УДК 681.3

А.М.СЕРГІЄНКО, к.т.н., с.н.с.

ДОСКОНАЛИЙ КІСТЯК ГРАФА АЛГОРИТМА

Дано визначення досконалого кістяка алгоритма, показано, що за його допомогою складається розклад за стратегією найскорішого призначення і запропоновано алгоритм його побудови. Досконалий кістяк алгоритма дає змогу на його основі ефективно представити алгоритми в САПР обчислювальних спеціалізованих засобів і покращити процес їхньої структурної оптимізації.

The perfect maximum spanning tree (PMST) of the algorithm is proposed. It was shown that the ASAP – time table is built on the base of PMST in the natural way. The algorithm of PMST deriving from the SDF graph is proposed. PMST gives the opportunity to represent the algorithm effectively and to optimize the structural solution more quickly in the system-level CAD tools.

Статичне складання розкладу алгоритму – це основна операція при синтезі як паралельних програм, так і паралельних обчислювачів. Найбільш поширений метод складання статичного розкладу оснований на інтерпретації графа синхронних потоків даних (ГСПД) [1].

В роботі [2] запропоновано представляти ГСПД в багатовимірному просторі у вигляді конфигурації алгоритма (КА) $K_G=(K,D,A)$, де K – матриця векторів-вершин $\boldsymbol{K}_i$, що відповідають операторам алгоритма, D – матриця векторів-дуг безпосередніх інформаційних залежностей між операторами, A – матриця інцидентності ГСПД. В простому випадку можна використати трьохвимірний простір. Тоді в векторі-вершині $\boldsymbol{K}_i = <k_i,s_i,t_i>$ координати k_i, s_i, t_i дорівнюють типу оператора (наприклад, k=1 – множення), номеру процесорного елемента (ПЕ), де виконується цей оператор і такту, в якому записується в регістр або пам”ять результат цього оператора.

Еквівалентним структурним рішенням обчислювача відповідають КА, які відрізняються різними матрицями K і D. Однією з умов коректності КА є та, що серед векторів $\boldsymbol{K}_i$ не повинно бути двох однакових. Так як між матрицями КА є лінійний зв”язок, то матрицю D можна обчислити з виразу

$$D = KA. \qquad (1)$$

КА розщеплюється на просторову конфігурацію і конфігурацію подій, яким відповідають структура обчислювача і розклад виконання операторів. При цьому вектори $\boldsymbol{K}_i=<k_i, s_i, t_i>$ розкладаються на вектори $\boldsymbol{K}_{Si}=<k_i,s_i>$, що відповідають координатам ПЕ, і вектори $\boldsymbol{K}_{Ti} = <t_i>$, які означають моменти виконання відповідного операторів в ПЕ $\boldsymbol{K}_{Si}$. Тоді часова складова $\boldsymbol{D}_{\mathrm{Ti}} = <t_i>$ вектора залежності $\boldsymbol{D}_{\mathrm{i}}$ дорівнює затримці t_i пересилки або обробки відповідної змінної. Конфігурація подій представляє собою розклад виконання алгоритму, а його пошук – це знаходження значень $\boldsymbol{K}_{Ti}$.

Якщо в ГСПД є цикли міжітераційних залежностей, то сума векторів-дуг $\boldsymbol{D}_j$, що входять в будь-який з контурів графа дорівнює нулю, тобто для i-го контура

$$\sum b_{i,j}\, \boldsymbol{D}_j = 0,$$

де $b_{i,j}$ – елемент i-й строки цикломатичної матриці ГСПД. В такому контурі можна виділити вектор-дугу $\boldsymbol{D}_{Bj} = <0,0,-kL>$ міжітераційної залежності, який означає затримку на k ітерацій, де L – тривалість цикла обчислень або ітерації алгоритма в тактах.

