УДК 004.2

ОПТИМАЛЬНИЙ РОЗПОДІЛ ОБМЕЖЕНИХ РЕСУРСІВ В МУЛЬТИПРОЦЕСОРНИХ СИСТЕМАХ

Косолап А. І. – д-р фіз.-мат. наук, професор, професор кафедри комп'ютерних наук та інформаційних технологій Дніпровського національного університету ім. Олеся Гончара, Дніпро, Україна.

АНОТАЦІЯ

Актуальність. В роботі розглядаються мультипроцесорні системи, які складаються з безлічі процесорів з загальною оперативною пам'яттю. Ефективність функціонування таких систем залежить від операційної системи. Вона повинна забезпечити рівномірне завантаження процесорів завданнями, при якому пікове навантаження на оперативну пам'ять буде мінімальним. Це досить складна проблема. В даній роботі вона розв'язується шляхом побудови оптимізаційних моделей та розробкою ефективних евристичних алгоритмів. Дана проблема розв'язується в два етапи. На першому етапі знаходиться оптимальне завантаження процесорів завданнями, а на другому – мінімізація пікового навантаження оперативної пам'яті. Побудовано декілька оптимізаційних моделей цієї задачі, для розв'язування яких ефективним є метод точної квадратичної регуляризації. Розроблені також ефективні евристичні алгоритми. Проведені порівняльні обчислювальні експерименти, які підтверджують ефективність запропонованої технології розв'язування даної проблеми.

Мета роботи. Розробка математичних оптимізаційних моделей, методів та алгоритмів оптимального розподілу ресурсів в мультипроцесорних системах.

Метод. Ефективним є двоетапний розв'язок даної проблеми. Запропоновано декілька оптимізаційних моделей, які містять булеві змінні. Такі моделі досить складні для знаходження оптимальних розв'язків. Для їх розв'язування пропонується використовувати метод точної квадратичної регуляризації. Цей метод оптимізації використовується вперше для даного класу задач, тому він потребував розробки відповідного алгоритмічного забезпечення. В операційних системах, як правило, реалізуються евристичні алгоритми. Тому пропонуються ефективні евристичні алгоритми, які використовують фінальний принцип, що значно покращує розв'язок задачі.

Результати. Побудовані нові оптимізаційні моделі розподілу обмежених ресурсів в мультипроцесорних системах. Розроблені ефективні евристичні алгоритми, які реалізовані програмно засобами VBA в пакеті Excel. Розроблене також програмне забезпечення для введення початкових даних оптимізаційних моделей, що спрощує їх розв'язування. Приведені результати обчислюваних експериментів.

Висновки. Розроблена нова ефективна технологія оптимального розподілу ресурсів в мультипроцесорних системах. Розроблені евристичні алгоритми, які реалізовані програмно. Проведені обчислювальні експерименти підтверджують ефективність запропонованої технології розв'язування задачі.

КЛЮЧОВІ СЛОВА: мультипроцесорні системи, оптимізаційні моделі, задачі з булевими змінними, евристичні алгоритми, метод точної квадратичної регуляризації, фінальний принцип.

АБРЕВІАТУРИ

Solver – надбудова Excel;
OpenSolver – надбудова Excel з відкритим кодом;
EQR – точна квадратична регуляризація.

НОМЕНКЛАТУРА

t_i – надбудова Excel з відкритим кодом;
t_{ij} – час виконання i-го завдання на j-му процесорі;
v_i – обсяг оперативної пам'яті для i-го завдання;
T – мінімальний час виконання всіх завдань;
V – пікове навантаження на оперативну пам'ять;
s, r – параметри квадратичної регуляризації;
d – змінна методу EQR;
x_{ij} – булева змінна, яка дорівнює одиниці, якщо i-те завдання виконується j-м процесором;
$||x||^2$ – квадрат Евклідової норма вектору x;
t_{ik} – булева змінна, яка дорівнює одиниці, якщо i-те завдання виконується в момент часу k;
x_i – час початку виконання i-го завдання;
n – кількість завдань мультипроцесорної системи;
m – кількість процесорів мультипроцесорної системи;
i – змінна індексів сум, параметрів та змінних;
j – змінна індексів сум, параметрів та змінних;
k – змінна індексів сум, параметрів та змінних;
$g_i(t, x_i, v_i)$ – кусково-постійна функція графіку i-го завдання;
I_j – множина завдань j-го процесора.

ВСТУП

Сьогодні існує безліч важливих задач в різних галузях людської діяльності, які ще не можуть бути ефективно розв'язані за допомогою сучасних комп'ютерних систем. Такі задачі відносяться до класу NP-складних. Потужності сучасних комп'ютерних систем недостатньо для розв'язування таких задач. Ці задачі можуть бути розв'язані за рахунок вдосконалення математичних моделей, розробки нових та ефективних методів та алгоритмів, а також за рахунок збільшення потужності обчислювальних систем. В наш час збільшення потужності обчислювальних систем відбувається шляхом розробки мультипроцесорних систем. З'являються багатоядерні процесори, будуються мультипроцесори, які можуть містити сотні і тисячі процесорів. Відповідно такі системи будуть працювати ефективно, якщо всі процесори будуть завантажені рівномірно, а ресурси оперативної пам'яті будуть розподілятися оптимально. За ефективність роботи мультипроцесорних систем відповідає

DOI 10.15588/1607-3274-2025-2-18

їх операційна система. В наш час мультипроцесорні операційні системи знаходяться в процесі розвитку.

