відкритих середовищ для *Advanced Distributed Learning*.

IEEE Standard for Computer-Based Learning (*http://www.educause.edu/nlii/rfp/proposals/ieee.html*) – *IEEE* є однією з найбільших організацій з розробки стандартів (*Standards Development Organizations, SDO*), акредитованою в Американському національному інституті стандартів (*American National Standards Institute, ANSI*). Розробляє стандарти для "освітньої технології" (*Learning Technology*) *IEEE Learning Technology Task Force*. Передбачається об'єднання специфікацій з *AICC*, *IMS* та *ADL* в єдиний стандарт *IEEE*.

National Center for Supercomputing Applications – Educational Division. (*http://www.ncsa.uiuc.edu/edu/html/lete.html*) *NCSA* є давнім лідером в розробці програмного забезпечення для Інтернет, і тепер розробляє ряд проектів, що можуть вплинути на промислові стандарти в галузі навчальних систем.

Висновок

Успіх побудови системи навчання з використанням *Web*-технологій залежить від вдалого вибору структури та складових частин системи. Система навчання із застосуванням *Web* є комплексом окремих завершених програмно-технічних рішень (системи донесення навчального матеріалу, зберігання, тестування, дискусій, текстових повідомлень тощо). Тому особливо важливим є питання організації взаємодії її компонентів. Для цього доцільно використовувати функціонально повні платформи, такі, як *Lotus Notes/Domino*, *Microsoft Back Office*, *Oracle 8i*.

УДК 681.3

ФОРМАЛІЗАЦІЯ АЛГОРИТМУ АВТОМАТИЧНОЇ ПОБУДОВИ ОПТИМАЛЬНОГО РОЗКЛАДУ ПОСЛІДОВНОСТІ РОБІТ, ЩО ЗАДАЄТЬСЯ АЦИКЛІЧНИМ СПРЯМОВАНИМ ГРАФОМ

© Олександр Павлов, Людмила Аксенова, Ольга Кулікова
Національний технічний університет України "КПІ", м. Київ, пр. Перемоги, 37

Розглядається програмний продукт розв'язання задачі побудови оптимального розкладу послідовності робіт, відношення порядку на якій задається ациклічним

спрямованим графом загального виду. Алгоритм реалізує послідовність дій побудови оптимального розкладу для строго послідовно-паралельного графа [1]. Узагальнення алгоритму здійснюється ітераційним аналізом графа загального вигляду та декомпозиції його на множини максимальних пріоритетів, що є послідовно-паралельними підграфами. Досліджується ефективність застосованих алгоритмічних рішень.

The subject is software for the optimal schedule construction on the sequence of works with the precedence constraints assigned by the non-circle directed graph. The base of above mentioned software is an algorithm of the optimal schedule construction on the set of works with the precedence constraints assigned by series-parallel graph [1]. This algorithm is generalized by means of the general-form graph iteration analysis and decomposition onto the maximal priority sets which are the series-parallel sub-graphs. The effectiveness of used algorithmic methods is investigated.

Постановка задачі

Існує частково впорядкована множина $J = (j_1, j_2, ..., j_n)$ завдань, що обслуговуються одним пристроєм. Для кожного завдання j відомі тривалість $l_j \geq 0$ його обслуговування та вага w_j (довільні дійсні числа). Завдання обслуговуються послідовно і безперервно.

Розв'язання задачі формування оптимального розкладу на множині робіт J передбачає пошук такої послідовності обслуговування завдань, сумарний зважений момент закінчення виконання якої є мінімальним:

$$F = \sum_{k=1}^{n} w_{j_{[k]}} c_{j_{[k]}} \to \min, \qquad (1)$$

де $c_{j_{[k]}}$ – момент закінчення виконання завдання, що стоїть у припустимому розкладі на k-й позиції; $c_{j_{[k]}} = \sum_{s=1}^{k} l_{j_{[s]}}$.