В роботі [2] запропоновано знаходити координати векторів $\boldsymbol{K}_i$ на основі кістяка ГСПД. Кістяк – це мінімальний зв"язний підграф графа. Число дуг m в кістяку дорівнює числу вершин n без однієї, тобто $m = n - 1$. Кістяк одержують послідовним видаленням будь-яких дуг, які називають поперечними, доки граф зберігає зв"язність [3]. В нашому випадку, будемо видаляти тільки ті поперечні дуги, які кінцем інцидентні вершинам, в які входять кілька дуг.

Конфігурація алгоритма, який заданий кістяком, має таку властивість, що

$$D_o = KA_o; \tag{2}$$
$$K = D_o\, A_o^{-1}; \tag{3}$$

де A_o – матриця інцидентності кістяка ГСПД. Відмітимо, що матриця векторів-дуг D_o має розміри $rm = rn$, де r – розмірність простору, причому до кістяка добавлена базисна дуга, яка власне не належить до кістяка, а зв"язує початок системи координат з деякою вершиною кістяка.

Для зпрощення, нехай кожен оператор алгоритму виконується рівно за один такт. Тоді пошук розкладу алгоритму, граф якого є кістяком, полягає в наступному. Необхідно призначити всім векторам-дугам координати $\boldsymbol{D}_{Ti} = 1$ і знайти K_T з відношення (3). Решту координат векторів $\boldsymbol{K}_i$ знаходять з умов коректності КА. Це вийде найбільш швидкий розклад, так як кожен оператор виконується за один такт і немає зайвих затримок. Якщо одержана матриця K не задовольняє критерій оптимальності, наприклад, якихось ресурсів необхідно занадто багато, то деякі вектори $\boldsymbol{D}_{Ti}$, зв"язані з цими ресурсами, змінюють і повторюють пошук розкладу.

Якщо мається на увазі алгоритм з графом загального виду, то має сенс шукати розклад на основі його кістяка, як показано вище з застосуванням відношень (1) і (3). Але при цьому з"являються складнощі з появою неправильних розкладів, які потрібно знаходити і вилучати під час комбінаторної оптимізації розкладу. Пошук розкладу в такий спосіб суттєво залежить від вибору кістяка ГСПД. Щоб вибрати кістяк, який би завдавав менше клопоту при пошуку розкладу, розглянемо фактори, які викликають появу некоректного розкладу.

По-перше, в графі може бути *стягуюча дуга*, тобто така, що замикає маршрут з кількох інших дуг, як наприклад: $\boldsymbol{D}_4 = \boldsymbol{D}_1 + \boldsymbol{D}_2 + \boldsymbol{D}_3$. При цьому виникає неприпустима ситуація, коли стягуюча дуга входить в кістяк, тому що часові складові векторів-дуг, які з"єднає ця дуга, можуть бути від"ємними. Приклад такого випадку показаний на рис.1,а. Якщо прийняти вісь *ot* за вісь часу, то часова складова вектору $\boldsymbol{D}_3$ виявиться від"ємною.

По-друге, в графі є *поперечні дуги*, тобто такі дуги, які не є стягуючими, але додавання яких до кістяка робить в графі контури [3]. При цьому поперечна дуга $\boldsymbol{D}_i$ може виявитися такою, що якщо після визначення розкладу по (3) знайти координати $\boldsymbol{D}_i$ з виразу (1), то може скластись ситуація, коли $\boldsymbol{D}_{Ti} < 0$, тобто такий розклад буде неправильним. Така поперечна дуга $\boldsymbol{D}_i$ показана на рис.1,б. Поперечні дуги не входять в кістяк за визначенням. І в більшості випадків якщо вдало вибрати поперечну дугу, наприклад, на рис.1.в, - не $\boldsymbol{D}_i$, а $\boldsymbol{D}_k$, то можна побудувати відповідний кістяк, з допомогою якого одержати коректний розклад, використовуючи рівняння (3). Таким чином, якщо поперечна дуга входить в коротший маршрут з однієї вершини в другу, то завжди можна скласти коректний розклад.

