УДК 519.854

С.В. Мінухін, Д.С. Лєнько, М.І. Сухонос

Харківський національний економічний університет, Харків

ДОСЛІДЖЕННЯ АЛГОРИТМІВ МІНІМІЗАЦІЇ СУМАРНОГО ЧАСУ ЗАПІЗНЮВАННЯ ЗАВДАНЬ З ДИРЕКТИВНИМИ СТРОКАМИ ВИКОНАННЯ НА ОСНОВІ РАНГОВОГО ПІДХОДУ

Розглянуто та проаналізовано методи вирішення задачі мінімізації сумарного часу запізнювання завдань з директивними строками виконання на обчислювальному ресурсі. Досліджено евристичні алгоритми вирішення задачі на основі рангового підходу. Проведені обчислювальні експерименти щодо обґрунтування ефективності досліджуваних алгоритмів на основі розрахунку відносної похибки та часу реалізації для різної кількості завдань вхідної черги . Доведена можливість використання алгоритмів в системах реального часу.

Ключові слова: *ресурс, гамільтонов шлях, оптимальний розклад, час запізнювання, директивний строк, похибка, часова складність .*

Вступ

Сучасні інформаційно-комунікаційні системи визначаються наявністю багатої кількості різних типів ресурсів – інформаційних обчислювальних, тощо, їх розподіленостю, управління якими потребує підвищення ефективності та вдосконалення систем управління розподіленим обробленням, зокрема, на локальних ресурсах**.** Одним зі завдань є розроблення ефективних алгоритмів планування ресурсів та оптимізації виконання робіт згідно з обраними критеріями. Особливістю розподілених систем є використання обчислювальних ресурсів, серед яких виділимо кластеризовані та некластеризовані, що характеризують різні підходи до їх об'єднання для організації високопродуктивних систем.

Мета даної роботи – дослідження евристичних алгоритмів побудови оптимального розкладу виконання завдань з директивними строками на обчислювальному ресурсі за критерієм мінімізації сумарного часу запізнювання на основі рангового підходу [1, 2].

Постановка задачі

У сучасних розподілених системах, що використовують у якості ресурсів кластери, процесорні елементи, робочі станції (наприклад, Грід-системах [1]), тощо, завдання, які поступили для вирішення на якийсь ресурс, мають різні вартості їх реалізації та директивні строки виконання. Часові відхилення від директивних строків приводять до подорожчання процесів проведення обчислень (штрафів) в обчислювальній системі та порушення умов виконання завдань. Тому представляється актуальним розроблення ефективних алгоритмів, які повинні відповідати вимогам щодо оперативності (часової складності) їх реалізації та адекватності з точки зору мінімізації похибки відносно точного рішення.

Однією з таких задач є побудова оптимального розкладу виконання завдань на одиночному ресурсі (пристрої) обчислювальної системи з метою мінімізації сумарного часу запізнювання виконання усіх завдань, що поступають безперервно, одночасно та згідно з обраною стратегією їх вирішення на ресурсі.

Змістовно постановка такої задачі для визначення оптимальної послідовності виконання завдань (розкладу) з мінімальним сумарним часом запізнювання формулюється наступним чином. Припустимо, що на одиночний ресурс поступила множина незалежних робіт $J = \{j_1, j_2, ..., j_n\}$, кожна з яких безперервно виконуватиметься на ньому. При цьому відомі тривалість виконання кожного завдання t_j та директивний строк його виконання D_j. Необхідно визначити такий порядок (послідовність) виконання усіх завдань, які одночасно поступають на ресурс (пристрій), який мінімізує сумарний час запізнювання усіх завдань вхідної черги.

Формалізація наведеної постановки у вигляді математичної моделі виглядає так: необхідно побудувати розклад виконання завдань, який мінімізує функцію

$$f = \sum_{j=1}^{n} \max(0, C_j - D_j) \to \min, \qquad (1)$$

де C_j – фактичний момент завершення завдання j; D_j – директивний строк виконання завдання; n – кількість завдань вхідного потоку.

Поставлена задача є відомою задачею, основні дослідження та отримані в них результати щодо складності її розв'язання у синтезованому вигляді наведено в табл. 1. Основні результати та методи рішення цієї задачі приведені в роботах [3 – 9]. Згідно з роботами [3, 5, 8, 9], ця задача відноситься до NP-складних задач. В роботі [10] розроблено ефективний точний алгоритм та наведений приклад його практичної реалізації для кількості робіт 150. У роботі [4] для вирішення даної задачі розглянуті алгоритми динамічного програмування. Недоліком пе-

релічених методів є те, що рішення цієї задачі в реальному часі затруднене внаслідок експоненціального зростання кількості аналізованих варіантів зі зростанням кількості робіт та, відповідно, й часової складності алгоритмів.