Вказана задача є *NP*-складною у сильному сенсі і залишається такою, якщо усі тривалості та ваги дорівнюють 1. Її можна розв'язати за поліноміальний час, якщо порядок передування є "лісом" або послідовно-паралельним графом. Якщо замість умов передування ввести індивідуальні моменти надходження, то така задача буде *NP*-складною у сильному сенсі, навіть якщо усі ваги завдань дорівнюють одиниці.

Формування оптимального розкладу послідовності робіт з відношенням порядку, що задається послідовно-паралельним графом

1. Прочитується файл, який містить послідовність елементів множини вершин спрямованого ациклічного графа $\{x_1(n_1, w_1, l_1, a_{11}, a_{21}, \dots, ak_1), x_2(n_2, w_2, l_2, a_{12}, a_{22}, \dots, ak_2), \dots, x_m(n_m, w_m, l_m, a_{1m}, a_{2m}, \dots, ak_m)\}$.

Причому кожна робота x*i* характеризується номером n_i, вагою w_i, тривалістю виконання l_i та множиною робіт $a1_i, a2_i, \dots, ak_i$, які їй безпосередньо передують. Для незалежної роботи $\{a1_i, a2_i, \dots, ak_i\} = \varnothing$.

Наявний варіант формального алгоритму не потребує інформації щодо робіт, яким

безпосередньо передує робота, що розглядається.

2. Будується припустима послідовність робіт, причому незалежні роботи виносяться на початок послідовності та впорядковуються за спаданням пріоритетів. На них формується початковий набір часткових перестановок, які надалі можуть перевпорядковуватись, мінятися місцями та об'єднуватись. Отже, у пам'яті ЕОМ формується така структура:

$$p_1 = \{x_1\},\ p_2 = \{x_2\},\ \ldots,\ p_r = \{x_r\},\ y = \{x_{r+1}, x_{r+2}, \ldots, x_m\},\ p(x_1) \geq p(x_2) \geq \ldots \geq p(x_r), \quad (2)$$

де r - кількість незалежних робіт у вихідній множині, $p(x_i) = w_i / l_i$ - пріоритет i-ї роботи.

3. Виконується базова процедура оптимізації, яка реалізує алгоритм мінімізації зваженого моменту для послідовно-паралельного графа [2], описання формального варіанта якої подається нижче.

4. Одержаний розклад зберігається у файлі, обчислюється значення функціонала, на вимогу користувача формується файл протоколу дій, які виконує програма.

Базова процедура побудови оптимального розкладу

Існує впорядкований набір часткових перестановок, який на початку роботи процедури має вигляд (2). У загальному вигляді перестановки pj складаються з впорядкованих послідовностей робіт - ланцюжків, конструкцій та складних конструкцій [2], які було сформовано протягом попередніх кроків алгоритму. Крім того, під час виконання програми початкові часткові перестановки, своєю чергою, можуть об'єднуватись у конструкції та складні конструкції, формуючи множини максимального пріоритету.

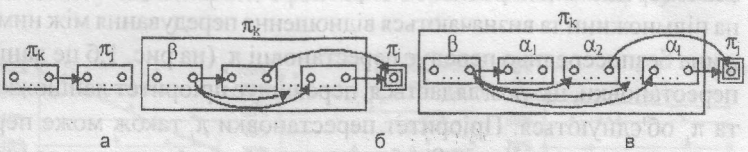

Рис.1. Варіанти передування

З припустимої послідовності s, що містить залишок вихідної множини робіт, процедурі одна по одній передаються роботи $x_{r+1}, x_{r+2} \ldots x_m$. Робота, що передається, також формально вважається частковою перестановкою $\pi_j = \{x_{r+j}\}$, $r+1 \leq j \leq m$. Отже, у пам'яті комп'ютера формується така послідовність:

$$[\pi_1]\ \ [\pi_2]\ \ \ldots\ \ [\pi_k]\ \ [[\ \pi_j\ ]]\ \ [\sigma = \{x_{j+1}, x_{j+2}, \ldots, x_m\}].$$

Процедура аналізує відношення даної та попередньої перестановок, при цьому визначається:

1. Пріоритет даної перестановки $p(\pi_j)$;
2. Пріоритет попередньої перестановки $p(\pi_k)$;
3. Наявність відношення передування між даною та попередньою перестановками $\pi_k > \pi_j$;
4. Наявність безпосереднього передування $\pi_k >> \pi_j$.

