УДК 004.051

# ОСОБЛИВОСТІ ПРАКТИЧНОГО ЗАСТОСУВАННЯ ПОКАЗНИКІВ ОБЧИСЛЮВАЛЬНОЇ СКЛАДНОСТІ АЛГОРИТМІВ

*В.І. Шинкаренко*

Дніпропетровський національний університет залізничного транспорту
ім. академіка В. Лазаряна
49010, Дніпропетровськ, вул. академіка Лазаряна, 2,
e-mail: ccp@diit-70.dp.ua

Розглянута властивість обчислювальної складності алгоритмів. Уточнена термінологія. Розглянуті показники обчислювальної складності та методики їх визначення: класичні за Д. Кнутом та асимптотичні. Показані особливості інтерпретації цих показників. Виявлена можлива залежність показників обчислювальної складності алгоритмів від обчислювальних пристроїв (ЕОМ), як можливості збільшення так і зниження. Показані проблеми та особливості визначення показників обчислювальної складності при алгоритмічній реалізації наближених методів рішення задач. Особливості практичного застосування показників обчислювальної складності алгоритмів.

Property of computational complexity of algorithms is considered. Terminology is defined more accurately. Parameters of computational complexity and methods of their definition are considered: classical by D. Knuth and asymptotical. Peculiarity of interpretation of these parameters is shown. Possible dependence of parameters of computational complexity of algorithms from executive devices (computer) is revealed, opportunities as increases and decrease. Problems and peculiarities of definition computational complexity parameters for algorithmic realization of approximated calculations are shown. Peculiarity of practical application of computational complexity parameters for algorithms.

## Вступ

Для обгрунтованого застосування алгоритмів необхідно знати, а потім і вивчати їх властивості. Одна з категорій властивостей складається з таких як: кількість операцій та операндів, кількість унікальних операцій та операндів, обсяг програми (міри Холстеда) [1], цикломатична міра Мак-Кейба [2] та інші. Досить широкий огляд таких властивостей надано в [3, 4]. Ця категорія властивостей вирізняється тим, що вони можуть бути визначені за представленням алгоритмів та не потребують їх виконання. Самі властивості зазвичай визначаються через відповідні показники.

Інша категорія властивостей алгоритмів – це властивості, які проявляються лише при виконанні алгоритмів або у багаторазовому виконанні. Такі властивості будемо вважати експлуатаційними характеристиками. До них віднесемо часову та функціональну ефективність алгоритмів [5, 6].

Однією із найважливіших властивостей алгоритмів є обчислювальна складність (ОСА). Ця властивість пов'язана з кількістю обчислень необхідних для досягнення результату.

Відразу розмежуємо обчислювальну складність як властивість алгоритмів і відповідні метрики та показники цієї властивості. До визначення показників ОСА є два принципово різних підходи: імовірнісний та статистичний. Історично склалось так, що імовірнісний підхід, з'явившись першим став домінуючим у цьому питанні.

Зважаючи на те що властивість ОСА виявляється при виконанні алгоритмів, та її значущість саме при виконанні, віднесемо цю властивість до сукупності експлуатаційних характеристик алгоритмів.

Кількість обчислень, а далі і обчислювальна складність залежить від вхідних даних. Тобто, якщо алгоритм реалізує відображення $X \to Y$ (де $X$ – множина вхідних даних, для кожного елемента якого алгоритм за скінчений час має знайти результат, який є деяким елементом з $Y$), то існує деяке відображення:

$$X \to K, K \subset N, \tag{1}$$

де $k \in K$ – загальна кількість виконаних обчислювальних операцій для отримання елемента з $Y$ за заданими вхідними даними $з\ X$, або кількість кожної з операцій ($k \in K \subset N^n$), якщо враховується різновид операцій.

Властивість ОСА цілком характеризується відображенням (1), яке як показник цієї властивості не використовується із деяких причин, насамперед через складність визначення. Наприклад, для алгоритмів сортування в такому разі необхідно встановити залежність кількості обчислень від кількості та значень елементів, що підлягають сортуванню. Навіть для таких простих та досить добре досліджених алгоритмів це досить складна задача.

Потрібні деякі інші показники ОСА (і вони існують), котрі можна визначити двояко: апріорно на основі аналізу алгоритму і апостеріорно, підрахунком кількості обчислень під час його виконання. У даній роботі буде розглянуто виключно першу можливість.

Навіщо потрібні показники обчислювальної складності? Безумовно вони опосередковано дають уявлення про час виконання алгоритмів. Мабуть тому серед фахівців з теорії алгоритмів та розробників алгоритмів нема одностайності у термінології. Одні в [7–13] цю властивість алгоритмів називають обчислювальною складністю

(computational complexity), інші [14–20] – часовою складністю (time complexity), просто складністю – [21–32], інакше, або ніяк не визначаючи властивість алгоритмів та виконують аналіз алгоритмів з визначенням часу або кількості обчислень [33–47]. У роботі [15] для часових характеристик алгоритмів, мабуть вперше, застосовано найбільш точний термін "часова ефективність". Для адекватного застосування суттєвостей, їх властивостей та показників цих властивостей слід обережно поводитись з термінологією. Визначені терміни мають якнайкраще відображати сутності. У даній роботі, поряд з іншим, аналізуються зазначені терміни та обґрунтовується їх застосування.

