

## **АЛГОРИТМИ МІНІМІЗАЦІЇ СУМАРНОГО ЗАПІЗНЮВАННЯ РОБІТ НА ОДИНОЧНОМУ ПРИСТРОЇ НА ОСНОВІ ВИЗНАЧЕННЯ НАЙКОРОТШОГО ГАМІЛЬТОНОВОГО ШЛЯХУ В ГРАФІ ТА ПРАВИЛ ДОМІНУВАННЯ**

© Мінухін С., 2013

Запропоновано метод мінімізації сумарного запізнювання на одиночному пристрої на основі визначення найкоротшого гамільтонового шляху в довільному графі та алгоритми його реалізації з використанням правил домінування, які покращують їх ефективність та не зменшують час виконання. Запропоновано метрики для оцінки ефективності використання правил домінування. Наведені результати експериментальних досліджень алгоритмів, які обґрунтовують ефективність запропонованої модифікації за рахунок отримання локальних оптимальних рішень.

**Ключові слова:** гамільтонів шлях, граф, оптимальний розклад, сумарне запізнювання, директивний строк, вага, правило домінування.

The paper proposes a method, algorithms and its implementations using dominance rules for minimizing the total tardiness on a single device-based on shortest Hamiltonian path in a arbitrary graph that improve the efficiency and not reduce the execution time. Metrics for evaluating the effectiveness of the dominance rules are proposed. The experimental results of algorithms are developed that justify the effectiveness of the proposed modifications by getting local optimal solutions during procedure.

**Key words:** Hamiltonian path, graph, the optimal schedule, total tardiness, due date, weight, dominance rule.

### **Вступ**

Сучасні інформаційно-комунікаційні системи визначаються наявністю багатої кількості різних типів ресурсів – інформаційних, обчислювальних тощо, їх розподіленістю, управління якими потребує підвищення ефективності та вдосконалення систем управління розподіленим обробленням, зокрема на локальних ресурсах. Одним із завдань є розроблення ефективних алгоритмів планування розкладу та оптимізації виконання робіт за обраними критеріями в системах реального часу на обчислювальних пристроях (наприклад, вузлах Грід-систем, робочих станціях (Network of workstations, desktop Grids), тощо). Значний інтерес становить виконання завдання оптимізації роботи локального ресурсу таких систем з метою зменшення, а, можливо, й взагалі уникнення запізнювання в випадках, коли завдання характеризується директивним строком виконання. Це дозволить зменшити ризики, пов'язані з можливістю плати штрафних санкцій, які призводить до економічних збитків користувачів. У роботах [1–4] розглянуто огляди розв'язання задачі мінімізації сумарного запізнювання, основні методи її розв'язання – в [5–11]. У статті [12] запропоновано метод мінімізації сумарного часу запізнювання (метод оптимізації за напрямком) на основі побудови найкоротшого гамільтонового шляху в повнозв'язному графі та на основі

рангового підходу з часовою складністю  $O(n^3)$  ( $n$  – кількість вершин в графі), в [13, 14] – висунута гіпотеза про те, що в випадках, коли існує декілька альтернативних шляхів, треба обирати такий, в якому вершина, що додається на поточному ранзі до гамільтонового шляху, повинна мати менший директивний строк.

Мета статті – розроблення та дослідження алгоритмів побудови оптимального розкладу виконання завдань для мінімізації сумарного часу запізнювання на одиночному обчислювальному пристрої на основі побудови найкоротшого гамільтонового шляху в довільному графі та правил домінування і метрик оцінки ефективності їх використання.

### Постановка задачі

Припустимо, що на одиночний ресурс надходить множина незалежних робіт  $J = \{J_1, J_2, \dots, J_n\}$ , кожна з яких безперервно виконуватиметься на ньому. При цьому відомі тривалість виконання кожного завдання  $L_j$  та директивний строк його виконання  $D_j$ . Необхідно визначити такий порядок (послідовність) виконання усіх завдань, які одночасно надходять на ресурс (пристрій), який мінімізує сумарний час запізнювання усіх завдань вхідної черги  $TT_S = \sum_{j \in S} \max(0, C_j - d_j)$ , де  $C_j$  – час завершення роботи  $j$ . Така задача визначається  $1||\sum T_i$ , для

розв'язання якої в [12, 13] отримані часова складність та вперше запропоновано для отримання локального оптимуму використовувати правила домінування, в якості котрого запропоновано правило EDD (Earliest Due Date). Для обґрунтування гіпотези та експериментального її доведення розглянемо декілька прикладів, що ілюструють роботу процедури на основі довільного повнозв'язного графа з вершинами, які визначають роботи тестової послідовності, суть методу та використання правил домінування.

Припустимо, що є множина робіт  $J_1(4, 7)$ ,  $J_2(5, 6)$ ,  $J_3(3, 7)$ ,  $J_4(3, 4)$ , для якої потрібно побудувати оптимальний розклад. Кожна робота  $J$  характеризується двома параметрами:  $L$  – тривалість роботи,  $d$  – директивний строк, й таким чином представляється як  $J(L, d)$ .

Граф, який відбиває цю послідовність робіт, зображений на рис. 1.