Виділяються три варіанти поєднання параметрів.

1. $p(\pi_j) > p(\pi_k)$ та $\pi_k >/ \pi_j$ - пріоритет перестановки, що розглядається, перевищує пріоритет перестановки, що стоїть на попередній позиції, при цьому їх не поєд-

нують відношення передування.

У такому випадку перестановка π_j переноситься через перестановку π_k, і виконується наступний крок циклу оптимізації: визначається співвідношення з перестановкою π_{k-1} і т.д.

2. $p(\pi_j) > p(\pi_k)$ та $\pi_k >> \pi_j$ або $\pi_k > \pi_j$ - перестановка, що стоїть перед тією, що розглядається, безпосередньо передує останній або містить роботу, яка є безпосереднім попередником перестановки π_j, причому пріоритет перестановки π_j є вищим, тобто при поєднанні з π_j перестановка π_k "підсилюється".

Різні варіанти передування показані на рис. 1.

У випадках а) та б) перестановці π_j безпосередньо передує робота (ланцюжок) або конструкція, у якій усі ланцюжки-сусіди загального попередника β так само безпосередньо їй передують. У подібних випадках відбувається об'єднання перестановок $\pi_k' = \{\pi_k, \pi_j\}$, і на наступному кроці оптимізації обробляється вже перестановка π_k'.

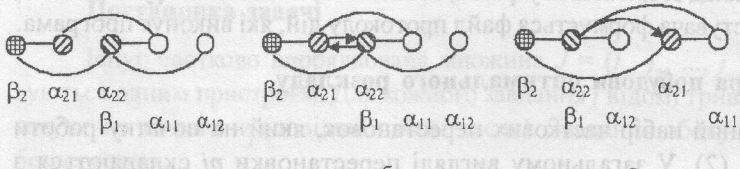

Рис.2. Обгрунтування можливості виконання "перестановки погіршення"

Я к щ о перестановка π_k - це конструкція, яка не є безпосереднім попередником перестановки π_j, відбувається розділення конструкції π_k на підмножини та визначаються відношення передування між ними. Якщо деяка підмножина безпосередньо передує перестановці π_k (на рис. 1,б це ланцюжок α_2), і пріоритет перестановки, що розглядається, перевищує пріоритет ланцюжка α_2, то підмножини α_2 та π_k об'єднуються. Пріоритет перестановки π_j також може перевищувати пріоритет загального попередника β, або перестановка π_k виявляється ще одним його сусіднім ланцюжком. Далі процедура оптимізації обробляє вже елементи множини сусідів загального попередника β у конструкції π_k. Після впорядкування одержана перестановка π_k' розглядається як нова підмножина у циклі базової процедури оптимізації, тобто з'ясовується її позиція відносно попередньої перестановки (якщо така є) і т.д.

3. У всіх інших випадках цикл впорядкування припиняє свою роботу і далі розглядається наступна робота вихідної послідовності.

Залежно від вихідної множини робіт можливе одержання кількох варіантів оптимального розкладу, але усі такі варіанти розрізняються лише послідовністю підмножин, що мають однаковий пріоритет, а це не впливає на остаточне значення показника якості. У всіх випадках переставлення подібних підмножин це значення не покращує.

Процедура розділення на підмножини

Для дослідження структури конструкції застосовується процедура розділення на підмножини. Ця процедура обробляє лише роботи об'єднаної перестановки $\{\pi_k, \pi_j\}$, що їй передаються.

З усіх робіт початкової частини перестановки π_k до загального попередника включно формується перша часткова перестановка π_α. Далі послідовно розглядаються наступні роботи, і, у випадку підсилення та передування, об'єднуються у нові часткові перестановки. Якщо деяка робота x_k характеризується меншим значенням пріоритету

або не має попередника серед робіт перестановки, що стоїть перед нею, вона формує наступну часткову перестановку і т.д. відповідно до базового алгоритму побудови оптимального розкладу.

Одержаний набір перестановок-сусідів впорядковується згідно з їх пріоритетами.

