УДК 519.684.4

ПАРАЛЕЛЬНИЙ АЛГОРИТМ СИНТЕЗУ ЕМПІРИЧНИХ МОДЕЛЕЙ ОПТИМАЛЬНОЇ СКЛАДНОСТІ НА ЗАСАДАХ ГЕНЕТИЧНИХ АЛГОРИТМІВ

М. І. Горбійчук, М. О. Слабінога, В. М. Медведчук

Івано-Франківський національний технічний університет нафти і газу,
вул. Карпатська, 15, м. Івано-Франківськ, 76019, e-mail: gorb@nung.edu.ua

Успішна реалізація індуктивного методу самоорганізації моделей можлива лише у тому випадку, коли число вхідних величин і степінь полінома невеликі числа. Цей недолік індуктивного методу самоорганізації моделей у значній мірі вдається усунути, якщо математичну модель будувати з використанням методу, який ґрунтується на ідеях генетичних алгоритмів. Розроблений метод значно розширює клас емпіричних моделей і дозволяє синтезувати моделі оптимальної складності спираючись на зовнішній критерій відбору моделей. У той же час зі збільшення розмірності задачі синтезу емпіричних моделей зростають затрати машинного часу на їх програмну реалізацію. Тому актуальною науковою задачею є зменшення затрат машинного часу, що дозволить синтезувати емпіричні моделі високої розмірності. Одним із шляхів розв'язання поставленої задачі – розпаралелення алгоритму синтезу моделей оптимальної складності. Аналіз алгоритму побудови емпіричних моделей оптимальної складності показав, що найбільш затратними операціями є розв'язання системи лінійних алгебраїчних рівнянь та обчислення виходу системи. Ці операції виконуються багаторазово. Для зменшення затрат машинного часу розроблені паралельні алгоритми та визначені їх характеристики – прискорення та ефективність.
Ключові слова: система, ген, хромосома, критерій пристосування, розмірність задачі, система рівнянь, програмна реалізація, прискорення, ефективність.

Успешная реализация индуктивного метода самоорганизации моделей возможна только в том случае, когда число входных величин и степень полинома небольшие числа. Этот недостаток индуктивного метода самоорганизации моделей в значительной степени удается устранить, если математическую модель строить с использованием метода, основанного на идеях генетических алгоритмов. Разработанный метод значительно расширяет класс эмпирических моделей и позволяет синтезировать модели оптимальной сложности, опираясь на внешний критерий отбора моделей. В то же время с увеличением размерности задачи синтеза эмпирических моделей растут затраты машинного времени на их программную реализацию. Поэтому актуальной научной задачей является уменьшение затрат машинного времени, что позволит синтезировать эмпирические модели высокой размерности. Одним из путей решения поставленной задачи - распараллеливание алгоритма синтеза моделей оптимальной сложности. Анализ алгоритма построения эмпирических моделей оптимальной сложности показал, что наиболее затратными операциями являются решения системы линейных алгебраических уравнений и вычисления выхода системы. Эти операции выполняются многократно. Для уменьшения затрат машинного времени разработаны параллельные алгоритмы и определены их характеристики - ускорение и эффективность.
Ключевые слова: система, ген, хромосома, критерий приспособления, размерность задачи, система уравнений, программная реализация, ускорение, эффективность.

The successful implementation of the of inductive method for self-organizing model is possible only if the number of input variables and the degree of the polynomial are small numbers. This lack of inductive method for self-organizing models can be resolved if the mathematical model is built using a method that is based on the idea of genetic algorithms. The method significantly extends the class of empirical models and allows to synthesize optimal model complexity based on external criteria of selection models. At the same time, the increase of the empirical models synthesis problem dimension causes the increase of their software implementation time. Therefore, the actual scientific problem is to reduce the computing time, and solving this problem will allow the synthesis of high dimension empirical model. The parallelization algorithm for optimal complexity model synthesis is one of the possible solutions. Analysis algorithm for constructing empirical models of optimal complexity showed that the most costly operations are the

solution of linear algebraic equations systems and the system output computing. These operations are performed in loop. To reduce the cost of computing time the parallel algorithms are developed and their characteristics, acceleration and efficiency, are defined.
Key words: gene, chromosome, criterion of adaptation, the dimension problem of the system of equations, software implementation, the acceleration efficiency.

Вступ

В основі емпіричного моделювання багатьох явищ і процесів лежить широко відомий метод найменших квадратів (МНК), зміст якого у наступному. Нехай розглядається деякий об'єкт чи система, функціонування яких характеризується вектором вхідних величин $\bar{x} = (x_1, x_2, \mathbf{K} x_n)^T$ і вихідною величиною y. У результаті спостережень за вхідними і вихідною величинами отримують множину значень – $\bar{x}^{(1)}$, $\bar{x}^{(2)}$, $\mathbf{K}$, $\bar{x}^{(N)}$ та $Y^{(1)}$, $Y^{(2)}$, $\mathbf{K}$, $Y^{(N)}$, де N - кількість спостережень. Сукупність векторів $\bar{x}^{(1)}$, $\bar{x}^{(2)}$, $\mathbf{K}$, $\bar{x}^{(N)}$ утворюють так звану матрицю спостережень, рядки якої $\bar{x}^{(1)}$, $\bar{x}^{(2)}$, $\mathbf{K}$, $\bar{x}^{(N)}$, тобто

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \mathbf{L} & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \mathbf{L} & x_n^{(2)} \\ \mathbf{L} & \mathbf{L} & \mathbf{L} & \mathbf{L} \\ x_1^{(N)} & x_2^{(N)} & \mathbf{L} & x_n^{(N)} \end{bmatrix}.$$

У МНК припускають, що структура моделі відома, яку частіше всього вибирають лінійною відносно її параметрів, тобто

$$y = \sum_{k=0}^{r} a_k f_k(\bar{x}), \qquad (1)$$

де a_k, $k = \overline{1,r}$ – параметри моделі.

