УДК 681.3

С.Д. ПОГОРІЛИЙ, О.О. КАМАРДІНА, О.І. БАВИКІН

ПРО ПІДХІД ДО РОЗПАРАЛЕЛЮВАННЯ АЛГОРИТМУ ФЛОЙДА-УОРШАЛЛА

Abstract: *Floyd-Warshall's algorithm formalization is executed with the use of mathematical means of the systems of algorithmic algebras modified. Conversion strategies of basic algorithm into a parallel one are offered and the parallel regular chart of algorithm is obtained. The chain of equivalent transformations is executed and the spectrum of the modified charts of Floyd-Warshall's algorithm is got.*
Key words: *routing, routing protocols, Floyd-Warshall's algorithm, data parallelism, parallel regular chart of algorithm, schemes' equivalent transformations, process.*

Анотація: *Виконано формалізацію алгоритму Флойда-Уоршалла з використанням математичного апарату систем алгоритмічних алгебр модифікованих. Запропоновано стратегії розпаралелювання та одержано паралельну регулярну схему алгоритму. Виконано низку еквівалентних перетворень і отримано спектр модифікованих схем алгоритму Флойда-Уоршалла.*
Ключові слова: *маршрутизація, протоколи маршрутизації, алгоритм Флойда-Уоршалла, розпаралелювання за даними, паралельна регулярна схема алгоритму, еквівалентні перетворення схем, процес.*

Аннотация: *Выполнена формализация алгоритма Флойда-Уоршалла с использованием математического аппарата систем алгоритмических алгебр модифицированных. Предложены стратегии распараллеливания и получена параллельная регулярная схема алгоритма. Выполнена цепь эквивалентных преобразований и получен спектр модифицированных схем алгоритма Флойда-Уоршалла.*
Ключевые слова: *маршрутизация, протоколы маршрутизации, алгоритм Флойда-Уоршалла, распараллеливание по данным, параллельная регулярная схема алгоритма, эквивалентные преобразования схем, процесс.*

1. Вступ

Актуальною задачею мереж є задача маршрутизації і ефективність їх функціонування, що значною мірою визначається методом розв'язання цієї задачі. Дослідження в цій галузі проводяться вже тривалий час і деякі їх результати розглянуто в [1–7].

Незважаючи на велику увагу, яку приділяють проблемі маршрутизації, всі сучасні алгоритми мають певні недоліки:

– неефективність використання у великих мережах;

– протокол, в основу якого покладено той чи інший алгоритм, функціонує в межах домену або між доменами;

– алгоритм використовує фіксовану метрику;

– алгоритм не враховує альтернативні маршрути тощо [1–3].

Крім того, задача маршрутизації в Internet розв'язується мільйони разів на добу. Тому час виконання пошуку оптимального маршруту стає критичним параметром алгоритмів маршрутизації. Вимагається максимально можливе підвищення продуктивності.

У зв'язку з цим група з розвитку Internet IRTF (Internet Research Task Force), яка координує довгострокові дослідницькі проекти за стеком протоколів TCP/IP, займається створенням нових протоколів та пошуком нових, перспективних методів розв'язання задачі маршрутизації для подолання існуючих недоліків на якісно новому рівні.

Алгоритм Флойда-Уоршалла розглядається як альтернативний варіант застосування в перспективних протоколах маршрутизації (Floyd-Warshall) [4, 5]. Це обумовлено тим, що він:

– є ефективним на щільних графах;

– дозволяє знайти найкоротші шляхи для всіх пар вершин орієнтованого графа;

– час знаходження найкоротших шляхів у графі, що має n вершин, є сталим і пропорційний n^3.

Існує багато архітектур, що можуть забезпечити підвищення швидкодії алгоритму. Важливим класом серед них є паралельні.

Метою цієї роботи є мінімізація часу виконання алгоритму Флойда-Уоршалла за рахунок його розпаралелювання. Для цього в роботі запропоновано застосування математичного апарату модифікованих систем алгоритмічних алгебр В.М. Глушкова (САА-М).

2. Формалізація алгоритму Флойда-Уоршалла

Алгоритм Флойда-Уоршалла (детально алгоритм розглянуто у [4, 5]) дозволяє розв'язати задачу знаходження найкоротшого шляху для мережі, що містить n вершин, за n^3 ітерацій. Алгоритм не забороняє ребра від'ємної ваги, але виключає цикли від'ємної ваги. Для алгоритму важливо, які вершини використовуються як проміжні. Характеристикою шляху є максимальний номер проміжної вершини (у фіксованій нумерації вершин) [4].