В мультипроцесорних системах операційна система визначає, які завдання будуть виконуватися на яких процесорах і в якому порядку, щоб пікове навантаження на загальну оперативну пам'ять було мінімальним. Ця задача може розв'язуватися шляхом побудови оптимізаційних моделей, або розробкою ефективних евристичних алгоритмів. Оптимізаційні моделі будуть містити булеві змінні і можуть бути лінійними або квадратичними. Для розв'язування таких задач, як правило, використовуються методи розгалужень та границь. Це точні методи, але для знаходження оптимальних розв'язків вони потребують досить багато часу. Такі методи реалізовані програмно, зокрема в надбудовах Solver, OpenSolver пакету Excel. Надбудова OpenSolver з відкритим кодом більш ефективна та дозволяє розв'язувати задачі великої розмірності. Але обчислювальні експерименти показали, що ці програми часто передчасно завершують розрахунки і не знаходять оптимальні розв'язки. Показано, що кращою альтернативою цим програмам є метод точної квадратичної регуляризації [1], який використовує теж програму OpenSolver, але для перетвореної задачі. Цей метод потребує для розв'язування задачі досить мало часу і знаходить кращі розв'язки задачі.

Альтернативою оптимізаційним моделям є розробка ефективних евристичних алгоритмів. Такі алгоритми можуть розв'язувати задачі, навіть великої розмірності, за долі секунди. Показано, що такі алгоритми можуть бути вдосконалені шляхом використання фінального принципу. Цей принцип полягає в тому, що для розв'язування задачі використовується два алгоритму. Після завершення роботи першого алгоритму запускається наступний алгоритм, який покращує розв'язок задачі. Як показали обчислювальні експерименти такий принцип дозволяє майже завжди знаходити оптимальні розв'язки в задачі розподілу навантаження мультипроцесорних систем.

Об'єктом дослідження даної роботи є мультипроцесорні системи, операційні системи таких систем та їх оптимальне функціонування.

Предметом дослідження є оптимізаційні моделі мультипроцесорних систем та ефективні евристичні алгоритми оптимального розподілу ресурсів в таких системах.

Метою даного дослідження є розробка оптимізаційних моделей мультипроцесорних систем з розподілом обмежених ресурсів та побудова відповідних ефективних евристичних алгоритмів.

1 ПОСТАНОВКА ЗАДАЧІ

Розглядається мультипроцесорна система, яка містить m процесорів рівної потужності та n завдань, які необхідно виконати ($n > m$). Для кожного i-го завдання відомий час виконання t_i та обсяг оперативної пам'яті v_i. Необхідно виконати всі завдання за мінімальний час так, щоб пікове навантаження на оперативну пам'ять було мінімальним. Таким чином, вхідними даними задачі є набір параметрів (n, m, t_i, v_i, i =1,…,n). Вихідними даними задачі є x_{ij} – час початку виконання i-го завдання на j-му процесорі, або послідовність їх виконання на кожному процесорі, мінімальний час T виконання всіх завдань та мінімальне пікове навантаження V на оперативну пам'ять мультипроцесорної системи.

Критерієм оптимізації буде мінімальне значення V при мінімальному T. Задача має декілька обмежень. Зокрема, кожне завдання повинно бути виконано, одночасно кожен процесор може виконувати тільки одне завдання, що означає $x_{ij} + t_i \le x_{kj}$ або $x_{kj} + t_k \le x_{ij}$, тобто або i-те завдання передує виконанню k-го завдання, або навпаки для кожного процесору. Ці умови рівнозначні квадратичним обмеженням $(x_{ij} + t_i - x_{kj})(x_{kj} + t_k - x_{ij}) \le 0$. Термін виконання кожного завдання буде менше T, тобто $x_{ij} + t_i \le T$, $\forall ij$, а обсяг оперативної пам'яті в кожний момент часу не перевищує значення V. Виконання останнього обмеження достатньо перевірити в момент початку виконання кожного завдання. Таку умову задати досить складно і вона буде визначена при побудові математичної моделі шляхом введення додаткових булевих змінних. Крім того, необхідно врахувати технічні обмеження, зокрема, в момент початку виконання i-го завдання на j-му процесорі може виконуватися не більше $m - 1$ завдань на інших процесорах. Таким чином, вихідними даними задачі є набір (x_{ij}, $\forall ij$, T, V).

Дана задача розглядається, коли кожне завдання виконується без переривань або з перериваннями. Для задачі з перериваннями задається квант часу q. Тоді кожне завдання виконується не більше кванту часу, після чого виконання завдання переривається і процесор починає виконувати наступне завдання та згодом повертається на виконання перерваного завдання. Кожну частину перерваного завдання можна розглядати як окреме завдання, що призводить до збільшення кількості завдань в задачі з перериваннями.

Дана задача розв'язується в два етапи. На першому етапі всі завдання розподіляються по процесорам системи так, щоб час виконання всіх завдань був мінімальним. Якщо всі t_i є цілими і час роботи процесорів відрізняється не більше ніж на одиницю, то такий розподіл завдань буде оптимальним.

На другому етапі, після розподілу завдань, розв'язується задача мінімізації пікового навантаження оперативної пам'яті мультипроцесорної системи. Це досягається шляхом переупорядкування виконання завдань на кожному процесорі.

Розглянута задача допускає різні узагальнення. Зокрема завдання можуть поступати в реальному часі, термін їх виконання може бути невідомий, також такі завдання можуть бути взаємопов'язані, що накладає додаткові лінійні умови на порядок виконання завдань, наприклад, $x_{ij} + t_i \le x_{kj}$. Задачі з цими узагальненнями можуть бути ефективно розв'язані, якщо знайдені ефективні алгоритми розв'язування поставленої задачі.