Відмітимо, що поряд з часовою складністю алгоритмів у якості метрик продуктивності використовуються такі характеристики, як кількість робіт вхідного потоку, для якого за мінімальний час побудована оптимальна послідовність виконання робіт.

Таблиця 1

Порівняльний аналіз методів та алгоритмів розв'язання задачі мінімізації середнього сумарного часу запізнювання

Метод	Складність	Автори
Декомпозиція / динамічне програмування	$O(n^4 \sum p_i)$	Лоулер
Декомпозиція / динамічне програмування	50 робіт	Бейкер, Шраге
Декомпозиція / динамічне програмування	$O(n^7/\varepsilon)$	Лоулер
Динамічне програмування	$\leq$ 50 робіт	Поттс и Ван Вассенхове
Вдосконалені межі	$O(n^6 \log n + n^6/\varepsilon)$	Ковальов М.Ю.
Парне-непарне розбиття	$O(n \sum p_i)$ для окремого випадку	Лазарєв О.О.
Процедура округлення та вдосконалених меж	$O(n^5 \log n + n^5/\varepsilon)$	Коуламас

Формалізація задачі мінімізації сумарного часу виконання завдань з директивними строками на обчислювальному ресурсі

Для формалізації моделі та методу вирішення приведеної задачі розглянемо граф *G*, в якому кожній вершині відповідає завдання, а ребро (*i,j*) характеризується ваговою характеристикою:

$$T_j = \max(0, C_j - D_j) \text{ (рис. 1).}$$

Тоді задачу мінімізації функціонала (1) можна розглядати як задачу визначення такого обходу всіх вершин графа *G*, для якого довжина побудованого при цьому шляху була мінімальною.

Така постановка приводить до задачі визначення найкоротшого гамільтонового шляху в графі G.

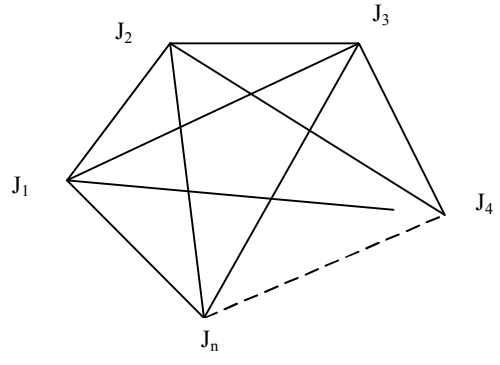

Рис. 1. Граф G

Особливістю даної задачі є той факт, що ваги ребер графа, що витікають із довільної вершини J_i , не є постійними величинами, а залежать від того, за яким порядком пройдені ті вершини, що ведуть до вершини J_i, оскільки від цього залежить величина C_j, яка визначається сумою тривалостей робіт, які попереджують виконання роботи J_i, тобто значенням

$$C_j = \sum_{l_i \in \mu_{sj}} l_i \, .$$

Незважаючи на наявність часткових результатів, що відносяться до спеціальних класів графів, в загальному випадку задача визначення гамільтонових шляхів є недостатньо вивченою. Так, наприклад, немає ефективної процедури знаходження гамільтонового шляху в довільному графі. Критерії існування гамільтонових шляхів в графах, що приведені в роботах Поша, Неша-Уїльямса [11], є загальними і не придатними для довільних графів, які часто зустрічаються на практиці. Не менш важливим, з погляду практичного застосування алгоритмів, що розробляються, є дослідження можливості розроблення паралельних алгоритмів розв'язання задачі пошуку найкоротшого гамільтонового шляху з використанням технологій паралельного програмування та оброблення інформації в багатопроцесорних системах. Нехай всі можливі стани деякої системи визначаються графом G(V,E) з n вершинами, де вершини відповідають можливим станам системи.