Алгоритм використовує три матриці:

1. W – матриця суміжності, що відображає топологію мережі.
2. D – матриця ваги найкоротших шляхів:
   - d_{ij} – елемент матриці D, що знаходиться в i-тому стовпчику та j-тому рядку;
   - $D^{(0)}$ – матриця, всі елементи якої визначаються за правилом: $d_{ij}^0 = Вага\ ребра(i, j)\ i \neq j,$ якщо у графі існує ребро (дуга) (i, j);
   - вазі ребра d_{ij}^k на k-тому кроці присвоюється найменше з двох можливих значень:
     - вага ребра (i, j) на $k-1$ кроці ($d_{ij}^{(k-1)}$);
     - вага суми ребер $(i,k)+(k, j)$ на $k-1$ кроці $(d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$;
   - $D^{(n)}$ – результуюча матриця ваги найкоротших шляхів.
3. Π – матриця передування, що містить найкоротші шляхи:
   - π_{ij} – елемент матриці Π, що знаходиться в i-тому стовпчику та j-тому рядку;
   - $\pi_{ij}^{(k)}$ – вершина, яка передує вершині j на найкоротшому шляху з вершини i у вершину j з проміжними вершинами з множини $\{1,2,...,k\}$;
   - $\Pi^{(0)}$ – матриця, всі елементи якої визначаються за правилом:
     $$\pi_{ij}^{(0)} \begin{cases} nil, якщо\ i = j\ або\ вага\ ребра(i, j)\ дорівнює\ \infty; \\ i, якщо\ i \neq j\ та\ вага\ ребра(i, j) < \infty. \end{cases}$$

Процедура, яка реалізує послідовний алгоритм Флойда-Уоршалла, виглядає так [4]:

Floyd-Warshall (W)

$n \leftarrow rows[W]$

$D^{(0)} \leftarrow W$

$for\ k \leftarrow 1\ to\ n$

$do\ \ for\ i \leftarrow 1\ to\ n$

$do\ \ for\ j \leftarrow 1\ to\ n$

$$do\ d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$$

$$(\pi_{ij}^{(k)} = \pi_{ij}^{(k-1)}) \vee (\pi_{ij}^{(k)} = \pi_{kj}^{(k-1)})$$

$return\ D^{(n)}$.

Входом даної процедури є матриця W ваги ребер графа $n \times n$.

Виходом є результуюча матриця $D^{(n)}$ ваги найкоротших шляхів.

Підходи до одержання поряд з матрицею $D^{(n)}$ самих шляхів розглянуті в [4].

Нижче наведено приклад мережі з дев'ятьма вузлами. Ваги ребер представлені на рис. 1.

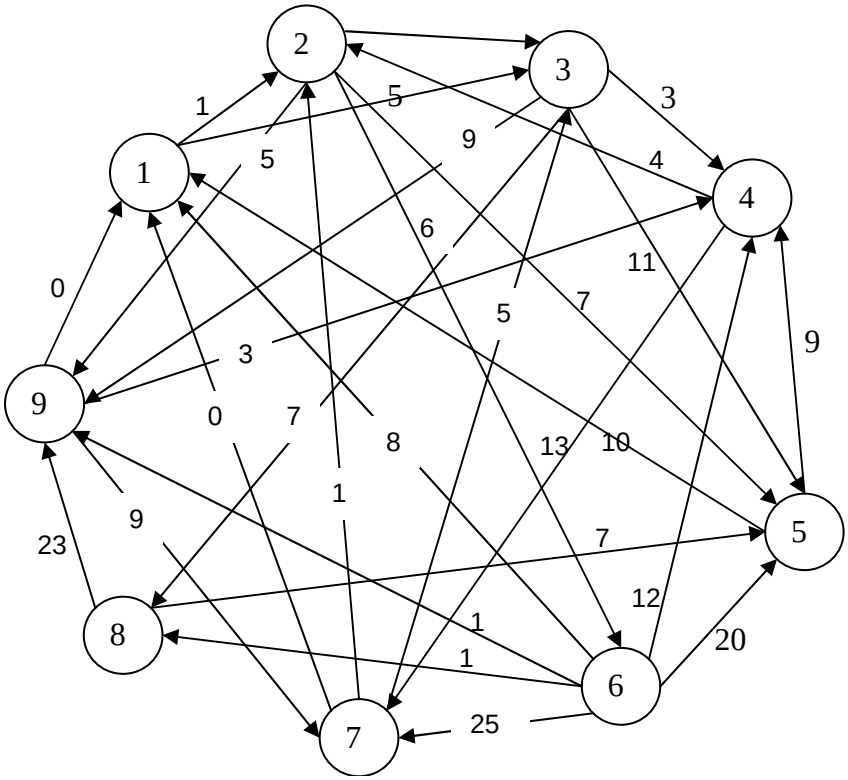


Рис. 1. Граф, що ілюструє мережу з дев'яти вузлів

Матриця суміжності $D^{(0)}$, що відображає цей граф, та матриця передування $\Pi^{(0)}$ мають такий вигляд:

