DOI 10.36074/csriteenat.ed-1.06

УДК 681.3

Опубліковано 3 листопада 2020 року

ПАРАЛЕЛЬНІ ОБЧИСЛЮВАЛЬНІ СТРУКТУРИ ДЛЯ РІШЕННЯ ЗАДАЧ ДИСКРЕТНОЇ ОПТИМІЗАЦІЇ

ПЕТРЕНКО Ольга Євгенівна

кандидат технічних наук, доцент, доцент кафедри

Харківський національний університет радіоелектроніки

НОСИК Андрій Михайлович

кандидат технічних наук, старший науковий співробітник,

доцент кафедри «Мультимедійних інформаційних технологій і систем»

Національний технічний університет «Харківський політехнічний інститут»

УКРАЇНА

Анотація: *Проблеми автоматизації різних процесів управління і обробки даних на основі застосування сучасних засобів обчислювальної техніки є на сьогодні однією з головних задач інформаційних технологій. Успішне рішення цих проблем залежить від сумісних зусиль різних фахівців. Одержані цікаві результати в області розробки і дослідження нових математичних моделей ряду важливих процесів, створення ефективних математичних методів рішення задач, що виникають при побудові автоматизованих систем, зокрема, задач дискретної і комбінаторної оптимізації в різних постановках. У монографії надані паралельні обчислювальні структури для рішення задач дискретної оптимізації.*

ВСТУП.

Математичний аналіз принципів побудови супер-ЕОМ і інших високопродуктивних систем значною мірою ускладнюється більшою їхньою розмаїтістю, численними поняттями й відсутністю формалізованих принципів дослідження.

Варто виділити монографію В. В. Воєводіна [2], в якій зроблена спроба об'єднати дослідження обчислювальних методів і обчислювальних структур у єдиний процес, і розглянути питання відображення проблем обчислювальної математики й обчислювальних структур. В науковому виданні зроблений песимістичний вивід про те, що немає ніяких підстав сподіватися на одержання загального завдання відображення проблем математики на обчислювальні структури за

допомогою деякого універсального методу. Вивід базується на тім, що алгоритми можна зображувати ациклічно орієнтованими графами, а ряд завдань, пов'язаних з аналізом можливості ефективної реалізації довільного алгоритму A1, якому відповідає граф G1, на обчислювальній структурі заданій графом G2, ставляться до NP-повних завдань.

Мета роботи надати архітектури паралельних обчислювальних структур для рішення задач дискретної оптимізації.

ПРОБЛЕМИ РОЗВИТКУ ПАРАЛЕЛЬНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ І ЇХНЬОГО МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ.

Алгоритм можна подавати у вигляді графів, при цьому природно покласти, що алгоритм і його правило, яке описує, дозволяють визначити: множину змінних, у перетворенні яких полягає реалізація алгоритму; множину операцій, які виконуються у процесі реалізації алгоритму; відповідність, що показує які результати виконання попередніх операцій є аргументами для кожної операції.

Нехай заданий деякий алгоритм A (рис. 1). Його можна зобразити у вигляді підмножини алгоритмів $\{A_j\}$ $j = \overline{(1,n)}$, які необхідно виконати в певній послідовності, заданої графом G(V,E). В останньому кожна вершина відповідає алгоритму A_j, а кожне ребро між вершинами i–j, спрямоване від вершини i до j, існує тільки в тому випадку, якщо алгоритм A_j можна виконати тільки після реалізації алгоритму A_i.

На рис. 1-а наведена структурна схема деякого алгоритму A, задана у вигляді графа G(V,E), з якого видно, що для його реалізації спочатку потрібно виконати алгоритм A_1. Результати роботи цього алгоритму є вхідними даними для виконання алгоритмів A_2, A_3, A_4**.** Після їхнього виконання реалізується алгоритм A_5, а алгоритм A_6 спрацьовується після алгоритмів A_3, A_4, а A_7 – слідом за A_5 і A_6. Очевидно, що число різних подань графів алгоритму A визначається множиною способів, які використовуються для рішення задачі [1].

У принципі, граф G (рис. 1-а) можна розглядати як структуру програмного виробу, що реалізує алгоритм A. Тоді $\{A_j\}$ - множина програмних модулів, з яких складається програма реалізації алгоритму A. Очевидно, що якщо в кожну вершину графа G (рис. 1-а) помістити процесор, орієнтований на виконання алгоритму A_j, то ми одержимо обчислювальну структуру, що дозволяє реалізувати алгоритм A.

