View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Electronic Archive of Kyiv Polytechnic Institute

УДК 519.854.2

ПАВЛОВ О.А.,
МІСЮРА О.Б.,
ХАЛУС О.А.,
КОСТИК Д.Ю.,
ЛИСЕЦЬКИЙ Т.М.

ОПТИМІЗАЦІЯ МОДИФІКОВАНОГО ПДС-АЛГОРИТМУ ЗАДАЧІ МІНІМІЗАЦІЇ СУМАРНОГО ЗАПІЗНЕННЯ ВИКОНАННЯ ЗАВДАНЬ

Наводиться блок-схема алгоритму розв'язання задачі мінімізації сумарного запізнення, нові правила відсікань та декомпозиції алгоритму для зменшення сумарного часу обчислення, статистика розв'язання задач.

The logic diagram for algorithm of the total tardiness minimization, new truncation and decomposition rules for the algorithm to decrease the total calculation time, and the statistics of the tasks solution are given.

Вступ

У теорії розкладів (ТР) особливе значення мають задачі з одним приладом. Результати, отримані при дослідженні таких задач, можуть бути використані для побудови алгоритмів вирішення складніших задач з багатьма приладами і багатостадійних задач, що виникають на практиці. Класичні задачі ТР з одним приладом є схематичними теоретичними моделями багатьох задач, що зустрічаються на практиці. Дослідження таких задач допомагає вивчити фундаментальні властивості і структуру практичних задач, що сприяє побудові ефективних алгоритмів їх рішення.

«Мінімізація сумарного запізнення виконання незалежних завдань з директивними строками одним приладом» (МСЗ)

Припустимо, що задано множину незалежних завдань $J = \{j_1, j_2, \ldots, j_n\}$, кожне з яких складається з однієї операції. Для кожного завдання відома тривалість виконання l_j і директивний строк виконання D_j. Завдання надходять у систему одночасно в момент $d_j = 0, j = \overline{1,n}$. Переривання не допускаються. Необхідно побудувати розклад виконання завдань для одного приладу, що мінімізує сумарне запізнення при виконанні завдань:

$$f = \sum_{j=1}^{n} \max\left(0, C_j - D_j\right),$$

Де C_j – момент завершення виконання завдання j.

Огляд відомих методів розв'язання цієї задачі приведений у [1]. Згідно Ду і Люнгу, задача є NP-складною.

На кафедрі АСОІУ було запропоновано ефективний точний ПДС-алгоритм (алгоритм із поліноміальною й експоненційною складовими) розв'язання задачі, заснований на новому підході до розв'язання задач з директивними строками, що полягає в оптимальному використанні резервів часу незапізнених завдань, що дозволяє розв'язувати задачі з числом завдань, істотно більшим, ніж 500.

Блок-схема алгоритму

Даний алгоритм складається з двох етапів – попереднього етапу і етапу оптимізації. Спочатку виконується попередній етап, який включає сортування всіх завдань за тривалістю виконання та так звані вільні перестановки. Ціль вільних перестановок перемістити кожне завдання на максимально можливу позицію, на якій воно не буде запізнюватися, якщо на інтервалі перестановки є завдання, що запізнюються. Вихідна послідовність є вхідною для наступного оптимізаційного етапу алгоритму. На цьому етапі спочатку визначається множина конкуруючих завдань (блок 1). Далі для кожного наступного завдання із цієї множини виконується оптимізація підпослідовності з початку до поточного завдання. Оптимізація починається із пошуку позиції p вбудовування завдання, на якій воно не буде запізнюватися (блок 3). Якщо на позиції p знаходиться завдання, яке не запізнюється (блок 10), то виконується вбудовування поточного завдання на позицію p (блок 11) і перехід на блок 20. В іншому випадку аналізуються резерви часу завдань, що передують в поточній послідовності завданню, що

розглядається (блок 12). У випадку, якщо резерви є, завдання вбудовується на нову позицію p^H (блок 16) і виконується оптимізація за рахунок резервів на інтервалі $[1, p^H]$. Якщо завдання зайняло більш ранню позицію, тоді відбувається перехід на блок 20 (перехід А). Якщо резерви відсутні, то корегується позиція вбудовування (блок 15), і якщо позиція вбудовування менше g, то відбуваєть відбувається перехід до блока 20. Інакше на блок 22 (перехід В). У блоці 20 завдання впорядковуються за неспаданням їх тривалостей. Завдання $j^H_{[p]}$ помічається *.