DOI 10.15588/1607-3274-2025-2-18

2 ОГЛЯД ЛІТЕРАТУРИ

Перші публікації присвячені балансуванню навантаження мультипроцесорних систем розпочалися ще в 70-х роках минулого століття [2–3]. В роботі [4] було показано, що майже всі задачі, пов'язані з мультипроцесорними системами, відносяться до класу NP-складних. В той час побудовані математичні моделі даного класу задач мали переважно теоретичний характер. Складність проблеми спонукала розробку безлічі евристичних алгоритмів рівномірного завантаження процесорів [5–11], що важливо для ефективної паралельної обробки даних. Такі дослідження актуальні до нашого часу [12]. Значна увага була приділена розробці ефективних алгоритмів балансування навантаження мультипроцесорних систем для різних умов їх функціонування. Такі алгоритми можна знайти в монографії [13], де також приведені оптимізаційні квадратичні моделі, але без урахуванням пам'яті. Зауважимо, що задачу оптимального завантаження мультипроцесорної системи можна перетворити до мультимодальної задачі про рюкзак [13–14], але така задача теж не враховує завантаженість оперативної пам'яті мультипроцесорної системи.

3 МАТЕРІАЛИ І МЕТОДИ

Досить просто побудувати оптимізаційну модель оптимального розподілу завдань в мультипроцесорній системі по критерію мінімізації часу завершення виконання всіх завдань. Така модель має наступний вигляд

$$\min\{T \mid \sum_{i=1}^{n} t_i x_{ij} \le T, j = 1,...,m, \sum_{j=1}^{m} x_{ij} = 1, i = 1,...,n, \\ x_{ij} = 0 \vee 1, \forall ij\}, \quad (1)$$

де T – час виконання завдань, x_{ij} – булеві змінні. Змінна $x_{ij} = 1$, якщо i-те завдання виконується на j-му процесорі, інакше така змінна дорівнює нулю. Дану модель легко розширити на випадок, коли деякі завдання повинні виконуватися в заданому порядку. Тоді модель (1) необхідно доповнити лінійними обмеженнями вигляду $x_{ij} + t_i \le x_{kj}$, яке означає, що k-те завдання буде виконуватися після i-го завдання на j-му процесорі. Таких обмежень може бути декілька.

Якщо процесори різної потужності, то задача оптимального розподілу завдань буде мати вигляд

$$\min\{T \mid \sum_{i=1}^{n} t_{ij} x_{ij} \le T, j = 1,...,m, \sum_{j=1}^{m} x_{ij} = 1, i = 1,...,n, \\ x_{ij} = 0 \vee 1, \forall ij\}, \quad (2)$$

Задачі (1)–(2) є лінійними з булевими змінними. Кількість змінних дорівнює $nm+1$, а кількість обмежень $n + m + 1$. Ці задачі можна розв'язати в пакеті Excel надбудовою OpenSolver. Для задач малої розмірності ця програма дозволяє отримати оптимальні розв'язки. Але вже для 24 завдань і чотирьох процесорів ця програма не знайшла оптимальний розв'язок. Наведемо дані цієї задачі. Час виконання завдань дорівнює (3, 49, 11, 41, 14, 38, 3, 16, 21, 16, 38, 17, 25, 11, 26, 28, 17, 12, 32, 45, 32, 24, 38, 29). Програма OpenSolver знайшла наступний час роботи кожного процесору (148, 153, 153, 152) в той час, як оптимальним часом роботи кожного процесору є (152, 151, 151, 152), що на одну одиницю часу краще. Зі зростанням розмірності задачі ця різниця буде збільшуватися, крім того, час розв'язування задачі (1) програмою OpenSolver буде швидко зростати.

Квадратична регуляризація задачі (1) зводить її до наступної

$$\max\{\|x\|^2 \mid T + s + (r-1)\|x\|^2 \le d, \sum_{i=1}^{n} t_{ij} x_{ij} \le T, j = 1,..m, \\ \sum_{j=1}^{m} x_{ij} = 1, i = 1,...,n, \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij}(1 - x_{ij}) + r\|x\|^2 \le d, 0 \le x \le 1\}. \quad (3)$$

Задача (3) не містить булевих змінних, тому її розв'язування програмою OpenSolver займає долі секунди. Для отримання оптимального розв'язку задачі (1) метод EQR потребує 10–20 розв'язувань задачі (3) програмою OpenSolver, для зростаючих значень d доти не виконається умова $r\|x\|^2 = d$ з заданою точністю.

Після розподілу завдань кожен процесор може виконувати свої завдання в довільній послідовності. Але, якщо кожне завдання потребує заданий обсяг оперативної пам'яті, то послідовність виконання завдань кожним процесором буде впливати на пікове навантаження оперативної пам'яті. Побудуємо модель мінімізації пікового навантаження оперативної пам'яті.