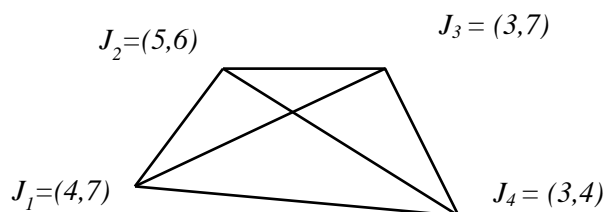


Рис. 1. Повнозв'язний граф з чотирьох вершин (робіт)

### Алгоритми мінімізації сумарного часу запізнювання з правилами домінування

Для пояснення роботи алгоритмів використовуємо матрицю, в якій стовпцям відповідають ранги (шари), а рядкам – лінії, кількість яких визначається кількістю робіт, для яких треба побудувати розклад (графік) виконання (див. табл. 1). Щоб охарактеризувати значення поточного запізнювання множини робіт на кожному ранзі, використовується позначення  $T(j) j_1 j_2 \dots j_n$ , де  $J$  – кількість рангів (стовпців),  $j_1 j_2 \dots j_n$  – кількість вершин (робіт), які входять до поточного шляху до будь-якої вершини (роботи) на поточному ранзі (стовпці) матриці. Отже, загальне (сумарне) запізнювання всіх робіт буде визначатися на останньому ранзі. На початку процедури роботи не впорядковані, запізнювання кожної з них дорівнює 0, і, отже, їх сумарне запізнювання дорівнює 0.

На першій горизонталі (в першому рядку) матриці будуюмо всі шляхи від робіт  $J_2, J_3, J_4$  до роботи  $J_1$ , на другому рядку – шляхи від робіт  $J_1, J_3, J_4$  до роботи  $J_2$ , в третьому рядку – шляхи від робіт  $J_1, J_2, J_4$  до роботи  $J_3$ , в четвертому рядку – шляхи від робіт  $J_1, J_2, J_3$  до роботи  $J_4$ . Далі відповідно до алгоритму необхідно для кожного рядка (лінії) вибрати мінімальні шляхи: для рядка 1 –  $T(2)_{3,1} = 3 + (4 - 7) = 0$  і  $T(2)_{4,1} = 3 + (4 - 7) = 0$ ; для рядка 2 –  $T(2)_{3,2} = 3 + (5 - 6) = 2$  та  $T(2)_{4,2} = 3 + (5 - 6) = 2$ ; для рядка 3 –  $T(2)_{4,3} = 3 + (3 - 7) = -1$ ; для рядка 4 –  $T(2)_{3,4} = 3 + (3 - 4) = 2$ .

Після цього на ранзі 3 вибираємо: в першому рядку –  $T(3)_{4,3,1} = 6 + (4 - 7) = 3$  та  $T(3)_{3,4,1} = 6 + (4 - 7) = 3$ ; в другому рядку –  $T(3)_{4,3,2} = 6 + (5 - 6) = 5$  та  $T(3)_{3,4,2} = 6 + (5 - 6) = 5$ ; в третьому рядку –  $T(3)_{4,1,3} = 7 + (3 - 7) = 3$ ; в четвертому рядку –  $T(3)_{3,1,4} = 7 + (3 - 4) = 6$ . Далі будуюмо шляхи з вершин рангу 3 до вершин рангу 4: для рядка 1 –  $TT_{4,3,2,1} = 0 + 0 + 5 + 8 = 13$ ; для рядка 2 –  $TT_{4,3,1,2} = 0 + 0 + 3 + 9 = 12$  та  $TT_{4,1,3,2} = 0 + 0 + 3 + 9 = 12$ . Отже, найкоротший гамільтонів шлях графа (він наведений на рис. 4) включає послідовність робіт  $J_4 J_3 J_1 J_2$ . З табл. 1 випливає, що є «конкуруючі» шляхи, з яких можна вибрати тільки один – випадково або за допомогою правил домінування (наприклад, EDD [1–3, 11, 13]), які дають локально-оптимальне рішення, або за допомогою «стягнення» усіх шляхів до наступного рангу, тобто обирати всі побудовані шляхи (без відсікання) для наступного рангу. Якщо на будь-якому ранзі величини запізнювання усіх робіт дорівнюють одна одній, використовується правило домінування EDD: наприклад, на ранзі 3  $T(3)_{4,3,1} = 6 + (4 - 7) = 3$  та  $T(3)_{3,4,1} = 6 + (4 - 7) = 3$ . Вибираємо  $T(3)_{4,3,1}$ , тому що  $d_4 < d_3$ . Процедура триває, поки не досягне рангу 4, на якому остаточно вибирається найкоротший гамільтонів шлях в графі з мінімальним значенням запізнювання. Це й буде сумарним запізнюванням, тобто шуканий розв’язок задачі. Отже, в розглянутому прикладі оптимальний режим визначається послідовністю  $J_4, J_3, J_1, J_2$  з сумарним запізнюванням  $TT_{j_4 j_3 j_1 j_2} = 12$ . Послідовність побудови гамільтонового шляху в графі на рис. 1 на кожному ранзі показана на рис. 2–4.

