КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 519.6 DOI:10.34229/2707-451X.22.2.4

В.К. ЗАДІРАКА, О.М. ХІМІЧ, І.В. ШВІДЧЕНКО

МОДЕЛІ КОМП'ЮТЕРНИХ ОБЧИСЛЕНЬ

Показана важливість урахування похибок заокруглення в сучасних комп'ютерних технологіях розв'язання задач обчислювальної та прикладної математики з заданими значеннями характеристик якості за точністю та швидкодією. Розглянуті наступні класи задач: розв'язання систем лінійних алгебраїчних рівнянь, чисельне інтегрування, задач цифрової обробки сигналів, задача Коші, задач комп'ютерної стеганографії та використання багаторозрядної арифметики для контролю та зменшення похибки заокруглення при розв'язанні задач трансобчислювальної складності.

Ключові слова: *комп'ютерні технології, похибка заокруглення, послідовні, паралельні та квантові моделі обчислень, оцінка складності.*

1. Загальні положення. Постановка задачі. Нехай $F(I_0)$, $A(X)$, $C(Y)$ – класи задач обчислювальної (чи прикладної) математики, алгоритмів та моделей обчислювальних пристроїв (комп'ютерів), а I_0, X, Y – множини параметрів, від яких суттєво залежать відповідні класи задач.

Будемо вважати, що для побудови ε-розв'язку задачі $f \in F \varepsilon \geq 0$ (наближеного розв'язку, деяка міра похибки якого не перевищує $a \in A$) використовується алгоритм , який реалізується на комп'ютері $c \in C$ і орієнтований на використання інформації I_0 про клас F та інформації $I_n(f)$ про конкретну задачу класу. Інформація (інформаційний оператор) $I_n(f)$ може бути задана, наприклад, у вигляді набору функціоналів $I_n(f) = (i_1(f), i_2(f), \ldots, i_n(f))^T$ від елементів задачі f.

Таким чином, для побудови ε-розв'язку використовується модель обчислень, що описується за допомогою $I_0, I_n(f)$, X, Y.

Якість обчислювального процесу (о.п.) переробки вихідних даних, результатом якого є ε-розв'язок, характеризується обчислювальною складністю – кількістю деякого обчислювального ресурсу, необхідного для побудови ε-розв'язку, що називають також ціною або затратами. Найбільш уживаними характеристиками обчислювальної складності є процесорний час $T = T(I_n(f), X, Y, \varepsilon))$ та комп'ютерна пам'ять $M = M(I_n(f), X, Y, \varepsilon)$, необхідні для обчислення ε-розв'язку.

Загальна ситуація побудови наближеного розв'язку задачі при обмежених обчислювальних ресурсах може бути описана умовами:

$$\rho(E(I_n(f), X, Y)) \leq \varepsilon, \quad (1)$$

$$T(I_n(f), X, Y, \varepsilon) \leq T_0(\varepsilon), \quad (2)$$

$$M(I_n(f), X, Y, \varepsilon) \leq M_0(\varepsilon), \quad (3)$$

де ε, T_0, M_0 – задані числа, $\rho(\cdot)$ – деяка міра похибки наближеного розв'язку.

Якість наближеного розв'язку характеризується у загальному випадку повною похибкою $\left(E\left(I_n(f), X, Y\right)\right)$, яка є сумою трьох складових: $E_H\left(I_0, I_n(f), Y\right)$ – похибки, яка обумовлена неточністю вихідної інформації; $E_\mu\left(I_0, I_n(f), X\right)$ – похибки методу; $E_\tau\left(I_n(f), X, Y\right)$ – похибки за рахунок заокруглень.

Нехай $A(\varepsilon, X)\left(A(\varepsilon, X) \subseteq A(X)\right)$ – множина обчислювальних алгоритмів (о.а.), для яких виконується умова (1), тобто алгоритмів обчислення ε-розв'язку за даних умов обчислень. О.а., для якого виконуються умови (1), (2) будемо називати T-ефективним, $A(\varepsilon, T_0)\left(A(\varepsilon, T_0) \subseteq A(\varepsilon, X)\right)$ – множина T-ефективних о.а.

В залежності від моделі обчислень та обмежень (1), (2) можна виділити такі випадки:

$$A(\varepsilon, X) \neq \varnothing,\ A(\varepsilon, T_0) \neq \varnothing, \tag{4}$$

$$A(\varepsilon, X) = \varnothing, \tag{5}$$

$$A(\varepsilon, X) \neq \varnothing,\ A(\varepsilon, T_0) = \varnothing. \tag{6}$$

Тобто, побудова обчислювального процесу при заданих умовах обчислень пов'язана з розв'язанням таких проблем:

- існування ε-розв'язку (можливість побудови ε-розв'язку за даних умов обчислень);
- існування T-ефективних о.а.;
- можливість побудови реального о.п. за даних умов обчислень.

Зауважимо, що у випадку (4) ε-розв'язок та T-ефективний о.а. існують, але питання побудови реального о.п. за даних умов обчислень може бути відкритим. Це пов'язано з тим, що параметри реального о.п., як правило, оцінюються на основі оцінок похибки наближеного розв'язку. Тобто доводиться користуватися наближеними значеннями параметрів, якість такого наближення визначається точністю оцінок похибки. При використанні мажорантних оцінок відхилення можуть бути значними. Можливості, що можуть бути використані для переходу від випадків (5) і (6) до випадку (4), пов'язані: із вихідними даними (використання апріорної інформації для звуження класу задач, вибір набору функціоналів $I_n(f) \in \tilde{I}$ ($\tilde{I}$ – клас інформаційних операторів) і розмірності n, перехід до нового класу вихідних даних, уточнення інформації $I_n(f)$); з алгоритмами (використання оптимальних за точністю та близьких до них алгоритмів і алгоритмів, орієнтованих на даний тип моделі обчислень); із моделлю обчислювальної системи (вибір архітектури та параметрів з урахуванням специфіки класу задач); із коригуванням обмежень ε, T_0.

