УДК 519.854.2

ВАЩУК Ф.Г.,
ПАВЛОВ О. А.,
МІСЮРА О. Б.,
МЕЛЬНИК О.О.

СКЛАДАННЯ РОЗКЛАДІВ СУМАРНОГО ВИПЕРЕДЖЕННЯ І ЗАПІЗНЕННЯ ІЗ НАЛАГОДЖЕННЯМИ, ЩО ЗАЛЕЖАТЬ ВІД ПОСЛІДОВНОСТІ

Розглядається задача складання розкладів за критерієм мінімізації сумарного випередження і запізнення відносно директивних строків при виконанні незалежних завдань одним приладом (МВЗ) при наявності налагоджень. Запропоновано евристичний алгоритм пошуку локального оптимального розв'язку задачи, що розглядається. Експериментальні дослідження показали ефективність розробленого алгоритму, що дозволяє розв'язувати задачи великої розмірності за короткий час.

We consider the one machine scheduling problem of minimizing the total earliness and tardiness of independent tasks against due dates (ET) with setups. We propose a heuristic algorithm to search a local optimal solution of the problem. Experimental studies have shown the effectiveness of the algorithm, which allows to solve large dimensional problems for a short time.

Вступ

Складанню розкладів відносно директивних строків приділяється значна увага в літературі. Одна з причин цього явища – зростаючий тиск конкуренції на міжнародних ринках: фірми повинні запропонувати велику розмаїтість різних та індивідуальних виробів, у той час як клієнти очікують, що замовлені товари будуть поставлені вчасно. Принцип виробництва «точно в строк» установлює, що необхідна кількість товарів повинна бути вироблена або поставлена точно в заданий час. Виконання роботи з випередженням приводить до витрат на складування, у той час як запізнення робіт – до штрафів і, в остаточному підсумку, втраті доброзичливості клієнтів і репутації фірми.

Постановка задачі. Множина з n незалежних завдань $J = \{j_1, j_2, ..., j_n\}$ повинна бути призначена на виконання без переривань на одному приладі, який може працювати не більш, ніж з одним завданням одночасно. Прилад і завдання передбачаються безупинно доступними з моменту часу нуль, а простої приладу не допускаються. Завдання j, де $j = 1, 2, ..., n$, вимагає часу виконання p_j і в ідеалі повинне бути закінчене у свій директивний строк d_j. Для окремих завдань задано час налагодження s_{ij}, це означає, що в розкладі, у якому завдання j виконується відразу після завдання i, повинен бути час налагодження s_{ij} одиниць часу між моментом завершення завдання i, позначеним через C_i, і часом початку завдання j, що дорівнює $C_j - p_j$. Протягом цього періоду налагодження ніяке інше завдання не може бути виконано приладом. Часи налагодження є *залежними від послідовності*, тому що вони залежать як від i, так і від j. Для будь-якого заданого розкладу випередження та запізнення завдання j можуть бути визначені виразами (2) і (3). Ціль полягає в тім, щоб знайти розклад, який мінімізує сумарне випередження і запізнення всіх завдань:

$$\sum_{j=1}^{n}\left(E_j + T_j\right),$$

де випередження і запізнення визначаються як, відповідно,

$$E_j = \max(0, d_j - C_j) = (d_j - C_j)^+,$$
$$T_j = \max(0, C_j - d_j) = (C_j - d_j)^+.$$

Ця задача (позначимо її МВЗН) узагальнює задачу МВЗ і відноситься до класу *NP*-повних. В [1] розглядається актуальність задачі та приводиться огляд літератури по складанню розкладів «точно в строк» і розкладів з налагодженнями. Представлено алгоритм віток і границь і ефективна евристична процедура одержання локального оптимуму.

Алгоритми локального пошуку широко використовуються як практичний підхід для розв'язання задач комбінаторної оптимізації. Починаючи із припустимого розв'язку, ці алгоритми ітеративно пробують поліпшити поточний розв'язок, вишукуючи кращий розв'язок в околиці поточного розв'язку, поки не знайдений локальний оптимум. Ефективність цих алгоритмів якісно залежить від визначення околиці: при більших околицях якість розв'язку краще, але час обчислень більше. Тому на прак-

тиці більші околиці не корисні, якщо тільки вони не можуть ефективно досліджуватися.

В основу розв'язання задачі покладено новий підхід до розв'язання задачі МВЗ, викладений у [2, 3]. У даній статті ми розширюємо цей підхід для більш загальної та більш практичної задачі, що з'являється особливо у виробничих задачах складання розкладів – задачі із часами налагодження.