Представимо виконання завдань у вигляді графіка, де кожне завдання зображено відрізком, довжина якого дорівнює часу виконання завдання, а по осі ординат такий відрізок співпадає з обсягом оперативної пам'яті v_i. Тоді сума всіх відрізків буде графіком завантаження оперативної пам'яті. Представимо кожний відрізок кусково-постійною функцією $g_i(t, x_i, v_i)$. Отримаємо наступну модель задачі

$$\min\{V \mid x_i + t_i \le T, i = 1,...,n, (x_i + t_i - x_j)(x_j + t_j - x_i) \le 0, \\ \forall i \ne j, \sum_{i=1}^{n} g_i(x_j, x_i, v_i) \le V, j = 1,...n\}. \quad (4)$$

Задача (4) містить нелінійні квадратичні обмеження, які задають умову послідовного виконання завдань на кожному процесорі. Або i-те завдання передує виконанні j-го завдання, або навпаки. Обмеження задачі (4) неопуклі і тому породжують її мультимодальність. Ускладнює задачу і обмеження на оперативну

DOI 10.15588/1607-3274-2025-2-18

пам'ять, яке містить кусково-постійні функції. Задача (4) містить мінімальну кількість змінних n+1 та кількість обмежень $\frac{n}{2}(\frac{n}{m}-1)+2n$ Не дивлячись на невелику розмірність задачі (4), її складно розв'язати, що пов'язано з розривністю функцій $g_i(t, x_i, v_i)$.

Розглянемо альтернативну модель задачі (4). Будемо допускати, що час виконання кожного завдання є цілим числом. Поставимо у відповідність кожному i-му завданню послідовність бінарних чисел t_{ik}. Число чисел такої послідовності дорівнює T. Якщо завдання обробляється процесором, то відповідні числа послідовності дорівнюють одиниці, інакше вони дорівнюють нулю. Тоді задача мінімізації пікового навантаження оперативної пам'яті буде мати вигляд

$$\min\{V \mid \sum_{i=1}^{n} v_i t_{ik} \le V, k=1,...,T, \sum_{k=1}^{T} t_{ik} = t_i, i=1,...,n, \sum_{i\in I_j} t_{ik} \le 1, \forall jk, t_{ik} = 0 \vee 1, \forall ik\}, \quad (5)$$

де I_j – множина завдань, що виконуються j-м процесором. Задача (5) є лінійною з булевими змінними. Кількість змінних дорівнює nT+1, а кількість обмежень $T(n+1)+n$. Наприклад, для розглянутої вище задачі з 4-ма процесорами і 24 завданнями будемо мати 3648 булевих змінних і 5824 обмеження. Задача (5) допускає виконання завдань з перериваннями. Для виконання завдань без переривань необхідно добавити наступні обмеження

$$\sum_{k=1}^{T-1}(t_{ik}+t_{ik+1})^2 \ge 4(t_i-1)+1, i=1,...,n. \quad (6)$$

Обмеження (6) зводить задачу (5) до квадратичної. Враховуючи велику розмірність задачі (5) її досить складно розв'язати навіть для задач невеликої розмірності.

Будемо вдосконалювати модель (4). Для цього введемо булеві змінні z_{ij} = 1, якщо i-те завдання виконується в момент початку виконання j-го завдання і z_{ij} = 0 інакше. Тоді повинні виконуватися обмеження

$$(x_i + t_i - x_j)(x_j - x_i) \ge 0, \forall i \ne j,$$

які виконується тільки тоді, коли в момент початку j-го завдання виконується i-те завдання. Але перший множник повинен бути більше нуля, тому

$$(x_i + t_i - x_j - \varepsilon)(x_j - x_i) \ge 0, \forall i \ne j, \quad (9)$$

де ε –мале додатне число. Це квадратичне обмеження замінимо лінійними обмеженнями

$$Q(z_{ij}-1) \le (x_i + t_i - x_j - \varepsilon), \forall i \ne j, \quad (10)$$

$$Q(z_{ij}-1) \le (x_j - x_i), \forall i \ne j, \quad (11)$$

де $Q = \sum_{i=1}^{n} t_i$ (велике додатне число). Тоді обмеження на оперативну пам'ять будуть мати вигляд

$$v_i + \sum_{j\ne i} z_{ji} \le V, i=1,...,n. \quad (12)$$

Заміна останнього обмеження в задачі (4) на обмеження (9)–(12) призводить до нової моделі

$$\begin{gathered}\min\{V \mid x_i + t_i \le T, i=1,...,n,\\ (x_i + t_i - x_j)(x_j + t_j - x_i) \le 0, \forall i \ne j,\\ Q(z_{ij}-1) \le (x_i + t_i - x_j - \varepsilon), \forall i \ne j,\\ Q(z_{ij}-1) \le (x_j - x_i), \forall i \ne j,\\ v_i + \sum_{j\ne i} z_{ji} \le V, i=1,...,n\}.\end{gathered} \quad (13)$$

Квадратичні обмеження задачі (13) теж можна замінити лінійними

$$(x_i + t_i - x_j) \le Q(1-y_{ij}), \forall i \ne j, \quad (14)$$

$$(x_j + t_j - x_i) \le Q y_{ij}, \forall i \ne j, \quad (15)$$

де y_{ij} = 1, якщо i-те завдання виконується раніше j-го завдання і y_{ij} = 0, якщо навпаки. Задача (13) після заміни квадратичних обмежень на (14)–(15) стає лінійною.

Кількість додатних значень z_{ij} буде дорівнювати $n(m-1)$, тому додаємо ще обмеження

$$\sum_{i=1}^{n}\sum_{j=1}^{m} z_{ij} = n(m-1). \quad (16)$$

Зауважимо, що якщо на множині завдань задано часткове упорядкування, то дані моделі необхідно доповнити відповідними лінійними обмеженнями.

Альтернативою розробленим оптимізаційним моделям можуть бути евристичні алгоритми. Ми пропонуємо два таких алгоритму. Перший алгоритм буде розподіляти завдання по процесорам, а другий – мінімізувати пікове навантаження оперативної пам'яті.