 DOI 10.36074/csriteenat.ed-1.06

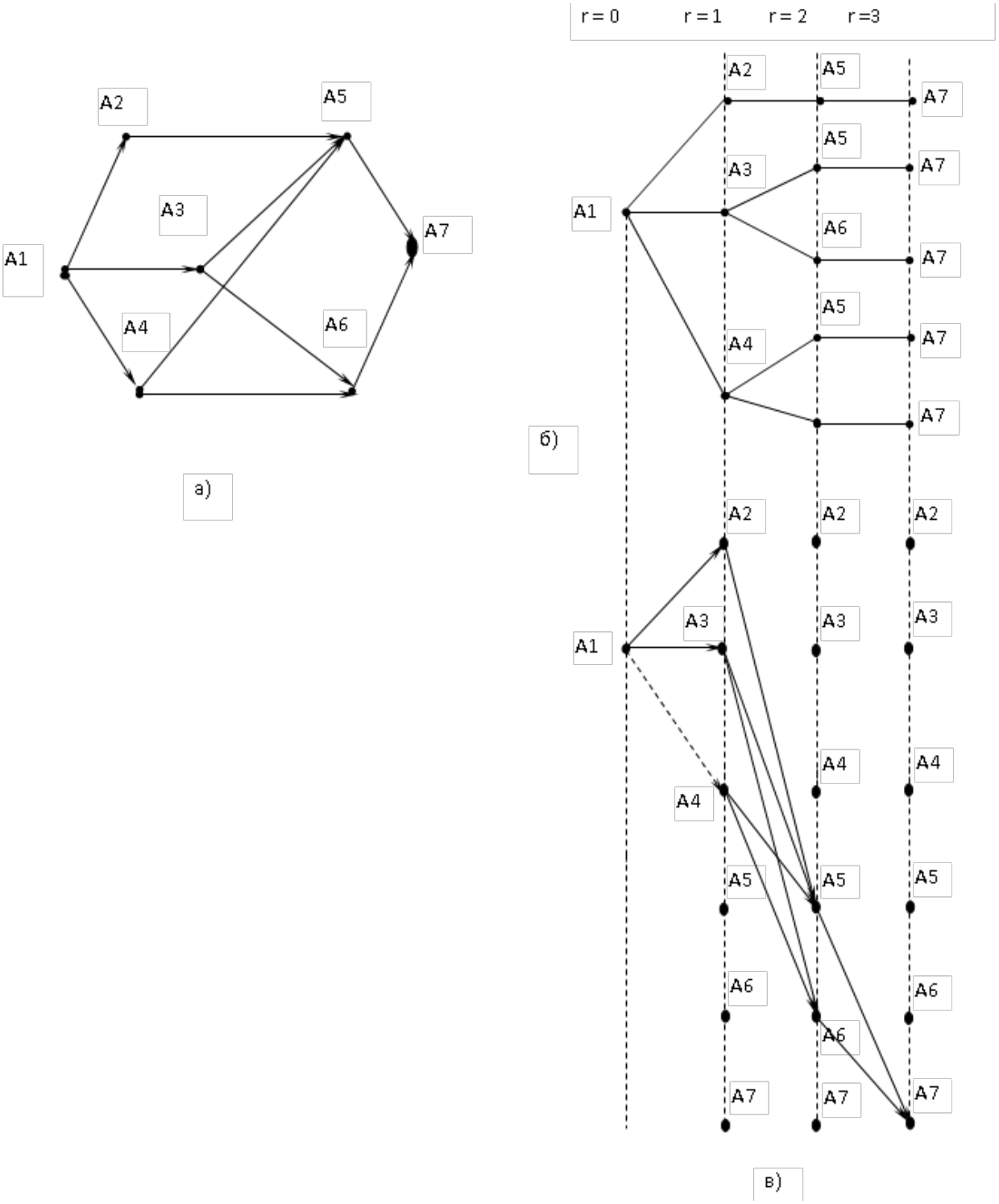


Рис. 1. **Структура графа G**

Припустимо, що кожний процесор $П_j$ виконує алгоритм A_j відразу, як тільки на його вхід надійдуть усі необхідні дані. Позначимо t_j- час реалізації кожного з алгоритмів A_j на відповідному процесорі $П_j$. Тоді, якщо привласнити вершинам графа G(V,E) усі ваги t_j , мінімальний час реалізації алгоритму А не може перевищити довжину критичного

(самого довгого) шляху M_{17} у графі G. Очевидно, що така обчислювальна структура ідеальна для даного подання графа алгоритму А, оскільки вона забезпечує дозвіл завдання за мінімальний час.

Ранг критичного шляху (число ребер у шляху) у графі G(V,E), який відповідає деякому алгоритму А, будемо називати глибиною алгоритму А, а максимальне число алгоритмів $A_j \in A$, яких можна одночасно виконати на довільному r-м кроці реалізації алгоритму А, - шириною алгоритму. Надалі глибину й ширину довільного алгоритму позначимо буквами а й h . Припустимо, що кожний з алгоритмів $\{A_j\} \in A$ реалізується на будь-якому процесорі $П_j$.

Нехай задана структурна схема алгоритму А графом G(V,E). Виникає завдання визначення оптимальної стратегії розподілу алгоритмів $\{A_j\}$ за процесорами $\{П_j\}$, при якій час рішення задачі відрізняється від довжини критичного шляху в графі G(V,E) на мінімально можливу величину. Обрана стратегія їхнього використання й повинна визначити обчислювальну структуру. Якщо ширина алгоритму дорівнює h, то природно спробувати обмежитися h-процесорами для реалізації алгоритму А.

Для цього уявимо граф G(V,E) алгоритму А в вигляді стягнутого дерева всіх шляхів. Воно будується по матриці суміжності $B = \| b_{ij} \|$ від деякої вершини S таким чином:

1. Позначимо вершину S і беремо S-й рядок матриці B.

1. За цим рядком вибираємо номер вершин j_1, для яких $b_{sj} \neq 0$, і утворимо множина вершин, що мають від вершини S шлях рангу r = 1 (вершини першого ярусу).

2. Вершини r-го ярусу зі шляхами рангу r від вершини S перебувають почерговим переглядом рядків вершин j_{r-1}, в (r - 1) ярусі, які не зустрічалися в шляху від S до $j_{r-1.}$

3. Побудова дерева триває до одержання шляхів максимального можливого рангу, або до тих пір, поки для вершин j_{r-1} не виявиться, що всі вершини вже ввійшли в шлях $\mu s j_{r-1}$.

У загальному випадку число розгалужень у такому дереві (N - 1), тобто для цього утворення буде потрібний (N-1) регістр пам'яті. Щоб уникнути подібного роду труднощів, перейдемо від дерева D усіх шляхів до стягнутого дерева всіх шляхів (рис. 1-в). Воно може бути отримано стягуванням однойменних вершин дерева D на кожному ярусі. При цьому вершини з однаковими номерами на кожному ярусі впорядковуються в

 DOI 10.36074/csriteenat.ed-1.06

горизонтальні лінійки. Як видно з (рис. 1-в), множина шляхів у стягнутому дереві шляхів така ж, як і в дереві D (рис. 1-б), але для прочитання шляхів на кожній горизонтальній лінійці дозволяється бути один раз.

Розглянемо стратегію розподілу процесорів за алгоритмами A_i у припущенні, що в нас є N-процесори і число вершин у графі алгоритму так само дорівнює N.

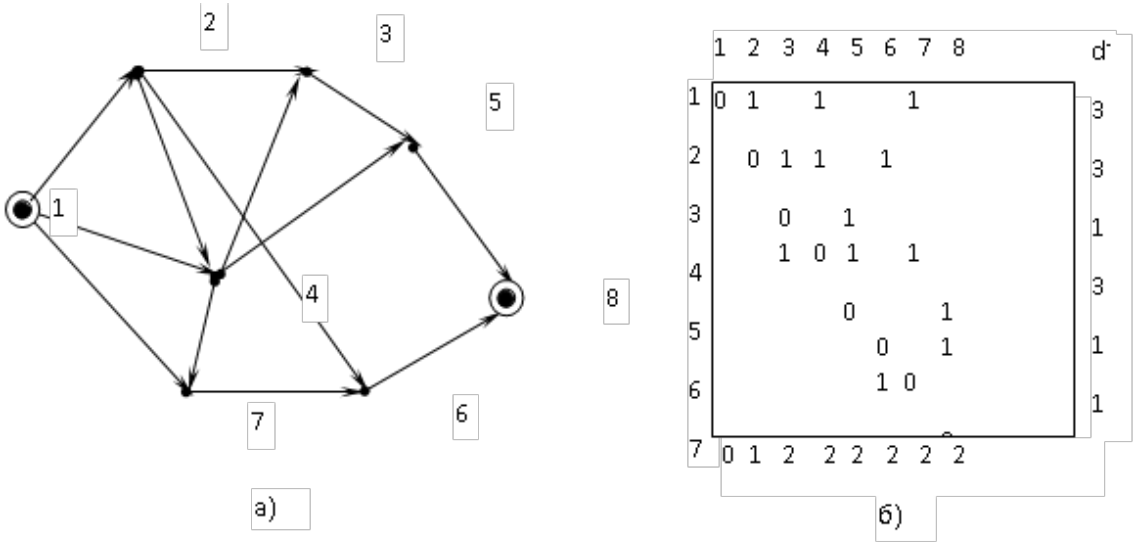
Стратегія полягає в тім, щоб задіяти процесори у відповідності зі стягнутим деревом шляхів графа алгоритму А. Інакше кажучи, на першому кроці працює процесор $П_1$, реалізуючи алгоритм A_1, розташований на першому ярусі (r = 1). Далі по матриці суміжності У встановлюється зв'язок між процесорами $П_1$, $П_2$, $П_3$ і $П_4$. Після закінчення роботи всіх процесорів $П_2$, $П_3$, $П_4$ відповідно до матриці суміжності У встановлюється зв'язок процесорів $П_2$, $П_3$, $П_4$ із процесорами $П_5$, $П_6$. Результати роботи процесорів $П_2$, $П_3$, $П_4$ передаються на процесори $П_5$, $П_6$ далі процесори $П_5$, $П_6$ реалізують алгоритми A_5, A_6 .

Після закінчення їхнього виконання, по матриці суміжності встановлюється зв'язок між процесорами $П_5$, $П_6$ і процесором $П_7$ та йому передається вся необхідна інформація для роботи. У такий спосіб рішення завдання здійснюється в процесі побудови стягнутого дерева шляхів. На кожному етапі обчислень взаємодіють процесори, що відповідають вершинам у стягнутому дереві шляхів, розташованим на n-м і r+1 ярусі.