Якщо воно не запізнюється, то відбувається перехід до наступного конкуруючого завдання (блок 2). Інакше знаходиться завдання, помічене * або ** із більшою тривалістю (блок 22) і запам'ятовується поточна послідовність (блок 24). Далі знайдене завдання (* або **) вбудовується після нової позиції p^H (блок 25) і виконується оптимізація підпослідовності за вище наведеними правилами (блок 26).

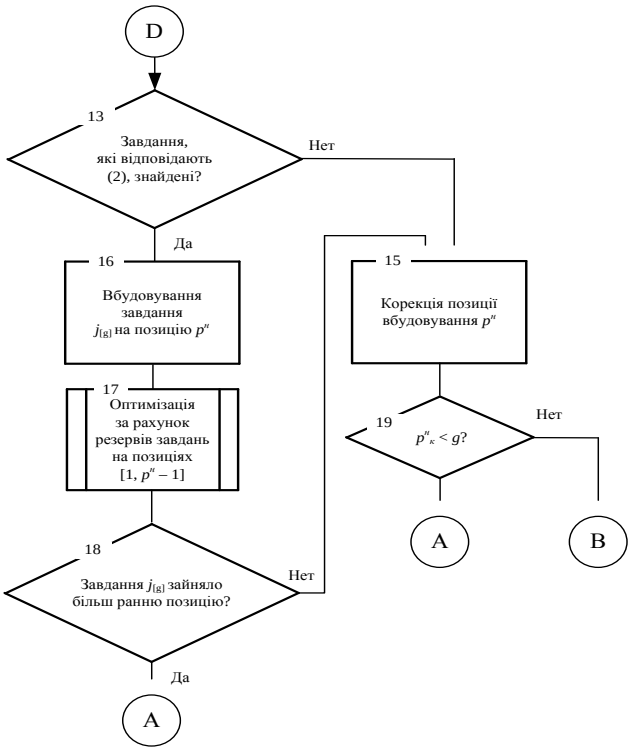
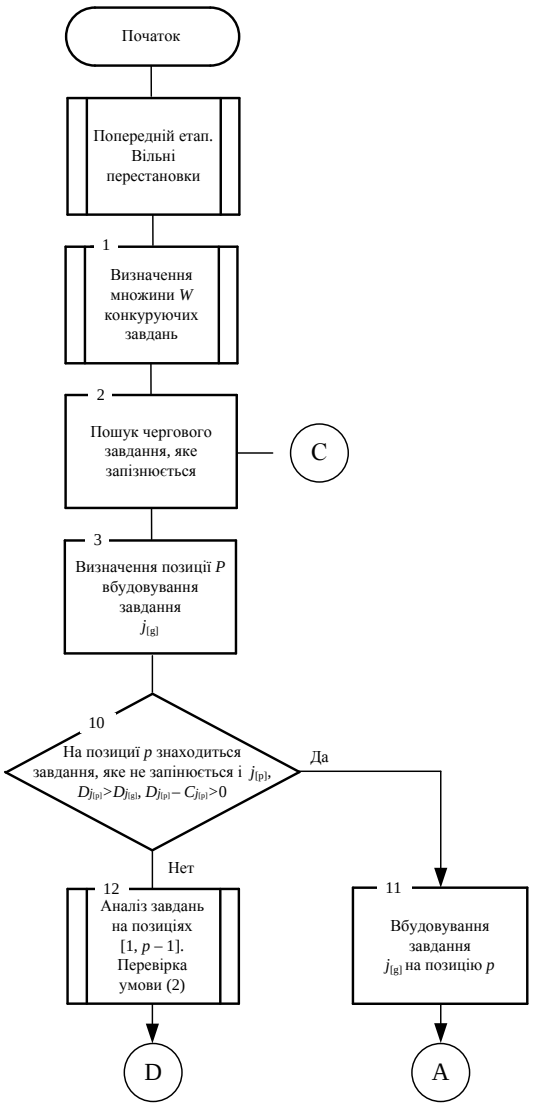