Розглянемо такі поперечні дуги, наявність яких може призвести до неправильного розкладу. Таку дугу назвемо *критичною поперечною дугою*. В графі можуть бути рівнозначні маршрути з однієї вершини в іншу, наприклад, як на рис. 1,г, де $\boldsymbol{D}_1 + \boldsymbol{D}_2 + \boldsymbol{D}_3 = \boldsymbol{D}_4 + \boldsymbol{D}_5 + \boldsymbol{D}_6$. Тут критичною поперечною дугою є така, що замикає рівнозначний маршрут в контур, тобто $\boldsymbol{D}_3$. Також критичною поперечною дугою є така дуга, яка належить контуру, котрий формує підграф з декількома джерелами. Типовий такий підграф зображено на рис. 1,г і для нього $\boldsymbol{D}_1 - \boldsymbol{D}_3 = \boldsymbol{D}_2 - \boldsymbol{D}_4$.

В обох випадках розглянуті підграфи належать до класу узгоджених графів. *Узгодженим графом* називається граф, в кожному циклі якого кількість дуг – парна, а сума цих векторів-дуг дорівнює нулю [4]. Слід відмітити, що в кістяк необхідно включити повністю хоча б один з рівнозначних маршрутів. Тому якщо в начальному графові відмітити всі критичні поперечні дуги, то частина їх обов"язково попаде в результуючий кістяк, що може привести до неправильного розкладу. Тоді щоб одержати коректний розклад, необхідно в кістякові відмітити вершини, в які заходять критичні поперечні дуги і при пошуку розкладу перевіряти і коректувати часові складові дуг, які заходять в відмічені вершини.

По-третє, якщо алгоритм циклічний з міжітераційними залежностями, то вони в ГСПД відповідають дугам міжітераційних залежностей $\boldsymbol{D}_{Bj}$, причому, наприклад, для графа на рис.1,є $\boldsymbol{D}_{Bj} = -(\boldsymbol{D}_1 + \boldsymbol{D}_2 + \boldsymbol{D}_3)$ і $\boldsymbol{D}_{BTi} = -L$.

Вищеперелічені фактори необхідно враховувати при складанні розкладу. Щоб таких врахувань була мінімальна кількість, пропонується використовувати такий кістяк ГСПД, який спеціально підібраний для знаходження розкладу. Назвемо такий кістяк *досконалим*. Враховуючи викладене вище, можна зформулювати наступне визначення.

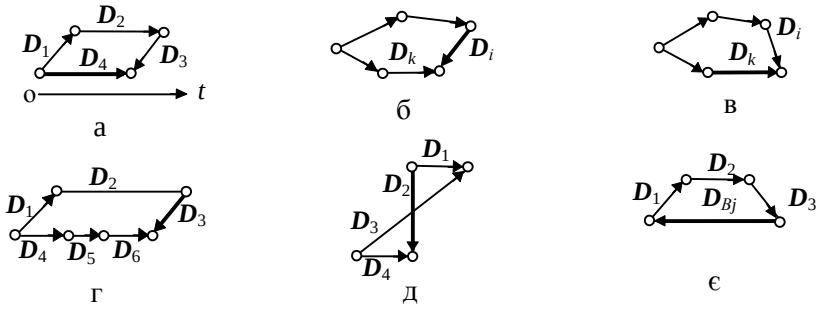

Рис.1. Приклади підграфів при побудові кістяка

Досконалий кістяк – це кістяк ГСПД, в якому: а) відсутні стягуючі дуги, б) відсутні дуги міжітераційних залежностей, в) дуги входять в найдовші маршрути, г) відмічені вершини, в які заходять критичні поперечні дуги.

Для досконалого кістяка можна довести наступні твердження.

Твердження 1. До досконалого кістяка належить критичний шлях алгоритму.