Обчислення часу виконання алгоритму на конкретній ЕОМ за відомою кількістю застосовуваних операцій неможливе через причини, пов’язані з багатьма засобами розгалуження обчислень такими як конвеєризація, наявність декількох конвеєрів або процесорів, ММХ – технологія тощо; непередбачуваністю часу виконання допоміжних операцій, таких як доступ до даних з їх хешуванням [48–50]. Тому для властивості алгоритму пов'язаної з кількістю обчислень будемо вважати більш доцільним використання терміну "обчислювальна складність", а з часом виконання –"часова ефективність".

Безумовно обчислювальна складність є тією властивістю алгоритму, яка суттєво впливає на часову ефективність, але практичне значення останньої є незаперечно найбільш пріоритетне.

Наявність показників часової ефективності алгоритмів потребується двома задачами практичного програмування. Перша полягає у виборі найбільш ефективного алгоритму з числа альтернативних при розробці програмного забезпечення (ПЗ). Друга – розробка нових алгоритмів та визначення їх ефективності для визнання їх придатності до застосування. Друга задача може виникнути й при розробці ПЗ, при відсутності потрібних алгоритмів, або їх непридатності за низькою ефективністю.

Для обґрунтованого застосування методології слід знати та уважно ставитись до її об'єктивних обмежень та особливостей. Метою даної роботи є дослідження можливості та обґрунтованості застосування імовірнісних методів аналізу алгоритмів з визначення обчислювальної складності до оцінки їх часової ефективності в контексті вирішення саме цих задач програмування.

Для цього розглянуті існуючі методи дослідження та показники ОСА, які об'єднані у два напрями. Перший пов'язаний з підрахунком операцій, другий – з асимптотичними оцінками.

## Аналіз алгоритмів за Д. Кнутом

Перший підхід до аналізу алгоритмів викладено у класичній роботі Д. Кнута 1972 р. [39]. Цьому передувало чимало досліджень окремих алгоритмів, виконаних різними дослідниками в 1955 – 1970 рр., що свідчать посилання Д. Кнута при аналізі алгоритмів [40].

Згідно Д. Кнута аналіз алгоритмів виконується таким чином:

– робляться деякі припущення щодо вхідних даних алгоритму. Наприклад, для алгоритмів сортування – щодо кількості елементів у масиві, заповнення масиву (різними випадковими числами, чи з можливістю однакових чисел, ступеня і порядку початкової відсортованості даних);

– для кожного кроку алгоритму, згідно обраних припущень, обчислюється кількість його виконання. Обчислення ґрунтуються на теорії імовірностей та комбінаториці. Для деяких команд кількість повторів однакова для кожного виконання алгоритму (можливо з деякими припущеннями щодо даних). Для інших команд визначається четвірка: <мінімальна, середня та максимальна кількість разів виконання команди, та дисперсія до середньої кількості > в залежності від кількості елементів даних;

– знаючи кількість повторів та час виконання кожного кроку нескладно обчислити час виконання алгоритму в цілому на конкретному комп'ютері. Д. Кнут приймає час виконання кожної команди деякої віртуальної машини MIX одиницю – *u*, та визначає час виконання алгоритму в найгіршому, середньому та найкращому випадках.

Приміром, час виконання алгоритму сортування пухирцем на машини MIX дорівнює $< \min : (8N + 1)u;$

$ave : (7{,}75N^2 + O(N \log N))u; \quad \max : (7{,}5N^2 + 0{,}5N + 1)u .>$

**Особливості інтерпретації результатів.** За сучасними уявленнями показником ОСА (для віртуальної машини MIX) запропонована нечітка функція часу виконання алгоритму від кількості елементів даних. Тобто аргументом функції є кількість елементів даних, а значенням – нечітке значення часу (*min, ave, max*) – класичне трикутникове уявлення нечіткої величини.

Нечіткість пов’язана з невизначеністю властивостей даних ( для алгоритмів сортування ступеня початкової відсортованості даних та їх порядку).

Зазначимо, що показники ОСА у вигляді нечіткої функції не дуже придатні для прийняття рішень і не відповідають вимогам до значень метрик якості програм. Згідно стандарту ISO/IEC 9126–2 рекомендовано застосовувати 5 видів шкал вимірювання значень [51]:

– номінальна шкала – розподілення значень за класами, наявність/відсутність деякої якості. Наприклад, метрика, яка визначає наявність серед вхідних даних алгоритму імовірної величини. Ця метрика поділяє алгоритми на детерміновані та імовірнісні за обчислювальними методами;

– порядкова – дозволяє впорядкувати характеристики за будь-яким порядком;

– інтервальна – визначає "відстань" між характеристиками;

– відносна – значення розрізняються щодо обраної одиниці, або еталону;

– абсолютна – визначає абсолютне значення характеристики.