Ми розглянули приклад (рис. 1), коли структурна схема алгоритму складена так, що алгоритми A_j на кожному ярусі одержують необхідні вихідні дані для їхньої реалізації. У загальному випадку при довільному завданні структурної схеми алгоритму ця умова може не дотримуватися (рис. 2).

Наприклад, якщо структурна схема алгоритму А задана графом G'(V,E), зображеним на рис. 2-а. На рис. 2-б наведена матриця суміжності графа G і значень ступеня d^-– результату кожної вершини й d^+ - ступеня заходу. Ясно, що ступінь вершини графа G' дорівнює $d = d^- + d^+$. У цьому випадку побудова стягнутого дерева шляхів (рис. 2-б) відрізняється тільки лише тим, що від вершини і зв'язок з вершинами наступного ярусу встановлюється лише, коли ступінь заходу вершини досягла величини, обумовленої матрицею суміжності В.

Фактично це означає, що процесор з номером j займається з першої подачі на нього інформації, необхідної для реалізації алгоритму А, і він залишається зайнятим доти, поки число звертань до нього не досягне ступеня заходу d_j^+. Ступінь вершини графа G' дорівнює $d = d^- + d^+$. У цьому випадку побудова стягнутого дерева шляхів (рис. 2-б) відрізняється тільки лише тим, що від вершини і зв'язок з вершинами наступного ярусу встановлюється лише коли ступінь заходу вершини досягла величини, обумовленою матрицею суміжності В.



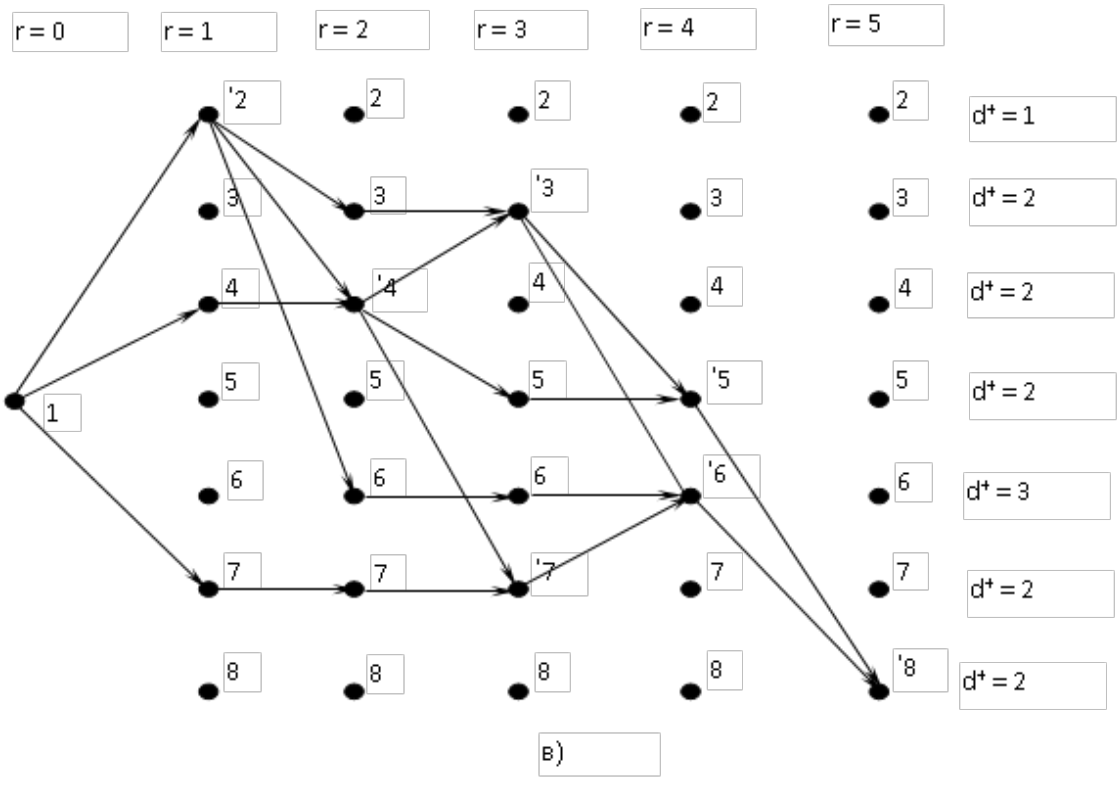


Рис. 3. **Структура графа G'**

 DOI 10.36074/csriteenat.ed-1.06

Фактично це означає, що процесор з номером j займається з першої подачі на нього інформації, необхідної для реалізації алгоритму А, і він залишається зайнятим доти, поки число звертань до нього не досягне ступеня заходу d_j^+. Це означає, що вся вихідна інформація для виконання алгоритму A_j отримана і процесор j здатний його реалізувати.

Відповідно до стягнутого дерева шляхів (рис. 2-в) виконання алгоритму А, наданого графом G (рис. 2-а), здійснюється таким чином. Процесор П1 виконує алгоритм A_1, далі відповідно до В, установлюються зв'язки із процесорами $П_2$, $П_4$, $П_7$ і на них пересилаються результати виконання A_1. При цьому процесор $П_2$ одержує всю необхідну інформацію й відпрацьовує алгоритм A_2. Процесори $П_4$, $П_7$ зайняті й перебувають у режимі очікування. Далі по У встановлюються зв'язки із процесорами $П_3$, $П_4$, $П_6$ і передається інформація, необхідна для їхньої роботи. Ступінь заходу вершини 4 стане рівною $d^+ = 2$, обумовленою матрицею В і, отже, процесор $П_4$ може виконати алгоритм A_4. На рис 2. номери процесорів, що звільнилися на кожному ярусі, позначені зірочками (*). Для процесора $П_4$ визначаємо по У зв'язку із процесорами $П_3$, $П_5$, $П_7$ при цьому процесори $П_3$ і $П_7$ виконують свої алгоритми та передають інформацію на процесори $П_5$, $П_6$, які реалізують алгоритми A_5, A_6 і передають вихідні дані на процесор $П_8$. Останній і завершить виконання алгоритму А.

Для здійснення такої організації обчислень передбачається використати обчислювальну структуру (рис. 3), що ми й будемо називати циклічною паралельною обчислювальною структурою.

Вона містить n-процесорів (що складаються з пам'яті M_i і арифметичного логічного пристрою A_i), входи й виходи яких приєднані до комутатора розмірності (n*n). Комутатор управляється пристроєм управління ведення й виводу (ПУВВ) інформації відповідно до матриці В.

Фактична така обчислювальної структура й реалізує побудову стягнутого дерева шляхів графа, забезпечуючи зв'язок між процесорами при переході від ярусу до ярусу за допомогою ПУВВ і комутатора К, на основі матриці В. Обчислювальний процес у такій структурі можна зобразити у вигляді паралельно обертових кілець, рух яких синхронний або асинхронний, залежно від прийнятої організації синхронізації.