Алгоритм 1.

Крок 1. Упорядкуємо всі завдання в порядку спадання часу їх виконання.

Крок 2. Будемо роздавати послідовно (в порядку черги) завдання тим процесорам, які перші закінчують обробку завдань.

Крок 3. Перевіряємо, чи можна обміняти завдання двох процесорів з найбільшим та найменшим часом роботи при якому різниця часів роботи по модулю

DOI 10.15588/1607-3274-2025-2-18

цих процесорів буде мінімальною. Далі цей процес повторюємо до тих пір, доти така заміна можлива, інакше робота алгоритму закінчується.

Існуючі алгоритми включають тільки перші два кроки. Але, як показали експерименти, ці два кроки не дають оптимальний розподіл завдань. Крок 3 реалізує фінальний принцип і може бути окремим алгоритмом.

Алгоритм 2.

Крок 1. Використовуємо розглянутий вище алгоритм 1 для розподілу завдань по процесорам.

Крок 2. Для непарних номерів процесорів упорядкуємо завдання по спаданню обсягів оперативної пам'яті, а для парних – по зростанню обсягів оперативної пам'яті.

Крок 3. Шляхом попарної перестановки завдань на кожному процесорі знайдемо перестановку, яка максимально зменшить пікове навантаження оперативної пам'яті. Цей процес продовжується доти така перестановка можлива.

В цьому алгоритмі на кроці 3 теж реалізується фінальний принцип.

Розглянуті алгоритми реалізовані програмно засобами VBA для Excel 10. На вході цієї програми маємо терміни виконання завдань та їх обсяги оперативної пам'яті, які заносяться на аркуш Excel в дві колонки. Подвійним клацанням миші викликається діалогове вікно в яке вноситься кількість завдань та кількість процесорів. Результат розрахунку (завдання кожного процесору та порядок їх виконання) виносяться на аркуш Excel разом з піковим навантаженням оперативної пам'яті. Для задач великої розмірності програма знаходить розв'язок задачі за долі секунди. В діалогове вікно програми можна ввести квант часу і тоді програма розіб'є кожне завдання квантом часу та знайде пікове навантаження оперативної пам'яті з перериваннями виконання завдань.

4 ЕКСПЕРИМЕНТИ

Експерименти проводились в пакеті Excel 10. Для розв'язування запропонованих оптимізаційних моделей використовувалась надбудова OpenSolver, а для розв'язування задач за алгоритмами була написана програма. Дані для часу виконання завдань та обсягів оперативної пам'яті обиралися цілі числа датчиком випадкових чисел в заданих межах. Для розв'язування задачі по запропонованим моделям надбудовою OpenSolver необхідно на аркуш Excel ввести початкові дані. Це час виконання кожного завдання, обсяг їх оперативної пам'яті та початкові змінні задачі. Крім того необхідно ввести формули цільових функцій та обмежень задачі. Для задач (1), (5) ввести формули досить просто, враховуючи те, що Excel дозволяє копіювати формули. Але для задач (4), (13) ввести формули досить складно. Наприклад, для задачі з $n = 24$ і $m = 4$ тільки одна формула (всього 24 таких формул) має вигляд

=H5+IF(AND(C14>=D14;C14<I14);I5;0)+IF(AND(C14>=D15;C14<I15);I6;0)+IF(AND(C14>=D16;C14<I16);I7;0)+IF(AND(C14>=D17;C14<I17);I8;0)+IF(AND(C14>=D18;C14<I18); I9; 0)+IF(AND(C14>=D19; C14<I19); I10;0)+IF(AND(C14>=E14;C14<J14);J5;0)+IF(AND(C14>=E15;C14<J15);J6;0)+IF(AND(C14>=E16; C14<J16); J7;0)+IF(AND(C14>=E17;C14<J17);J8;0)+IF(AND(C14>=E18;C14<J18);J9;0)+IF(AND(C14>=E19;C14<J19);J10;0)+IF(AND(C14>=F14;C14<K14);K5;0)+IF(AND(C14>=F15;C14<K15);K6;0)+IF(AND(C14>=F16;C14<K16); K7;0)+IF(AND(C14>=F17;C14<K17);K8;0)+IF(AND(C14>=F18;C14<K18);K9;0)+IF(AND(C14>=F19;C14<K19); K10; 0)

Щоб не допустити помилок, була розроблена програма, яка вводить формули для цих задач.

Проведені експерименти показали, що алгоритм 1 майже завжди знаходить оптимальний розв'язок задачі (1). Розглянемо приклад з трьома процесорами та 15 завданнями t = (5, 9, 1, 3, 7, 6, 2, 2, 6, 9, 4, 1, 8, 8, 4). Програма OpenSolver і алгоритм 1 знайшли оптимальний розв'язок даної задачі, який показаний в табл. 1. Але ці розв'язки різні.

Таблиця 1 – Розв'язок задачі (1)

OpenSolver			Алгоритм 1		
P_1	P_2	P_3	P_1	P_2	P_3
5	9	3	5	6	4
6	1	7	9	2	1
2	6	2	1	2	8
4	1	9	3	6	8
8	8	4	7	9	4

Якщо враховувати обсяг оперативної пам'яті для даної задачі, наприклад, v = (38, 31, 30, 24, 12, 11, 12, 17, 28, 30, 31, 26, 21, 15, 3) для кожного завдання, то розв'язування задачі (5) програмою OpenSolver протягом трьох годин не дало результату. Розмірність приведеної задачі: 375 булевих змінних і 415 обмежень. Алгоритм 2 знайшов розв'язок цієї задачі (табл. 2) з піковим навантаженням на оперативну пам'ять 80.