Оптимізація математичного забезпечення розв'язання прикладних задач і прогрес обчислювальної техніки складають порівнянні внески у розширення можливостей розв'язування складних задач, зокрема, в зменшення обчислювальної складності.

Розглянемо вплив похибки заокруглення чисел на обчислювальну складність. Гіпотеза [1] про те, що для отримання розв'язку з точністю $O(\varepsilon)$ достатньо обчислювати значення функції f і виконувати проміжні обчислення при реалізації о.а. з $O\left(\ln \varepsilon^{-1}\right)$ двійковими розрядами, знайшла підтвердження при розв'язуванні багатьох задач. Отже, якщо при побудові ε-розв'язку програма використовує числові масиви загальним обсягом N чисел, то для їх зберігання потрібна пам'ять

$O\left(N\ln\varepsilon^{-1}\right)$. Далі, на прикладі окремих класів задач розглянемо, як може впливати похибка заокруглення на можливість обчислення ε-розв'язку і складність о.п..

Нехай

$$\rho(E_{и}(I_n(f),a,c)\le\varepsilon_1<\varepsilon, \tag{7}$$
$$\rho(E_{\mu\tau}(I_n(f),a,c)>\varepsilon_2,\ \varepsilon_2=\varepsilon-\varepsilon_1,$$

де $E_{\mu\tau}=E_{\mu}+E_{\tau}$, $a\in A$ і виконуються співвідношення (зокрема, при чисельному інтегруванні звичайних диференціальних рівнянь, обчисленні інтегралів і інших класах задач) [2–6]

$$E_{\mu}=O(n^{-p})\ \ E_{\tau}=\mathrm{O}(n\cdot 2^{-\tau}), \tag{8}$$

де p – порядок точності чисельного методу, τ – довжина мантиси у двійковому записі чисел у режимі плаваючої коми. При цьому

$$\varepsilon^0_{\mu\tau}(\tau)=\min_n\rho\left(E_{\mu\tau}\left(I_n(f)\right),a,c\right)=O\left(n_0^{-p}(\tau)\right), \tag{9}$$

де $n_0(\tau)=\mathrm{O}\left(2^{\tau/(p+1)}\right)$, а $E_{\mu}\left(n_0(\tau)\right)=O\left(n_0^{-p}(\tau)\right)$, $E_{\tau}\left(n_0(\tau)\right)=O\left(n_0^{-p}(\tau)\right)$.

При $n<<n_0$ домінує похибка методу. Вона може бути зменшена шляхом використання оптимальних наборів I_n, збільшення n (з урахуванням (9)), використання оптимальних за точністю о.а. і близьких до них, переходу до іншого класу вихідних даних I_n (з метою підвищити порядок точності) і відповідних йому о.а.

При $n>>n_0$ домінує похибка заокруглень. Зменшення величини $\rho(E_{\mu\tau})$ може бути досягнуто безпосереднім зменшенням n (з урахуванням (9)) або використанням тих же можливостей, що і при $n<<n_0$ (крім збільшення n), а також збільшенням τ. Із співвідношень (8) і (9) випливає, що виконання обмеження $\varepsilon^0_{\mu\tau}\le\varepsilon_2$ пов'язано з умовами:

$$n=O\left(\varepsilon_2^{-1/p}\right),\ \ \tau=O\left(\log\varepsilon_2^{-1}\right),\ \ \varepsilon_2\to 0. \tag{10}$$

Розглянемо випадок (6). Нехай ε-розв'язок обчислюється на одному процесорі за допомогою лінійного о.а. за час

$$T(\varepsilon)=T_I(\varepsilon)+T_a(\varepsilon), \tag{11}$$

де $T_I(\varepsilon)$ – процесорний час обчислення набору функціоналів $I_n(f)$ (інформаційна складність), $T_a(\varepsilon)$ – процесорний час реалізації о.а. при заданій інформації $I_n(f)$ (комбінаторна складність). При цьому

$$T_I(\varepsilon)=n(\varepsilon)\beta_f(\varepsilon)\alpha(\varepsilon),$$
$$T_a(\varepsilon)=n(\varepsilon)\beta_a\alpha(\varepsilon),$$

де $\alpha(\varepsilon)$ – час виконання "середньої" операції при обчисленні ε-розв'язку, $\beta_f(\varepsilon)$ – середня кількість операцій обчислення функціонала i_j, β_a – середня кількість операцій, що пов'язана з використанням одного функціонала i_j при реалізації о.а..

Зауважимо, що $\beta_f(\varepsilon)$ не залежить від ε, якщо функціонали в $I_n(f)$ можуть бути обчислені точно при точних арифметичних операціях. Тоді

$$T(\varepsilon)=n(\varepsilon)\cdot\beta\cdot\alpha(\varepsilon),\quad \beta=\beta_f+\beta_a. \tag{12}$$

Якщо $\alpha(\varepsilon)$ відповідає операції множення двох чисел, то з огляду на (10) і того, що при розпаралелюванні операцій множення [7] $\alpha(\tau) = \mathrm{O}(\tau)$, отримаємо

$$T(\varepsilon_2) = O(\varepsilon_2^{-1/p} \log \varepsilon_2^{-1}), \varepsilon_2 \to 0. \tag{13}$$

Як видно з (13), випадок (6) може бути наслідком неточності вихідної інформації: при $\varepsilon_2 << \varepsilon_1$, (див. (7)) умова $\rho(E_{\mu\tau}) \le \varepsilon_2$ ставить жорсткі вимоги до n, τ і, отже, до процесорного часу. Для забезпечення умови $A(\varepsilon, T_0) \neq \varnothing$, можливо, достатньо взяти вихідні дані з більш високою точністю: $\varepsilon_2 \approx \varepsilon_1$.