Перейдемо до простору з $n \cdot (n-1)$ станами. Для цього кожному з n станів поставимо у відповідність ще (n-1) стан, що характеризує спосіб досягнення наступного стану, який складається зі множини {1, 2, ..., n}. При цьому у якості станів, що додаються, визначимо поняття рангу шляху в графі G(V,E). Це означає, що з вершини s графа G(V,E) в деяку довільну вершину s_j можна потрапити шляхом рангу r = 1, використовуючи одне ребро, шляхом рангу r = 2 – використовуючи 2 ребра і так далі, й шляхом рангу r = n – 1 – використовуючи (n – 1) ребро. Такий простір станів можна представити у вигляді стягнутого дерева всіх шляхів, представленого у вигляді графа D (рис. 2).

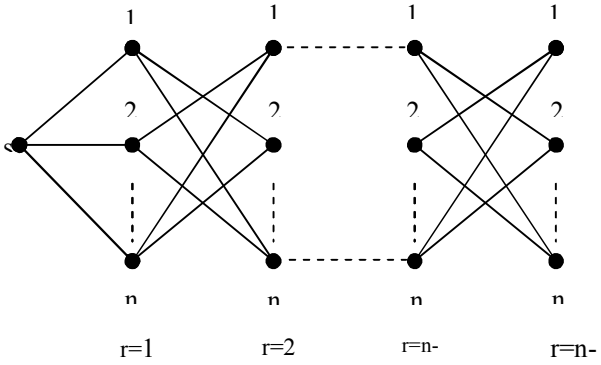

Рис. 2. Стягнуте дерево всіх шляхів графу D

Дерево всіх шляхів D містить n горизонтальних лінійок і (n – 1) ярус. При визначенні шляхів прохо-

дження по кожній горизонтальній лінійці допускається тільки один раз згідно з задачею пошуку найкоротшого гамільтонового шляху. Виходячи із стягнутого дерева шляхів, для довільної вершини s_j безліч шляхів, ведучих в цю вершину з деякої вершини s, можна представити в такому вигляді:

$$m_s(j) = m_{sj}^{r=1} \cup m_{sj}^{r=2} \cup .... \cup m_{sj}^{r=n-1};\ j = \overline{(1, n-1)}. \quad (2)$$

Таким чином, використовуючи граф D і ввівши правила формування шляхів наступного рангу, можна із довільної вершини s поетапно побудувати шляхи довільного рангу аж до рангу r = n – 1. При такому підході до вирішення завдання визначення найкоротшого гамільтонового шляху в графі D потрібно побудувати найкоротші шляхи рангу r=n-1 від вершини 1,2, ..., n до решти всіх вершин графа і потім серед них вибрати найкоротший.

Такий підхід та алгоритми його реалізації запропоновані та досліджені в роботі [2].

В виразі (2) та на рис. 2 $m_{sj}^r = \{\mu_{sj}^r\}$ – підмножини шляхів з довільної вершини s в деяку вершину s_j графа G(V,E) рангу r.

Обчислювальною особливістю даного алгоритму є те, що при виборі наступної вершини у графі значення $c_i - d_i$ для деяких вершин будуть рівні:

$$C_i - D_i = C_j - D_j = \ldots = C_k - D_k,$$

де k – кількість наступних робіт з однаковою мінімальною різницею $C_i - D_i$. Таким чином, це можна використовувати для оптимізації роботи алгоритму за допомогою використання такого домінуючого правила:

$$D_1 < D_2 < \ldots < D_k.$$

Це означає, що у випадку рівних значень різниць $C_i - D_i$ для декількох робіт буде вибиратися та робота, для якої директивний строк виконання є найменшим.

Послідовність проведення комп'ютерного експерименту та аналіз результатів

Розроблений програмний продукт, за допомогою якого проводилося тестування запропонованих алгоритмів, реалізує такі алгоритми пошуку рішення: алгоритм повного перебору, алгоритм оптимізації за напрямком та алгоритм оптимізації за напрямком з використанням домінуючого правила.

Для тестування розробленого програмного забезпечення з метою підвищення рівня адекватності отриманих результатів була використана бібліотека OR-Library [12] та генератор стохастичних послідовностей, що генерував послідовності з різними характеристиками.

Тестові приклади (instances) генеруються наступним чином. Кожній роботі j ($j = \overline{1, N}$) призначується час її оброблення l_j – ціле число, що генерується за рівномірним законом розподілу на інтервалі [1–100]. Для генерації директивних строків використовуються різні рівномірні розподіли з метою моделювання завдань різної складності. Для заданого відносного діапазону директивних строків (range of due dates, RDD) з множини {0.2, 0.4, 0.6, 0.8, 1.0} та заданого середнього фактору запізнення (tardiness factor, TF) з множини {0.2, 0.4, 0.6, 0.8, 1.0} за рівномірним законом розподілу генеруються цілочисельні директивні строки виконання робіт d_j з інтервалу

$$\left[L \cdot \left(1 - TF - \frac{RDD}{2}\right), L \cdot \left(1 - TF + \frac{RDD}{2}\right)\right],\ L = \sum_{j=1}^{N} l_j. \quad (3)$$

Таким чином, для одного спостереження маємо 25 різних завдань.