Задача полягає у тому, щоб на основі спостережень за вхідними і вихідною величинами, визначити параметри a_k, $k = \overline{1,r}$ моделі (1) таким чином, щоб якомога точніше наблизити вихід системи до виходу моделі. За критерій такого наближення вибирають суму квадратів відхилень розрахункових $y^{(i)}$ від експериментальних значень $Y^{(i)}$, $i = \overline{1,N}$. При відомій матриці спостережень X і вибраних параметрах моделі (1) можна обчислити її вихід

$$\bar{y} = F\bar{a}, \qquad (2)$$

де $F = \begin{bmatrix} f_0(\bar{x}^{(1)}) & f_1(\bar{x}^{(1)}) & \mathbf{L} & f_r(\bar{x}^{(1)}) \\ f_0(\bar{x}^{(2)}) & f_1(\bar{x}^{(2)}) & \mathbf{L} & f_r(\bar{x}^{(2)}) \\ \mathbf{L} & \mathbf{L} & \mathbf{L} & \mathbf{L} \\ f_0(\bar{x}^{(N)}) & f_1(\bar{x}^{(N)}) & \mathbf{L} & f_r(\bar{x}^{(N)}) \end{bmatrix}$;

$\bar{a} = (a_0, a_1, \mathbf{K}, a_r)$ – вектор параметрів моделі (1);

$\bar{y} = (y^{(1)}, y^{(2)}, \mathbf{K}, y^{(N)})$ – обчислені значення моделі (1) у кожній точці спостережень.

За відомими значеннями $Y^{(i)}$ та $y^{(i)}$ обчислити критерій наближення (апроксимації)

$$J(\bar{a}) = \sum_{i=1}^{N} \left(Y^{(i)} - y^{(i)}\right)^2, \qquad (3)$$

який подамо у векторному вигляді

$$J(\bar{a}) = \left(\bar{Y} - \bar{y}\right)^T \left(\bar{Y} - \bar{y}\right).$$

Враховуючи значення $\bar{y}$, яке визначається формулою (2), отримуємо, що

$$J(\bar{a}) = \left(\bar{Y} - F\bar{a}\right)^T \left(\bar{Y} - F\bar{a}\right). \qquad (4)$$

Значення параметрів моделі (1) обчислюють із умови, що критерій наближення (4) набуде мінімального значення відносно вектора параметрів $\bar{a}$. Мінімізація (4) приводить до такого результату:

$$M_F \bar{a} = F^T \bar{Y}, \qquad (5)$$

де $M_F = F^T F$ – матриця Фішера.

У тому випадку, коли розмірність вектора $\bar{a}$ невелика і матриця M добре обумовлена, із рівняння (5) можна безпосередньо визначити

$$\bar{a} = M_F^{-1} F^T \bar{Y}.$$

У переважній більшості випадків структура моделі (1) невідома, що приводить до необхідності вибору як структури самої моделі, так і її параметрів. Критерій наближення (3) є внутрішнім критерієм [1] і його застосування приводить до помилкового висновку: чим складніша модель, тим вона точніша. Оскільки на вихід системи накладається перешкода (допускають, що вона адитивна), то надмірна

точність моделі може значно спотворити об'єктивно існуючу функціональну залежність між виходом системи і її входами.

Тому для вибору структури моделі (1) акад. О. Г. Івахненком був запропонований метод, який дістав назву індуктивний метод самоорганізації моделей [2], ідейну сторону якого визначає теорема Геделя. Ця теорема стверджує, що ніяка система аксіом не може бути логічно замкнутою: завжди знайдеться така теорема, для доведення якої необхідне зовнішнє доповнення – розширення початкової системи аксіом. Стосовно задачі синтезу моделі геделівський підхід означає застосування зовнішнього критерію для однозначного вибору структури моделі із заданого класу моделей. Критерій називають зовнішнім, якщо його обчислення ґрунтується на даних, які не використовувались при розв'язанні задачі (3). Це означає, що всі дані, отримані у результаті експерименту, розбиваються на дві частини – навчальну N_A і перевірну N_B.

Переважно для вибору структури моделі (1) використовують критерії регулярності

$$\Delta^2(B)=\frac{\sum_{i=1}^{N_B}\left(Y^{(i)}(B)-y^{(i)}(B)\right)^2}{\sum_{i=1}^{N_B}Y^{(i)}(B)^2} \qquad (6)$$

і зміщення

$$\Delta^2(A,B)=\frac{\sum_{i=1}^{N}\left(y^{(i)}(A)-y^{(i)}(B)\right)^2}{\sum_{i=1}^{N}\left(Y^{(i)}\right)^2}, \qquad (7)$$

де $y^{(i)}(A)$, $y^{(i)}(B)$ – значення виходу моделі, що обчислені відповідно на множинах експериментальних значень N_A і N_B.

Якщо вибраний критерій регулярності (6), то витирають такий розподіл даних експерименту [3]: $N_A=0,7N$ і $N_B=0,3N$, а при виборі критерію (7) – $N_A=0,5N$ і $N_B=0,5N$.

Реалізація індуктивного методу самоорганізації моделей здійснюється поетапно: перший етап генерування моделей-претендентів (у певному порядку підвищення їх складності); другий вибір найкращої моделі за мінімумом критерію селекції (6) або (7).

