 19 & 23 & 15 & 6 & * & 24 & 10 \\ 3 & 5 & 1 & 9 & 18 & 24 & * & 14 \\ 11 & 9 & 13 & 5 & 4 & 10 & 14 & * \end{Vmatrix} \quad M^{(8)} = \begin{Vmatrix} * & 1 & 1 & 2 & 8 & 5 & 3 & 4 \\ 2 & * & 1 & 2 & 8 & 5 & 3 & 4 \\ 3 & 1 & * & 2 & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & * & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & 8 & * & 5 & 3 & 5 \\ 2 & 4 & 1 & 8 & 6 & * & 3 & 5 \\ 3 & 1 & 7 & 2 & 8 & 5 & * & 4 \\ 2 & 4 & 1 & 8 & 8 & 5 & 3 & * \end{Vmatrix}$$

Основні показники розрахунку даного прикладу за допомогою звичайного алгоритму (4–5) наведені у табл. 2.

*Таблиця 2*

| $T$ | $K$ $(V^2 - V)$ | $V_{\min}$ | *Марно* | | *Корисно* | | $V_t$ | | | | | *ОД БІТ* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | *K1* | % | *K2* | % | $\Sigma K \times V$ | *Марно* | | *Корисно* | | |
| | | | | | | | | *K3* | % | *K4* | % | |
| 1 | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | 512 |
| 2 | 56 | 56 | 30 | 53,6 | 26 | 46,4 | 448 | 240 | 53,6 | 208 | 46,4 | 512 |
| 3 | 56 | 56 | 40 | 71,4 | 16 | 28,6 | 448 | 320 | 71,4 | 128 | 28,6 | 512 |
| 4 | 56 | 56 | 44 | 78,6 | 12 | 21,4 | 448 | 352 | 78,6 | 96 | 21,4 | 512 |
| 5 | 56 | 56 | 46 | 82,1 | 10 | 17,9 | 448 | 368 | 82,1 | 80 | 17,9 | 512 |
| 6 | 56 | 56 | 52 | 92,9 | 4 | 7,1 | 448 | 416 | 92,9 | 32 | 7,1 | 512 |
| 7 | 56 | 56 | 54 | 96,4 | 2 | 3,6 | 448 | 432 | 96,4 | 16 | 3,6 | 512 |
| 8 | 56 | 56 | 56 | 100 | 0 | 0 | 448 | 448 | 100 | 0 | 0 | 512 |
| $\Sigma$ | 392 | 392 | 322 | **82,1** | 70 | **17,9** | 3136 | 2576 | 82,1 | 560 | 17,9 | 4096 |

В даній таблиці такі наступні позначення:

$T$ – номер ітерації; $K$ – кількість елементів, що потрібно розраховувати в кожній ітерації; $K1$ – кількість елементів із кількості $K$, що розраховуються марно, тобто корекція їх значень не здійснюється; $K2$ – кількість елементів із $K$, в результаті розрахунку яких здійснюється корекція (знаходяться нові найкоротші маршрути); $K3$ – кількість марно здійснених операцій

додавання у кожній ітерації; *К4* – кількість операцій додавання, що призвели до корекції (*К2*) елементів матриці ( *A* ); *ОД* – обсяг даних, який потрібно зберегти в пам'яті ЕОМ (із розрахунку, що для запам'ятовування кожного значення елемента матриці ( *A* ) для простоти узятий тільки один байт (8 біт), тобто, для запам'ятовування кожної матриці ( *A* ) потрібно $V^2 \times 8 = 64 \times 8 = 512$ (біт).

Аналіз отриманих результатів показує, що навіть для не дуже складної задачі при розрахунку її алгоритмом (4–5) марно витрачається досить багато часу та обчислювальних ресурсів; тільки 17,9 % від загальної кількості потрібних операцій додавання здійснюється на користь, а інші – марно. Крім того, велика кількість потрібних для вирішення задачі ітерацій призводить до того, що потрібно запам'ятати дуже великий обсяг проміжної інформації, яка, з точки зору споживача, не є корисною.

Тепер розрахуємо цей приклад (задачу) за допомогою розробленого алгоритму.

