дование сплавов для термопар: Сб. тр. / Гипроцветметобработка. – М., 1967. – Вып. 24. Т.II. – с. 54-65. 4. Саченко А.О., Кочан В.В., Турченко В.О. Дистрибутивна сенсорна жа підвищеної ефективності / Вісн. ДУ "Львівська політехніка", 1998. – с77-80. 5. Sachenko A., Kochan V., Turchenko V., Intelligent Distributed Sensor Network, Proceedings of the 15th IEEE Instrumentation and Measurement Technology Conference, St. Paul, USA – May 18-20, 1998, vol. 1, pp. 60-66. 6. Sachenko A., Kochan V., Turchenko V., Tymchyshyn V., Vasylkiv N., Intelligent Nodes for Distributed Sensor Network, Proceedings of the 16th IEEE Instrumentation and Measurement Technology Conference, Venice, Italy - May 24-26, 1999, vol. 3, pp. 1479-1484. 7. Golovko V., Grandinetti L., Kochan V., Laopoulos T., Sachenko A., Turchenko V. Sensor Signal Processing Using Neural Networks, Africon'99, Cape Town, South Africa, Sep 29-Oct 1, 1999, in press.

УДК 681.3

Черкаський М.В.
ДУ "Львівська політехніка", кафедра ЕОМ

МОДЕЛІ НЕФОРМАЛЬНИХ АЛГОРИТМІВ

Розглядаються дві моделі неформальних алгоритмів - алфавітний оператор і блок-схема (система Ляпунова). Вказуються способи зменшення часової складності алфавітних операторів та структурної складності блок-схеми алгоритмів.

Вступ. Словесне визначення поняття алгоритму як точного припису, який задає обчислювальний процес, що починається з довільного початкового даного (з деякої сукупності можливих для даного алгоритму початкових даних) і спрямований на отримання повністю визначеного цим початковим даним результату [1] лише в неявній формі перераховує сім параметрів алгоритму: систему початкових даних і умов, системи проміжних і кінцевих результатів, правило початку і закінчення, правило безпосереднього перероблення і правило виводу результатів, але не вказує, як утворюються ці параметри для конкретних задач.

Словесне тлумачення алгоритму є недостатнім для дослідження ряду математичних проблем, до яких зокрема належала необхідність доведення того, що деяка задача не має розв'язання. Тому виникла потреба в моделях, які дозволяли б обмеженою кількістю загально зрозумілих елементарних операцій розв'язувати будь-яку задачу або доводити, що задача не має розв'язання. Було запропоновано декілька уявних (математичних) моделей, які одержали назву формальних алгоритмічних систем (ФАС). ФАС ініціювали розвиток матричної теорії алгоритмів, предметом досліджень якої були характеристики складності алгоритмів. Прикладами ФАС є машина Тьюрінга-Поста, неформальні алгоритми Маркова, рекурсивні функції тощо. З точки зору наближення математичних моделей до реальних обчислювальних машин перевагу серед ФАС має машина Тьюрінга.

Машина Тьюрінга дозволяє визначати декілька характеристик складності алгоритмів розв'язання задач – часову, місткісну, опису програми. Часова складність оцінюється кількістю елементарних кроків машини Тьюрінга, яка потрібна для розв'язання задачі. Місткісна складність оцінюється кількістю комірок пам'яті, яка потрібна для розв'язання задачі. Складність опису програми – це кількість команд програми.

Але підрахунок часової складності алгоритмів з кроків машини Тьюрінга не має практичного використання. Альтернативним шляхом є одержання значень часової складності та деяких інших характеристик складності з аналізу моделей неформальних алгоритмів.

Неформальні алгоритми.

Під неформальними будемо розуміти алгоритми, яким правило безпосереднього перероблення задане переліком взаємозв'язаних операцій, але не вказано, яким чином передбачається виконання цих операцій. Звідси випливає, що неформальні алгоритми відрізняються від формальних тим, що кількість властивостей алгоритму обмежена детермінованістю, масовістю та дискретністю. Елементарність в загальному випадку не є властивістю неформальних алгоритмів. Операції неформальних алгоритмів є складними. Вони можуть бути задані обчислювальними та логічними процедурами, елементарними та спеціальними функціями або діями, які описуються словесно у деякому абстрактному алфавіті. Прикладами моделей неформальних алгоритмів є: алфавітний оператор, послідовність речень, яка використовує символи деякого абстрактного алфавіту, система Ляпунова, графічні моделі. Послідовність операторів на алгоритмічних мовах – програма – також є неформальним алгоритмом. Перелік моделей неформальних алгоритмів не вичерпується наведеними. Для нашого викладу важливим є питання, чи можуть вони бути основою для одержання еквівалентних їм неформальних або формальних алгоритмів зі зменшеними значеннями характеристик складності, наприклад, з мінімізованою часовою складністю. Розглянемо це питання на прикладах двох неформальних алгоритмів: алфавітного оператора і блок-схеми алгоритму.