Розрізняють три способи генерування моделей–претендентів. Перший із них комбінаторний [3], за яким вибір моделей–претендентів здійснюється шляхом прирівнювання до нуля деяких коефіцієнтів у виразі (1), що дає змогу отримати сукупність моделей. Вибір найкращої із них здійснюється на основі одного із критеріїв селекції (6) або (7). Другий спосіб відомий як метод групового врахування аргументів (МГУА) [4], у якому генерування моделей-претендентів здійснюється за допомогою багаторядної процедури. У першому ряду селекції комбінують всі можливі пари аргументів (вхідних величин), і для кожної із них утворюють моделі, наприклад, у вигляді повного полінома. Із всіх часткових моделей, утворених у такий спосіб, вибирають K найкращих за одним із критеріїв селекції. Із виходів цих K моделей знову утворюють комбінації всіх можливих пар, які є входами другого ряду селекції. Для кожної із цих пар знову формують часткові моделі, із яких вибирають K найкращих. Процес нарощування рядів селекції відбувається до тих пір поки вибрий критерій селекції зменшує своє значення. Оцінка параметрів часткових моделей здійснюється за допомогою МНК. Третій метод [2] подібний до другого. Різниця лише у тому, що на кожному ряді селекції часткові моделі утворюють шляхом прирівнювання до нуля певного числа їх коефіцієнтів.

Недоліком комбінаторного методу селекції моделей є необхідність великого числа перебору часткових моделей. Якщо початковою моделлю вибраний повний поліном степені m, то загальне число моделей претендентів буде 2^M-1, де M – загальне число членів початкового полінома. Навіть сучасні ЕОМ не здатні реалізувати комбінаторний алгоритм селекції моделей при значній кількості вхідних величин і при високій степені початкового полінома. МГУА синтезує моделі, в яких фігурують проміжні змінні, що значно утруднює процес переходу до вхідних змінних системи. Сказане відноситься і до третього методу, оскільки він по суті є модифікацією МГУА.

Із усіх трьох методів найпривабливішим є комбінаторний метод, оскільки він дає змогу отримати оптимальну модель, де аргументами виступають вхідні змінні системи. Для зняття проблеми великої розмірності комбінаторного методу застосуємо генетичний підхід. Як емпіричну модель виберемо поліном степені m

$$y=\sum_{i=0}^{M-1}a_i\prod_{j=1}^{n}x_j^{s_{ji}}, \qquad (8)$$

де a_i – коефіцієнти полінома; s_{ji} – степені аргументів, які повинні задовольняти обмеженню:

$$\sum_{j=1}^{n} s_{ji} \le m .$$

Число членів M полінома (8) визначають за такою формулою [5]:

$$M = \frac{(m+n)!}{m!n!}. \qquad (9)$$

При комбінаторному методі синтезу моделей оптимальної складності із повного полінома (8) отримують модель, де частина коефіцієнтів a_i, $i=\overline{1,M}$ моделі (8) набувають нульових значень. Утворимо впорядковану послідовність, де на i-тому місці буде знаходитись одиниця або нуль в залежності від того чи параметр a_i, $i=\overline{1,M}$ моделі (8) буде відмінним від нуля або матиме значення нуль. У теорії генетичних алгоритмів така упорядкована послідовність носить назву хромосоми, а атомарний елемент (одиниця або нуль) хромосоми – це ген. Набір хромосом утворює популяцію. Важливим поняттям у теорії генетичних алгоритмів є функція пристосування, яка визначає ступінь пристосованості окремих особин у популяції. Вона дає змогу із усієї популяції вибрати ті особини, що є найбільш пристосованими, тобто ті яким відповідає найменше (найбільше) значення функції пристосування. У задачі синтезу моделей оптимальної складності як функцію пристосування виберемо критерій селекції (6) або (7).

Як і класичний генетичний алгоритм [6], алгоритм синтезу моделей оптимальної складності, що ґрунтується на засадах генетичних алгоритмів, складається із таких кроків.

К1. Формування початкової популяції (ініціалізація). На першому кроці роботи алгоритму випадковим чином формулюється популяція із I особин, кожна із яких є хромосомою довжиною M. Число генів у хромосомі визначається за формулою (9).

К2. Оцінка пристосованості хромосоми у популяції. Для кожної хромосоми обчислюється значення критерію селекції (6) або (7) наступним чином. Якщо вибраний критерій селекції (6), то формуються матриці F_A і F_B розмірами $N_A \times M$ і $N_B \times M$. Із матриці F_A вилучається i-тий стовпець, якщо на i-тій позиції хромосоми знаходиться нуль; якщо одиниця, то відповідний стовпець залишається без змін. У результаті отримаємо матрицю $\tilde{F}_A$, із якої вилучено c стовпців (за кількістю нулів у хромосомі). Розмір такої матриці буде $N_A \times (M-c)$. Аналогічно отримаємо матрицю $\tilde{F}_B$ розміром $N_B \times (M-c)$. На множині точок N_A обчислюються ненульові коефіцієнти a_{Aj}, $j=\overline{1,M-c}$ моделі (8) шляхом розв'язання нормального рівняння Гауса, яке видозмінюється наступним чином:

$$\tilde{M}_{F,A}\bar{a}_A = \tilde{F}_A^T \bar{Y}_A, \qquad (10)$$

де $\bar{a}_A = \left(a_{A0}, a_{A1}, \mathbf{K}, a_{A,M-c-1}\right)^T$ – вектор ненульових параметрів моделі, який асоційований з черговою хромосомою; $\tilde{M}_{F,A} = \tilde{F}_A^T \tilde{F}_A$. $\bar{Y}_A = \left(Y^{(1)}, Y^{(2)}, \mathbf{K}, Y^{(N_A)}\right)$ – вектор експериментальних значень на множині N_A.