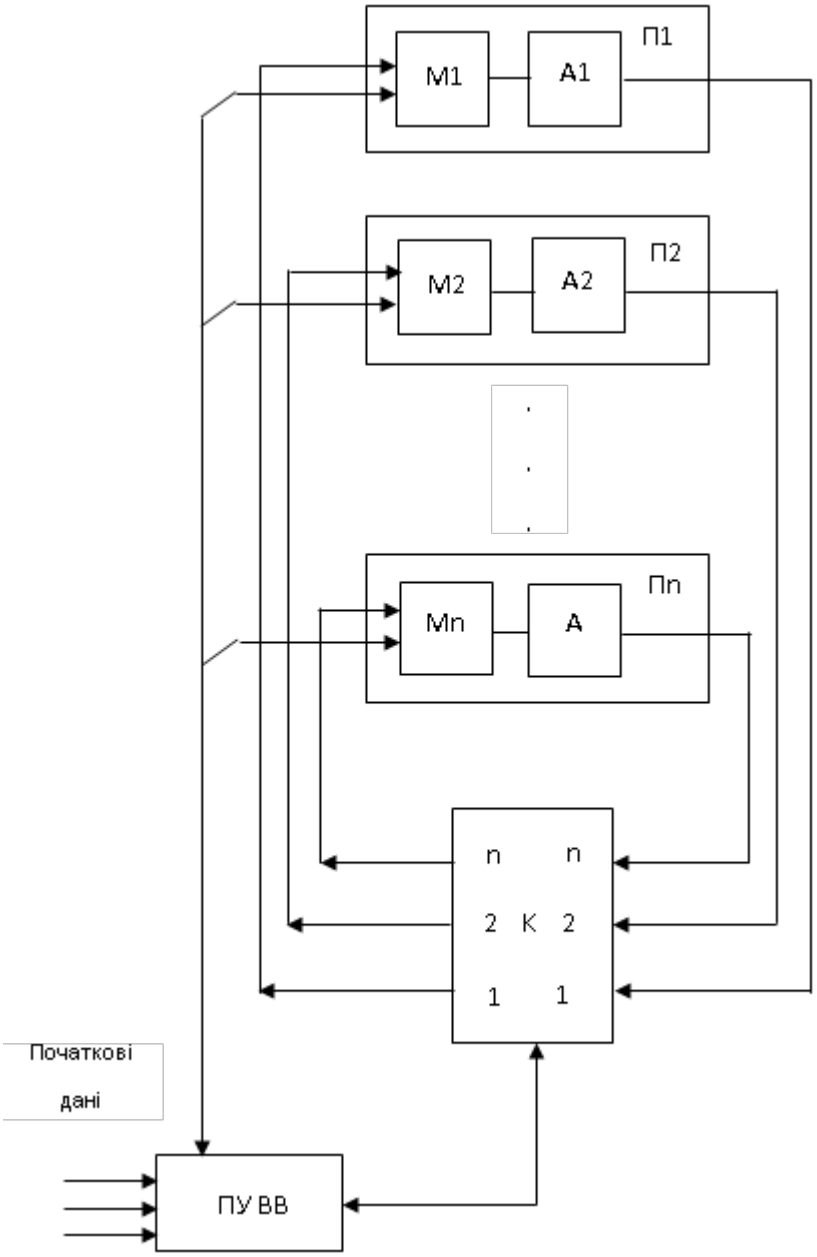


Рис. 3. **Обчислювальна структура**

Слід зазначити, що якщо число процесорів дорівнює N – числу вершин у структурній схемі алгоритму, заданого графом G, – то на кожному кроці роботи процесорів частина з них простоює. Якщо максимальна ширина алгоритму A дорівнює h, то одночасно на довільному ярусі буде працювати не більше h-процесорів. Такою кількістю процесорів можна обмежитися в кільцевій структурі, за рахунок чого зменшити число процесорів, що простоюють. Це досяжно, якщо при переході від ярусу до ярусу в дереві стягнутих шляхів будуть автоматично змінюватися номери процесорів, що комутують, відповідно до матриці суміжності B.

 DOI 10.36074/csriteenat.ed-1.06

Архітектура кільцевої структури здобуває вигляду, показаною на рис. 4, де ПКП (пристрій керування перенумерацією комутатора К) для залучення процесорів, що відповідають наступному ярусу стягнутого дерева шляхів.

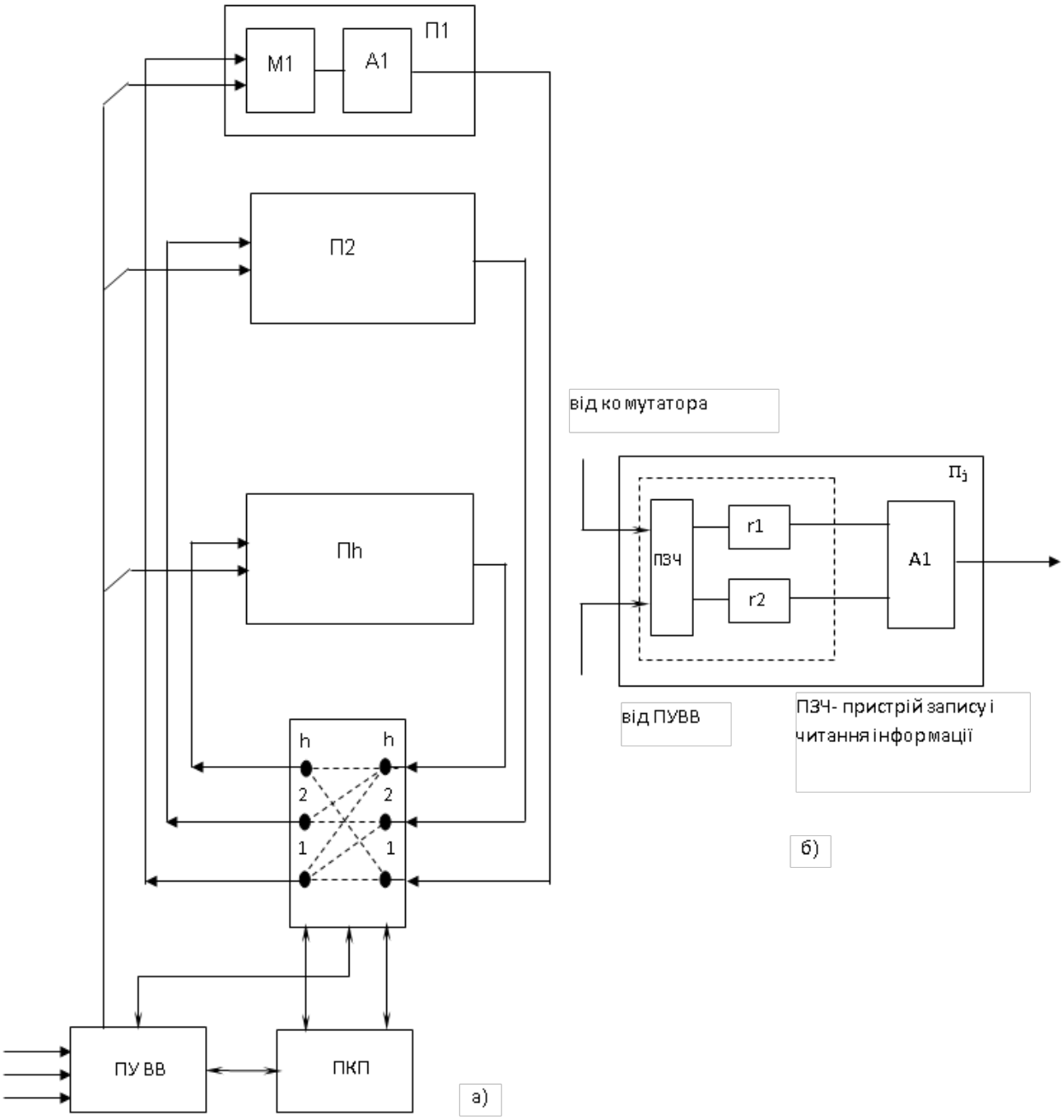


Рис. 4. **Архітектура кільцевої циклічної ПОС**

При реалізації алгоритму А на кільцевій структурі час виконання алгоритму відрізняється від довгі критичного шляху тільки на величину at_k, де t_k – час спрацьовування комутатора. При побудові комутатора на основі програмувальних логічних матриць t_k визначається часом перемикання логічного елемента "І".