Показник у вигляді нечіткої функції не відповідає ні одній з стандартизованих шкал.

**Залежність обчислювальної складності від даних.** Д Кнут зауважив, що "кожний метод має свої переваги та недоліки, тому він виявляється ефективнішим за інші при деяких конфігураціях даних і апаратури" [39] (тут метод слід розуміти як алгоритм).

У сенсі сказаного розглянемо припущення при яких визначалися показники ОСА при дослідженні алгоритмів сортування.

Щодо даних вважається, що елементи є випадковими і різними. Те, що елементи є різними з точки зору сортування записів з ключами відповідає більшості практичних випадків.

Але припустимо, що елементи є цілими числами $a_i \in [0, m]$, де $m << M$, $M$ – кількість елементів масиву. Звісно, що зменшення $m$ призведе до зменшення ОСА більшості алгоритмів, а при $m$=0 (елементи однакові), скажімо, для сортування пухирцем буде $N$ порівнянь і нуль перестановок тобто, обчислювальна складність значно зменшиться.

Як бачимо, обчислювальна складність багатьох алгоритмів сортування залежить від початкового ступеня відсортованості даних.

З цього зробимо висновок, що ОСА залежить не лише від кількості елементів, а і від них самих, тобто від даних в цілому.

**Залежність обчислювальної складності від виконавчих пристроїв.** Щодо "апаратури" зроблено такі припущення:

– всі команди виконуються однаковий час;

– час виконання команд не залежить від даних.

Друге припущення неявне (не зазначене Д. Кнутом) і витікає з першого.

Розглянемо друге припущення. Всі числа при обчисленнях на ЕОМ моделюються деякими модельними числами. Так цілі числа можуть моделюватись 2, 4, 8 байтовими числами зі знаком або без нього. Розрядність числа при цьому не залежить від його значення. Числа 5 і 31100, наприклад, будуть однаково представлені 16–бітовими (2–байтовими) модельними числами. Час виконання конкретної операції над даними, представленими однаковими модельними числами, однаковий.

Але уявимо, що значення елементів масиву $a_i$ – занадто великі і моделюються 16–байтовими числами. Якщо ЕОМ не має команд обробки таких чисел, то обчислення виконуються з двома 8–байтовими за декілька операцій. Подальше зростання значень $a_i$ призведе до моделювання 24, 32 … – байтовими числами і відповідно до зростання кількості операцій. Тобто чим більші модельні числа тим довші за часом відповідні операції, а відтак і час виконання алгоритмів.

Це ще раз підтверджує тезу, що обчислювальна складність алгоритмів залежить від даних в цілому, а не лише від їх кількості або обсягу.

З іншого боку, якщо ЕОМ має команди процесора з 8, 16, 32 … байтовими модельними числами, ОСА залишиться незмінною при відповідному зростанні значень чисел.

Можуть бути заперечення: для порівняння 16 байтових чисел, необхідно два порівняння (з умовними переходами) 8–байтових, тобто треба змінити алгоритм.

Але, по–перше, алгоритм не відображає представлення даних, які проектуються при розробці програми за наявним алгоритмом.

По–друге, відмітимо, що на мові програмування Ада модельні числа вибираються транслятором і відповідно моделюються операції над ними: у випадку коли ЕОМ має операції обробки таких модельних чисел, транслятор використовує ці операції, коли не має – генерується відповідна послідовність команд для виконання однієї операції. Алгоритм у вигляді програми на мові Ада ніяк це не відображає.

Наведені аргументи зазначають те, що кількість обчислень за алгоритмом (тобто ОСА) залежить від виконавчого пристрою.

Ще один такий аргумент. Припустимо, що операції над розрядами чисел виконуються не так як на сучасних ЕОМ паралельно–послідовно, а суто послідовно, як це зазвичай виконується вручну. Однією операцією буде операція над одним розрядом. Тоді кількість операцій буде залежати від розрядності елементів і обчислювальна складність значно зросте.

Зважаючи на викладене можна говорити що неявні операції алгоритмів, які виконуються або можуть виконуватись виконавчими пристроями, впливають на ОСА.

Отже, ОСА залежить від засобів виконання алгоритмів і при її визначенні слід враховувати ці особливості.

**Зв'язок між обчислювальною складністю та часовою ефективністю.** Акцентуємо увагу на трьох моментах. По–перше, кінцевим призначенням аналізу алгоритмів Д. Кнут вважав обчислення часу виконання алгоритмів. По–друге, час виконання обчислюється для *конкретного* комп'ютера. Далі, Д. Кнут вважає, що предметом області аналізу алгоритмів є визначення для заданого алгоритму його робочих характеристик. Хоча, сенсом усієї роботи є саме аналіз алгоритмів, але самі робочі характеристики, а тим більш метрики їх виміру ніяк не пойменовані і не визначені.