$$D^{(0)}=\begin{pmatrix}0&1&5&\infty&\infty&\infty&\infty&\infty&\infty\\ \infty&0&13&\infty&7&6&\infty&\infty&5\\ \infty&\infty&0&3&11&\infty&\infty&7&9\\ \infty&4&\infty&0&\infty&\infty&13&\infty&\infty\\ 10&\infty&\infty&9&0&\infty&\infty&\infty&\infty\\ 8&\infty&\infty&12&20&0&25&1&1\\ 0&1&5&\infty&\infty&\infty&0&\infty&\infty\\ \infty&\infty&\infty&\infty&7&\infty&\infty&0&23\\ 0&\infty&\infty&3&\infty&\infty&9&\infty&0\end{pmatrix};\ \Pi^{(0)}=\begin{pmatrix}nil&1&1&nil&nil&nil&nil&nil&nil\\ nil&nil&2&nil&2&2&nil&nil&2\\ nil&nil&nil&3&3&nil&nil&3&3\\ nil&4&nil&nil&nil&nil&4&nil&nil\\ 5&nil&nil&5&nil&nil&nil&nil&nil\\ 6&nil&nil&6&6&nil&6&6&6\\ 7&7&7&nil&nil&nil&nil&nil&nil\\ nil&nil&nil&nil&8&nil&nil&nil&8\\ 9&nil&nil&9&nil&nil&9&nil&nil\end{pmatrix}.$$

Матриці $D^{(9)}$ та $\Pi^{(9)}$, які є результатом роботи алгоритму Флойда-Уоршалла, для вищезазначеного графа такі:

$$D^{(9)}=\begin{pmatrix}0&1&5&8&8&7&15&8&6\\ 5&0&10&8&7&6&14&7&5\\ 9&7&0&3&11&13&16&7&9\\ 9&4&14&0&11&10&13&11&9\\ 10&11&15&9&0&17&22&18&16\\ 1&2&6&4&8&0&10&1&1\\ 0&1&5&8&8&7&0&8&6\\ 17&18&22&16&7&24&29&0&23\\ 0&1&5&3&8&7&9&8&0\end{pmatrix};\ \Pi^{(9)}=\begin{pmatrix}nil&1&1&3&2&2&9&6&2\\ 9&nil&9&9&2&2&9&6&2\\ 9&4&nil&3&3&4&4&3&3\\ 9&4&9&nil&2&2&4&6&2\\ 5&1&1&5&nil&2&4&6&2\\ 9&9&9&9&8&nil&9&6&6\\ 7&7&7&3&2&2&nil&6&2\\ 5&5&5&5&8&5&5&nil&8\\ 9&1&1&9&2&2&9&6&nil\end{pmatrix}.$$

Для дослідження алгоритму, трансформації та розпаралелювання виконаємо його формалізацію. Запишемо алгоритм у вигляді параметричної регулярної схеми з використанням модифікованих систем алгоритмічних алгебр САА-М.

Формалізована версія алгоритму має такий вигляд (V – множина вершин графа):

$A0$

$(n=card(V))*$

$*(D^{(0)}=W)*$

$*(\textit{формування } \Pi^{(0)})*$

$$*\left(\underset{k\,in\,V}{\{}\ \underset{i\,in\,V}{\{}\ \underset{j\,in\,V}{\{}\ (d_{ij}^{(k)}=\min(d_{ij}^{(k-1)},d_{ik}^{(k-1)}+d_{kj}^{(k-1)})*((\underset{d_{ij}^{(k-1)}\le d_{ik}^{(k-1)}+d_{kj}^{(k-1)}}{\Pi_{ij}^{(k)}=\Pi_{ij}^{(k-1)}})\vee(\Pi_{ij}^{(k)}=\Pi_{kj}^{(k-1)})*\right.$$

$*(j\ \textit{наступний вузол})\}*$

$*(i\ \textit{наступний вузол})\}*$

$\left.*(k\ \textit{наступний вузол})\}\right)*$

$*(return\ D^{(n)},\Pi^{(k)}),$

де оператори:

- $n = card(V)$;
- $D^{(0)} = W$;
- формування $\Pi^{(0)}$;
- $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$;
- $\Pi_{ij}^{(k)} = \Pi_{ij}^{(k-1)}$;
- $\Pi_{ij}^{(k)} = \Pi_{kj}^{(k-1)}$;
- (j =наступний вузол);
- (i =наступний вузол);
- (k =наступний вузол);
- $return\ D^{(n)}, \Pi^{(k)}$.

Предикати:

- $i\ in\ V$;
- $j\ in\ V$;
- $k\ in\ V$;
- $d_{ij}^{(k-1)} \le d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$.

Застосуємо САА-М для оптимізації алгоритму Флойда-Уоршалла.

3. Розпаралелювання алгоритму

Можливі різні підходи до еквівалентних перетворень вихідного алгоритму (рис. 2).