Рис. 1 – Блок-схема алгоритму (ч. 1)



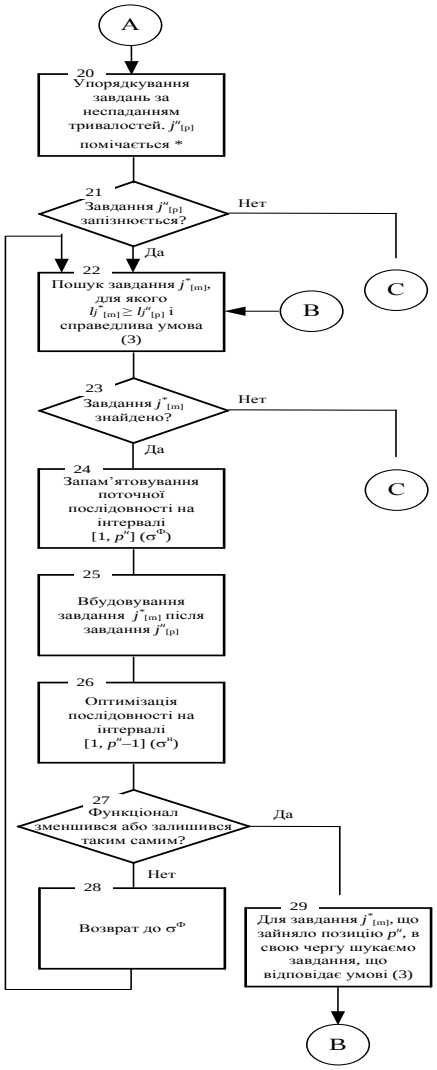


Рис. 2 – Блок-схема алгоритму (ч. 2)

Якщо функціонал збільшився (блок 27), то повертається попередня послідовність (блок 28), перехід до наступної * або **. Якщо таких завдань не знайдено, то перехід до наступного конкуруючого завдання. Якщо ж функціонал зменшився (у блоці 27), то послідовність зберігається, і відбувається пошук * або ** для завдання яке було вбудоване (блок 29).

Нові відсікання

Для покращення загальної швидкодії алгоритму [2] було проведено ряд досліджень за методикою [4]. За результатами цих ДОСліджень було виявлено ряд нових закономірностей та розроблені нові відсікання для існуючого алгоритму, які базуються на наступних твердженнях.

Твердження 1. Якщо конкуруюче завдання $j_{[g]}$ після виконання для нього ітерацій оптимізації зайняло ту ж саму позицію g в поточній послідовності σ^k, то завдання $j_{[i]}$, що слідують за завданням $j_{[g]}$, для яких виконується: $Dj_{[i]} \geq Dj_{[g]}$, $Lj_{[i]} \geq Lj_{[g]}$, в оптимальному розкладі не можуть займати позицію меншу, ніж g. В цьому випадку здійснюється декомпозиція послідовності σ^k на підпослідовності меншого розміру, і якщо позиція вбудовування такого завдання $j_{[i]}$ дорівнює p+1, то для цього завдання не виконуються блоки 12, 13, 17, 18.

Твердження 2. Нехай в послідовності σ, що отримана після виконання вільних перестановок, для всіх завдань на інтервалі [k, n] виконується: $Cj_{[i]} - Dj_{[i]} \geq 0$, i = $\overline{1, n}$. Позначимо цю послідовність $\sigma^{ЗАП}$. Якщо після виконання ітерації оптимізації для завдання $j_{[k]} \in \sigma^{ЗАП}$ це завдання зайняло ту ж саму позицію k, то завдання $j_{[i]}$, $i > k$, для яких виконується $Dj_{[i]} \geq Dj_{[k]}$, вилучаються з множини конкуруючих за резерви попередніх завдань і займають в оптимальному розкладі позиції $r \geq i$.