Алгоритм розв'язання задачі мвзн

Позначимо $r_j = \max(0, d_j - C_j)$ – резерв завдання j. Очевидно, $r_j = E_j$.

Алгоритм, що представляється, складається із двох етапів. На першому етапі (блоки 1 і 2) налагодження не враховуються. У блоці 1 розв'язується задача мінімізації сумарного запізнення при виконанні незалежних завдань одним приладом МСЗ [3]. Алгоритм побудовано на перестановках і полягає в оптимальному використанні завданнями, що запізнюються, резервів завдань, що не запізнюються. Таким чином, реалізується зменшення сумарного запізнення за рахунок зменшення сумарного випередження. У блоці 2 здійснюється зменшення значення сумарного випередження і запізнення за допомогою послідовного збільшення моментів початку виконання завдань.

Позначимо $r_{\min} = \min\{r_j\}$; N_r – число завдань з резервами ($r_j > 0$); $N_з$ – число завдань, що запізнюються (в їх число включаються завдання з нульовим резервом).

Твердження [4] (використання резервів завдань, що випереджають). Якщо в послідовності σ виконується $N_r > N_з$, то при збільшенні початку виконання завдань на величину, рівну $r_{\min}$, значення функціонала випередження/запізнення зменшується на величину $(N_r - N_з)\, r_{\min}$.

Розглядається послідовність, що отримана в результаті розв'язання задачі МСЗ. На кожній ітерації за умови $N_r \geq N_з$ збільшуються моменти початку виконання завдань у поточній послідовності на $r_{\min}$. Такі процедури виконуються, поки не виконається умова $N_r < N_з$. Отриману послідовність позначаємо σ^R.

На другому етапі здійснюється оптимізація послідовності σ^R з урахуванням налагоджень приладу, що залежать від послідовності, за допомогою реалізації процедури локального пошуку ефективного розв'язання поставленої задачі. Аналізується послідовність σ^R. Для тих завдань, для яких задані налагодження приладу, включаємо ці налагодження у тривалості виконання завдань. Отриману послідовність позначаємо σ^{R1}.

Використовуються наступні типи перестановок: API (adjacent pairwise interchange – суміжна попарна перестановка), NAPI (nonadjacent pairwise interchange – несуміжна попарна перестановка), EBSR (extraction and backward-shifted reinsertion – витяг і повторна вставка зі зрушенням назад) и EFSR (extraction and forward-shifted reinsertion – витяг і повторна вставка зі зрушенням уперед), які діють на послідовність σ таким чином. Нехай задана пара індексів $i < j$, така, що $\sigma = \rho\, i\, \pi\, j\, \omega$.

API(i, j): $\rho\, i\, j\, \omega \Rightarrow \rho\, j\, i\, \omega$ (потрібне $\pi = \varnothing$);

NAPI(i, j): $\rho\, i\, \pi\, j\, \omega \Rightarrow \rho\, j\, \pi\, i\, \omega$;

EBSR(i, j): $\rho\, i\, \pi\, j\, \omega \Rightarrow \rho\, j\, i\, \pi\, \omega$;

EFSR(i, j): $\rho\, i\, \pi\, j\, \omega \Rightarrow \rho\, \pi\, j\, i\, \omega$.

Опис етапу 2 алгоритму

1. У послідовності σ^{R1} знаходимо завдання j, виконання якого вимагає максимального часу налагодження приладу: s_{kj}, де завдання k займає позицію, що безпосередньо передує завданню j.
2. Знаходимо в σ^{R1} завдання i, для яких $s_{ij} < s_{kj}$. Організуємо список S цих завдань, упорядкований за неспаданням значень часу налагодження.
3. Будуємо нові поточні послідовності завдань таким чином. Завдання j з позиції k+1 за допомогою виконання відповідного оператора переносимо на позицію i+1, на якій налагодження для завдання j мінімальне. У зв'язку зі зміною послідовності змінюємо значення налагоджень: налагодження завдання j на новій позиції і налагодження завдання, що безпосередньо слідує за новою позицією завдання j. Перераховуємо часи виконання завдань і визначаємо значення функціонала. Аналогічно будуємо наступну поточну послідовність, переставляючи в послідовності σ^{R1} завдання j після наступного завдання зі списку S і виконуючи всі процедури, описані в п. 3. Така побудова поточної послідовності виконується, поки не буде переглянутий весь список можливих позицій перестановки завдання j.
4. Із всіх побудованих поточних послідовностей вибираємо послідовність із найменшим значенням функціонала, у цій послідовності завдання j позначається зірочкою з метою виключення його з подальшого розгляду, позначимо цю послідовність σ^{R2}, перехід на

п. 1, розглядаючи в якості σ^{R1} послідовність σ^{R2}. У послідовності σ^{R2} знаходимо непомічене зірочкою завдання *j*, виконання якого вимагає максимального часу налагодження приладу, і виконуємо кроки 1–3. Якщо в черговій поточній послідовності відсутні непомічені завдання, що вимагають налагодження, то алгоритм закінчує роботу.