У загальному випадку під час перевпорядкування перестановок можливе утворення підмножини нижчого рівня, яка також потребуватиме розділення на підмножини з подальшим перевпорядкуванням. У програмі така можливість передбачається шляхом рекурсії виклику процедури розділення на підмножини.

Після закінчення роботи даної процедури підмножини, що розглядаються, об'єднуються у множини максимального пріоритету, які й повертаються базовій процедурі. Далі вони вводяться у загальний цикл впорядкування.

Якщо під час побудови розкладу на множині робіт, що задається графом загального вигляду, зустрічаються лише випадки алгоритму для послідовно-паралельного графа, то одержаний за розглянутою схемою розклад є оптимальним для даної індивідуальної задачі.

Виконання "перестановок погіршення"

"*Перестановкою погіршення*" називаємо таке перевпорядкування підмножин вихідного набору робіт, при якому підмножина переноситься через перестановки вищого пріоритету для того, щоб після об'єднання з попередником сформувати конструкцію, пріоритет якої дасть нам змогу здійснювати ефективне перенесення, і, як результат, дана конструкція займе у розкладі більш ранню позицію.

а б в

Рис.3. Дослідження ефективності формального алгоритму: а) залежність від кількості робіт у вихідному графі; б) внесок базового та вдосконаленого алгоритмів; в) залежність від розкиду значень ваг та тривалостей.

Обгрунтування припустимості подібних дій полягає в такому. На рис. 2,а показана структура, яка передбачає можливість успішного застосування "перестановки погіршення". Існує первинний попередник β_1, одна з сусідніх робіт (ланцюжків, конструкцій) якого пов'язана відношенням передування з роботою α_{21}. Причому і α_{21}, і β_1, в свою чергу, є сусідніми роботами (ланцюжками, конструкціями) вторинного попередника β_2. Наявна послідовність *p*-впорядкована в результаті роботи алгоритму мінімізації зваженого моменту для послідовно-паралельного графа. Зв'язок α_{21}-α_{11} порушує умови послідовної паралельності, і можлива ситуація, коли пріоритет об'єднаних робіт α_{22} і α_{12} перевищить пріоритет іншої сусідньої роботи вторинного попередника β_2.

Для зведення подібних задач до класу послідовно-паралельних вводяться два взаємовиключальні зв'язки, і далі розглядаються одержані послідовно-паралельні графи. Результуючі функціонали порівнюються, і надалі розглядається той з варіантів, який якнайкраще оптимізує вихідний розклад. Так досліджуються усі можливі варіанти розкладу на графах, утворених з вихідного введенням додаткових зв'язків, а побудовані для них оптимальні розклади також є оптимальними і для вихідного, більш розрідженого,

графа.

Отже, перший з двох можливих випадків передбачає введення зв'язку α_{21}-α_{22}. Ми одержуємо послідовно-паралельний граф (за винятком надлишкових зв'язків α_{21}-α_{11} та β_2-α_{22}), але оскільки в результаті роботи базового алгоритму вихідний набір підмножин вже було *p*-впорядковано, така послідовність робіт зберігається, тобто одержаний розклад є оптимальним як для графа з додатковим зв'язком, так і для вихідного, більш розрідженого графа.

Другий і останній з усіх можливих випадок передбачає введення зв'язку α_{22}-α_{21}. Одержаний послідовно-паралельний підграф p-впорядковується і приводиться до вигляду, показаного на рис. 2,в. Такий граф також містить надлишкові зв'язки (β_2-α_{21} та α_{22}-α_{21}), які не порушують хід базового алгоритму. Вибирають найкращий варіант на основі порівняння показників якості двох сформованих розкладів.

Отже, аналізуються $2N$ варіанти розкладу, де N – кількість "конкуруючих" пар сусідніх робіт вторинних попередників β_2, які зустрічаються у вихідному графі. Тобто замість перебирання усіх варіантів впорядкування застосовується цілеспрямований вибір одного з двох можливих.

Після кожної вдалої перестановки програма обробляє одержаний розклад базовою процедурою оптимізації для впорядкування тієї частини вихідної послідовності, з якої було вилучено підмножину, яка здійснила "перестановку погіршення".