Таблиця 2 – Розв'язок задачі алгоритмом 2

t			v		
P_1	P_2	P_3	P_1	P_2	P_3
5	6	4	38	11	31
9	2	1	31	12	26
1	2	8	30	17	21
3	6	8	24	28	15
7	9	4	12	30	3

Далі задача (5) для даного прикладу розв'язувалась методом EQR з використанням надбудови OpenSolver. За одну хвилину було знайдено розв'язок цієї задачі з піковим навантаженням на оперативну пам'ять рівним 72, при цьому завдання виконувалися з перериваннями.

DOI 10.15588/1607-3274-2025-2-18

Задачі розміром $n = 9$ і $m = 3$ розв'язувались також по моделі (13)–(16) програмою OpenSolver. Такі задачі були розв'язані за 30 сек. Але задача розміром $n = 12$ і $m = 3$ не була розв'язана програмою OpenSolver за 3 години.

Таким чином, для розв'язування задач малих розмірностей по запропонованим моделям може бути використана програма OpenSolver. Для більших розмірностей необхідно використовувати метод EQR, складовою частиною якого є програма OpenSolver.

Порівняння результатів розрахунків по моделям і алгоритмам показали, що розв'язки майже завжди співпадають. Це свідчить про високу ефективність запропонованих евристичних алгоритмів.

5 РЕЗУЛЬТАТИ

Для задачі оптимального навантаження мультипроцесорної системи та мінімізації пікового навантаження на її оперативну пам'ять розроблено декілька оптимізаційних моделей, а також два евристичних алгоритми. Для розв'язування задачі використовується потужна надбудова OpenSolver, а алгоритми реалізовані у вигляді програми. Як показали обчислювальні експерименти, розроблена програма часто знаходить оптимальні розв'язки задачі. Пропонується уточнювати результати, отримані евристичними алгоритмами, за допомогою запропонованих моделей. Безпосереднє розв'язування задачі по приведеним моделям програмою OpenSolver можливе тільки для задач малої розмірності. Для задач великої розмірності пропонується використовувати метод EQR.

6 ОБГОВОРЕННЯ

На відміну від однопроцесорних систем, мультипроцесорні системи є значно складнішими у плані їх оптимального функціонування. В сучасних операційних системах мультипроцесорних систем уже реалізовані алгоритми рівномірного завантаження процесорів завданнями. Над їх вдосконаленням розробники операційних систем ще працюють. Більш складною задачею в таких системах є оптимальний розподіл ресурсів. В першу чергу це оперативна чи кеш пам'ять. Не дивлячись на стрімке зростання її обсягів, вона залишається дефіцитним ресурсом. Оптимальне навантаження оперативної пам'яті мультипроцесорної системи є досить складною задачею. Для її розв'язування необхідні прості оптимізаційні моделі, а також ефективні евристичні алгоритми. Евристичні алгоритми, як правило, реалізуються в операційних системах. Оптимізаційні моделі необхідні для перевірки ефективності евристичних алгоритмів та проведення досліджень в галузі мультипроцесорних систем.

Складність задачі оптимального розподілу ресурсів в мультипроцесорних системах призводить до мультимодальних моделей. Такі моделі носять комбінаторний характер та містять булеві змінні. Існуючі програми для розв'язування таких задач, як правило, використовують методи розгалужень та границь. Такі методи будують бінарне дерево розв'язків підзадач і кількість таких підзадач до отримання точного розв'язку задачі може бути досить значною, навіть для задач малої розмірності. Це залежить від структури обмежень. Програма OpenSolver легко знаходить розв'язки в задачах про рюкзак з 1000 і більше булевими змінними, але стикається зі значними обчислювальними проблемами при розв'язуванні поставленої в даній роботі задачі оптимального розподілу ресурсів в мультипроцесорних системах.

Використання методу EQR значно спрощує розв'язування поставленої задачі. Це пов'язано з тим, що модель з булевими змінними перетворюється до моделі з неперервними змінними. Такі задачі програма OpenSolver досить швидко розв'язує для тисяч змінних. Крім того, метод EQR використовує квадратичну регуляризацію, яка спрощую структуру задачі.

На даний час проведено достатньо обчислювальних експериментів, які підтверджують ефективність обраної технології розв'язування складної задачі оптимального розподілу ресурсів в мультипроцесорних системах.

ВИСНОВКИ

В роботі розглянута актуальна задача оптимального розподілу ресурсів в мультипроцесорних системах. Зокрема, задача оптимального навантаження таких систем з мінімізацією пікового навантаження оперативної пам'яті. Задача розглядається як без переривань у виконанні завдань, так і з перериваннями. Для даної задачі побудовано декілька оптимізаційних моделей та евристичні алгоритми.

Дана задача може бути узагальнена для взаємопов'язаних завдань. Для таких завдань моделі необхідно доповнити мережевими графіками, які визначають взаємозалежність завдань. Для цього необхідно доповнити моделі задачі лінійними обмеженнями, які визначають порядок виконання завдань.

Наукова новизна роботи полягає в тому, що двоетапна постановка задачі з мінімізацією пікового навантаження оперативної пам'яті є новою. Новими є також запропоновані оптимізаційні моделі, крім задачі (1). Ці моделі містять мінімальну кількість змінних та обмежень у порівнянні з існуючими моделями для подібного класу задач. Евристичні алгоритми використовують, запропонований автором, фінальний принцип, який значно покращує розв'язки задачі. Для приведених моделей вперше використано розроблений автором метод EQR.