На даному етапі додано ще два алгоритми для оптимізації алгоритму. Перше відсікання полягає у декомпозиції поточній послідовності. При виконанні умов твердження 1 ряд блоків алгоритму не виконуються, що суттєво скорочує час розв'язання задачі. Друге відсікання полягає у маркуванні завдань, які задовольняють умовам твердження 2, позначками «прапорець» та виключенням їх з ряду конкуруючих завдань. При загальній схемі розв'язання алгоритму, наведеному у [3, 4], всі завдання, які промарковані «прапорцем», не розглядаються як конкуруючи, та вважаються на «своїй» позиції. Вони використовуються лише для спускання «*» або «**». Таким чином, ми отримуємо відсікання дуже великої кількості конкуруючих завдань, що сприяє покращенню швидкодії.

Визначення ефективності нових відсікань

За допомогою системи моделювання для дослідження ефективності ПДС-алгоритму (алгоритму із поліноміальною й експоненційною складовими) [2] було проведено ряд тестів, які показали, що додавання нових відсікань сприяє зменшенню сумарного часу роботи алгоритму у кілька разів, а у деяких класів задач у 70-80 разів. У таблиці 1 приведені деякі результати попереднього тестування версій програмного забезпечення до та після додавання відсікань.

Таблиця 1. Середній час розв'язання задач (сек.)

Розмірність	Середній час розв'язання без додаткових відсікань	Середній час розв'язання з додатковими відсіканнями
50	0,12	0,07
100	2,1	0,5
150	9,4	2,7
300	42,2	12,1
400	114,7	32,8
500	311,7	89,0
1000	1009,8	348,2

Таблиця 2. Порівняння часу розв'язання з відомими методами

№ прикладу	Сумарне запізнення	Час розв'язання відомими методами (мс)	Час розв'язання ПДС-алгоритмом (мс)
46	83	250	52
50	56	150	50
131	178	267	20
134	119	250	58
135	1169	1833	313
136	299	133	80
138	204	250	49
139	69	150	62

Загальна ефективність алгоритму

У таблиці 2 приводиться час розв'язання відомих прикладів [6] розмірністю 500 завдань. Всі випробування проводились на комп'ютері з наступними параметрами: CPU Pentium CORE 2 Duo 1.9ГГц; RAM 2048Мб; OS Microsoft Windows Vista.

Висновок

Розроблено нові правила відсікань до алгоритму, зроблено оптимізацію програмної реалізації, перебудова деяких блоків алгоритму для зменшення сумарного часу обчислення. Створена програмна реалізація функції декомпозиції на задачах великої розмірності, що дає змогу у майбутньому використовувати її для розгалужування алгоритму на багатопроцесорні системи. Планується зробити ще ряд відсікань та розбиття вхідних задач на підкласи, що дозволить запускати ту чи іншу гілку алгоритму, більш пристосовану для даної задачі. Для підвищення ефективності розв'язання задачі на багатопроцесорних системах у майбутньому планується розпаралелювати декомпозиційну складову алгоритму.

Перелік посилань

1. Павлов А.А., Теленик С.Ф. Информационные технологии и алгоритмизация в управлении.– К.: Техника.– 2002.– 344 с., ISBN 966-575-045-3.
2. Система моделювання для дослідження ефективності ПДС-алгоритму задачі мінімізації сумарного запізнення виконання завдань // Павлов А.А., Місюра О.Б., Халус О.А, Беньковський С.Б., Лисецький Т.М., Костик Д.Ю. / Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка. К.: "ВЕК+", 2007.– №47.– С.215-220
3. Павлов О.А., Місюра О.Б., Халус О.А. Модифікований ефективний ПДС-алгоритм рішення задачі мінімізації сумарного запізнення виконання незалежних завдань одним приладом / Радиоэлектроника и информатика.– Харьков: ХНУР, 2007.– №1(36).– С.21-23
4. Павлов А.А., Мисюра Е.Б. Эффективный точный ПДС-алгоритм решения задачи о суммарном запаздывании для одного прибора // Системні дослідження та інформаційні технології. – 2004.– №4.– С.30-59
5. Шейко В., Кушнаренко Н. Організація та методика науково-дослідницької діяльності: Підручник. – К.: Знання-Прес, 2002. – 295 с.
6. Fisher, M.L. (1976) A dual algorithm for the one machine scheduling problem. Mathematical Programming, 11, 229-251. Тестові задачі доступні в Інтернет на сторінці http://www.bilkent.edu.tr/~bkara.