Для аналізу достатньої кількості завдань з точки зору репрезентативності тестової вибірки згенеровано 1, 2, 3 та 4 спостережень, що включають 25, 50, 75 та 100 завдань відповідно, з кількістю робіт 40, 60, 80, 100, 120, 150. Після побудови розкладу методом оптимізації за напрямком були проведені обчислення математичного очікування та середньоквадратичного відхилення мінімального, максимального та середнього часу запізнювання. Результати розрахунків наведені в табл. 2.

Результати, наведені в табл. 2, обґрунтовують результат, який має практичне значення: для репрезентативності тестової вибірки достатньо генерувати спостереження з кількістю 50 – 75 завдань: математичне очікування середнього часу запізнювання та середньоквадратичне відхилення для 75 завдань (а в деяких випадках – і для 50 завдань) практично не змінюються у порівнянні зі 100 завданнями.

Для завдань з кількістю робіт з 3 по 11 було згенеровано 3 спостереження, що включають 75 завдань. Побудова оптимального розкладу робіт проводилося методами повного перебору та оптимізації за напрямком.

Час роботи алгоритму повного перебору при кількості робіт більше 11 зростає експоненціально, тому для більшої кількості робіт для отримання рішення використовуються евристичні алгоритми.

Для оцінки якості апроксимації евристичних алгоритмів відносно алгоритму повного перебору були використані математичне очікування та середньоквадратичне відхилення відносної похибки за формулами:

$$M_\varepsilon = \sum_{j=1}^{N} \frac{t_j^o - t_j^T}{t_j^T} \Big/ N, \quad (4)$$

$$\sigma_\varepsilon = \sqrt{\frac{1}{N} \sum_{j=1}^{N} \left(\frac{t_j^o - t_j^T}{t_j^T} - M_\varepsilon \right)^2}, \quad (5)$$

де t_j^T, t_j^o – час запізнення j-ої роботи, що отриманий методом повного перебору та методом оптимізації за напрямком відповідно.

Таблиця 2

Результати статистичного аналізу репрезентативності тестової вибірки

Кількість завдань	Математичне сподівання			Середньоквадратичне відхилення		
	максимального часу запізнення	мінімального часу запізнення	середнього часу запізнення	максимального часу запізнення	мінімального часу запізнення	середнього часу запізнення
25 завдань						
40	709,92	32,84	342,187	27,971	549,705	337,105
60	1022,48	30,84	523,672	29,035	822,106	503,169
80	1364,72	29,08	688,694	25,622	1133,877	677,804
100	1686,48	28,76	835,879	26,387	1379,46	821,931
120	2069	24,32	1082,625	28,322	1728,345	1040,906
150	2518,52	35,72	1275,112	30,085	2036,681	1224,168
50 завдань						
40	696,1	35,6	356,743	29,587	557,169	340,691
60	1022,52	30,18	504,036	29,666	805,652	471,755
80	1348,84	27,88	687,72	26,536	1097,985	674,183
100	1718,58	32,96	870,478	31,616	1358,188	817,339
120	2016,38	27,82	1009,328	27,374	1624,131	961,854
150	2561,68	26,42	1309,204	28,265	2125,411	1257,644
75 завдань						
40	719,48	32,373	363,001	28,444	571,983	347,355
60	1072,307	32,067	544,909	28,631	848,556	515,632
80	1385,84	30,627	708,471	28,997	1109,223	668,298
100	1712,6	29,107	867,537	28,434	1374,111	839,027
120	2068,16	33,16	1061,638	28,033	1683,515	1020,554
150	2536,56	28,56	1281,752	28,165	2117,503	1264,267
100 завдань						
40	722,01	31,15	364,285	28,624	566,215	341,192
60	1033,5	27,02	516,133	24,311	815,324	489,944
80	1380,33	29,06	705,098	28,278	1112,458	672,496
100	1743,5	31,65	882,503	28,137	1390,689	840,868
120	2049,1	29,79	1029,385	28,467	1683,822	1007,888
150	2529,23	30,2	1283,289	28,16	2084,723	1248,029

Для оцінки часової ефективності розглянутих алгоритмів було проведено експерименти для порівняльного аналізу часової складності алгоритму повного перебору та двох евристичних алгоритмів оптимізації за напрямком.