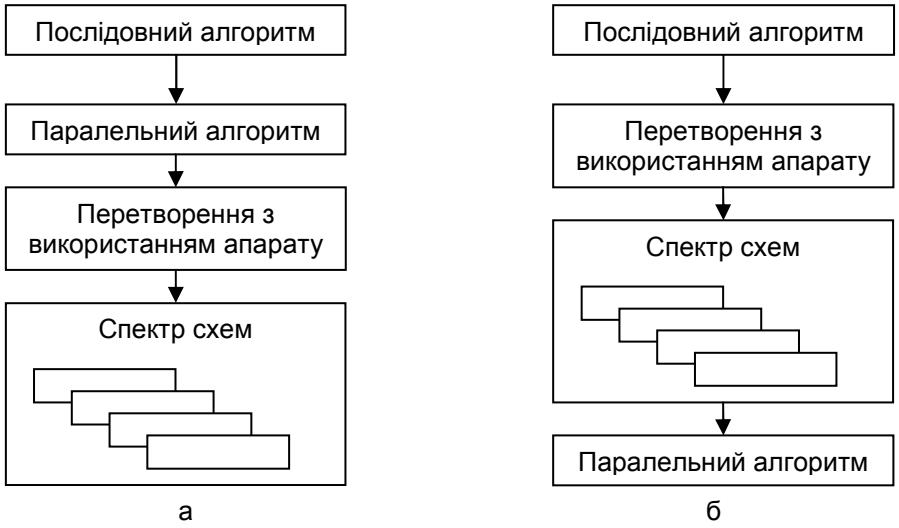


Рис. 2. Підходи до еквівалентних перетворень вихідного алгоритму

Підхід, що відповідає рис. 2а, є більш доцільним, тому що алгоритм проектується з урахуванням майбутньої роботи на паралельній архітектурі і враховуватиме її специфіку.

Підхід, що відповідає рис. 2б, буде направлений на створення алгоритму для послідовної архітектури. Наступне «перетворення» такої схеми в паралельну не буде враховувати деяких особливостей паралельної архітектури, як при першому підході.

Першим кроком у перетворенні вихідного послідовного алгоритму в паралельний є розробка стратегії розпаралелювання. Виходячи з парадигм паралельного програмування, одним із видів паралелізму є паралелізм за даними.

Відповідно до цього розглянемо один з можливих варіантів розпаралелювання алгоритму Флойда-Уоршалла, який базується на однорозмірному розбитті матриці D по L процесорах.