Д. Кнут визначав час виконання для віртуальної машини MIX, показово встановивши однаковим час виконання всіх її команд. Це припущення не є обтяжливим: методика легко модифікується, якщо час виконання

команд різний, але заздалегідь відомий. При тодішньому стані обчислювальної техніки такий підхід був обґрунтованим і не мав заперечень.

При сучасному стані обчислювальної техніки, з застосуванням різних засобів розгалуження обчислень, час виконання команд не є постійним і може розглядатись лише як імовірнісна або нечітка величина.

Тому перехід від кількісних обчислень до часу не є прийнятним для сучасних ЕОМ. Водночас кількість обчислень є об'єктивною характеристикою алгоритмів.

## Асимптотичні показники обчислювальної складності алгоритмів

Для визначення ступеня зростання ОСА при зростанні обсягів даних використовують асимптотичні показники ОСА. Перш за все це показник точної асимптотичної оцінки, який як і в [41] позначимо $\Theta(g(n))$. Це означає що знайдуться константи $c_1, c_2 > 0$ та $n_0$ такі, що $c_1 g(n) \le f(n) \le c_2 g(n)$ при $n > n_0$. Де $f(n)$ – показник ОСА, який підлягає оцінці, а $g(n)$ – відповідна асимтотика. В якості $g(n)$ за звичай беруться функції $n, n\log(n), n^2, n^3 \ldots, 2^n$. При цьому точна асимптотична оцінка поєднує в собі дві – верхню і нижню асимптотичні оцінки.

Асимптотичні оцінки ОСА здебільшого, але не завжди співпадають з асимптотичними оцінками часу виконання алгоритмів. Прискорення обчислень з причини розгалуження обчислень при конвеєрній обробці команд і виконанні на декількох процесорах не залежить від обсягів даних, що оброблюються.

Показники асимптотичної ОСА можна вважати метриками з порядковою шкалою. За нею алгоритми розподіляються на класи, які визначають потенційний час виконання алгоритмів при значних обсягах даних. Наприклад, алгоритми сортування поділяються на класи з асимптотичною ОСА $\Theta(n^2)$ і $\Theta(n\log(n))$.

Найбільш корисною є класифікація за якою алгоритми розподілені на *P* алгоритми, тобто такі, що обчислення за ними обмежені поліноміальною асимптотикою і *NP* – алгоритми, обчислення яких не обмежується поліноміальною асимптотикою.

Асимптотичні показники ОСА розкривають потенційні можливості щодо кількості обчислень та часу виконання алгоритмів за певних умов, а саме значного зростання обсягів даних, зменшенню похибок обчислень та ін. Таке розподілення на класи є корисним для встановлення потенційної цінності алгоритму, потенційних можливостях його використанні в різноманітних ПЗ.

При вирішенні задачі вибору алгоритму, цей показник може бути корисним, якщо потреби конкретного програмного засобу дійсно пов'язані зі значними вимогами до обсягів даних.

Якщо мається декілька альтернативних алгоритмів одного класу за асимптотичними показниками ОСА, обґрунтований вибір серед них за цим показником неможливий, але він надає можливість звузити коло прийнятних алгоритмів.

Може здатися, що в цьому випадку може бути вирішальним допоміжний показник:

$$\theta = \lim_{i \to \infty} \frac{f(n)}{g(n)},$$

де $f(n)$ має асимптотичною ОСА $\Theta(g(n))$. Звісно алгоритм з показником $\theta = 1$ значно корисніший ніж алгоритм з $\theta = 1000$ (при однаковій $\Theta(n)$). Тобто більш точно визначена асимптотика (оцінка за показником $\theta$) може впорядкувати алгоритми одної обчислювальної складності за $\Theta(n)$.

Але для задач оцінки алгоритму, який щойно розроблено це мало що додає, а для задачі вибору алгоритму при близьких асимптотах також, зважаючи на перспективу наперед невизначеного впливу розгалуження обчислень при їх виконанні на сучасних ЕОМ. Для багатьох алгоритмів асимптотичні показники є досить близькими, а різниця показника $\theta$ навіть у декілька разів може бути нівельована різним ступенем розгалуження обчислень та ступенем хешування.

Розглянемо ситуацію, яка викладена у [52]. Деякий процесор має спеціалізований функціональний пристрій для якихось обчислень, наприклад, виразів вигляду $\sum w_i x_i$, який виконує цю макрооперацію паралельно–послідовно.

Якщо в алгоритмі є цикл для обчислення зазначеної суми, то його обчислення має обчислювальну складність $\Theta(n)$. Уявимо, що транслятор розпізнає такий цикл і перетворює у відповідну команду процесору і процесор виконує лише одну команду (операцію). Тим самим транслятором та обчислювальним пристроєм ОСА знижена до $\Theta(1)$. Тобто наявний алгоритм і програма на різних обчислювальних пристроях може мати різну обчислювальну складність.