Результати проведеного обчислювального експерименту з кількістю робіт в діапазоні з 3 по 11 для алгоритмів повного перебору та оптимізації за напрямком (з використанням та без використання домінуючого правила) наведені в табл. 3.

Для оцінки часової складності роботи досліджуваних алгоритмів було побудовано апроксимуючі функції – для алгоритмів повного перебору (6), оптимізації за напрямком без використання (7) та з використанням домінуючого правила (8), які мають такий вигляд:

Таблиця 3

Результати проведення експерименту для алгоритмів повного перебору та оптимізації за напрямком

Кількість робіт	Час виконання алгоритмом повного перебору, мс	Час виконання алгоритмом оптимізації за напрямком, мс	Час виконання алгоритмом оптимізації за напрямком з використанням домінуючого правила, мс	Математичне сподівання відносної похибки часу запізнювання для методу оптимізації, %	Математичне сподівання відносної похибки часу запізнювання для методу оптимізації з використанням домінуючого правила, %
3	11	1	1	26,69	26,69
4	3	1	1	3,35	2,35
5	12	2	3	11,13	11,79
6	80	4	5	21,51	35
7	525	6	7	25,65	25,5
8	4448	9	10	10,11	8,21
9	40342	13	14	12,51	10,93
10	414706	18	20	27,66	36,77
11	4644922	25	28	3,73	5,23

$$T(N)_{\text{пов.перебор}} = 51820,05n^3 - 944767,87n^2 + \\ +5329583,96n - 9176552,53. \quad (6)$$

$$T(N)_{\text{опт.}} = 0,03n^3 - 0,22n^2 + 1,18n - 2,02. \quad (7)$$

$$T(N)_{\text{опт.D}} = 0,05n^3 - 0,55n^2 + 3,22n - 5,72. \quad (8)$$

Таким чином, проведені експериментальні дослідження запропонованих в даній роботі евристичних та точного алгоритмів побудови оптимального розкладу послідовності робіт цілком відповідають теоретичній отриманій в роботі [2] залежності складності виконання від кількості завдань $O(n^3)$.

На рис. 3, 4 наведені отримані залежності часу виконання робіт методом оптимізації за напрямком та з використанням домінуючого правила, відповідно, на рис. 5 – відносна похибка евристичного алгоритму в зрівнянні з алгоритмом повного перебору.

Експеримент проводився на персональному комп'ютері з двохядерним процесором з тактовою частотою 3,2 ГГц та з оперативною пам'яттю 4 ГБ під управлянням ОС Windows 7. Для забезпечення точності вимірювання часу виконання обчислень експерименти проводилися на одному ядрі процесора.

Аналіз результатів, наведених на рис. 3 – 5, свідчить про те, що математичне очікування похибки досліджуваних алгоритмів відносно точного для кількості 11 робіт в потоці не перевищує 25%, що, враховуючи експоненціальне зростання часової складності алгоритму, є досить прийнятним результатом для його практичного застосування, наприклад, в однопроцесорних та багатопроцесорних обчислювальних системах в реальному часі. Відмітимо, що при кількості робіт до 10 алгоритм з використанням домінуючого правила надає більш точний результат, тобто, має меншу ніж алгоритм оптимізації за напрямком відносну похибку, а алгоритм оптимізації за напрямком в діапазоні 8 – 11 робіт має досить прийнятну з точки зору точності відносну похибку в межах від 3% до 12 %. Результати проведення експерименту з кількістю робіт 20, 40, 60, 80, 100, 120 для алгоритму оптимізації за напрямком (з використанням та без використання домінуючого правила) наведені в табл. 4.