Залежність (13) показує, що в аналізованому випадку оптимальні за порядком за точністю о.а. є також оптимальними за порядком за швидкодією. Можливість збільшення оптимального порядку (p) пов'язана з переходом до деякого нового класу вихідних даних ($\tilde{I}$). Зменшення константи у співвідношенні $n = \mathrm{O}\left(\varepsilon^{-1/p}\right)$ досягається вибором оптимального набору $I_n(f)$ за певних припущень щодо затрат на такий вибір. Наприклад, при чисельному інтегруванні з використанням набору значень підінтегральної функції це буде оптимальний розподіл вузлів, задіяних в $I_n(f)$. Значення β для даної задачі залежить від способу обчислення функціоналів у $I_n(f)$, о.а., архітектури комп'ютера. Алгоритми обчислення функціоналів i_j можуть будуватися з урахуванням складності машинних операцій з метою зменшення $t_f(\varepsilon)$. Це зауваження стосується і величини $\alpha(\varepsilon)$, яка може бути зменшена також вибором параметрів моделі обчислень (наприклад, керуванням величиною τ при реалізації ітераційних процесів).

2. Про моделі високопродуктивних обчислень

Постійно зростає потік математичних моделей, для чисельної реалізації яких недостатня продуктивність (швидкодія) традиційних однопроцесорних комп'ютерів. Джерелом таких задач є дослідження в ядерній фізиці, екології (прогнозування і керування системами), моделюванні клімату, довгостроковому прогнозі погоди, криптографії та інші.

Розглянемо можливості використання принципів паралельної обробки даних, квантової механіки для організації високопродуктивних обчислень.

2.1. Паралельні обчислення

Висока продуктивність сучасних обчислювальних систем досягнута у значній мірі завдяки впровадженню процесів паралельної обробки даних. Важливими задачами організації швидких паралельних обчислень є розробка паралельних алгоритмів, які відображають паралелізм задачі, узгодження таких алгоритмів з можливостями паралельних комп'ютерів.

Для попереднього аналізу обчислювальної складності паралельних алгоритмів може бути використана ідеалізована модель паралельних обчислень, у якій k ідентичних процесорів мають необмежену пам'ять, що доступна кожному з них без конфліктів. Кожен процесор за одиничний інтервал часу (крок обчислень) може точно виконати одну із бінарних арифметичних операцій. Часом виконання інших операцій можна знехтувати. k операцій, реалізованих паралельно, виконуються за один крок обчислень. Час розв'язування задачі дорівнює числу паралельних кроків обчислень.

Паралельний алгоритм, реалізований на k процесорах будемо називати k-алгоритмом.

Як міра паралелізму k-алгоритму використовується прискорення (паралельного алгоритму) $S_k = T_1 / T_k$, де T_k – час обчислень при розв'язуванні задачі k-алгоритмом, T_1 – мінімальний час

розв'язування задачі на одному процесорі. В більшості випадків відомі лише оцінки T_1, які можуть відноситися до послідовного алгоритму, що розпаралелюється.

Зважаючи на те, що в паралельних алгоритмах обсяг обчислень може бути більший ніж в (кращому) послідовному, не всі k процесорів використовуються в k-алгоритмі на кожному кроці обчислень, скористаємося співвідношенням $T_k = \nu_k T_1 \varphi(k)$, де $\nu_k \geq 1$ – величина, що характеризує можливе збільшення обсягу обчислень при переході від послідовного до паралельного алгоритму, $\varphi(k) = \sum_1^k \alpha_i / i$, $\sum_1^k \alpha_i = 1$, $\alpha_i \geq 0$, α_i – частка обсягу обчислень, що виконується одночасно i-процесорами при реалізації k-алгоритму.

Нехай $T_{k1} = \nu_k T_1 / k$ – час обчислень при рівномірному розподілі обсягу обчислень між процесорами. Тоді $T_k = T_{k1} + T_{k2}$, де $T_{k2} = \nu_k T_1(\varphi(k) - 1/k)$ – втрати часу за рахунок простоїв деяких процесорів. Формула для прискорення може бути записана у вигляді

$$S_k = \frac{k}{\nu_k(1+\delta_k)},\ \delta_k = T_{k2} / T_{k1}, \tag{14}$$

тобто, максимальне прискорення ($S_k = k$) досягається, коли перехід від послідовного до паралельного алгоритму відбувається без збільшення обсягу обчислень ($\nu_k = 1$) при рівномірному розподілі обчислень між процесорами ($\alpha_k = 1$).

Наведемо приклади використання цієї моделі для побудови та аналізу складності k-алгоритмів.

Нехай нелінійна неперервна функція $f(x)$ має в інтервалі [0,1] один нуль і приймає на кінцях відрізка значення різних знаків. Для локалізації нуля в інтервалі довжини ε використовується послідовний метод бісекції. При використанні k-алгоритму на кожній ітерації одночасно обчислюються значення функції у k рівновіддалених точках. Тоді $S_k = \log_2(k+1)$ [8]. Тобто виграш часу суттєво відрізняється від максимального. В цьому випадку простоїв немає, але обсяг обчислень великий порівняно з послідовним алгоритмом: $\nu_k = k / \log_2(k+1)$.

При розпаралелюванні обчислення скалярного добутку k-вимірних векторів (нехай $k = 2^m$): $\nu_k = 1,\ T_k = m+1,\ T_{k2} = m-1,\ S_k = (2k-1)/(m+1)$.

Рекурентні схеми обчислень часто використовуються при розв'язанні різних класів задач (лінійної алгебри, звичайних диференціальних рівнянь тощо). Обчислення за такими схемами $x_n = \psi(x_{n-1}, ..., x_{n-q})$, де $x_{-q}, \ldots, x_0$ – задані, $n = \overline{1, N}$, суто послідовні і для обчислення x_N потрібно $O(N)$ кроків незалежно від кількості процесорів. Якщо рекурентна схема лінійна, то вона може бути перетворена до такої форми, яка легко розпаралелюється за алгоритмом рекурентного здвоювання. Зокрема, при обчисленні x_N за паралельним алгоритмом на основі скалярної формули $x_n = a_n x_{n-1} + b_n$, маємо $S_k(x_N) = \frac{2k+1}{3+\delta}$, $\delta = O\left(\frac{k \log_2 k}{N}\right)$. В даному разі прискорення обмежується величиною $\nu_k \approx 3/2$ і простоями.