Відмітимо також, що кількість паралельних операцій у спеціалізованому функціональному пристрої завжди обмежена фізичними можливостями. Якщо обсяги даних перевищуватимуть можливості процесору заміна циклу однією операцією буде неможлива, і подальше збільшення обсягів даних, все ж призведе до обчислювальної складності $\Theta(n)$. Але може статися так, що обсяг даних при всіх практичних застосуваннях дозволить вищезазначену заміну. Тобто обчислювальна складність у практичному застосуванні буде $\Theta(1)$.

Підсумовуючи зазначимо можливу залежність асимптотичних показників ОСА від виконавчих пристроїв.

## Особливості обчислювальної складності алгоритмів наближених обчислень

Розглянемо обчислювальну складність алгоритмів наближених обчислень – методів знаходження кореня алгебраїчного рівняння, скінчених різниць, кінцевих елементів тощо. ОСА є досить важливою характеристикою для таких алгоритмів тому, що час їх виконання є достатньо великим навіть на сучасних ЕОМ і на кластерах.

Розглянемо алгоритм пошуку кореня рівняння з неперервною функцією за методом поділу відрізку навпіл, який показано псевдокодом.

| | |
|---|---|
| *обчислити* $y_a = f(a)$ | (1) |
| *цикл поки* $b - a > \varepsilon$ *повторювати* | (2) |
| *обчислити* $x = (a+b)/2$ | (3) |
| *обчислити* $y_x = f(x)$ | (4) |
| *якщо* $y_a * y_x < 0$ *то* | (5) |
| *обчислити* $a = x$ | (6) |
| *обчислити* $y_a = f(a)$ | (7) |
| *інакше* | |
| *обчислити* $b = x$ | (8) |
| *кінець циклу* | |

Обчислювальна складність алгоритму визначається кількістю виконання циклу, яка дорівнює [53]:

$$N_{ц} \approx \log_2((b-a)/\varepsilon) - 1,$$

а кількість обчислень алгоритму:

$$N_A \approx c \cdot \log_2((b-a)/\varepsilon) \cdot 1{,}5 N_f,$$

де $c$ – коефіцієнт, пов'язаний з обчисленням виразів у блоках 2 … 5, $N_f$ – обчислювальна складність функції $f(x)$, яка може обчислюватись за формулою, рядами (і тоді обчислювальна складність обчислення функції буде залежати від похибки обчислень) або іншим чином.

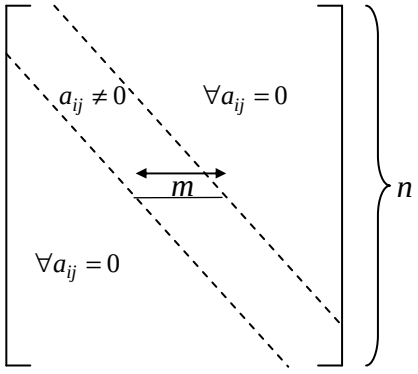
Як бачимо у випадку таких алгоритмів обчислювальна складність визначається вхідними даними $(a, b, \varepsilon)$ і ніяким чином не пов'язана з обсягами та кількістю вхідних даних. Показник асимптотичної ОСА визначається за умов $\varepsilon \to 0$.

Друга особливість: ОСА залежить від ОСА вкладеного (неявного) алгоритму обчислення функції.

Якщо ж функція у вигляді ряду обчислюється апаратно, то виявляється залежність ОСА і від виконавчого пристрою.

Розглянемо дещо інший приклад. Припустимо, що мається алгоритм розв'язку задачі плоскої деформації за методом кінцевих елементів. Вхідними даними є геометричний опис об'єкта дослідження у вигляді однієї або декількох областей, їх фізичних властивостей, місця прикладання та значення зовнішніх сил, закріплення та похибки. Алгоритм складається з чотирьох частин. У першій частині виконується розбиття областей на трикутникові елементи, у другій – формування системи лінійних алгебраїчних рівнянь, у третій – вирішення отриманої системи рівнянь і в четвертій візуалізація або аналіз пласко–напруженого стану.

ОСА складається з обчислювальної складності частин, серед яких найбільш обтяжливим є розв'язок системи рівнянь. Це зважаючи на те що кількість кінцевих елементів може вимірюватись тисячами й більше. Асимптотична ОСА алгоритму розв'язку системи алгебраїчних рівнянь за методом Гауса $O(n^3)$. Але у даному випадку вона значно менша за рахунок того, що матриця коефіцієнтів рівнянь є смугастою і переважна більшість коефіцієнтів дорівнює нулю. Асимптотичний показник ОСА становить $O(m^2 \cdot n)$, де $n$ кількість рівнянь, яка дорівнює подвоєній кількості вершин трикутників; $m$ – ширина смуги з ненульових коефіцієнтів рівнянь. За звичай $m$, що найменш на порядок менша за $n$.