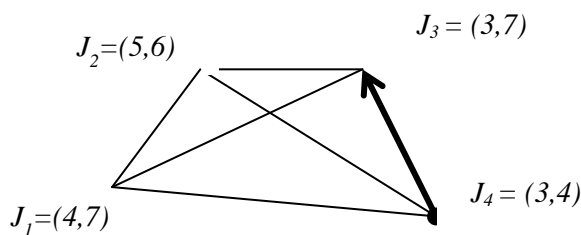


Рис. 2. Вибір вершини на ранзі 2

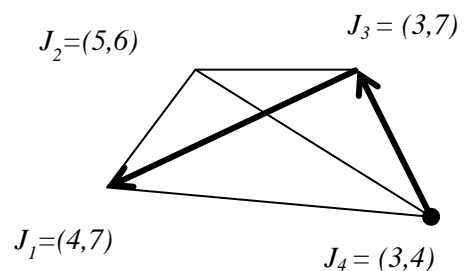


Рис. 3. Вибір вершини на ранзі 3

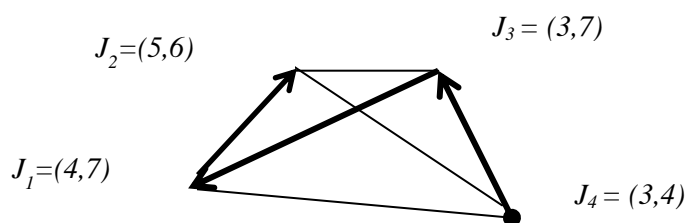


Рис. 4. Вибір вершини на ранзі 4

У подальшому в табл. 1–3, 5, 6 оптимальні шляхи на кожному ранзі матриць виділені напівжирним.

У табл. 2 наведено приклад роботи алгоритму з використанням правила MDD за критерієм  $\min\{\max(p_j, d_j - t)\}$  [15].

У табл. 3 наведений приклад застосування пропонованого алгоритму (оптимізації за напрямком) без використання правил домінування.

Аналіз отриманих результатів (див. табл. 1–3) свідчить про те, що можна поліпшити результат (значення сумарного запізнювання), а в деяких випадках, доволі значно, використовуючи правила EDD та MDD, що дозволяє зменшити сумарне запізнювання робіт без використання правил, що може значно вплинути на остаточний результат в разі великої кількості робіт у послідовності, при цьому кількість вершин гамільтонового шляху, отриманих на основі правил, становить дві з чотирьох вершин повнозв'язного графа (рис. 1).

Таблиця 1

**Приклад роботи алгоритму з включенням правила EDD**

Вершина	Ранг 1	Ранг 2	Ранг 3	Ранг 4
J <sub>1</sub> (4;7)	T(1) <sub>1</sub> =(0+4)–7=–3	T(2) <sub>21</sub> =(5+4)–7=2 T(2) <sub>31</sub> =(3+4)–7=0 T(2) <sub>41</sub> =(3+4)–7=0	T(3) <sub>421</sub> =(8+4)–7=5 T(3) <sub>431</sub> =(6+4)–7=3 T(3) <sub>341</sub> =(6+4)–7=3	TT <sub>3421</sub> =0+2+5+8=15
J <sub>2</sub> (5;6)	T(1) <sub>2</sub> =(0+5)–6=–1	T(2) <sub>12</sub> =(4+5)–6=3 T(2) <sub>32</sub> =(3+5)–6=2 T(2) <sub>42</sub> =(3+5)–6=2	T(3) <sub>412</sub> =(7+5)–6=6 T(3) <sub>432</sub> =(6+5)–6=5 T(3) <sub>342</sub> =(6+5)–6=5	TT <sub>4312</sub> =0+0+3+9=12 TT <sub>4132</sub> =0+0+3+9=12
J <sub>3</sub> (3;7)	T(1) <sub>3</sub> =(0+3)–7=–4	T(2) <sub>13</sub> =(4+3)–7=0 T(2) <sub>23</sub> =(5+3)–7=1 T(2) <sub>43</sub> =(3+3)–7=–1	T(3) <sub>413</sub> =(7+3)–7=3 T(3) <sub>423</sub> =(8+3)–7=4	
J <sub>4</sub> (3;4)	T(1) <sub>4</sub> =(0+3)–10=–7	T(2) <sub>14</sub> =(4+3)–4=3 T(2) <sub>24</sub> =(5+3)–4=4 T(2) <sub>34</sub> =(3+3)–4=2		