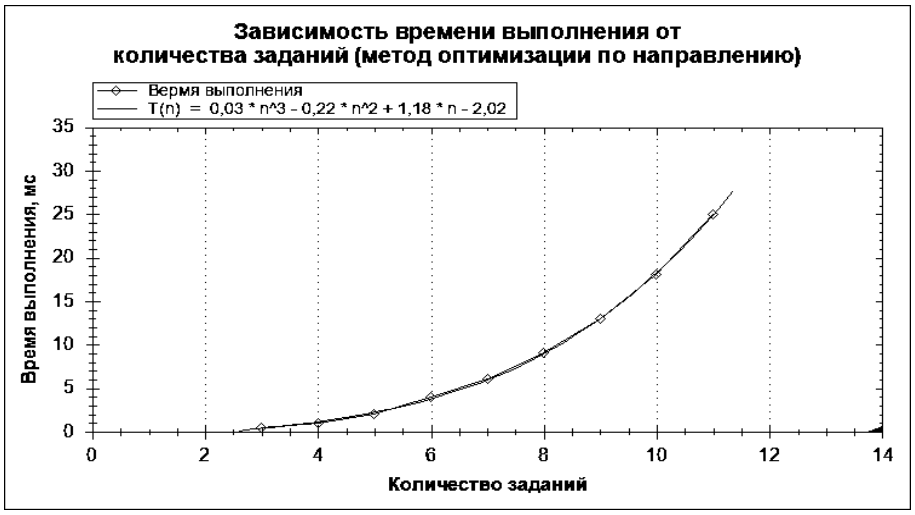

Рис. 3. Залежність часу виконання від кількості робіт для евристичного алгоритму оптимізації

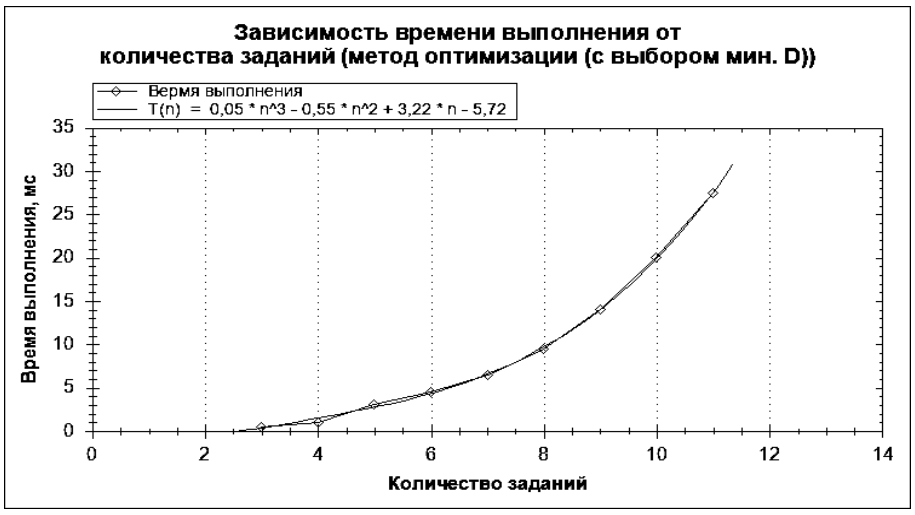

Рис. 4. Залежність часу виконання від кількості робіт евристичного алгоритму оптимізації з використанням домінуючого правила

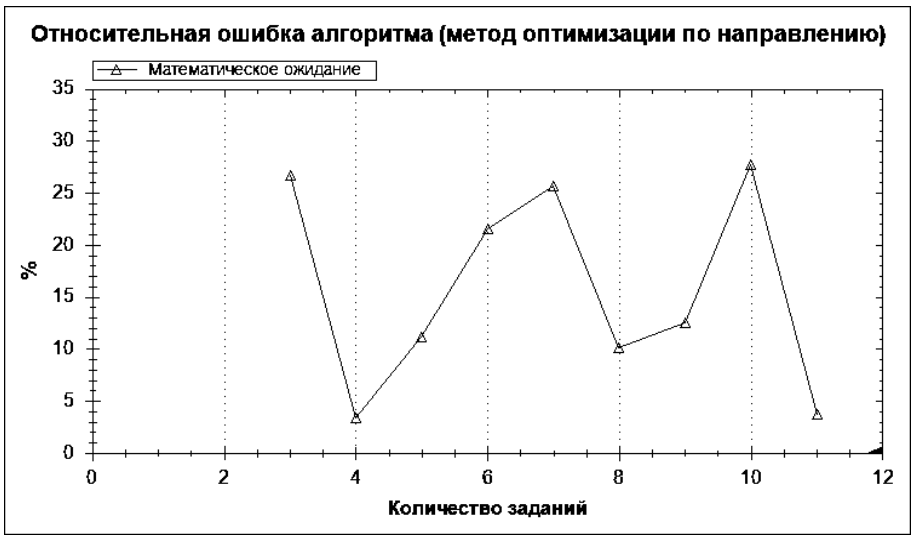

Рис. 5. Відносна похибка евристичного алгоритму оптимізації за напрямком

Таблиця 4

Результати проведення експерименту для алгоритму оптимізації за напрямком та оптимізації за напрямком з використанням домінуючого правила