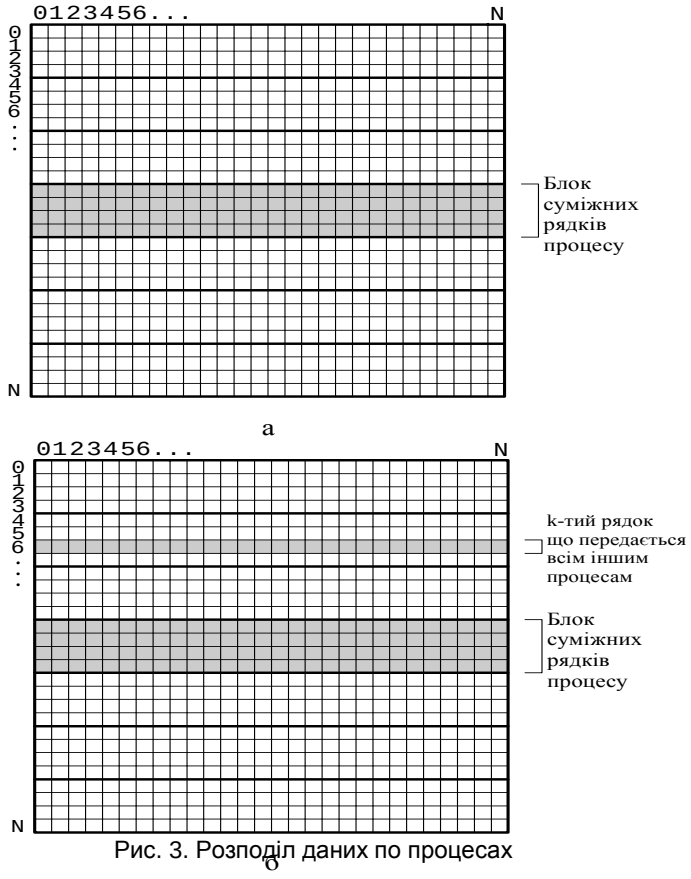


Рис. 3. Розподіл даних по процесах

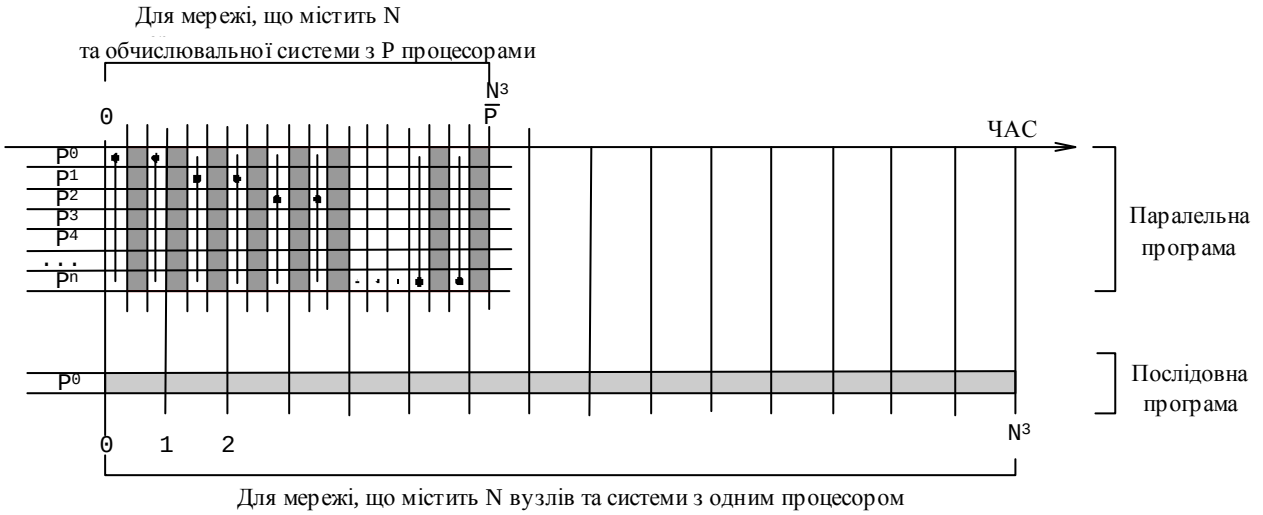
Кожний процесор обслуговуватиме «свій» процес. Необхідно розподілити по блоках суміжні рядки матриці $D^{(0)}$ по L процесах. Для кожного такого блока визначається номер першого рядка блока та кількість рядків у блоці. На рис. 3а наведено розміщення даних одного процесу у вигляді блока суміжних рядків.

На рис. 3б зображено k-тий крок алгоритму із власним блоком суміжних рядків процесу та k-тим рядком, що передається з іншого процесу. На кожному кроці один з процесів передає один рядок всім іншим процесам. Спочатку нульовий процес $P^{(0)}$ передає свій перший рядок, після чого всі процеси виконують операцію $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$. Потім $P^{(0)}$ передає наступний рядок і т.д. Коли процес $P^{(0)}$ передасть усі свої рядки, починає передачу процес $P^{(1)}$. Процес закінчується, коли останній процес передасть свій останній рядок. Спочатку нульовий процес $P^{(0)}$ передає свій перший рядок, після чого всі процеси виконують операцію $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$. Потім $P^{(0)}$ передає наступний рядок і т.д. Коли процес $P^{(0)}$ передасть усі свої рядки, починає передачу процес $P^{(1)}$. Процес закінчується, коли останній процес передасть свій останній рядок.

Графічно цей підхід та його відмінність від послідовного варіанта показано на рис. 4.

Розглянемо інший підхід до розпаралелювання алгоритму, що базується на дворозмірному розбитті матриці D. Матриця $D^{(k)}$ розбивається на *p* блоків розміром $(n/\sqrt{p}) \times (n/\sqrt{p})$. Кожен блок оброблятиметься одним з *p* процесів. Процесу P_{ij} відповідає блок матриці $D^{(k)}$, верхній лівий кут якого $((i-1)n/\sqrt{p}+1, (j-1)n/\sqrt{p}+1)$, а правий нижній кут – $(in/\sqrt{p},\ jn/\sqrt{p})$. Кожний процес модифікує «свою» частину матриці протягом ітерації. На рис. 5 зображено техніку дворозмірного розбиття

матриці.



де

чергова ітерація послідовного алгоритму;

чергова ітерація паралельного алгоритму;

занесення до буфера рядка для передачі іншим процесам;

отримання рядка з буфера;

Pn процес за номером n.

Рис. 4. Паралельний та послідовний варіанти алгоритму

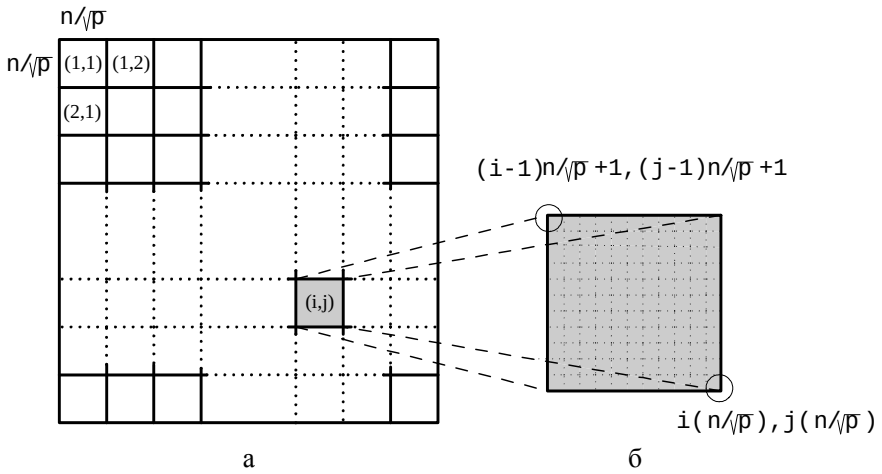


Рис. 5. Розподіл даних по процесах при дворозмірному розбитті матриці D

На k-тій ітерації алгоритму кожен процес P_{ij} потребує певних сегментів k-того рядка та k-того стовпчика матриці $D^{(k-1)}$. Наприклад, для обчислення $d_{l,r}^{(k)}$ процес повинен отримати блоки $d_{l,k}^{(k-1)}$ та $d_{k,r}^{(k-1)}$.