У загальному випадку нелінійної функції ψ не вдається прискорити обчислення порівняно з $O(N)$ шляхом розпаралелювання обчислень [9].

Рекурентні схеми лежать в основі методів чисельного інтегрування задачі Коші для звичайних диференціальних рівнянь [2, 10]. Якщо система рівнянь нелінійна, то розпаралелювання обчислень

на основі традиційних (послідовних) чисельних методів проводиться у більшості випадків на кроці інтегрування. При цьому використовується паралелізм системи рівнянь: i-й процесор обчислює m_i компонент розв'язку ($m_1+...+m_k=m$ – розмірність системи). З ростом m і арифметичної складності правих частин покращуються можливості рівномірного розподілу обчислень між процесорами і досягається $S_k \approx k$.

При використанні блочних схем інтегрування для організації паралельних обчислень може бути використаний паралелізм чисельної схеми: одночасно обчислюються ітерації для кожного з вузлів блоку. При цьому спрощується задача рівномірного розподілу обчислень між процесорами, але кількість паралельних гілок обмежується розміром блоку.

У випадку системи лінійних звичайних диференціальних рівнянь може бути використаний паралелізм задачі: вона зводиться до послідовності задач меншої складності, із розв'язків яких шляхом рекурентного здвоювання можна обчислити розв'язок основної задачі. Цей шлях, як правило, пов'язаний з великими ν_k.Тому доцільність його використання вирішується конкретною обчислювальною ситуацією.

Чи завжди придатні такі алгоритми (побудовані в умовах ідеалізованої моделі обчислень) для реальних обчислень? Не завжди. Паралельний алгоритм може бути чисельно нестійким. Це, наприклад, паралельні алгоритми обчислення арифметичних виразів [9] та паралельні алгоритми здвоювання. При розв'язуванні задач лінійної алгебри з розпаралелюванням обчислень чисельна стійкість алгоритмів може істотно погіршуватись при значному зростанні числа процесорів [11]. Разом з тим паралельний алгоритм метода пристрілки розв'язування крайових задач для звичайних диференціальних рівнянь має кращу чисельну стійкість ніж послідовний.

Паралельні комп'ютери мають обмежену кількість процесорів. Багато які з паралельних алгоритмів ефективні тільки при великій кількості процесорів. Це стосується методів, які не використовувались для послідовних обчислень, а побудовані спеціально для паралельних обчислень [9].

Затрати на обмін даними можуть бути обмежуючим фактором у ефективному використанні паралельного алгоритму [9, 11].

В конкретних паралельних комп'ютерах реалізовані певні форми паралельної обробки даних, тому для ефективних паралельних обчислень функціональні зв'язки алгоритму мають бути узгоджені з комутаційними схемами комп'ютера.

Для оцінки прискорення паралельного алгоритму можна скористатись формулою (14) за умов, що при обчисленні на однопроцесорному комп'ютері використовується пам'ять з такими ж обсягом і швидкодією, як і на багатопроцесорній системі, а $\delta_k = (T_{k2} + T_{k3}) / T_{k1}$, де T_{k3} – час підготовки даних (синхронізації, обміну даними між процесорами).

Як видно з наведеного, для отримання прискорення близького до максимального, має бути досить великим обсяг обчислень на одиницю затрат при підготовці даних, досить малі простої і надлишок обчислень.

Значення величин ν_k, α_k, T_{k2} при конкретних умовах обчислень залежать від паралелізму задачі, о.а., вихідних даних, моделі обчислень.

Збільшення величини α_k може досягатися вибором моделі обчислень, яка краще відповідає паралелізму задачі, і використанням відповідних о.а., а при заданій моделі обчислень максимізація величини α_k може досягатись вибором о.а. і зведенням даної задачі до іншої (паралелізм якої узгоджується з архітектурою комп'ютера і тому має меншу обчислювальну складність).

Досягненню максимального прискорення може перешкоджати існування у обчисленнях послідовних розрахунків, які не можуть бути розпаралелені. Нехай f є частка послідовних обчислень у

алгоритмі обробки даних, що застосовується, тоді прискорення відповідно до третього закону Амдаля при використанні p процесорів обмежується величиною $S_p \le \dfrac{1}{f+(1-f)/p}$.

Згідно цього закону за наявності всього 10 % послідовних команд у обчисленнях ефект використання паралелізму не може перевищувати 10-кратного прискорення обробки даних.

Закон Амдаля характеризує одну з найсерйозніших проблем у галузі побудови паралельних алгоритмів (алгоритмів без певної частки послідовних обчислень практично не існує). Такі алгоритми ще називають поперемінно послідовно паралельні алгоритми (ПТМ алгоритми).

Слід також зазначити, що закон Амдаля розглядається в припущенні, що частка послідовних розрахунків f є постійною величиною і не залежить від параметра n, що визначає обчислювальну складність розв'язуваної задачі. Однак для великого класу задач величина $f = f(n)$ є спадною функцією від n (зокрема, для задач тріангуляції та тридіагоналізації у лінійній алгебрі), і в цьому випадку прискорення для фіксованого числа процесорів може бути збільшено за рахунок збільшення обчислювальної складності задачі, що розв'язується.

Ще одна важлива характеристика паралельних алгоритмів – їх масштабованість, здатність ефективно використовувати процесори у разі підвищення складності обчислень. Паралельний алгоритм називають масштабованим, якщо за зростання кількості процесорів він забезпечує збільшення прискорення.