Якщо $t_k < \min \{t_j\}$, то час виконання алгоритму А на циклічній обчислювальній структурі буде в точності дорівнювати довжині критичного шляху.

ВАРІАНТИ ПОБУДОВИ ПАРАЛЕЛЬНОЇ ОБЧИСЛЮВАЛЬНОЇ СТРУКТУРИ ЦИКЛІЧНОГО ТИПУ ДЛЯ ЗАДАЧ ОПТИМІЗАЦІЇ.

Процес реалізації алгоритму А розбивається на множину алгоритмів $\{A_j\}$, кожний з яких у свою чергу знову розбивається на підмножину алгоритмів $\{A_j\}'$ і т. д., до тих пір, поки алгоритм буде зображувати просто одну з базових операцій $y_i \in Y$. Множина Y утворює функціонально повну систему. Тому процесори $П_i$ доцільно створювати з використанням транспютерної технології побудови високопродуктивних мікропроцесорів, утворених за принципом RISC – архітектури, яка має локальну пам'ять, що становить два регістри й пристрої запису та зчитування інформації. RISC процесор містить арифметичний пристрій A_i (рис. 5), що забезпечує виконання базових арифметичних операцій. Процесор у цьому випадку реалізує зазначені операції над двома числами й, отже, структурна схема алгоритму А повинна подаватися у вигляді орієнтованого графа, в якому ступінь заходу вершини не більше двох, а ступеня результату – довільна. У процесорах можна звільнитися від ПЗЧ (рис. 5), тобто розпаралелити процес запису вихідної інформації в регістрі пам'яті, що вимагає додаткового комутатора К, який працює в паралель із основними. При цьому архітектура обчислювальної системи буде мати вигляд як на рис. 5.

Для оцінки роботи алгоритмів на таких системах використаємо критерій – коефіцієнт прискорення:

$$K_y = \frac{T_1}{T_n} \quad , \tag{1}$$

де: T_1 час реалізації алгоритму А на гіпотетичній однопроцесорної ЕОМ з такою ж швидкодією, як і $П_j$, і оперативною пам'яттю, рівною сумарній пам'яті $П_j$, при наявності необхідного числа зовнішніх пристроїв зі швидкостями обміну інформацією як і в циклічній обчислювальній системі;

T_n – час реалізації алгоритму А на циклічній обчислювальній системі, що містить n-процесорів.

Якщо алгоритм А зображений структурною схемою, заданою графом G, що містить n-вершин, то:

 DOI 10.36074/csriteenat.ed-1.06

$$T_1 = \sum_{i=1}^{n} t_i + nt_0, \tag{2}$$

де: t_i – час виконання і – базової операції,
t_0 – час обміну.

Час роботи алгоритму А на структурах, зображено на рис. 5.6, можна визначити так:

$$T_n = \sum_{j=1}^{a} t_j^{max} + at_0 \quad , \tag{3}$$

де: t_j^{max} – найбільше із часів базових операцій, виконуваних одночасно декількома процесорами на j-му кроці;
а – глибина алгоритму А, заданого графом G.

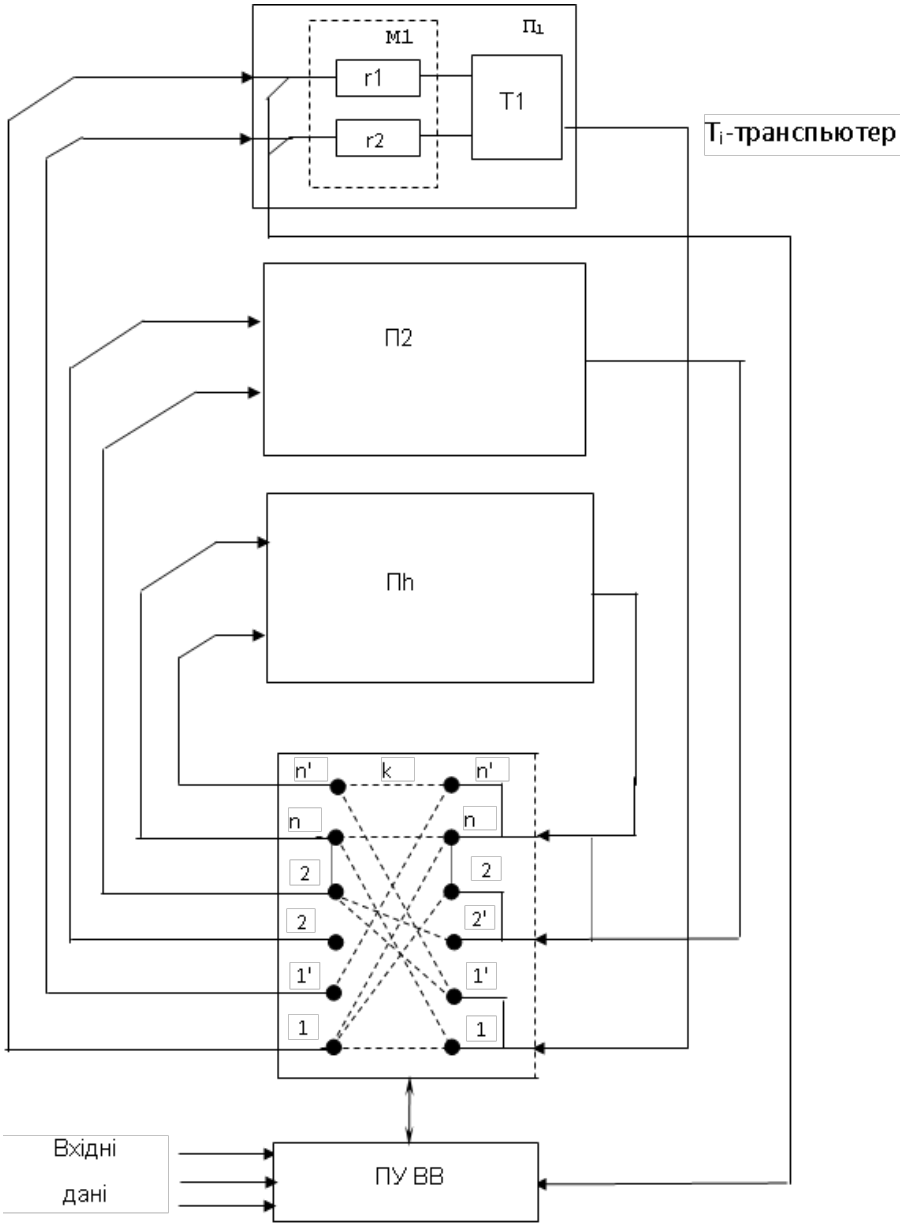


Рис. 5. **Обчислювальна структура**

Тоді коефіцієнт прискорення прийме вигляд:

$$K_y = \frac{\sum_{i=1}^{n} t_i + nt_0}{\sum_{j}^{a} t_j^{max} + at_0} . \quad (4)$$

Для спрощення аналізу покладемо $t = \overline{t_k}$ – середнє значення по i = (1,n), а $t_j^{max} = \overline{t_a}$ – середнє значення по j = (1,a), при цьому

$$К_y = \frac{n}{a}\alpha \quad , \quad (5)$$

де: $\alpha = \frac{\overline{t_k \pm t_0}}{\overline{t_a \pm t_0}}$.

Як видно зі співвідношення (5) зі зменшенням глибини алгоритму А зростає коефіцієнт прискорення.