Доведення від супротивного. Нехай в досконалий кістяк не входить критичний шлях. Тоді в нього не входить деякий ланцюг дуг, що належать критичному шляхові. Це означає, що через крайні вершини цього ланцюга проходить по кістяку або стягуюча дуга, або більш короткий маршрут, що порушує визначення досконалого кістяка.

Твердження 2. Розклад, побудований на основі досконалого кістяка і рівняння (3) відноситься до розкладу за принципом найскорішого призначення (ASAP).

Доведення. Жадібний алгоритм ASAP оснований на топологічному сортуванні (нумерації) вершин ациклічного графа алгоритма згідно з їхнім частковим порядком, починаючи з начальної вершини і в призначенні на ресурси вершин в порядку їхньої нумерації [5]. В досконалому кістякові, як і будь-якому графові, топологічне сортування, тобто розмітка вершин натуральними зростаючими числами, починається з вершини-джерела, продовжується на суміжних вершинах і закінчується на вершинах-стоках. Таким чином, відсортовані вершини формують яруси, причому номери вершин наступного яруса завжди більші номерів вершин попереднього яруса, вершини якого виконуються раніше вершин наступного яруса [3].

Нехай вектор-вершини $\boldsymbol{K}_i=<k, s_i, t_i>$ і $\boldsymbol{K}_j=<k, s_j, t_j>$ мають топологічні номери i і j , $i < j$ і призначаються на ресурс k –го типу. Тоді, якщо одержано розклад типу ASAP, то момент виконання $\boldsymbol{K}_i$ повинен бути не більшим момента виконання $\boldsymbol{K}_j$, тобто $t_i \le t_j$, якщо вершини призначаються на різні ресурси, $t_i < t_j$, якщо вони призначені на один і той самий ресурс, а

якщо ці вершини зв"язані дугою, то різниця цих моментів буде мінімальною.

Дійсно, згідно з вимогою коректності КА, $<k, s_i, t_i> \neq <k, s_j, t_j>$ і тоді в першому випадку, $s_i \neq s_j$, а t_i не може бути більшим t_j, в другому випадку $s_i = s_j$, тоді повинно бути $t_i < t_j$. І в останньому випадку, оскільки вершини зв"язані дугою $\boldsymbol{D}_i$, часовій компоненті якої надано мінімальне значення, наприклад, 1 такт, то з рівності і $\boldsymbol{K}_j = \boldsymbol{K}_i + \boldsymbol{D}_i$ виходить, що $t_j = t_i+1$.

На основі визначення досконалого кістяка і доведених тверджень можна запропонувати наступний алгоритм побудови досконалого кістяка. Слід відмітити, що в більшості випадків пошуку розкладу розглядають граф з однією вершиною-джерелом і однією вершиною-витоком. В даному випадку ГСПД без дуг міжітераційних залежностей має декілька джерел і декілька витоків, і відповідно, в ньому є кілька критичних шляхів.

Начальні дані: орієнтований ГСПД G_A з m_I джерелами і m_O стоками. Результат – граф досконалого кістяка G_O.

1. Видалити всі дуги міжітераційної залежності. Відмітити вершини, в які заходять критичні поперечні дуги. Вибрати поточне джерело (вершину вводу або кінцеву вершину дуги міжітераційної залежності), для якого заданий (якщо заданий) мінімальний момент виконання, або джерело з номером 1. Задати пустий граф кістяка $G_O = \varnothing$.

2. Знайти з поточного джерела до m_0 стоків, які відмітити як поточні. Знайти критичні шляхи з поточного джерела в поточні стоки. Граф G_O зробити рівним об"єднанню графа G_O і цих критичних шляхів, тобто $G_O := G_O \cup G_{CRk}$, де G_{CRk} – критичний шлях з поточного джерела в поточний k –й стік. Якщо поточний стік відмічено як альтернативний, то G_{CRk} – критичний шлях максимальної довжини серед критичних шляхів в альтернативні стоки, а решта критичних шляхів в альтернативні стоки не розглядаються.