Можливий спосіб характеристики властивостей масштабованості полягає у наступному. Оцінимо накладні витрати, що мають місце під час виконання паралельного алгоритму $T_0 = pT_p - T_1$.

Накладні витрати виникають внаслідок необхідності організації взаємодії процесорів, виконання деяких додаткових дій, синхронізації паралельних обчислень тощо.

Використовуючи введені позначення можна отримати нові оцінки для часу паралельного розв'язання задач та прискорення:

$$T_p = \frac{T_1 + T_0}{p}, \quad S_p = \frac{T_1}{T_p} = \frac{pT_1}{T_1 + T_0}.$$

Застосовуючи отримані співвідношення, ефективність використання процесорів паралельного алгоритму можна представити так:

$$E_p = \frac{S_p}{p} = \frac{T_1}{T + T_0} = \frac{1}{1 + T_0 / T_1}.$$

Останній вираз показує, що якщо складність розв'язуваної задачі – фіксована ($T_1 = const$), то при зростанні числа процесорів ефективність зменшуватиметься, як правило, за рахунок зростання накладних витрат T_0. При фіксації числа процесорів ефективність їх використання можна покращити шляхом підвищення складності розв'язуваної задачі T_1. Оцінимо максимально допустиме прискорення, виходячи з наявної частки послідовних обчислень у паралельних алгоритмах. Позначимо

$$g(n) = \frac{\tau(n) * 69}{\tau(n) + \eta(n)/p},$$

де $\tau(n), \eta(n)$ – часи послідовної та паралельної частин обчислень відповідно, тобто

$$T_1 = \tau(n) + \eta(n), \quad T_p = \tau(n) + \eta(n)/p.$$

З урахуванням введених позначень можна отримати

$$\tau(n) = g(\tau(n) + \eta(n) / p), \quad \eta(n) = (1-g)p(\tau(n) + \eta(n) / p),$$

що дозволяє отримати оцінку для прискорення

$$S_p = \frac{T_1}{T_2} = \frac{\tau(n) + \eta(n)}{\tau(n) + \eta(n) / p} = \frac{\left(\tau(n) + \eta(n) / p\left(g + (1-g)p\right)\right)}{\tau(n) + \eta(n) / p},$$

яка після спрощення зводиться до вигляду Густавсона – Барсіса

$$S_p = g + (1-g)p = p + (1-p)g.$$

Для прикладу в задачі підсумовування значень при використанні p процесорів час паралельного виконання становить

$$T_p = n / p + \log_2 p,$$

що відповідає послідовній частці

$$g = \frac{\log_2 p}{(n / p) + \log_2 p}.$$

За рахунок збільшення числа сумованих значень величина g може стати достатньо малою, забезпечуючи отримання ідеального можливого прискорення $S_p = p$.

Ще одна характеристика ефективних паралельних обчислень – збалансованість обчислень.

Так, наприклад, застосування послідовної схеми розподілу даних для паралельного розв'язування систем лінійних алгебраїчних рівнянь $Ax = b$ призведе до нерівномірного обчислювального навантаження процесорів: у міру виключення (на прямому ході) невідомих у методі Гаусса для все більшої частини процесорів обчислення будуть завершені і процесори будуть простоювати (так званий ефект Гайдна). Вирішення проблеми балансування обчислень полягає у використанні циклічної схеми для розподілу даних між процесорами та їх обробки. Рядково-циклічна схема вперше була запропонована в [12]. В цьому випадку матриця A поділяється на блоки рядків вигляду:

$$A=(A^*_1,...,A^*_p)^T, \; A^*_i = (a_{kl}), \; k = i + mp, \; 0 \le m < p, \; l = 1, n.$$

Це означає, що в A^*_i знаходяться рядки з номерами: $i, i+p, i+2p, ..., i+(p-1)p$. В такому випадку в i-му процесорі обробляються рядки з A^*_i. За рахунок цього вдалося асимптотично для $n \gg p$ досягти $S_p = p$.

Надалі для балансування процесорів, найбільше застосування отримав клітковий блочно-циклічний алгоритм, обробки даних, який до того ж забезпечує кешизацію обчислень. Популярності цієї схеми сприяло використання рекурентного представлення блочного алгоритму для методів на основі триангуляції матриць. Зокрема, для методу Гаусса, на k-ому блочному кроці $k = (1, 2, ..., l)$ алгоритм матиме наступний вигляд [5]:

$$A_{11} = L_{11}U_{11}; \; U_{12} = (L_{11})^{-1} A_{12}; \; L_{21} = A_{21}(U_{11})^{-1}; \; \tilde{A}_{22} = A_{22} - L_{21}U_{12}, \tag{15}$$

де матричні блоки відповідають наступній факторизації матриці:

$$A^{(k)} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = P \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix} = P \begin{pmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} \end{pmatrix},$$

де підматриця $A^{(k)}$ порядку $r = n-(k-1)s$ містить останні r рядків і r стовпчиків матриці A, A_{11} – блок, який має розмір $s \times s$, A_{12} – блок, який має розмір $s \times (r-s)$, A_{21} – блок, який має розмір $(r-s) \times s$, прямокутна підматриця, що містить блоки, розташовані нижче провідного, а A_{22} – блок, який має розмір $(r-s) \times (r-s)$. P – матриця перестановок у випадку вибору головного елементу, s – розмір блоку, $l = n/s$.

Представлення (15) дозволяє ефективно реалізувати блочно-циклічний процес обробки матриць, кешизацію обчислень та використати для розробки програмного забезпечення стандартні бібліотеки програм [6].

2.2. Квантові моделі обчислень

За останні роки значно зріс інтерес до квантових комп'ютерів [13–17]. Ідея використання можливостей квантової механіки при організації обчислень стає все більш привабливішою, експерименти у цій галузі проводяться в США, Китаї, Німеччині та інших країнах світу. Створені перші квантові комп'ютери, які вже виходять за рамки лабораторій.