За знайденими коефіцієнтами $\bar{a}_A$ поліноміальної моделі на множині точок N_B обчислюють значення

$$\bar{y}(B) = \tilde{F}_B \bar{a}_A . \qquad (11)$$

Знаючи $\bar{y}(B)$, за формулою (6) обчислюють значення функції пристосування $\Delta_j^2(B)$, $j=\overline{1,I}$ для кожної хромосоми із початкової популяції.

У тому випадку, коли як функцію пристосування використовують критерій зміщення (7), тоді складають рівняння (10), яке розв'язують методом Гауса відносно параметрів $\bar{a}_A$. Після цього обчислюють $\bar{y}(A) = \tilde{F}_A \bar{a}_A$ за формулою (11). Отримані значення $\bar{y}(A)$ і $\bar{y}(B)$ дають змогу знайти значення $\Delta_j^2(A,B)$, $j=\overline{1,I}$ для кожної хромосоми із популяції.

К3. Перевірка умови зупинки алгоритму. Визначають

$$\Delta_m^2(B) = \min_j \Delta_j^2(B), \qquad (12)$$

$$\Delta_m^2(A,B) = \min_j \Delta_j^2(A,B). \qquad (13)$$

Якщо мінімальне значення (12) або (13) критерію селекції (6) або (7) не перевищує деякого заданого значення E, то відбувається зупинка обчислень. Зупинка обчислень також може відбутись у випадку, коли у результаті виконання алгоритму немає суттєвого зменшення функції пристосування або у тому випадку, коли виконано задане число ітерацій.

Після виконання однієї із трьох умов із чергової популяції вибирається та хромосома

ch^*, для якої виконується умова (12) або (13). Ця хромосома задає структуру моделі оптимальної складності і формує матрицю F^* таким чином, що із початкової матриці вилучаються стовпці, які асоційовані з нульовими значеннями відповідних генів. Перерахунок параметрів моделі (8) здійснюється на всій множині точок N за допомогою МНК.

К4. Селекція хромосом. За розрахованими на другому кроці алгоритму значеннями функції пристосування здійснюється відбір тих хромосом, які будуть брати участь у створенні потомків для нової популяції. Такий відбір проводиться за принципом природного відбору, коли найбільші шанси у створенні нової популяції мають хромосоми з найкращими значеннями функції пристосування (6) або (7). Найпоширенішим методами селекції є метод рулетки і турнірний метод [6]. Метод рулетки можна застосовувати тоді, коли функція пристосованості додатна, що робить його придатним лише для задач максимізації. Турнірний метод можна використовувати як у задачах максимізації, так і у задачах мінімізації.

При турнірній селекції всі хромосоми розбиваються на підгрупи з наступним вибором із кожної підгрупи хромосоми з найкращою пристосованістю. Підгрупи можуть мати довільний розмір, але частіше за все популяцію ділять на підгрупи по 2 – 3 особини у кожній.

К5. Формування нової популяції потомків. Над відібраними особинами на четвертому кроці алгоритму здійснюють відозміну за допомогою двох основних операторів: схрещування і мутації. Слід зауважити, що оператор мутації застосовується рідше у порівнянні з оператором схрещування. Вірогідність схрещування досить висока ($0 \le P_c \le 1$). Тоді як вірогідність мутації складає $0 \le P_m \le 0{,}1$. Ймовірність мутації P_m можна задати шляхом вибору випадкового числа із інтервалу $[0;0{,}1]$ для кожного гена. Мутації підлягають ті гени (шляхом заміни одиниці на нуль і навпаки), для яких розігране число виявиться меншим або рівним P_m. Мутація може здійснюватись як над пулом родичів, так і над пулом потомків. Оператор схрещування застосовується до пулу потомків у такий спосіб. Із утвореної популяції особин випадковим чином вибирається пара хромосом і генерується випадкове число P_z із інтервалу $[0;1]$. Якщо виконується умова $P_z \le P_c$, то над парою хромосом здійснюється схрещування. У протилежному випадку пара хромосом залишається без змін. Для пари хромосом, які підлягають схрещуванню, випадковим чином розігрується позиція гена (локус), яка визначає точку схрещування. Оскільки кожна хромосома має M генів, то точка схрещування L_c – це натуральне число менше M. Тому фіксація точки схрещування зводиться до вибору натурального випадкового числа із інтервалу $[1; L_c - 1]$. Дія оператора схрещування приводить до того, що із пари родичів утворюється нова пара потомків: на позиціях від $L_c + 1$ до M хромосоми у парі обмінюються своїми генами.