3. Знайти в G_A вершини, що не належать критичним шляхам, з яких ведуть дуги в вершини цих критичних шляхів. Ці дуги з G_A видалити а поточні стоки відмінити. Знайдені вершини призначити поточними стоками. Якщо такі вершини не знайдені, то перехід на п.5.

4. Знайти критичні шляхи, що ведуть з поточного джерела в поточні стоки. Граф G_O зробити об"єднанню графа G_O і цих критичних шляхів. Переход на п. 3.

5. В графі G_A видалити всі дуги, що з"єднують вершини, що входять в граф G_O, а також дуги, що ведуть в поточні стоки, в які були знайдені критичні шляхи в п.4. Вибрати наступне поточне джерело, як в п.1. Знайти стоки одержаного графа G_A і вважати їх поточними. Якщо стоки вже належать також графу G_O, то вони відмічаються як альтернативні.

6. Якщо не одержано пустий граф G_A, то перейти на п.2., інакше кінець.

На рис. 2 показаний приклад побудови досконалого кістяка згідно з даним алгоритмом.

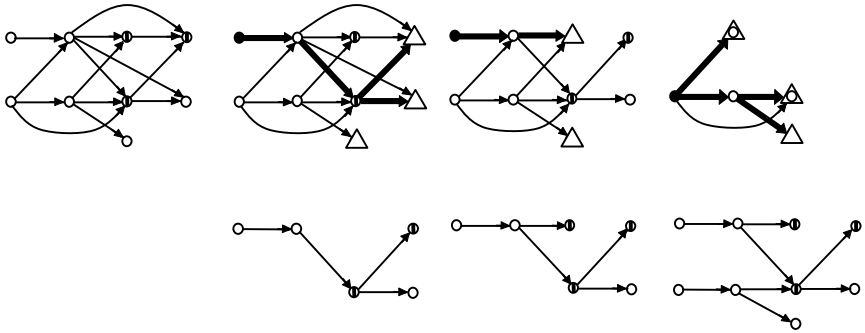

Рис.2. Приклад побудови досконалого кістяка. Верхній ряд – G_A, нижній ряд – G_O. а – начальний граф, б-г – стадії побудови кістяка; ● – джерело, △ – стік, ⧋ – альтернативний стік, ◑ – вершина, в яку заходять критичні поперечні дуги, ➞ – дуга з критичного шляху

Таким чином, в статті дано визначення досконалого кістяка алгоритма і доведено, що в нього входить критичний шлях алгоритма і показано, що за допомогою його натуральним чином складається розклад за стратегією найскорішого призначення. На основі властивостей досконалого кістяка запропоновано алгоритм його побудови. Досконалий кістяк алгоритма дає змогу на його основі ефективним чином представити відомості про алгоритм, який відображається в апаратні або програмні засоби і прискорити пошук структурного рішення, або користувацької програми, що відповідає заданим критеріям ефективності, таким як мінімальні час виконання і апаратні витрати.

Список посилань

1. Bhattacharyya S. S., Leupers R., Marwedel P. Software Synthesis and Code Generation for Signal Processing Systems // IEEE Trans. on Circuits and Systems—II: Analog and Digital Signal Processing, – 2000. –V47. –№9. –p.849-875.

2. Каневский Ю.С., Овраменко С.Г., Сергиенко А.М. Отображение регулярных алгоритмов в структуры специализированных процессоров // Электрон. Моделирование.–2002.–Т.24.–№2.–С. 46-59.

3. Евстигнеев В.А. Применение теории графов в программировании / Под ред. А.П.Ершова. – М: Наука. –1985. –352 с.

4. Воеводин В.В. Математические модели и методы в параллельных процессорах. –М.: Наука, –1986. –296 с.

5. The Systhesis Approach to Digital System Design / P. Micheli, U. Lauther, P. Duzy – Eds. Kluwer Academic Pub. –1992. –415 p.