Таблиця 2

**Приклад роботи алгоритму з включенням правила MDD**

Вершина	Ранг 1	Ранг 2	Ранг 3	Ранг 4
J <sub>1</sub> (4;7)	T(1) <sub>1</sub> =(0+4)–7=–3	T(2) <sub>21</sub> =(5+4)–7=2 T(2) <sub>31</sub> =(3+4)–7=0 T(2) <sub>41</sub> =(3+4)–7=0	T(3) <sub>421</sub> =(8+4)–7=5 T(3) <sub>431</sub> =(6+4)–7=3 T(3) <sub>341</sub> =(6+4)–7=3	TT <sub>4321</sub> =0+0+5+8=13
J <sub>2</sub> (5;6)	T(1) <sub>2</sub> =(0+5)–6=–1	T(2) <sub>12</sub> =(4+5)–6=3 T(2) <sub>32</sub> =(3+5)–6=2 T(2) <sub>42</sub> =(3+5)–6=2	T(3) <sub>412</sub> =(7+5)–6=6 T(3) <sub>432</sub> =(6+5)–6=5 T(3) <sub>342</sub> =(6+5)–6=5	TT <sub>4312</sub> =0+0+3+9=12 TT <sub>4132</sub> =0+0+3+9=12
J <sub>3</sub> (3;7)	T(1) <sub>3</sub> =(0+3)–7=–4	T(2) <sub>13</sub> =(4+3)–7=0 T(2) <sub>23</sub> =(5+3)–7=1 T(2) <sub>43</sub> =(3+3)–7=–1	T(3) <sub>413</sub> =(7+3)–7=3 T(3) <sub>423</sub> =(8+3)–7=4	
J <sub>4</sub> (3;4)	T(1) <sub>4</sub> =(0+3)–4=–1	T(2) <sub>14</sub> =(4+3)–4=3 T(2) <sub>24</sub> =(5+3)–4=4 T(2) <sub>34</sub> =(3+3)–4=2		

Задачу мінімізації сумарного зваженого запізнювання на одному пристрої  $TWT_S = \sum_{j \in S} \max(0, (C_j - d_j) \cdot w_j) \cdot (1 || \sum w_i T_i)$  можна сформулювати так. Множина робіт, проіндексованих від

1 до n, має бути оброблена без переривань на одиночному обчислювальному пристрої, який може обробляти тільки одну роботу в певний момент часу. Усі роботи надходять до пристрою в момент часу 0. Робота характеризується часом її оброблення  $L_i$  (процесорним часом,  $p_i$ ), директивним

строком  $d_i$  та вагою  $w_i$ . Для зручності усі роботи впорядковані згідно з правилом EDD, так що  $d_i < d_j$ ; якщо  $d_i = d_j$ , тоді  $p_i < p_j$ ; якщо  $p_i = p_j$ , тоді  $w_i < w_j$  для всіх  $i, j$  ( $i < j$ ). Якщо робота  $i$  завершиться після свого директивного строку  $d_i$ , виникає штраф за запізнювання (зважений). Для експериментального дослідження оцінки впливу включення правил домінування, використовуються правила, які наведені в табл. 4.

Таблиця 3

**Приклад роботи алгоритму без включення правил домінування**

Вершина	Ранг 1	Ранг 2	Ранг 3	Ранг 4
$J_1 (4;7)$	$T(1)_1=(0+4)-7=-3$	$T(2)_{21}=(5+4)-7=2$ $T(2)_{31}=(3+4)-7=0$ $T(2)_{41}=(3+4)-7=0$	$T(3)_{321}=(8+4)-7=5$ $T(3)_{431}=(6+4)-7=3$ $T(3)_{341}=(6+4)-7=3$	$TT_{4321}=0+0+5+8=13$
$J_2 (5;6)$	$T(1)_2=(0+5)-6=-1$	$T(2)_{12}=(4+5)-6=3$ $T(2)_{32}=(3+5)-6=2$ $T(2)_{42}=(3+5)-6=2$	$T(3)_{312}=(7+5)-6=6$ $T(3)_{432}=(6+5)-6=5$ $T(3)_{342}=(6+5)-6=5$	$TT_{3412}=0+0+3+9=14$ $TT_{3142}=0+0+6+9=15$
$J_3 (3;7)$	$T(1)_3=(0+3)-7=-4$	$T(2)_{13}=(4+3)-7=0$ $T(2)_{23}=(5+3)-7=1$ $T(2)_{43}=(3+3)-7=-1$		
$J_4 (3;4)$	$T(1)_4=(0+3)-4=-1$	$T(2)_{14}=(4+3)-4=3$ $T(2)_{24}=(5+3)-4=4$ $T(2)_{34}=(3+3)-4=2$	$T(3)_{314}=(7+3)-4=6$ $T(3)_{324}=(8+3)-4=7$	

Таблиця 4

**Правила домінування для сумарного зваженого запізнювання**

Правило	Назва	Опис	Формула визначення пріоритету
WEDD	Weighted Earliest Due Date	Зважене з найбільш раннім директивним строком	$\max \left\{ \frac{w_j}{d_j} \right\}$
WMDD	Weighted Modified Due Date	Зважене з модифікованим директивним строком	$\min \left\{ \frac{\max(p_j, d_j - t)}{w_j} \right\}$
WSPT	Weighted Shortest Processing Time	Зважене з найкоротшим процесорним часом виконання	$\min \left\{ \frac{p_j}{w_j} \right\}$

Приклади роботи алгоритмів з включенням правила домінування WMDD в роботу базового алгоритму для зваженого випадку та без правила наведені в табл. відповідно 5 та 6.

**Експериментальне дослідження алгоритмів та аналіз отриманих результатів**

Для експериментального дослідження запропонованих алгоритмів розроблена їх програмна реалізація, за допомогою якої розраховані залежності часу виконання та сумарного запізнювання для базового алгоритму (оптимізації за напрямком [12]) та алгоритмів з використанням правил домінування для незваженого та зваженого випадків (див. рис. 5–10) на основі методики, яка використана в статті [14].