Алфавітний оператор.

Алфавітний оператор має вигляд:

$$\Gamma(x_1, x_2, ..., x_n) = (y_1, y_2, ..., y_m), \quad (1)$$

де Г – оператор відображення, який задає сукупність математичних перетворень вхідних даних.

$(x_1, x_2, ..., x_n)$ – підмножина вхідних даних з системи початкових даних;

$(y_1, y_2, ..., y_m)$ – підмножина вихідних даних з системи кінцевих результатів;

Алфавітний оператор передбачає наявність параметрів і властивостей, крім однієї Математичні перетворення не задаються елементарними операціями, тому алфавітний оператор є неформальним алгоритмом.

Математичний зміст алфавітного оператора виду функції можна подати дискретними арифметичними операціями, це дозволяє кількісне визначення арифметичної часової складності алгоритму. Значення інших характеристик складності – таких, як місткісна, опису програми безпосередньо підрахувати неможливо.

Алфавітні оператори є об'єктом перетворення з метою одержання еквівалентних їм алфавітних операторів з мінімізованою часовою складністю. Ефективними способами зменшення часової складності алфавітного оператора є еквівалентні перетворення, апроксимація, використання попередніх обчислень, у тому числі табличний спосіб, різні форми розбиття множин початкових даних та проміжних результатів на підмножини та роздільні операції над ними, розбиття множини можливих результатів на підмножини, зміна порядку обчислень, дихотомічний пошук розв'язання задачі, розбиття задачі на підзадачі та інші. Всі ці способи широко використовуються при адаптації алгоритмів розв'язання задач на комп'ютерах. Як правило, в процесі мінімізації часової складності алгоритму використовують відразу декілька способів. Наприклад, при побудові алгоритму

швидкого перетворення Фур'є одночасно використовуються еквівалентні перетворення, апроксимація, використання попередніх обчислень, розбиття масивів вхідних даних, проміжних результатів і кінцевих результатів на підмножини, зміна порядку обчислень. Слід зауважити, що при побудові ефективних часових неформальних алгоритмів такі суто "апаратні" способи, як конвеєризація і паралелізм, не використовуються.

Блок-схема алгоритму.

На відміну від алфавітного оператора блок-схема алгоритму у неявній формі задає правила початку та правило закінчення. Правило безпосереднього перероблення наведено операційними та умовними вершинами. Для кожної вершини відома операція, яку потрібно виконати. Ця операція також може бути подана неформальним алгоритмом у вигляді блок-схеми, алфавітним оператором або у іншій формі. Тобто, для моделі неформального алгоритму, що розглядається, властивість елементарності кроку також не виконується. Поряд з арифметичними операційними вершинами можуть плануватися операції іншої природи, наприклад, розпізнання, пересилання, видачі алгоритмів управління та інші. Тому для неформальних алгоритмів часова складність в загальному випадку не використовується. Може бути використана часова складність, яка дорівнює кількості відпрацювань вершин з врахуванням кількості операцій в циклах. Але такий підхід не має практичного значення тому, що однозначність між часовою складністю у цій формі і реальним часом виконання операцій алгоритму на комп'ютері можна встановити з неприпустимою помилкою. Місткісна складність також не може бути пов'язана з кількістю вершин блок-схеми.

Натомість, складність опису програми визначається однозначно кількістю вершин вводу/виводу, операційних та умовних. Крім того, з'являється принципова можливість розрахування структурної складності, що визначається ступенем нерівномірності зв'язків, які з'єднують вершини блок-схеми.

Тобто, подання неформального алгоритму блок-схемою дозволяє оптимізувати алгоритм двох характеристик складності-опису програми і структури. Способи зменшення складності – опису програм пов'язані зі способами векторизації програм. Крім того, зменшення складності досягається використанням еквівалентних перетворень, використанням попередніх обчислень, розбиттям задач на підзадачі. Останній спосіб є основою об'єктно-орієнтованого програмування. Зменшення структурної складності доцільно у тих випадках, коли воно сприяє мінімізації складності опису програм. Наприклад, використання поточної залежності при векторизації дозволяє зменшити значення обох видів складності, в той час як розщеплення циклів спрощує лише складність опису програми.

Висновки

1. Для ефективної характеристики складності комп'ютерної реалізації переваги мають моделі неформальних алгоритмів перед формальними алгоритмічними системами.

2. Для моделі неформальних алгоритмів не існує властивість "елементарність", тому множина цих моделей має більшу потужність порівняно з формальними алгоритмічними способами.

3. Модель неформального алгоритму – алфавітний оператор – доцільно використовувати для побудови ефективних алгоритмів з часової складності.

4. Модель неформального алгоритму, який заданий блок-схемою, доцільно використовувати для побудови алгоритму складності опису програми.

Успенский В.А., Семенов А.Л. Теория алгоритмов: основные открытия и приложения. М., 1987, с.-286.