У випадку, коли кожний процесор спеціалізується на виконанні алгоритмів якоїсь підмножини алгоритмів $\{A_j\}$, можлива організація обчислень циклічного типу на обчислювальній структурі, зображеної на рис. 6.

У структурі зображеній на рис. 6 пам'ять процесора $П_i$ включає (n - 1) регістр, закріплений за кожним $П_i$ процесором (i = $\overline{1,n}$); комутатор $К_i$; пристрій управління записом і зчитуванням; арифметико-логічні блоки (за числом регістрів); пристрій аналізу $ПА_i$, що за певним критерієм вибирає з результатів роботи A_j найкращий. Кільця циркуляції інформації замикаються через блок вибору напрямку передачі інформації (БВН), він послідовно здійснює опитування процесорів $П_i$. Інформація процесора I=1 за допомогою комутатора $К_i$ записується в регістри пам'яті всіх процесорів $П_i$, закріплених за процесором з номером l. Після закінчення опитування процесорів вони виконують операції за отриманими даними, а результати їхньої роботи знову за допомогою БВН і комутаторів К заносяться в регістри пам'яті, закріпленої за процесорами. Описані операції виконуються до повної реалізації алгоритму А. Для цього варіанту коефіцієнт прискорення визначається вираженням:

$$K_y = \frac{\sum_{i=1}^{n} t_i + nt_0}{\sum_{j=1}^{a} t_j^{max} + \alpha\, nt_0}. \quad (6)$$

 DOI 10.36074/csriteenat.ed-1.06

Щоб забезпечити прискорення, у цьому випадку повинна виконуватися нерівність:

$$\sum_{i=1}^{n} t_i \quad \sum_{j=1}^{a} t_j^{\max} > nt_0(a \quad 1) \tag{7}$$

при $t_i = t_k$ *і* $t_j^{\overline{max}} = t_0$

$$\frac{n}{a} > \frac{t_k \pm nt_0}{t_a \pm t_0}. \tag{8}$$

У реальних завданнях час виконання алгоритму $A_j \in A$ відмінний. Тому якщо $\Delta t = \left(t_1^{\max} - t_2^{\min}\right)$ – час між закінченням виконання самого трудомісткого алгоритму $A_1 \in \{A_i\}$ і найпростішого $A_2 \in \{A_j\}$ використати для опитування процесорів $П_i$, то прискорення, що випливає з (6) може зрости до величини обумовленої співвідношенням (5). Останнє еквівалентно пропозиції, що n = 1 у знаменнику дробу, обумовленої співвідношенням (6). Таку структуру кільцевої мережі ефективно прийняти для визначення найкоротших шляхів рішення завдань динамічного програмування й варіаційного обчислення.

Реалізація кільцевих структур (рис. 4, 5) можлива й на базі багатоканальних систем зв'язку, що використовують частотне або тимчасове ущільнення. Кількість каналів зв'язку буде визначатися числом регістрів пам'яті, які є у наявності у процесорах $П_i$, тобто кожний процесор забезпечується багатоканальним приймачем і передавачем. Розглянуті структури циклічного типу зможуть успішно конкурувати з конвеєрними обчислювачами, побудованими на базі однорідних обчислювальних систем.

Для задач ЦЛП із БЗ паралельна форма узагальненої процедури A_0 збігається із графом $D\Delta$ [1]. У цьому випадку паралельна обчислювальна структура циклічного типу має вигляд, зображений на рис. 5. Як видно з рис. 6 роль комутатора виконує процесор $П_x$. Процесори $П_i$ і = (1,n) служать для зберігання й вибору шляху, яке відповідає обраному правилу відсікань $\{L_w\}$. ПУВВ відповідає за початкове завантаження вихідних даних (ваг вершин, каліброваних векторів і т. д.) та вивід результату після одержання припустимого або екстремального рішення. Усім процесом управляє пристрій управління (ПУ). Кожний $П_i$ процесор відповідає і-тій вершині графа ΔD і зберігає постійно всі властиві її ваги. На цій структурі завдання вирішується в такий спосіб. Пристрій керування формує n раз керуючу послідовність тактів. У кожному такті формується множина шляхів у процесорі $П_x$, що

відповідають номеру k даного такту. Для цього із процесорів $П_i$ і J k вибираються, згідно з правилами відсікань $\{L_w\}$, шляхи, які узагальнюються в $П_x$. Після цього результат сформованої множини записується в k-і процесор. І так доти, поки всі множини поточного рангу не будуть сформовані.