К6. Після виконання оператора схрещування відбувається перехід до кроку *К2.*

Реалізація генетичного алгоритму синтезу емпіричних моделей оптимальної складності показала, що значна частина машинного часу витрачається на розв'язування системи лінійних алгебраїчних рівнянь (5). Так у роботі [7] синтезована емпірична модель, яка описує залежність температури газу T_v на виході турбіни низького тиску газоперекачувального агрегату (ГПА) від таких технологічних параметрів як: частота обертання $n_н$ вала відцентрового нагнітача, температура T_{in} та тиск P_{in} природного газу на вході нагнітача, ступінь підвищення тиску ε, температура T_c та тиск P_c навколишнього середовища. Проведений авторами роботи [7] аналіз експериментальних даних показав, що зміна тиску навколишнього середовища незначно впливає на зміну величини T_v. Значно суттєвіший вплив на T_v мають попередні значення ступені підвищення тиску. Це пояснюється тим, що ГПА разом з прилеглими трубопроводами є динамічною системою, у якій відбувається накопичення газу, що спричинює не миттєву, а поступову зміну тиску природного газу на виході відцентрового нагнітача. Тому для опису залежності T_v від технологічних параметрів була вибрана модель (8), у якій степінь полінома $m = 4$ і $x_1^{(i)} = n_н^{(i)}$, $x_2^{(i)} = \varepsilon^{(i)}$, $x_3^{(i)} = \varepsilon^{(i-1)}$, $x_4^{(i)} = \varepsilon^{(i-2)}$, $x_5^{(i)} = \varepsilon^{(i-3)}$, $x_6^{(i)} = P_{in}^{(i)}$, $x_7^{(i)} = T_c^{(i)}$, $i = \overline{4, N}$. Таким чином, розмірність початкової моделі (8) для $m = 4$ і $n = 7$ згідно формули (9) буде - $M = 330$.

Нормальне рівняння (10) запишемо у такій формі:

$$\hat{A}\bar{a}_A = \hat{b}, \qquad (14)$$

де $\mathbf{A} = \mathbf{M}_{F,a}$; $\mathbf{b} = \mathbf{F}_A^T \overline{Y}_A$.

Рівняння (14) будемо розв'язувати методом гаусового виключення, де система (14) приводиться до верхньої діагональної форми за такими формулами [8]:

$$\mathbf{a}_{ig}^{(i)} = \frac{\mathbf{a}_{ig}^{(i-1)}}{\mathbf{a}_{ii}^{(i-1)}},\ i = \overline{1, M-c}; \qquad (15)$$

$$\mathbf{a}_{kg}^{(i)} = \mathbf{a}_{kg}^{(i-1)} - \mathbf{a}_{ig}^{(i)} \mathbf{a}_{kj}^{(i-1)},\ k = \overline{i+1, M-c},$$

$$j = \overline{1, M-c+1}, \qquad (16)$$

де $\mathbf{a}_{ig}^{(i)}$, $\mathbf{a}_{kg}^{(i)}$ – i-тий та k–тий рядок розширеної матриці, яка утворена шляхом приєднання до матриці $\mathbf{A}$ вектор-стовпця $\mathbf{b}$; $\mathbf{a}_{ig}^{(0)} = \mathbf{a}_{ig}$, $i = \overline{1, M-c}$; $\mathbf{a}_{kg}^{(0)} = \mathbf{a}_{kg}, k = \overline{i+1, M-c}$.

В обчислювальному процесі цикл за індексом k є внутрішнім по відношенню до зовнішнього циклу за індексом i. Результатом застосування ітераційних процедур (15) і (16) є верхня діагональна матриця з одиницями на головній діагоналі

$$U = \begin{bmatrix} 1 & u_{12} & u_{13} & \mathbf{L} & u_{1,M-c} & u_{1,M-c+1} \\ 0 & 1 & u_{23} & \mathbf{L} & u_{2,M-c} & u_{2,M-c+1} \\ 0 & 0 & 0 & \mathbf{L} & u_{3,M-c} & u_{3,M-c+1} \\ \mathbf{L} & \mathbf{L} & \mathbf{L} & \mathbf{L} & \mathbf{L} & \mathbf{L} \\ 0 & 0 & 0 & \mathbf{L} & 1 & u_{M-c,M-c+1} \end{bmatrix},$$

де $u_{ij} = a_{ij}^{(i)}$, $i = \overline{1, M-c}$, $j = \overline{i+1, M-c+1}$.

Таким чином, рівняння (14) перетвориться в еквівалентну систему лінійних алгебраїчних рівнянь

$$\mathbf{U}\,\overline{a}_A = \overline{b}_A, \qquad (17)$$

де $\mathbf{U}$ – квадратна матриця розміром $M-c$, яка утворена із матриці U шляхом вилучення останнього стовпця; $b_{A,i} = u_{i,M-c+1}$, $i = \overline{1, M-c}$.

Рівняння (14) розв'язується методом зворотної прогонки, починаючи з останнього рівняння. Очевидно, що

$$a_{A,M-c} = b_{A,M-c}. \qquad (18)$$

Інші значення $a_{A,i}$ обчислюються за такою ітераційною процедурою:

$$a_{A,i} = b_{A,i} - \sum_{j=i+1}^{M-c} u_{ij} a_{A,j},\ i = \overline{M-c-1, 1}. \qquad (19)$$

На відміну від класичного методу Гауса [9], де приведення до діагонального вигляду матриці U відбувається шляхом відніманням одних рівнянь, помножених на відповідні числа, із інших рівнянь, у результаті отримують верхню діагональну матрицю з ненульовими коефіцієнтами на головній діагоналі, тобто $u_{ii} \neq 0$, $i = \overline{1, M-c}$. У такому випадку для отримання розв'язку рівняння (14) необхідно праву частину співвідношення (19) розділити на u_{ii}.

Отже, після приведення системи рівнянь (14) до виду (17) відпадає необхідність в операції ділення при обчислення $a_{A,i}$. Як відомо, на виконання операції ділення витрачається найбільше машинного часу у порівнянні з іншими алгебраїчними операціями. Тому обчислення параметрів моделі (8) за рекурентними співвідношеннями (18) і (19) дає відчутну економію машинного часу у порівнянні з класичним методом Гауса.