Безумовно, сучасні суперкомп'ютери працюють швидше старих, але прогрес у цьому напрямку має межу. Так що і суперкомп'ютери майбутнього не зможуть розв'язувати обчислювальні задачі, які мають експоненціальну складність.

Розглянемо, наприклад, задачу про розкладання цілого числа x на прості множники. Очевидний спосіб – це спробувати поділити x на числа від 2 до $\sqrt{x}$. Якщо число x має n знаків у двійковому запису, то прийдеться перебрати $\approx \sqrt{x} \approx 2^{n/2}$ варіантів. Час роботи сучасних алгоритмів факторизації асимптотично може бути оцінений виразом $\exp\left(2{,}08 \cdot n^{1/3} \cdot (\log n)^{2/3}\right)$. Навіть у цьому випадку, щоб розкласти на множники число з мільйону знаків, не вистачить часу життя Всесвіту.

Однак, існують інші моделі обчислень, які дозволяють прискорити процес обчислень для деяких спеціальних класів задач. Справа у тому, що звичайні комп'ютери не використовують усіх можливостей, які надає природа: в природі є багато процесів, зовсім не схожих на операції з нулями та одиницями. Можна спробувати використати ці операції для створення аналогової обчислювальної машини. Наприклад, інтерференція світла може використовуватись для обчислення перетворення Фур'є.

Однак у більшості випадків виграш у швидкості не є принциповим. Причина полягає у тому, що рівняння класичної фізики ефективно розв'язуються на звичайному цифровому комп'ютері. Таким чином, класична фізика досить "проста" з обчислювальної точки зору.

Квантова механіка в цьому сенсі більш цікава. Розглянемо, наприклад, систему із n спінів, кожний спін має два базисні стани (0 = "спін вгору" і 1 = "спін вниз"), а вся система має 2^n базисних станів $|x_1,...,x_n\rangle$ (кожна із змінних $x_1,...,x_n$ приймає значення 0 або 1). Згідно до загальних принципів квантової механіки, можливими станами системи є також суперпозиції вигляду $\sum_{x_1,...,x_n} C_{x_1,...,x_n} |x_1,...,x_n\rangle$, де $C_{x_1,...,x_n}$ – комплексні числа, які звуться амплітудами. Суперпозиція це новий математичний об'єкт – вектор у 2^n-вимірному комплексному просторі. Квадрат модуля амплітуди $\left|C_{x_1,...,x_n}\right|^2$ дорівнює ймовірності виявити систему в базисному стані $|x_1,...,x_n\rangle$ при вимірюванні значень змінних x. (Зазначимо, що таке вимірювання руйнує суперпозицію). Отже, має

виконуватись умова $\sum_{x_1,...,x_n} \left|C_{x_1,...,x_n}\right|^2 = 1$. Таким чином, загальний стан системи (тобто суперпозиція) – це вектор одиничної довжини у 2^n-вимірному комплексному просторі. Зміна стану за певний проміжок часу описується унітарною матрицею розміру $2^n \times 2^n$. Якщо проміжок часу дуже малий ($<< h/j$, де j – енергія взаємодії), то ця матриця досить проста, кожен з її елементів можна легко обчислити, якщо ми знаємо взаємодію між спінами. Якщо ж ми хочемо пізнати зміну стану системи за великий проміжок часу, то доведеться перемножувати такі матриці. Для цього треба виконати експоненційно велику кількість операцій. На сьогодні невідомо ніякого способу спростити це обчислення і, скоріш за все, моделювання квантової механіки – експоненційна складна обчислювальна задача. Можливо це саме твердження сформулювати таким чином: квантова система ефективно розв'язує складну обчислювальну задачу – моделює себе саму.

Чи можна використовувати квантові системи для розв'язування інших обчислювальних задач? Яка має бути математична модель квантового комп'ютера, в тій же мірі не залежна від фізичної реалізації, як і методи класичних обчислень? У 1985 році Д. Дойч [13] запропонував конкретну математичну модель – квантову машину Тьюрінга, а у 1989 році – еквівалентну, але більш зручну модель – квантові схеми [13].

Що таке квантова схема? Нехай у нашому розпорядженні є N спінів, кожний з яких знаходиться в окремій шухляді та ідеально ізольований від зовнішнього світу. В кожний момент часу ми можемо вибрати, за нашим розсудом, будь-які два спіни і подіяти на них довільною унітарною матрицею 4×4. Послідовність таких операцій називається квантова схема. Кожна операція визначається парою номерів спінів та шістнадцятьма комплексними числами, тому квантову схему можна записати на папері.

Для того, щоб використати квантову схему для обчислення функції $F: B^n \to B^m$, треба вміти вводити вихідні дані, зробити обчислення та зчитати результат. Ввести у квантовий комп'ютер послідовність $(x_1,...,x_n)$ нулів та одиниць – значить приготувати початковий стан $\left|x_1,...,x_n,0...,0\right\rangle$. (Обсяг вихідних даних n значно менший за загальну кількість "комірок пам'яті", тобто спінів, N. Комірки, які залишаються, заповнюються нулями). До початкового стану застосовується квантова схема, яка залежить від задачі, але не від конкретних вхідних даних. В результаті виникає квантовий стан

$$\left|\psi(x_1,...,x_n)\right\rangle = \sum_{y_1,...,y_N} C_{y_1,...,y_N}(x_1,...,x_n)\left|y_1,...,y_N\right\rangle, —\circ$$

який залежить від $(x_1,...,x_n)$. Тепер потрібно зчитати результат. Припустимо, що відповідь має міститись у перших m бітах рядка $(y_1,...,y_N)$, тобто ми шукаємо такі $(y_1,...,y_m)$, що $(y_1,...,y_m) = F(x_1,...,x_n)$. Для отримання відповіді робиться вимір значень усіх спінів. Результатом виміру може бути будь-яка послідовність нулів та одиниць $(y_1,...,y_N)$. Ймовірність отримати її дорівнює $\left|C_{y_1,...,y_N}(x_1,...,x_n)\right|^2$. Таким чином, квантовий комп'ютер може, з деякою ймовірністю, дати якусь відповідь. Квантова схема є "правильною" для деякої функції F, якщо правильна відповідь $(y_1,...,y_m) = F(x_1,...,x_n)$ отримається з ймовірністю, яка досить близька до одиниці. Якщо повторити ці обчислення декілька разів і вибрати ту відповідь, яка зустрічається частіше, можна зменшити ймовірність похибки до скільки завгодно малої величини [13].