Кількість робіт	Час виконання алгоритмом оптимізації за напрямком, мс	Час виконання алгоритмом оптимізації за напрямком з використанням домінуючого правила, мс
20	416	455
40	5080	5704
60	31481	37814
80	114175	123917
100	298052	305349
120	655921	668275

Експериментальні залежності часу виконання від кількості робіт апроксимовані методом найменших квадратів – кубічним поліномом, в результаті чого отримані такі функції:

$$T(N)_{опт.} = 1,04n^3 - 110,80n^2 + 4230,16n - 49458,11;$$

$$T(N)_{опт.D} = 1,04n^3 - 111,56n^2 + 4433,82n - 53891,34.$$

Результати розрахунків наведені на рис. 6.

З наведених результатів видно, що час роботи алгоритму з домінуючим правилом декілька більше, але він забезпечує менше значення критерію (1).

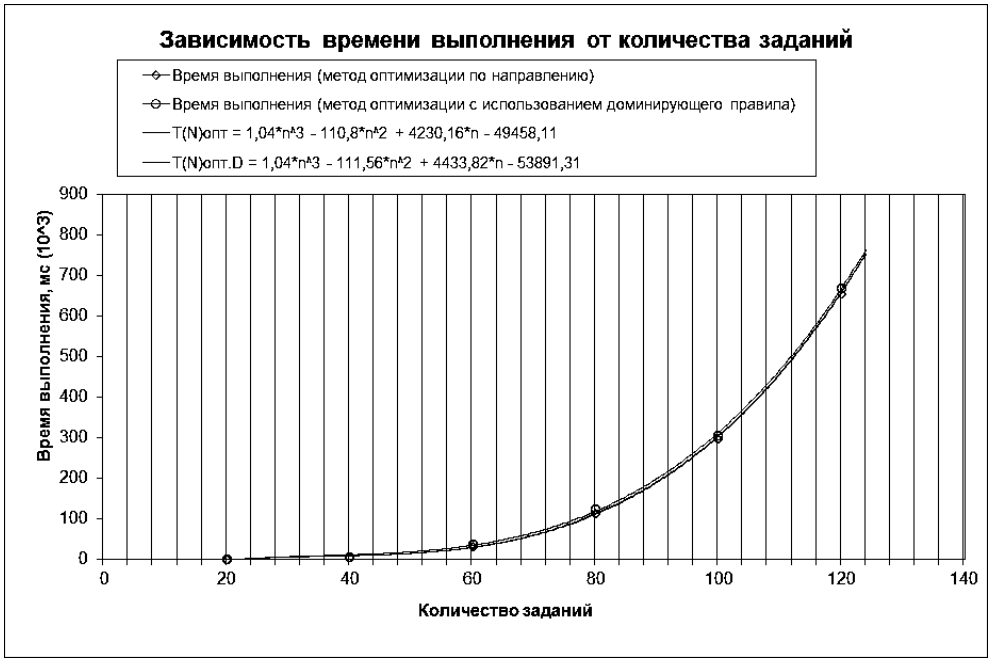

Рис. 6. Залежність часу роботи алгоритмів для 20, 40, 60, 80, 100 та 120 робіт

Висновки

1. Проведений аналіз існуючих методів вирішення задачі мінімізації сумарного часу запізнювання завдань на обчислювальному ресурсі, який показав, що розроблення ефективних алгоритмів з малою часовою складністю для використання в системах реального є актуальним завданням.

3. Розроблено рекомендації щодо визначення репрезентативності тестової вибірки для отримання адекватних результатів обчислювального експерименту.

3. Проведені дослідження евристичного алгоритму, запропонованого в роботі [2], на основі програмної реалізації алгоритму та тестових прикладів бібліотеки OR-Library, які обгрунтували теоретичну часову складність O (n^3).

4. Отримана відносна похибка для евристичних алгоритмів свідчить про можливість їх використання при вирішенні задачі мінімізації сумарного часу запізнювання на одиночному обчислювальному ресурсі.

5. Запропонована модифікація евристичного алгоритму на основі домінуючого правила, використання якого дозволяє зменшити значення критерія (1), а часова складність алгоритму при цьому практично не змінюється.