Сказане справедливо і для LU – метода, у відповідності з яким матриця $\mathbf{A}$ подається у вигляді добутку двох матриць L і U_R. Перша із них – нижня діагональна з одиницями на головній діагоналі, а друга – верхня діагональна матриця. Потім прямою і зворотною прогонками розв'язуються трикутні системи

$$L\overline{\gamma} = b_A,\ U_R \overline{a}_A = \overline{\gamma}.$$

Іншим способом зменшення затрат машинного часу є розпаралелення алгоритму приведення матриці $\mathbf{A}$ до верхньої трикутної матриці U. На цю операцію витрачається більша частина часу з всієї частки часу, яка витрачається на розв'язування системи рівнянь (14).

Суть такого алгоритму у наступному. На стадії ініціалізації (початковій стадії) вся матриця $\mathbf{A}$ розміщається у робочому просторі першого процесу – майстра. Перший крок – майстер відправляє, по можливості, рівні шари матриці $\mathbf{A}$ іншим процесам (робітникам) так, що кожний із процесів, включаючи і перший отримує підматриці $\mathbf{A}_i$, $i = \overline{1, q}$, де q – загальна кількість процесів. Після відправки майстер нормує перший рядок своєї підматриці $\mathbf{A}_1$ за формулою (15) і тут же відсилає значення $\mathbf{a}_{1g}^{(1)}$ іншим робітникам. Робітники, отримавши рядок $\mathbf{a}_{1g}^{(1)}$, кожний із них у відповідності з формулою (16), де $k = \overline{1, M-c}$, перераховують елементи

своїх підматриць, включаючи і елемент $a_{qi,M-c+1}$. Одночасно майстер обчислює нові значення елементів за формулою (16), починаючи з другого рядка. У результаті такого перерахунку робітники отримують підматриці A_i, $i=\overline{2,q}$, у яких нульовими будуть перші стовпці. У матриці A_1 перший стовпець буде мати нульові елементи, крім першого $a_{11}^{(1)}$, який дорівнюватиме одиниці.

Другий крок – майстер нормує другий рядок своєї підматриці і значення $a_{2g}^{(2)}$, яке обчислене за формулою (15), посилає всім робітникам. Ті, одночасно з майстром, модифікують свої підматриці у відповідності з процедурою (16). Причому майстер змінює індекс k від 3 до $M-c$, а робітники відповідно від 2 до $M-c$. У підсумку довжина другого рядка з ненульовими елементами скоротилася на одиницю (перший елемент дорівнює нулю, а другий – одиниці). Підматриці, які розміщені у робочих просторах робітників, будуть вміщувати по два нульових стовпці.

Процес модифікації елементів підматриць здійснюється циклічно за наведеною схемою до тих пір поки майстер не завершить приводити матрицю до верхнього діагонального вигляду. У підсумку після закінчення першого циклу у робочому просторі майстра буде зберігатись верхня прямокутна матриця з одиницями на головній діагоналі, а у робочих просторах робітників отримаємо підматриці, число нульових стовпців яких визначається числом рядків матриці майстра.

На початку другого циклу робітники надсилають свої підматриці майстру, де відбувається їх об’єднання. Потім майстер об’єднану підматрицю розділяє на q частин і розсилає їх всім робітникам, включаючи і себе. Майстер нормує перший рядок своєї підматриці. При цьому провідним елементом буде перший ненульовий елемент першого рядка підматриці. Модифікований перший рядок, у відповідності з формулою (15), майстер розсилає всім робітникам, котрі модифікують свої рядки за формулою (16). У подальшому обчислення відбуваються так, як це було у першому циклі. Алгоритм продовжує працювати до тих пір поки розмір чергового шару, що підлягає відправленню, не стане рівним одиниці.

Ефективність розробленого алгоритму перевірялася на процесорі з 4 фізичними ядрами, кожне із яких розбивалось на два віртуальні потоки. Входом алгоритму служили квадратні матриці розміром 800, 1200, 2400 і 3200. Результати роботи алгоритму відображає табл. 1.

Основними показниками, які характеризують паралельний алгоритм, є його прискорення S_q та ефективність E_q [10].

Прискоренням паралельного алгоритму називають відношення часу виконання алгоритму T_1 одним процесом до часу виконання алгоритму T_q у системі із q процесів, тобто

$$S_q = T_1 / T_q . \qquad (20)$$

За даними табл. 1 побудована залежність прискорення паралельного алгоритму від кількості процесів (рис. 1), яка показує, що із збільшенням кількості процесів величини S_q зростають і при певному значенню q досягають максимуму, а потім значення S_q зменшується. Це пояснюється тим, що з ростом кількості процесів зростають затрати часу на обмін даними між ними. Очевидно, що для кожного розміру матриці існує оптимальне число процесів. У нашому випадку для $N=800$ оптимальним числом процесів є 8, а для $N \in \{1200; 2400; 3200\}$ будемо мати $q^* = 4$.

Таблиця 1 – Результати приведення квадратних матриць до верхнього діагонального вигляду

Розмір матриці	Кількість процесів/ Затрати часу на виконання алгоритму, с					
	1	2	4	8	16	32
800	1,03093	0,524869	0,428326	0,304387	0,451943	0,593357
1600	8,71222	4,69282	3,47132	4,12222	5,06584	5,78872
2400	29,3391	15,3732	13,4345	14,4151	15,6563	17,064
3200	67,7167	40,8149	39,3524	37,1931	32,4264	36,5317