Таблиця 5

## Приклад роботи алгоритму з включенням правила WMDD

Вершина	Ранг 1	Ранг 2	Ранг 3	Ранг 4
$J_1 (4;7,2)$	$WT(1)_1 = ((0+4)-7)*2 = -6$	$WT(2)_{21} = ((5+4)-7)*2 = 4$ $WT(2)_{31} = ((3+4)-7)*2 = 0$ $WT(2)_{41} = ((3+4)-7)*2 = 0$	$WT(3)_{321} = ((8+4)-7)*2 = 10$ $WT(3)_{431} = ((6+4)-7)*2 = 6$ $WT(3)_{341} = ((6+4)-7)*2 = 6$	$TWT_{4321} = 0+0+5*4+8*2 = 36$
$J_2 (5;6,4)$	$WT(1)_2 = ((0+5)-6)*4 = -4$	$WT(2)_{12} = ((4+5)-6)*4 = 12$ $WT(2)_{32} = ((3+5)-6)*4 = 8$ $WT(2)_{42} = ((3+5)-6)*4 = 8$	$WT(3)_{312} = ((7+5)-6)*4 = 24$ $WT(3)_{432} = ((6+5)-6)*4 = 20$ $WT(3)_{342} = ((6+5)-6)*4 = 20$	$TWT_{4312} = 0+0+3*2+9*4 = 42$ $TWT_{3142} = 0+0+3*5+9*4 = 51$
$J_3 (3;7,5)$	$WT(1)_3 = ((0+3)-7)*5 = -20$	$WT(2)_{13} = ((4+3)-7)*5 = 0$ $WT(2)_{23} = ((5+3)-7)*5 = 5$ $WT(2)_{43} = ((3+3)-7)*5 = -5$		
$J_4 (3;4,3)$	$WT(1)_4 = ((0+3)-4)*3 = -3$	$WT(2)_{14} = ((4+3)-4)*3 = 9$ $WT(2)_{24} = ((5+3)-4)*3 = 12$ $WT(2)_{34} = ((3+3)-4)*3 = 6$	$WT(3)_{314} = ((7+3)-4)*3 = 18$ $WT(3)_{423} = ((8+3)-7)*5 = 20$	

Таблиця 6

## Приклад роботи алгоритму без правила WMDD

Вершина	Ранг 1	Ранг 2	Ранг 3	Ранг 4
$J_1 (4;7,2)$	$WT(1)_1 = ((0+4)-7)*2 = -6$	$WT(2)_{21} = ((5+4)-7)*2 = 4$ $WT(2)_{31} = ((3+4)-7)*2 = 0$ $WT(2)_{41} = ((3+4)-7)*2 = 0$	$WT(3)_{421} = ((8+4)-7)*2 = 10$ $WT(3)_{431} = ((6+4)-7)*2 = 6$ $WT(3)_{341} = ((6+4)-7)*2 = 6$	$TWT_{4321} = 0+6+20+8*2 = 42$
$J_2 (5;6,4)$	$WT(1)_2 = ((0+5)-6)*4 = -4$	$WT(2)_{12} = ((4+5)-6)*4 = 12$ $WT(2)_{32} = ((3+5)-6)*4 = 8$ $WT(2)_{42} = ((3+5)-6)*4 = 8$	$WT(3)_{412} = ((7+5)-6)*4 = 24$ $WT(3)_{432} = ((6+5)-6)*4 = 20$ $WT(3)_{342} = ((6+5)-6)*4 = 20$	$TWT_{3412} = 0+6+6+9*4 = 48$ $TWT_{4132} = 0+0+15+9*4 = 51$
$J_3 (3;7,5)$	$WT(1)_3 = ((0+3)-7)*5 = -20$	$WT(2)_{13} = ((4+3)-7)*5 = 0$ $WT(2)_{23} = ((5+3)-7)*5 = 5$ $WT(2)_{43} = ((3+3)-7)*5 = -5$	$WT(2)_{413} = ((7+3)-7)*5 = 15$ $WT(2)_{423} = ((8+3)-7)*5 = 20$	
$J_4 (3;4,3)$	$WT(1)_4 = ((0+3)-4)*3 = -3$	$WT(2)_{14} = ((4+3)-4)*3 = 9$ $WT(2)_{24} = ((5+3)-4)*3 = 12$ $WT(2)_{34} = ((3+3)-4)*3 = 6$		

## Оцінка ефективності включення правил домінування до базового алгоритму

Для отримання локального оптимуму (напрямку оптимізації з застосуванням певного правила домінування) визначається за двома метриками: відношення кількості вершин гамільтонового шляху, отриманих з правилами домінування, до загальної кількості вершин – щільність  $\sum_{GP} v_{dr} / n$ ,

де  $v_{dr}$  – множина вершин графа, отриманих на основі певного правила домінування; відносне зменшення часу запізнювання  $(TT - TT_{dr}) / TT$ , де  $TT_{dr}$  – сумарний час запізнювання з включенням в алгоритм правила домінування;  $TT$  – сумарний час запізнювання без включення правила домінування.