Ширина смуги визначається як:

$$m = 2 \max_{(i,j)\in\Gamma} (i - j + 1),$$

де $i$, $j$ – номери вершин, $\Gamma$ – множина ребер, кожне з яких визначається парою номерів вершин, які ребро з'єднує.

Так, ширина смуги визначається вдалою нумерацією ребер. У випадку, як на рис. (а) оптимальною буде нумерація вершин за рядками, на рис. (б) – також, у цьому випадку ширина смуги буде визначатись кількістю вершин у рядку найбільш наближеного до діаметру. При більш складній формі областей послідовність нумерації наперед визначити занадто складно. Тому частина формування системи рівнянь має застосовувати алгоритм нумерації для зменшення ширини смуги.

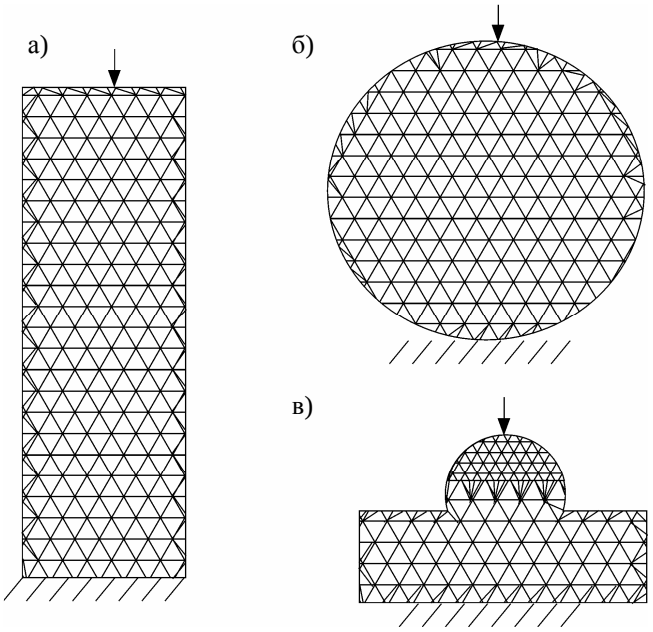



Рисунок. Розрахункові схеми з розбиттям областей на трикутникові скінчені елементи

Оптимальний розв'язок задачі нумерації є NP–повною задачею. Її розв'язок при значній кількості вершин (навіть порядку тисячі) є неможливим за прийнятний час. Тому алгоритм нумерації може бути або евристичним, або стохастичним і не гарантує оптимальності, але здатен реально значно зменшити ширину смуги.

Значення ширини смуги складно визначити апріорно. Та й значення кількості рівнянь деколи також, якщо сітка нерівномірна, а як того потребує дотримання однакових похибок у різних частинах області, густішає у місцях концентрації напружень (рис. в).

Підсумовуючи, ОСА третьої частини алгоритму (й усього алгоритму) залежить від значень $n$, що є результатом роботи першої частини алгоритму і $m$, що є результатом другої частини. Безумовно, ОСА всього алгоритму, як і інших, визначається вхідними даними, в першу чергу формою області та заданою похибкою. Але, його ОСА залежить від того, наскільки вдало виконана нумерація вузлів. Тобто ОСА залежить від функціональної ефективності [6] деякої частини алгоритму. Алгоритм має можливості впливати на свою ОСА.

Ще більш складною є ситуація щодо наближених стохастичних алгоритмів, наприклад генетичних алгоритмів. Автор не має відомостей щодо методів визначення показників ОСА для таких алгоритмів і практичного визначення показників ОСА для будь–яких з них. Навіть можливість визначення показників ОСА для стохастичних алгоритмів є під сумнівом.

## Висновки

Обчислювальна складність алгоритму – провідний фактор, що визначає його часову ефективність. Досить важливою передумовою її використання є відсутність потреби виконання алгоритму для її визначення. Але при застосуванні показників ОСА слід враховувати таке:

– при виборі алгоритму з числа альтернативних для застосування його в конкретному ПЗ можуть застосовуватись лише асимптотичні показники ОСА. Вони дозволяють визначити найбільш конкурентноздатні алгоритми. У випадку однакової асимптотичної ОСА ці показники не дозволяють розв'язати таку задачу взагалі;

– показники ОСА у вигляді нечіткої функції не відповідають вимогам стандартів щодо метрик якості;

– ОСА залежить від обчислювальних механізмів, на яких передбачається виконання алгоритмів. Реальна ОСА може бути як знижена, так і підвищена засобами обчислювальних механізмів;

– ОСА може бути змінена алгоритмічно: функціональна ефективність однієї частини алгоритму може суттєво впливати на ОСА іншої частини.

Крім того, зазначимо, що обчислювальна складність алгоритмів залежить від усієї множини вхідних даних. Залежність ОСА від обсягу вхідних даних слід розглядати як важливий типовий окремий випадок. Інший типовий випадок – залежність ОСА від похибки наближених обчислень.