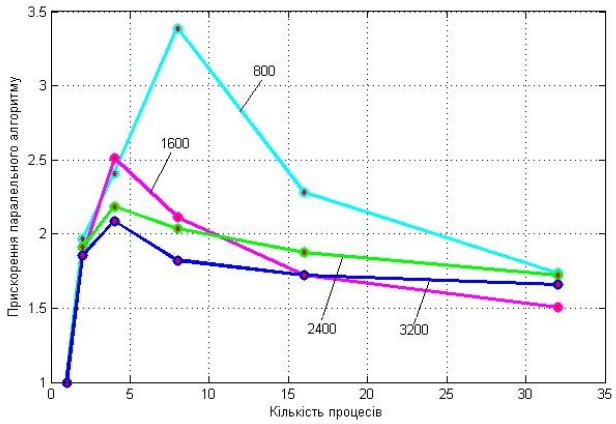

Рисунок 1 – Залежність прискорення паралельного алгоритму від кількості процесів

Ефективністю алгоритму паралельного алгоритму називається величина

$$E_q = S_q / q. \qquad (21)$$

Оскільки $S_q \le q$ [10], то завжди $E_q \le 1$. Очевидно, що $E_q = 1$ в ідеальному випадку, коли $S_q = q$. Сказане ілюструє рис. 2, із якого видно, що $E_q < 1$. Крім того існує максимальне значення E_q^* для різних розмінностей матриць. Як видно із рис.2 зі збільшенням розмірності матриці падає значення E_q^*.

При реалізації генетичного алгоритму для кожної хромосоми необхідно не тільки розв'язувати рівняння (14), але й обчислювати вихід моделі для визначення пристосованості хромосоми у процесі відбору на кроці *K2*. При цьому багаторазово приходиться обчислювати добуток матриці на вектор, який у загальному випадку матиме такий вигляд:

$$\bar{y} = F\bar{a}, \qquad (22)$$

де $\bar{y}$ – вектор значень виходу моделі; F – матриця спостережень; $\bar{a}$ – вектор параметрів моделі. Розміри матриці F і вектора $\bar{a}$ визначаються кількістю одиниць у хромосомі.

Кожний k–тий компонент вектора $\bar{y}$ обчислюється шляхом скалярного множення k–того рядка на вектор $\bar{a}$. Кількість таких скалярних добутків визначається кількістю рядків матриці F. Припустимо, що розмірність матриці F – $p \times l$. Тоді для обчислення добутку (22) необхідно виконати pl операцій множення і $p(l-1)$ операцій додавання. Загальна кількість операцій множення і додавання, які необхідно виконати для обчислення добутку (22), становитиме

$$T_1 = p\left(l\tau_m + (l-1)\tau_a\right) Z = p(2l-1).$$

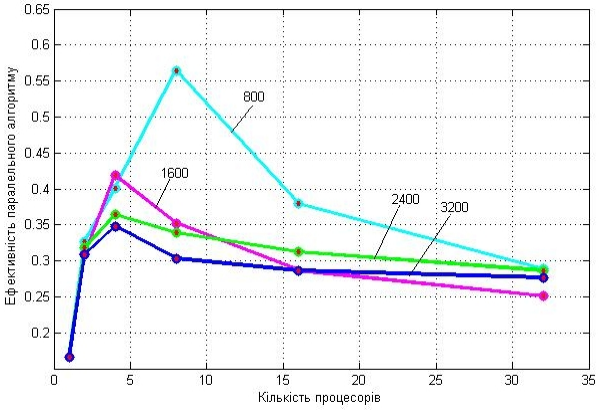

Рисунок 2 – Залежність ефективності паралельного алгоритму від кількості процесів

Будемо обчислювати добуток (22) на q паралельних процесах. Тоді

$$\begin{bmatrix} F_1 \\ F_2 \\ \mathbf{L} \\ F_q \end{bmatrix} \bar{a} = \begin{bmatrix} F_1\bar{a} \\ F_2\bar{a} \\ \mathbf{L} \\ F_q\bar{a} \end{bmatrix},$$

де підматриця F_i має розмір $p_i \times l$, $i = \overline{1,q}$.

Оскільки у загальному випадку число рядків матриці F не кратне числу q, то матрицю F розділимо на дві матриці $\hat{F}_1$ і $\hat{F}_2$. Потім матрицю $\hat{F}_1$ розділимо на s однакових за розміром підматриць, а матрицю $\hat{F}_2$ на $q-s$ однакових за розміром підматриць. Відповідно кожна із s і $q-s$ підматриць будуть мати розміри $p_i^{(1)} \times l$ та $p_k^{(2)} \times l$, де $i = \overline{1,s}$, $k = \overline{1,q-s}$. Оскільки всі процеси працюють паралельно, то кожен із них виконає Z_i операцій додавання і віднімання

$$Z_i = \frac{\pi_i}{q_i}(2l-1), \; i = 1, 2,$$

де $\pi_1 \times l$ і $\pi_2 \times l$ – відповідно розміри матриць $\hat{F}_1$ і $\hat{F}_2$; $q_1 = s$; $q_2 = q - s$.

Припустимо, що операція множення виконується за τ_m, додавання – за τ_a одиниць

часу. Тоді час T_1 виконання операції (22) одним процесом обчислимо за такою формулою (без врахування затрат на обмін даними, синхронізацію і конфлікти пам'яті):

$$T_1 = p\left(l\tau_m + (l-1)\tau_a\right),$$

а затрати часи T_q у системі із q процесорів будуть такими:

$$T_q = \max\left(T_{q1}, T_{q2}\right),$$

де $T_{qi} = \frac{\pi_i}{q_i}\left(l\tau_m + (l-1)\tau_a\right)$, $i = 1, 2$.