Нижче наведені результати розрахунку цієї задачі за допомогою даного алгоритму у вигляді матриць ( *A* ) і ( *M* ) в кожній ітерації.

$$A^{(1)} = \begin{Vmatrix} * & 2 & 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & * & 0 & 4 & 0 & 0 & 0 & 0 \\ 2 & 0 & * & 0 & 0 & 25 & 1 & 0 \\ 0 & 4 & 0 & * & 10 & 0 & 0 & 5 \\ 0 & 0 & 0 & 10 & * & 6 & 0 & 4 \\ 0 & 0 & 25 & 0 & 6 & * & 25 & 0 \\ 0 & 0 & 1 & 0 & 0 & 25 & * & 16 \\ 0 & 0 & 0 & 5 & 4 & 0 & 16 & * \end{Vmatrix} \quad M^{(1)} = \begin{Vmatrix} * & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & * & 0 & 2 & 0 & 0 & 0 & 0 \\ 3 & 0 & * & 0 & 0 & 3 & 3 & 0 \\ 0 & 4 & 0 & * & 4 & 0 & 0 & 4 \\ 0 & 0 & 0 & 5 & * & 5 & 0 & 5 \\ 0 & 0 & 6 & 0 & 6 & * & 6 & 0 \\ 0 & 0 & 7 & 0 & 0 & 7 & * & 7 \\ 0 & 0 & 0 & 8 & 8 & 0 & 8 & * \end{Vmatrix}$$

$$A^{(1)} = \begin{Vmatrix} * & 2 & 2 & 6 & 16 & 22 & 3 & 11 \\ 2 & * & 4 & 4 & 14 & 20 & 5 & 9 \\ 2 & 4 & * & 8 & 18 & 24 & 1 & 13 \\ 6 & 4 & 8 & * & 9 & 15 & 9 & 5 \\ 16 & 14 & 18 & 9 & * & 6 & 18 & 4 \\ 22 & 20 & 24 & 15 & 6 & * & 24 & 10 \\ 3 & 5 & 1 & 9 & 18 & 24 & * & 14 \\ 11 & 9 & 13 & 5 & 4 & 10 & 14 & * \end{Vmatrix} \quad M^{(1)} = \begin{Vmatrix} * & 1 & 1 & 2 & 4 & 5 & 3 & 4 \\ 2 & * & 1 & 2 & 8 & 5 & 3 & 4 \\ 3 & 1 & * & 2 & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & * & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & 8 & * & 5 & 3 & 5 \\ 2 & 4 & 1 & 8 & 6 & * & 3 & 5 \\ 3 & 1 & 7 & 2 & 8 & 5 & * & 4 \\ 2 & 4 & 1 & 8 & 8 & 5 & 3 & * \end{Vmatrix}$$

$$A^{(2)} = \begin{Vmatrix} * & 2 & 2 & 6 & 15 & 21 & 3 & 11 \\ 2 & * & 4 & 4 & 13 & 19 & 5 & 9 \\ 2 & 4 & * & 8 & 17 & 23 & 1 & 13 \\ 6 & 4 & 8 & * & 9 & 15 & 9 & 5 \\ 15 & 13 & 17 & 9 & * & 6 & 18 & 4 \\ 21 & 19 & 23 & 15 & 6 & * & 24 & 10 \\ 3 & 5 & 1 & 9 & 18 & 24 & * & 14 \\ 11 & 9 & 13 & 5 & 4 & 10 & 14 & * \end{Vmatrix} \quad M^{(2)} = \begin{Vmatrix} * & 1 & 1 & 2 & 8 & 5 & 3 & 4 \\ 2 & * & 1 & 2 & 8 & 5 & 3 & 4 \\ 3 & 1 & * & 2 & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & * & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & 8 & * & 5 & 3 & 5 \\ 2 & 4 & 1 & 8 & 6 & * & 3 & 5 \\ 3 & 1 & 7 & 2 & 8 & 5 & * & 4 \\ 2 & 4 & 1 & 8 & 8 & 5 & 3 & * \end{Vmatrix}$$

$$A^{(3)} = \begin{Vmatrix} * & 2 & 2 & 6 & 15 & 21 & 3 & 11 \\ 2 & * & 4 & 4 & 13 & 19 & 5 & 9 \\ 2 & 4 & * & 8 & 17 & 23 & 1 & 13 \\ 6 & 4 & 8 & * & 9 & 15 & 9 & 5 \\ 15 & 13 & 17 & 9 & * & 6 & 18 & 4 \\ 21 & 19 & 23 & 15 & 6 & * & 24 & 10 \\ 3 & 5 & 1 & 9 & 18 & 24 & * & 14 \\ 11 & 9 & 13 & 5 & 4 & 10 & 14 & * \end{Vmatrix} \quad M^{(3)} = \begin{Vmatrix} * & 1 & 1 & 2 & 8 & 5 & 3 & 4 \\ 2 & * & 1 & 2 & 8 & 5 & 3 & 4 \\ 3 & 1 & * & 2 & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & * & 8 & 5 & 3 & 4 \\ 2 & 4 & 1 & 8 & * & 5 & 3 & 5 \\ 2 & 4 & 1 & 8 & 6 & * & 3 & 5 \\ 3 & 1 & 7 & 2 & 8 & 5 & * & 4 \\ 2 & 4 & 1 & 8 & 8 & 5 & 3 & * \end{Vmatrix}$$

Основні показники розрахунку даного прикладу за допомогою розробленого алгоритму наведені у табл. 3.