Як показано на рис. 6, блок $d_{l,k}^{(k-1)}$ належить процесу, розташованому в тому ж рядку, а блок $d_{k,r}^{(k-1)}$ – процесу, розташованому в тому самому стовпчику, що і процес P_{ij}. Дані між процесами

передаються таким чином: на k-тій ітерації кожний з $\sqrt{p}$ процесів, маючи тільки частину k-того рядка, посилає її $\sqrt{p}-1$-шому процесу, що знаходиться в k-тому стовпчику. Відповідно кожний з $\sqrt{p}$ процесів, маючи тільки частину k-того стовпчика, посилає її $\sqrt{p}-1$-шому процесу, що знаходиться в k-тому рядку.

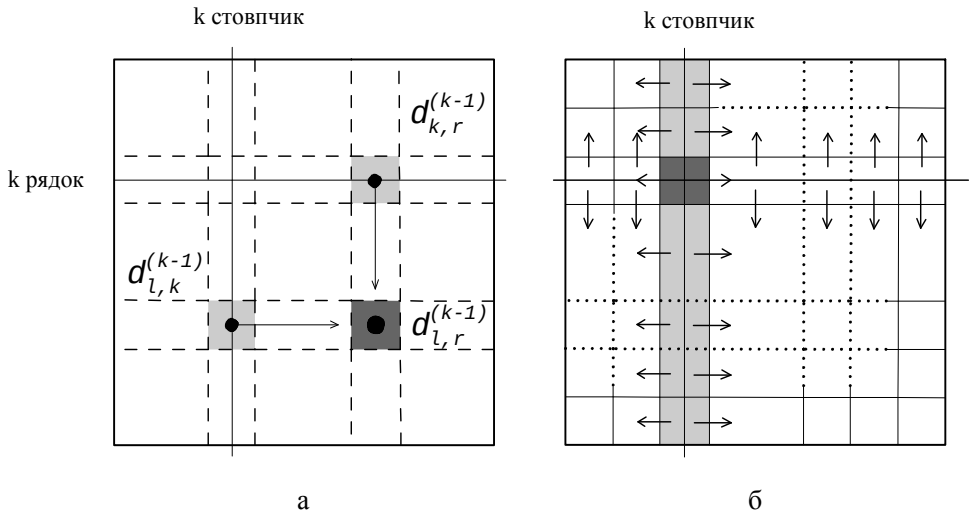


Рис. 6. Пересилки даних між процесами

Інший підхід ґрунтується на факті відсутності у внутрішньому циклі процедури, що реалізує алгоритм Флойда-Уоршалла, залежності за даними. Таким чином, всі ітерації цього циклу можна виконувати паралельно.

4. Еквівалентні перетворення паралельної регулярної схеми алгоритму

$A1$

$(n = card(V))*$

$*(D^{(0)} = W)*$

$*(формування\ \Pi^{(0)})*$

$$*\left(\{\ \#(P_i = buf)\#* \#\{ \{ (d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})*((\Pi_{ij}^{(k)} = \Pi_{ij}^{(k-1)}) \vee (\Pi_{ij}^{(k)} = \Pi_{kj}^{(k-1)})* \right.$$

$$\left. *(j\ наступний\ вузол)\}*(i\ наступний\ вузол)\}\#*(k\ наступний\ вузол)\}) *(return\ D^{(n)}, \Pi^{(k)}),$$

де nrl – номер першого рядка у блоці;

nr – загальна кількість рядків у блоці.

В результаті перетворення схеми послідовного алгоритму до схеми паралельного видно, що складність алгоритму залишилась на рівні V^3, але кількість ітерацій одного з внутрішніх циклів зменшилась.

Формалізуємо деякі фрагменти схеми А0.1. Введемо таку систему позначень:

$$d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) ::= ([d_{ij}^{(k-1)} < d_{ik}^{(k-1)} + d_{kj}^{(k-1)}] d_{ij}^{(k)} = d_{ij}^{(k-1)}, d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}.$$

Для спрощення сприйняття та наступних перетворень одразу ж перейдемо до неінтерпретованої схеми:

$$d_{ij}^{(k-1)} < d_{ik}^{(k-1)} + d_{kj}^{(k-1)} ::= U_1;$$

$$d_{ij}^{(k)} = d_{ij}^{(k-1)} ::= A;$$

$$d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)} ::= B.$$

Тоді прийдемо до такої короткої форми:

$$d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) ::= ([U_1]A, B).$$

Позначимо:

$$P_i = buf ::= F;$$

$$d_{ij}^{(k)} \le d_{ik}^{(k-1)} + d_{kj}^{(k-1)} ::= U_2;$$

$$\Pi_{ij}^{(k)} = \Pi_{ij}^{(k-1)} ::= G;$$

$$\Pi_{ij}^{(k)} = \Pi_{kj}^{(k-1)} ::= H;$$

$$k \; in \; V ::= U_3;$$

$$j \; in \; V ::= U_4;$$

$$i < nrl + nr ::= U_5;$$

$$return \; D^{(n)}, \Pi^{(k)} ::= K.$$

З урахуванням наведених інтерпретацій перейдемо від частково інтерпретованої схеми А0.1 до неінтерпретованої схеми А0.2:

$A0.2$

$\{[U_3]\#F\#*$

$\quad \#\{[U_5]$

$\quad\quad \{[U_4]([U_1]A, B) * ([U_2](G \vee H), E)\} *$

$\quad \}\#*$

$\} * K.$

Введемо ще одне позначення:

$([U_1]A, B) * ([U_2](G \vee H), E) ::= N;$

$A0.2.1$

$\{[U_3]\#F\#*\#\{[U_5]\{[U_4]N\}\}\#\} * K.$

В результаті таких перетворень ми прийшли до неінтерпретованої схеми. Скористаємося властивостями операторних конструкцій САА-М [8].

$$[\lambda]A * \{[\beta]B\} * C\} = \underline{\lambda} \,||\, \overline{\underline{\lambda}} * A * \{[\beta]B\} * C * \{[\lambda]A * C\}.$$

Тоді

$$\{[U_3]\#F\#^*\#\{[U_5]\{[U_4]N\}\}\#\} = \underline{U}_3 \| \overline{\underline{U}}_3 * F^*\#\{[U_5]\{[U_4]N\}\}\#^*\{[U_3]F\}.$$

Після наведеного вище перетворення перейдемо до схеми

$A0.3$

$$\underline{U}_3 \| \overline{\underline{U}}_3 * F^*\#\{[U_5]\{[U_4]N\}\}\#^*\{[U_3]F\} * K.$$

У цій схемі ми зробили з двох вкладених циклів два послідовних цикли, що зменшило складність обчислень.

Скориставшись цим співвідношенням ще раз, позбавимось другого вкладеного циклу, перейшовши до схеми А0.3.1:

$$\#\{[U_5]\{[U_4]N\}\}\#=\#\{[U_5]^* E^*\{[U_4]N\}^* E\}\#=\#\underline{U}_5 \| \overline{\underline{U}}_5 * E^*\{[U_4]N\}^* E^*\{[U_5]E^* E\}\#=$$
$$=\#\underline{U}_5 \| \overline{\underline{U}}_5 ^*\{[U_4]N\}^*\{[U_5]E\}\#;$$

$A0.3.1$

$$\underline{U}_3 \| \overline{\underline{U}}_3 ^*\#F\#^*\#(\underline{U}_5 \| \overline{\underline{U}}_5 ^*\{[U_4]N\}^*\{[U_5]E\})\#^*\{[U_3]\#F\#\} * K.$$

Детально розглянувши цю схему, можна побачити, що $\{[U_5]E\}$ не має сенсу, оскільки тіло циклу є тотожний оператор. Тож цей цикл може бути вилученим зі схеми.

$A0.3.2$

$$\underline{U}_3 \| \overline{\underline{U}}_3 ^*\#F\#^*\#(\underline{U}_5 \| \overline{\underline{U}}_5 ^*\{[U_4]N\})\#^*\{[U_3]\#F\#\} * K.$$

Скориставшись властивістю циклу ${}_\alpha\{A^* B\}={}_\alpha\{A\} \| \ {}_\alpha\{B\}$, перейдемо до схеми

$A0.3.3$

$$\underline{U}_3 \| \overline{\underline{U}}_3 ^*\#F\#^*\#(\underline{U}_5 \| \overline{\underline{U}}_5 ^*\{[U_4]([U_1]A,B)^*([U_2](G \vee H),E\})\#^*\{[U_3]\#F\#\}) * K;$$

$A0.3.4$

$$\underline{U}_3 \| \overline{\underline{U}}_3 ^*\#F\#^*\#(\underline{U}_5 \| \overline{\underline{U}}_5 ^*(\{[U_4]([U_1]A,B)\} \| \{[U_4]([U_2](G \vee H),E)\}))\#^*\{[U_3]\#F\#\} * K.$$

На цьому етапі маємо можливість розпаралелити альтернативу з використанням фільтру $[U_2](G \vee H),E = \underline{U}_2 ^*(G \vee H) \| \overline{\underline{U}}_2 * E$. Підставивши праву частину співвідношення в попередню формулу, отримаємо

$A0.3.5$

$$\underline{U}_3 \| \overline{\underline{U}}_3 ^*\#F\#^*\#(\underline{U}_5 \| \overline{\underline{U}}_5 ^*(\{[U_4]([U_1]A,B)\} \| \{[U_4](\underline{U}_2 ^*(G \vee H) \| \overline{\underline{U}}_2 * E)\}))\#^*\{[U_3]\#F\#\} * K.$$

Можна побачити, що конструкція $(\underline{U}_2 ^*(G \vee H) \| \overline{\underline{U}}_2 * E)$ є надлишковою. Виконання другої паралельної гілки не має жодного сенсу. Тобто альтернативу $([U_2](G \vee H),E)$ зі схеми А0.3.4 можна замінити на менш складну операцію $\underline{U}_2 ^*(G \vee H)$ завдяки використанню операції фільтрації.