Отже,

$$T_q = \max\left(\frac{\pi_1}{s}, \frac{\pi_2}{q-s}\right)\left(l\tau_m + (l-1)\tau_a\right).$$

Знаючи T_1 і T_q, за формулами (20) і (21) обчислимо прискорення

$$S_q = \frac{p}{\max\left(\frac{\pi_1}{s}, \frac{\pi_2}{q-s}\right)}$$

і ефективність

$$E_q = \frac{p}{q\max\left(\frac{\pi_1}{s}, \frac{\pi_2}{q-s}\right)}$$

паралельного алгоритму.

У тому випадку, коли число рядків p матриці F кратне числу паралельних процесів q, тоді $\pi_1 = \pi_2 = \frac{p}{2}$, $s = \frac{q}{2}$ і $\max\left(\frac{\pi_1}{s}, \frac{\pi_2}{q-s}\right) = \frac{p}{q}$. Отже,

$$S_q = q \text{ і } E_q = 1. \qquad (23)$$

Формули (23) визначають прискорення і ефективність паралельного алгоритму без врахування затрат на обмін даними, синхронізацію і конфлікти пам'яті і визначають асимптотику паралельного алгоритму множення матриці на вектор.

Припустимо, що час T_0 затрачений на обмін даними, синхронізацію і конфлікти пам'яті пропорційний величині T_q, тобто $T_0 = \alpha T_q$. Тоді час затрачений на виконання паралельного алгоритму $\tilde{T}_q = T_q + T_0$, або враховуючи значення T_0, матимемо $\tilde{T}_q = T_q(1+\alpha)$. Тепер знайдемо прискорення паралельного алгоритму з врахуванням часових затрат на обмін даними, синхронізацію і конфлікти пам'яті

$$\tilde{S}_q = \frac{T_1}{T_q(1+\alpha)}.$$

Оскільки $S_q = \frac{T_1}{T_q}$, то $\tilde{S}_q = \frac{S_q}{1+\alpha}$. Із формули (21) випливає, що $\tilde{E}_q = \frac{S_q}{q(1+\alpha)}$. Враховуючи те, що $E_q = \frac{S_q}{q}$ і $E_q = 1$, отримаємо $\tilde{E}_q = \frac{1}{1+\alpha}$.

Таким чином, врахування затрат на обмін даними, синхронізацію і конфлікти пам'яті зменшує показники ефективності роботи паралельного алгоритму і це зменшення відбувається за законом гіперболи: чим більше значення коефіцієнта α тим швидше зменшуються показники $\tilde{S}_q$ і $\tilde{E}_q$. Це означає, що для випадку інтенсивних обмінів, численних конфліктів пам'яті та великих витрат на синхронізацію використання кількох паралельних процесів може виявитись менш вигідним, ніж використання одного.

ВИСНОВКИ

Синтез емпіричних моделей оптимальної складності на засадах генетичних алгоритмів вимагає багаторазового розв'язання системи лінійних алгебраїчних рівнянь і знаходження добутку матриці на вектор, які мають значні розміри. На ці операції витрачається більша частина із загального машинного часу. Для зменшення затрат машинного часу розроблені паралельні алгоритми та визначені їх характеристики – прискорення і ефективність у залежності від кількості паралельних процесів.

Проведені експериментальні дослідження паралельних алгоритмів показали, що є певна залежність між кількістю процесів і показниками, що характеризують їх ефективність. Існує оптимальне число паралельних процесів, які забезпечують максимальне значення таких показників якості паралельних алгоритмів як прискорення та ефективність. Це пояснюється тим, що із збільшенням кількості паралельних процесів зростають затрати машинного часу на обмін даними, синхронізацію та конфлікти пам'яті.

1. Ермаков С. М. Математическая теория оптимального эксперимента: учебное пособие / С. М. Ермаков, А. А. Жиглявский. – М.: Наука, 1987. – 320 с. 2. Ивахненко А. Г. Индуктивный метод самоорганизации моделей сложных систем / А. Г. Ивахненко. – К.: Наукова думка, 1981. – 286 с. 3. Ивахненко А. Г. Справочник по типовым программам моделирования / А. Г. Ивахненко, Ю. В. Коппа, В. С. Степашко и др. – К.: Техніка, 1980. – 180 с. 4. Івахненко О. Г. Передбачення випадкових процесів / О. Г. Іваненко, В. Г. Лапа. – К.: Наукова думка, 1969. – 420 с. 5. Горбійчук М. І. Індуктивний метод побудови математичних моделей газоперекачувальних агрегатів природного газу / М. І. Горбійчук, М. І. Когутяк, Я. І. Заячук // Нафтова і газова промисловість. – 2008. – № 5. – С. 32 – 35. 6. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский; пер. с польск. И. Д. Рудинского. – М.: Горячая линия-Телеком, 2004. – 452 с. 7. Горбійчук М. І. Метод синтезу математичних моделей на засадах генетичних алгоритмів / М. І. Горбійчук, М. І. Когутяк, О. Б. Василенко, І. В. Щупак // Розвідка та розробка нафтових і газових родовищ. – 2009. – № 4 (33). – С. 72-79. 8. Оленев Н. Н. Параллельное программирование в MatLab и его приложение / Н. Н. Оленев, Р. В. Печенкин, А. М. Чернецов. – М.: ВЦ РАН, 2007. – 120 с. 9. Вержбицкий В. М. Основы численных методов: учебник для вузов / В. М. Вержбицкий. – М.: Высшая школа, 2002. – 840 с. 10. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем / Дж. Ортега; пер. с англ. Х. Д. Икрамова, И. Е. Капорина под ред. Х. Д. Икрамова. – М.: Мир, 1991. – 367 с.

Поступила в редакцію 10.12.2013р.

Рекомендували до друку докт. техн. наук, проф. Юрчишин В. М. та докт. техн. наук, доц. Лютак І. З.