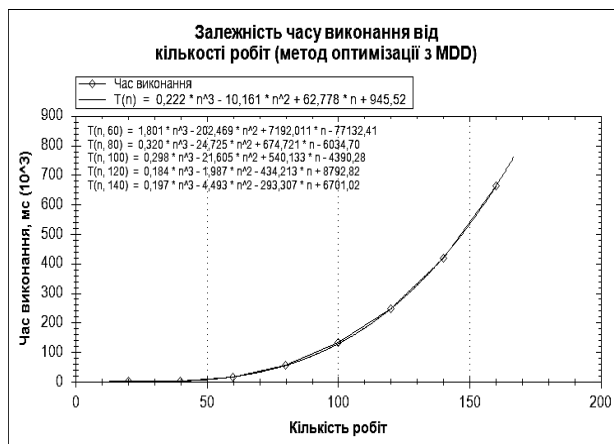
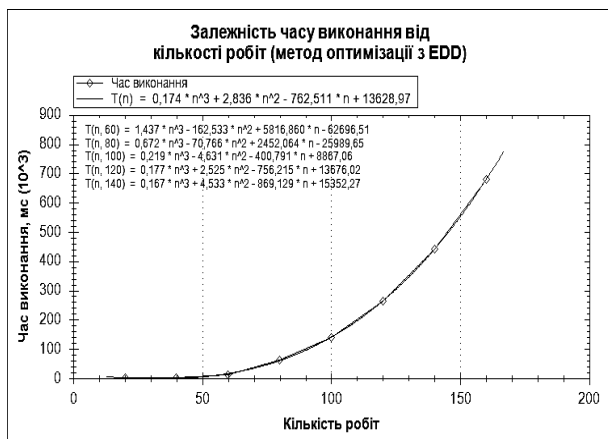


Рис. 5. Залежність часу виконання від кількості робіт для базового алгоритму з використанням правил домінування EDD та MDD

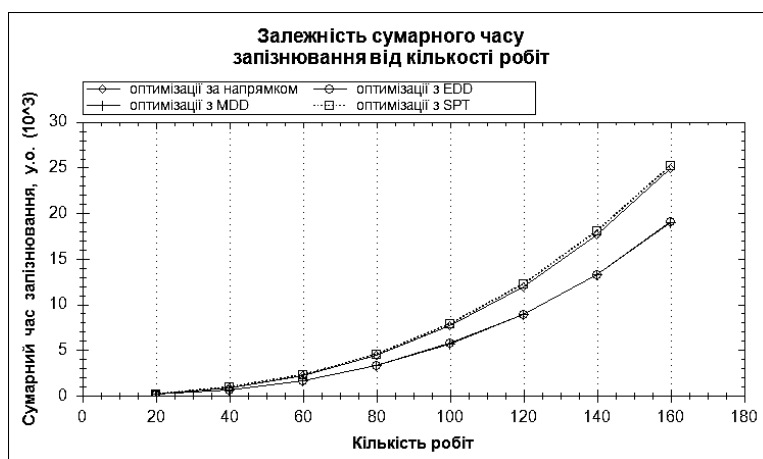


Рис. 6. Залежність сумарного запізнювання від кількості робіт для алгоритмів повного перебору та з використанням правил домінування EDD та MDD

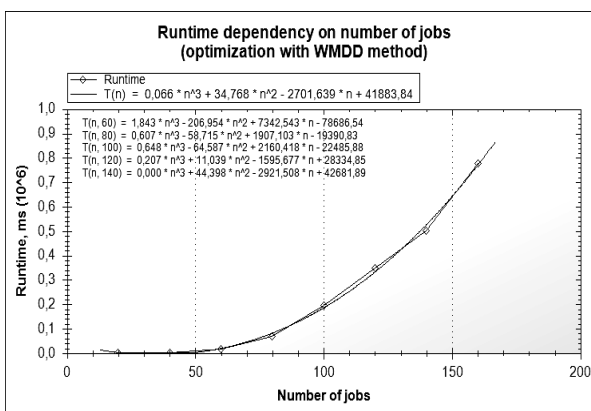
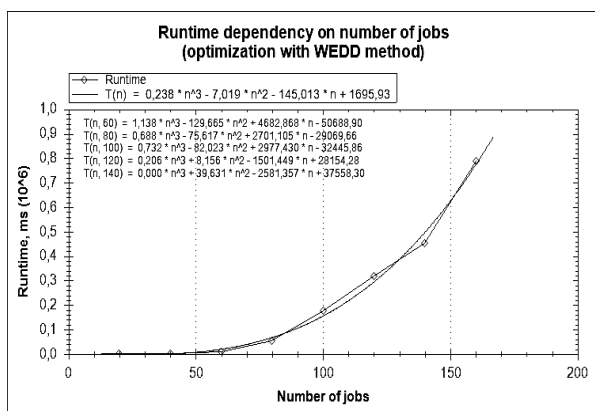


Рис. 7. Залежність часу виконання від кількості робіт для базового алгоритму з використанням правил домінування WEDD та WMDD