Зауважимо, що обчислювальна задача може бути представлена як задача обчислення функції. Наприклад, якщо ми хочемо розв'язати задачу про розкладання цілого числа X на прості множники, то $(x_1,...,x_n)=x$ (у двійковому запису), а $F(x)$ – список простих множників (при деякому двійковому кодуванні).

Нещодавно було розроблено алгоритм факторизації числа на квантовому комп'ютері, час роботи якого оцінюється виразом $O(n^{2+\varepsilon})$ де ε – деяке мале число [18]. Це означає, що криптосистеми з відкритими ключами, криптостійкість яких ґрунтується на складності задачі факторизації, можуть бути зламані.

Другий приклад ґрунтується на тій же роботі П. Шора [18], який показав розв'язність на квантовому комп'ютері задачі знаходження дискретного логарифму числа в мультиплікативних цілочисельних групах і задачі розкладання цілого числа на множники за поліноміальний (щодо довжини) час. Хоча і не доведено, що ці задачі належать до NP-повних, про їх складність говорить відсутність ефективного алгоритму їх розв'язку, на чому ґрунтуються деякі сучасні криптографічні алгоритми. Третій приклад – це пошук необхідного запису у невпорядкованій базі даних. Тут виграш не такий великий: для знаходження одного запису з N потрібно близько $\sqrt{N}$ операцій на квантовому комп'ютері замість N на класичному.

Д. Бонех і Р. Ліптон [19], використовуючи методику П. Шора, побудували квантові алгоритми для розв'язання задачі розкриття "неявної лінійної форми" і періоду довільної функції на абелевих групах (задача обчислення дискретного логарифма в цілочисельних групах Z_p – окремий випадок цих задач). Важливість цього результату доповнюється тим, що квантовий розв'язок задачі обчислення дискретного логарифма автоматично розширюється на всі абелеві групи, у тому числі на поля Галуа й еліптичні криві (які широко застосовуються у криптографії).

Відомі також інші застосування. Прослуховування інформації, яка передається квантовим каналом зв'язку, неможливе без внесення у повідомлення спотворень, які можуть бути виявлені користувачами каналу. Використання цього ефекту у квантовій криптографії [20] дає можливість двом абонентам, які раніше не обмінювались ніякою секретною інформацією, абсолютно таємно обмінюватись інформацією "під носом" у противника, який хоче її перехопити. Квантові методи дають також можливість розв'язати ряд специфічних задач: за їх допомогою взаємно не довіряючи одна одній сторони можуть прийняти спільне рішення на основі конфіденційної інформації, майже не розкриваючи одна одній її таємниці.

Виникають два питання.

1. Для яких задач квантове обчислення дає виграш у порівнянні з класичним (квантова перевага)?;

2. Яку систему можна використати для фізичної реалізації квантового комп'ютера? (Це не обов'язково має бути система спінів).

З приводу першого питання додамо наступне:

- моделювання довільної квантової системи за поліноміальну кількість кроків;
- криптоаналіз;
- квантові генератори випадкових послідовностей;
- прогноз властивостей молекул та кристалів;
- проектування мікроскопічних електронних пристроїв;
- ефективна реалізація перетворень Фур'є;

– оптимізаційні задачі (частково);
– розробка нових лікарняних засобів;
– розшифрування генома;
– дослідження космічного простору;
– метеорологія;
– дослідження у галузі ядерної енергетики;
– штучний інтелект;
– використання блокчейн технологій у квантових системах для зберігання інформації та керуванні роботою таких систем тощо;
– квантове машинне навчання.

Стосовно другого питання. Квантові обчислення ґрунтуються на побудові траєкторії від стандартного початкового стану до складного кінцевого стану. Найбільша проблема – це надчутливість до збурень, які порушують траєкторію випадковим чином. Такі збурення, відбуваються завдяки неконтрольованим зв'язкам з зовнішніми шумами.

Попри це вже створені перші квантові комп'ютери, які вийшли за межі лабораторних досліджень (США, Китай, Німеччина, Японія, Росія, Канада) наведені у таблиці.

ТАБЛИЦЯ

Рік	Фірма, країна	Назва КК	Кількість кубітів
2016	IBM	"Quantum Experience"	5
2019	IBM	"Falcon"	27
2019	Google	"Sycamore"	53
2020	США	Hummingbird	65
2021	IBM для Німеччини	"Quantum System One"	27
2021	Канада для Німеччини	"Юліха"	5000
2021	Росія	"Штріх-М-Казань"	51
2021	IBM	"Eagle"	127
2021	Японія	"Фугаку"	127
2021	Китай	"Zuchongzhi"	56
2022	IBM	"Osprey"	433
2023*	IBM	"Condor"	1121

*Condor близький до так званої квантової переваги – точці, в якій квантовий комп'ютер зможе перебільшити класичні комп'ютери.

В найближчі три роки будуть створені прототипи квантових комп'ютерів, які подолають рубіж у тисячу кубітів.

Висновки. Продемонстрований зв'язок оцінок складності обчислювальних алгоритмів з архітектурою комп'ютерів та моделями обчислень.

Наведено характеристики перших квантових комп'ютерів (2016 – 2022 рр.), які вийшли за межі лабораторних досліджень.