Практична цінність роботи полягає в тому, що її результати можуть бути використані для вдосконалення сучасних мультипроцесорних операційних систем. Отримані результати можуть бути використані також для подальших досліджень в мультипроцесорних системах.

Майбутні напрями дослідження та розробки будуть включати наступні узагальнення даної задачі. Зокрема, розгляд гетерогенних мультипроцесорних

DOI 10.15588/1607-3274-2025-2-18

систем, врахування інших обмежених ресурсів та зниження електроспоживання в таких системах.

ПОДЯКА

Робота виконана за кошти державного бюджету НДР ДНУ ім. О. Гончара «Інтелектуальна обробка даних в комп'ютерних інформаційних системах» (державний регістраційний номер 0122U001397), яка виконувалась з 01.01.2022 по 31.12.2024 роки.

ЛІТЕРАТУРА

1. Kosolap A. A new method for global optimization / A. Kosolap // ESAIM: Proceedings and surveys. – 2021. – Vol. 71. – P. 121–130. DOI 10.1051/proc/202171121.
2. Ramamoorthy C. V. Optimal Scheduling Strategies in a Multiprocessor System/ C. V. Ramamoorthy, K. M. Chandy and M. J. Gonzalez // IEEE Transactions on Computers. – 1972. – Vol. C-21, No. 2. – P. 137–146. DOI: 10.1109/TC. 1972.5008918.
3. Coffman E. G. An Application of Bin-Packing to Multiprocessor Scheduling/ E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson //SIAM Journal of Computing.– 1978.– Vol. 7, Iss. 1. – P. 1–17. DOI: 10.1137/0207001.
4. Garey M. R. Complexity Results for Multiprocessor Scheduling under Resource Constraints/ M. R. Garey and D. S. Johnson // SIAM Journal on Computing. – 1975. – Vol. 4, Iss. 4. – P. 397–411. DOI: 10.1137/0204035.
5. Dilawar N. A Review of Power Efficient Load Balancing Algorithms for Multicore Systems / N. Dilawar, M. Zakarya and I. Ur Rahman// World Applied Sciences Journal. – 2013. – Vol. 27(9). – P. 1175–1182. DOI: 10.5829/idosi. wasj. 2013.27.09.1627.
6. Scheduling in Distributed Computing Systems Analysis, Design & Models / [D. P. Vidyarthi, D. K. Sarker, A. K. Tripathi, L. T. Yang]. – Springer Science+Business Media, LLC, 2009. – 300 p.
7. Khawatreh S. A. An Efficient Algorithm for Load Balancing in Multiprocessor Systems / S. A. Khawatreh // International Journal of Advanced Computer Science and Applications. – 2018. – Vol. 9, No. 3. – P. 160–164. DOI: 10.14569/ IJACSA.2018.090324.
8. Kermia O. Load Balancing and Efficient Memory Usage for Homogeneous Distributed Real-Time Embedded Systems / O. Kermia and Y. Sorel // International Conference on Parallel Processing – Workshops, Portland, OR, USA. – 2008. – P. 39–46. DOI: 10.1109/ICPP-W.2008.20.
9. Teixeira R. B. Shared resources in multiprocessor real-time systems scheduled by RUN / R. B. Teixeira, G. Lima // Real-Time Syst. – 2022. – No. 58. – P. 153–188. DOI 10.1007/ s11241-021-09374-3.
10. Walter R. A characterization of optimal multiprocessor schedules and new dominance rules / R. Walter, A. Lawrinenko // Journal of Combinatorial Optimization. – 2020. – Vol. 40. – P. 876–900. DOI 1007/s10878-020-00634-9.
11. Mehrabi A. An Adaptive Genetic Algorithm for Multiprocessor Task Assignment Problem with Limited Memory / A. Mehrabi, S. Mehrabi, Ali D. Mehrabi [Electronic resource]. – Access mode: Access mode: https://www.oalib.com/research/2169214
12. Revilla-Duarte U. Proactive Load Balancing to Reduce Unnecessary Thread Migrations on Chip Multi-Processor (CMP) System / [U. Revilla-Duarte, M. A. Ramırez-Salinas, L. A. Villa-Vargas, A. Tchernykh] // Computación y Sistemas. – 2024. – Vol. 28, No. 2. – P. 623–645. DOI: 10.13053/CyS-28-2-4403.
13. Kellerer H. Knapsack Problems / H. Kellerer, U. Pferschy, D. Pisinger. – Springer-Verlag Berlin Heidelberg, 2004. – 554 p. DOI l0.I007/978-3-540-24777-7.
14. Hill R. R. Generation Methods for Multidimensional Knapsack Problems and their Implications / R. R. Hill, C. S. Hiremath // Systemics, Cybernetics and Informatics. – 2007. – Vol. 5, № 5. – P. 59–64. DOI: 10.54808/JSCI.21.04.42

Стаття надійшла до редакції 14.01.2025.
Після доробки 18.04.2025.

UDC 004.2

OPTIMAL ALLOCATION OF LIMITED RESOURCES IN MULTIPROCESSOR SYSTEMS

Kosolap A. I. – Doctor of Physical and Mathematical Sciences, Professor, Professor of the Department of Computer Science and Information Technologies, Oles Honchar Dnipro National University, Dnipro, Ukraine.