Стосовно термінології. Мабуть є сенс за термінологією розподілити характеристики алгоритмів такі:

– обчислювальна складність алгоритму (ОСА), як характеристика алгоритму, пов'язана з кількістю обчислень при заданому представлені (зазначених у представленні операцій) для різних даних без урахування особливостей обчислювальних пристроїв. ОСА може бути визначена за представленням алгоритму;

– обчислювальна складність алгоритму орієнтованого на визначений обчислювальний пристрій або пристрої (ОСА+П). Характеристика пов'язана з кількістю обчислень при заданому представлені для різних даних з урахуванням набору примітивних операцій конкретних обчислювальних пристроїв;

– часова ефективність, як характеристика алгоритмів пов'язана з часом виконання алгоритму. Як окремий випадок – на конкретних пристроях.


1. *Холстед М.Х.* Начало науки о программах. – М.: Финансы и статистика, 1981. – 128 с.
2. *McCabe N.J. A* complexity measure // IEEE Trans. Sofware Eng. – 1976. – Vol. SE – 2, № 4. – p. 308 – 320.
3. *Черноножкин С.К.* Меры сложности программ. – Новосибирск, 1994. – 35 с. (Препр. / Ин–т систем информатики СО РАН; 94 – 21).
4. *Евстигнеев В.А., Кожевникова Г.П.* Топологические меры сложности программ. – Новосибирск, 1989. – 30 с. (Препр. / Ин–т точной механики и вычислительной техники. Новосибирский филиал; 89 – 23).
5. *Шинкаренко В.И.* Сравнительный анализ временной эффективности функционально эквивалентных алгоритмов // Проблемы программирования. – 2001. – № 3/4 – С. 31–39.
6. *Шинкаренко В.И.* Функциональная эффективность нечетко специфицированных алгоритмов // Проблемы программирования. – 2006. – № 1 . – С. 24–33.
7. *Канаєва Н.М..* Дослідження локальних алгоритмів розв'язання блочних задач булевого програмування: Автореф. дис. … канд. фіз.–мат. наук: 01.05.01 / Дніпропетровський держ. ун–т – Дніпропетровськ., 2000. – 16 с.
8. *Катленд Н.* Вычислимость. Введение в теорию вычислимых функций. – М.: Мир, 1983. – 256 с.
9. *Ковтун И.В.* Поиск части оптимальной разметки некоторого *NP*–полного подкласса *(max,+)* задач // Управляющие системы и машины. – 2003. – № 6 . – С. 33–38.
10. *Кожевникова Г.П.* Структуры данных и проектирование эффективной вычислительной среды. – Львов: ЛГУ, 1986. – 176 с.
11. *Макконнел Дж.* Основы современных алгоритмов. – М.: Техносфера, 2004. – 368 с.
12. *Stephens R.* Ready–To–Run Visual Basic Algorithms. – Wiley Computer Publishing, 1998. – 448 p.
13. *Tarjan R.E.* Data structures and network algorithms. – Philadelphia: SIAM, 1983. – 131 p.
14. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – 536 с.
15. *Ахо А., Хопкрофт Дж., Ульман Дж.* Структуры данных и алгоритмы. – М.: Изд. дом "Вильямс", 2000. – 384 с.
16. *Глибовець М.М.* Основи комп'ютерних алгоритмів. – К.: Видав. дім "КМ Академія", 2003. – 452 с.
17. *Зубов В.С.* Справочник программиста. Базовые методы решения графовых задач и сортировки. – М.: Информ.–изд. дом "Филинъ", 1999. – 256 с.
18. *Павлюк О.В., Савчинський Б.Д.* Ефективний синтаксичний аналіз та розпізнавання структурованих зображень // Управляющие системы и машины. – 2005. – № 5 . – С. 13 – 24.
19. *Успенский В.А., Семенов А.Л.* Теория алгоритмов: основные открытия и приложения. – М.: Наука, 1987. – 288 с.
20. *Neapolitan R., Naimipour K.* Foundations of Algorithms Using C++ Pseudocode, Third Edition. – Jones and Bartlett Publishers, 2004. – 618 p.
21. *Анисимов А.В.* Алгоритмы модулярной редукции // Проблемы программирования. – 1999. – № 1 – С. 80–91.
22. *Василенко О.Н.* Теоретико–числовые алгоритмы в криптографии. – М.: МЦНМО, 2003. – 328 с.
23. *Винокур А.Б. Кожевникова Г.П.* Формализованный анализ сложности алгоритмов иерархического типа на основе распознавания их классификационных свойств. – Киев, 1986. – 30 с. (Препр. / Ин–т . кибернетики АН УССР; 86–9).