Список літератури

1. Пономаренко В.С. Методы и модели планирования ресурсов в Grid-системах: монография / В.С. Пономаренко, С.В. Листровой, С.В. Минухин, С.В. Знахур. – Х.: ИД «ИНЖЭК», 2008. – 408 с.

2. Мінухін С.В. Метод мінімізації часу виконання завдань з директивними строками на некластеризованому ресурсі обчислювальної системи / С.В. Мінухін // Інформаційно-керуючі системи на залізничному транспорті. – 2009. – №3. – С. 47 – 53.

3. Lawler E.L. A ''pseudo polynomial'' algorithm for sequencing jobs to minimize total tardiness. //Annals of Discrete Mathematics. – 1977. – № 1. – P. 331 – 342.

4. Baker K.R., Shrage L.E. Finding an optimal sequence by dynamic programming an extension to precedence-related tasks // Oper. Res. – 1978. –№26. – P. 111 – 120.

5. Lawler, E.L. A fully polynomial approximation scheme for the total tardiness problem. //Operations Research Letters. – 1982 – №1. – P. 207 – 208.

6. Potts C.N., Van Wassenhove L.N. Dynamic programming and decomposition approaches for the single machine total tardiness problem. //European Journal of Operational. – 1987. – №32. – P. 405 – 414.

7. Kovalyov M.Y. Improving the complexities of approximation algorithms for optimization problems. //Operations Research Letters. – 1995. –№17. – P. 85 – 87.

8. Koulamas C. A faster fully polynomial approximation scheme for the single machine total tardiness problem //European Journal of Operational Research. – 2009. – № 193. – P. 637 – 638.

9.Lazarev A.A., Gafarov E.R. Special case of the single machine total tardiness problem is NP¡hard.// 12th IFAC Symposium on Information Control Problems in Manufacturing. INCOM'2006.Preprints, Vol. III, Operational Research, May 17–19, 2006 – Saint-Etienne, France. – P. 155 – 157.

10. Павлов А.А., Мисюра Е.Б. Ефективный точный ПДС-алгоритм решения задачи о суммарном запаздывании для одного прибора // Системні дослідження та інформаційні технології. – 2004. – № 4. – С. 30 – 59.

11. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. – Вильямс, 2005. – 1296 с.

12. OR-Library [Електронний ресурс]. – Метод доступу: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/wt-info.html.

Надійшла до редколегії 24.03.2012

Рецензент: д-р техн. наук, проф. С.В. Лістровий, Українська державна академія залізничного транспорту, Харків.

ИССЛЕДОВАНИЕ АЛГОРИТМОВ МИНИМИЗАЦИИ СУММАРНОГО ВРЕМЕНИ ЗАПАЗДЫВАНИЯ ЗАДАЧ С ДИРЕКТИВНЫМИ СРОКАМИ ВЫПОЛНЕНИЯ НА ОСНОВЕ РАНГОВОГО ПОДХОДА

С.В. Минухин, Д.С. Ленько, М.И. Сухонос

Рассмотрены и проанализированы методы решения задачи минимизации суммарного времени запаздывания заданий с директивными сроками выполнения на вычислительном ресурсе. Исследованы эвристические алгоритмы решения задачи на основе рангового подхода. Проведенные вычислительные эксперименты по обоснованию эффективности исследуемых алгоритмов на основе расчета относительной погрешности и времени реализации для разного количества задач входной очереди. Доказана возможность использования алгоритмов в системах реального времени.

***Ключевые слова**: ресурс, гамильтонова путь, оптимальное расписание, время запаздывания, директивный срок, погрешность, временная сложность.*

INVESTIGATION OF ALGORITHMS FOR MINIMIZING THE TOTAL TARDINESS TIME PROBLEM WITH POLICY TERMS OF PERFORMANCE BASED RANK APPROACH

S.V. Minukhin, D.S. Lienko, M.I. Sukhonos

Methods for solving the problem of minimizing the total tardiness time divinity jobs with policy terms to meet the following computing resources are considered and analyzed. Investigated heuristic algorithms for the solution of the ranking based approach. The performed numerical experiments to validate the effectiveness of the algorithms studied by calculating the relative error and time complexity for different realizations of the input queue of jobs. Proved by the use of algorithms in real-time systems.

***Key words**: resource, Hamiltonian path, optimal schedule, the time delay, the due date, an error, the time complexity.*