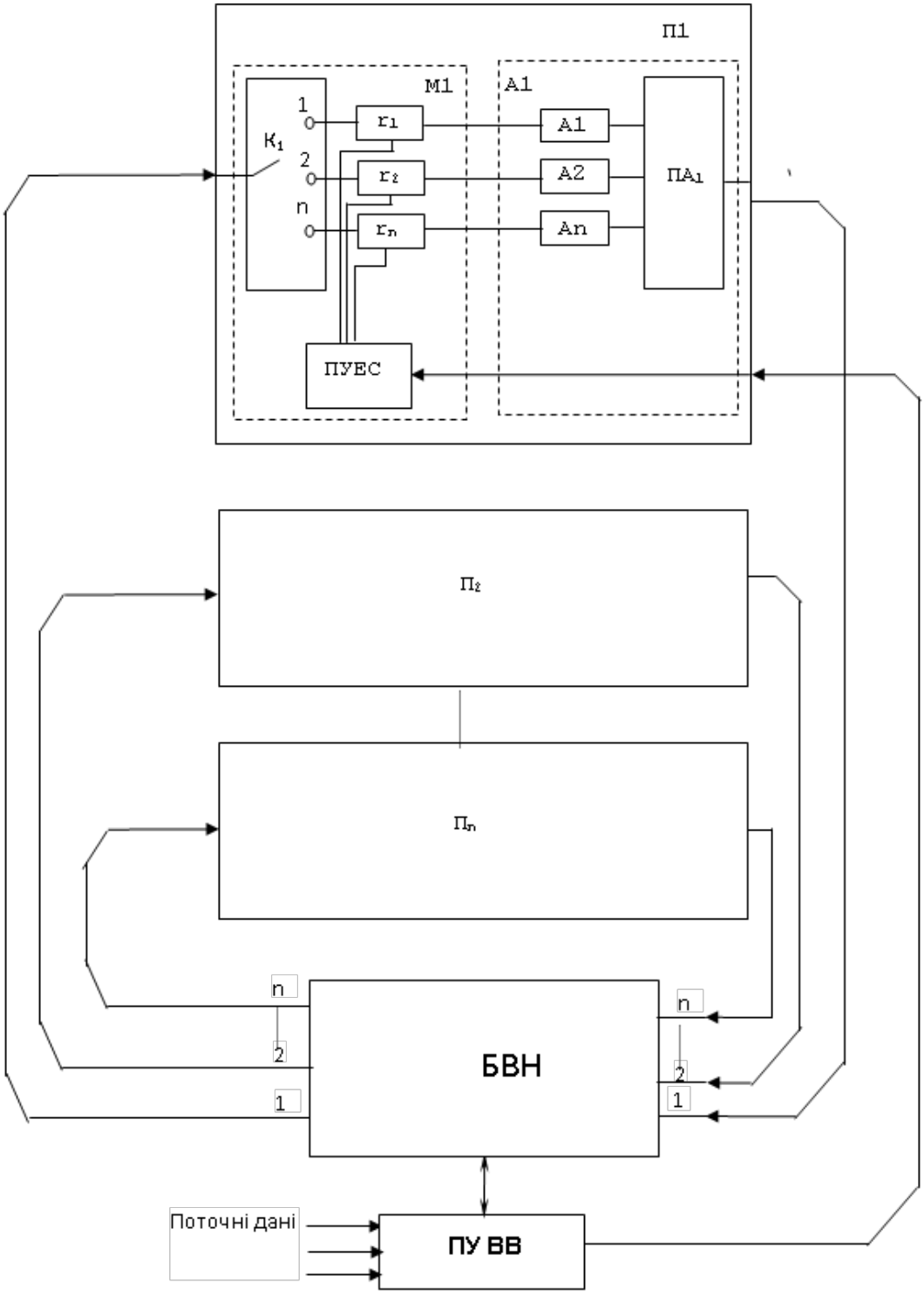


Рис. 6. **Обчислювальна структура**

 DOI 10.36074/csriteenat.ed-1.06

По закінченні формування шляхів рангу r виробляється побудова шляхів рангу r = r + 1 і так доти, поки або всі множини не будуть порожніми (що відповідає нульовому лічильнику векторів поточного рангу), або ранг шляху стане рівним n. Слід зазначити, що для організації зберігання множин як попереднього, так і наступного рангів, необхідно передбачити режим поділу адрес оперативної пам'яті усередині процесорів, щоб при записі множин сформованого рангу не затерти множини збережених векторів поточного рангу.

АРХІТЕКТУРА ПАРАЛЕЛЬНИХ ОБЧИСЛЮВАЛЬНИХ СТРУКТУР СИСТОЛІЧНОГО ТИПУ.

Розглянемо реалізацію алгоритмів паралельних обчислень на систолічних масивах.

Систолічний масив – це обчислювальна мережа, що володіє наступними властивостями:

1. Синхронність. Дані обчислюються ритмічно (при глобальному тактуванні) і пропускаються по мережі.

2. Модульність і регулярність. Масив містить модульні процесорні елементи з однорідними зв'язками. Більш того, обчислювальна мережа може необмежено розширюватися.

3. Просторова локальність і тимчасова локальність. Масив характеризується локально зв'язаною структурою меж з'єднань, тобто просторовою локальністю. У структурі існує розподілена затримка (щонайменше – одиничної тривалості), протягом якої можуть бути завершені прийом, обробка й видача сигналу від одного вузла до іншого, тобто тимчасова локальність.

4. Конвеєрність. Масив проявляє конвеєрність із лінійною швидкістю, тобто досягається прискорення обчислень 0(M), де M – число процесорних елементів (ПЕ). Тут ефективність роботи масиву визначається коефіцієнтом прискорення T_s/T_p, де T_s – час обробки на одному процесорі й Tp – час обробки на матричному процесорі.

Систолічні масиви добре пристосовані для НВІС-реалізації обчислювально-ємних алгоритмів. Вони мають такі переваги, як: модульність, регулярність, локальні міжз’єднання, високий ступінь конвеєрної мультиобробки й безперервний потік даних між ПЕ. Недолік систолічних масивів складається в глобальному керуванні їхнім

функціонуванням за допомогою потактової синхронізації. В апаратурній реалізації глобальна синхронізація призводить до проблем разфазування синхронізуючих імпульсів, відмовостійкості й пікової потужності.

Матричний процесор можна використати як прискорювач, приєднаний до сумісної провідної ЕОМ, або як самостійну ЕОМ, оснащену керуючим процесором. Провідна ЕОМ повинна забезпечувати системне керування й керування даними, встановлювати програму, що управляє всіма пристроями системи, і генерувати коди глобального керування. Вона може бути обрана із широкого діапазону машин, який охоплює різні рівні обчислювальної потужності.

Дана ПОС систолічного типу складається з 4 пристроїв (рис. 7): обчислювальний пристрій 1; обчислювальний пристрій 2; модуль пам'яті; блок керування систолічним процесором.

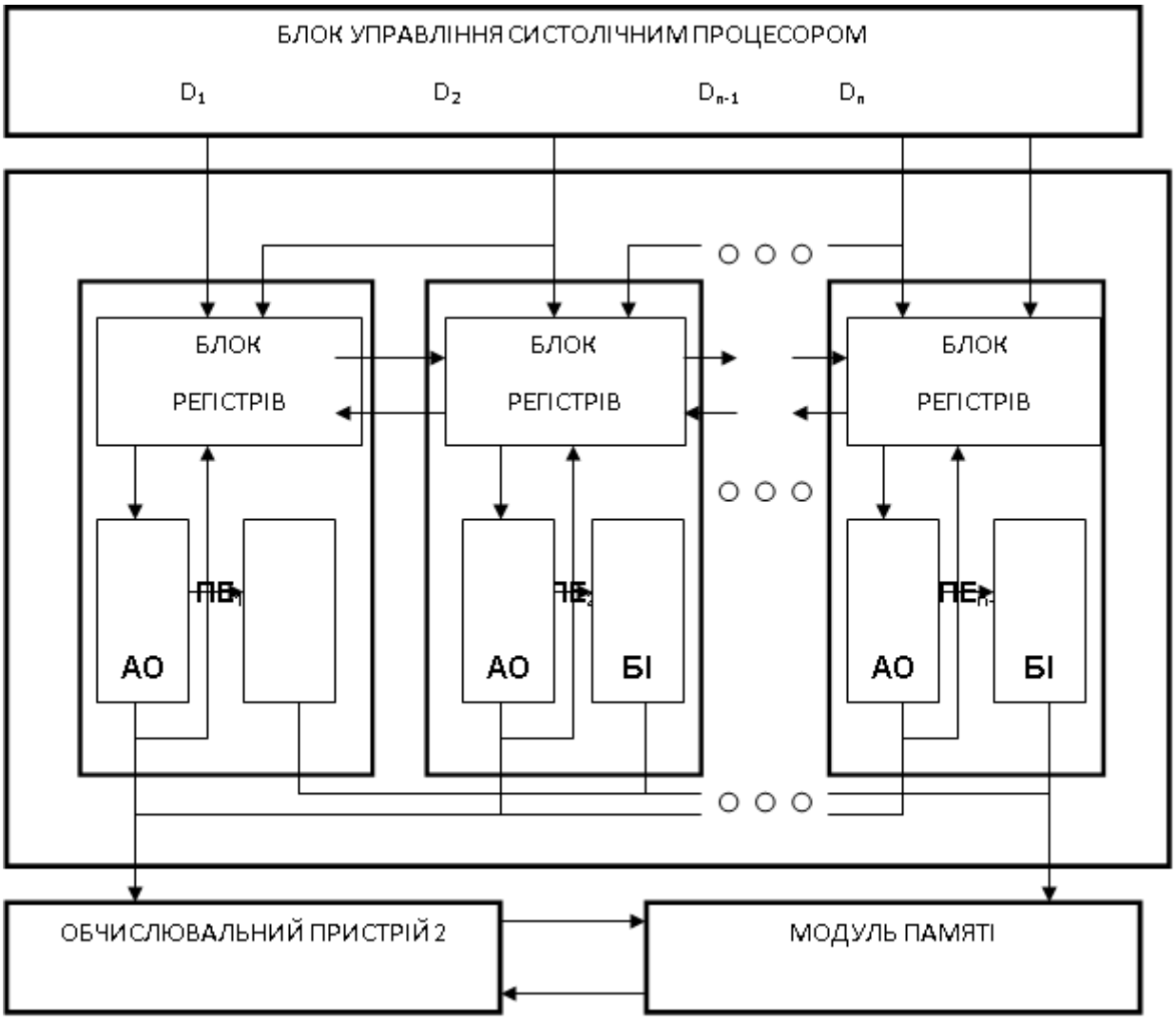


Рис. 7. **Архітектура ПОС систолічного типу**

 DOI 10.36074/csriteenat.ed-1.06

Розглянемо призначення, структуру й порядок виконання обчислень систолічною ПОС.