ABSTRACT

Context. The paper considers multiprocessor systems consisting of many processors with a common RAM. The efficiency of such systems depends on the operating system. It must ensure a uniform loading of processors with tasks, in which the peak load on RAM will be minimal. This is a rather complex problem. In this paper, it is solved by building optimization models and developing effective heuristic algorithms. This problem is solved in two stages. The first stage is the optimal loading of processors with tasks, and the second is the minimization of the peak load on RAM. Several optimization models of this problem have been built, for the solution of which the exact quadratic regularization method is effective. Effective heuristic algorithms have also been developed. Comparative computational experiments have been conducted, which confirm the effectiveness of the proposed technology for solving this problem.

Objective. Development of mathematical optimization models, methods, and algorithms for optimal resource allocation in multiprocessor systems.

Method. A two-stage solution to this problem is effective. Several optimization models containing Boolean variables are proposed. Such models are quite complex for finding optimal solutions. To solve them, it is proposed to use the method of exact quadratic regularization. This optimization method is used for the first time for this class of problems, so it required the development of appropriate algorithmic support. Heuristic algorithms are usually implemented in operating systems. Therefore, effective heuristic algorithms are proposed that use the final principle, which significantly improves the solution of the problem.

DOI 10.15588/1607-3274-2025-2-18

Results. New optimization models for the allocation of limited resources in multiprocessor systems have been constructed. Effective heuristic algorithms have been developed, which are implemented software-wise using VBA in the Excel package. Software for entering initial data for optimization models has also been developed, which simplifies their solution. The results of computational experiments are presented.

Conclusions. A new effective technology for optimal resource allocation in multiprocessor systems has been developed. Heuristic algorithms have been developed and implemented in software. Computational experiments have been conducted to confirm the effectiveness of the proposed technology for solving the problem.

KEYWORDS: multiprocessor systems, optimization models, problems with Boolean variables, heuristic algorithms, exact quadratic regularization method, final principle.

REFERENCES

1. Kosolap A. A new method for global optimization, *ESAIM: Proceedings and surveys*, 2021, Vol. 71, pp. 121–130. DOI 10.1051/proc/202171121.
2. Ramamoorthy C. V., Chandy K. M. and Gonzalez M. J. Optimal Scheduling Strategies in a Multiprocessor System, *IEEE Transactions on Computers*, 1972, Vol. C-21, No. 2, pp. 137–146. DOI: 10.1109/TC. 1972.5008918.
3. Coffman E. G., Garey Jr., M. R., and Johnson D. S. An Application of Bin-Packing to Multiprocessor Scheduling, *SIAM Journal of Computing*, 1978, Vol. 7, Iss. 1, pp. 1–17. DOI: 10.1137/0207001.
4. Garey M. R. and Johnson D. S. Complexity Results for Multiprocessor Scheduling under Resource Constraints, *SIAM Journal on Computing*, 1975, Vol. 4, Iss. 4, pp. 397–411. DOI: 10.1137/0204035.
5. Dilawar N., Zakarya M. and Rahman I. Ur A Review of Power Efficient Load Balancing Algorithms for Multicore Systems, *World Applied Sciences Journal*, 2013, Vol. 27(9), pp. 1175–1182. DOI: 10.5829/idosi. wasj. 2013.27.09.1627.
6. Vidyarthi D. P., Sarker D. K., Tripathi A. K., Yang L. T. Scheduling in Distributed Computing Systems Analysis, Design & Models. Springer Science+Business Media, LLC, 2009, 300 p.
7. Khawatreh S. A. An Efficient Algorithm for Load Balancing in Multiprocessor Systems, *International Journal of Advanced Computer Science and Applications*, 2018, Vol. 9, No. 3, pp. 160–164. DOI: 10.14569/ IJACSA.2018.090324.
8. Kermia O. and Sorel Y. Load Balancing and Efficient Memory Usage for Homogeneous Distributed Real-Time Embedded Systems, *International Conference on Parallel Processing*. Workshops, Portland, OR, USA, 2008, pp. 39–46. DOI: 10.1109/ICPP-W.2008.20.
9. Teixeira R. B., Lima G. Shared resources in multiprocessor real-time systems scheduled by RUN, *Real-Time Syst.*, 2022, No. 58, pp. 153–188. DOI 10.1007/ s11241-021-09374-3.
10. Walter R., Lawrinenko A. A characterization of optimal multiprocessor schedules and new dominance rules, *Journal of Combinatorial Optimization,* 2020, Vol 40, pp. 876–900. DOI 1007/s10878-020-00634-9.
11. Mehrabi A., Mehrabi S., Mehrabi Ali D. An Adaptive Genetic Algorithm for Multiprocessor Task Assignment Problem with Limited Memory. [Electronic resource]. Access mode: https://www.oalib.com/research/2169214
12. Revilla-Duarte U., Ramırez-Salinas M. A., Villa-Vargas L. A., Tchernykh A. Proactive Load Balancing to Reduce Unnecessary Thread Migrations on Chip Multi-Processor (CMP) System, *Computación y Sistemas,* 2024, Vol. 28, No. 2, pp. 623–645. DOI: 10.13053/CyS-28-2-4403.
13. Kellerer H., Pferschy U., Pisinger D. Knapsack Problems. Springer-Verlag Berlin Heidelberg, 2004, 554 p. DOI l0.I007/978-3-540-24777-7.
14. Hill R. R., Hiremath C. S. Generation Methods for Multidimensional Knapsack Problems and their Implications, *Systemics, Cybernetics and Informatics*, 2007, Vol. 5, № 5, pp. 59–64. DOI: 10.54808/JSCI.21.04.42.

DOI 10.15588/1607-3274-2025-2-18