$A0.3.6$

$$\underline{U}_3 \| \overline{\underline{U}}_3 ^*\#F\#^*\#(\underline{U}_5 \| \overline{\underline{U}}_5 ^*(\{[U_4]([U_1]A,B)\} \| \{[U_4](\underline{U}_2 ^*(G \vee H))\}))\#^*\{[U_3]\#F\#\} * K;$$

$A0.3.7$

$$\underline{U}_3 \| \overline{\underline{U}}_3 * \# F \# * \# (\underline{U}_5 \| \overline{\underline{U}}_5 * (\{[U_4]([U_1]A, B)\} \| \{[U_4 \vee \overline{\underline{U}}_2] * (G \vee H)\})) \# * \{[U_3] \# F \#\} * K;$$

$A0.3.8$

$$\underline{U}_3 \| \overline{\underline{U}}_3 * \# F * \# (\underline{U}_5 \| \overline{\underline{U}}_5 * (\{[U_4]\underline{U}_1 * A \| \overline{\underline{U}}_1 * B)\} \| \{[U_4 \vee \overline{\underline{U}}_2](G \vee H)\})) \# * (\{[U_3] \# F \#\}) * K;$$

$A0.3.9$

$$\underline{U}_3 \| \overline{\underline{U}}_3 * \# F \# * \# (\underline{U}_5 \| \overline{\underline{U}}_5 * ((\{[U_4 \vee \overline{\underline{U}}_1]A\} * \underline{U}_4 \| \{[U_4 \vee \underline{U}_1]B\} * \underline{U}_4) \| \{[U_4 \vee \overline{\underline{U}}_2](G \vee H)\})) \# *$$
$$* (\{[U_3] \# F \#\}) * K.$$

На виході трансформації одержано САА-М схему модифікованого паралельного алгоритму Флойда-Уоршалла.

5. Висновки

1. Обґрунтовано доцільність використання алгоритму Флойда-Уоршалла для формування перспективних протоколів маршрутизації.
2. Виконано формалізацію алгоритму з використанням САА-М і отримано регулярну схему алгоритму (А0).
3. Запропоновано стратегії розпаралелювання алгоритму Флойда-Уоршалла, які ґрунтуються на однорозмірному та дворозмірному розбитті матриці суміжності та на факті відсутності у циклі алгоритму залежності за даними.
4. Отримано паралельну регулярну схему алгоритму Флойда-Уоршалла та, з використанням еквівалентних перетворень, спектр модифікованих схем алгоритму (А0.2.1, А0.3.2, А0.3.9).

СПИСОК ЛІТЕРАТУРИ

1. Семёнов Ю.А. Телекоммуникационные технологии http://citforum.ru/nets/semenov/ (ГНЦ ИТЭФ).
2. Погорілий С.Д., Калита Д.М. Про підхід до формалізації протоколів маршрутизації на прикладі протоколу OSPF // Вестник МСУ. – 2000. – № 4. – С. 79–85.
3. Іванова Д., Калита Д. Дослідження, аналіз та трансформація алгоритму Беллмана-Форда // Вісник Міжнародного Соломонового Університету «МАГІСТЕРІУМ». – 2004. – № 9. – С. 109–118.
4. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы, построение и анализ. – М.: МЦНМО, 2000. – С. 510–535.
5. Седжвик Р. Фундаментальные алгоритмы на С++. Алгоритмы на графах. – СПб.: ООО «Диасофт ЮП», 2002. – С. 304–310.
6. Англо-русский толковый словар по комп'ютерам и Интернет http://termin.narod.ru/r/routing.htm http://termin.narod.ru/p/protocol.htm.
7. Погорілий С.Д., Калита Д.М. Оптимізація алгоритмів маршрутизації з використанням систем алгоритмічних алгебр // УсиМ. – 2000. – № 4. – С. 20–30.
8. Погорілий С.Д., Камардіна О.О. Дослідження та створення інструментальних засобів автоматизованої трансформації схем алгоритмів // Проблемы программирования. – 2004. – № 1–2. – С. 417–421.
9. Погорілий С.Д. Автоматизація наукових досліджень. Основоположні математичні відомості. Програмне забезпечення / Під ред. академіка АПН України О.В. Третяка. – К.:«ВПЦ Київський університет», 2002. – 290 с.
10. Grama A., Gupta A., Karypis G., Kumar V. Introduction to Parallel Computing, Second Edition. – Addison Wesley, 2003. – ISBN- 0-201-64865-2. – 856 p.
11. Internet Research Task Force (IRTF) http://www.irtf.org.