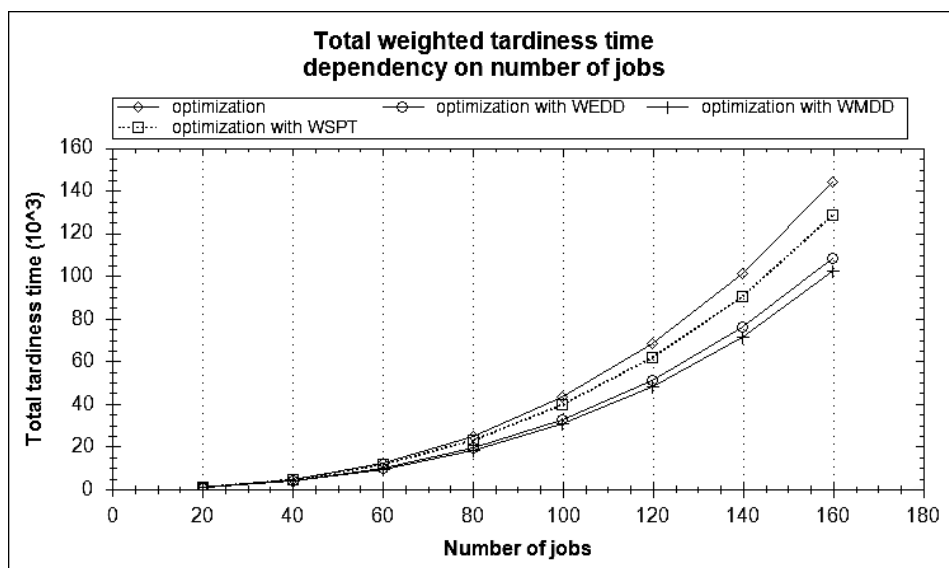


Рис. 8. Залежність сумарного запізнювання від кількості робіт для алгоритмів повного перебору та з використанням правил домінування WEDD та WMDD

Для дослідження ефективності застосування правил домінування, а саме, впливу на час виконання та сумарного запізнювання, на вихідних даних з такою кількістю робіт, яка зустрічається на практиці, проведені експерименти. Для цього згенеровано пакети з 50 завдань (instances, спостережень) з 20 – 160 робіт у кожному та оцінено час виконання базовим алгоритмом (без правил) та з використанням правил домінування, зменшення сумарного часу запізнювання по зрівнянню з алгоритмами без використання правил домінування, кількості локальних оптимальних рішень – відношення отриманих за допомогою правил домінування вершин в гамільтоновому шляху в графі по відношенню до загальної кількості вершин.

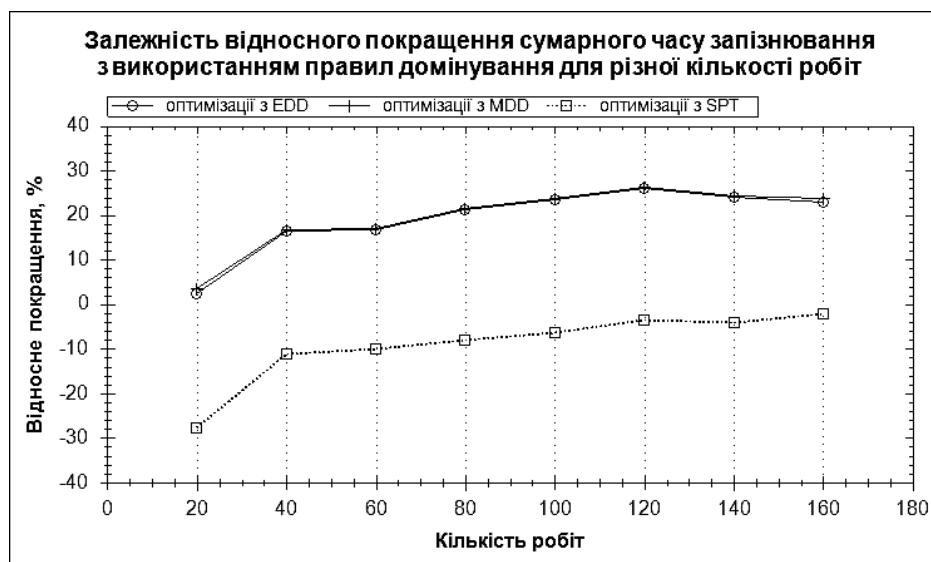


Рис. 9. Залежність зменшення сумарного часу запізнювання по відношенню до базового алгоритму з використанням правил домінування від кількості робіт, %