Список літератури

1. Бахвалов Н.С. О свойствах оптимальных методов решения задач математической физики. *Журнал. Вычислительная математика и математическая физика*. 1970. **10** (3). С. 555–568.
2. Иванов В.В. Методы вычислений на ЭВМ: Справочное пособие. Киев: Наук. думка, 1986. 584 с.
3. Иванов В.В., Бабич М.Д., Березовский А.И., Бесараб П.Н., Задирака В.К., Людвиченко В.А. Характеристики задач, алгоритмов и ЭВМ в комплексах программ вычислительной математики. Киев, 1984. 53 с. (Препринт/АН УССр, Ин-т кибернетики; 84–36).
4. Бесараб П.Н. О сложности интегрирования обыкновенных дифференциальных уравнений. *Кибернетика и системный анализ*. 1996. **3**. С. 122–130.
5. Химич А.Н., Молчанов И.Н., Попов А.В., Чистякова Т.В., Яковлев М.Ф. Параллельные алгоритмы решения задач вычислительной математики. Киев: Наук. думка, 2008. 247 с.
6. NetLib. http://www.netlib.org/ (звернення: 23.09.2022)
7. Карцев М.А. Арифметика цифровых машин. М.: Наука, 1969. 575 с.
8. Системы параллельной обработки / Под ред. Д. Ивенса. М.: Мир, 1985. 416 с.
9. Алгоритмы, математическое обеспечение и архитектура многопроцессорных систем / Под ред. В.Б. Котова и Й. Миклошко. М.: Наука, 1982. 336 с.
10. Хайрер Э., Нерсетг С., Ваннер Т. Решение обыкновенных дифференциальных уравнений. М.: Мир, 1990. 512 с.
11. Воеводин В.В. Математические модели и методы в параллельных процессах. М.: Наука, 1986. 296 с.
12. Михалевич В.С., Молчанов И.Н., Сергиенко И.В. и др. Численные методы для многопроцессорного вычислительного комплекса ЕС. М.: ВВИА им. проф. Н.Е. Жуковского, 1986. 401 с.
13. Китаев А., Шень А., Вялый М. Классические и квантовые вычисления. М.: Мир, 1988. 518 с.
14. Задірака В.К., Бабич М.Д., Березовський А.І., Бесараб П.М., Гнатів Л.О., Людвиченко В.О. *Т*-ефективні алгоритми наближеного розв'язання задач обчислювальної та прикладної математики. Київ-Тернопіль, "Збруч", 2003. 261 с.
15. Sergienko I.V., Zadiraka V.K., Lytvyn O.M. Elements of the General Theory of Optimal Algorithms. Springer, 2021. P. 378. https://doi.org/10.1007/978-3-030-90908-6
16. Нильсен М., Чанг И. Квантовые вычисления и квантовая информация. М.: Мир. 2006. 824 с.
17. Каптьол Є.Ю., Горбенко І.Д. Аналіз можливостей та особливості програмування задач криптології на квантовому комп'ютері. *Радіотехніка*. 2020. Вип. 202. С. 37–48. https://doi.org/10.30837/rt.2020.3.202.03
18. Shor P. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. *Proc. of the 35th Annual Symposium on Foundations of Computer Science. IEEE Computer Society Press*. 1994. P. 124–134.
19. Boneh D., Lipton R.J. Quantum cryptanalysis of hidden linear function. *Lect. Notes Comput. Sci.* 963. 1995. P. 424–437. https://doi.org/10.1007/3-540-44750-4_34
20. Cleve R. A note of computing Fourier transforms by quantum programs. 1994.

Одержано 02.09.2022

Задірака Валерій Костянтинович,
доктор фізико-математичних наук, професор, академік НАН України,
завідуючий відділом Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
https://orcid.org/0000-0001-9628-0454
zvk140@ukr.net

Хіміч Олександр Миколайович,
доктор фізико-математичних наук, професор, академік НАН України,
заступник директора Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
https://orcid.org/0000-0002-8103-4223
khimich505@gmail.com

Швідченко Інна Віталіївна,
кандидат фізико-математичних наук, старший науковий співробітник,
провідний науковий співробітник Інституту кібернетики імені В.М. Глушкова НАН України, Київ.
https://orcid.org/0000-0002-5434-2845
inetsheva@gmail.com

UDC 519.6

Valerii Zadiraka [*], Oleksandr Khimich, Inna Shvidchenko [*]

Models of Computer Calculations

V.M. Hlushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv
[*] Correspondence: zvk140@ukr.net, inetsheva@gmail.com

Introduction. The complexity of computational algorithms for solving typical problems of computational, applied, and discrete mathematics is analyzed from the perspective of the theory of computation, depending on the computer architecture and the used computing model: single-processor, multiprocessor, and quantum.

The following classes of problems are considered: systems of linear algebraic equations, the Cauchy problem for systems of ordinary differential equations, numerical integration, boundary value problems for ordinary differential equations, factorization of numbers, finding the discrete logarithm of a number in multiplicative integer groups, searching for the necessary record in an unordered database, etc.

The purposes of the paper are:

1. To investigate how the computational complexity depends on the computer architecture and the computational model.

2. To show that the construction of the computational process under the given conditions of calculations is related to the solution of the following problems:

– the existence ε-solution to the problem;

– the existence of T-effective computing algorithms;

– the possibility of building a real computing process under the given computing conditions.

3. To investigate the effect of rounding numbers on computational complexity (especially when solving problems of transcomputational complexity).

4. To give the complexity estimates and total error of the computational algorithm for a number of typical problems of computational, applied, and discrete mathematics.

The results. The complexity estimates of computational algorithms of the listed classes of problems for single-processor, multiprocessor and quantum computing models are given.

The main focus is on high-performance computing: using the principles of parallel data processing and quantum mechanics.

Conclusions. The connection of complexity estimates of computational algorithms with the architecture of computers and models of calculations is demonstrated.

The characteristics of the first quantum computers (2016 – 2022), which have gone beyond laboratory research, are given.

Keywords: computer technologies, rounding error, sequential, parallel and quantum computing models, complexity estimate.