24. *Гуц А.К.* Математическая логика и теория алгоритмов: учебное пособие. – Омск: Изд. Наследие. Диалог–Сибирь, 2003
25. *Зятьков Е.А.* Алгоритм сложности *O(nlogn)* минимизации максимума линейных функций. – Минск, 1993. – 16 с. (Препр. / Ин–т техн. кибернетики АН Белоруссии; 93 – 15).
26. *Пападимитриу Х. Стайглиц К.* Комбинаторная оптимизация. Алгоритмы и сложность. – М.: Мир, 1984. – 510 с.
27. *Сапоженко А.А.* Некоторые вопросы сложности алгоритмов: Учебное пособие. – М.: Изд. отдел ВМиК МГУ, 2001. – 46 с.
28. *Трауб Дж., Вожняковский Х.* Общая теория оптимальных алгоритмов. – М.: Мир, 1983. – 382 с.
29. *Черемушкин А.В.* Лекции по арифметическим алгоритмам в криптографии. – М.: МЦНМО, 2002. – 104 с.
30. *Harris S., Ross J.* Beginning algorithms. – Indianapolis: Wiley Publishing, Inc., 2006. – 564 p.
31. *Kreher D.L., Stinson D.R.* Combinatorial algorithms: generation, enumeration and search. – N.Y.: CRC Press, 1999. – 329 p.
32. *http://intsys.msu.ru/staff/vnosov/theoralg.htm Носов В.А.* Основы теории алгоритмов и анализа их сложности. – М., 1992. – 144 с.
33. *Абрамов С.А.* Элементы анализа программ. Частичные функции на множестве состояний. – М.: Наука, 1986. – 128 с.
34. *Анисимов А.В., Проценко В.С., Айрапетян Л.Р.* Алгоритмы неоднородной сортировки. – Киев, 1979. – 22 с. (Препр. / Ин–т . кибернетики АН УССР; 79–12).
35. *Бекнелл Д.М.* Фундаментальные алгоритмы и структуры данных в Delphi. – СПб.: ООО "ДиаСофтЮП", 2003 – 560 с.
36. *Вирт Н.* Алгоритмы + структуры данных = программа. – М.: Мир, 1985. – 406 с.
37. *Грин Д., Кнут Д.* Математические методы анализа алгоритмов. – М.: Мир, 1987. – 120 с.
38. *Гудрич М.Т., Тамассия Р.* Структуры данных и алгоритмы в Java . – Мн.: Новое знание, 2003. – 671 с.
39. *Кнут Д. Э.* Искусство программирования, том 1 // Основные алгоритмы. – М.: Изд. дом "Вильямс", 2000. – 720 с.
40. *Кнут Д. Э.* Искусство программирования, том 3 // Сортировка и поиск. – М.: Изд. дом "Вильямс", 2000. – 832 с.
41. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. – М.: МЦНМО, 2001. – 960 с.
42. *Макконелл Дж.* Анализ алгоритмов. Вводный курс. – М.: Техносфера, 2002. – 304 с.
43. *Пашко С.В.* Субоптимальные алгоритмы выполнения булевых операций над мультиполигонами. Часть 2 // Проблемы управления и информатики. – 2005. – № 3 – С. 60–75.
44. *Седжвик Р.* Фундаментальные алгоритмы на С. Анализ/Структуры данных/Сортировка/Поиск/Алгоритмы на графах. СПб: ООО "ДиаСофтЮП", 2003. – 1136 с.
45. *Сетин А.А.* О некоторых сложностных характеристиках неуниформных вычислений: Автореф. дис. … канд. физ.–мат. наук: 01.01.09 / МГУ – М., 1992. – 9 с.
46. *Трахтенброт Б.А.* Алгоритмы и вычислительные автоматы. – М.: Сов. радио, 1974. – 200 с.
47. *Mahmoud H.M.* Sorting. A distribution theory. – N.Y.: Wiley interscience publishing, Inc., 2000. – 394 p.
48. *Касперски К.* Техника оптимизации программ. Эффективное использование памяти. – СПб.: БХВ–Петербург, 2003. – 464 с.
49. *Шинкаренко В.И.* Зависимость временной эффективности алгоритмов и программ обработки больших объемов данных от их кэширования // Математические машины и системы. – 2007 – № 2. – С. 43–55.
50. *Шинкаренко В.И.* Временная оценка операций обработки структурированных данных с учетом конвейеризации и кэширования // Проблемы программирования. – 2006. – № 2–3 – С. 43–52.
51. *Андон Ф.И., Коваль Г.И., Коротун Т. М., Лаврищева Е.М., Суслов В.Ю.* Основы инженерии качества программных систем. – Киев: Академпериодика, 2007. – 670 с.
52. *Дорошенко А.Е., Рагозин Д.В.* Использование комплексных функциональных устройств микропроцессоров при оптимизирующей компиляции // Управляющие системы и машины. – 2004. – № 2 . – С. 46–55.
53. *Непомнящий В.А., Рякин О.М.* Прикладные методы верификации программ / Под ред. А.П. Ершова. – М.: Радио и связь, 1988. – 256 с.