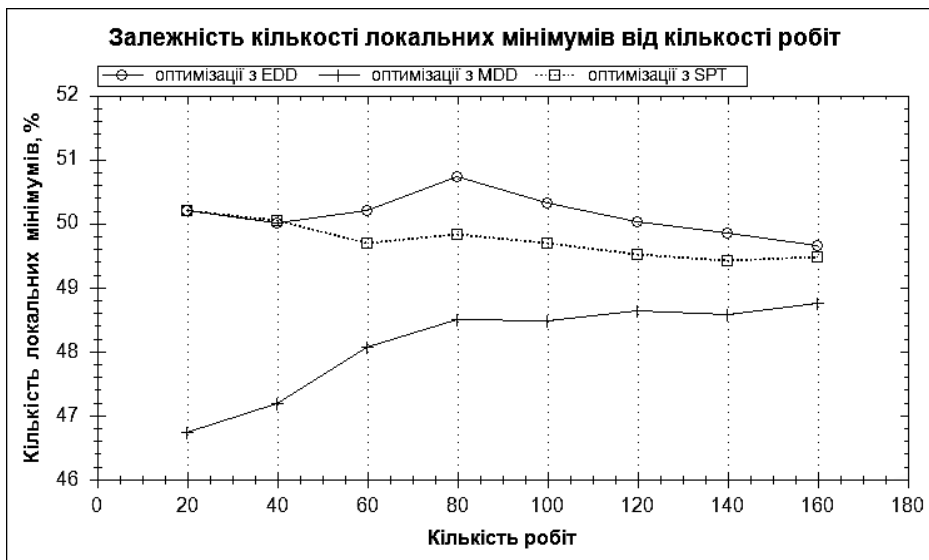


Рис. 10. Залежність кількості локально-оптимальних рішень, отриманих з використанням правил домінування, від кількості робіт, %

### Висновки

Аналіз отриманих експериментальних результатів дозволяє зробити такі висновки щодо ефективності використання запропонованих алгоритмів з використанням правил домінування для побудови оптимальних розкладів виконання завдань:

включення правил домінування до базового алгоритму не збільшує час виконання за алгоритмом, тому, з погляду часової складності, його ефективність не зменшується;

використання правил домінування для розглянутого класу алгоритмів мінімізації сумарного часу запізнювання дозволяє в середньому зменшити величину критерію мінімізації (відносне зменшення – до 20–25% для різних правил), що дозволяє зробити висновок про істотну ефективність їх застосування.

надалі планується провести дослідження алгоритмів в напрямку оцінювання ефективності використання правил домінування на основі запропонованих метрик для різних тривалостей робіт та співвідношень між тривалостями та директивними строками виконання робіт.

1. Koulamas C. The total tardiness problem: Review and extensions, *Operations Research* v.42 1994, pp.1025–1041.
2. Koulamas C. The single-machine total tardiness scheduling problem // Review and extensions, *European Journal of Operational Research*, v. 202 No.1 2010, pp. 1–7.
3. T. Sena, J. M. Suleka, Parthasarathi Dileepan, Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey, *Int. J. Production Economics*, v.83 No. 1 2003, – pp. 1–12.
4. J. Du, J.Y.-T. Leung, Minimizing total tardiness on one machine is NP-hard, *Mathematics of Operations Research*, 1990 15, pp. 483–495.
5. E.L. Lawler, A “pseudo polynomial” algorithm for sequencing jobs to minimize total tardiness, *Annals of Discrete Mathematics*, No1 1977, pp. 331–342.
6. K.R. Baker, L.E. Shrage, Finding an optimal sequence by dynamic programming an extension to precedence-related tasks, *Oper. Res.*, No 26 1978, pp. 111–120.
7. E.L. Lawler, A fully polynomial approximation scheme for the total tardiness problem, *Operations Research Letters*, No 1 1982, pp. 207–208.
8. C.N. Potts, L.N. Van Wassenhove, Dynamic programming and decomposition approaches for the single machine total tardiness problem, *European Journal of Operational*, No 32 1987, pp. 405–414.
9. M.Y. Kovalyov, Improving the complexities of approximation algorithms for optimization problems, *Operations Research Letters*, v. 17 March 1995, pp. 85–87.
10. C. Koulamas, A faster fully polynomial approximation scheme for the single machine total tardiness problem,

*European Journal of Operational Research*, No 193 2009, pp. 637–638. 11. Павлов А.А. Новый подход к решению задачи «Минимизация суммарного взвешенного опоздания при выполнении независимых заданий с директивными сроками одним прибором» / А.А. Павлов, Е.Б. Мисюра. // Системные исследования и информационные технологии. – 2002. – № 2. – С. 7–32. 12. Мінухін С.В. Метод мінімізації часу виконання завдань з директивними строками на некластеризованому ресурсі обчислювальної системи // Інформаційно-керуючі системи на залізничному транспорті. – 2009. – №3. – С. 47 – 53. 13. S. Minukhin, Efficient method for Single machine total tardiness problem // IV International Conference «Problems of Cybernetics and Informatics» (PCI'2012), September 12–14, 2012. Електронний ресурс – режим доступу: [www.pci2012.science.az/1/23.pdf](http://www.pci2012.science.az/1/23.pdf). 14. Мінухін С.В. Дослідження методів мінімізації сумарного часу запізнювання робіт з директивними строками на одиночному обчислювальному ресурсі / С.В. Мінухін, Д.С. Ленько // Проблеми і перспективи розвитку ІТ-індустрії. Системи обробки інформації. – 2013. – Вип. №3 (110). Т. 2. – С.30–35. 15. J. Bean, Accuracy of Modified date Rule / J. Bean, Hall D. Електронний ресурс – режим доступу: [www.deerblue.lib.umich.edu](http://www.deerblue.lib.umich.edu).