Обчислювальний пристрій 1 ($ОП_1$) призначений для виконання обчислень локальних екстремумів при заданому функціоналі й обмеженні, а також для визначення (обчислення) номера вершини, у якій локальний екстремум (ЛЕ) вже обчислено. Пристрій складається з n - 1 – процесорних систолічних комірок, реалізованих на основі простої структури, й характеризується однорідністю та локальністю зв'язків.

Кожний процесорний елемент має зв'язок тільки із сусіднім. Систолічні елементи працюють паралельно, після виконання обчислень відбувається обмін даними між сусідніми процесорними комірками [4-7].

У табл. 1 показаний порядок обчислень локальних екстремумів. На першому кроці (такті) буде отриманий у $ПЕ_1$ $ЛЕ_1$, на другому – у $ПЕ_2$ $ЛЕ_2$ і т. д.

Таблиця 1.

Порядок обчислень ПОС

Крок	Виконання обчислень		
	$ПЕ_1$	$ПЕ_2$	$ПЕ_3$
1	1-2	2-3	3-4
2	-	1-3	2-4
3	-	-	1-4
Ранг 1	$ЛЕ_1$	$ЛЕ_2$	$ЛЕ_3$
4	$ЛЕ_1$-3	$ЛЕ_2$-4	-
5	-	$ЛЕ_1$-4	-
Ранг 2	$ЛЕ_1$	$ЛЕ_2$	-
6	$ЛЕ_1$-4	-	-
Ранг 3	$ЛЕ_1$	-	-

Для обчислення глобального екстремума необхідно затратити $N \times (N-1)/2$ тактів, що на порядок нижче при обчисленні в однопроцесорному режимі.

Кожний процесорний елемент складається з:

– блоку регістрів (БР), призначеного для зберігання й забезпечення мікрооперації передачі інформації між регістрами БР сусідніх процесорних комірок;

– арифметичного обчислювача (АО), використовуваного для обчислення ЛЕ на підставі даних, що надходять із БР, вибору ЛЕ за максимальним значенням функціонала й мінімальним значенням

обмеження та пересилання його в блок визначення глобального екстремума $ОП_2$ для обчислення глобального екстремуму;

– блоку ідентифікації (БІ), службовця для визначення номера вершини, у якій ЛЕ визначений.

Обчислювальний пристрій 2 призначений для обчислення із поступаючих ЛЕ глобального екстремуму та формування вектора шляху. Пристрій складається із двох блоків:

– блоку визначення глобального екстремума (БВГЕ);

– блоку визначення вектора шляху (БВВШ).

Блок визначення глобального екстремуму служить для обчислення глобального екстремума й складається зі схеми порівняння, регістрів локального й глобального екстремумів.

Блок визначення вектора шляху використовується для обчислення номера останньої вершини визначення вектору шляху й складається із трьох суматорів, чотирьох лічильників, двох тригерів, регістрів адреси й проміжних обчислень.

Модуль пам'яті призначений для зберігання номерів вершин локальних екстремумів на кожному ранзі обчислень. Складається з дешифратора адреси й пам'яті.

Для виконання обчислень дані надходять одночасно в кожну систолічну комірку. Уведення даних здійснюється під управлінням блоку управління систолічним процесором з буферної пам'яті. Ця пам'ять використовується як буфер між високошвидкісною спеціалізованою шиною й низькошвидкісною шиною ЕОМ. За необхідності дані в буфері обновляються, зчитуються й обробляються систолічною матрицею. Завантаженням буферної пам'яті управляє інтерфейсний процесор, функціональне призначення якого складається в управлінні взаємодією ЕОМ і інтерфейсом системи, плануванні й перевірці обчислень, які виконуються систолічною матрицею.

Таким чином, запропонована архітектура ПОС дозволяє вирішувати задачі ЦЛП із БЗ у масштабі реального часу.

ВИСНОВКИ З ТЕМИ ДОСЛІДЖЕННЯ.

Порівняльний аналіз розроблених алгоритмів з відомими за вибраними показниками ефективності показав, що їх часова складність істотним чином залежить від рангу оптимального рішення, який може

 DOI 10.36074/csriteenat.ed-1.06

належати одній з трьох умовно виділених зон. Тому, об'єктивне порівняння алгоритмів можливо лише для рішень, що належать одній і тій же зоні [1].

З порівняння рішень за зонами видно, що найбільший виграш пропоновані алгоритми дають в другій зоні, де число припустимих рішень експоненціально. Це є важливою перевагою в порівнянні з відомими.

Ефективність виділення "коридору" на ярусі вище за ефективність виділення "коридору" на множинах, тому на останніх етапах багато етапних алгоритмів доцільно використовувати саме цю стратегію відсікань.

Дослідження погрішності наближених алгоритмів показало, що із збільшенням розмірності вирішуваної тестової задачі, вона стабілізується і для різних стратегій відсікань лежить в межах від 1 до 10%.

Розроблено метод паралельних обчислень на основі рангового підходу до рішення задачі ЦЛП з БЗ, який забезпечує лінійну залежність зростання продуктивності системи від кількості процесорних елементів. Застосування методу дозволяє: зменшити на 10 % похибку рішення задачі ЦЛП з БЗ при використанні стратегій MAX, MIN і MAX-MIN для відсікання безперспективних варіантів рішень; розробити паралельні алгоритми для реалізації стратегій MAX, MIN і MAX-MIN; розробити архітектури паралельних обчислювальних структур систолічного типу, які реалізують принцип циклічної обробки даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

[1] Пономаренко, В.С., Голубничий, Д.Ю. & Третяк, В.Ф. (2005). *Цілочисельне програмування в економіці.* Харків: Видавництво ХНУ.

[2] Воеводин, В.В. (1986). *Математические модели и методы в параллельных процессах.* Москва: Наука.

[3] Третяк, В.Ф., Місюра, О.М. & Більчук, В.М. (2017). Метод оптимізації структури розподіленої бази даних у вузлах інфокомунікаційної мережі хмарного середовища. *Наука і техніка Повітряних Сил Збройних Сил України*, (1), 92-96.

[4] Третяк, В.Ф. & Пашнєва, А.А. (2017). Оптимізація структури сховища даних у вузлах інфокомунікаційної мережі хмарного середовища. *Системи управління, навігації та зв'язку*, 4(44), 122-128

[5] Третяк, В.Ф., Кужель, І.Є. & Приходько, В.М. (2010). Використання технології реплікації у системі управління розподіленими базами даних. *Збірник наукових праць Харківського університету Повітряних Сил*, 2(24), 109-114.

[6] Третяк, В.Ф., Лістровий, С.В. а ін. (2014). Патент на корисну модель № 91199, Україна, G06 F 15/00 G06 F 17/00. Пристрій для визначення маршруту в графі. № u201400645; заяв.

23.01.2014; опубл. 25.06.2014.
[7] Третяк, В.Ф., Лістровий, С.В. а ін. (2014). Патент на корисну модель № 88779, Україна, G06 F 15/00. Пристрій для рішення задачі цілочисельного лінійного програмування з булевими змінними на основі рангового підходу. № u201313904; заяв. 29.11.2013; опубл. 25.03.2014.
[8] Третяк, В.Ф., Лістровий, С.В. а ін. (2014). Патент на корисну модель № 92959, Україна, МПК G06 F15/00. Спосіб динамічного кодування та захисту інформаційного ресурсу в інфокомунікаційних системах. № u201403988; заяв. 14.04.2014; опубл. 10.09.2014.