Обчислювальні результати

Алгоритм був закодований мовою C# у середовищі розробки Visual Studio 2010 під бібліотеку Microsoft .NET 4.0. Випробування проводилися на персональному комп'ютері із процесором Pentium CORE 2 Duo 2.0 ГГц із оперативною пам'яттю 2 Гбайта під управлінням ОС Microsoft Windows Vista. Досліджувалися задачі розмірності до 500 завдань.

Для визначення ефективності алгоритму були проведені дослідження залежності часу розв'язання задачі від загального числа завдань і кількості завдань, що вимагають налагодження приладу, заданого у відсотковому відношенні до загального числа завдань.

Схема генерації даних, запропонована Фішером [5], використовувалася для тестування алгоритму на різних типах прикладів, тип задачі визначається комбінацією фактора запізнення *T* і діапазону директивних строків *R*. Для кожної задачі спочатку генеруються тривалості виконання і часи налагоджень із рівномірного розподілу із заданими границями. Потім обчислюються директивні строки з розподілу, рівномірного на $\left[p^*(1-T-R/2), p^*(1-T+R/2)\right]$, де p^* – сума всіх тривалостей. Значення *T* і *R* вибираються з множин {0.2, 0.4, 0.6, 0.8} і {0.2, 0.4, 0.6, 0.8, l.0}, відповідно, даючи по 20 задач кожного типу.

Результати, що наведені в табл. 1, дані для випадку $T = 0{,}4$; $R = 0{,}8$. Найбільш складні задачі виникають в області $T = 0{,}6$; $R = 0{,}9$.

Табл. 1. Середній час розв'язання задач (мс)

Розмірність	Кількість завдань, що потрібують налагодження, %									
	10	20	30	40	50	60	70	80	90	100
50	32	32	33	33	33	34	34	34	35	35
100	103	104	105	106	107	109	110	111	113	114
150	181	183	184	186	188	189	191	193	195	197
200	233	235	236	237	239	241	242	244	246	248
250	268	270	273	276	279	282	286	289	293	297
300	326	330	335	340	346	352	358	365	371	378
350	338	349	361	374	387	401	416	431	448	464
400	444	453	464	475	487	499	512	526	540	555
450	476	491	508	525	544	562	583	604	626	649
500	542	560	580	600	622	644	669	693	720	746

Висновки

Ми представили алгоритм локального пошуку розв'язку задачі МВЗН, заснований на методі розв'язання задачі МВЗ, викладеному в [2, 3]. У нашій евристиці на першому етапі будується послідовність, у якій всі завдання виконуються максимально близько до директивних строків без урахування налагоджень. На етапі 2 задача розв'язується за допомогою операторів перестановок. Дослідження показали, що наш алгоритм дозволяє за прийнятний час ефективно розв'язувати задачі великої розмірності. Майбутні дослідження будуть спрямовані на вивчення поводження алгоритму при збільшенні числа завдань і різних значеннях *R* і *T* та на побудову відсікань, що скорочують час розв'язання.

Список літератури

1. F. Sourd. Earliness-tardiness scheduling with setup considerations / Computers & Operations Research № 32 (7). – 2005, – P.1849-1865.
2. Павлов О.А., Місюра О.Б., Мельников О.В. Дослідження властивостей та розв'язання задачі «Мінімізація сумарного штрафу як за випередження, так і за запізнення відносно директивних строків при виконанні незалежних завдань одним приладом» / Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка. К.: "ВЕК+", 2008.– №48.– С.3-6.
3. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка, – 2010.– 573 с.
4. Згуровский М.З., Павлов А.А., Мисюра Е.Б. ПДС-алгоритмы и труднорешаемые задачи комбинаторной оптимизации // Системні дослідження та інформаційні технології.– 2009.– №.4. – С.14-31.
5. Fisher M.L. A dual algorithm for the one-machine scheduling problem / Math. Programming 11. – 1976, – P.229-251.