*Таблиця 3*

| $T$ | $K$ $(V^2 - V)$ | $V_{min}$ | *Марно* | | *Корисно* | | $V_r$ | | | | | *ОД БІТ* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | *K1* | % | *K2* | % | $\Sigma$ $K \times V$ | *Марно* | | *Корисно* | | |
| | | | | | | | | *K3* | % | *K4* | % | |
| 1 | 56 | 56 | 14 | 25 | 42 | 75 | 336 | 84 | 25 | 252 | 75 | 512 |
| 2 | 56 | 56 | 44 | 78,6 | 12 | 21,4 | 336 | 264 | 78,6 | 72 | 21,4 | 512 |
| 3 | 56 | 56 | 56 | 100 | 0 | 0 | 336 | 336 | 100 | 0 | 0 | 512 |
| $\Sigma$ | 168 | 168 | 114 | **67,9** | 54 | **32,1** | 1008 | 684 | 67,9 | 324 | 32,1 | 1536 |

Порівнявши отримані результати (табл. 3) з результатами, які були отримані при розрахунку цієї ж задачі за допомогою звичайного алгоритму (табл. 2), видно, що:

1. Значно, більше ніж удвічі, знижується кількість ітерацій, потрібних для вирішення заданої задачі, і, як наслідок цього, також більше ніж удвічі зменшується як кількість елементів, що потрібно розраховувати (з 392 до 168), так і обсяг даних, який потрібно запам'ятати (з 4 096 біт до 1 536 біт – тільки для запам'ятовування матриці ($A$)).

2. Кількість потрібних операцій додавання для розрахунку найкоротших маршрутів дуже потужно, більше ніж утричі, знижується завдяки використанню розробленого способу (з 3 136 операцій до 1 008 при використанні розробленого алгоритму).

3. Кількість корисних операцій додавання при розрахунку найкоротших маршрутів завдяки розробленому алгоритму складає 32,1 %, що практично майже на 14 % більше, ніж при використанні звичайного алгоритму (17,9 %).

Слід зазначити, що отримані результати відносяться лише до даного конкретного прикладу, тобто при розрахунку інших задач результати будуть іншими, але, як показують результати математичного моделювання та практичного розрахунку за допомогою ПЕОМ різних за структурою (топологією) мереж ПД, поданих у вигляді графів, ці результати значно не відрізняються.

Таким чином, запропонований спосіб обліку поточних значень для розрахунку найкоротших маршрутів передачі інформації заслуговує на увагу і може бути практично використаний в якості окремого алгоритму у більш загальному алгоритмі адаптивної маршрутизації в мережах ПД спеціального призначення.

Однак суттєвим недоліком цього способу є те, що на початку кожної ітерації $T$ (крім першої) потрібно проводити додатковий етап присвоювання (9) елементів матриць $A^{(T)}$ і $M^{(T)}$ елементам цих же матриць, отриманих у попередній $(T-1)$ ітерації.

Зрозуміло, що, хоча операція присвоювання за потужністю й часом виконання, і є найпростішою, все ж таки їх може бути досить багато, особливо при зростанні кількості ітерацій для розв'язання задачі.

Другим недоліком даного способу є те, що він не враховує випадків, коли варто, чи не варто, розраховувати кожний елемент матриць ($A$ і $M$), можливо деякі елементи доцільно залишити без змін і не витрачати марно час на їх корекцію. Тому при розробці алгоритму пошуку найкоротших маршрутів передачі інформації в мережах ПД спеціального призначення необхідно додатково розробляти та використовувати нові способи, які б дозволили при спільному використанні їх з розглянутим в статті способом суттєво послабити вищевказані недоліки.

**ЛІТЕРАТУРА:**

1. *Девис Д., Барбер Д., Прайс У., Соломонидес С.* Вычислительные сети и сетевые протоколы / Под ред. С.И. Самойленко: Пер. с англ. – М.: Мир, 1982.
2. *Лазарев В.Г., Лазарев Ю.В.* Динамическое управление потоками информации в сетях связи. – М.: Радио и связь, 1983.
3. *Бертсекас Д., Галлагер Р.* Сети передачи данных: Пер. с англ. – М.: Мир, 1989.
4. *Зайченко Ю.П, Гонта Ю.В.* Структурная оптимизация сетей ЭВМ. – К.: Техника, 1986.

ПОНОМАРЬОВ Юрій Володимирович – начальник редакційно-видавничого відділу Житомирського військового інституту радіоелектроніки імені С.П. Корольова.

Наукові інтереси:

– методи та алгоритми маршрутизації передачі інформації у мережах передачі даних з комутацією пакетів.

Подано 23.12.1999.