Дослідження ефективності роботи програми

Для перевірки ефективності роботи формального алгоритму було створено програмний модуль генерації випадкових графів. Керують властивостями таких графів визначаючи кількість робіт, ймовірнісний коефіцієнт заповнення графа та розкид значень ваги та тривалості робіт. Ймовірнісним коефіцієнтом заповнення графа є число, що характеризує ймовірність появи кожної з робіт графа як безпосереднього попередника роботи, що розглядається.

Було згенеровано загалом 620 випадкових графів з різними параметрами, з яких програма, що реалізує базовий алгоритм, дала змогу побудувати оптимальний розклад для 438, тобто для 80% від їх загальної кількості. Треба також зауважити, що вдосконалена версія програми, яка виконує "перестановки погіршення", дала змогу покращити цей результат на 6% за рахунок графів з великою кількістю робіт та значним заповненням. Отже, ефективність формального алгоритму складає 86%, причому це значення можна збільшити при подальшому вдосконаленні програмного продукту.

Результати дослідження можливостей формального алгоритму під час складання оптимального розкладу для графів з різними параметрами є такими.

1. Залежність ефективності алгоритму від кількості робіт у вихідному графі показана на рис. 3,а. Статистичні дані свідчать про те, що збільшення складності графа призводить до лінійного спадання ефективності алгоритму.
2. Цікавим видається факт, що при невеликій кількості робіт у вихідному графі (до 30) вдосконалення базового алгоритму фактично не використовуються, і ефект від їх застосування незначний, в той час як при максимально можливій кількості робіт їх значення зростає. На рис. 3,б наочно підтверджується необхідність пошуку подальших методів вдосконалення базового алгоритму та роз-

ширення його можливостей для підвищення ефективності роботи програми.

3. Дослідження залежності ефективності формального алгоритму від розкиду значень ваг та тривалостей виконання робіт вихідного графа показали незначне зниження ефективності із зростанням максимуму значень w та l, що задаються випадково (рис. 3,в). Це пояснюється ускладненням структури множин максимального пріоритету, які формуються під час оптимізації розкладу.

Діючий варіант алгоритму можна застосовувати для графів, які мають довільні надлишкові відношення передування, які припустимо ігнорувати, якщо вони не порушують хід алгоритму, і при видаленні яких формується еквівалентний послідовно-паралельний підграф, оптимальний розклад для якого є також оптимальним і для вихідного графа [2].

Дані статистичних досліджень дають змогу зробити висновок про те, що найкращих результатів застосування цього програмного продукту досягаємо на графах, що містять до 50 робіт, максимальні значення ваг та тривалостей яких не перевищують 60. Оптимальний розклад на графах, що генеруються випадково, з кількістю робіт близько 100 буде формуватись, головним чином, за рахунок вдосконаленого алгоритму, який реалізує "перестановки погіршення" та рекурсивний аналіз конструкцій.

1. *Танаев В.С., Гордон В.С., Шафранский Я.М.* Теория расписаний. Одностадийные системы. М., 1984.
2. Конструктивные полиномиальные алгоритмы решения индивидуальных задач из класса NP / А.А. *Павлов, А.Б. Литвин, Е.Б. Мисюра* и др. К., 1993.
3. *Pavlov A.A., Pavlova L.A.* About one subclass of polynomially solvable problems from class "Sequencing jobs to minimize total weighted completion time subject to precedence constraints"//Вестник международного Соломоновского университета. № 1. С. 109-116.

УДК 681. 84. 087. 4

МЕТОДИ МОДИФІКАЦІЇ ЧАСОВОГО МАСШТАБУ МОВНОГО СИГНАЛУ В СИСТЕМАХ АНАЛІЗУ-СИНТЕЗУ МОВИ

© Роман Марцишин, Юрій Рашкевич
НУ "Львівська політехніка", м. Львів, вул.С. Бандери, 12

В статті наведено огляд моделей представлення мови та методів модифікації часового масштабу мовного сигналу в системах аналізу-синтезу мови.